

8-2020

## ACTIVE MAGNETIC RADIATION SHIELDING FOR LONG-DURATION HUMAN SPACEFLIGHT

Kristine Ferrone

Follow this and additional works at: [https://digitalcommons.library.tmc.edu/utgsbs\\_dissertations](https://digitalcommons.library.tmc.edu/utgsbs_dissertations)



Part of the [Medicine and Health Sciences Commons](#), [Other Astrophysics and Astronomy Commons](#),  
and the [Other Physics Commons](#)

---

### Recommended Citation

Ferrone, Kristine, "ACTIVE MAGNETIC RADIATION SHIELDING FOR LONG-DURATION HUMAN SPACEFLIGHT" (2020). *The University of Texas MD Anderson Cancer Center UTHealth Graduate School of Biomedical Sciences Dissertations and Theses (Open Access)*. 1019.  
[https://digitalcommons.library.tmc.edu/utgsbs\\_dissertations/1019](https://digitalcommons.library.tmc.edu/utgsbs_dissertations/1019)

This Dissertation (PhD) is brought to you for free and open access by the The University of Texas MD Anderson Cancer Center UTHealth Graduate School of Biomedical Sciences at DigitalCommons@TMC. It has been accepted for inclusion in The University of Texas MD Anderson Cancer Center UTHealth Graduate School of Biomedical Sciences Dissertations and Theses (Open Access) by an authorized administrator of DigitalCommons@TMC. For more information, please contact [digitalcommons@library.tmc.edu](mailto:digitalcommons@library.tmc.edu).

**ACTIVE MAGNETIC RADIATION SHIELDING FOR  
LONG-DURATION HUMAN SPACEFLIGHT**

by


*Kristine L. Ferrone, M.S., M.S., MBA*

APPROVED:



---

Stephen F. Kry, Ph.D.  
Advisory Professor



---

Charles E. Willis, Ph.D.



---

Jingfei Ma, Ph.D.



---

Fada Guan, Ph.D.



---

Leif E. Peterson, Ph.D.

---

APPROVED:

---

Dean, The University of Texas  
MD Anderson Cancer Center UTHealth Graduate School of Biomedical Sciences

**ACTIVE MAGNETIC RADIATION SHIELDING FOR  
LONG-DURATION HUMAN SPACEFLIGHT**

A

DISSERTATION

Presented to the Faculty of

The University of Texas

MD Anderson Cancer Center UTHealth

Graduate School of Biomedical Sciences

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

by

Kristine L. Ferrone, M.S., M.S., MBA  
Houston, Texas

August 2020

# ACTIVE MAGNETIC RADIATION SHIELDING FOR LONG-DURATION HUMAN SPACEFLIGHT

Kristine L. Ferrone, M.S., M.S., MBA

Advisory Professor: Stephen Kry, Ph.D.

Exploration of interplanetary space presents dramatic hazards to human survival. Space radiation hazards outside the protection of the Earth's magnetosphere can produce both acute and chronic health risks and thus become limiting factors for NASA's planned mission to Mars by the 2030s. Radiation exposure on a Mars mission is delivered primarily by high energy ions from galactic cosmic rays and moderate energy protons from solar particle events. The chronic radiation dose due to galactic cosmic rays on a typical Mars mission is on the order of 1 Sv, and additional acute radiation dose from solar flares can reach over 4 Sv, which is a potentially lethal dose. Hence radiation protection is a critical concern on these types of missions.

Various methods of radiation shielding have been proposed, from simple passive shielding via materials such as water, polyethylene, or aluminum, to active shielding systems comprised of electromagnetic fields. The concept of active magnetic shielding is to use high-temperature superconducting coils to induce very high magnetic fields around the spacecraft. The induced magnetic field will deflect incoming charged particles (solar particles and galactic cosmic rays), thereby reducing the particle fluence rate and radiation dose to astronauts behind the shield.

This project developed a model for determining the effectiveness of active magnetic shielding in reducing radiation dose to astronauts on an interplanetary mission. This research includes Monte Carlo simulations to determine the effectiveness of magnetic shielding in decreasing effective dose to astronauts in a variety of mission scenarios. Dozens of permutations of mission type, mission duration, solar cycle, shielding configuration, magnetic field, crew sex, crew age, and phantom type were simulated in GEANT4 to conduct a sensitivity analysis on the effect of varying each parameter on total crew effective dose for the mission.

Results indicate that magnetic shielding can reduce effective dose to astronauts on an interplanetary mission to within NASA's current limits, given a magnetic field of 7 T and/or advanced astronaut age. The detailed results serve to inform the human spaceflight community on the utility of active magnetic shielding as compared to passive or no shielding, based upon an end-to-end system model and comparison of several active magnetic shielding strategies.

## Table of Contents

Approval Page .....	i
Title Page .....	ii
Abstract .....	iii
Table of Contents .....	iv
List of Illustrations .....	vii
List of Tables .....	xiv
Abbreviations .....	xix
Specific Aims .....	1
1 Introduction .....	3
1.1 Motivation .....	3
1.2 History of Magnetic Shielding .....	8
1.2.1 The Pre-NASA Era .....	8
1.2.2 The 1960s .....	8
1.2.3 The 1970s .....	10
1.2.4 The 1980s .....	10
1.2.5 The 1990s to Early 2000s .....	11
1.2.6 The Mid to Late 2000s .....	12
1.2.7 The 2010s and Beyond .....	15
1.2.8 Discussion .....	16
1.3 Project Rationale .....	19
2 Research Design .....	18
2.1 Justification and Feasibility .....	22
2.2 Trade Tree Development .....	23
2.2.1 Mission Profile .....	23
2.2.2 Solar Cycle Position .....	27

2.2.3	Shielding Configurations .....	28
2.2.4	Magnetic Field.....	31
2.2.4.1	Safety of Human Exposure to Static Magnetic Fields.....	31
2.2.4.2	Magnetic Fringe Field Limits .....	33
2.2.5	Phantom Type.....	33
2.2.6	Astronaut Sex (Male/Female).....	35
2.2.7	Astronaut Age .....	36
2.3	Simulations.....	37
2.3.1	Space Environment.....	38
2.3.1.1	Solar Wind.....	39
2.3.1.2	Solar Particle Events (SPEs) .....	39
2.3.1.3	Galactic Cosmic Rays (GCRs).....	40
2.3.2	Spacecraft Architecture .....	48
2.3.2.1	Habitat Module .....	48
2.3.2.2	Crew Return Capsule .....	50
2.3.3	Shielding Configurations .....	52
2.3.3.1	Minimal Shielding .....	53
2.3.3.2	Water Shielding .....	54
2.3.3.3	Toroidal Magnetic Shielding.....	57
2.3.3.4	Solenoidal Magnetic Shielding.....	59
2.3.3.5	Race Track Magnetic Shielding .....	65
2.3.4	Dosimetry and Risk Analysis.....	68
2.3.4.1	Human Phantom Selection .....	68
2.3.4.2	Dosimetry Method .....	70
2.3.4.3	Evaluation of MIRD Phantom Improvements .....	81
2.3.4.4	Dosimetry in GEANT4.....	92

2.3.4.5	Risk Analysis .....	93
2.3.5	Simulation Methodology .....	94
3	Results and Discussion .....	97
3.1	Control Cases.....	97
3.1.1	Minimal Shielding .....	98
3.1.2	Water Shielding .....	101
3.2	Magnetic Shielding Cases .....	103
3.2.1	Zero Magnetic Field .....	103
3.2.2	Toroidal Magnetic Shielding .....	104
3.2.3	Solenoidal Magnetic Shielding .....	107
3.2.4	Race Track Magnetic Shielding .....	111
3.3	Shielding Effectiveness by Scenario and Dosimetry Approach .....	114
3.3.1	Effective Dose vs. Magnetic Field by Scenario .....	115
3.3.2	Absorbed Dose vs. Effective Dose Comparison .....	125
3.4	Relative Shielding Effectiveness.....	133
3.5	Volume, Mass, and Shielding Thickness Comparison .....	133
3.5.1	Total Volume of Magnetized Space.....	135
3.5.2	Total Structural Passive Shielding Mass.....	136
3.5.3	Effective Passive Shielding Thickness .....	139
3.5.4	Interpretation of Passive and Active Shielding Effects .....	141
3.6	Validation in Magnetic Shielding Literature .....	141
3.7	Results Summary .....	151
3.8	Relative Contribution by Incident Radiation Type.....	158
3.9	Organ Dose Equivalent Analysis .....	162
3.10	Sensitivity Analysis .....	176
3.10.1	SPE Effective Dose .....	177

3.10.2 GCR Effective Dose .....	177
3.10.3 Total Effective Dose .....	178
3.11 Cancer Risk and Safe Days in Space Analysis .....	182
3.12 Validation of Results.....	211
3.13 Engineering Challenges .....	213
4 Conclusion .....	216
4.1 Project Summary.....	216
4.2 Response to Hypothesis .....	217
4.3 Significance.....	220
4.4 Future Work .....	221
Appendix A1: GEANT4 GCR Source Macro (Solar Max, Z=1).....	221
Appendix A2: GEANT4 GCR Source Macro (Solar Max, Z=2).....	222
Appendix A3: GEANT4 GCR Source Macro (Solar Max, Z=3 to Z=26).....	223
Appendix A4: GEANT4 GCR Source Macro (Solar Min, Z=1).....	229
Appendix A5: GEANT4 GCR Source Macro (Solar Min, Z=2).....	229
Appendix A6: GEANT4 GCR Source Macro (Solar Min, Z=3 to Z=26).....	230
Appendix A7: GEANT4 Detector Construction Code .....	236
Appendix A8: MATLAB r2018a Script for Solenoidal Magnetic Field 1.5 T.....	427
Appendix A9: Organ Dose Calculations in Excel.....	431
Appendix A10: Summary Data Sheets for All Scenarios .....	432
Appendix A11: Beam Experiment Plan .....	489
Bibliography.....	503
Vita .....	537

## List of Illustrations

Figure 1: NASA Human Spaceflight Risk Summary (Francisco 2015).....	3
Figure 2: NASA Human Research Roadmap Space Radiation Risks.....	6
Figure 3: Options to Mitigate Space Radiation Risk on Interplanetary Missions .....	7
Figure 4: 50+ Years of Magnetic Shielding Studies.....	17
Figure 5: Timeline of Magnetic Shielding Publications with Contextual Information.....	18
Figure 6: Trade Tree.....	24
Figure 7: Summary of NASA’s Journey to Mars (Simonsen 2017) .....	25
Figure 8: Solar Cycle Sunspot Number Progression ( <a href="http://www.swpc.noaa.gov/">http://www.swpc.noaa.gov/</a> ).....	28
Figure 9: Schema for Monte Carlo Simulations in GEANT4.....	37
Figure 10: Space Radiation Spectrum (Cucinotta et al. 2013).....	38
Figure 11: Simulated Solar Flare Spectra .....	41
Figure 12: Spectra Near Solar Min/Max for Primary GCRs (Cucinotta et al. 2013).....	43
Figure 13: Spectra Near Solar Min/Max for Primary GCR Isotopes (Cucinotta et al. 2013) ..	43
Figure 14: GCR Data Collected by Spacecraft (O’Neill et al. 2015).....	44
Figure 15: Simulated GCR Spectra at Solar Max.....	47
Figure 16: Simulated GCR Spectra at Solar Min.....	47
Figure 17: GEANT4 Actual vs. Input Spectra.....	48
Figure 18: ISS <i>Destiny</i> Module .....	49
Figure 19: Description of ISS Protective Layers.....	49
Figure 20: Orion Crew Module .....	50
Figure 21: Simulated Spacecraft Architecture in GEANT4.....	52
Figure 22: Cutaway Model of ISS <i>Columbus</i> Module .....	53
Figure 23: Simulated Minimal Shielding in GEANT4.....	54
Figure 24: Dose at 5 cm Depth in Tissue for Various Materials (Adams et al. 2005) .....	55
Figure 25: Prototype Deployable Water Wall Shielding (Semones 2015).....	56

Figure 26: Simulated Water Shielding Configuration in GEANT4 .....	56
Figure 27: Simulated Toroidal Magnetic Field (created with Magnetism 3D) .....	57
Figure 28: Simulated Toroid Magnetic Shielding Configuration in GEANT4 .....	58
Figure 29: NIAC 6+1 Solenoid Magnetic Shielding Configuration (Westover et al. 2014).....	59
Figure 30: Simulated NIAC 6+1 Solenoid Magnetic Shielding Configuration in GEANT4.....	61
Figure 31: Magnetic Field Map (x-z plane) from MATLAB .....	65
Figure 32: ESA SR2S Race Track Shielding (Vuolo et al. 2016).....	66
Figure 33: Simulated Modified ESA Race Track Shielding Configuration in GEANT4.....	67
Figure 34: MIRD Phantom (Segars & Tsui 2009) .....	68
Figure 35: Male and Female Phantoms in GEANT4 .....	69
Figure 36: Modified MIRD Phantom with Active Bone Marrow Volumes in GEANT4 .....	79
Figure 37: MIRD Phantom with Eye Lenses in GEANT4.....	80
Figure 38: MIRD Phantom Under PA Diagnostic X-Ray Irradiation in GEANT4 .....	82
Figure 39: Simulated Diagnostic X-ray Spectra for Medical Example .....	83
Figure 40: ICRP Reference Fields (ICRP 2012).....	84
Figure 41: MIRD Phantom Under AP 662 keV Gamma Irradiation in GEANT4 .....	84
Figure 42: MIRD Phantom Under Isotropic 662 keV Gamma Irradiation in GEANT4 .....	84
Figure 43: MIRD Phantom Under Isotropic 1 GeV GCR Irradiation in GEANT4.....	85
Figure 44: Bone Marrow Dose Equivalent vs. kVp for Medical (PA) Example .....	86
Figure 45: Monoenergetic Bone Marrow Dose Equivalent for Medical (PA) Example.....	87
Figure 46: Monoenergetic and Spectral Bone Marrow Dose Equivalent for Medical Example	87
Figure 47: Bone Marrow Dose Equivalent for Radiation Protection (AP) Example .....	88
Figure 48: Bone Marrow Dose Equivalent for Radiation Protection (iso) Example .....	88
Figure 49: Bone Marrow Dose Equivalent for Space Radiation (iso) Example .....	89
Figure 50: Effect of Buildup on Bone Marrow Dose (Hendee & Ritenour 2002).....	90
Figure 51: Medical Example with Additional Test Energies .....	92

Figure 52: Minimal Shielding in SPE Environment .....	99
Figure 53: Minimal Shielding in GCR Environment .....	100
Figure 54: Total Mission (700 d) Dose for Minimal Shielding Cases.....	100
Figure 55: Water Shielding in SPE Environment.....	102
Figure 56: Water Shielding in GCR Environment .....	102
Figure 57: Total Mission (700 d) Dose with 20 cm Water Shielding.....	103
Figure 58: 0 T Toroidal Magnetic Shielding in SPE Environment .....	105
Figure 59: 0 T Toroidal Magnetic Shielding in GCR Environment.....	105
Figure 60: 1.5 T Toroidal Magnetic Shielding in GCR Environment.....	106
Figure 61: 3 T Toroidal Magnetic Shielding in GCR Environment.....	106
Figure 62: 7 T Toroidal Magnetic Shielding in GCR Environment.....	107
Figure 63: 0 T Solenoidal Magnetic Shielding in SPE Environment.....	108
Figure 64: 0 T Solenoidal Magnetic Shielding in GCR Environment.....	109
Figure 65: 1.5 T Solenoidal Magnetic Shielding in GCR Environment.....	109
Figure 66: 3 T Solenoidal Magnetic Shielding in GCR Environment.....	110
Figure 67: 7 T Solenoidal Magnetic Shielding in GCR Environment.....	110
Figure 68: 0 T Race Track Magnetic Shielding in SPE Environment.....	112
Figure 69: 0 T Race Track Magnetic Shielding in GCR Environment .....	112
Figure 70: 1.5 T Race Track Magnetic Shielding in GCR Environment .....	113
Figure 71: 3 T Race Track Magnetic Shielding in GCR Environment .....	113
Figure 72: 7 T Race Track Magnetic Shielding in GCR Environment .....	114
Figure 73: Effective Dose Rate vs. Magnetic Field (Female, Solar Max).....	115
Figure 74: Effective Dose Rate vs. Magnetic Field (Female, Solar Min).....	116
Figure 75: Effective Dose Rate vs. Magnetic Field (Male, Solar Max).....	117
Figure 76: Effective Dose Rate vs. Magnetic Field (Male, Solar Min).....	118
Figure 77: Effective Dose vs. Magnetic Field by Shielding Configuration .....	125

Figure 78: Absorbed Dose Rate vs. Magnetic Field (Female, Solar Max) .....	126
Figure 79: Absorbed Dose Rate vs. Magnetic Field (Female, Solar Min) .....	127
Figure 80: Absorbed Dose Rate vs. Magnetic Field (Male, Solar Max) .....	128
Figure 81: Absorbed Dose Rate vs. Magnetic Field (Male, Solar Min) .....	129
Figure 82: Dose Rate vs. Magnetic Field by Shielding Configuration .....	132
Figure 83: Effective Dose Rate vs. Magnetic Field by Shielding .....	134
Figure 84: Comparison of Total Volume, Passive Shielding Mass, Shielding Thickness ....	140
Figure 85: Total GCR Effective Dose vs. Magnetic Field (Toroid) .....	146
Figure 86: Annual Active Bone Marrow Dose Equivalent vs. Magnetic Field (Solenoid).....	148
Figure 87: Total GCR Effective Dose vs. Magnetic Field (Race Track) .....	150
Figure 88: Total Mission (700 d) Effective Dose (Max, Female) .....	153
Figure 89: Total Mission (700 d) Effective Dose (Max, Male) .....	154
Figure 90: Total Mission (700 d) Effective Dose (Min, Female) .....	155
Figure 91: Total Mission (700 d) Effective Dose (Min, Male) .....	156
Figure 92: Total Mission (700 d) Effective Dose Contributions (Max, Female) .....	158
Figure 93: Total Mission (700 d) Effective Dose Normalized Contributions (Max, Female). 158	
Figure 94: Total Mission (700 d) Effective Dose Contributions (Max, Male) .....	159
Figure 95: Total Mission (700 d) Effective Dose Normalized Contributions (Max, Male).....	159
Figure 96: Total Mission (700 d) Effective Dose Contributions (Min, Female) .....	160
Figure 97: Total Mission (700 d) Effective Dose Normalized Contributions (Min, Female) .	160
Figure 98: Total Mission (700 d) Effective Dose Contributions (Min, Male) .....	161
Figure 99: Total Mission (700 d) Effective Dose Normalized Contributions (Min, Male).....	161
Figure 100: Organ Dose Equivalent, Normalized (Max, Female) .....	164
Figure 101: Organ Dose Equivalent, Average Inverse Rank (Max, Female) .....	164
Figure 102: Organ Dose Equivalent, Frequency of Inverse Rank (Max, Female) .....	165
Figure 103: Organ Dose Equivalent, Normalized (Max, Male).....	165

Figure 104: Organ Dose Equivalent, Average Inverse Rank (Max, Male).....	166
Figure 105: Organ Dose Equivalents, Frequency of Inverse Rank (Max, Male) .....	166
Figure 106: Organ Dose Equivalent, Normalized (Min, Female) .....	167
Figure 107: Organ Dose Equivalent, Average Inverse Rank (Min, Female) .....	167
Figure 108: Organ Dose Equivalent, Frequency of Inverse Rank (Min, Female).....	168
Figure 109: Organ Dose Equivalent, Normalized (Min, Male).....	168
Figure 110: Organ Dose Equivalent, Average Rank (Min, Male) .....	169
Figure 111: Organ Dose Equivalents, Frequency of Rank (Min, Male).....	169
Figure 112: Organ Dose Equivalent, Normalized (Average, Female).....	170
Figure 113: Organ Dose Equivalent, Average Rank (Average, Female).....	170
Figure 114: Organ Dose Equivalents, Frequency of Rank (Average, Female) .....	171
Figure 115: Organ Dose Equivalent, Normalized (Average, Male) .....	171
Figure 116: Organ Dose Equivalent, Average Rank (Average, Male).....	172
Figure 117: Organ Dose Equivalents, Frequency of Rank (Average, Male) .....	172
Figure 118: Matrix of Organs at Risk (Female).....	175
Figure 119: Matrix of Organs at Risk (Male).....	175
Figure 120: Sensitivity Analysis for Total Mission (700 d) SPE Effective Dose.....	179
Figure 121: Radar Chart for Total Mission (700 d) SPE Effective Dose .....	179
Figure 122: Sensitivity Analysis for Total Mission (700 d) GCR Effective Dose .....	180
Figure 123: Radar Chart for Total Mission (700 d) GCR Effective Dose.....	180
Figure 124: Sensitivity Analysis for Total Mission (700 d) Total Effective Dose .....	181
Figure 125: Radar Chart for Total Mission (700 d) Total Effective Dose.....	181
Figure 126: REID vs. Age (Female, 700 d, Toroid) .....	183
Figure 127: REID vs. Age (Female, 700 d, Solenoid).....	184
Figure 128: REID vs. Age (Female, 700 d, Race Track) .....	185
Figure 129: REID vs. Age (Female, 700 d, Average Magnetic) .....	186

Figure 130: REID vs. Age (Female, 500 d, Toroid) .....	187
Figure 131: REID vs. Age (Female, 500 d, Solenoid).....	188
Figure 132: REID vs. Age (Female, 500 d, Race Track) .....	189
Figure 133: REID vs. Age (Female, 500 d, Average Magnetic) .....	190
Figure 134: REID vs. Age (Male, 700 d, Toroid).....	191
Figure 135: REID vs. Age (Male, 700 d, Solenoid).....	192
Figure 136: REID vs. Age (Male, 700 d, Race Track).....	193
Figure 137: REID vs. Age (Male, 700 d, Average Magnetic) .....	194
Figure 138: REID vs. Age (Male, 500 d, Toroid).....	195
Figure 139: REID vs. Age (Male, 500 d, Solenoid).....	196
Figure 140: REID vs. Age (Male, 500 d, Race Track).....	197
Figure 141: REID vs. Age (Male, 500 d, Average Magnetic) .....	198
Figure 142: Safe Days in Space vs. Age (Female, Toroid).....	199
Figure 143: Safe Days in Space vs. Age (Female, Solenoid).....	200
Figure 144: Safe Days in Space vs. Age (Female, Race Track).....	201
Figure 145: Safe Days in Space vs. Age (Female, Average Magnetic) .....	202
Figure 146: Safe Days in Space vs. Age (Male, Toroid).....	203
Figure 147: Safe Days in Space vs. Age (Male, Solenoid) .....	204
Figure 148: Safe Days in Space vs. Age (Male, Race Track).....	205
Figure 149: Safe Days in Space vs. Age (Male, Average Magnetic) .....	206
Figure 150: GMW 3472-70 Electromagnet; courtesy of GMW.....	492
Figure 151: GMW 3472-70 Electromagnet Set Up for Horizontal Irradiation .....	493
Figure 152: GMW 3472-70 Electromagnet Specifications .....	493
Figure 153: BV Thermal Systems Chiller Specifications.....	494
Figure 154: GMW 3472-70 Magnet Excitation Curve vs. Pole Gap (GMW 2009).....	494
Figure 155: GMW 3472-70 Magnet Excitation Curve vs. Current (GMW 2009).....	494

Figure 156: Gaussmeter GM2.....	495
Figure 157: GMW Magnet Field Maps (courtesy of A. Rubinstein).....	495
Figure 158: Matrixx <sup>TM</sup> Evolution (courtesy of IBA Dosimetry) .....	496
Figure 159: Zebra Detector (courtesy of IBA Dosimetry).....	496
Figure 160: Schematic of Deflection Angle Using Magnetic Shielding.....	502
Figure 161: Experiment Setup Modeled in GEANT4.....	502

## List of Tables

Table 1: BEIR VII and NASA/NCRP Effective Dose Limits.....	37
Table 2: Tissue Weighting Factors (ICRP 2007) .....	72
Table 3: Tissue Weighting Factors (modified for female MIRD phantom).....	73
Table 4: Tissue Weighting Factors (modified for male MIRD phantom).....	74
Table 5: Active Bone Marrow Distribution for 40-Year-Old Adult (Cristy 1981).....	75
Table 6: Active Bone Marrow Distribution for 40-Year-Old Adult (modified) .....	76
Table 7: Active Marrow Elemental Composition (Dant et al. 2013).....	77
Table 8: Active Bone Marrow Mass/Volume Calculations .....	77
Table 9: Equivalent Energy ( $h\bar{\nu}$ ) for Clinical X-ray Spectra .....	83
Table 10: Scenario Definitions for Bone Marrow Dose Estimation .....	86
Table 11: Example of a Summary Sheet Recorded for Each Scenario.....	96
Table 12: Effective Dose vs. Magnetic Field Exponential Fit Intercept, Exponent Summary	119
Table 13: Projected Magnetic Field Strengths to Reduce Effective Dose by 50% .....	120
Table 14: Average Difference in Effective Dose Rate by Magnetic Shielding .....	121
Table 15: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration ...	122
Table 16: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration ...	122
Table 17: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration ...	123
Table 18: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration ...	124
Table 19: Absorbed Dose vs. Magnetic Field Exponential Fit Intercept, Exponent Summary	130
Table 20: Projected Magnetic Field Strengths to Reduce Dose Rate by 50% vs. 0 T.....	131
Table 21: Average Difference in Absorbed Dose Rate by Magnetic Shielding Type.....	144
Table 22: Summary of Key Parameters from This and Prior Magnetic Shielding Studies...	144
Table 23: Summary of Results Comparison with Literature.....	151

Table 24: Relative Effectiveness of Shielding Configurations.....	156
Table 25: Organ Dose Ranks and Tissue Weighting Factors.....	174
Table 26: Sensitivity Analysis Calculations .....	178
Table 27: 3 <sup>rd</sup> Order Polynomial Coefficients for REID vs. Astronaut Age.....	207
Table 28: 4th Order Polynomial Coefficients for SDS vs. Astronaut Age.....	209
Table 29: Organ Dose Calculations in Excel.....	431
Table 30: Max / Aluminum / Female Summary Data Sheet.....	432
Table 31: Min / Aluminum / Female Summary Data Sheet.....	433
Table 32: Max / Aluminum / Male Summary Data Sheet.....	434
Table 33: Min / Aluminum / Male Summary Data Sheet.....	435
Table 34: Max / Aluminum / Water Summary Data Sheet .....	436
Table 35: Min / Aluminum / Water Summary Data Sheet .....	436
Table 36: Max / Water / Female Summary Data Sheet .....	437
Table 37: Min / Water / Female Summary Data Sheet .....	438
Table 38: Max / Water / Male Summary Data Sheet .....	439
Table 39: Min / Water / Male Summary Data Sheet .....	440
Table 40: Max / Toroid 0 T / Female Summary Data Sheet.....	441
Table 41: Min / Toroid 0 T / Female Summary Data Sheet .....	442
Table 42: Max / Toroid 0 T / Male Summary Data Sheet.....	443
Table 43: Min / Toroid 0 T / Male Summary Data Sheet.....	444
Table 44: Max / Toroid 1.5 T / Female Summary Data Sheet.....	445
Table 45: Min / Toroid 1.5 T / Female Summary Data Sheet.....	446
Table 46: Max / Toroid 1.5 T / Male Summary Data Sheet.....	447
Table 47: Min / Toroid 1.5 T / Male Summary Data Sheet.....	448
Table 48: Max / Toroid 3 T / Female Summary Data Sheet.....	449
Table 49: Min / Toroid 3 T / Female Summary Data Sheet .....	450

Table 50: Max / Toroid 3 T / Male Summary Data Sheet.....	451
Table 51: Min / Toroid 3 T / Male Summary Data Sheet.....	452
Table 52: Max / Toroid 7 T / Female Summary Data Sheet.....	453
Table 53: Min / Toroid 7 T / Female Summary Data Sheet .....	454
Table 54: Max / Toroid 7 T / Male Summary Data Sheet.....	455
Table 55: Min / Toroid 7 T / Male Summary Data Sheet.....	456
Table 56: Max / Solenoid 0 T / Female Summary Data Sheet.....	457
Table 57: Min / Solenoid 0 T / Female Summary Data Sheet.....	458
Table 58: Max / Solenoid 0 T / Male Summary Data Sheet.....	459
Table 59: Min / Solenoid 0 T / Male Summary Data Sheet.....	460
Table 60: Max / Solenoid 1.5 T / Female Summary Data Sheet.....	461
Table 61: Min / Solenoid 1.5 T / Female Summary Data Sheet.....	462
Table 62: Max / Solenoid 1.5 T / Male Summary Data Sheet.....	463
Table 63: Min / Solenoid 1.5 T / Male Summary Data Sheet.....	464
Table 64: Max / Solenoid 3 T / Female Summary Data Sheet.....	465
Table 65: Min / Solenoid 3 T / Female Summary Data Sheet.....	466
Table 66: Max / Solenoid 3 T / Male Summary Data Sheet.....	467
Table 67: Min / Solenoid 3 T / Male Summary Data Sheet.....	468
Table 68: Max / Solenoid 7 T / Female Summary Data Sheet.....	469
Table 69: Min / Solenoid 7 T / Female Summary Data Sheet.....	470
Table 70: Max / Solenoid 7 T / Male Summary Data Sheet.....	471
Table 71: Min / Solenoid 7 T / Male Summary Data Sheet.....	472
Table 72: Max / Race Track 0 T / Female Summary Data Sheet .....	473
Table 73: Min / Race Track 0 T / Female Summary Data Sheet .....	474
Table 74: Max / Race Track 0 T / Male Summary Data Sheet.....	475
Table 75: Min / Race Track 0 T / Male Summary Data Sheet.....	476

Table 76: Max / Race Track 1.5 T / Female Summary Data Sheet.....	477
Table 77: Min / Race Track 1.5 T / Female Summary Data Sheet.....	478
Table 78: Max / Race Track 1.5 T / Male Summary Data Sheet.....	479
Table 79: Min / Race Track 1.5 T / Male Summary Data Sheet.....	480
Table 80: Max / Race Track 3 T / Female Summary Data Sheet .....	481
Table 81: Min / Race Track 3 T / Female Summary Data Sheet .....	482
Table 82: Max / Race Track 3 T / Male Summary Data Sheet.....	483
Table 83: Min / Race Track 3 T / Male Summary Data Sheet.....	484
Table 84: Max / Race Track 7 T / Female Summary Data Sheet .....	485
Table 85: Min / Race Track 7 T / Female Summary Data Sheet .....	486
Table 86: Max / Race Track 7 T / Male Summary Data Sheet.....	487
Table 87: Min / Race Track 7 T / Male Summary Data Sheet.....	488
Table 88: List of Experimental Runs at the PTC.....	498

## Abbreviations

A	Ampere
A	Area
A	Atomic Mass
AES	Advanced Exploration Systems
Al	Aluminum
ALARA	As Low As Reasonably Achievable
Al-Li	Aluminum-Lithium
ARSSEM	Active Radiation Shield for Space Exploration Missions
AU	Astronomical Unit
B	Boron
B	Magnetic Field
BEIR	Committee on Biological Effects of Ionizing Radiation
Be	Beryllium
BFO	Blood-Forming Organs
BNL	Brookhaven National Laboratory
BON	Badhwar-O'Neill
BSCCO	Bismuth Lead Strontium Calcium Copper Oxygen
C	Carbon
Ca	Calcium
cGy	Centi Gray
CI	Confidence Interval
cm	Centimeter
CME	Coronal Mass Ejection
CNS	Central Nervous System
Cr	Chromium
CT	Computed Tomography
D	Dose
DDREF	Dose and Dose Rate Effectiveness Factor
DNA	Deoxyribonucleic Acid
DREF	Dose Rate Effectiveness Factor
E	Complete Elliptic Integral, 2 <sup>nd</sup> Kind
E	Effective Dose
E	Energy
EMI	Electromagnetic Interference

EMP	Electromagnetic Pulse
ESA	European Space Agency
EU	European Union
eV	Electronvolt
F	Fluorine
FASH	Female Adult Mesh
FAX	Female Adult Voxel
Fe	Iron
FLUKA	Fluctuating Kascades
g	Gram
GCR	Galactic Cosmic Ray
GEANT4	Geometry and Tracking 4
GEO	Geosynchronous Orbit
GeV	Giga Electronvolt
Gy	Gray
H	Dose Equivalent
H	Hydrogen
H	Dose Equivalent
H <sub>2</sub> O	Water
HDPE	High-Density Polyethylene
He	Helium
HTS	High Temperature Superconductor
HWE	Healthy Worker Effect
HZE	High Charge (Z) and Energy
HZETRN	High Charge (Z) and Energy Transport program
i	Current
ICNIRP	International Commission on Non-Ionizing Radiation
ICRP	International Commission on Radiation Protection
ICRU	International Commission on Radiation Units and Measurements
ISO	International Organization for Standardization
ISS	International Space Station
ITS	Intermediate Temperature Superconductor
K	Kelvin
K	Complete Elliptic Integral, 1 <sup>st</sup> Kind
keV	Kilo Electronvolt

kg	Kilogram
km	Kilometer
LEO	Low Earth Orbit
LET	Linear Energy Transfer
Li	Lithium
LIS	Local Interstellar Spectrum
LNT	Linear-Non-Threshold
LTS	Low Temperature Superconductor
m	Meter
MAARS	Magnet Architectures and Active Radiation Shielding
MASH	Male Adult Mesh
MAX	Male Adult Voxel
MeV	Mega Electronvolt
Mg	Magnesium
MgB <sub>2</sub>	Magnesium Diboride
mGy	Milli Gray
mm	Millimeter
MIRD	Medical Internal Radiation Dose
Mn	Manganese
MOF	Metal Organic Framework
MPCV	Multi-Purpose Crew Vehicle
MRI	Magnetic Resonance Imaging
mT	Milli Tesla
n	Number of turns
N	Number of particles
N	Nitrogen
Na	Sodium
NASA	National Aeronautics and Space Administration
NbTi	Niobium Titanium
NCRP	National Council on Radiation Protection and Measurements
Ne	Neon
NIAC	NASA Innovative Advanced Concepts
NOAA	National Oceanic and Atmospheric Administration
NRC	National Research Council
NSCR	NASA Space Cancer Risk

NSRL	NASA Space Radiation Laboratory
O	Oxygen
ORNL	Oak Ridge National Laboratory
P	Phosphorous
PEL	Permissible Exposure Limit
$\Phi$	Fluence Rate
PTC	Proton Therapy Center
Q	Quality Factor
r	Radius
RBE	Relative Biological Effectiveness
REID	Risk of Exposure-Induced Death
RF	Radiofrequency
RMS	Root Mean Square
Sc	Scandium
SDS	Safe Days in Space
$\sigma$	Standard Deviation
Si	Silicon
SI	International System of units
SLS	Space Launch System
SPE	Solar Particle Event
SR2S	Space Radiation Superconducting Shielding
Sv	Sievert
T	Tesla
t	Ton (metric)
TeV	Tera Electronvolt
Ti	Titanium
UN	United Nations
UNSCEAR	United Nations Committee on Exposure to Atomic Radiation
WHO	World Health Organization
$w_T$	Tissue Weighting Factor
XCAT	Extended Cardiac Torso
YBCO	Yttrium Barium Copper Oxide
Z	Atomic Number

## Specific Aims

Exploration of interplanetary space presents elevated hazards to human survival. Some of these hazards cause acute risks that can lead to complete loss of the crew and mission, while other hazards cause chronic risks, the effects of which may not manifest for several years upon the crew's return to Earth. Hazards of the natural space environment outside the protection of the Earth's magnetosphere, particularly ionizing radiation, cause both acute and chronic risks to crew survival and thus can become limiting factors for NASA's planned mission to Mars by the 2030s.

Radiation exposure on a mission beyond Earth orbit is primarily the result of high energy ions from galactic cosmic rays and moderate energy protons from solar particle events. The chronic effective radiation dose due to galactic cosmic rays on a typical Mars mission is on the order of 1 Sv when no shielding is present. Solar activity is sporadic, but a solar particle event could deliver an additional acute dose of up to 4 Sv, which is a potentially lethal dose. Hence, radiation protection is a critical concern on these types of missions.

Various methods of radiation shielding have been proposed, from simple passive shielding by materials such as water, polyethylene, or aluminum, to active shielding systems comprised of electromagnetic fields. The concept of active magnetic shielding is to employ high-temperature superconductors to induce very high magnetic fields (1-10 T) around the spacecraft. The magnetic field will deflect incoming charged particles, thereby reducing the radiation dose to astronauts behind the shield.

The goal of this project is to determine the value of active magnetic shielding in reducing radiation dose to astronauts on an interplanetary mission. **Our central hypothesis is active**

**magnetic shielding reduces effective radiation dose to astronauts on a Mars flyby mission by 50% versus no shielding.** This hypothesis will be investigated via the following specific aims:

**Specific Aim One:** Create a Monte Carlo model which includes the space radiation environment as a function of solar cycle and mission profile, a typical spacecraft architecture, passive and active shielding configurations, and an organ-based representation of an astronaut.

**Specific Aim Two:** Calculate space radiation-induced cancer risk to humans (astronauts) inside a typical spacecraft as a function of shielding type, magnetic field, astronaut age, and astronaut sex.

**Specific Aim Three:** Identify and calculate the factors with the greatest impact on effective dose and cancer risk to astronauts on interplanetary missions. Draw conclusions on the feasibility of various shielding configurations to mitigate risk for interplanetary travel.

The results of this project inform the human spaceflight community on the utility of magnetic shielding as compared to passive or minimal shielding, based upon an end-to-end model. Effective dose versus astronaut age and sex, magnetic field, and shielding configuration provide recommendations on future investment in magnetic shielding technology and addressing engineering challenges.

# 1 Introduction

## 1.1 Motivation

Exploration of interplanetary space presents elevated hazards to human survival. Some of these hazards cause acute risks that can lead to complete loss of the crew and mission, while other hazards cause chronic risks, the effects of which may not manifest for several years upon the crew's return to Earth (Battista 2016). Figure 1 below shows NASA's highest priority human spaceflight risks, reproduced from a NASA presentation at the 2015 Aerospace Medical Association Conference in Orlando, Florida (Francisco 2015). The risk of space radiation exposure is one of the top priority risks and is currently one of the least well-managed risks with the highest post-mission consequence. As of early 2020, NASA is considering whether radiation dose limits should be relaxed to accommodate interplanetary missions in the next decade, but no official decision has been announced.

Human Spaceflight Risks	In Mission Risk - Operations						Post Mission Risk – Long Term Health					
	Low Earth Orbit	Low Earth Orbit	Deep Space Sortie	Lunar Visit/Habitation	Deep Space Journey/Habitation	Planetary	Low Earth Orbit	Low Earth Orbit	Deep Space Sortie	Lunar Visit/Habitation	Deep Space Journey/Habitation	Planetary
	6 Months	12 Months	30 Days	1 year	1 Year	3 years	6 Months	12 Months	30 Days	1 year	1 Year	3 years
VIIIP	A	A	A	A	RM	RM	A	A	A	A	RM	RM
Renal Stone Formation	A	A	A	A	RM	RM	RM	RM	RM	RM	RM	RM
Inadequate food and nutrition	A	A	A	A	RM	RM	A	A	A	A	A	RM
Risk of Space Radiation Exposure	A	A	A	A	A	A	A	A	A	RM	RM	RM
Medications Long Term Storage	A	A	A	A	A	RM	A	A	A	A	A	RM
Acute and Chronic Carbon Dioxide	A	A	A	A	RM	RM	A	A	A	A	A	A
Inflight Medical Conditions	A	A	A	RM	RM	RM	A	A	A	A	RM	RM
Cognitive or Behavioral Conditions	A	A	A	A	RM	RM	A	A	A	RM	RM	RM
Risk of Bone Fracture	A	A	A	A	A	A	A	A	A	A	A	RM
Team Performance Decrements#	A	A	A	A	RM	RM	A	A	A	A	A	A
Reduced Muscle Mass, Strength	A	A	A	A	A	A	A	A	A	A	A	A
Reduced Aerobic Capacity	A	A	A	A	A	RM	A	A	A	A	A	A
Sensorimotor Alterations	A	A	A	A	A	RM	A	A	A	A	A	RM
Human-System Interaction Design#	A	A	A	RM	RM	RM	A	A	A	A	A	A
Injury from Dynamic Loads	A	A	RM	RM	RM	RM	A	A	RM	RM	RM	RM
Sleep Loss	A	A	A	A	RM	RM	A	A	A	A	RM	RM
Altered Immune Response	A	A	A	A	RM	RM	A	A	A	A	A	RM
Celestial Dust Exposure	N/A	N/A	A	TBD	TBD	TBD	N/A	N/A	A	TBD	TBD	TBD
Host-Microorganism Interactions	A	A	A	A	RM	RM	A	A	A	A	A	RM
Injury due to EVA Operations	A	A	A	RM	A	RM	A	A	A	RM	RM	RM
Decompression Sickness	A	A	A	A	RM	A	A	A	A	RM	A	RM
Toxic Exposure	A	A	A	A	A	A	A	A	A	A	A	A
Hypobaric Hypoxia	A	A	A	A	A	A	A	A	A	A	A	A
Space Adaptation Back Pain	A	A	A	A	A	A	N/A	N/A	N/A	N/A	N/A	N/A
Urinary Retention	A	A	A	A	A	A	A	A	A	A	A	A
Hearing Loss Related to Spaceflight	A	A	A	A	A	A	A	A	A	A	A	A
Orthostatic Intolerance	A	A	A	A	A	A	A	A	A	A	A	A
Injury from Sunlight Exposure - retired	A	A	A	A	A	A	A	A	A	A	A	A
Risk of electrical shock - Retired	A	A	A	A	A	A	A	A	A	A	A	A

A – Accepted    RM- Requires Mitigation    Green – low/very low consequence    Yellow – low to medium consequence    Red – high consequence

Figure 1: NASA Human Spaceflight Risk Summary (Francisco 2015); courtesy of NASA

The natural space environment outside the protection of the Earth's magnetosphere, particularly ionizing radiation, can produce both acute and chronic risks to crew survival (Carnell et al. 2016; Huff et al. 2016; Nelson et al. 2016; Patel et al. 2016; NCRP 1989, 2014) as well as risk of damage to spacecraft electronics (Bedingfield et al. 1996; Koons et al. 1999; Novikov et al. 2009; U.S. Air Force 1985).

Space radiation risks of particular concern for astronauts include carcinogenesis, degenerative tissue diseases, central nervous system effects, and acute radiation syndromes. The specific biological mechanisms for tissue damage in each of these categories remain uncertain, though several studies have begun to shed light on the general pathways including DNA damage, chromosome aberration, radiation-induced apoptosis, persistent oxidative stress, chronic inflammation, and accelerated tissue aging (Autsavapromporn et al. 2011, 2013; George et al. 2001; Hellweg & Baumstark-Khan 2007; Patel et al. 2016; Roig et al. 2010).

Insight into space radiation carcinogenesis can be gained from evaluating the risk of secondary cancers resulting from radiation therapy on Earth (Brodin et al. 2012). The risk of carcinogenesis is highest in adults for leukemia, lung, breast, stomach, colon, bladder, and liver cancers (Huff et al. 2016). The relative risk depends on many factors such as the total dose, radiation species and energy, sex and age of the astronaut at the time of exposure. Carcinogenesis risk requires further mitigation for any long-duration human missions traveling beyond Earth orbit (Huff et al. 2016).

Degenerative tissue diseases can affect the cardiovascular, respiratory, or digestive systems and also include cataractogenesis (Ainsbury et al. 2009; Cucinotta et al. 2001a). Animal models have also shown degenerative bone loss due to exposure to several radiation species (Hamilton et al. 2006). Again, insight into these risks can be obtained from late effects of radiation therapy

on Earth (Brodin et al. 2012) and are caused by radiation effects including DNA damage, persistent oxidative stress, chronic inflammation, and accelerated tissue aging (Patel et al. 2016). These risks also require further mitigation for any long-duration human missions traveling beyond Earth orbit.

Central nervous system (CNS) effects can be acute (occurring during flight) or late (occurring at some time after conclusion of the flight). Acute risks include altered cognitive function, impaired motor function, and behavior changes; late risks include neurological disorders, dementia, or premature aging (Nelson et al. 2016). The specific biological mechanisms for tissue damage in each of these categories are still under investigation. These risks require further mitigation for long-duration human missions beyond Earth orbit (Nelson et al. 2016).

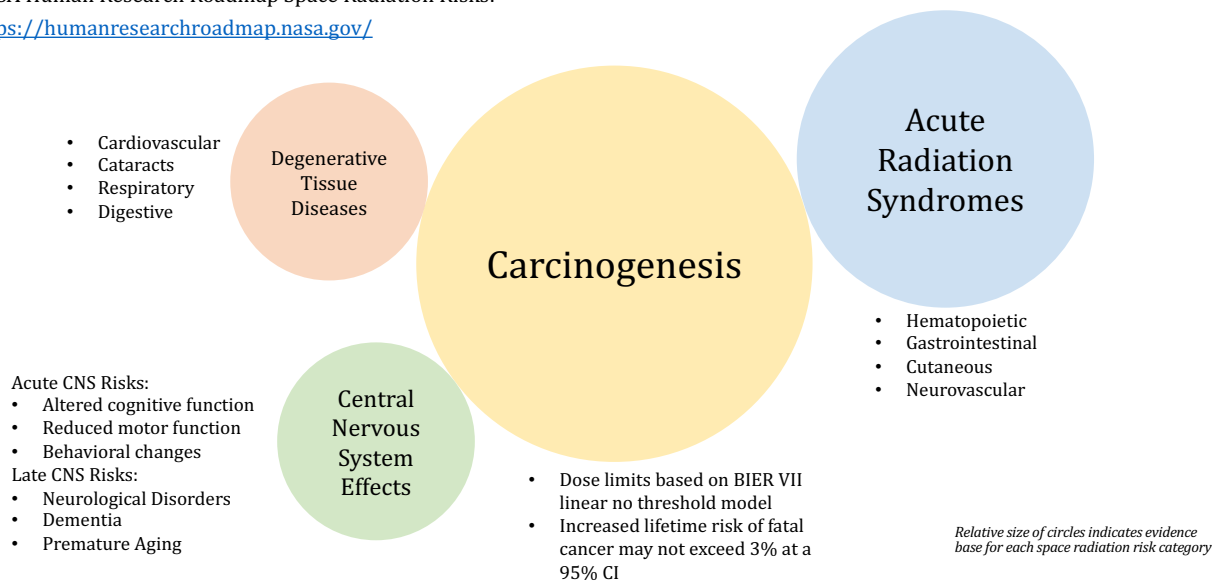
Acute radiation syndromes result from dynamic solar particle event exposures that produce a high dose rate for a period of hours to days. The effects can include hematopoietic, gastrointestinal, cutaneous, and neurovascular damage (Carnell et al. 2016). These risks can be compared to the effects of large radiation doses experienced on Earth and can have serious consequences for the crew during the mission (Wu et al. 2009). However, NASA currently considers this risk mitigated by available passive shielding and detection/warning systems (Carnell et al. 2016; Cougnet et al. 2005).

The considerable space radiation risks to astronauts are summarized in Figure 2 and could become limiting factors for NASA's planned mission to Mars by the 2030s. Thus, it is reasonable to conclude that human missions to Mars will not occur without first reducing space radiation risks (Koontz 2013; McKenna-Lawlor 2014; Sinclair 2000). The risk of acute radiation exposure from solar particle events can be reduced with current passive shielding methods (Semones 2015a; Badhwar & Cucinotta 1998) but not eliminated. Chronic radiation exposure from GCRs is

a more serious and current limiting factor (Setlow 1999), since current passive shielding techniques are not adequate to shield against high atomic number, high energy (HZE) ions (Bond et al. 2019; Manning & Singleterry 2020; Singleterry 2013). In some cases, passive shielding can even exacerbate the problem by inducing excessive secondary particle radiation (Geng et al. 2015; NRC 2008; Shinn et al. 1994; Slaba et al. 2013), though a recent study suggests that fragmentation can reduce overall biological effect at GCR energies by reducing the high-LET component (Zeitlin 2016).

NASA Human Research Roadmap Space Radiation Risks:

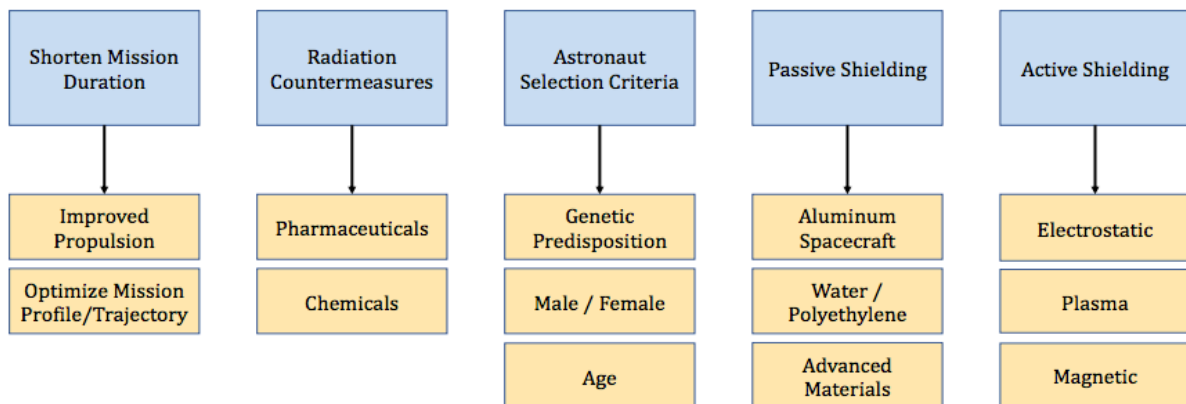
<https://humanresearchroadmap.nasa.gov/>



**Figure 2: NASA Human Research Roadmap Space Radiation Risks**

NASA and other space agencies are currently studying alternative methods for reducing space radiation risk, including reduced mission duration with advanced propulsion (Chang-Díaz et al. 2000), pharmaceutical or chemical countermeasures (A. Kennedy 2014; A. Kennedy & Wan 2011; Langell et al. 2008; McLaughlin et al. 2017), astronaut selection criteria (Chancellor et al. 2014; Javad Mortazavi et al. 2003), advanced materials for passive shielding (Bond et al. 2019; Manning & Singleterry 2020; Moses et al. 2018; Singleterry 2013; Wilson et al. 1998), and active shielding (Bamford et al. 2008; Vuolo et al. 2016; Westover 2014).

Some of these methods (see Figure 3) show promise but are technologically distant in the case of advanced propulsion or highly individualized in the case of countermeasures and astronaut selection criteria. In the case of active shielding methods, including electrostatic, plasma, and magnetic, there is not enough data yet available to determine their utility (AVCO- Everett Research Laboratory 1968; Buhler 2004; Frisina 1985; Levy & French 1967; Musenich et al. 2018; Townsend 1984; Tripathi et al. 2008; Westover et al. 2012). Active shielding techniques have been studied since the early 1960s, but few have graduated from the conceptual realm for several reasons including limited computing capability for modeling in early years and limited technology investment in the technology.



**Figure 3: Options to Mitigate Space Radiation Risk on Interplanetary Missions**

The human spaceflight community is anxious for justification on whether to invest substantial funding in the further development of active shielding methods, and decision-makers need tangible results and data to guide budgets and space policy over the next 10-20 years. The results of this project address that need by providing detailed simulation results to guide future investment and technology development strategy for magnetic shielding. Furthermore, if our experiments are successful in demonstrating that magnetic shielding makes sense scientifically, this should encourage further engineering studies to determine technical feasibility and designs for magnetic shielding systems that could facilitate human spaceflight to Mars, Jupiter, or asteroids.

## *1.2 History of Active Magnetic Shielding*

This section details the history of active magnetic shielding development to date, describing successes, challenges, and remaining technological gaps. A timeline of major milestones to date is provided in Figure 4. Trends are evident in the frequency of active shielding studies; these trends correspond to government investment in space exploration and the availability of enabling technologies such as high temperature superconducting materials. A graphical summary of these trends is provided in Figure 5.

### *1.2.1 The Pre-NASA Era*

The discovery of superconductivity was made by Onnes in 1911 (Onnes 1911). However, a sound and theoretical understanding of the phenomenon was not complete until the discovery in the 1950s of Type II superconducting materials that can retain superconducting properties in high magnetic fields (Tinkham 1996; Urban 1969). At that point, scientists and engineers began to conceive the idea of applying magnetism and superconductivity for space radiation protection. However, the unique historical context at the time indicated that “such research and development was oriented, of course, toward the advancement of rocket-borne weapons rather than rockets for space exploration and other peaceful purposes” (Logsdon 1995), including protection of satellites from natural and manmade radiation sources (Christofilos 1959) after the first U.S. satellites discovered the Earth’s trapped radiation belts (Van Allen et al. 1958).

### *1.2.2 The 1960s*

In 1962 U.S. President John F. Kennedy’s Rice University address and resulting space policy accelerated the objectives of the first three U.S. human spaceflight programs (Mercury, Gemini, and Apollo), aiming for a first human landing on the Moon by the end of the decade, with Mars missions not far behind (Logsdon 1995; J. Kennedy 1962). Recognizing the potential of electromagnetic shielding for space radiation protection of human spacecraft, the U.S. Air Force

and NASA sponsored several studies and workshops on electric and magnetic shielding on lunar and interplanetary missions in the 1960s (Atomic Energy Commission 1962, 1964; Baldwin et al. 1964; Bernert et al. 1964; Brown 1961; Dow et al. 1962; French 1970; Good et al. 1964; Kash 1965; Kash & Tooper 1962; Levine et al. 1966; Levine & Lepper 1968a, 1968b; Levy 1961, 1962; McDonald 1965; McDonald & Buntyn 1966; White 1963).

These studies and workshops covered the basis of several sweeping assumptions based on the current knowledge of the environment and technology available at the time. Thus, these studies focused primarily on mass comparisons with passive shielding (Kash 1963). Further, the analysis was performed only for moderate energy solar protons (~200–500 MeV) and electrons (Bhattacharjie & Michael 1964). The low dose rate GCRs were omitted from the analysis because the acute risk to astronauts from solar particles was considered to be the more serious space radiation risk at the time (Norwood & Gibbons 1963; NASA 1970). In addition, secondary particle showers created in passive shielding materials were mostly ignored in these studies, likely due to limited computing capability to conduct radiation transport simulations. Also, because only low temperature superconductors were known at the time, these designs included complex cryogenics systems to maintain the superconducting coils at liquid helium temperatures. The added complexity and power requirements of the cryogenics systems limited the possible shielding configurations. Finally, these early studies on active shielding in general assumed that engineering solutions would solve any technology gaps.

Towards the end of the decade, however, magnetic shielding had almost become a mainstream concept, with active shielding topics dominating discussions at the Special Sessions on Protection Against Space Radiation at the American Nuclear Society meeting in 1967 (Reetz & O'Brien 1968), and Wernher von Braun published a report on magnetic shielding in *Popular Science* in 1969 (von Braun 1969).

### 1.2.3 The 1970s

Magnetic shielding development stagnated in the U.S. throughout the 1970s, with few new publications (Levine & Lepper 1971; Paluszek 1978). During the same time, many investigations were reported on other space applications of superconductors, including spacecraft propulsion by magnetic induction (Engelberger 1970), high field magnets for particle physics analysis, magnetometers, digital electronics, microwave and infrared detectors, gravitational instruments, and high-Q superconducting cavities and oscillators (Sullivan 1978).

Concurrently, the promise of a near-term Mars mission dwindled following the early termination of the Apollo program (Space Task Group 1969) and U.S. President Richard Nixon's announcement to focus on low earth orbit with the development of the Space Shuttle (Nixon 1972). This redirection of U.S. space priority reduced NASA's interest in developing exploration technologies, and thus U.S. studies on active shielding ground to a halt. The Soviets, however, began investigating active shielding concepts during this time (Morozov et al. 1971; Trukhanov & Morozov 1970; Grishin et al. 1978).

### 1.2.4 The 1980s

U.S. human spaceflight was rejuvenated in the early 1980s as the Space Shuttle took flight and plans for Space Station Freedom were initiated under U.S. President Ronald Reagan (Reagan 1983, 1984). However, deeper exploration missions were not considered a near-term priority. In addition, the years of 1986-1987 were consumed with the investigation of the Space Shuttle *Challenger* accident, resulting in a halt of all other mission priorities and a focus on safety improvements (Presidential Commission on the Space Shuttle *Challenger* Accident 1986). Despite these obstacles, NASA began to consider using magnetic shielding to protect against cosmic ray ions as well as electrons and solar protons (Townsend 1983; Birch 1982), though most of the analyses remained focused on mass savings as compared to passive shielding.

### 1.2.5 The 1990s to Early 2000s

The discovery of high temperature (70–100 K) superconducting materials in 1986 (Bednorz & Müller 1986), newly available advanced computing capabilities, the success of large magnets within particle accelerators (Spillantini 2014), U.S. President George H.W. Bush's announcement of the *Space Exploration Initiative* in 1989 (NASA 1990; Synthesis Group on America's Space Exploration Initiative 1991), and the U.S. Congressional report *Exploring the Moon and Mars: Choices for the Nation* in 1991 (U.S. Congress Office of Technology Assessment 1991) sparked new activities related to magnetic shielding for deep space missions in the early 1990s (F. Cocks 1991; F. Cocks & Watkins 1993; Herring & Merrill 1991; Landis 1991).

If configured correctly, the new high temperature superconductors could reach an equilibrium temperature in their superconducting range in space without the need for complex cryogenic refrigeration (Heinen & Connolly 1990), allowing the magnets to be deployed away from the spacecraft. In this concept, thin, flexible films coated with superconducting powder are deployed at a distance, reducing the current and stored energy required to produce the same level of shielding as a spacecraft-mounted coil (Hilinski & F. Cocks 1994). However, the size and complexity of deploying such a system hindered further development of the concept.

In the late 1990s and early 2000s and under U.S. President Bill Clinton's 1996 *National Space Policy* (White House National Science and Technology Council 1996), NASA focused on the Space Shuttle and International Space Station programs, and there was little motivation to further develop enabling technologies for interplanetary human spaceflight. The concept of magnetic shielding was again tabled, though several review articles were published on the topic (J. Cocks et al. 1997; Spillantini 2000; Sussingham et al. 1999; Townsend 2000), and the European Space Agency (ESA) chartered a Topical Team in 2002 to study questions related to passive and active shielding for solar radiation. The ESA team recommended magnetic shielding technology be

developed for solar particle event storm shelters by 2025 (Spillantini et al. 2007). However, as Townsend points out in his review, even into the 21<sup>st</sup> century “very few analyses of the efficacy of active shielding methods for protecting spacecraft crews consider the total spectrum (GCR and SPE) likely to be encountered on a deep space mission. Nearly all analyses have focused solely on SPE protons, thereby ignoring the biologically damaging GCR spectrum” (Townsend 2000).

The loss of Space Shuttle *Columbia* in 2003 further set back exploration plans, as NASA grounded all flights for over two years to focus on the accident investigation and to implement technical and cultural safety improvements (Columbia Accident Investigation Board 2003).

#### *1.2.6 The Mid to Late 2000s*

In 2004, U.S. President George W. Bush announced the *Vision for Space Exploration*, officially targeting human missions to the Moon and Mars by the 2020s (NASA 2004; President’s Commission on Implementation of United States Space Exploration Policy 2004; White House 2004). Subsequently, the question of how to best protect the astronauts against interplanetary space radiation was again brought to light, and a workshop was held at NASA Marshall Space Flight Center in 2004 to assess a list of “Revolutionary Physical Sciences Radiation Protection Strategies” assembled by NASA Headquarters. Active shielding methods dominated the discussion, and results of the meeting were published via survey articles (Adams et al. 2005; Spillantini et al. 2007), criticisms (Townsend 2005a), and even popular science (Parker 2005, 2006) detailing the four major categories of active shielding: electrostatic, plasma, confined magnetic field and unconfined magnetic field.

NASA’s report from the 2004 workshop of the Advanced Radiation Protection Working Group identified the advantages and disadvantages of four different types of active shielding:

- The **electrostatic** shield concept is to use a strong electric field to deflect incoming solar and cosmic ray particles. The electric field required is on the order of  $10^{10}$  V. The concept was deemed impractical because of the safety implications of such a strong electric field; the secondary radiation would also be a concern.
- The **plasma** shield concept is to use a magnetic field to trap charged particles, creating a plasma that will induce a strong electric field to deflect incoming solar and cosmic ray particles. The electric field required is on the order of  $10^{10}$  V, and some configurations also involve a large magnetic field. The accelerated development of trapped radiation belts quickly reduces the effectiveness of this type of shield; thus, the concept was also deemed impractical.
- The **confined magnetic field** concept is to use a strong magnetic field to deflect incoming solar and cosmic ray particles using a magnetic coil configuration that minimizes or eliminates fringe fields. A double-walled torus was suggested to minimize crew exposure to fringe fields. Previous studies found that the mass required for the configuration was greater than the mass of comparable passive shielding material, thus this concept was also deemed impractical.
- The **unconfined magnetic field** concept is to use a strong magnetic field to deflect incoming solar and cosmic ray particles using a magnetic coil configuration that permits fringe fields to act on particles at large distances from the magnet. Many possible configurations of an unconfined magnetic field were considered. However, the calculations on the amount of stored energy required for a fully-deployed superconducting shield were approximately  $10^{15}$  J. Unless a very large, very weak field can be produced via multiple coils, the concept was also deemed impractical.

In summary, the first three types of active shielding were deemed impractical due to mass, power, and safety concerns, while the unconfined magnetic field option was highlighted for further study (Adams et al. 2005).

In the mid to late 2000s, several assumptions and simplifications from previous active shielding studies were challenged (Townsend 2005a). The assumptions from the 1960s and 1970s focused only on solar particles, to track only primary radiation, to allow magnetic fields to penetrate the habitable volume, and to broadly assume engineering would solve any lagging technology gaps that severely oversimplified the problem. More detailed studies on effective dose behind shielding (Townsend 2005b) magnetic field neutralization inside spacecraft (Neuberth & Westphal 2011; S. Shepherd & J. Shepherd 2009), and advanced superconducting materials (Kervendal et al. 2009) emerged that tracked these important variables and updated the effective dose estimates for interplanetary missions. Advanced radiation transport codes were developed (HZETRN), or modified (FLUKA, GEANT4) for space radiation applications (Truscott et al. 2000) to enable more complex simulations of active and passive shielding methods (Heinbockel et al. 2009).

In the National Research Council's 2008 report *Managing Space Radiation Risk in the New Era of Space Exploration*, several knowledge gaps relating to magnetic shielding feasibility were identified and recommendations were provided for future studies. These recommendations included accurate modeling of the space radiation environment to include all relevant particle types and energies as well as detailed transport analysis that considers the production and interaction of secondary particles (NRC 2008).

By the end of the decade, the small satellite revolution was underway beginning with the 1.3 kg, 10x10x10 cm<sup>3</sup> CubeSat, and the miniaturization of space hardware was a salient topic. Mini-

magnetic shields were proposed for satellites, for propulsion (Winglee et al. 2000) as well as for radiation protection (Bamford et al. 2008).

#### *1.2.7 The 2010s and Beyond*

Into the 2010s, Mars remained the ultimate target destination for NASA, despite setbacks including the cancellation of the Constellation program in 2010 (Bolden & Holdren 2010) following recommendations from the Augustine Commission (Review of U.S. Human Spaceflight Plans Committee 2009). Several studies were commissioned to further develop active shielding technology through simulated field configurations, engineering trade studies, and comparisons to advanced passive shielding methods (Ambroglini et al. 2016; Battiston et al. 2011, 2013; Bruce & Baudouy 2015; Durante 2014; Geng et al. 2015; Joshi et al. 2013; Musenich et al. 2014; Papini & Spillantini 2014; Peroni 2017; Singleterry et al. 2015; Spillantini 2010, 2011, 2014; Washburn et al. 2014, 2015; Westover 2014; Westover et al. 2012). Multiple patents were also granted on the topic (Kinstler 2012; Neuberth & Westphal 2011), and an updated analysis on the limits of magnetic shielding was also published (Musenich et al. 2018). The European Union commissioned the Space Radiation Superconducting Shielding (EU FP7 SR2S) project in 2015 to complete Monte Carlo simulations of active magnetic shielding (Vuolo et al. 2016).

In 2017, U.S. President Donald Trump's *Presidential Memorandum on Reinvigorating America's Human Space Exploration Program*, details the priority for NASA, "Beginning with missions beyond low-Earth orbit, the United States will lead the return of humans to the Moon for long-term exploration and utilization, followed by human missions to Mars and other destinations" (White House 2017). With this new vision in mind, there is great motivation for further development of active magnetic shielding as an enabling technology for human exploration missions.

### *1.2.8 Discussion*

The development of magnetic shielding as an enabling technology appears to depend on the prospect of near-term human exploration missions. Figure 5 shows the total NASA budget since inception corrected for inflation to 2018 U.S. dollars and the number of publications on magnetic shielding for each year since 1958. Important milestones are also marked to provide additional context.

Motivating funding agencies such as NASA to allocate space exploration funding for advanced technology development, regardless of the current space policy environment, requires solid justification from scientific studies that include prototyping and testing. Our project studies a complete active shielding system that includes accurate models of the space radiation environment, realistic models of an interplanetary spacecraft, organ-based human phantoms that can more accurately estimate effective radiation dose and exposure-induced cancer risk, laboratory-based prototyping and testing for model validation, and sensitivity analysis of many mission factors (e.g. mission type and duration, solar activity, shielding type, magnetic field, and crew characteristics). Favorable results from this study provide justification for funding agencies such as NASA to provide increased funding for the development of active magnetic shielding as an enabling technology for human exploration missions.

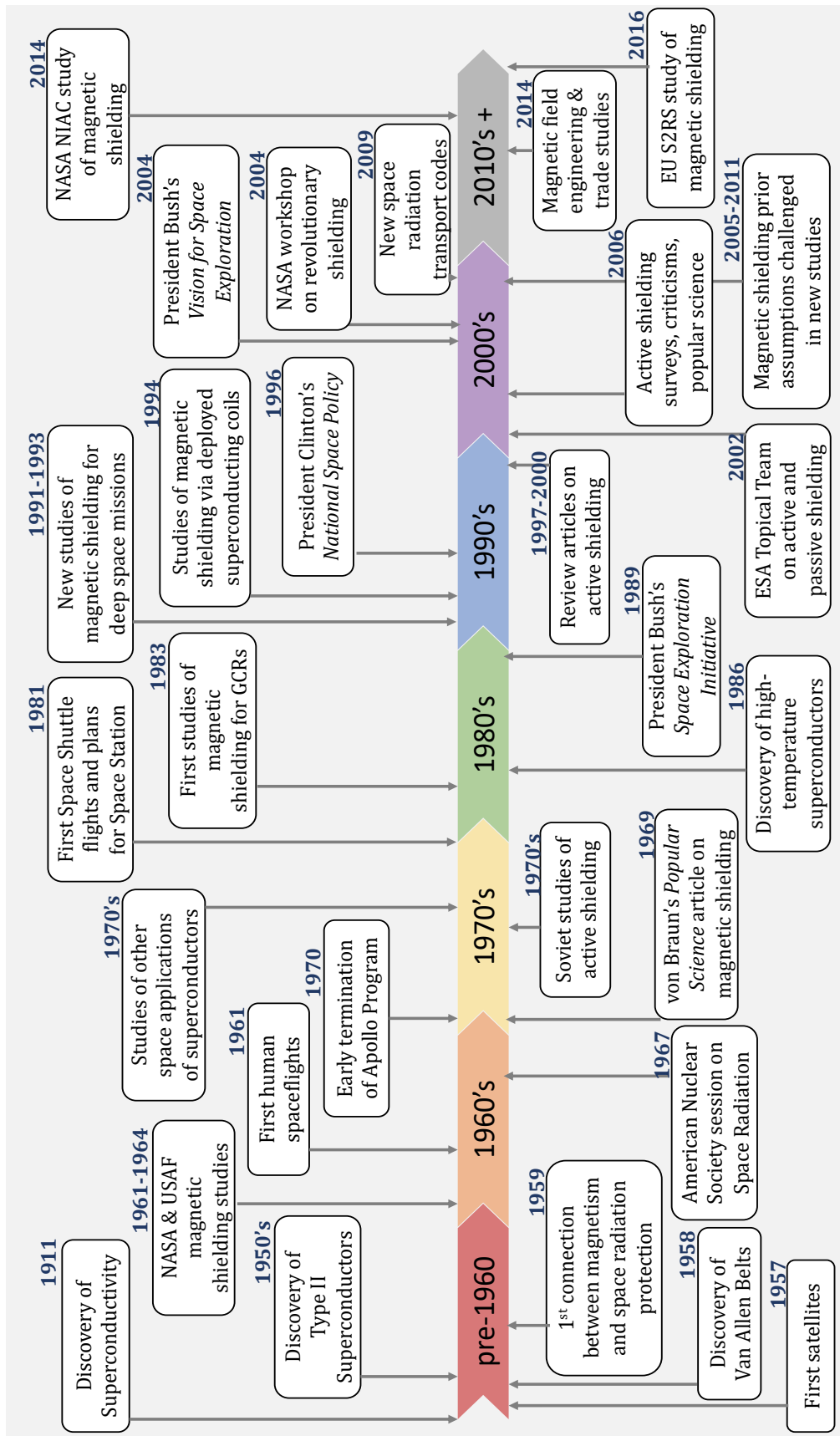


Figure 4: 50+ Years of Magnetic Shielding Studies

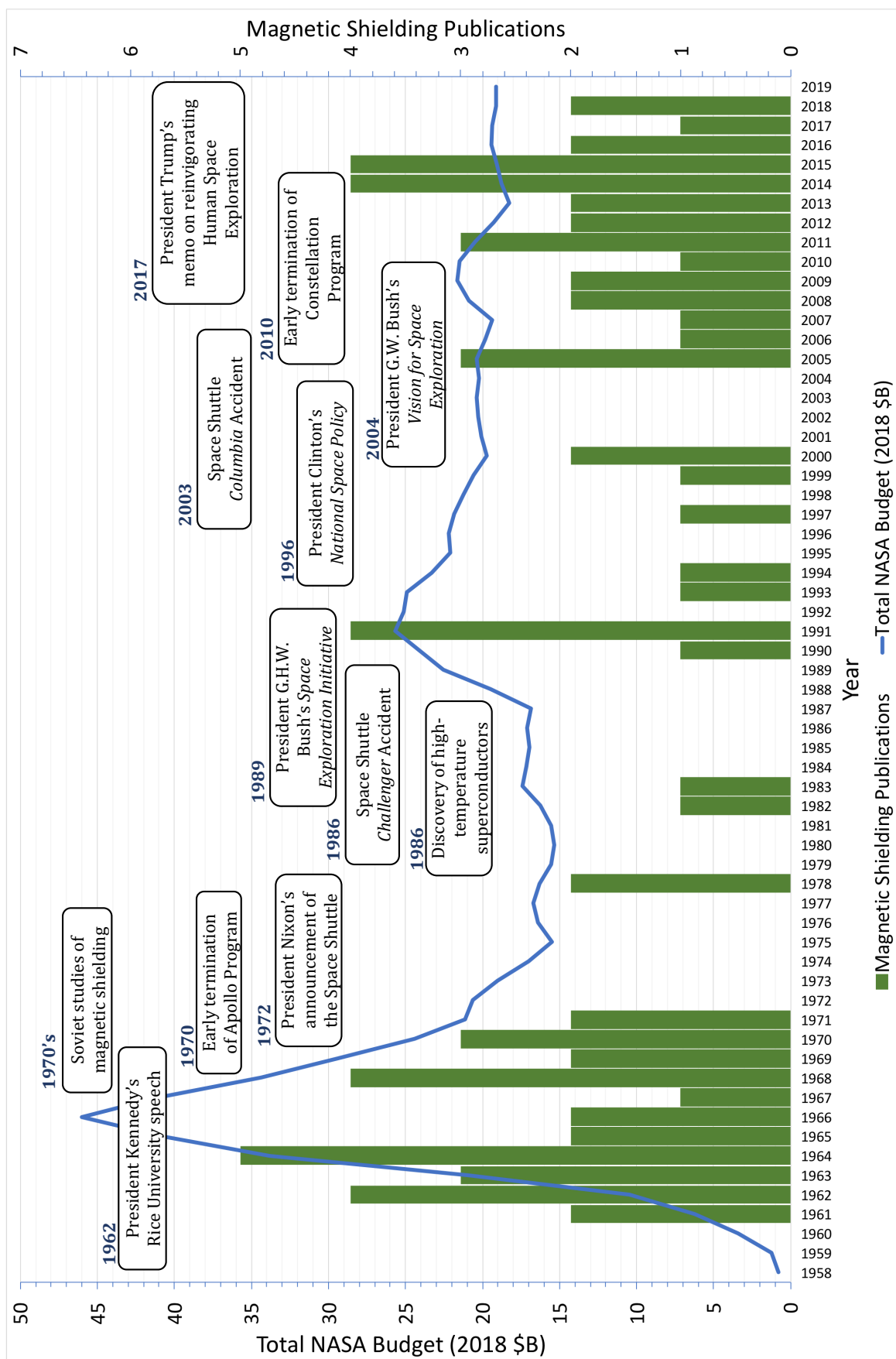


Figure 5: Timeline of Magnetic Shielding Publications with Contextual Information

### *1.3 Project Rationale*

As described in the previous section, active magnetic shielding has been discussed for several decades as a viable option reducing space radiation risk to astronauts, but there remain several gaps in the existing knowledge base on this topic. We strive to fill several of these gaps with the current project.

Previous studies on active magnetic shielding have used vastly different configurations of superconducting magnets as well as various radiation transport codes, space environment models, and spacecraft geometry (Geng et al. 2015; Vuolo et al. 2016; Spillantini 2010, 2011, 2014; Washburn et al. 2014, 2015; Westover 2014; Westover et al. 2012). Therefore, to date there has been no viable way to compare the results of these studies to determine their relative effectiveness. In this project, we compared several of the latest shielding configurations on a common platform to not just determine their effectiveness relative to each other but also relative to a passive water shielding case and a minimal shielding case. This analysis can assist space agencies in determining the benefits of specific types of shielding for a variety of mission scenarios and to prioritize technology development according to their specific needs.

In addition, previous studies on active magnetic shielding did not incorporate calculations for effective dose or a projection of lifetime cancer risks (Geng et al. 2015; Vuolo et al. 2016; Spillantini 2010, 2011, 2014; Washburn et al. 2014, 2015; Westover 2014; Westover et al. 2012). While the dose reduction data is valuable in determining the effectiveness of any shielding type, in order to assess the risk of radiation-induced cancer under current standards, an organ-based human phantom and effective dose calculation is required. In this project, we used a standard organ-based human phantom and international guidelines for effective dose calculation and cancer risk projection. The use of the organ-based phantom also allowed us to analyze the relative impact of organs and qualitatively rank them in order of importance. This analysis allows

space agencies to estimate the impact of the most important known risk of space radiation exposure: carcinogenesis. With this information, decision-makers can determine if current policies are adequate or too restrictive depending on the shielding available for a given mission.

Finally, no prior study on active magnetic shielding has yet completed a full analysis of the impact of astronaut age and sex on shielding effectiveness and safe days in space (Geng et al. 2015; Vuolo et al. 2016; Spillantini 2010, 2011, 2014; Washburn et al. 2014, 2015; Westover 2014; Westover et al. 2012). In this project, we used space agency standards for dose limits to determine the effectiveness of each shielding type based on astronaut age and sex. We also included the first safe days in space analysis for active magnetic shielding configurations as compared to the passive water shielding case and the minimal shielding case. This analysis allows space agencies to determine the appropriate shielding type and magnetic field for a given mission scenario and duration for a specific crew of astronauts based on age and sex. This, in turn, could allow decision-makers to tailor shielding, mission duration, or crew selection according to mission priorities rather than applying a general policy which could be overly conservative for certain crews or missions.

This project has filled gaps in knowledge on the topic of active magnetic shielding for human spaceflight. While there is still work to be done, particularly in the areas of engineering design and experimental validation through prototyping, the human spaceflight community may gather substantial insight on near-term technology development priorities by reviewing the results of this project.

## 2 Research Design

During interplanetary travel, the space radiation environment outside of Earth's geomagnetic shielding consists primarily of a combination of high energy heavy ions from galactic cosmic rays (GCRs) and moderate energy protons from solar particle events (SPEs). The chronic radiation dose due to GCRs on a typical Mars mission is on the order of 1 Sv (Letaw et al. 1989; Simonsen 2017). Solar activity is sporadic, but solar particle events could deliver an additional acute dose of up to 4 Sv (F. Cocks & Watkins 1993; English et al. 1973), which is a potentially lethal dose. Hence, radiation protection is a critical concern on these types of missions. Various methods of radiation shielding have been proposed, from simple passive shielding by materials such as water or aluminum, to active shielding systems comprised of electric and magnetic fields.

The concept of active magnetic shielding is to employ high-temperature superconducting coils to induce very high magnetic fields (on the order of 1-10 T) around the spacecraft. The induced magnetic field will deflect incoming charged particles, thereby reducing the particle fluence rate and radiation dose to astronauts behind the shield.

The goal of this project is to create a model for determining the value of active magnetic shielding in reducing radiation dose to astronauts on an interplanetary mission. **Our central hypothesis is that active magnetic shielding reduces effective radiation dose to astronauts on a Mars flyby mission by 50% versus no shielding.** This hypothesis will be investigated via the following specific aims:

**Specific Aim One:** Create a Monte Carlo model which includes the space radiation environment as a function of solar cycle and mission profile, a typical spacecraft architecture, passive and active shielding configurations, and an organ-based representation of an astronaut.

**Specific Aim Two:** Calculate space radiation-induced cancer risk to humans inside a typical spacecraft as a function of shielding type, magnetic field, astronaut age, and astronaut sex.

**Specific Aim Three:** Identify and calculate the factors with the greatest impact on effective dose and cancer risk to astronauts on interplanetary missions. Draw conclusions on the feasibility of various shielding configurations to mitigate risk for interplanetary travel.

## *2.1 Justification and Feasibility*

Although deterministic methods exist (Washburn et al. 2014), the Monte Carlo method was used in this study due to its suitability for radiation transport using a statistical approach to analyze random phenomena. The Monte Carlo schema for approaching physics problems includes formulating the problem mathematically, developing a statistical interpretation of the problem, creating an algorithm for sampling the distribution, estimating the uncertainty in parameters, optimizing simulation using variance reduction methods, and estimating the solution with a generated sample and associated uncertainties (Vassiliev 2017).

The underlying physics models and radiation transport code in the open-source Monte Carlo simulation toolkit, GEANT4 (Agostinelli et al. 2003), have been validated experimentally for space radiation applications (Ivantchenko et al. 2012; Tang & Smith 2010; Truscott et al. 2000), and GEANT4 is the only radiation transport code currently available that can simulate dynamic electromagnetic fields.

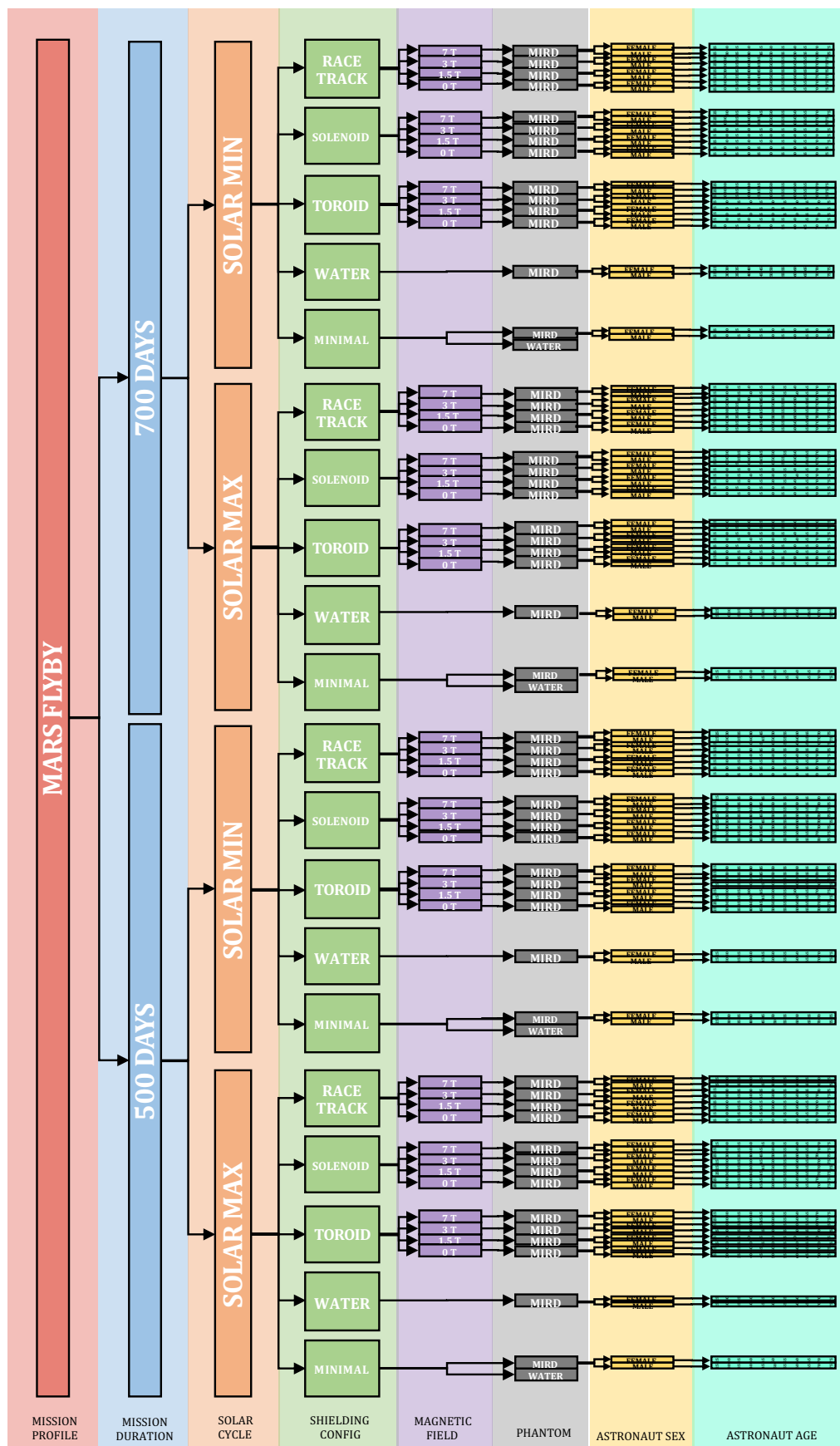
## 2.2 Trade Tree Development

In order to assess the scope of options for developing our assortment of Monte Carlo simulations in GEANT4, we developed a trade tree of parameters of interest (see Figure 6). The parameters we considered were mission profile, mission duration, solar activity, shielding configuration, magnetic field, phantom type, astronaut sex, and astronaut age. Each of these parameters is discussed in the following sections, including justification for our choices.

### 2.2.1 Mission Profile

NASA has considered human missions to Mars for decades (Portree 2001). The current strategy for human space exploration is published in the 2015 report entitled *NASA's Journey to Mars: Pioneering the Next Steps in Space Exploration* (NASA 2015a). In this report, NASA divides the mission strategy into three phases: Earth-reliant, proving ground, and Earth-Independent (see Figure 7).

In the *Earth-reliant phase*, NASA will continue to study human health on the International Space Station and develop technologies needed for deeper space missions, such as closed-loop environmental control and life support systems, advanced communications systems, and *in situ* resource utilization (NASA 2015a). This phase is planned for the near-term, through the retirement of the International Space Station in the 2020s (Simonsen 2017). Throughout this phase, the space radiation challenge will remain manageable, as astronauts will continue to live and work for in low Earth orbit (LEO) under protection of the Earth's magnetosphere, and missions will continue to be 6 months to 1 year in duration (NASA 2015a).



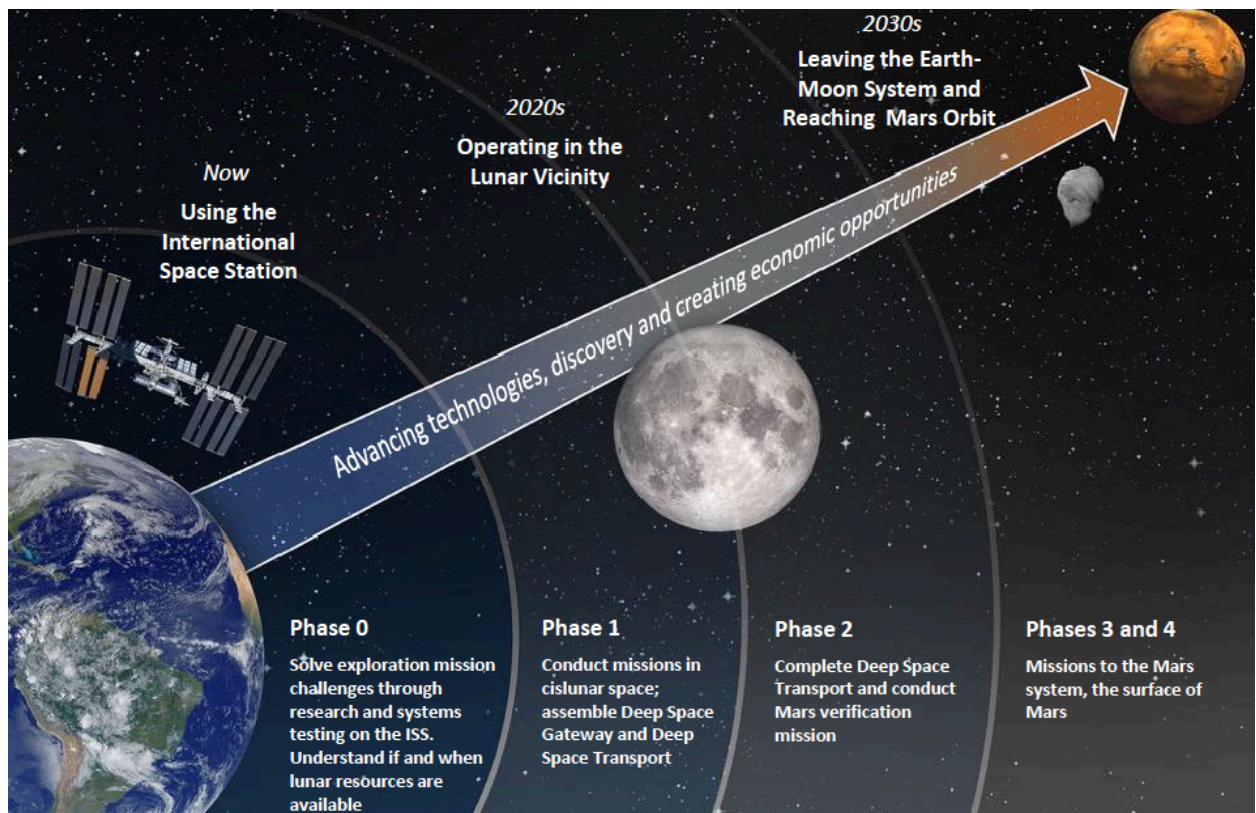


Figure 7: Summary of NASA's Journey to Mars (Simonsen 2017); courtesy of NASA

In the *proving ground phase*, NASA will launch missions to the lunar vicinity to test newly developed technologies and hardware needed for Mars missions, most importantly deep space habitat and robotics systems (NASA 2015a). This phase is planned for the 2020s (Simonsen 2017). Throughout this phase, astronauts will experience a harsher space radiation environment than in LEO, including the full strength of solar flares, thus the spacecraft will be outfitted to protect against medium-energy protons (~200–500 MeV). However, since missions are planned to be on the order of only 1-2 months in duration (NASA 2015a), doses from cosmic rays will remain well within permissible limits, and advanced shielding to protect against high energy heavy ions is not critical.

In the *Earth-independent phase*, NASA will conduct missions to interplanetary space, the Mars vicinity (NASA 2009), and ultimately the Martian surface (NASA 2015a). These missions

require humans and complex space hardware to function with minimal maintenance and resupply, with no simple emergency abort options. This phase is planned for the 2030s (Simonsen 2017). Throughout this phase, astronauts will experience the full spectrum of space radiation outside the protection of Earth's magnetosphere, and missions are planned for 2-3 years duration (NASA 2015a). Hence in this phase, advanced shielding techniques are most critical to protect astronauts from accumulated dose from high energy cosmic rays in addition to acute dose from solar flares.

Therefore, based on this current strategy for human space exploration, this study focused on space radiation protection in the Earth-independent phase, where astronauts are exposed to the full spectrum of space radiation for 2-3 years. While NASA's ultimate goal is to send humans to the Martian surface, several options exist there for adequate radiation protection, including pre-deployed shelters and regolith berm shielding. Hence this study focused on the interplanetary phase, where advanced shielding should have the greatest impact in reducing radiation dose and risk to the crew.

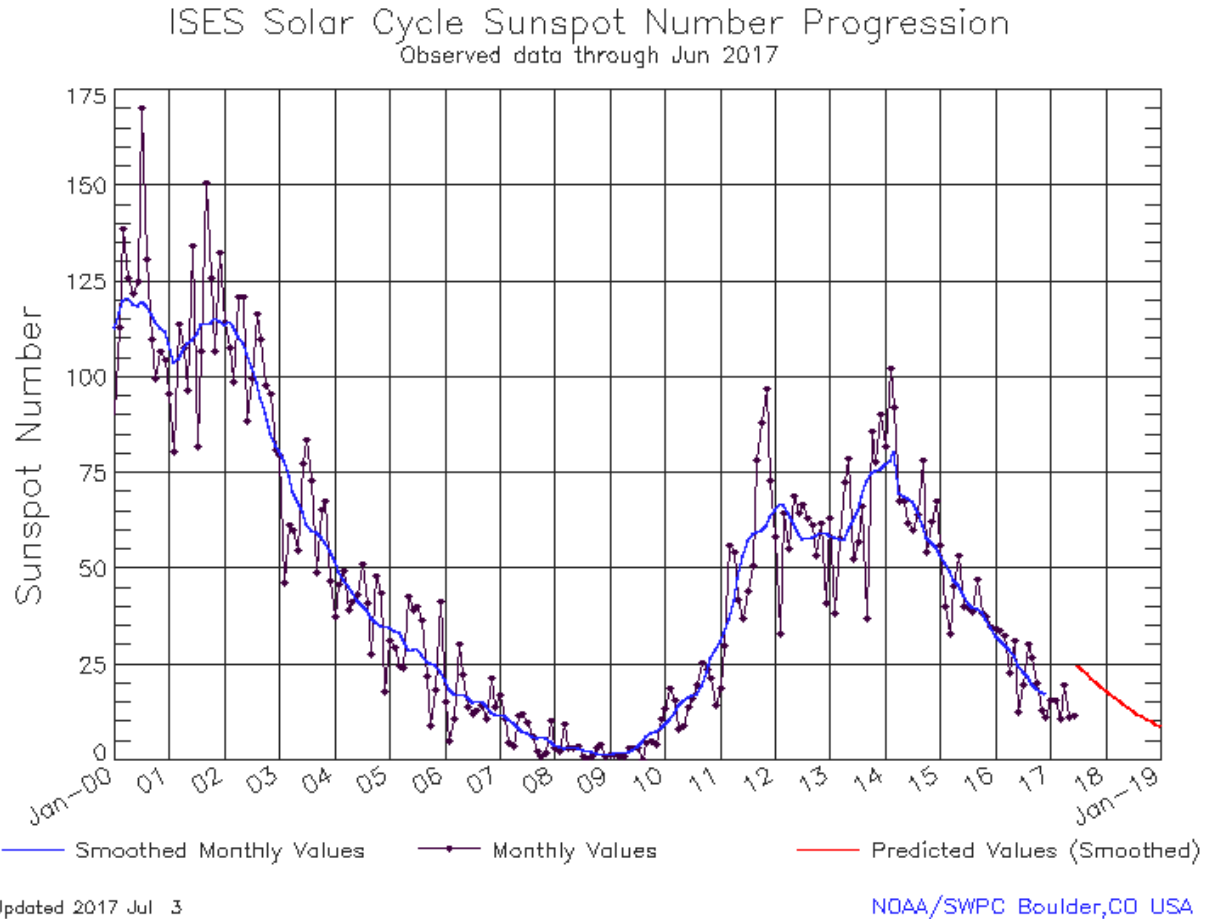
Thus, the mission profile chosen for this study was a Mars flyby mission; a mission that launches from Earth on a single launch vehicle on a direct trajectory to Mars with a slingshot around the planet and a free return to Earth. The duration for this mission type varies based on the positions of Earth and Mars in their respective orbits during the flight. Astrodynamics personnel at The Aerospace Corporation (El Segundo, California) calculated mission durations for Mars flyby launch opportunities in 2018 and 2021 to be approximately 500 and 600 days, respectively, but that more opportunities exist over the next few decades for missions of approximately 700 days (Radcliffe et al. 2016). In light of this analysis, the mission profile we chose to simulate in this study was a Mars flyby trajectory of 700 days. This selection provides simplicity in defining the space radiation environment, that is, it is unnecessary to consider the

complicating effects of the Martian environment or shielding provided by the planetary surface. Our selection also serves as a worst-case estimate for mission duration, so the results are conservative with respect to shorter mission durations. Later, we expanded our analysis to consider mission durations of both 500 and 700 days.

### *2.2.2 Solar Cycle Position*

The solar cycle is characterized by the changing magnetic field of the Sun. During periods of solar maximum activity (“solar max”), the solar magnetic field is strongest and more dark sunspots appear, representing magnetic field loops emerging from the surface. Whereas during periods of solar minimum activity (“solar min”), the solar magnetic field is weakest and several days may pass with zero sunspots visible. The number of visible sunspots (Wolf number,  $\bar{W}$ ) is tracked by space scientists and used as a measure of solar activity over time; a complete solar cycle is typically 11 years. Solar cycle historical data and predictions are published online by the National Oceanic and Atmospheric Administration (NOAA)’s Space Weather Prediction Center (<http://www.swpc.noaa.gov/>). Figure 8 shows the recent solar cycle progression.

The SPE and GCR spectra vary with the 11-year solar cycle, with SPEs occurring more frequently and at higher intensity at solar max while GCRs decrease by a factor of 2-10 (Ehresmann et al. 2016) at solar max due to Forbush modulation (Forbush 1969). The complex effects on the overall space radiation spectrum due to solar activity requires us to vary this parameter in our analysis. Thus, we ran simulations both with simulated solar max and solar min conditions to gauge whether there is a preferred window to conduct interplanetary missions.



**Figure 8: Solar Cycle Sunspot Number Progression (<http://www.swpc.noaa.gov/>); courtesy of NOAA**

### 2.2.3 Shielding Configurations

The traditional definition of shielding is, “an alteration of the radiations through interactions with intervening materials by which the intensity is decreased” (Wilson et al. 2001). There are hundreds of options for both passive and active shielding configurations.

Passive shielding works on the basis of providing a barrier material between the incident radiation and the astronaut inside the spacecraft. Hydrogenous materials (high hydrogen content and low atomic number) are found most efficient per unit mass for passive shielding against space radiation (Adams et al. 2005; Simonsen & Nealy 1991; Simonsen et al. 1990; Spillantini et al. 2007; Wilson et al. 1998, 2001). This is due to several factors including more nuclei for the

same shielding thickness, fewer neutrons in the nuclei resulting in fewer secondary neutrons, and smaller charge resulting in fewer secondary electrons and photons (Adams et al. 2005). Several passive shielding materials have been studied, including water, liquid hydrogen, polyethylene, methane, aluminum, metal hydrides, metal organic frameworks (MOFs), carbon composites, and “Z-graded” materials comprised of layers of varying density (Atwell et al. 2014; Barghouty & Thibeault 2006; Cucinotta et al. 2012a; De Angelis et al. 2004; Manning & Singleterry 2020; Rojdev et al. 2009; Singleterry 2013). Passive shielding materials that serve a mission function (such as drinking water, structural components, or fuel) are preferred for the mass savings associated with their multifunctionality (Durante 2014). Innovative passive shielding placement strategies have also been investigated, including deployable water wall shielding (Semones 2015a) and wearable garment shielding (Baiocco et al. 2018; Gaza et al. 2018; Milstein et al. 2015).

Active shielding works on the basis of deflecting charged particles away from the spacecraft with an electromagnetic field. Electrostatic, plasma, and magnetic shielding have been considered, but safety concerns with the large electric fields associated with electrostatic and plasma shields have limited their feasibility with current technology (Adams et al. 2005). Therefore, magnetic shielding is considered the most likely form of active shielding in the near- to medium-term.

Magnetic shielding works by taking advantage of the Lorentz force which acts via a magnetic field to change the direction of charge particle trajectories. The Lorentz force is defined by:

$$\vec{F} = q\vec{V} \times \vec{B}$$

where  $q$  is the net charge of the incident particle,  $\vec{v}$  is the velocity vector of the incident particle,  $\vec{B}$  is the vector magnetic field, and  $\vec{F}$  is the vector resultant force. The cross-product results in particle motion perpendicular to the magnetic field and typically manifested as a bending of the charged particle trajectory through the magnetic field.

High-temperature superconducting (HTS) materials are found most efficient per unit mass for active shielding against space radiation (Battiston et al. 2011). This is due to several factors including a higher magnetic field per unit mass than intermediate- or low-temperature superconducting magnets, fixed magnets, electrostatic fields, or plasma fields. HTS operating in the 70+ K region are also preferred due to the decreased need for complex cryogenic cooling systems. Several high-temperature superconducting materials have been studied for active shielding applications, including Yttrium-Barium-Copper-Oxide (YBCO), Bismuth Lead Strontium Calcium Copper Oxide (BSCC), and Magnesium Diboride ( $\text{MgB}_2$ ) (Battiston et al. 2013; Kervendal et al. 2009; Nose et al. 1990). Classical superconducting materials, including Niobium-Titanium (NbTi) and Niobium-Tin (NbSn) have also been analyzed for their applicability for active magnetic shielding (Cougnet et al. 2005).

Based on the literature and current space agency objectives, we narrowed the field of options to five selections: minimal shielding (spacecraft only), passive shielding (water), and toroidal, solenoidal, or race track active magnetic shielding. The minimal and passive shielding options served as our control cases; hence we did not consider advanced passive shielding materials. We chose active magnetic shielding options that have shown the most promise over the last decade, have national or international space agency support, and are relatively simple designs. These configurations are described in more detail in the Sections 2.3.3.3, 2.3.3.4, and 2.3.3.5.

#### *2.2.4 Magnetic Field*

The ability of a magnetic shield to deflect incoming space radiation particles depends highly upon the primary magnetic field. While it is clear that a higher magnetic field is more effective at reducing radiation dose to astronauts on an interplanetary mission, higher magnetic fields require larger masses of superconducting coils, mechanical support infrastructure, power supplies, and cooling equipment. Therefore, we performed simulations at several magnetic fields to search for an optimal cost-benefit ratio.

We chose 0, 1.5, 3, and 7 T magnetic fields for several reasons. First, running a simulation with the magnetic field switched off (0 T) allowed us to separate the passive shielding effect of the magnetic shielding structures from the active shielding effect of the applied magnetic fields. Next, 1.5, 3, and 7 T are the typical magnetic fields used in magnetic resonance imaging (MRI) magnets, so we know that the technology currently exists to produce fields of these magnitudes. Second, several studies have indicated that magnetic fields between 1 and 8 T are effective in deflecting a high percentage of space radiation particles (Geng et al. 2015; Singleterry et al. 2015; Vuolo et al. 2016; Westover 2014). Finally, magnetic fields of this magnitude have relatively small fringe fields that could potentially reach the habitable volume of our spacecraft. Concerns regarding human exposure to magnetic fields are described below.

##### *2.2.4.1 Safety of Human Exposure to Static Magnetic Fields*

While every effort was made to minimize the residual magnetic field inside the habitable volume of a spacecraft equipped with magnetic shielding, it is possible that the interior magnetic fringe field may be nonzero. To place an upper boundary on the allowable interior magnetic fringe field, we looked to studies on the safety of MRI, which have offered insight into the effects of magnetic fields on the human body.

While the gradient and radiofrequency (RF) fields associated with MRI have been shown to cause peripheral nerve stimulation and tissue heating during exposure (Formica & Silvestri 2004), there is no clear evidence of adverse health effects from extended exposure to *static* magnetic fields (Schenck 2000) such as astronauts could be exposed to as fringe field from magnetic shielding.

Also, while it has been shown in pre-clinical trials that motion within a static magnetic field can induce a small current in the blood volume (Kinouchi et al. 1996), at fields as high as 8 T this current does not appear strong enough to create deleterious effects to the body systems (Togawa et al. 1967; World Health Organization 2006).

There is also evidence of a static magnetic field effect on electronic spin states during photochemical and organic chemistry reactions involving free radicals (McLauchlan 1981), but biological significance of these effects has not been reported (Schenck 2000).

Finally, there is anecdotal evidence of transient sensory effects, including vertigo and nausea, during voluntary motion in a strong magnetic field (de Vocht et al. 2006; Schenck et al. 1992; World Health Organization 2006); these effects are not considered harmful and quickly dissipate upon exiting the field. However, according to the World Health Organization's 2006 report on static magnetic field exposure, "Together with possible effects on eye-hand coordination, the optimal performance of workers executing delicate procedures...could be reduced [during exposure to static magnetic fields], with a concomitant impact on safety" (World Health Organization 2006); therefore, exposure to high static magnetic fields could affect astronauts' ability to perform complex mission tasks. Further, any compounding effects introduced by the microgravity environment during a space mission are not yet known.

#### *2.2.4.2 Magnetic Fringe Field Limits*

Given the lack of clear evidence regarding the effects to human systems due to prolonged static magnetic field exposure, the lack of information on possible compounding effects of the microgravity environment, and indications that astronaut performance and mission success could potentially be compromised during prolonged static magnetic field exposure, the upper limit for interior magnetic fringe fields for this study was set conservatively according to industry standards.

The International Commission on Non-Ionizing Radiation Protection (ICNIRP)'s 1994 report provides an occupational limit of 200 mT (time-weighted average for entire work period) for exposure to static magnetic fields. The corresponding continuous exposure limit for the general population is 40 mT (ICNIRP 1994). Thus, for this study we limited magnetic fringe fields inside the spacecraft to 40 mT, not to exceed 200 mT.

#### *2.2.5 Phantom Type*

A critical step in developing any radiation protection solution is choosing how to best model the human body (ICRU 1992; Xu & Eckerman 2010) to perform dosimetry to analyze radiation risk and establish limits (Adams 1992; NCRP 1993, 2000; Peterson & Kovyrshina 2015; Swenberg et al. 1993).

Historically, the term “phantom” was used to describe a physical representation of the human body for radiation protection testing. However, in recent decades, computational phantoms have been developed, and the term “phantom” can apply to either the physical or computational variety (Xu & Eckerman 2010). Over 100 computational phantoms have been developed since the 1960s; many of these seek to represent specific patient types such as infants, children, and pregnant women, where others seek to improve the accuracy of dosimetric estimates (ICRU

1992; Xu & Eckerman 2010). The broad categories of computational phantoms include basic, stylized, and voxelized/mesh.

Basic phantoms include water (or solid water) containers of various dimensions, relying on the fact that the human body is comprised primarily of water. These phantoms have the advantage of being inexpensive, simple to use, and simple to model. However, it is clear that a homogeneous water phantom is not an accurate depiction of the complex heterogeneous human body. Therefore, water phantoms should be used where appropriate, but when under more stringent requirements, stylized or voxelized phantoms should be used to model the human body.

Stylized phantoms typically model the human body as a series of solid volumes representing the major organs and structures. These phantoms have the advantages of applying unique materials and densities to different organs and enabling the calculation of effective dose which is most closely tied to induced cancer risk. However, these phantoms are more difficult to model, require more complex code, and consume more computational resources than basic phantoms do. Because of this, stylized phantoms should be used when requirements dictate a higher fidelity model, while the use of basic phantoms is appropriate in less complex cases for the sake of efficiency.

Voxelized or mesh phantoms are based on imaging of actual human patients, either via CT or MRI. The image slices from these 3D imaging techniques are then compiled, and the organ volumes and regions of the body are built as a parameterization of the voxel volumes. These phantoms have the advantages of being based on real anatomy and can incorporate details such as heart rate, breathing, and pathology (Segars & Tsui 2009). However, these phantoms are still the representation of a very small number of actual patients. Thus, the applicability of the voxelized phantom to the general population is not much higher than the applicability of a stylized

phantom, due to the individual differences between each person's anatomy. Also, these phantoms are much more difficult to model, require even more complex code, and consume additional computational resources. Because of this, voxelized or mesh phantoms should be used when requirements dictate the need for special cases such as heart rate, breathing, or pathology, and stylized or basic phantoms should be used as appropriate in more generic cases for the sake of efficiency.

For this study, the human astronaut was modeled as either a basic water phantom or the male or female adult version of the stylized MIRD (medical internal radiation dose) phantom based on the ICRP Reference Man (Cristy & Eckerman 1987a, 1987b; ICRP 1975; Snyder et al. 1974, 1978; Xu & Eckerman 2010). The geometry of organ volumes for the MIRD phantom were modified from a GEANT4 example (/examples/advanced/human\_phantom).

We also considering using NASA stylized phantoms, the Computerized Anatomical Man (CAM) and Computerized Anatomical Female (CAF) (Billings & Yucker 1973; Atwell et al. 1996; Slaba et al. 2009), as well as CT-based voxelized phantoms, including the MAX/FAX or MASH/FASH (Cassola et al. 2010; Kramer et al. 2003, 2004) and the XCAT (Belley et al. 2014) phantoms. However, these phantoms proved to be quite complex to implement in GEANT4 and did not provide enough extra value to our analysis to justify the additional effort.

#### *2.2.6 Astronaut Sex (Male/Female)*

Several studies show sex differences in carcinogenesis from radiation exposure (Cucinotta et al. 2013). Specifically, lung cancer risk appears to be significantly higher for females than males receiving equal doses of radiation, even for never-smokers (Sun et al. 2007; Zang & Wynder 1996). Also, females are susceptible to additional cancers over males due to having more sex-specific radiosensitive organs (breasts, uterus, ovaries, cervix) than males (testes,

prostate). Because of these and other differences, the effective dose limits for females are set lower than for males of the same age.

For this study, we compared total effective dose to both male and female astronauts as represented by a traditional stylized phantom, ICRP Reference Man (aka ORNL or MIRD phantom) (Cristy & Eckerman 1987a, 1987b; ICRP 1975; Snyder et al. 1974, 1978; Xu & Eckerman 2010), as well as a standard water phantom.

### 2.2.7 Astronaut Age

In NASA's *Evidence Report: Risk of Radiation Carcinogenesis*, age at radiation exposure is considered to be a significant factor in increasing cancer risk (Huff et al. 2016; UNSCEAR 2008; Cucinotta et al. 2013). Averaging the career effective dose limits from NASA and NCRP (Cucinotta et al. 2012b; Huff et al. 2017; Kim et al. 2011; NASA 2015b; NCRP 2000; Semones 2015b) and BEIR VII (NRC 2006) across both sexes gives annual total body effective dose limits of between 0.5 and 2.2 Sv, depending on age and mission duration. The limits in Table 1 represent the lower of the annual dose limit extrapolated to 500 or 700 days or the career dose limit. These values are accepted to represent a 3% increase in lifetime risk of fatal cancer at a 95% confidence interval based on a linear-non-threshold dose conversion. In this study, we compared results for each simulation scenario to the limits for 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, and 75-year-old astronauts to determine whether certain mission profiles are permissible for older astronauts but not younger ones. The limits and results for 65, 70, and 75-year-old astronauts are extrapolated because the regulatory guidance only includes ages up to 60 years.

Table 1: BEIR VII and NASA/NCRP Effective Dose Limits

Crew Age (years)	500 day Limit		700 day Limit	
	Female (Sv)	Male (Sv)	Female (Sv)	Male (Sv)
25	0.5	0.8	0.5	0.8
30	0.7	1.0753	0.7	1.1
35	0.8556	1.113	0.9	1.4
40	0.8836	1.1507	1.1	1.611
45	0.9418	1.2021	1.3	1.68
50	1	1.2534	1.4	1.7548
55	1.0856	1.3493	1.5199	1.889
60	1.1712	1.4452	1.6397	2.0233

### 2.3 Simulations

According to the schema shown in Figure 9, the key elements of our Monte Carlo simulations in GEANT4 included models of the space environment, spacecraft architecture, shielding configurations, human phantoms, and dosimetry methodology. The design of each of these elements with a description of major assumptions is provided in the sections below.

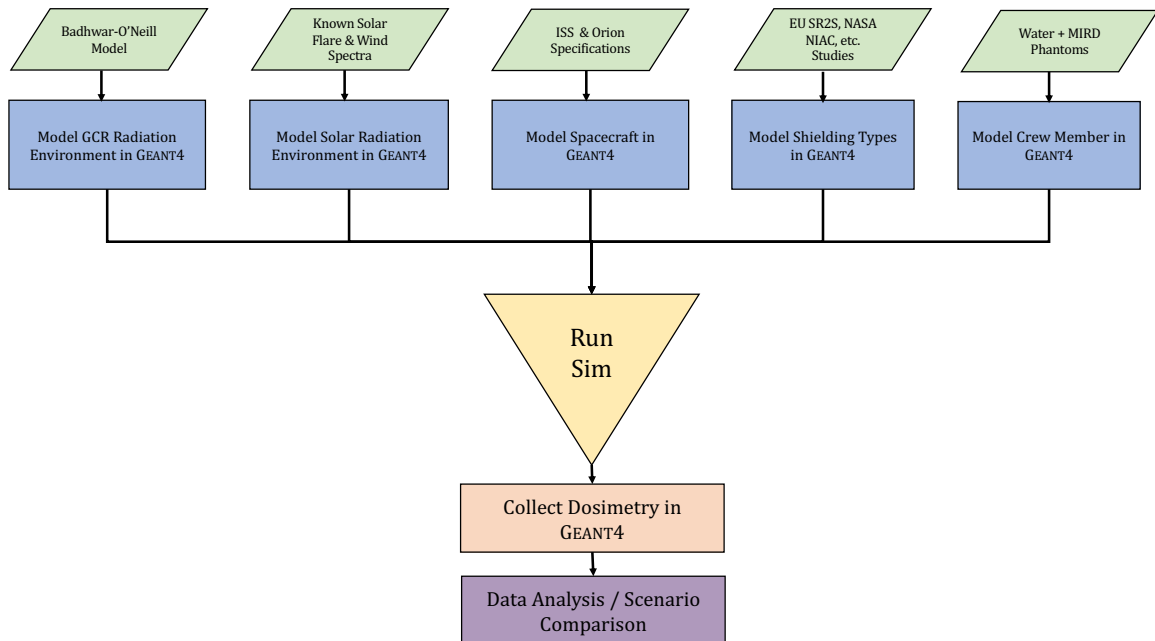


Figure 9: Schema for Monte Carlo Simulations in GEANT4

### 2.3.1 Space Environment

Any radiation protection question begins with defining the source of the hazard. For this case, the hazard was the interplanetary space radiation environment, particularly solar wind protons, solar particle events (SPE) protons, and galactic cosmic rays (GCRs). The flux density vs. particle energy spectra for these particles (Cucinotta et al. 2013) are highlighted in Figure 10. As discussed previously, these spectra vary with the 11-year solar cycle with SPEs occurring more frequently and at higher intensity at solar max but GCRs are modulated by the strong solar magnetic field at solar max. Therefore, we ran separate simulations for periods of solar max and solar min conditions.

We assumed the space radiation spectra are constant between Earth (1 AU) and Mars (1.524 AU), because studies based on spacecraft sensor data show the variation is less than 5% (Gieseler et al. 2007). This selection ensures our results are conservative.

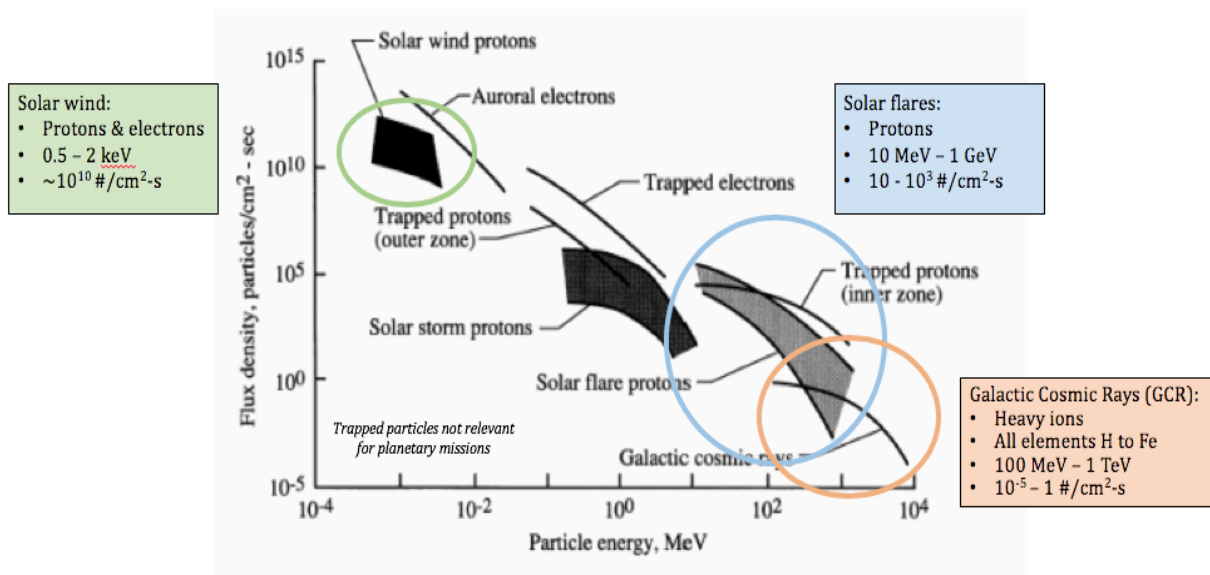


Figure 10: Space Radiation Spectrum (modified from Cucinotta et al. 2013); courtesy of NASA

Simulation of the space radiation environment in GEANT4 requires stratification of the continuous spectra into a discrete distribution and then initiating or starting quantities of particles

at specific mass, charge, and energy according to the contents of each bin of the discrete distribution. Below is a description of how we simulated each type of space radiation.

#### *2.3.1.1 Solar Wind*

The solar wind is the manifestation of the expanding solar atmosphere and is a low energy plasma of mostly protons and electrons. At solar max, the solar wind travels between 400 and 800 km/s at a fluence rate of  $10^{10} - 10^{12}$  particles/cm<sup>2</sup>/s at 1 AU. In contrast, at solar min, the solar wind travels between 250 and 400 km/s at a fluence rate of  $10^{10} - 10^{11}$  particles/cm<sup>2</sup>/s at a distance of 1 AU (Kamide & Chian 2007; Roberts 2011). Protons in this velocity range have kinetic energy on the order of keV, and electrons in this velocity range have kinetic energy on the order of only eV. Both of these particle types are of negligible consequence at distance of 1 AU or greater with minimal shielding. Thus, we ran only verification simulations with protons and electrons of these energies to ensure negligible dose was delivered to our phantom under these conditions.

#### *2.3.1.2 Solar Particle Events (SPEs)*

A solar flare is the area of substantially increased solar emission surrounding a sunspot. The increased emissions paired with a shock wave from a coronal mass ejection (CME) causes most of the observed solar particle events (SPEs) near Earth (Cucinotta et al. 2013). SPEs at Earth are characterized by a time delay after a solar flare for propagation of the particles to a distance of 1 AU, increasing fluence rate over a period of hours to days to a maximum, and finally a slow decay back to background over a period of days (Shea & Smart 1990). At solar max, SPEs occur at a rate of about 10-30 per calendar year or approximately two per month, and at solar min, SPEs occur at a rate of 0-4 per calendar year or approximately one every 6 months (Shea & Smart 1990). Particularly intense SPEs were measured in November 1960, August 1972, and October 1989, with the fluence of protons above 30 MeV of at least  $4 \times 10^9$  protons/cm<sup>2</sup> (Cucinotta

et al. 2013). NASA considered SPEs to be significant in terms of astronaut health for protons of 30 MeV and higher at a total fluence of  $10^8$  protons/cm<sup>2</sup> at a distance of 1 Astronomical Unit (AU) from the sun (Cucinotta et al. 2013).

To simulate SPEs at solar max, we initiated 20 average solar flares per year with a distribution of 0.6 10 MeV protons, 0.39 30 MeV protons, 0.01 100 MeV protons, and  $1 \times 10^{-6}$  250 MeV protons with a uniform fluence of  $10^8$  particles/cm<sup>2</sup> at 1 AU (Adams et al. 2011). Also, we initiated one “worst-case” solar flare once per mission with a distribution of 0.6 10 MeV protons, 0.39 30 MeV protons, 0.01 100 MeV protons,  $1 \times 10^{-5}$  250 MeV protons, and  $1 \times 10^{-9}$  500 MeV protons with a uniform fluence of  $10^9$  particles/cm<sup>2</sup> at 1 AU. See Figure 11 for the simulated spectra for both average and worst-case solar flares. Therefore, for a typical 700-day mission, 38 average SPEs and 1 worst-case SPE were used in our simulations for solar max.

To simulate SPEs at solar min, we initiated 2 average solar flares per year with a distribution of 0.6 10 MeV protons, 0.30 MeV protons, 0.01 100 MeV protons, and  $1 \times 10^{-6}$  250 MeV protons with a uniform fluence of  $10^8$  particles/cm<sup>2</sup> at 1 AU. See Figure 11 for the simulated spectra for average solar flares. Therefore, for a typical 700-day mission, 4 average SPEs were used in our simulations for solar min.

#### 2.3.1.3 *Galactic Cosmic Rays (GCRs)*

Galactic Cosmic Rays (GCRs) are high energy, high atomic number ions completely stripped of electrons. GCRs permeate interplanetary space isotropically from outside the solar system. Most GCRs are believed to originate from within the local galaxy from supernova explosions, though the origin of extremely high energy GCRs still remains unknown (Ginzburg 1996). GCRs are made up of approximately 90% protons, 5% alpha particles, and 5% heavier nuclei up to iron in the energy range of  $10^8$  to  $10^{10}$  eV/nucleon. At solar max, GCR spectra are hardened by the

higher solar magnetic field via Forbush modulation (Forbush 1969) by approximately a factor of 2-10.

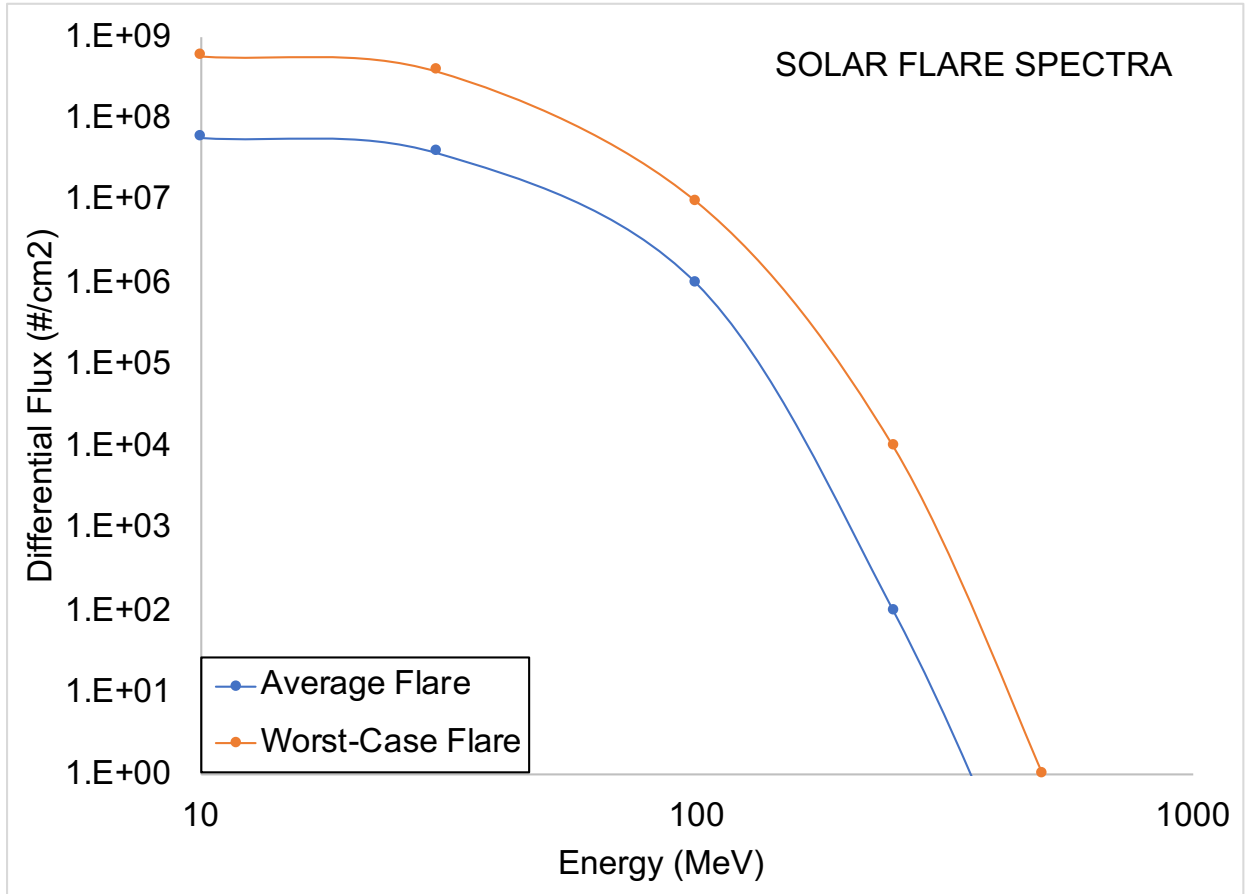


Figure 11: Simulated Solar Flare Spectra

Badhwar and O'Neill developed a representation for the local interstellar spectrum (LIS) of GCRs based on data collected from spacecraft sensors as a function of kinetic energy/nucleon (E) given by (NASA 2013; O'Neill 2006):

$$LIS(E) = \frac{j_0 \beta^\delta}{(E + E_0)^\gamma}$$

where:

$j_0$ ,  $\delta$ ,  $\gamma$  = free parameters determined by experimental data

$\beta$  = particle velocity relative to the speed of light

$E_0$  = rest energy per nucleon of an ion, ~938 MeV/n

Using this method, the root mean square (RMS) error for each component of the GCR composition is less than 10% (Cucinotta et al. 2013) as compared to spacecraft sensor data. This model was updated in 2014, with the new LIS function given by (Golge et al. 2015; O'Neill et al. 2015):

$$LIS(Z, T) = j_0(Z)(T_N + E_0)^{\gamma(Z)} \beta_N^{-1} \beta^{\delta(Z)} (T + E_0)^{-\gamma(Z)}$$

where:

$j_0$ ,  $\delta$ , and  $\gamma$  = free parameters determined by experimental data

$\beta$  = particle velocity relative to the speed of light

$T$  = kinetic energy per nucleon of the particle (MeV/n)

$E_0$  = the rest energy per nucleon of an ion, ~938 MeV/n

$\beta_N$  = relative velocity at  $T_N = 35$  GeV/n

Note the International Organization for Standardization (ISO) also published a standard GCR model in 2004 (ISO 2004), which has been used by research groups outside the United States. We chose the Badhwar O'Neill (BON) 2014 model for our simulations because this is the NASA standard.

Published GCR spectra based on the BON 2014 model and spacecraft measurements are provided for reference; see Figures 12 through 14.

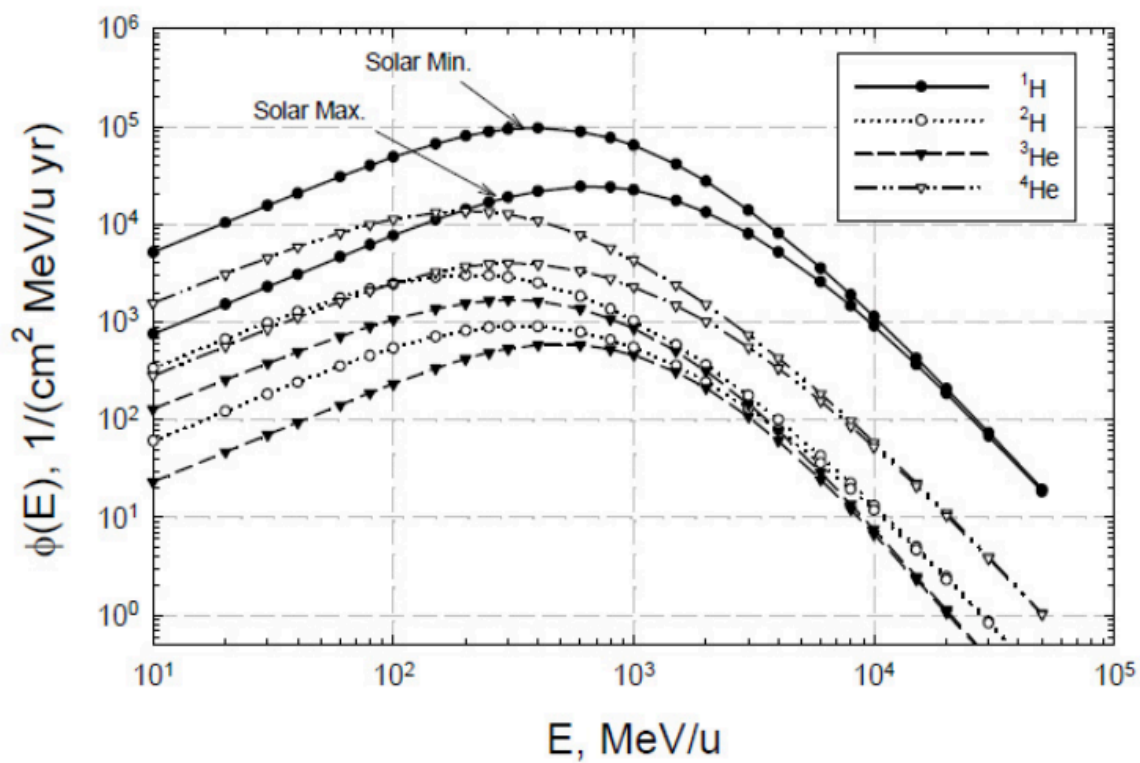


Figure 12: Spectra Near Solar Min/Max for Primary GCRs (Cucinotta et al. 2013); courtesy of NASA

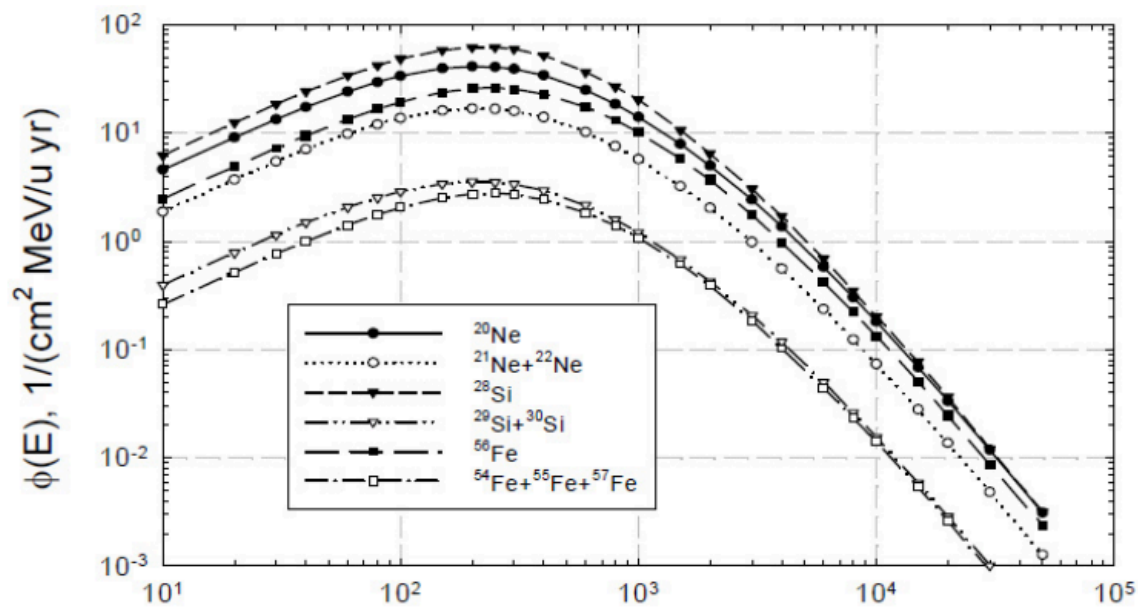


Figure 13: Spectra Near Solar Min/Max for Primary GCR Isotopes (Cucinotta et al. 2013);

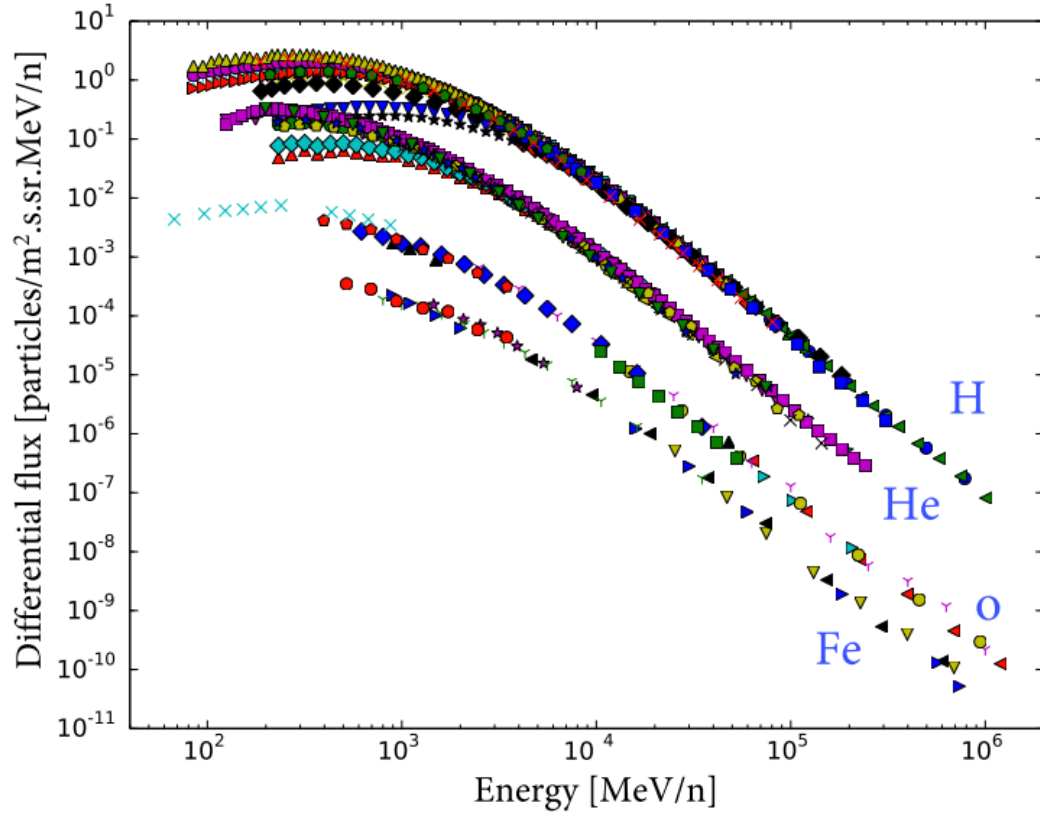


Figure 14: GCR Data Collected by Spacecraft (O'Neill et al. 2015); courtesy of NASA

To simulate GCRs at solar max, we initiated proton (H), alpha (He), lithium (Li), beryllium (Be), boron (B), carbon (C), nitrogen (N), oxygen (O), fluorine (F), neon (Ne), sodium (Na), magnesium (Mg), aluminum (Al), silicon (Si), phosphorous (P), calcium (Ca), scandium (Sc), titanium (Ti), vanadium (V), chromium (Cr), manganese (Mn), and iron (Fe) ions at energy/nucleon of 10 MeV, 100 MeV, 1 GeV, 10 GeV, 100 GeV, and 1 TeV using the BON 2014 model described above to determine the differential fluence rate, reducing the differential fluence rate of particles of  $\leq 100$  MeV/nucleon by a factor of 2-10 to account for Forbush modulation, as shown in Figure 15. We then integrated the differential fluence rate ( $\Phi$ ) over energy (E), surface area (A), and mission duration (t) to determine the total number of particles (N) of each type to initiate for each simulation run.

$$N_{max} = \iiint \Phi dE dA dt$$

To simulate GCRs at solar min, we initiated proton (H), alpha (He), lithium (Li), beryllium (Be), boron (B), carbon (C), nitrogen (N), oxygen (O), fluorine (F), neon (Ne), sodium (Na) magnesium (Mg), aluminum (Al), silicon (Si), phosphorous (P), calcium (Ca), scandium (Sc), titanium (Ti), vanadium (V), chromium (Cr), manganese (Mn), and iron (Fe) ions at energy/nucleon of 10 MeV, 100 MeV, 1 GeV, 10 GeV, 100 GeV, and 1 TeV using the BON 2014 model described above to determine the differential fluence rate, as shown in Figure 16. We then integrated the differential fluence rate ( $\Phi$ ) over energy (E), surface area (A), and mission duration (t) to determine the total number of particles (N) of each type to initiate for each simulation run.

$$N_{min} = \iiint \Phi dE dA dt$$

For surface area (A), we chose the simulation world volume to be a sphere of radius 40 m with a surface area of  $2.011 \times 10^8 \text{ cm}^2$ . Selecting this volume for the surface area allowed us to calculate a total number of particles (N) emanating from the world sphere regardless if the particles strike the spacecraft. If we had instead chosen the spacecraft volume for calculating the surface area, we would have to ensure that all initiated particles strike the spacecraft volume for accurate accounting. However, we were interested in modeling particles passing just outside the spacecraft in the event that they might be redirected towards the spacecraft by our magnetic shielding configurations. We were also interested in modeling particles passing through the spacecraft air volume but not striking the astronaut directly to ensure we simulated dose from secondary particles. Thus, we chose to model a range of particle incidence angles ( $\pm 5^\circ$  from a line passing through the center of the sphere).

Once we determined the relative fluence rate ( $\Phi$ ) of each particle and energy species using the BON 2014 model and then integrated over the world volume surface area (A), particle

energies ( $E$ ), and elapsed time ( $t$ ), we obtained a list of the total number of each species and energy of GCR particle to initiate. We used these values to assign a relative intensity weighting to each ion species in GEANT4 via source macros (see Appendix A1-A6). Due to the complexity of modeling the GCR environment, we ran test simulations to compare our input spectra to the GEANT4 output. The results of these tests were that the GEANT4 output agreed very well with the input spectra (see example in Figure 17), giving us confidence in accuracy of our model of the space radiation environment.

Because the GCR spectrum is dominated by Hydrogen ( $Z=1$ ) and Helium ( $Z=2$ ) ions (90% and 5% of the integral spectrum, respectively), the number of total particles required to produce enough of the other GCR ions ( $Z=3$  through  $Z=26$ ) for each simulation becomes unmanageable. In order to manage this, we separated out Hydrogen and Helium from the overall GCR spectrum and run those ions separately for each scenario. This allowed us to run a much smaller total number of particles and still obtain good statistics for each ion species.

Note: Van Allen belt radiation is often noted as a substantial space radiation hazard for human spaceflight missions. The Van Allen belts consist mostly of protons and electrons trapped by Earth's magnetic field into two bands below an altitude of 40,000 km (Van Allen et al. 1958). For most interplanetary mission trajectories, Van Allen belt radiation is only a concern during the first few hours after launch. Once the spacecraft reaches an altitude of 40,000 km, Van Allen belt radiation becomes negligible, leaving GCR and SPE sources as the greatest radiation concerns for the majority of the mission. However, if the selected mission architecture requires on-orbit staging prior to Earth departure, Van Allen belt proton and electron intensities should be considered in the selection of optimal staging orbits and dwell times.

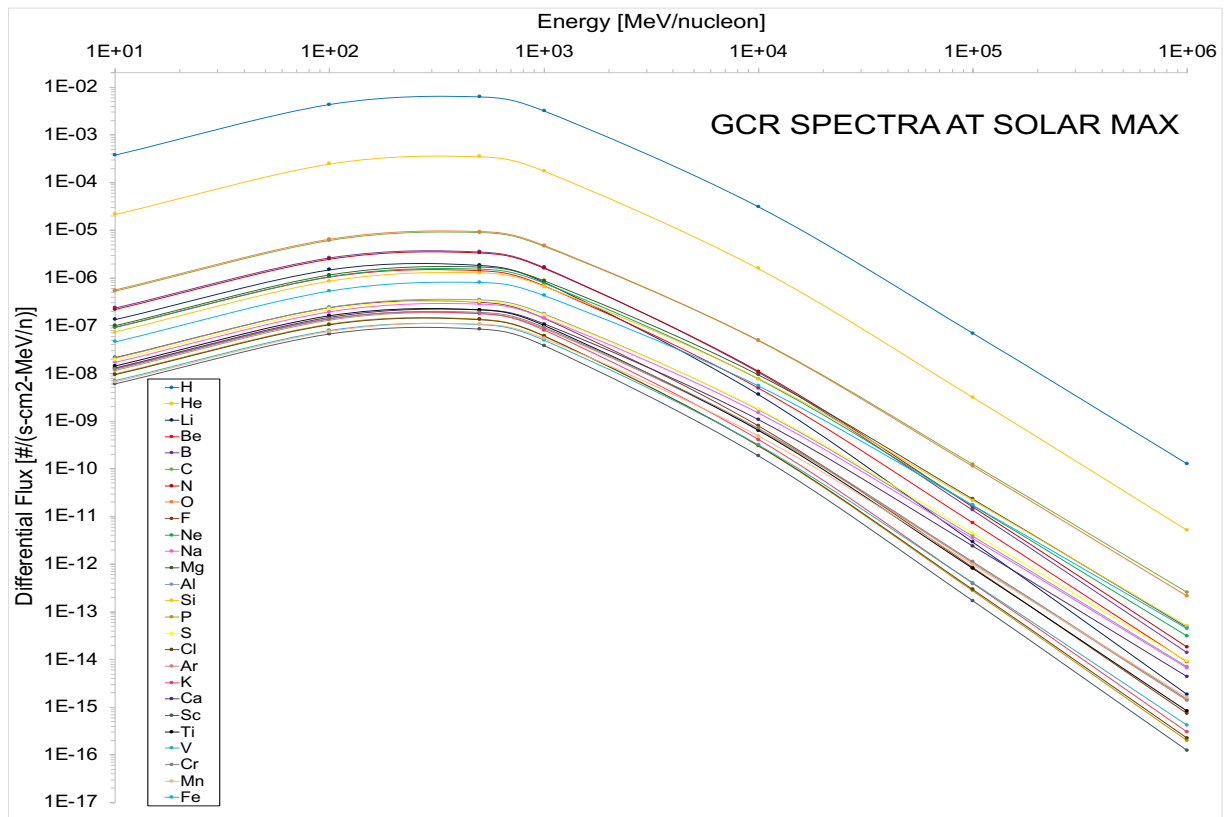


Figure 15: Simulated GCR Spectra at Solar Max

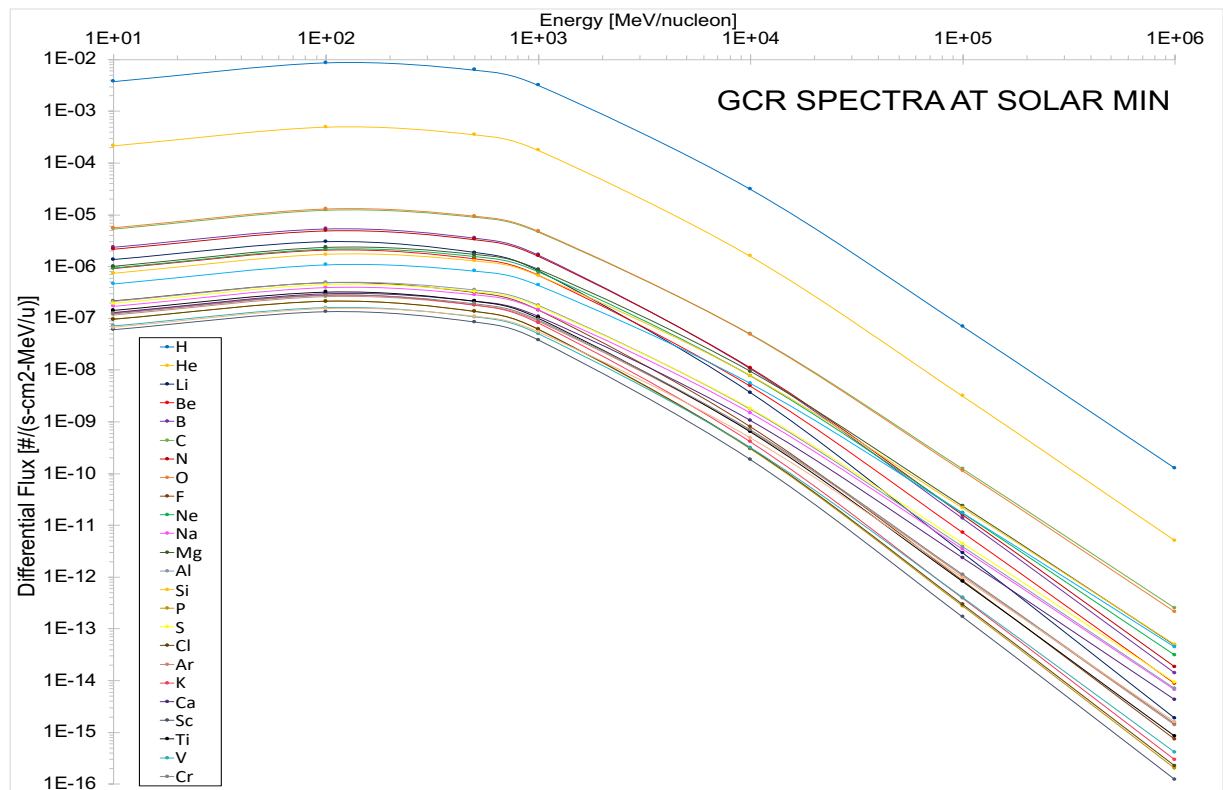


Figure 16: Simulated GCR Spectra at Solar Min

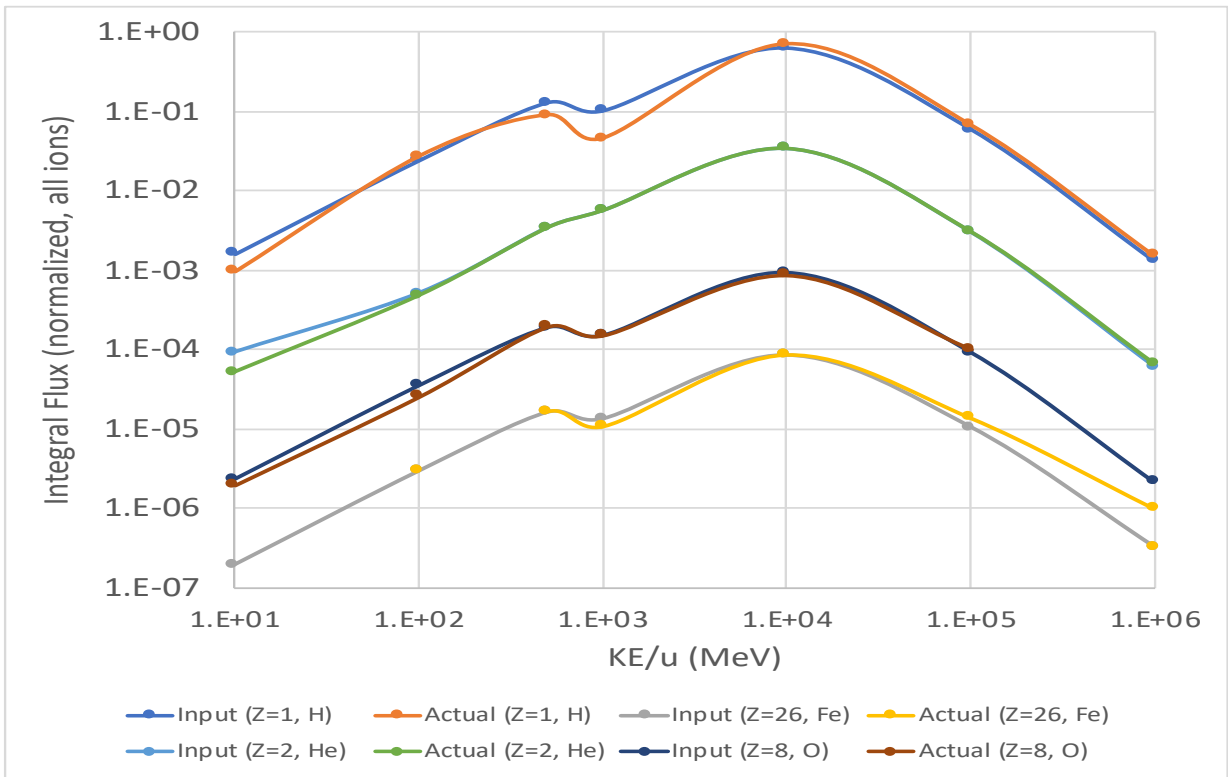


Figure 17: GEANT4 Actual vs. Input Spectra

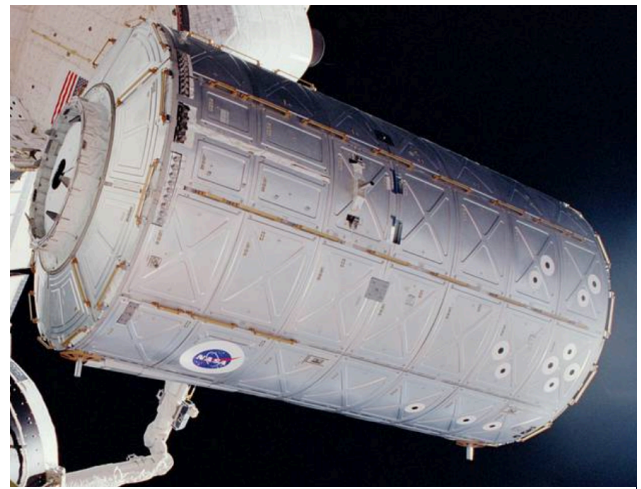
### 2.3.2 Spacecraft Architecture

Based on the results of several recent studies (NASA 2009; Radcliffe et al. 2016), the spacecraft configuration for an interplanetary mission will likely include a habitat module and a crew capsule for Earth re-entry. Additional modules including inflatables, descent modules, landers, etc. could possibly be included depending on the mission objectives, but these designs vary widely between architectures. Thus, we chose to include only a habitat and crew return capsule as our baseline spacecraft architecture.

#### 2.3.2.1 Habitat Module

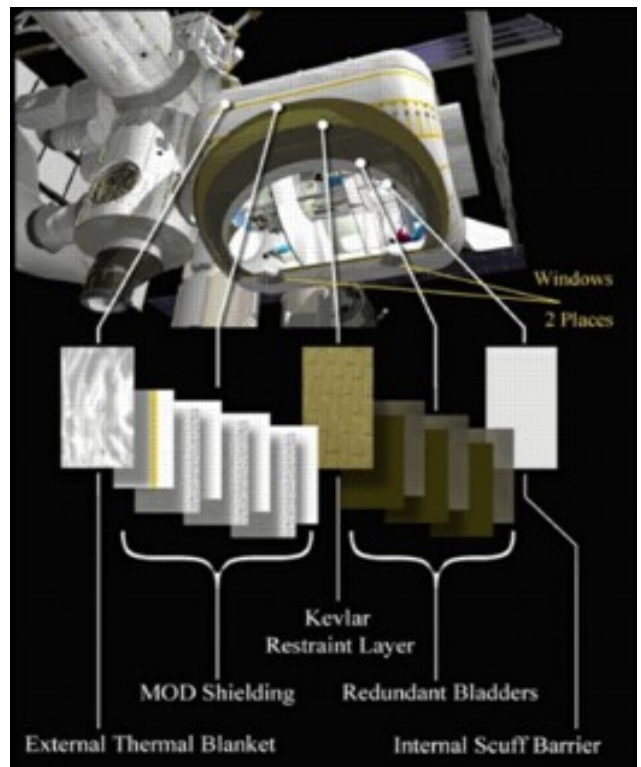
The International Space Station (ISS) U.S. Laboratory module (aka *Destiny*) is an excellent standard for the structure of a human space habitat (see Figure 18). This is due to its design based on international standards and its proven success in low Earth orbit since February 2001.

The *Destiny* module is roughly cylindrical with length 8.53 m and radius 2.13 m (NASA 2015c, 2015d). The end caps of the module are domed slightly to accommodate the hatches. The shell of the module is a dual layer of 2-3 mm of aluminum with a 10 cm spacing to create a Whipple effect. The module is covered with a 1.5 cm protective blanketing layer made of Mylar, Dacron®, and Kevlar® advanced fabrics (NASA 2001).



**Figure 18: ISS *Destiny* Module**  
(courtesy of NASA)

To simulate a deep space habitat in GEANT4, we selected the materials and geometry to resemble the *Destiny* module with a few modifications for simplicity and to accommodate minimum habitable volume recommendations for long-duration interplanetary missions.



**Figure 19: Description of ISS Protective Layers**  
(courtesy of NASA)

The shell of our simulated habitat was 1.8 cm of aluminum. This approximates the more complex ISS Whipple shields for which detailed specifications are difficult to locate in published literature. Our simulated habitat was then covered with 5 mm each of Mylar, Dacron®, and Kevlar® (Finckenor & Dooling 1999). This approximates the complex thermal and debris protective barriers shown in Figure 19. Note that our simulated spacecraft did not include windows, as these may vary considerably for interplanetary missions.

The minimum habitable volume recommended by NASA for a long-duration mission is 25 m<sup>3</sup> per person (Whitmire et al. 2015). In this case, habitable volume is defined as "...the volume left available to the crew after accounting for the loss of volume due to deployed equipment, stowage, trash, and any other structural inefficiencies and gaps (nooks and crannies) that decrease the functional volume" (NASA 2010). While the *Destiny* module has a gross volume of approximately 125 m<sup>3</sup>, the habitable volume is likely 50% smaller due equipment, stowage, etc. Thus, a module identical to *Destiny* could likely accommodate a maximum of 2 crew members on a long-duration mission. Assuming we prefer a reasonable crew size of 4 for an interplanetary mission, the module size must increase to a minimum of 100 m<sup>3</sup> habitable volume and approximately 200 m<sup>3</sup> of gross volume. Thus, our simulated habitat was a cylinder of length 10 m and radius 2.5 m with domed end caps, providing approximately 200 m<sup>3</sup> of gross volume and 100 m<sup>3</sup> of habitable volume for a desired crew size of 4.

#### 2.3.2.2 Crew Return Capsule

The NASA Multi-Purpose Crew Vehicle (MPCV, aka *Orion*) is an excellent standard for the structure of a crew capsule for Earth re-entry and landing. This is due to its design based successful vehicles such as the Apollo crew module and the Russian Soyuz

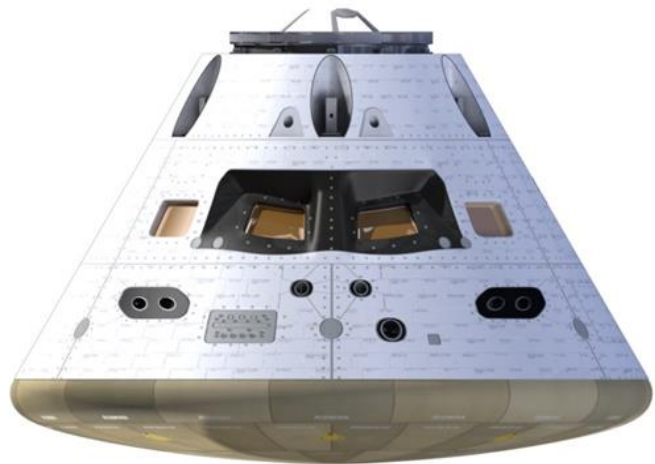


Figure 20: Orion Crew Module  
(courtesy of NASA)

descent stage as well as its comprehensive mission assurance and validation process.

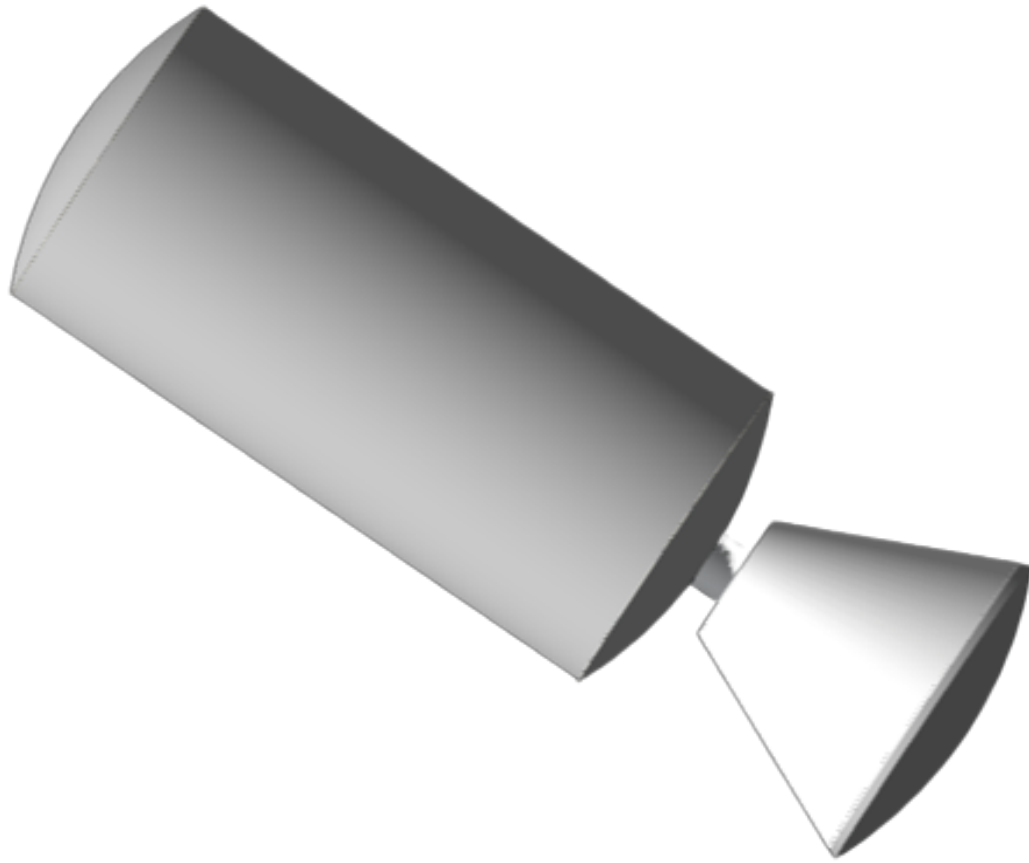
*Orion* is roughly conical with a lower radius of 2.5 m and a height of 3.3 m (Howell 2018a), providing 8.95 m<sup>3</sup> of habitable volume (NASA 2011). The bottom of the spacecraft is domed to accommodate the heat shield for atmospheric re-entry upon return to Earth (see Figure 20).

The shell of the spacecraft is aluminum-lithium (Al-Li) alloy 2195, the same material that was used for the Space Shuttle Super Lightweight External Tank. This alloy is composed of 94.2% aluminum, 4% copper, 1% lithium, 0.4% silver, and 0.4% magnesium (NASA 2005).

The *Orion* heat shield is a 4 cm thickness titanium truss structure with a composite substrate of AVCOAT ablative material, the same material that was used for the Apollo crew module heat shields in the 1960s and 1970s (Howell 2018b). AVCOAT is a low-density glass-filled epoxy-novolac compound of roughly 59% hydrogen, 34% carbon, and 7% oxygen (Crouch & Walberg 1969).

To simulate the crew return capsule in GEANT4, we selected the materials and geometry to resemble the *Orion* crew module with a few modifications for simplicity. The shell of our simulated crew return capsule was a conical section of 1.8 cm thickness Al-Li 2195, with upper radius 1 m and lower radius 2.5 m including a rounded end cap. We also included a 4 cm thickness AVCOAT layer on the bottom to approximate the heat shield.

The simulated crew return capsule was attached to the simulated habitat module via a small cylindrical adapter on one of the habitat's domed end caps. Figure 21 depicts the simulated spacecraft architecture in GEANT4.



**Figure 21: Simulated Spacecraft Architecture in GEANT4**

### *2.3.3 Shielding Configurations*

The next step in the radiation protection challenge is to identify the shielding strategy, method, and design. For this case, the shielding strategy was to maximize efficiency in reducing radiation dose while minimizing mass, power, and complexity for spaceflight. To provide the most informative comparison of shielding efficacy, in this study we compared our active magnetic shielding designs to two control cases using passive shielding. The parameters each of these configurations are described below. See also Appendix A7 for details of how the geometry is modeled in GEANT4.

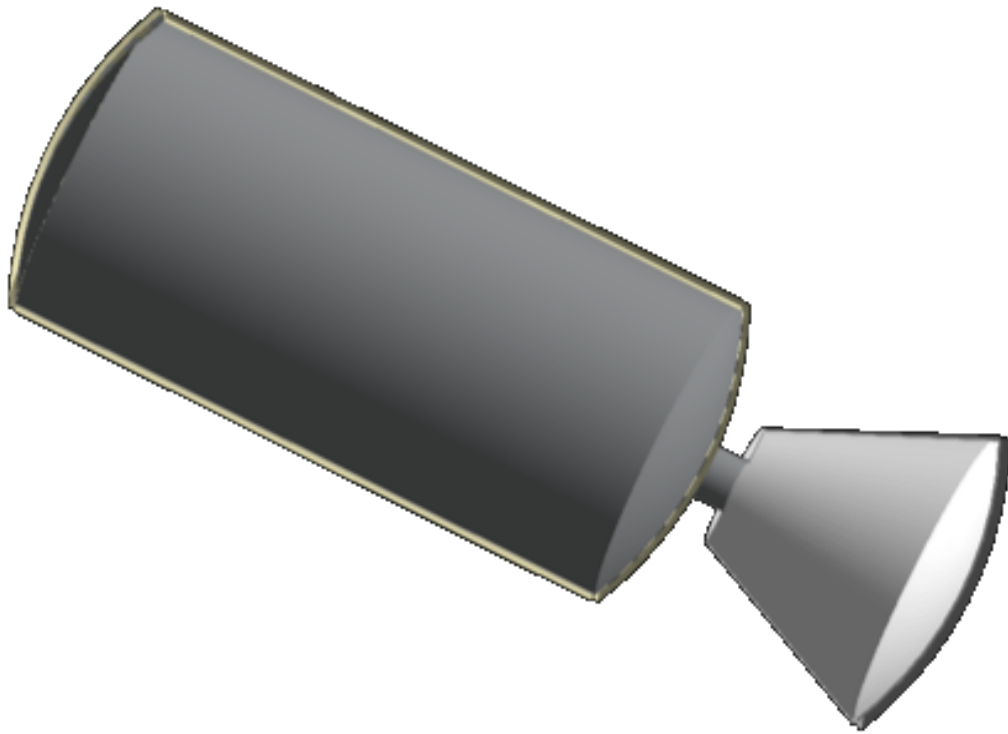
### 2.3.3.1 Minimal Shielding

The walls of typical spacecraft, such as the ISS modules (see Figure 22), provide approximately  $5 \text{ g/cm}^2$  of aluminum shielding (Durante 2014; Wilson et al. 1997). Dacron®, Kevlar®, and mylar layers of thermal and debris protective blanketing provide a varying amount of additional shielding (Finckenor & Dooling 1999).

To simulate this shielding configuration in GEANT4, we used a cylindrical 1.8 cm thickness aluminum spacecraft (corresponding to areal density of  $5 \text{ g/cm}^2$ ) of 10 m length and 2.5 m radius with domed end caps. The aluminum spacecraft shell was covered with 5 mm blankets of Mylar, Dacron®, and Kevlar®. Figure 23 depicts the simulated minimal shielding configuration in GEANT4.



Figure 22: Cutaway Model of ISS Columbus Module  
(courtesy of NASA)



**Figure 23: Simulated Minimal Shielding in GEANT4**

#### *2.3.3.2 Water Wall Shielding*

In several studies on shielding materials and methods, hydrogenous materials (high hydrogen content and low atomic number) are found most efficient per unit mass for passive shielding against space radiation (Adams et al. 2005; Simonsen & Nealy 1991; Simonsen et al. 1990; Spillantini et al. 2007; Wilson et al. 1998, 2001). This is due to several factors including more nuclei for the same shielding thickness, fewer neutrons in the nuclei resulting in fewer secondary neutrons, and smaller charge resulting in fewer secondary electrons and photons (Adams et al. 2005). A graphical depiction of the relative shielding effectiveness of different materials from the literature is reproduced in Figure 24.

Both water and polyethylene have high hydrogen content as well as favorable characteristics for spaceflight (stable, liquid or solid states, non-reactive). Based on NASA radiation transport analysis (Cucinotta et al. 2013; Slaba et al. 2017), 20 g/cm<sup>2</sup> of either of these materials would be a suitable choice for passive shielding on an interplanetary mission.

However, the shielding method proposed by NASA for near-term human exploration missions is deployable water wall shielding, as shown in Figure 25 (Semones 2015a). Further, water has several possible uses for human life support, and thus holds a distinct advantage over polyethylene in this regard. Therefore, in this study we selected 20 g/cm<sup>2</sup> (20 cm thickness) water as a passive shielding configuration, in combination with the shielding provided by the aluminum spacecraft and protective blanketing layers.

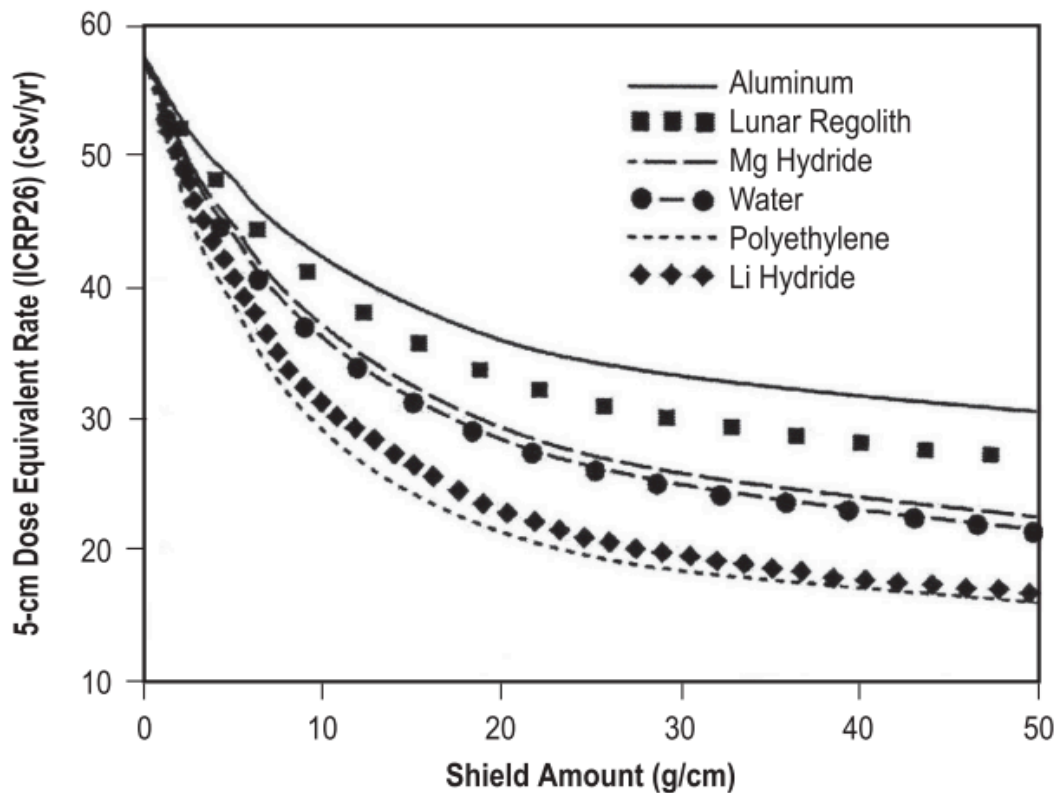
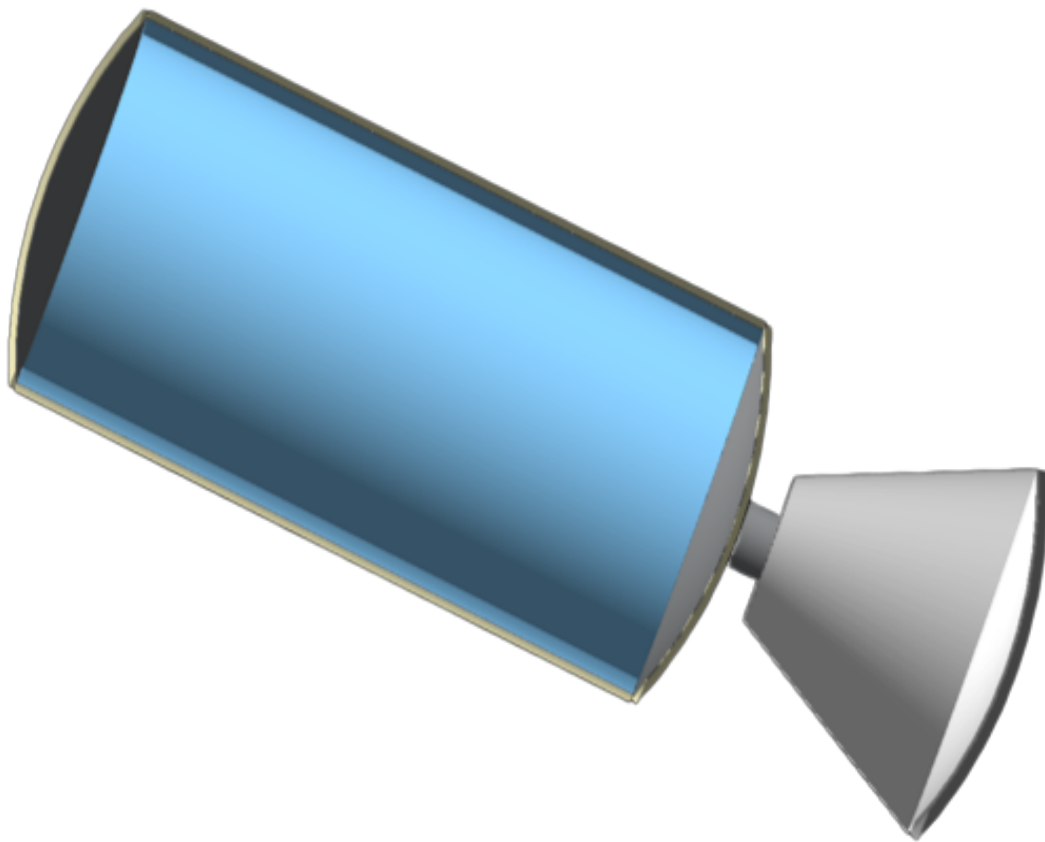


Figure 24: Dose at 5 cm Depth in Tissue for Various Materials (Adams et al. 2005); courtesy of NASA

To simulate this shielding configuration in GEANT4, we used the same cylindrical 1.8 cm thickness aluminum spacecraft of 10 m length and 2.5 m radius with domed end caps and 5 mm blankets of Mylar, Dacron®, and Kevlar®, and we line the inside of the spacecraft cylinder with a 20 cm layer of water. Figure 26 depicts the simulated water shielding configuration in GEANT4.



**Figure 25: Prototype Deployable Water Wall Shielding (Semones 2015); courtesy of NASA**



**Figure 26: Simulated Water Shielding Configuration in GEANT4**

### 2.3.3.3 Toroidal Magnetic Shielding

Toroidal magnets are often considered for space radiation shielding due to their confined magnetic fields (Battiston et al. 2013). In this type of magnetic field configuration, the field lines are contained within the toroid (see Figure 27). Thus, there is negligible fringe field reaching the habitable volume,

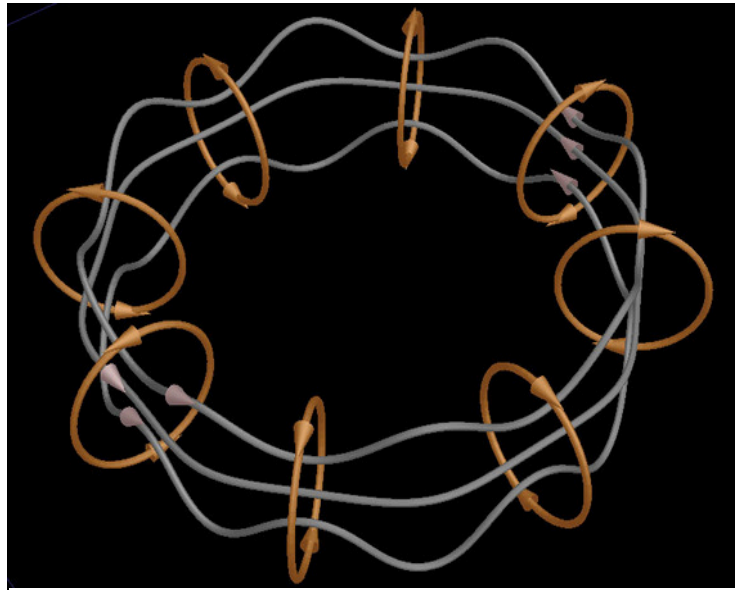


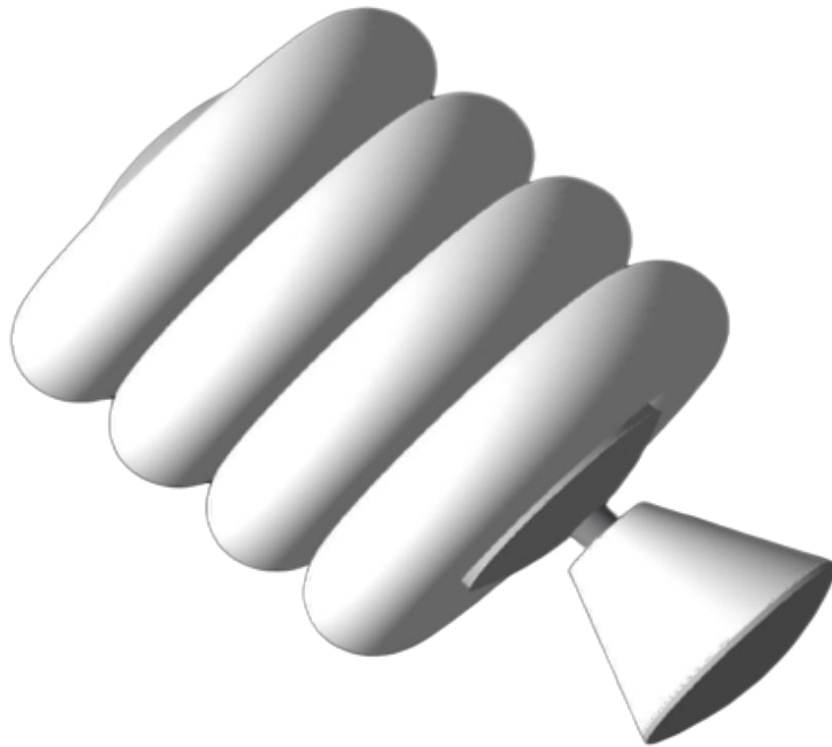
Figure 27: Simulated Toroidal Magnetic Field (created with Magnetism 3D)

reducing concern for magnetic field effects on the health of astronauts and function of electronic equipment. Toroids wrapping a cylindrical spacecraft offer protection from all directions with exception of the endcaps, which can be reinforced with passive shielding or smaller diameter toroids (Vuolo et al. 2016).

Several studies have examined the effectiveness of toroidal magnets for space radiation shielding (Battiston et al. 2011; Geng et al. 2015; Kash & Tooper 1962; Musenich et al. 2014; Papini & Spillantini 2014; Spillantini 2011, 2014; Washburn 2014; Westover 2014). These studies have shown that superconducting materials such as magnesium diboride ( $\text{MgB}_2$ ) and yttrium-barium-copper-oxide ( $\text{YBa}_2\text{Cu}_3\text{O}_{7-x}$  aka YBCO) are capable of carrying the high current required ( $\sim 10^6$  A) to produce magnetic fields in the 1-8 T range with reasonable structural, cooling, and overall mass requirements.

Since ours was a medical physics assessment rather than an engineering analysis, we relied on results of published engineering studies for the feasibility of our overall configuration including materials, structure, current, and magnetic field. Therefore, we simulated the materials and structures to a first approximation for simplicity.

To simulate toroidal magnetic shielding in GEANT4, we first used the same cylindrical 1.8 cm thickness aluminum spacecraft of 10 m length and 2.5 m radius with domed end caps and 5 mm blankets of Mylar, Dacron®, and Kevlar®. We then ringed the exterior of the simulated spacecraft with 4 toroidal YBCO magnets of cross-sectional radius 1.25 m and thickness of 2 cm to simulate coils of superconducting tape and support structure. To each toroid, we applied a constant magnetic field (0, 1.5, 3, or 7 T) in the tangential direction throughout the toroid. Figure 28 depicts the simulated toroidal shielding configuration in GEANT4.

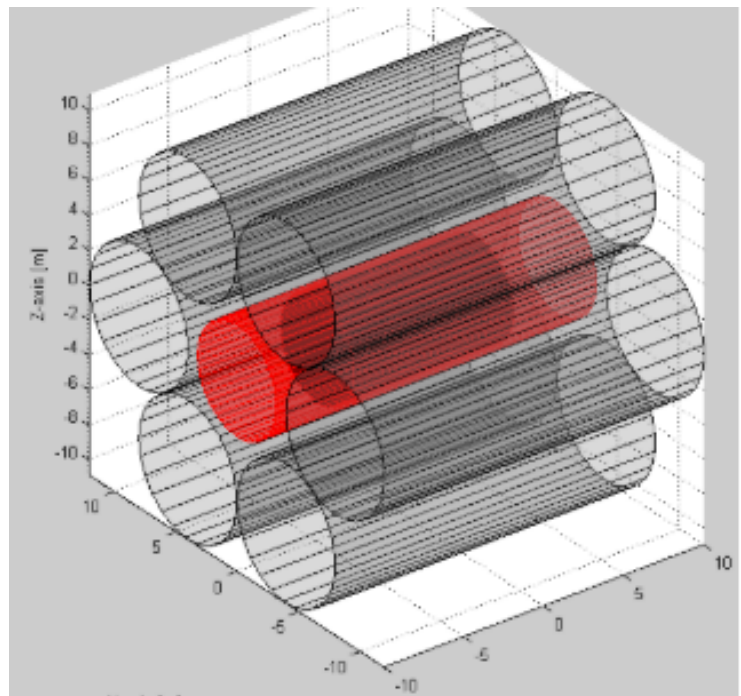


**Figure 28: Simulated Toroid Magnetic Shielding Configuration in GEANT4**

Prior to deciding on the configuration described above, we explored the use of 10 toroidal YBCO magnets of cross-sectional radius 0.5 m. In this configuration, the bending power (the product of magnetic field and path length of particles in the field) of each toroid was not adequate to deflect GCR particles at moderate magnetic fields. That is, the deflection of the particles was insufficient to prevent them from impinging on the spacecraft interior. Switching to fewer toroids with larger cross-sectional radius increased the bending power of each and improved the ability of the shielding toroids to deflect GCR particles at moderate magnetic fields.

#### 2.3.3.4 Solenoidal Magnetic Shielding

A common element in proposed magnetic shielding configurations is the solenoid, due to its simple design and reliability. Solenoids are often used for their strong and mostly uniform magnetic field along the central axis (Barker 1950). However, the much lower, highly variable magnetic field outside the solenoid is often ignored or characterized simply by its main characteristics, i.e. internal field, external field, dipole



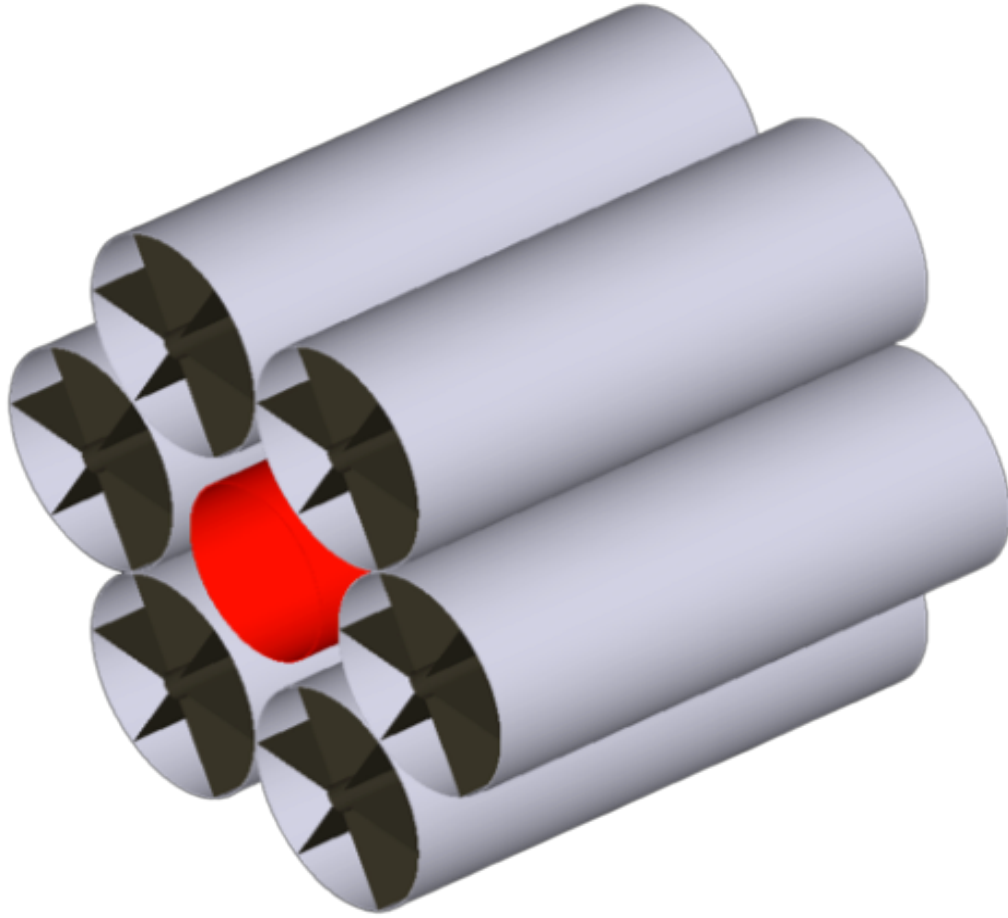
**Figure 29: NIAC 6+1 Solenoid Magnetic Shielding Configuration (Westover et al. 2014); courtesy of NASA**

field, fringe field regions (Farley & Price 2001). For this study, however, the magnetic field outside the solenoid is very important to include, since it is able to act at a distance to deflect incoming space radiation particles.

Several studies have examined the effectiveness of solenoidal magnets for space radiation shielding (Peroni 2017; Washburn et al. 2015; Westover 2014; Westover et al. 2012). These studies have suggested and analyzed several solenoidal field configurations. However, the NIAC 6+1 configuration (see Figure 29) appears most promising due to its effectiveness and relatively simple geometry. These studies have also shown that superconducting materials such as magnesium diboride ( $\text{MgB}_2$ ) and yttrium-barium-copper-oxide ( $\text{YBa}_2\text{Cu}_3\text{O}_{7-x}$  aka YBCO) are capable of carrying the high current required ( $\sim 10^6$  A) to produce magnetic fields in the 1-8 T range with reasonable structural, cooling, and overall mass requirements.

Since ours was a medical physics assessment rather than an engineering analysis, we relied on the results of published engineering studies for the feasibility of our overall configuration including materials, structure, current, and magnetic field. Therefore, we simulated the materials and structures to a first approximation for simplicity.

To simulate solenoidal magnetic shielding in GEANT4, we first used the same cylindrical 1.8 cm thickness aluminum spacecraft of 20 m length and 2.5 m radius with domed end caps and 5 mm blankets of Mylar, Dacron®, and Kevlar®. We then placed 6 solenoidal YBCO magnets off-axis from the simulated spacecraft with length of 20 m, radius of 4 m, and thickness of 5 cm to simulate coils of superconducting tape and support structure. We also included a solenoidal correction magnet surrounding the spacecraft of length 20 m, radius 3.2 m, and thickness 5 cm; we applied a small magnetic field in the opposite direction to this solenoid to minimize the residual magnetic field within the habitat module. The support structure for each of the six outer coils included a central graphite (carbon) cylinder of 0.5 m radius, 1 cm thickness, and 20 m length. Each outer coil also included six radial graphite (carbon) plates of 3.5 m width, 2.5 mm thickness, and 20 m length (Westover et al. 2012). Figure 30 depicts the simulated solenoidal shielding configuration in GEANT4.



**Figure 30: Simulated NIAC 6+1 Solenoid Magnetic Shielding Configuration in GEANT4**

In order to accurately characterize the complete magnetic field of a solenoid, most solutions use numerical integration (Antokhin et al. 1999). However, an exact solution using elliptic integrals was published by NASA in 1960 (Callaghan & Maslen 1960) and particle accelerator physicists have worked out additional analytical solutions (Gorlov & Holmes 2018). We used the magnetic field equations published by Callaghan and Maslen in our GEANT4 models to quantify the magnetic field at a given point in the interior volume of each solenoid.

In the cylindrical coordinate system  $(r, \theta, z)$ , the equation for the radial component ( $B_r$ ) of a solenoid's magnetic field is given by:

$$B_r = \frac{\mu n i}{\pi} \sqrt{\frac{a}{r}} \left[ \frac{2 - k^2}{2k} K(k) - \frac{E(k)}{k} \right]_{\xi_-}^{\xi_+}$$

where:

$\mu$  = permeability

$n$  = number of turns per unit coil length (N/L)

$i$  = current in each filament

$a$  = solenoid coil radius

$L$  = solenoid length

$$\xi_{\pm} = z \pm \frac{L}{2}$$

$$k = \sqrt{\frac{4 a r}{\xi^2 + (a+r)^2}}$$

$K$  = complete elliptic integral, first kind

$E$  = complete elliptic integral, second kind

The equation for the axial component ( $B_z$ ) of a solenoid's magnetic field is given by:

$$B_z = \frac{\mu n i}{4} \left[ \frac{\xi k}{\pi \sqrt{a r}} K(k) + \frac{(a-r)\xi}{|(a-r)\xi|} \lambda_0(\varphi, k) \right]_{\xi_-}^{\xi_+}$$

where:

$\mu$  = permeability

$n$  = number of turns per unit coil length

$i$  = current in each filament

$a$  = solenoid coil radius

$L$  = solenoid length

$$\xi_{\pm} = z \pm \frac{L}{2}$$

$$k = \sqrt{\frac{4 a r}{\xi^2 + (a+r)^2}}$$

$K$  = complete elliptic integral, first kind

$\lambda_0$  = Heuman lambda function

$$\varphi = \tan^{-1} \left| \frac{\xi}{a-r} \right|$$

The permeability ( $\mu$ ) for these simulations is determined by:

$$\mu = \mu_s \times \mu_0$$

where  $\mu_s$  is the relative permeability of the material and  $\mu_0$  is the permeability of free space ( $4\pi \times 10^{-7} \frac{T m}{A}$ ).

For these simulations, we used YBCO superconducting material with a relative permeability,  $\mu_s = 0.27$  (Nose et al. 1990). Therefore, the overall permeability for these simulations is:

$$\mu = 0.27 \times 4\pi \times 10^{-7} \frac{T m}{A} = 3.39 \times 10^{-7} \frac{T m}{A}$$

For the purposes of these simulations, the individual values for the number of turns per unit coil length ( $n$ ) and the current in each filament ( $i$ ) were not determined. However, their product ( $ni$ ) can be estimated for a given scenario from the equation for the axial magnetic field inside the solenoid,  $B_0$  (Barker 1950) and the above calculation for permeability ( $\mu$ ):

$$B_0 = \mu n i \rightarrow n i = \frac{B_0}{\mu}$$

For the three axial magnetic fields used in our scenarios (1.5 T, 3 T, and 7 T), the product  $ni$  was estimated as:

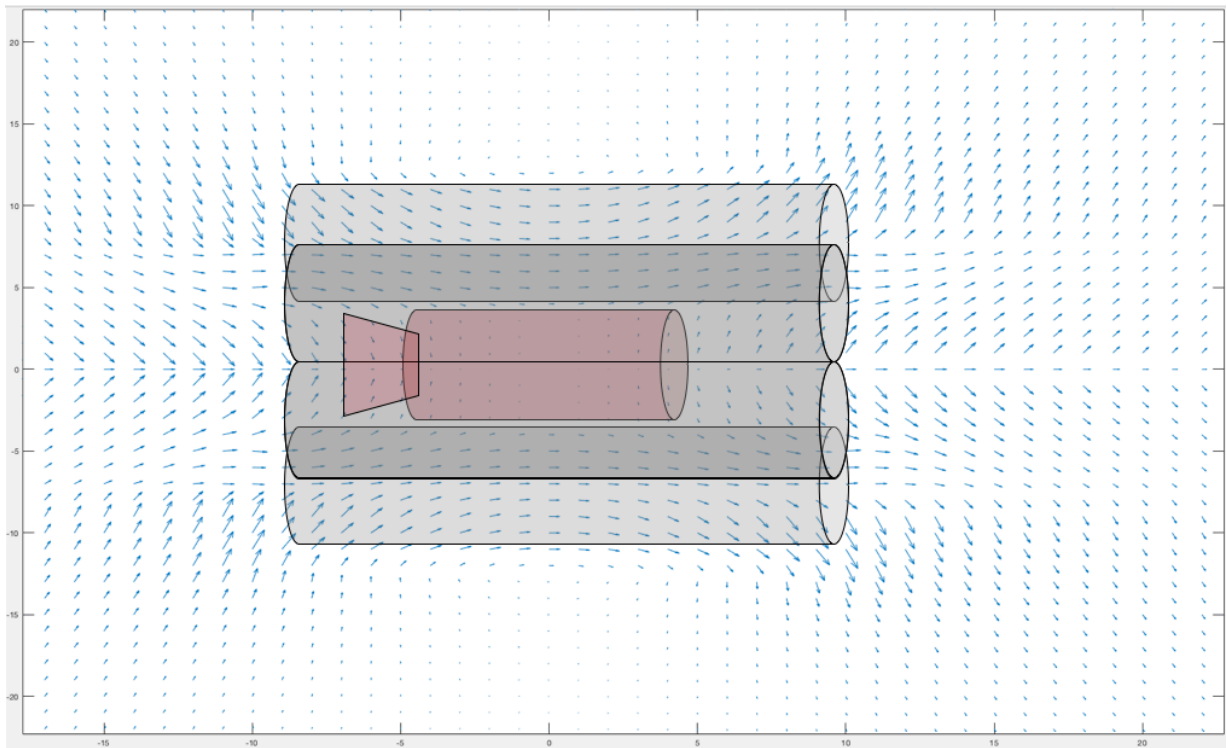
$$ni \cong \frac{B_0}{\mu} = \frac{1.5 \text{ T}}{3.39 \times 10^{-7} \frac{\text{T m}}{\text{A}}} = 4.42 \times 10^6 \frac{\text{A}}{\text{m}}$$

$$ni \cong \frac{B_0}{\mu} = \frac{3 \text{ T}}{3.39 \times 10^{-7} \frac{\text{T m}}{\text{A}}} = 8.85 \times 10^6 \frac{\text{A}}{\text{m}}$$

$$ni \cong \frac{B_0}{\mu} = \frac{7 \text{ T}}{3.39 \times 10^{-7} \frac{\text{T m}}{\text{A}}} = 2.06 \times 10^7 \frac{\text{A}}{\text{m}}$$

For the elliptic integrals, K and E, and the Heuman lambda function,  $\lambda_0$ , we used MATLAB (version r2018a) with the *elfun18* add-on function set. The MATLAB script we wrote titled *Solenoid\_6plus1\_txt.m* (see Appendix A8) produced a text file containing tabulated values of the magnetic field components ( $B_x$ ,  $B_y$ , and  $B_z$ ) at a resolution of 1 m. (Note: The resolution is low due to the large volume covered. With this 1 m resolution, the tabulated field list is over 500,000 lines; with any finer resolution, the cost in computing time rendered the simulation intractable.) The exported text file provides a field map for the simulation world volume, an 80 m cube. Figure 31 depicts the total magnetic field map (using the MATLAB *quiver* function) for the x-z plane for the 6+1 solenoidal configuration with transparent overlays to highlight the locations of solenoids and the habitat module. Using this method within MATLAB, we were able to optimize the magnetic field of the correction magnet and ensure that the magnetic field within the habitat module was less than 5 mT.

We then used a custom tabulated 3D magnetic field class modified from a GEANT4 example (/examples/advanced/purging\_magnet) to apply the field map to the simulation world volume. During each particle event in the simulation, the magnetic field for a given point was determined by linear interpolation of the magnetic field at the nearest mapped points.



**Figure 31: Magnetic Field Map (x-z plane) from MATLAB**

#### 2.3.3.5 Race Track Magnetic Shielding

The innovative “race track” magnetic shielding configuration is described by the European Space Agency (ESA) Space Radiation Superconducting Shield (SR2S) study (Ambrogini et al. 2016; Battiston et al. 2013; Vuolo et al. 2016). In this shielding configuration, superconducting coils are shaped into elongated loops (resembling race tracks) and placed at small angle intervals around the spacecraft to form an elongated toroid (see Figure 32). This configuration takes advantage of the benefits of toroidal magnetic shielding by providing negligible fringe field inside

a cylindrical habitat with isotropic radiation protection with exception of the endcap regions, which can be reinforced with passive shielding.

The ESA studies present and analyze several configurations of race track magnetic shielding. However, we chose configuration A due to its effectiveness and relatively simple geometry. These studies have also shown that superconducting materials such as magnesium diboride ( $\text{MgB}_2$ ) and yttrium-barium-copper-oxide ( $\text{YBa}_2\text{Cu}_3\text{O}_{7-x}$  aka YBCO) are capable of carrying the high current required ( $\sim 10^6$  A) to produce magnetic

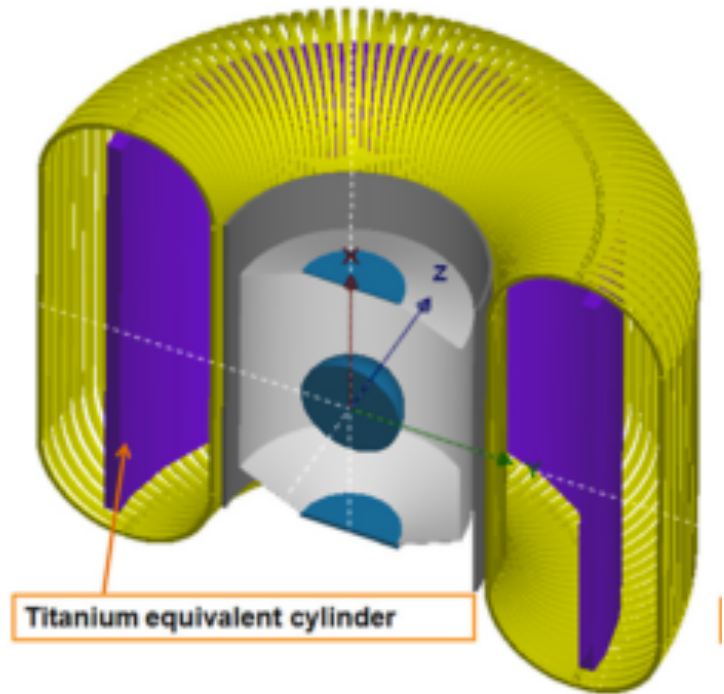


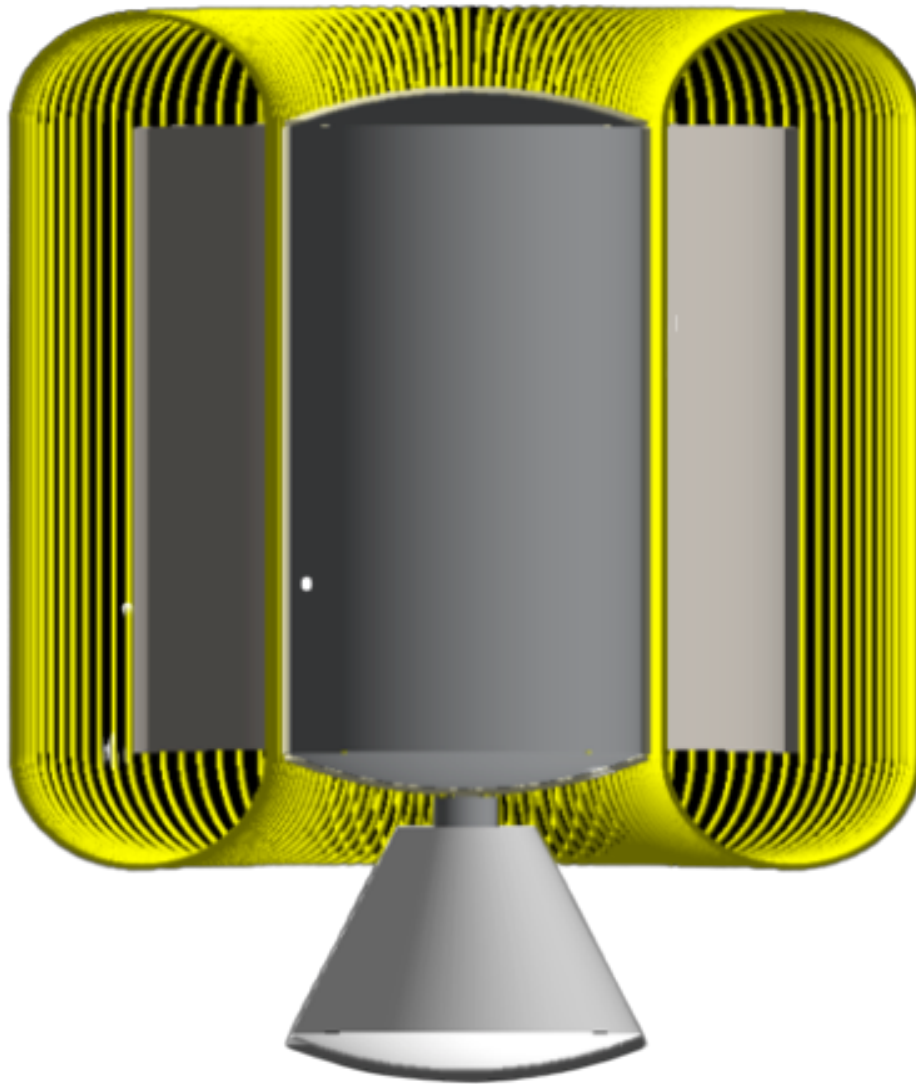
Figure 32: ESA SR2S Race Track Shielding (Vuolo et al. 2016); courtesy of Elsevier

fields in the 1-8 T range with reasonable structural, cooling, and overall mass requirements.

Since ours was a medical physics assessment rather than an engineering analysis, we relied on the results of published engineering studies for the feasibility of our overall configuration including materials, structure, current, and magnetic field. Therefore, we simulated the materials and structures to a first approximation for simplicity.

To simulate race track magnetic shielding in GEANT4, we first used the same cylindrical 1.8 cm thickness aluminum spacecraft of 10 m length and 2.5 m radius with domed end caps and 5 mm blankets of Mylar, Dacron®, and Kevlar®. We then assemble YBCO race track

superconducting coils of 13.6 m height, 3.6 m width and 5 cm wall thickness and place these at  $3^\circ$  intervals surrounding the spacecraft, for a total of 120 race track loops. A titanium forming cylinder was placed inside the race track loops. Note the original configuration in the ESA S2RS study chose  $\text{MgB}_2$  superconducting material instead of YBCO for mass savings. We chose YBCO because of its higher critical temperature. To the interior of the overall toroid, we applied a constant magnetic field (0, 1.5, 3, or 7 T) in the tangential direction throughout the volume. Figure 33 depicts the simulated race track shielding configuration in GEANT4.



**Figure 33: Simulated Modified ESA Race Track Shielding Configuration in GEANT4**

### 2.3.4 Dosimetry and Risk Analysis

#### 2.3.4.1 Human Phantom Selection

The final step in developing a radiation protection solution is choosing how to best model the human body (ICRU 1992; Xu & Eckerman 2010; ICRP 2009) and perform dosimetry to analyze radiation risk and establish limits (Adams 1992; NCRP 1993, 2000; Peterson & Kovyrshina 2015; Swenberg et al. 1993). For this study, the human astronaut was modeled as either a simple water phantom or the male or female adult version of the stylized MIRD phantom based on the ICRP Reference Man (Cristy & Eckerman 1987a, 1987b; ICRP 1975; Snyder et al. 1974, 1978; Xu & Eckerman 2010). The geometry of organ volumes for the MIRD phantom were modified from a GEANT4 example (/examples/advanced/human\_phantom). The stylized MIRD phantom as represented in the literature (Snyder et al. 1974, 1978) is reproduced in Figure 34. Our recreation of the stylized male and female MIRD phantoms modified from the GEANT4 example are presented Figure 35.

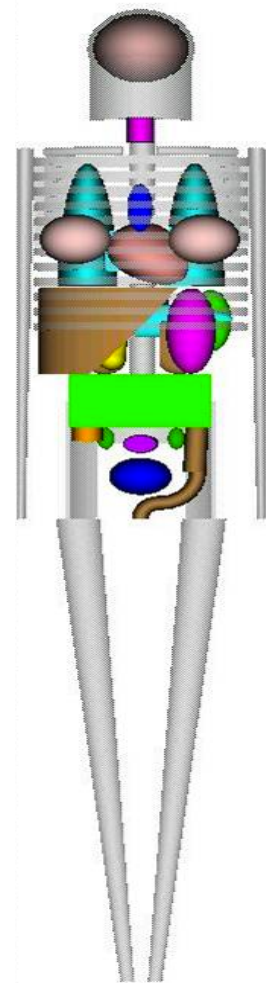


Figure 34: MIRD Phantom (Segars & Tsui 2009); courtesy of IEEE

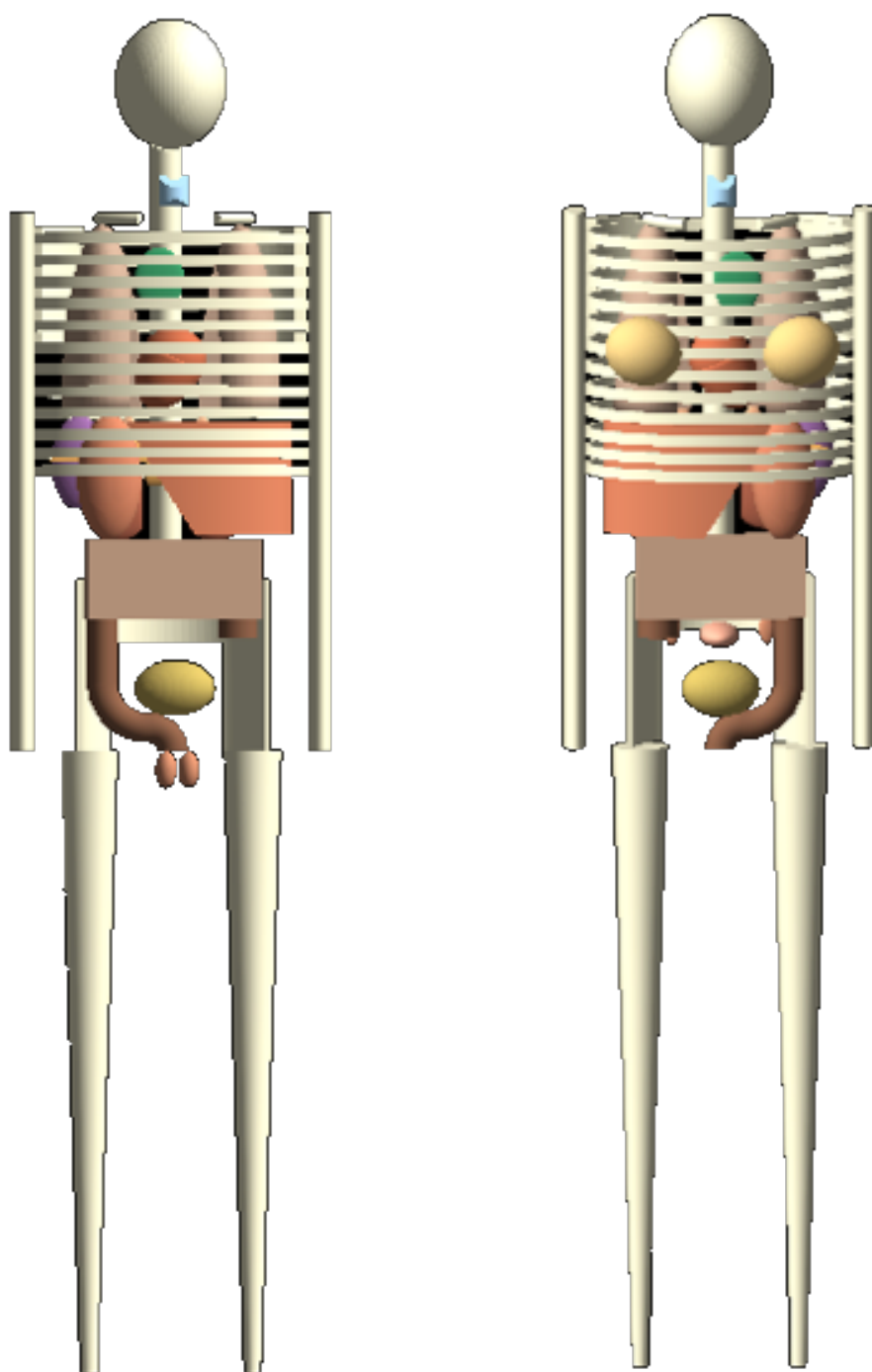


Figure 35: Male and Female Phantoms in GEANT4

#### 2.3.4.2 Dosimetry Method

Absorbed dose (D) is the fundamental dose quantity that describes the energy deposited by ionizing radiation. It has the SI unit of joule per kilogram (J/kg) or gray (Gy) and is given by (ICRP 1991, 2007):

$$D = \frac{d\bar{\epsilon}}{dm}$$

where:

$d\bar{\epsilon}$  = mean energy imparted by ionizing radiation

$dm$  = mass

Absorbed dose was calculated in our simulations by collecting the energy deposited in each volume per unit mass of the volume. This calculation was done within our GEANT4 code and also includes a calculation of the standard error of the mean dose deposited in each volume of interest.

Dose equivalent (H) is the dose quantity that adjusts absorbed dose (D) to account for the relative biological effectiveness of different types and energies of ionizing radiation. This adjustment is made scaling the absorbed dose (D) by the quality factor (Q). Dose equivalent (H) has the SI unit of joule per kilogram (J/kg) or Sievert (Sv) and is given by (ICRP 1991, 2007):

$$H = DQ$$

where:

D = absorbed dose

Q = quality factor

The quality factor (Q) is based on the ionization density along the track of a charged particle in water. The quality factor (Q) is defined by a particles linear energy transfer (LET) in water and is given by (ICRP 1991, 2007):

$$Q = \left\{ \begin{array}{ll} 1 & LET < 10 \frac{keV}{\mu m} \\ 0.32LET - 2.2 & 10 \leq LET \leq 100 \frac{keV}{\mu m} \\ \frac{300}{\sqrt{LET}} & LET > 100 \frac{keV}{\mu m} \end{array} \right\}$$

Note that although radiation weighting factors ( $w_R$ ) have largely replaced the quality factor (Q) in radiation protection literature over the past several years (ICRP 2003), the space radiation community still prefers to use the quality factor (Q) method because of the large contribution of heavy ions to the spectra (ICRP 2003). Furthermore, NASA has released updates to the quality factor (Q) method specifically to address track structure effects from space radiation, but the updated quality factors ( $Q_{NASA}$ ) are still in development (Borak et al. 2014; Cucinotta et al. 2005). Due to gaps in understanding the complete biological effects of HZE radiation, further study is needed to reduce the uncertainties in estimating the risk of carcinogenesis from space radiation (Cucinotta et al. 2001b, 2004; Chancellor et al. 2018).

The absorbed dose delivered by any given particle in our simulations was converted to dose equivalent (H) by multiplying the absorbed dose (D) by the quality factor (Q) based on the linear energy transfer (LET) of the particle. This calculation was done within our GEANT4 code and also includes a calculation of the standard error of the dose equivalent to each volume of interest.

Effective dose (E) is the dose quantity that corrects for the relative sensitivities of organs to radiation dose. It is defined as a weighted sum of the equivalent doses to all organs of the body

based on tissue weighting factors ( $w_T$ ). Effective dose (E) has the SI unit of joule per kilogram (J/kg) or Sievert (Sv) and is given by (ICRP 1991, 2007):

$$E = \sum_T w_T H_T$$

where:

T = tissue or organ

$w_T$  = tissue weighting factor

$H_T$  = dose equivalent in the tissue or organ

The tissue weighting factors ( $w_T$ ) given by ICRP Publication 103 are defined reproduced below in Table 2 (ICRP 2007).

**Table 2: Tissue Weighting Factors (ICRP 2007)**

Organ	wT
Bone Marrow (Red)	0.12
Colon	0.12
Lung	0.12
Stomach	0.12
Breast	0.12
Gonads	0.08
Bladder	0.04
Esophagus	0.04
Liver	0.04
Thyroid	0.04
Bone Surface	0.01
Brain	0.01
Salivary Glands	0.01
Skin	0.01
<i>Remainder:</i>	0.12
adrenals, extra thoracic region, gall bladder, heart, kidneys, lymphatic nodes, muscle, oral mucosa, pancreas, small intestine, spleen, thymus, uterus, prostate	

Our phantom represented a 70 kg human, but the total mass of organs used to calculate effective dose was ~10 kg. This is important to note because tallying dose to a smaller total mass

and volume produced higher uncertainties for the same number of incident particles in a given simulation vs. using a 70 kg water phantom.

Our phantom did not include volumes for esophagus, bone surface, salivary glands, or skin, so we added the sum of their respective tissue weighting factors to remainder weighting factor. For the female phantom, this brought the total weighting factor for remainder to 0.19. Our male phantom further did not include breast volumes, so we added the value of the breast tissue weighting factor (0.12) to the remainder weighting factor as well. For the male phantom, this brought the total weighting factor for remainder to 0.31.

Our phantom did not include volumes for the following remainder tissues: extra thoracic region, gall bladder, lymphatic nodes, muscle, oral mucosa, or prostate. In order to compensate for this, we applied the total remainder weighting factor to the average of the equivalent dose to the remainder tissues we do have: adrenals, heart, kidneys, pancreas, small intestine, spleen, thymus, and uterus. Because present and missing tissues are evenly distributed throughout the body, we believe this compensation to be an adequate estimation. Table 3 and Table 4 show the tissue weighting factors we used in our models, including the adjustments described above.

**Table 3: Tissue Weighting Factors (modified for female MIRD phantom)**

<b>Organ</b>	<b>wT</b>
Active (Red) Bone Marrow	0.12
Large Intestine	0.12
Lungs	0.12
Stomach	0.12
Breasts	0.12
Gonads	0.08
Bladder	0.04
Liver	0.04
Thyroid	0.04
Brain	0.01
<i>Remainder:</i>	0.19
adrenals, heart, kidneys, pancreas, small intestine, spleen, thymus, uterus	

**Table 4: Tissue Weighting Factors (modified for male MIRD phantom)**

<b>Organ</b>	<b>wT</b>
Active (Red) Bone Marrow	0.12
Large Intestine	0.12
Lungs	0.12
Stomach	0.12
Gonads	0.08
Bladder	0.04
Liver	0.04
Thyroid	0.04
Brain	0.01
<i>Remainder:</i>	0.31
adrenals, heart, kidneys, pancreas, small intestine, spleen, thymus	

The traditional methods for calculating dose to active bone marrow in Monte Carlo simulations are to estimate the dose based on surrogate volumes or to use the average dose to the soft tissue (Dant et al. 2013; Gialousis et al. 2008; Kvinnsland et al. 2001; Nie & Richardson 2009; Snyder et al. 1974, 1978; Xu & Eckerman 2010). While these methods are acceptable for most radiation protection studies, there are limitations of each including the inability to calculate dose to a specific bone marrow volume and dose inaccuracies for certain energy ranges and radiation species. Newer models have solved portions of these issues for specific applications primarily within the nuclear medicine community and involving internal radiation sources (Dant et al. 2013; Nie & Richardson 2009; Cristy & Eckerman 1987a, 1987b), but these models vary in complexity and general applicability. For our simulations we chose to collect dose deposited to the active marrow directly by modeling the marrow volumes in our GEANT4 geometry. The distribution of active bone marrow in an average 40-year-old adult is given in Table 5 (Cristy 1981).

**Table 5: Active Bone Marrow Distribution for 40-Year-Old Adult (Cristy 1981)**

<b>Bone</b>	<b>% Active Marrow</b>
Cranium	7.6
Mandible	0.8
Scapulae	2.8
Clavicles	0.8
Sternum	3.1
Ribs	16.1
Cervical Vertebrae	3.9
Thoracic Vertebrae	16.1
Lumbar Vertebrae	12.3
Sacrum	9.9
Os Coxae	17.5
Femora	6.7
Tibiae, fibulae, patellae	0
Tarsals, metatarsals, phalanges	0
Humeri	2.3
Ulnae, radii	0
Carpals, metacarpals, phalanges	0

In order to simulate active bone marrow distribution in our 70 kg MIRD phantom, we assigned daughter volumes for bone marrow within the bone volumes included in the MIRD phantom. Because the MIRD phantom does not include a mandible, we increased the distribution of marrow in the cranium to account for the additional amount. Similarly, because the MIRD phantom does not include a sternum, we increased the marrow distribution in the ribs to account for that amount. Because the MIRD phantom includes separate volumes for each rib, we also split that marrow percentage equally between the 12 rib volumes. Further, because the MIRD phantom includes one total volume for mid-lower spine rather than separate volumes thoracic vertebrae, lumbar vertebrae, and sacrum, we distributed the sum of those three marrow percentages evenly across the single mid-lower spine volume. Also, because the MIRD phantom includes one total volume for leg bones and one total volume for arm bones, we distributed the sum of those marrow percentages evenly across the larger volumes. Further, the MIRD phantom does not include hands or feet, but that does not present a problem because the lower extremities have zero active bone marrow in the 40-year-old adult (Cristy 1981). Finally, because the MIRD phantom includes separate volumes for left and right bones for clavicle, scapula, legs, and arms, we divided the marrow distribution for the pair evenly between the left and right side. Table 6

shows the active bone marrow distribution we used in our models, including the adjustments described above.

**Table 6: Active Bone Marrow Distribution for 40-Year-Old Adult (modified)**

<b>Bone</b>	<b>% Active Marrow</b>
Cranium	8.4
Left Scapula	1.4
Right Scapula	1.4
Left Clavicle	0.4
Right Clavicle	0.4
Rib 1	1.6
Rib 2	1.6
Rib 3	1.6
Rib 4	1.6
Rib 5	1.6
Rib 6	1.6
Rib 7	1.6
Rib 8	1.6
Rib 9	1.6
Rib 10	1.6
Rib 11	1.6
Rib 12	1.6
Upper Spine	3.9
Mid-Lower Spine	38.4
Pelvis	17.5
Left Leg Bone	3.35
Right Leg Bone	3.35
Left Arm Bone	1.15
Right Arm Bone	1.15

The average adult bone mass is approximately 15% of the total body mass. Thus, for our 70 kg adult phantom, the bone mass should be approximately 10.5 kg. The total mass of bone marrow in adults is approximately 4% of the total body mass (Martins et al. 2009), with approximately half of this marrow being hematopoietic/red (active) marrow. Thus, for our 70 kg adult phantom, the total bone marrow mass is approximately 3 kg and red marrow mass is approximately 1.5 kg. The density of active marrow is  $1.06 \text{ g/cm}^3$ , with elemental composition as shown in Table 7 (Dant et al. 2013).

Table 7: Active Marrow Elemental Composition (Dant et al. 2013)

Element	Active Marrow Composition (% by mass)
Hydrogen	10.2
Carbon	14.3
Nitrogen	3.4
Oxygen	70.8
Sodium	0.2
Phosphorus	0.3
Sulphur	0.3
Chlorine	0.2
Potassium	0.3

In order to size the bone marrow daughter volume for each bone in the MIRD phantom in GEANT4, we performed calculations based on the active bone marrow distribution, average total bone marrow mass in adults, bone marrow density and composition, and the size and shape of each bone volume. A summary of these calculations is included in Table 8. Figure 36 shows a cutaway view of our modified MIRD phantom in GEANT4.

Table 8: Active Bone Marrow Mass/Volume Calculations

Bone	Mass (g)	Net Volume (cm <sup>3</sup> )	Density (g/cm <sup>3</sup> )	% Total Active Marrow	Marrow Volume Shape	Marrow Volume Calculation	Marrow Volume Dimensions (cm)
Cranium	1398.05	728.15	1.92	-	-	-	-
Cranium Marrow	126.00	118.87	1.06	8.40%	Subtraction Ellipsoid	$V = \frac{4}{3}\pi(ABC - abc)$	A = 6.48, B = 9.48, C = 7.48 a = 6.32, b = 9.32, c = 7.32
Left Scapula	155.77	81.13	1.92	-	-	-	-
Left Scapula Marrow	21.00	19.81	1.06	1.40%	Subtraction Elliptical Tube Segment	$V = \pi(ABH - abh) \times \frac{\theta}{2\pi}$	A = 18.2, B = 9.8, H = 8.199 a = 17.8, b = 9.8, h = 8.199 $\theta \sim 0.4$ rad
Right Scapula	156.25	81.38	1.92	-	-	-	-
Right Scapula Marrow	21.00	19.81	1.06	1.40%	Subtraction Elliptical Tube Segment	$V = \pi(ABH - abh) \times \frac{\theta}{2\pi}$	A = 18.2, B = 9.8, H = 8.199 a = 17.8, b = 9.8, h = 8.199 $\theta \sim 0.4$ rad
Left Clavicle	15.38	8.01	1.92	-	-	-	-
Left Clavicle Marrow	6.00	5.66	1.06	0.40%	Torus Segment	$V = \pi r^2 \times 2\pi R \times \frac{\theta}{2\pi}$	r = 0.5, R = 10, $\theta = 0.69$ rad
Right Clavicle	15.38	8.01	1.92	-	-	-	-
Right Clavicle Marrow	6.00	5.66	1.06	0.40%	Torus Segment	$V = \pi r^2 \times 2\pi R \times \frac{\theta}{2\pi}$	r = 0.5, R = 10, $\theta = 0.69$ rad
Rib 1	66.62	34.70	1.92	-	-	-	-
Rib 1 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 2	67.52	35.17	1.92	-	-	-	-
Rib 2 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 3	68.35	35.60	1.92	-	-	-	-
Rib 3 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 4	67.37	35.09	1.92	-	-	-	-

Rib 4 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 5	67.43	35.12	1.92	-	-	-	-
Rib 5 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 6	67.22	35.01	1.92	-	-	-	-
Rib 6 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 7	67.54	35.18	1.92	-	-	-	-
Rib 7 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 8	67.43	35.12	1.92	-	-	-	-
Rib 8 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 9	67.91	35.37	1.92	-	-	-	-
Rib 9 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 10	67.14	34.97	1.92	-	-	-	-
Rib 10 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 11	67.10	34.95	1.92	-	-	-	-
Rib 11 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Rib 12	67.95	35.39	1.92	-	-	-	-
Rib 12 Marrow	24.00	22.64	1.06	1.60%	Subtraction Elliptical Tube	$V = \pi(ABH - abh)$	A = 16.85, B = 9.65, H = 1.398 a = 16.65, b = 9.45, h = 1.398
Upper Spine	136.73	71.21	1.92	-	-	-	-
Upper Spine Marrow	58.50	55.19	1.06	3.90%	Elliptical Tube	$V = \pi(abh)$	a = 1.53, b = 1.91, h = 6
Middle Lower Spine	407.17	212.07	1.92	-	-	-	-
Middle Lower Spine Marrow	574.50	541.98	1.06	38.30%	Elliptical Tube	$V = \pi(abh)$	a = 1.7, b = 2.12, h = 47.98
Pelvis	681.71	355.06	1.92	-	-	-	-
Pelvis Marrow	262.50	247.64	1.06	17.50%	Subtraction Elliptical Tube Segment	$V = \pi(ABH - abh) \times \frac{\theta}{2\pi}$	A = 11.68, B = 11.79, H = 21.998 a = 11.51, b = 11.51, h = 21.998 $\theta \sim \pi$ rad
Left Leg Bone	2596.48	1352.33	1.92	-	-	-	-
Left Leg Bone Marrow	50.25	47.41	1.06	3.35%	Cylinder	$V = \pi r^2 h$	r = 0.434, h = 79.798
Right Leg Bone	2596.48	1352.33	1.92	-	-	-	-
Right Leg Bone Marrow	50.25	47.41	1.06	3.35%	Cylinder	$V = \pi r^2 h$	r = 0.434, h = 79.798
Left Arm Bone	1541.62	802.93	1.92				
Left Arm Bone Marrow	17.25	16.27	1.06	1.15%	Elliptical Tube	$V = \pi(abh)$	a = 0.197, b = 0.38, h = 69
Right Arm Bone	1539.85	802.01	1.92				
Right Arm Bone Marrow	17.25	16.27	1.06	1.15%	Elliptical Tube	$V = \pi(abh)$	a = 0.197, b = 0.38, h = 69
<b>Total Bone Mass</b>	<b>12050.44</b>						
<b>Total Active Marrow Mass</b>	<b>1498.50</b>						

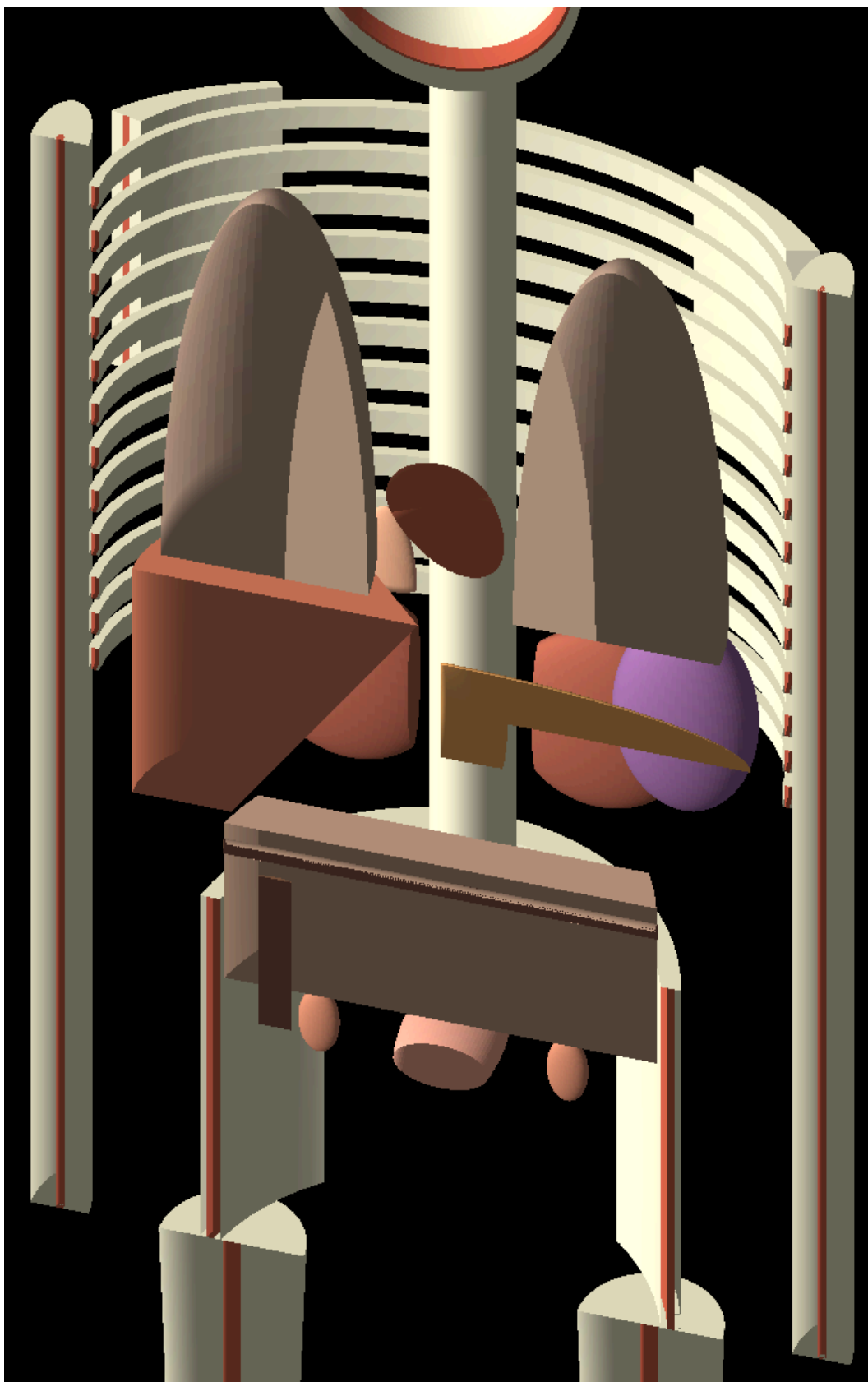


Figure 36: Modified MIRD Phantom with Active Bone Marrow Volumes in GEANT4

The MIRD phantom did not include eye lenses, but NASA is concerned with doses to the eye lens of  $>8$  mSv (Ainsbury et al. 2009; Cucinotta et al. 2001a) during spaceflight. Therefore, we added the eye lenses to our modified MIRD phantom in GEANT4. The mass of each eye lens in the adult is approximately 0.22 g, and the dimensions of the lens are approximately 4.6 mm radius and 3.1

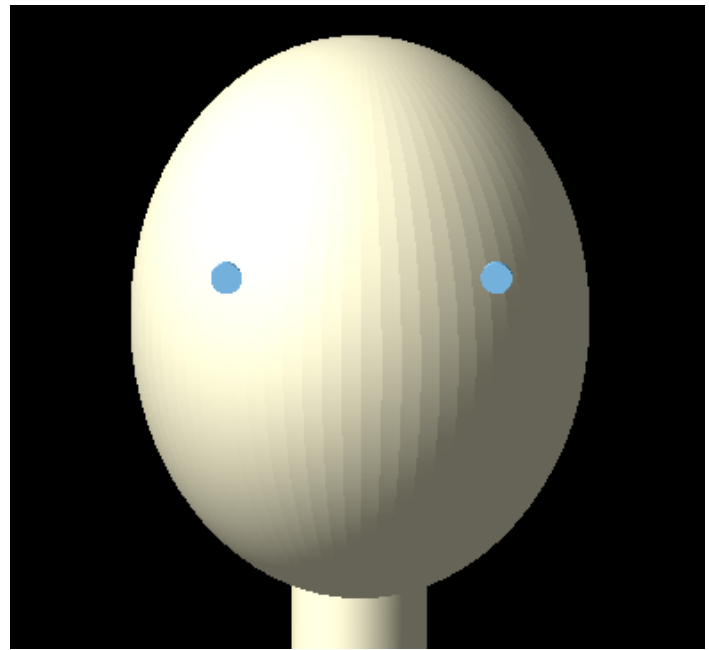


Figure 37: MIRD Phantom with Eye Lenses in GEANT4

mm thickness (Augusteyn 2010; ICRP 2002; Martins et al. 2009). The material of the eye lens is available in the GEANT4 NIST material table as “G4\_EYE\_LENS\_ICRP.” Figure 37 depicts our MIRD phantom modified with the addition of eye lenses. With this addition, our models could be easily updated to calculate eye lens dose, but it was considered outside the objectives for this study.

The MIRD phantom does not include skin, though skin is assigned a tissue weighting factor by the ICRP. The total mass of skin in adults is approximately 9% of the total body mass (Martins et al. 2009), and the total surface area of the skin is approximately  $1.9 \text{ m}^2$  for males and  $1.66 \text{ m}^2$  for females (ICRP 2002). Thus, for our 70 kg adult phantom, the total skin mass is 6.3 kg and the total surface area of the skin is  $1.78 \text{ m}^2$  (average of male and female). This is distributed over the outer volumes of the MIRD phantom (head, trunk, arms, legs). Our models could be modified to calculate skin dose, but it was considered to be outside the objectives for this study.

#### *2.3.4.3 Evaluation of MIRD Phantom Improvements*

The MIRD phantom, modified to include the active bone marrow volumes, provides improved fidelity to the overall model of the human body. To determine the value of these modifications, we performed an assessment comparing prior approaches for estimating active bone marrow dose with our direct measurement method.

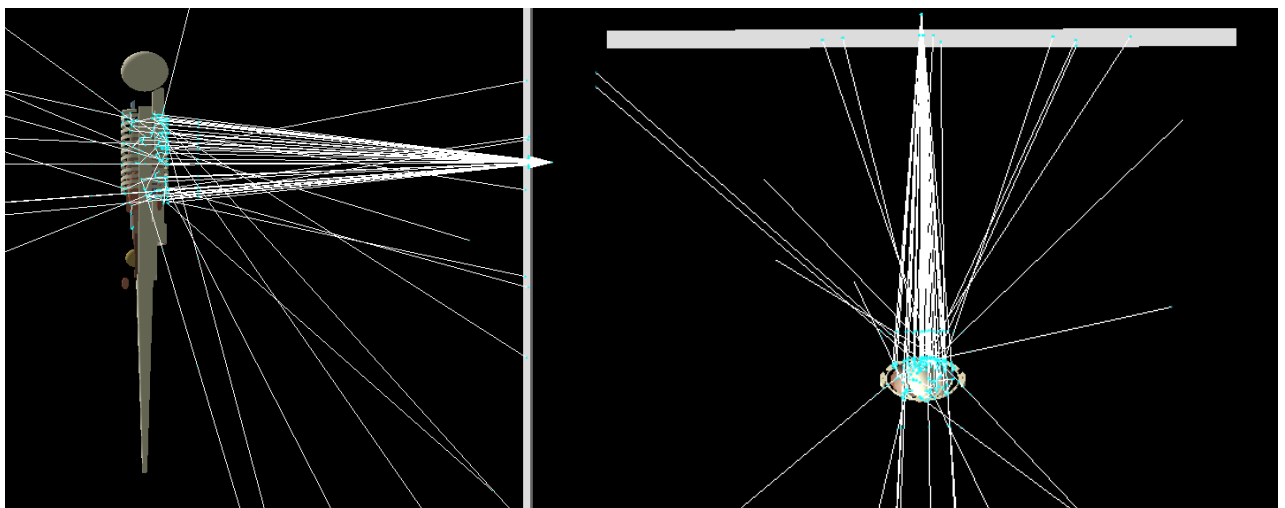
There are several prior approaches to estimating active bone marrow dose using the MIRD phantom. The method presented in the early MIRD documentation was to assume bone and marrow to be uniformly distributed within the bone volume despite known limitations:

The skeletal system represents the total skeleton and includes both bone and bone marrow. The material is considered to be homogeneously distributed in the skeleton. This is clearly a compromise due to our present inability to represent more accurately the bone and marrow spaces. (Snyder et al. 1974)

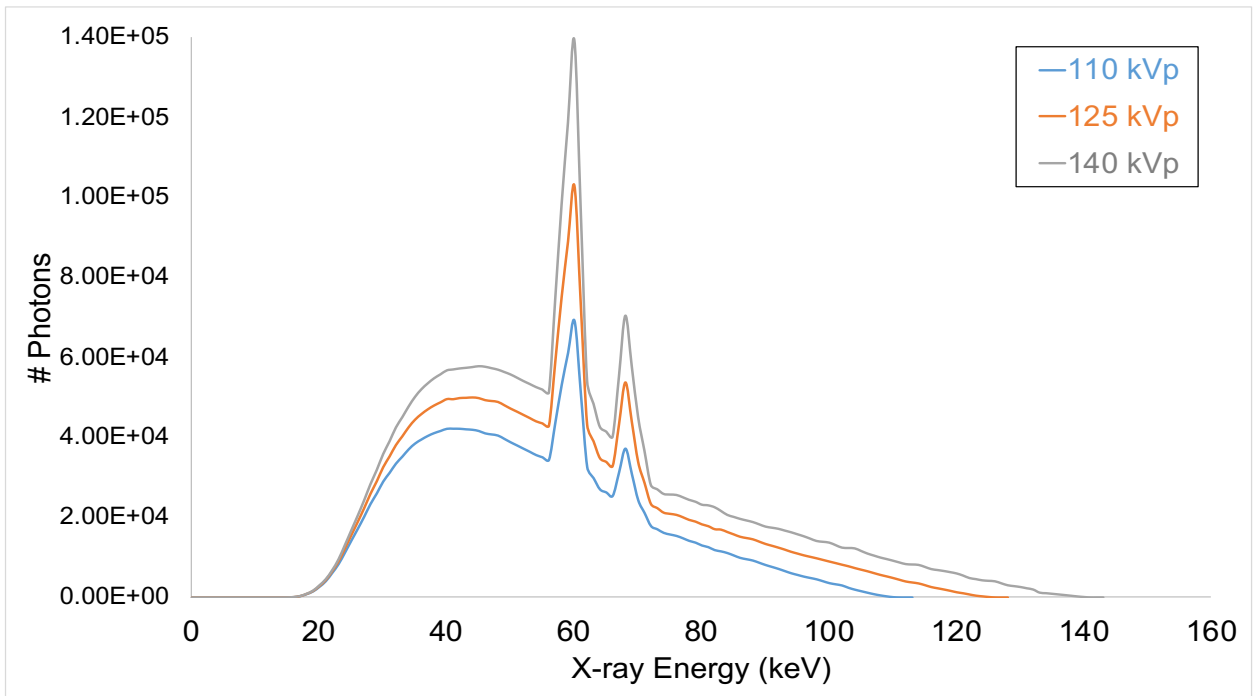
Using this method, the bone marrow dose is taken as the average skeletal dose to homogeneous bone/marrow volumes. Another option was to represent the total skeletal volumes (including bone and marrow) as solid bone and take the marrow dose to be a proportion of that total dose based on the assumed bone marrow distribution (Xu & Eckerman 2010). A third option was to calculate and average soft tissue dose from the other organs within the MIRD phantom and assign an average soft tissue dose to the total bone marrow volume (Cristy & Eckerman 1987a). The assumptions underlying each of these prior methods introduce dosimetry errors for radiation of all energies but are particularly inaccurate at energy ranges dominated by the photoelectric effect ( $<200$  keV) (Cristy & Eckerman 1987b) due to the strong dependence of that interaction on the atomic number or density of the target material. We tested and compared each of these three cases against our improved fidelity version of the MIRD phantom.

Phantoms are generally evaluated for their effectiveness in medical and radiation protection examples; therefore, we chose to compare the performance of our improved fidelity MIRD phantom in example medical and radiation protection applications as well as in a space radiation application. For the medical example, we selected a posterior-anterior (PA) orientation and simulated a chest X-ray examination in GEANT4 (see Figure 38) with a point source collimated to a 14" x 17" (35.6 cm x 43.2 cm) photon field set 70" (178 cm) behind the exit of the phantom to simulate a 72" (183 cm) source-to-image distance and a 2" (5.1 cm) patient-to-detector distance. Detector assembly and patient supports were not modeled, so backscatter from those were not accounted for. We selected with typical diagnostic X-ray energy spectra (110, 125, and 140 kVp) (see Figure 39) derived from a computational semi-empirical method (Boone & Chavez 1996; Boone & Siebert 1997) as well as monoenergetic x-rays of relevant clinical energies (20, 30, 40, 50, 60, 80, 100, 120, and 140 keV).

Note that it is possible with GEANT4 to create simulations of complete X-ray tubes and the effects of different filters (Bonifacio et al. 2005; Guthoff et al. 2012). We chose to represent a tungsten target with a standard diagnostic filter and collimator to allow us to focus on the accuracy of the bone marrow dosimetry in an example of interest to clinicians and researchers.



**Figure 38: MIRD Phantom Under PA Diagnostic X-Ray Irradiation in GEANT4**



**Figure 39: Simulated Diagnostic X-ray Spectra for Medical Example**

The clinical X-ray spectra we selected for this example (110, 125, 140 kVp) can be represented by their maximum energy (kVp) or by their equivalent energy (keV). The equivalent energy ( $h\bar{\nu}$ ) for a given kVp is determined by the total mass attenuation coefficient ( $\frac{\mu}{\rho}$ ) and the half-value layer (HVL) at that kVp (Johns & Cunningham 1983) using the equation:

$$HVL = \frac{0.693}{\frac{\mu}{\rho}}$$

For the clinical X-ray spectra we used in our medical example (110, 125, 140 kVp), the equivalent energies are provided in Table 9.

**Table 9: Equivalent Energy ( $h\bar{\nu}$ ) for Clinical X-ray Spectra**

kVp	$h\bar{\nu}$ (keV)
110	40
125	43
140	46

For the radiation protection example, we selected the anterior-posterior (AP) and isotropic (iso) configurations from the ICRP reference fields (see Figure 40) (ICRP 2012) and simulated an accidental exposure to common radioactive isotopes cesium ( $^{137}\text{Cs}$ ) and cobalt ( $^{60}\text{Co}$ ) in GEANT4 with a 200 cm x 40 cm planar source 1 m in front of the phantom (see Figure 41) or an isotropic photon source around the phantom (see Figure 42) at  $^{137}\text{Cs}$  and  $^{60}\text{Co}$  photon emission energies (0.662, 1.17, and 1.33 MeV).

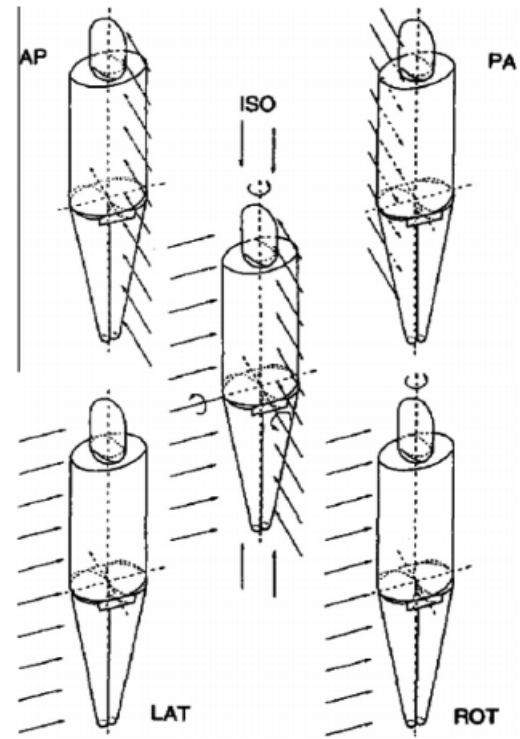


Figure 40: ICRP Reference Fields (ICRP 2012)

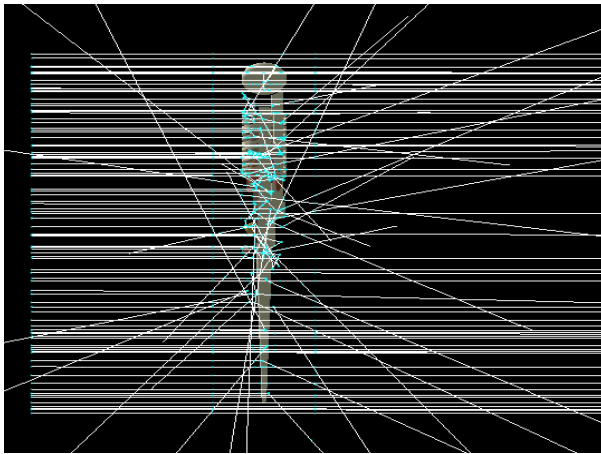


Figure 41: MIRD Phantom Under AP 662 keV Gamma Irradiation in GEANT4

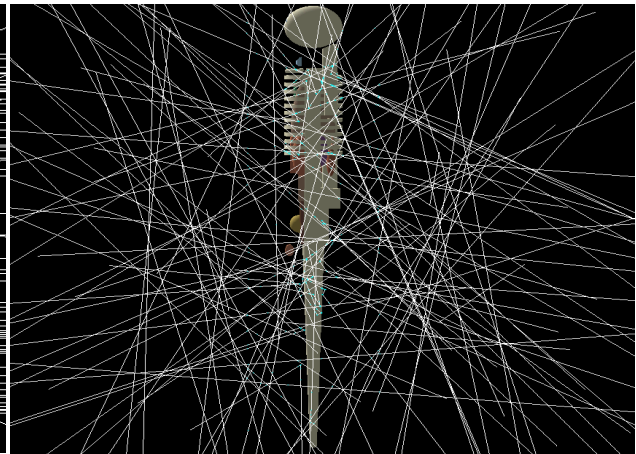
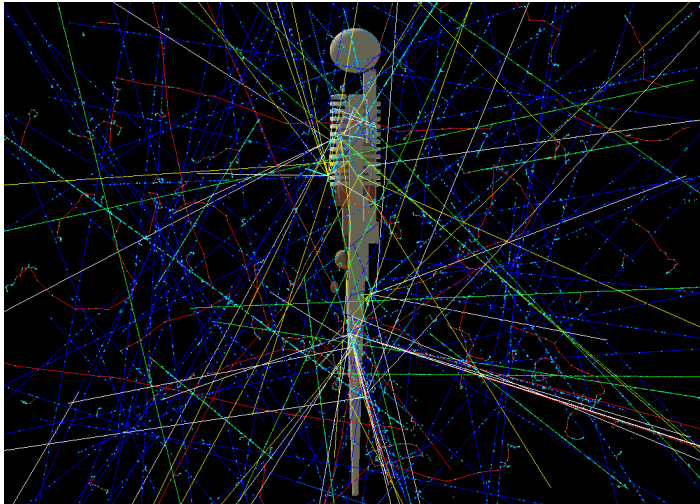


Figure 42: MIRD Phantom Under Isotropic 662 keV Gamma Irradiation in GEANT4

For the space radiation example, we selected an isotropic (iso) configuration and simulated an unshielded exposure to GCR radiation in GEANT4 (see Figure 43) at relevant energies (10 MeV, 100 MeV, 1 GeV, 10 GeV, 100 GeV, 1 TeV).



**Figure 43: MIRD Phantom Under Isotropic 1 GeV GCR Irradiation in GEANT4**

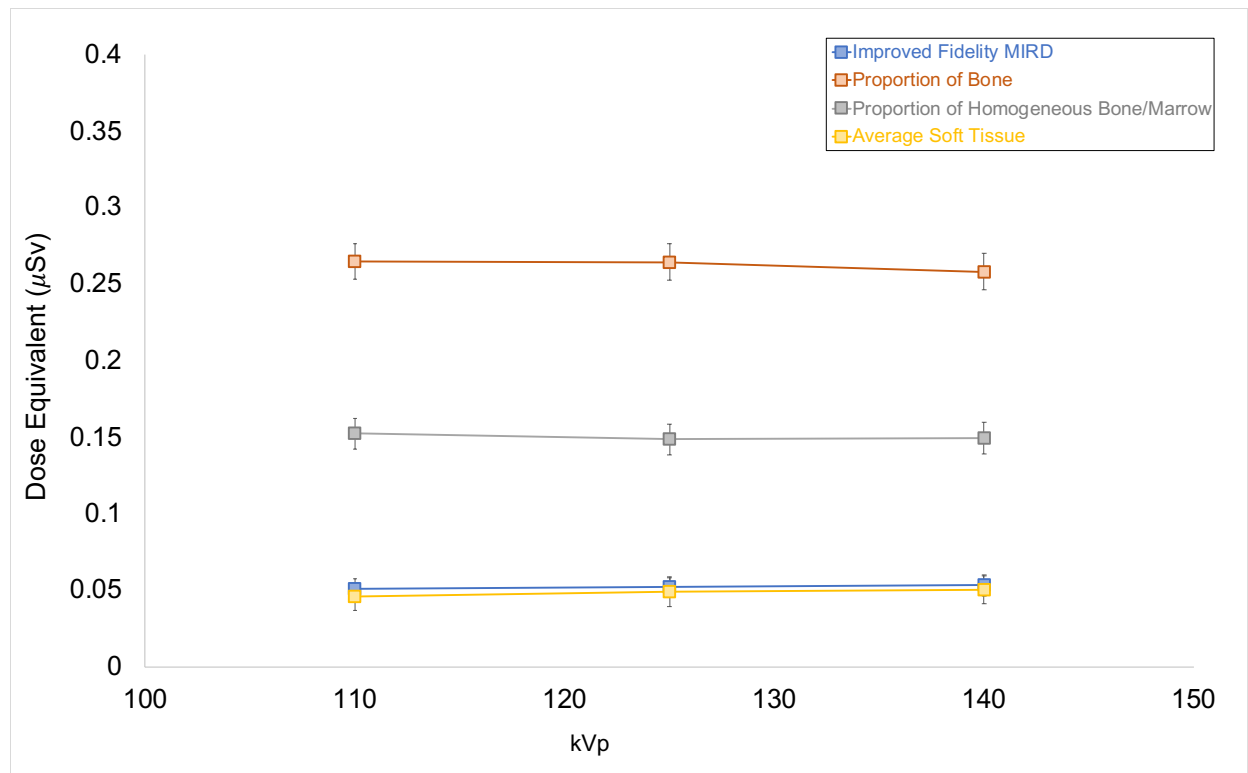
We compared our results using the improved fidelity MIRD phantom to the results from the three comparison methods (see Table 10). For the first comparison method, we represented the entire bone/marrow volumes as solid bone material. The dose to this total bone volume was calculated and a proportion of that dose assigned to the

marrow based on total body bone marrow distribution (Cristy 1981; Snyder et al. 1974, 1978). For the second comparison method, we used another prior method of bone marrow dose estimation that replaces the bone and active bone marrow volumes with a homogeneous mixture of bone and bone marrow material. The dose to this total homogeneous volume was calculated and a proportion of that dose assigned to the marrow based on total body bone marrow distribution (Cristy 1981; Xu & Eckerman 2010). For the third comparison method, we estimated the active marrow dose to be the average dose to the other soft tissues (Cristy & Eckerman 1987a). Figures 46 through 49 show the relative dose equivalent delivered to the bone marrow in each case.

For the medical example, we ran simulations in GEANT4 using the full source spectra depicted in Figure 39 for each of the four selected bone marrow dose calculation methods. The results of these simulations are provided in Figure 44. These measurements using the spectral source were also added to the monoenergetic measurements in Figure 45; this combined plot is shown in Figure 46. Error bars represent the standard error of the mean from the Monte Carlo simulations.

**Table 10: Scenario Definitions for Bone Marrow Dose Estimation**

Example Type	Radiation Field	Radiation Species	Energy	Active Bone Marrow Dosimetry Method
Medical	PA	X-ray	20, 30, 40, 50, 60, 80, 100, 120, 140 keV 110,125,140 kVp	1. Improved-fidelity MIRD dose 2. Proportion of bone dose 3. Proportion of homogeneous bone/marrow dose 4. Average soft tissue dose
Radiation Protection	AP iso	Gamma	662 keV 1.17, 1.33 MeV	1. Improved-fidelity MIRD dose 2. Proportion of bone dose 3. Proportion of homogeneous bone/marrow dose 4. Average soft tissue dose
Space Radiation	iso	Proton Alpha Heavy Ion	10, 100 MeV 1, 10, 100 GeV 1 TeV	1. Improved-fidelity MIRD dose 2. Proportion of bone dose 3. Proportion of homogeneous bone/marrow dose 4. Average soft tissue dose



**Figure 44: Bone Marrow Dose Equivalent vs. kVp for Medical (PA) Example**

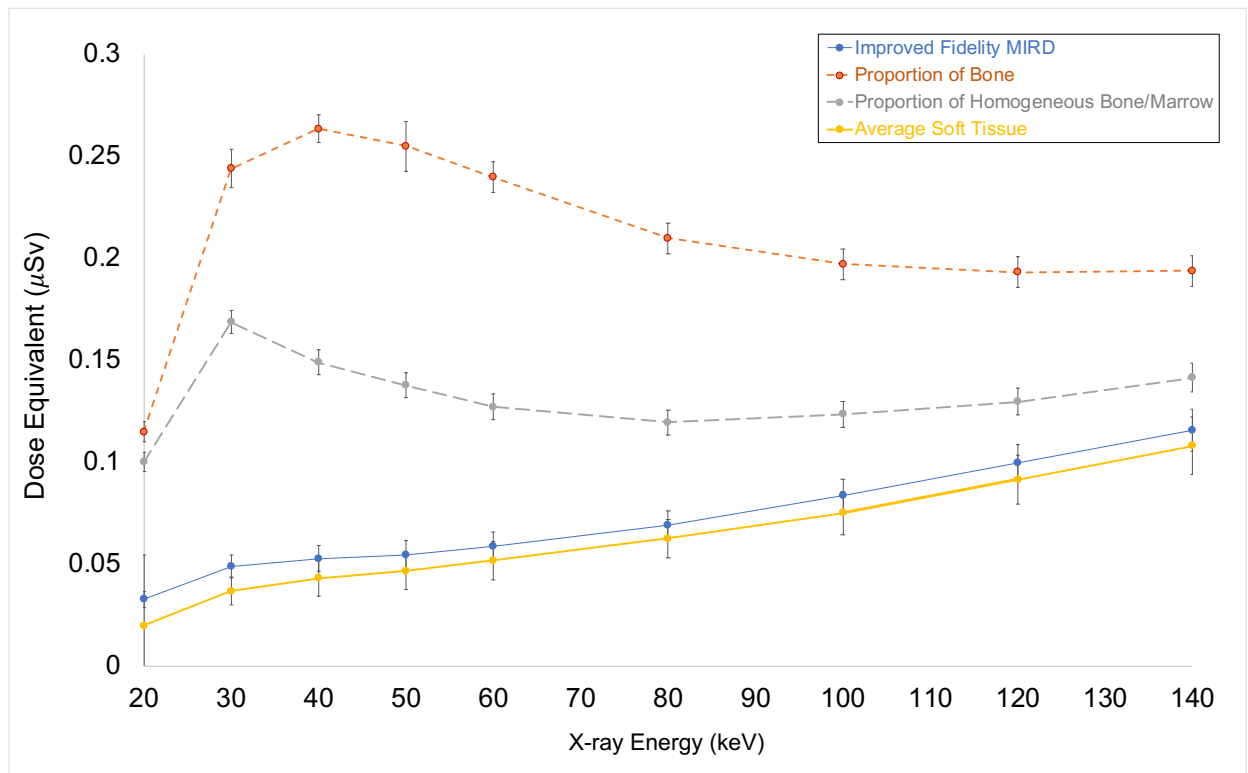


Figure 45: Monoenergetic Bone Marrow Dose Equivalent for Medical (PA) Example

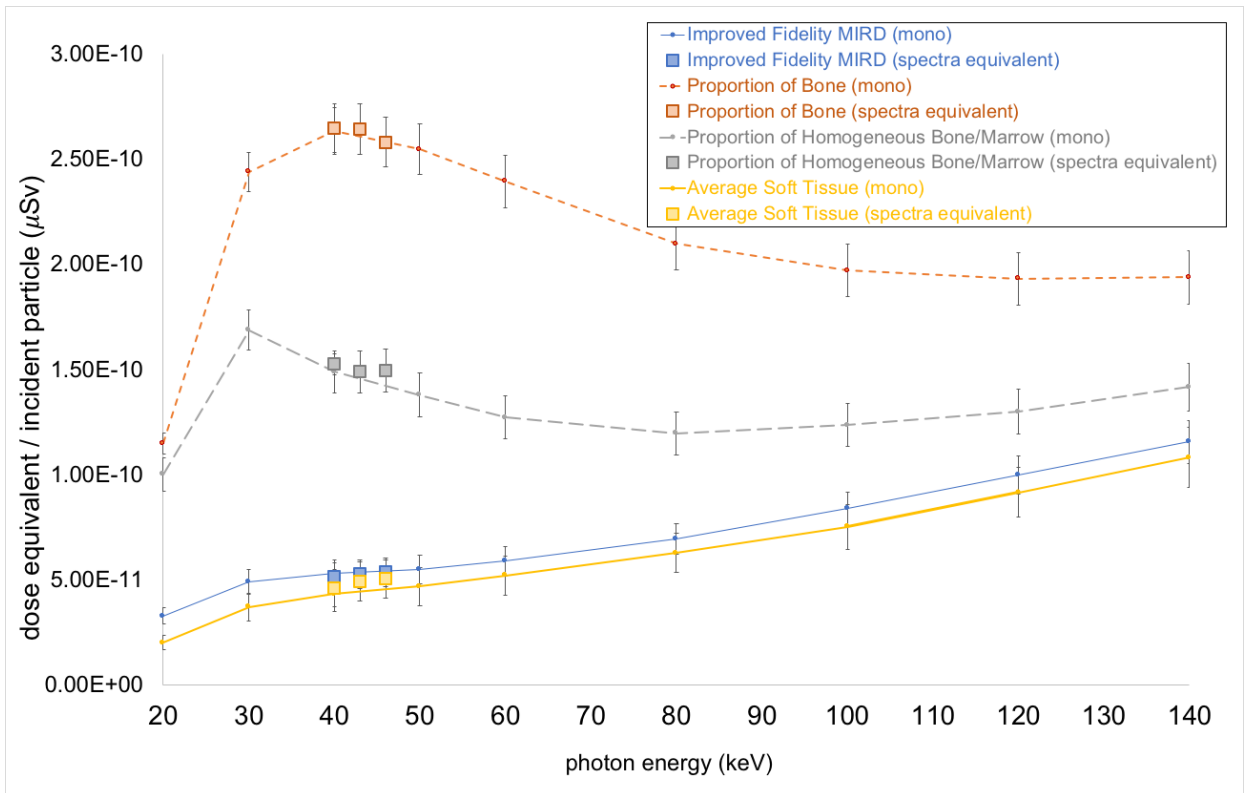


Figure 46: Monoenergetic and Spectral Bone Marrow Dose Equivalent for Medical Example

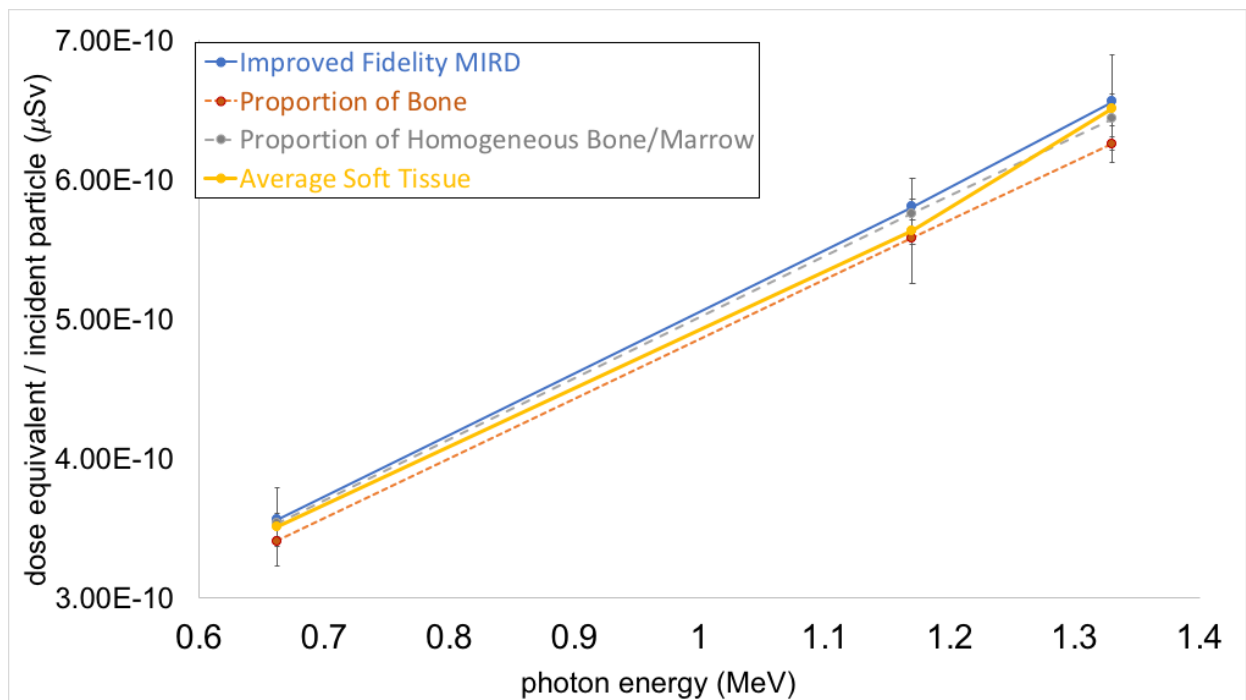


Figure 47: Bone Marrow Dose Equivalent for Radiation Protection (AP) Example

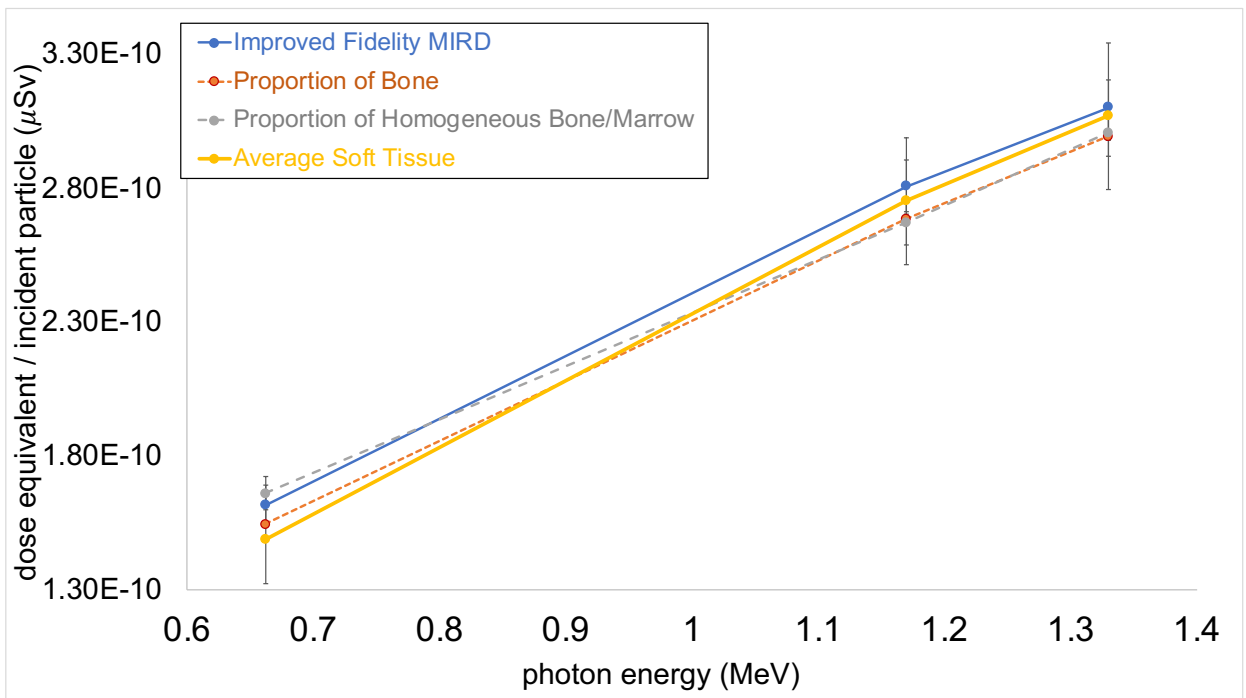


Figure 48: Bone Marrow Dose Equivalent for Radiation Protection (iso) Example

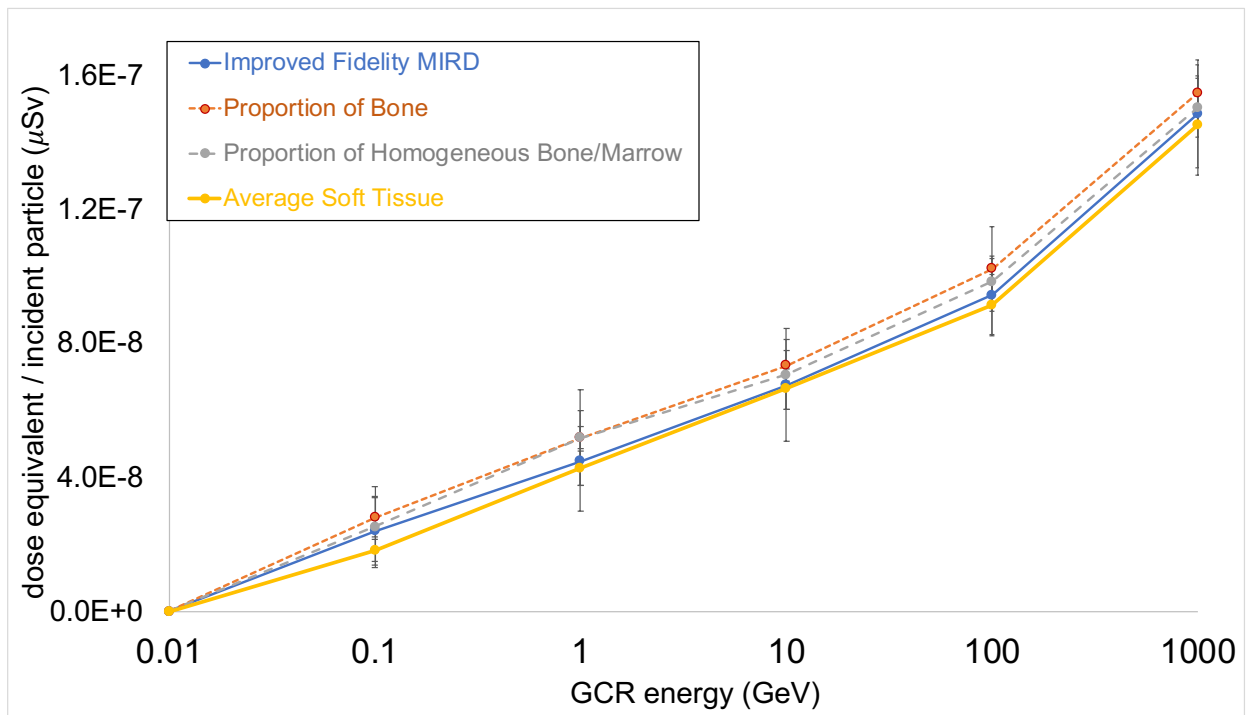
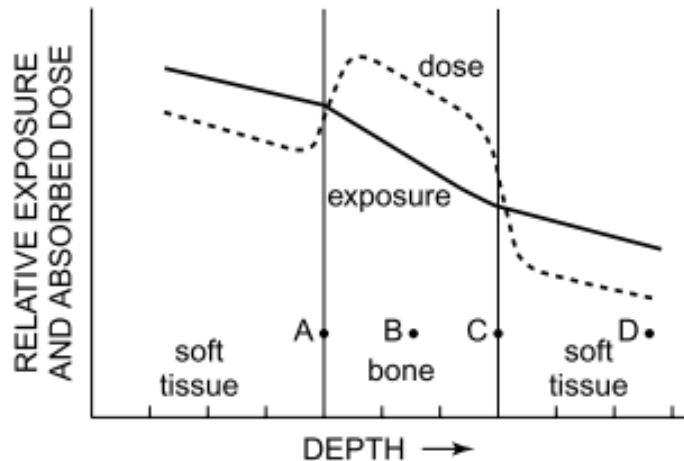


Figure 49: Bone Marrow Dose Equivalent for Space Radiation (iso) Example

Figure 46 shows that the improved fidelity bone marrow model recorded a smaller bone marrow dose than the traditional methods based on a proportion of solid bone dose or an average homogenous bone/marrow volume dose. This discrepancy is consistent with the higher atomic number ( $Z$ ) of solid bone and homogenous bone/marrow volumes, such that the lower energy X-ray interactions are dominated by the photoelectric effect which depends on target material atomic number by approximately  $Z^3$  (Sprawls 1995). Due to variations in the interaction coefficient ( $\frac{\mu}{\rho}$ ) at clinical X-ray energies, dose varies by approximately a factor of three between soft tissue and bone (Johns & Cunningham 1983). This factor is clearly seen in the difference in bone marrow dose between the four methods in Figure 46. The simple soft tissue average dose is the closest to what we measured with our improved fidelity MIRD model, though this method underestimates the bone marrow dose by between 6% at 140 keV and 30% at 20 keV. This

discrepancy is consistent with the presence of buildup in the bone tissue (see Figure 50), which creates an increase in bone marrow dose as compared to other soft tissue dose (Hendee & Ritenour 2002). The bone marrow dose observed using the homogeneous bone/marrow volume falls between the dose observed in the



**Figure 50: Effect of Buildup on Bone Marrow Dose (Hendee & Ritenour 2002); courtesy of Wiley Online Publishing**

solid bone and average soft tissue methods, which is what we would expect based on simple averaging of the effects and interaction coefficients. These results indicate that prior indirect estimation methods using the MIRD phantom can potentially overestimate the active bone marrow dose by more than a factor of three across X-ray energies, confirming limitations provided in the MIRD literature (Cristy & Eckerman 1987b). As active bone marrow dose is assigned a high tissue weighting factor in the calculation of effective dose and translation to cancer induction risk (ICRP 2007), a discrepancy of this size could create a sizeable overestimate of cancer induction risk for a given radiation dose. While our improved fidelity model is still not an exact replica of the human body, the bone marrow representation and direct dose tallying could improve the accuracy of effective dose and the risk of medical radiation-induced cancers.

Figures 47 and 48 show the difference between bone marrow dose values from the three prior estimation methods and those reported from the improved fidelity MIRD model were within the envelope defined by the error bars established as the standard deviation of the mean of the improved fidelity MIRD data. Thus, the differences in dose values are not statistically significant at gamma energies from common radioactive isotopes in the AP or iso configurations. The implication of this segment of our analysis is that the simplest bone marrow dose estimation

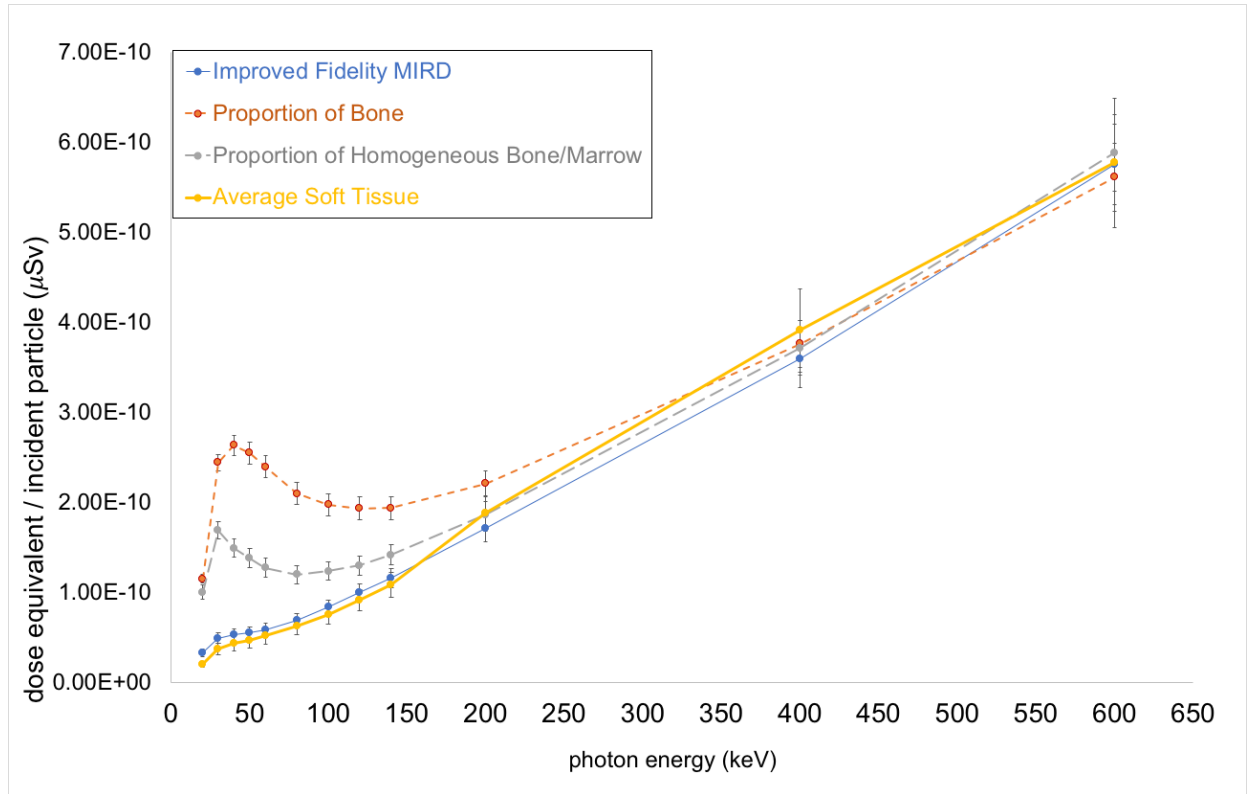
method (average soft tissue dose) can reasonably be used for radiation protection applications involving AP gamma radiation in this energy range.

Figure 49 shows that the difference in bone marrow dose is less pronounced at GCR energies than for X-ray energies, but that traditional methods can still overestimate or underestimate the bone marrow dose at GCR energies. Since active bone marrow dose is assigned a high tissue weighting factor in the calculation of effective dose and translation to cancer induction risk (ICRP 2007), a discrepancy even of this small size could create a sizeable overestimate of cancer induction risk for a given radiation dose. While our improved fidelity model is still not an exact replica of the human body, the improved bone marrow representation could improve the accuracy of effective dose and the risk of space radiation-induced cancers.

In summary, our improved fidelity MIRD phantom includes direct measurement of bone marrow dose equivalents and increases the realism of the phantom. Comparing these direct measurements to prior methods for indirectly estimating bone marrow dose equivalent across several example cases from medical, space radiation, and radiation protection applications shows interesting findings. For each scenario we studied, average soft tissue dose equivalent was closest to the directly measured bone marrow dose equivalents with some exceptions described above. Medical physicists should tailor the use of this improved fidelity MIRD phantom based on their specific application.

Because we saw different behavior in our dosimetry data at lower energies in the medical example than we saw at slightly higher energies in the radiation protection examples, our team was interested in conducting a small sanity check experiment to validate our results. Using the geometry from the medical example (PA X-ray), we ran simulations at test energies of 200, 400, and 600 keV. The results of these simulations were appended to the results from the lower

energies, and this combined data is displayed in Figure 51. This figure shows that the differences in dose observed at lower energies in the medical example do indeed converge at the medium energies leading into those used in the radiation protection examples.



**Figure 51: Medical Example with Additional Test Energies**

#### 2.3.4.4 Dosimetry in GEANT4

The GEANT4 toolkit (Agostinelli et al. 2003) is well-validated for use in models including space radiation (Geng et al. 2015; Ivantchenko et al. 2012; Truscott et al. 2000). Environmental models and dosimetry methods are user-defined but are often based on example codes included with the GEANT4 software package. For our project, we leveraged several example codes including `/examples/advanced/human_phantom` and `/examples/advanced/purging_magnet`. For dosimetry, however, our team developed our own method for scoring organ doses and calculating effective dose. Each organ was given a unique ID number and the numbers were then used to record the energy deposited in each volume during a simulation. Organ doses were

calculated from energy deposited, mass, and LET (see Appendix A9 for the complete Excel organ dose calculations).

#### *2.3.4.5 Risk Analysis*

Amongst the general population, cancer is responsible for 22% of all deaths (Siegel et al. 2019). Projection of lifetime risk of radiation-induced cancer for astronauts on an interplanetary mission is challenging due to the uncertainties involved (ICRP 2013; Wilson 2000). The terrestrial radiation-induced cancer risk estimates are derived primarily from follow-up studies of Japanese atomic bomb survivors, workers in the nuclear power and nuclear weapons complex, nuclear and x-ray accident victims, and members of the public exposed to fallout from atmospheric nuclear tests, or environmental sources such as radon (UNSCEAR 2017; NRC 2006; NRC 1998; Carlsen et al. 2001; Peterson & Miller 2008; Peterson & Cucinotta 1999). Aside from A-bomb survivors, Chernobyl liquidators, and radiation accident victims, the majority of nuclear and medical workers have been exposed to long-term chronic doses of low-LET radiation, which differs from the chronic high LET radiation in space. However, dose and dose rate effectiveness factors (DDREF) exist to attempt to translate between the two (NCRP 1997; Peterson & Cucinotta 1999; Peterson & Kovyreshina 2015). Further, the population of astronauts differs from the general population in that they are generally healthy, normal weight nonsmokers, and thus should have a reduced risk of certain cancers due to the Healthy Worker Effect (HWE) (Peterson et al. 1993; Peterson & Kovyreshina 2015).

The linear-non-threshold (LNT) hypothesis indicates that even very small radiation doses may cause detrimental health effects and that the probability of these effects is proportional to the absorbed dose (NRC 2006; Cohen 2002; ICRP 2007). Alternative hypotheses include supra-linear, linear with lower threshold, and radiation hormesis, but the “LNT hypothesis is considered to be the best approach to managing risk from radiation exposure” (ICRP 2007).

Given the uncertainties involved and considering the unique population of astronauts, NASA and other research agencies have sponsored studies on the risks of space radiation specifically for this population (Cucinotta et al. 2012b, 2013; Kim et al. 2011; NRC 2012; Peterson & Kovyreshina 2015). Effective dose has the unique characteristic that it directly relates to the risk of exposure-induced death (REID). NASA's current limit for REID is 3% at a 95% confidence interval (CI). Our method described previously for calculation of effective dose including 95% CI for carcinogenesis risk estimates is backed by prior studies and literature (NRC 2008; Setlow 1999). However, there are several other modifications to the standard cancer risk models that could be employed for future iterations of this research for comparison.

### *2.3.5 Simulation Methodology*

For the minimal shielding case, we ran simulations with all three phantom types (water, MIRD male, MIRD female) at both solar min and solar max environmental conditions. These six cases serve as the baseline for all further scenarios in this study. From these we immediately discovered that the dose equivalent to the water phantom fell between the effective dose for MIRD male and MIRD female phantoms. This is a good check on our methodology, showing that in a uniform radiation field the dose to a water phantom of the same mass as our MIRD phantoms would provide an approximately average dose calculation. Yet because we wish to delineate between male and female cases in our study, the water phantom was not used beyond this calibration check.

For each remaining scenario, we ran four simulations to calculate GCR effective dose: one for each phantom type (MIRD male, MIRD female) with both solar min and solar max environment conditions. Each of these simulations included three runs: proton ( $Z=1$ ) GCRs, alpha ( $Z=2$ ) GCRs, and all remaining ( $Z=3$  to  $Z=26$ ) GCRs. The reason to split the GCRs into three runs is that protons and alphas dominate the GCR spectrum, accounting for over 95% of the total

fluence. Had we initiated all GCRs together in one run, we would have orders of magnitude more protons and alphas than we need in order to get the number of other GCR ions we need for good statistics. Therefore, for the sake of efficiency, we split out the protons and alphas from the overall GCR spectrum and ran those separately.

For each scenario, we ran two simulations to calculate SPE effective dose: one for an average SPE and one for a worst-case SPE. These SPE spectra do not typically vary with the solar cycle; rather the frequency of SPEs varies substantially with solar cycle. While worst-case SPEs are not expected to occur at all during solar min, one of these is reasonably expected per mission at solar max. Average SPEs are expected to occur at a rate of approximately 2 per year at solar min and approximately 20 per year at solar max.

To calculate the final effective dose values for each scenario, we scaled the three GCR dose calculations to 500 and 700 days and summed them together, then we scaled the average and worst-case SPE dose calculations based on the solar cycle and summed them together. Error propagation was conducted at each step of the calculation. We kept a separate Excel workbook for each scenario (see Appendix A9 for the effective dose calculation spreadsheet we developed for use in each scenario) and each included a summary data sheet such as the example in Table 11 (see Appendix A10 for summary data sheets for each scenario). The total number of particle histories for each scenario can be found in Table 12.

Table 11: Example of a Summary Sheet Recorded for Each Scenario

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Aluminum							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0401	0.0011	2.81%	0.0305	0.0026	8.49%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.4416	0.0115	2.62%	0.3462	0.0276	7.96%
		Total SPE Dose for Mission	1.9811	0.0135	0.68%	1.5159	0.0319	2.10%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.8653	0.0180	2.08%	0.1029	0.0030	2.93%
		GCR H (Z=1) Dose	1.2997	0.0513	3.94%	0.8079	0.0280	3.47%
		GCR He (Z=2) Dose	0.2609	0.0103	3.95%	0.1643	0.0116	7.07%
		Total GCR Dose for Mission	2.4259	0.0553	2.28%	1.0750	0.0305	2.83%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	4.4070	0.0569	1.29%	2.5909	0.0441	1.70%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (Sv/d)	$\sigma$ (Sv/d)	SPE (Sv/d)	$\sigma$ (Sv/d)	TOTAL (Sv/d)	$\sigma$ (Sv/d)
		Active Bone Marrow	0.00393177	0.00033337	0.00421171	0.00040112	0.00814348	0.00052157
		Brain	0.00246323	0.00003074	0.00380120	0.00008246	0.00626444	8.8003E-05
		Thyroid	0.00171903	0.00018958	0.00034556	0.00014402	0.00206459	0.00023808
		Lungs	0.00318861	0.00010087	0.00292258	0.00010294	0.00611119	0.00014412
		Heart	0.00501226	0.00055316	0.00242949	0.00011639	0.00744174	0.00056527
		Thymus	0.00801567	0.00080797	0.00184189	0.00032259	0.00985757	0.00086998
		Stomach	0.00632231	0.00057046	0.00188598	0.00009624	0.00820829	0.00057852
		Large Intestine	0.00250416	0.00004111	0.00214338	0.00018639	0.00464754	0.00019087
		Small Intestine	0.00244641	0.00003381	0.00197076	0.00012903	0.00441717	0.00013338
		Liver	0.00261382	0.00003230	0.00239159	0.00012355	0.00500541	0.0001277
		Kidneys	0.00257791	0.00006303	0.00300418	0.00013176	0.00558209	0.00014606
		Adrenals	0.00240976	0.00028203	0.00595249	0.00081898	0.00836226	0.00086618
		Pancreas	0.00225399	0.00012615	0.00177713	0.00017684	0.00403112	0.00021723
		Spleen	0.00258906	0.00009559	0.00105282	0.00009272	0.00364188	0.00013317
		Bladder	0.00240931	0.00013599	0.00131072	0.00021353	0.00372003	0.00025315
		Uterus	0.00252418	0.00012754	0.00342035	0.00028080	0.00594452	0.00030841
		Ovaries	0.00400550	0.00060153	0.00366234	0.00092544	0.00766784	0.00110375
		Breasts	0.00203844	0.00004689	0.00384426	0.00012046	0.00588270	0.00012927

### 3 Results and Discussion

#### 3.1 Control Cases

In this section and the following section on magnetic shielding cases, GEANT4 screenshots are provided for visual representation of the various simulation scenarios in this project. In these images, incident space radiation tracks are color-coded corresponding to the ion species. Because there are a limited number of colors available for the visualization of particle tracks, ions (heavy nuclei stripped of all electrons) are grouped by mass:

- Proton ( $^1\text{H}$ ) = blue
- Alpha ( $^4\text{He}$ ) = green
- Lithium ( $^7\text{Li}$ ) ion = green
- Beryllium ( $^9\text{Be}$ ) ion = green
- Boron ( $^{11}\text{B}$ ) ion = green
- Carbon ( $^{12}\text{C}$ ) ion = green
- Nitrogen ( $^{14}\text{N}$ ) ion = green
- Oxygen ( $^{16}\text{O}$ ) ion = green
- Fluorine ( $^{19}\text{F}$ ) ion = yellow
- Neon ( $^{20}\text{Ne}$ ) ion = yellow
- Sodium ( $^{23}\text{Na}$ ) ion = yellow
- Magnesium ( $^{24}\text{Mg}$ ) ion = yellow
- Aluminum ( $^{27}\text{Al}$ ) ion = yellow
- Silicon ( $^{28}\text{Si}$ ) ion = yellow
- Phosphorus ( $^{31}\text{P}$ ) ion = magenta
- Sulfur ( $^{32}\text{S}$ ) ion = magenta
- Chlorine ( $^{35}\text{Cl}$ ) ion = magenta

- Argon ( $^{40}\text{Ar}$ ) ion = magenta
- Potassium ( $^{39}\text{K}$ ) ion = magenta
- Calcium ( $^{40}\text{Ca}$ ) ion = magenta
- Scandium ( $^{45}\text{Sc}$ ) ion = red
- Titanium ( $^{48}\text{Ti}$ ) ion = red
- Vanadium ( $^{51}\text{V}$ ) ion = red
- Chromium ( $^{52}\text{Cr}$ ) ion = red
- Manganese ( $^{55}\text{Mn}$ ) ion = red
- Iron ( $^{56}\text{Fe}$ ) ion = red

Neutron, gamma, electron, positron and other particle species are hidden from the visualization, though their interactions and dose deposition are recorded. Interaction points are colored cyan.

### 3.1.1 Minimal Shielding

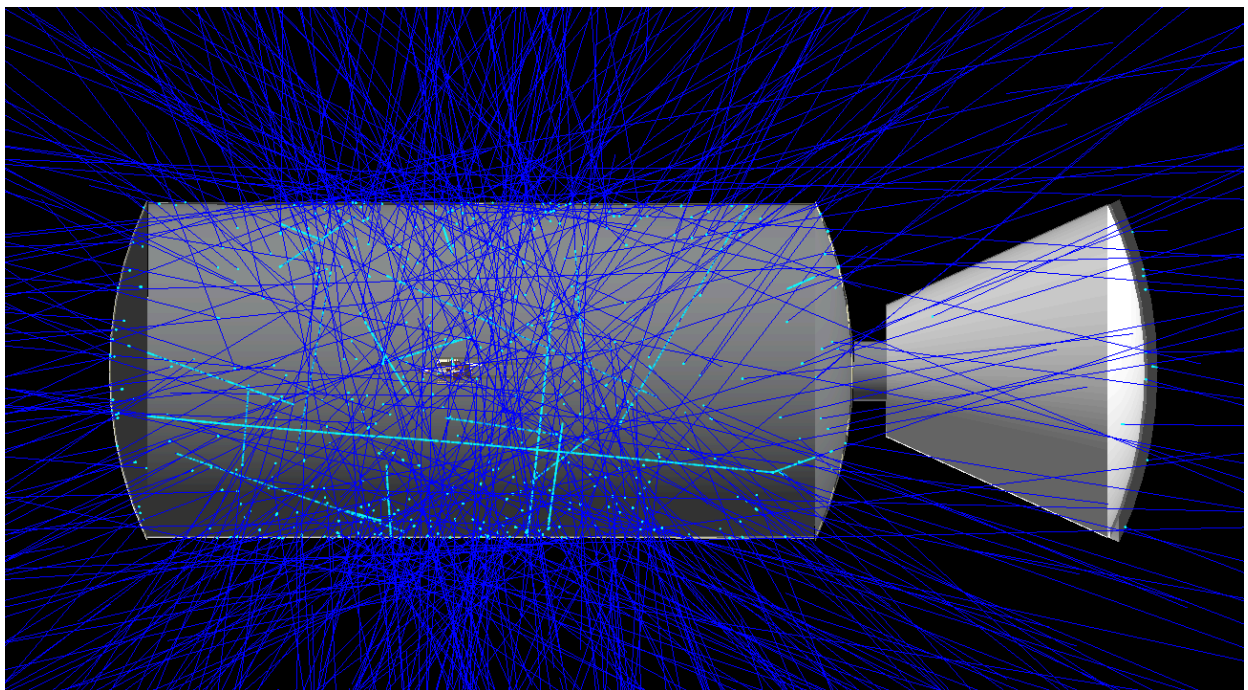
For this project, our baseline control case is a minimal shielding configuration. In this configuration, the only radiation shielding provided to the astronauts inside the spacecraft is due to the spacecraft itself. The passive shielding provided by a typical spacecraft wall is on the order of 5 g/cm<sup>2</sup> from layers of aluminum and thermal or debris blanketing. Details on this shielding configuration and how we have modeled it in GEANT4 can be found in Section 2.3.3.1.

Screenshots from the simulations of the minimal shielding case under both SPE and GCR incident radiation are provided in Figure 52 and Figure 53. Here we can see that the incident radiation passes through to the interior of the habitable volume almost unimpeded by the exterior wall of the spacecraft. Lower energy SPE protons (blue) are stopped by the spacecraft wall, but

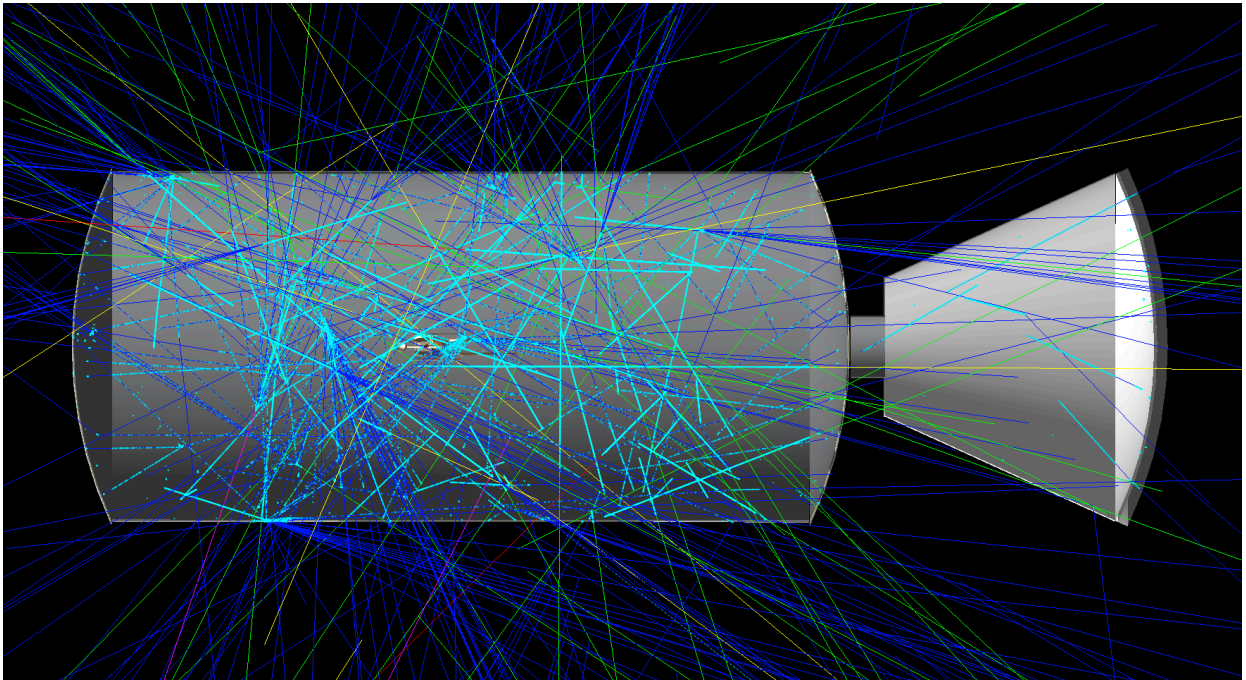
higher energy SPE protons (blue) and almost all GCR particles (blue, green, yellow, magenta, red) pass through and can deliver dose to the astronaut.

Results from the minimal shielding cases on a 700-day mission are provided in Figure 54. From this figure, it is immediately apparent that none of the minimal shielding scenarios are currently feasible within permissible dose limits. In every case, the total mission effective dose (dose equivalent for water phantom) exceeds the permissible limits. The error bars in this figure represent  $\pm 2\sigma$  ( $\sigma$ : standard error), or a 95% CI, as required by NASA (Cucinotta et al. 2012b).

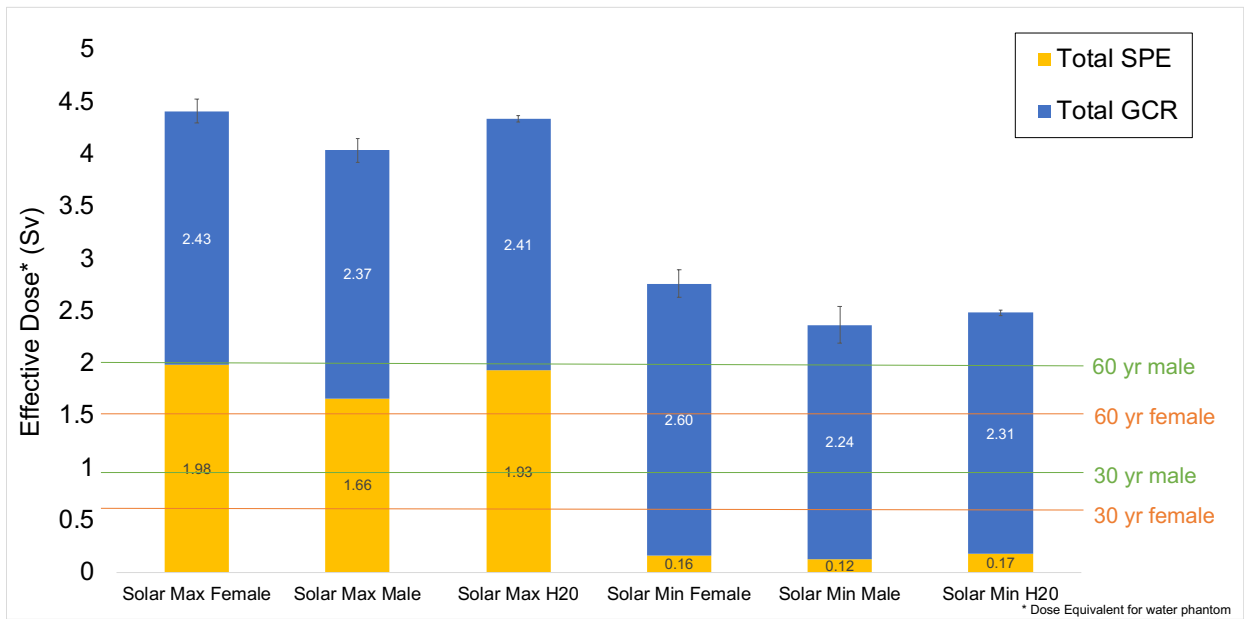
It is also apparent from Figure 54 that the difference in frequency of occurrence of SPEs at solar max and solar min introduces an approximately 10-fold difference in the effective dose (dose equivalent for water phantom) due to SPE protons. GCR dose is relatively unaffected by the change in solar cycle, though the spectrum is slightly modified by Forbush modulation at solar max.



**Figure 52: Minimal Shielding in SPE Environment**



**Figure 53: Minimal Shielding in GCR Environment**



**Figure 54: Total Mission (700 d) Dose for Minimal Shielding Cases**

Finally, Figure 54 shows that for both solar cycle cases, the dose equivalent calculated using the water phantom is intermediate in value to the effective doses calculated using the modified MIRD male and female phantoms. This result suggests that our GEANT4 dosimetry using the modified MIRD phantoms is reporting reasonable values.

### 3.1.2 Water Shielding

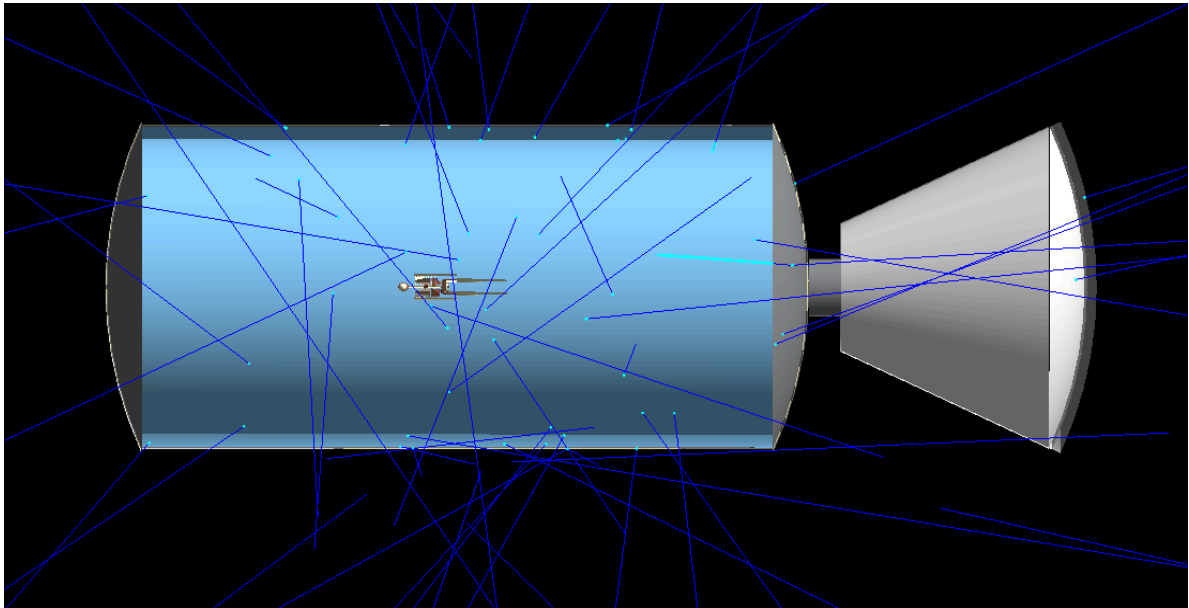
For this project, our alternate control case is a water shielding configuration. In this configuration, passive radiation shielding is provided to the astronauts inside the spacecraft from the spacecraft as well as a 20 cm lining of water. The passive shielding provided by a typical spacecraft wall is on the order of 5 g/cm<sup>2</sup> from layers of aluminum and thermal or debris blanketing, and the additional shielding provided by the water layer is approximately 20 g/cm<sup>2</sup>. Details on this shielding configuration and how we have modeled it in GEANT4 can be found in Section 2.3.3.2.

Screenshots from the simulations of the water shielding case under both SPE and GCR incident radiation are provided in Figure 55 and Figure 56. Here we can see that SPE protons (blue) are almost entirely stopped by the passive shielding, but almost all GCR particles (blue, green, yellow, magenta, red) still pass through and can deliver dose to the astronaut. Hence, water shielding is highly effective at shielding SPE proton radiation, but ineffective at shielding GCR heavy ion radiation.

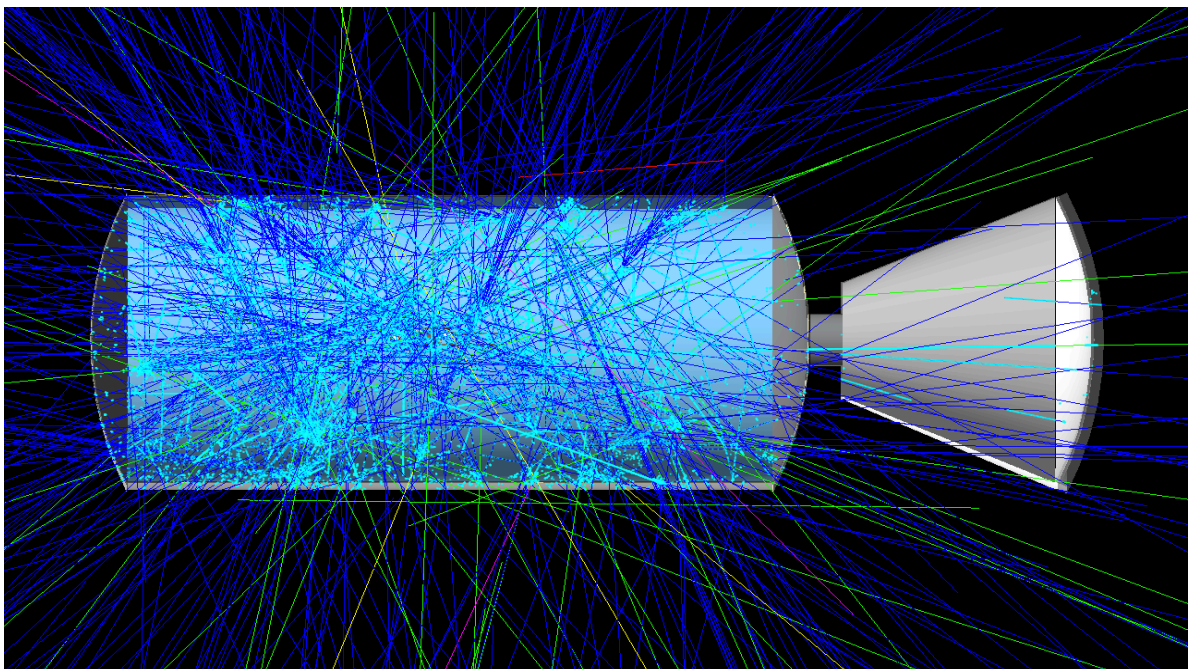
Results from the water shielding cases are provided in Figure 57. From this figure, it is apparent that none of the water shielding scenarios are compliant with permissible dose limits. In every case, the total mission effective dose exceeds the permissible limits by ~10 – 100%. The error bars in this figure represent  $\pm 2\sigma$ , or a 95% CI, as required by NASA (Cucinotta et al. 2012b).

It is also apparent from Figure 57 that the water shielding is highly effective at reducing the dose from SPEs, providing approximately 90% reduction in SPE effective dose (dose equivalent for water phantom) in each case. It is also apparent that the difference in frequency of occurrence of SPEs at solar max and solar min continues to contribute an approximately 10-fold difference

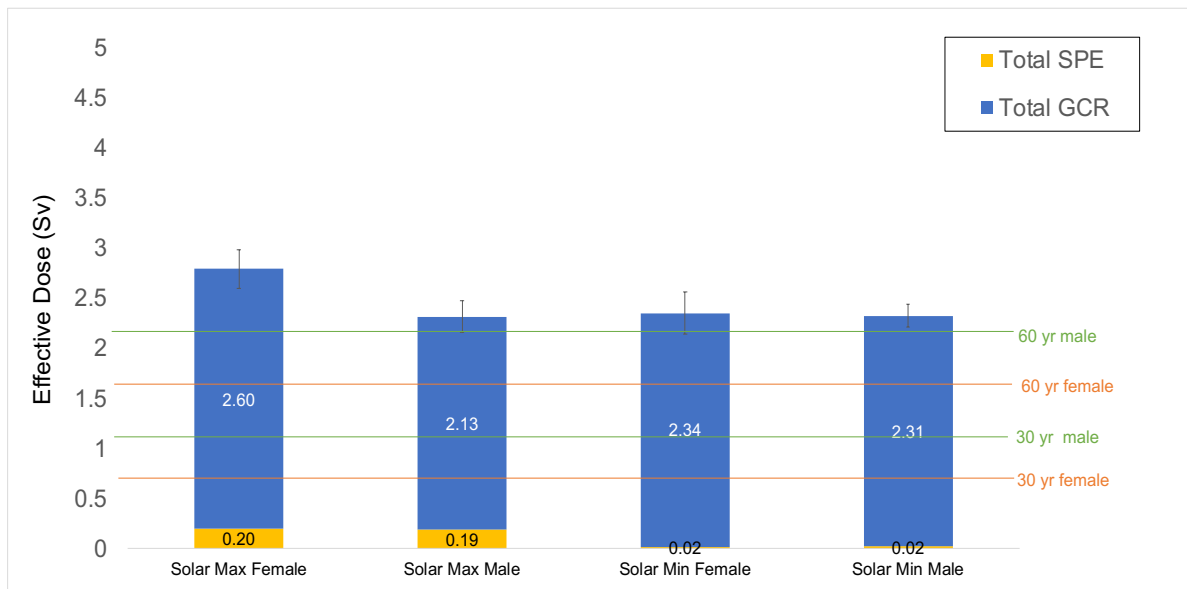
in the effective dose (dose equivalent for water phantom) from SPE protons. GCR dose is mostly unaffected by the change in solar cycle, though the spectrum is slightly modified by Forbush modulation at solar max. GCR dose is also mostly unaffected by the presence of the water shielding, with the total mission GCR dose remaining fairly constant across for both solar cycle positions.



**Figure 55: Water Shielding in SPE Environment**



**Figure 56: Water Shielding in GCR Environment**



**Figure 57: Total Mission (700 d) Dose with 20 cm Water Shielding**

The results from both the minimal and water shielding configurations show that even though simple passive shielding can almost eliminate the risk of high SPE doses, the GCR dose requires a different shielding approach.

## 3.2 Magnetic Shielding Cases

### 3.2.1 Zero Magnetic Field

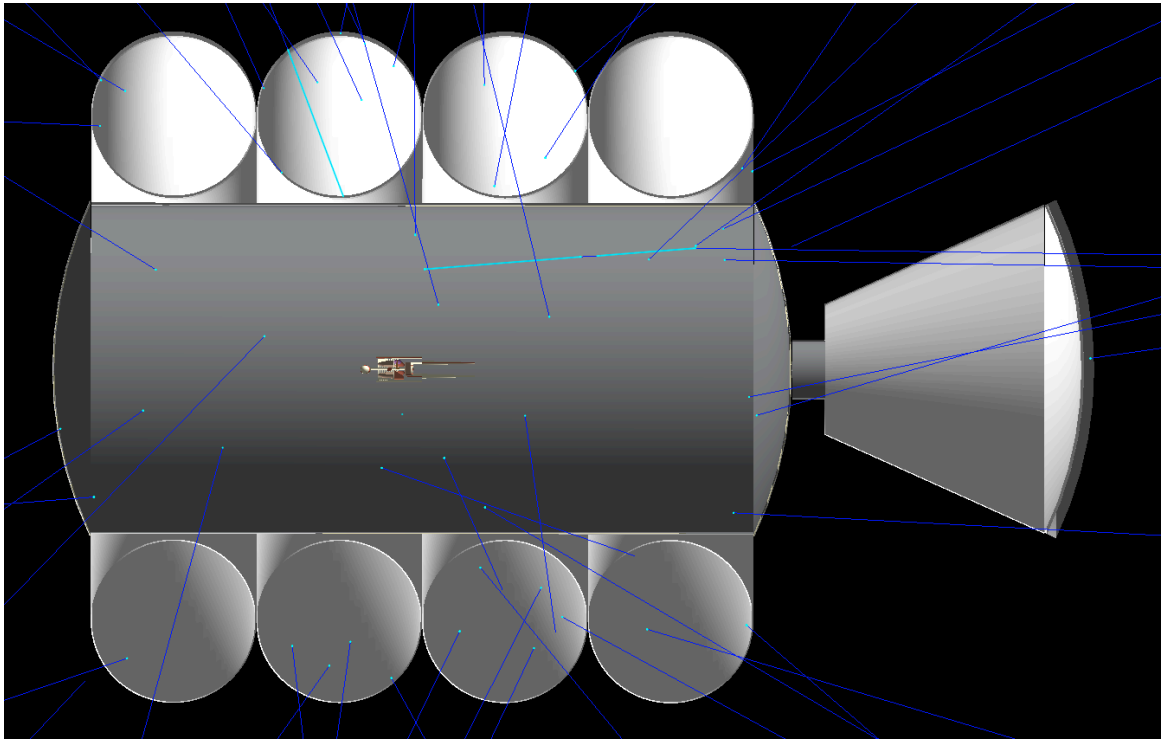
In this project, we built and analyzed several active magnetic shielding configurations to the highest fidelity as possible with our current knowledge, resources, and tools. Our project is one of the first to simulate the structure, mass, and associated fringe fields of the active magnetic shielding system within our Monte Carlo models. While the goal of this project is to compare the active magnetic shielding performance in reducing dose to the astronauts inside the spacecraft, we cannot ignore the fact that the active magnetic shielding structures themselves offer an amount of passive shielding. In order to isolate the dose-reducing effects of the active magnetic shielding from the passive shielding effects of the system's structure, we ran each magnetic shielding case also with the magnetic field switched off for comparison.

### 3.2.2 Toroidal Magnetic Shielding

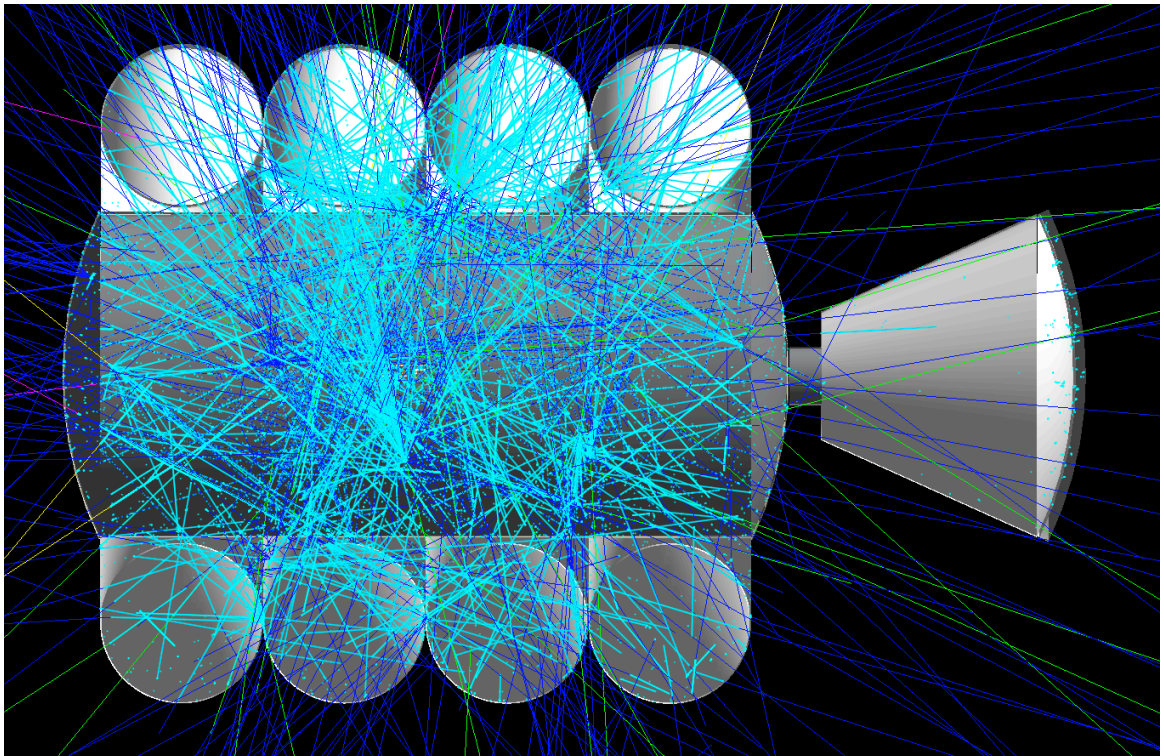
Our first experimental magnetic shielding case is a toroidal shielding configuration. In this configuration, radiation shielding is provided to the astronauts inside the spacecraft from four 2.5 m cross-sectional diameter superconducting toroid magnets. Since toroidal magnetic fields are confined, this configuration includes negligible fringe field acting at a distance. Details on the geometry of this shielding configuration and how we modeled it in GEANT4 can be found under Shielding Configurations in Section 2.3.3.3.

Results from the toroidal shielding cases are provided in Figure 88, Figure 89, Figure 90, and Figure 91. From these figures, it is apparent that the effectiveness of the toroidal magnetic shielding configuration is highly dependent on the magnetic field strength. None of the toroidal magnetic shielding cases meet the strictest effective dose limit (30-year-old female astronaut). For older and/or male astronauts, the total mission effective dose is within permissible limits for the 7 T toroidal shielding case. The error bars in these figures represent  $\pm 2\sigma$ , or a 95% CI, as required by NASA (Cucinotta et al. 2012b).

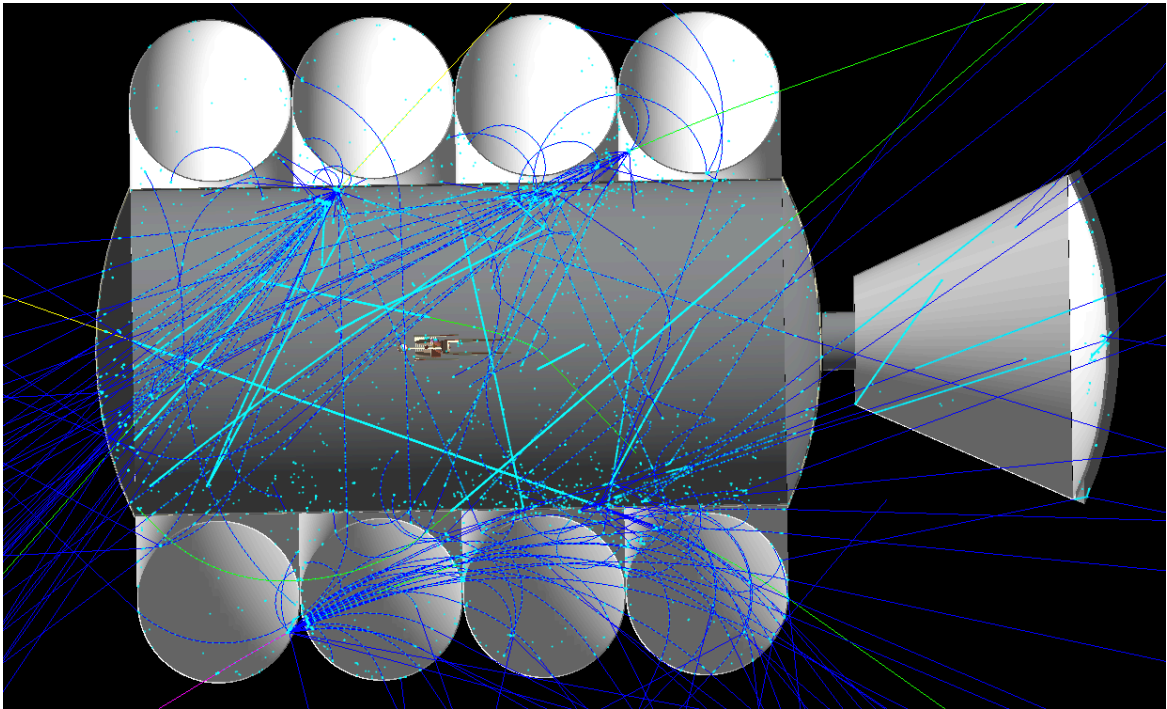
Screenshots from the simulations of the toroidal shielding case under both SPE and GCR incident radiation are provided in Figure 58, Figure 59, Figure 60, Figure 61, and Figure 62. Here we can see that SPE protons (blue) are almost entirely stopped by the passive shielding effect of the magnetic shielding structure, but GCR particles (blue, green, yellow, magenta, red) are still able pass through and can deliver dose to the astronaut depending on the magnetic field. As the magnetic field increases, the radius of curvature of the particle deflection decreases and particles of increasingly higher rigidity (particle momentum divided by its charge) are turned away from the habitable volume.



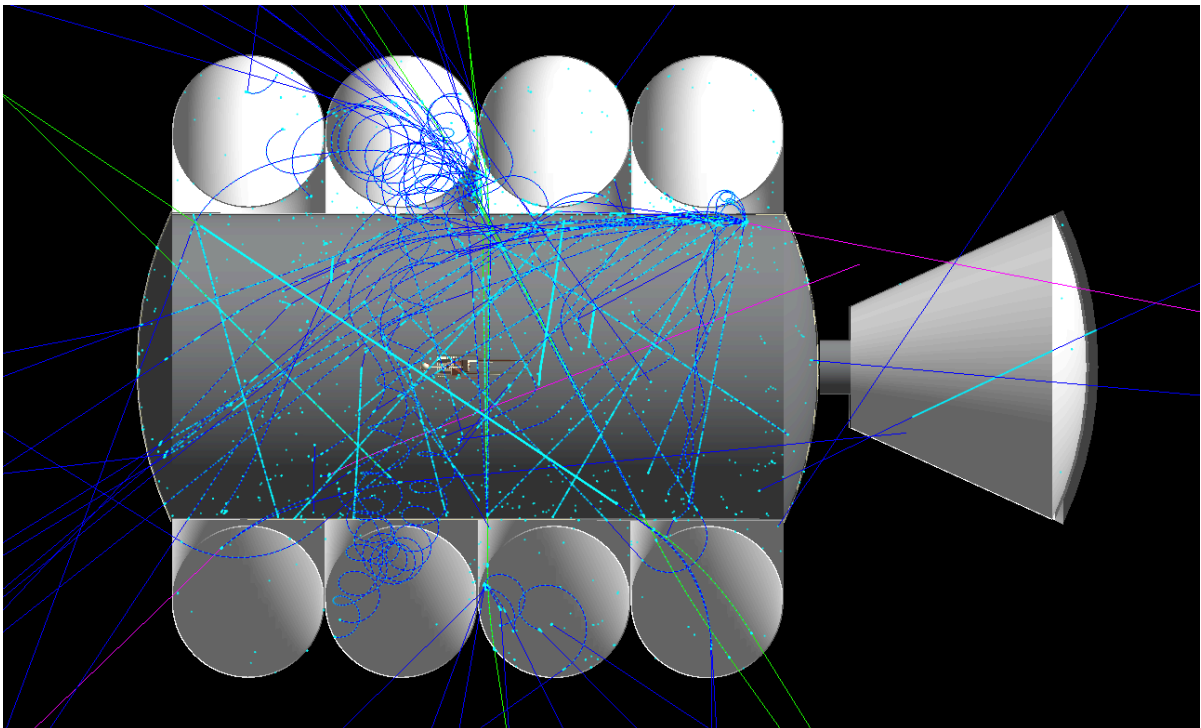
**Figure 58: 0 T Toroidal Magnetic Shielding in SPE Environment**



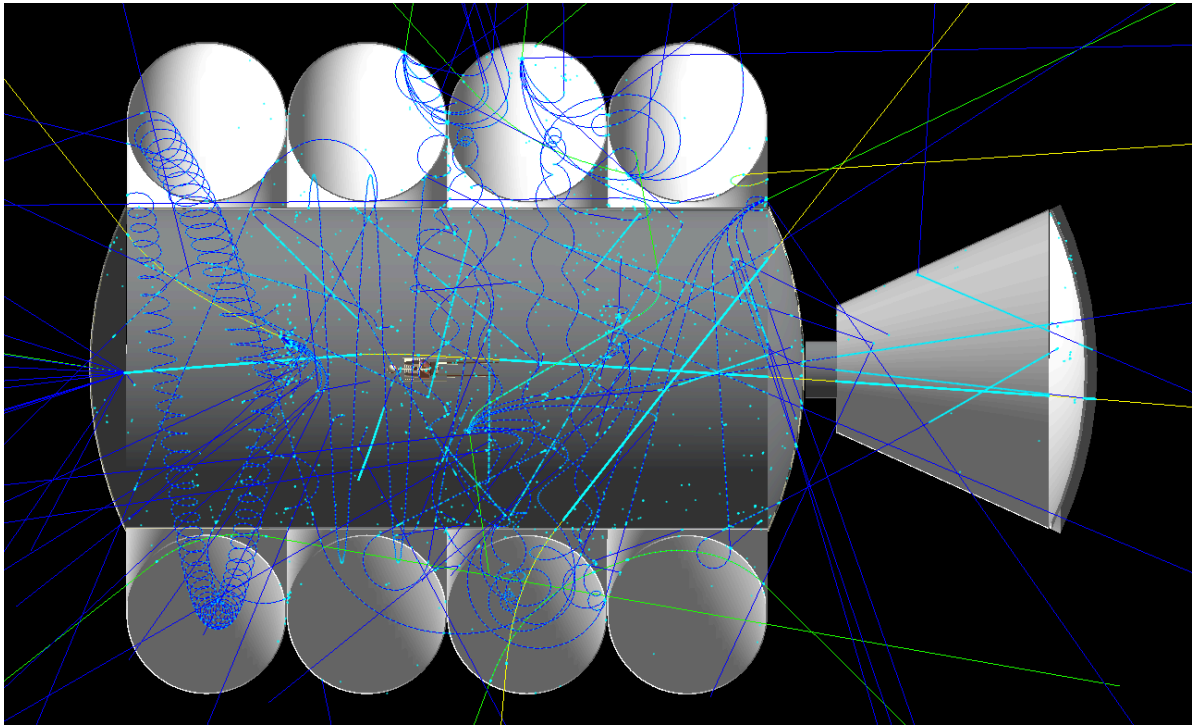
**Figure 59: 0 T Toroidal Magnetic Shielding in GCR Environment**



**Figure 60: 1.5 T Toroidal Magnetic Shielding in GCR Environment**



**Figure 61: 3 T Toroidal Magnetic Shielding in GCR Environment**



**Figure 62: 7 T Toroidal Magnetic Shielding in GCR Environment**

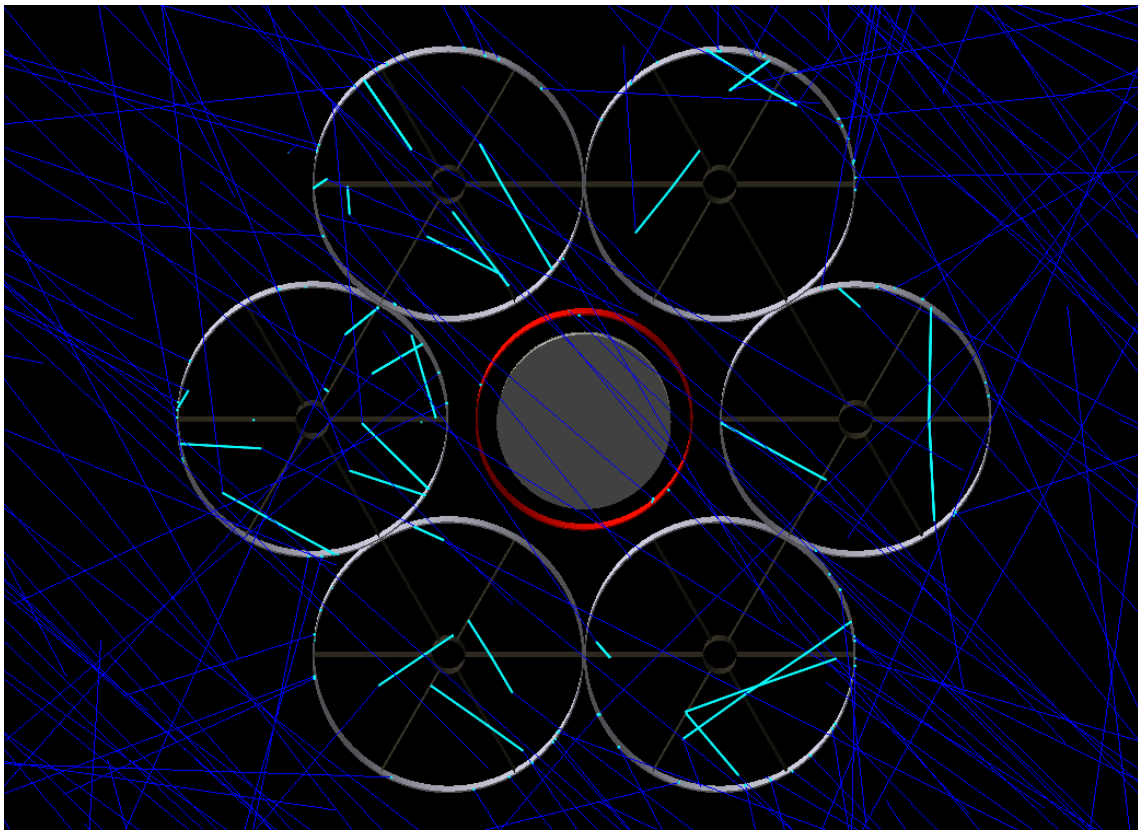
### 3.2.3 Solenoidal Magnetic Shielding

Our second experimental magnetic shielding case is a solenoidal shielding configuration. In this configuration, radiation shielding is provided to the astronauts inside the spacecraft from six 4 m cross-sectional radius superconducting solenoid magnets and one 3.2 m cross-sectional radius correction superconducting solenoid magnet. Since solenoidal magnetic fields are unconfined, this configuration also includes a small fringe field acting at a distance. Details on the geometry of this shielding configuration and how we modeled it in GEANT4 can be found under Shielding Configurations in Section 2.3.3.4.

Results from the solenoidal shielding cases are provided in Figure 88, Figure 89, Figure 90, and Figure 91. From these figures, it is apparent that the effectiveness of the solenoidal magnetic shielding configuration is highly dependent on the magnetic field strength. The 7 T solenoidal magnetic shielding case meets the strictest effective dose limit (30-year-old female astronaut).

For older astronauts, the total mission effective dose is within permissible limits for the 3 T solenoidal shielding case. The error bars in these figures represent  $\pm 2\sigma$  ( $\sigma$ : standard error), or a 95% CI, as required by NASA (Cucinotta et al. 2012b).

Screenshots from the simulations of the solenoidal shielding case under both SPE and GCR incident radiation are provided in Figure 63, Figure 64, Figure 65, Figure 66, and Figure 67. Here we can see that SPE protons (blue) are almost entirely stopped by the passive shielding effect of the magnetic shielding structure, but GCR particles (blue, green, yellow, magenta, red) are still able pass through and can deliver dose to the astronaut depending on the magnetic field. As the magnetic field increases, the radius of curvature of the particle deflection decreases and particles of increasingly higher rigidity are turned away from the habitable volume.



**Figure 63: 0 T Solenoidal Magnetic Shielding in SPE Environment**

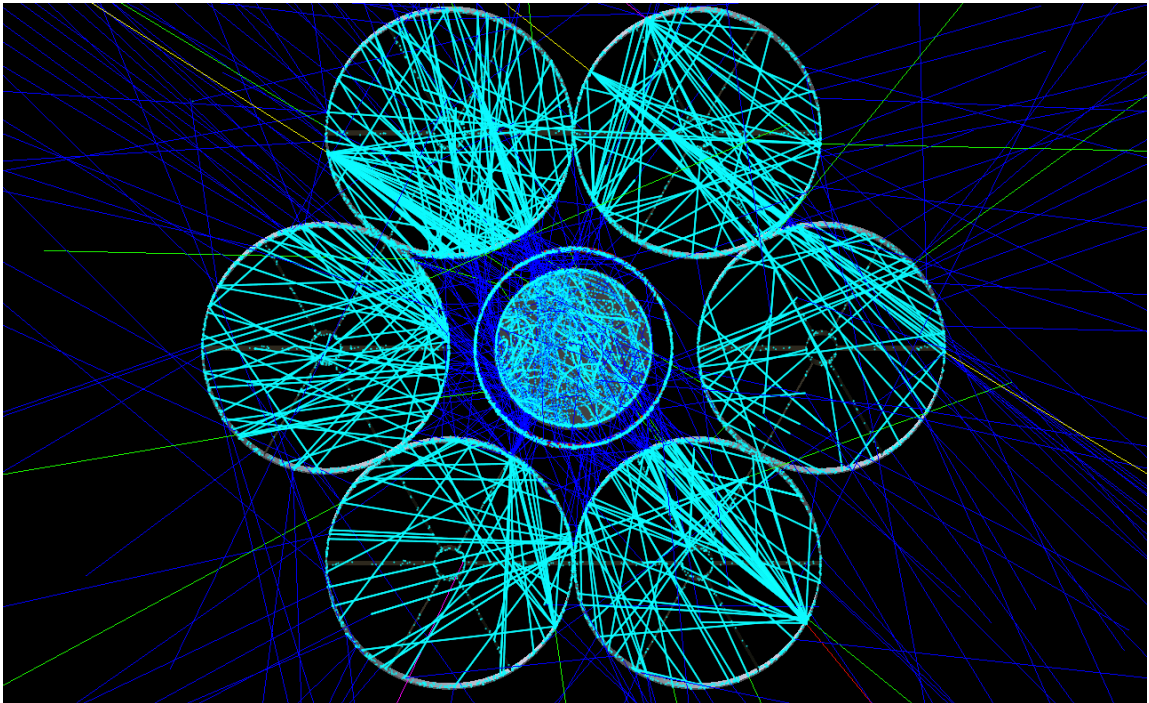


Figure 64: 0 T Solenoidal Magnetic Shielding in GCR Environment

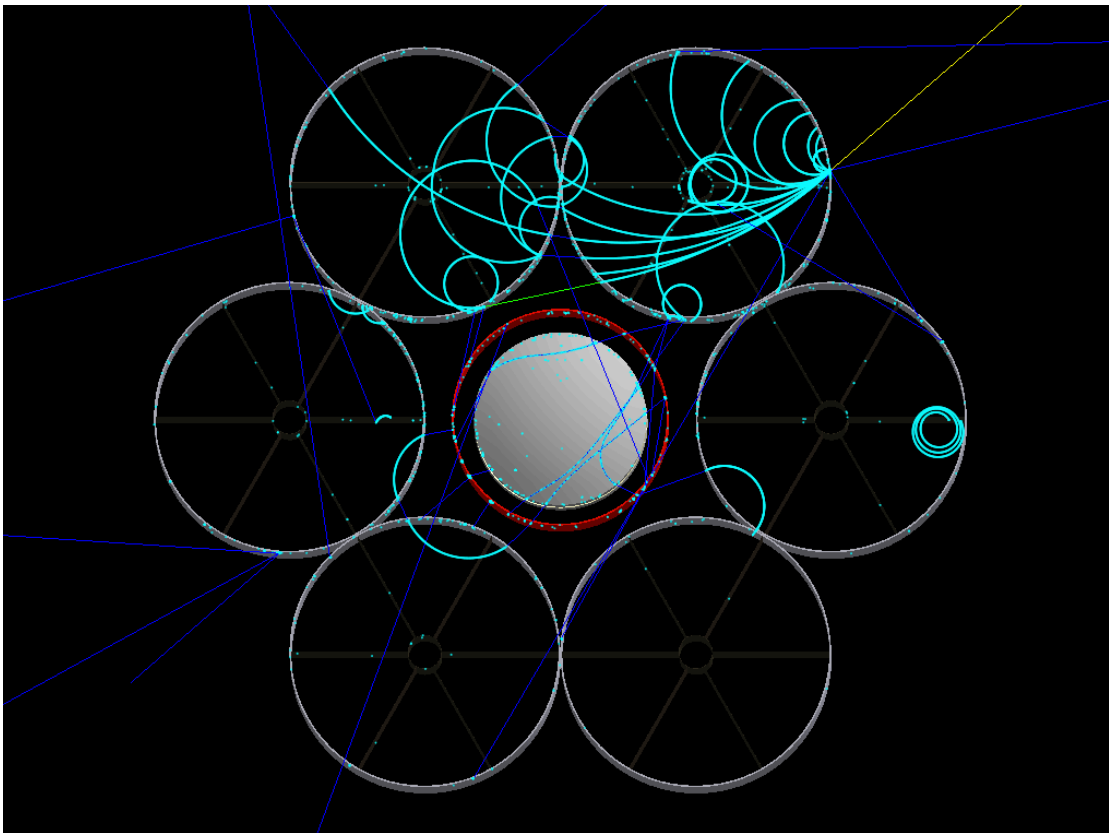


Figure 65: 1.5 T Solenoidal Magnetic Shielding in GCR Environment

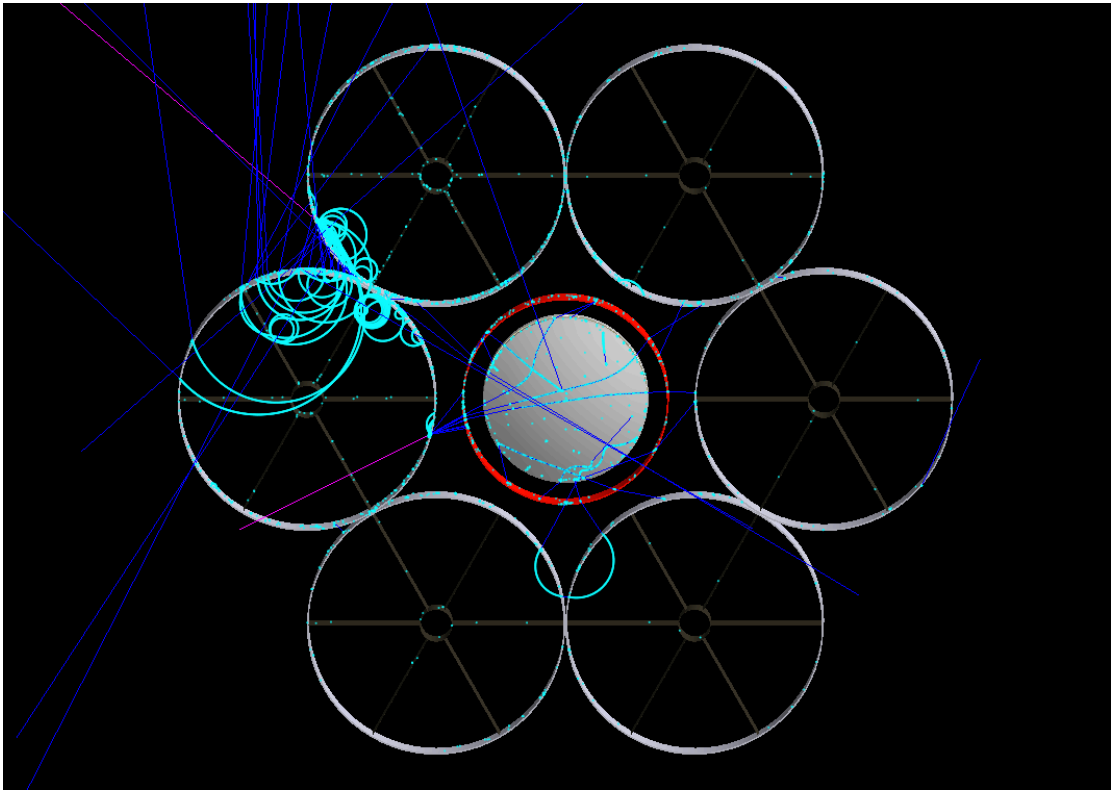


Figure 66: 3 T Solenoidal Magnetic Shielding in GCR Environment

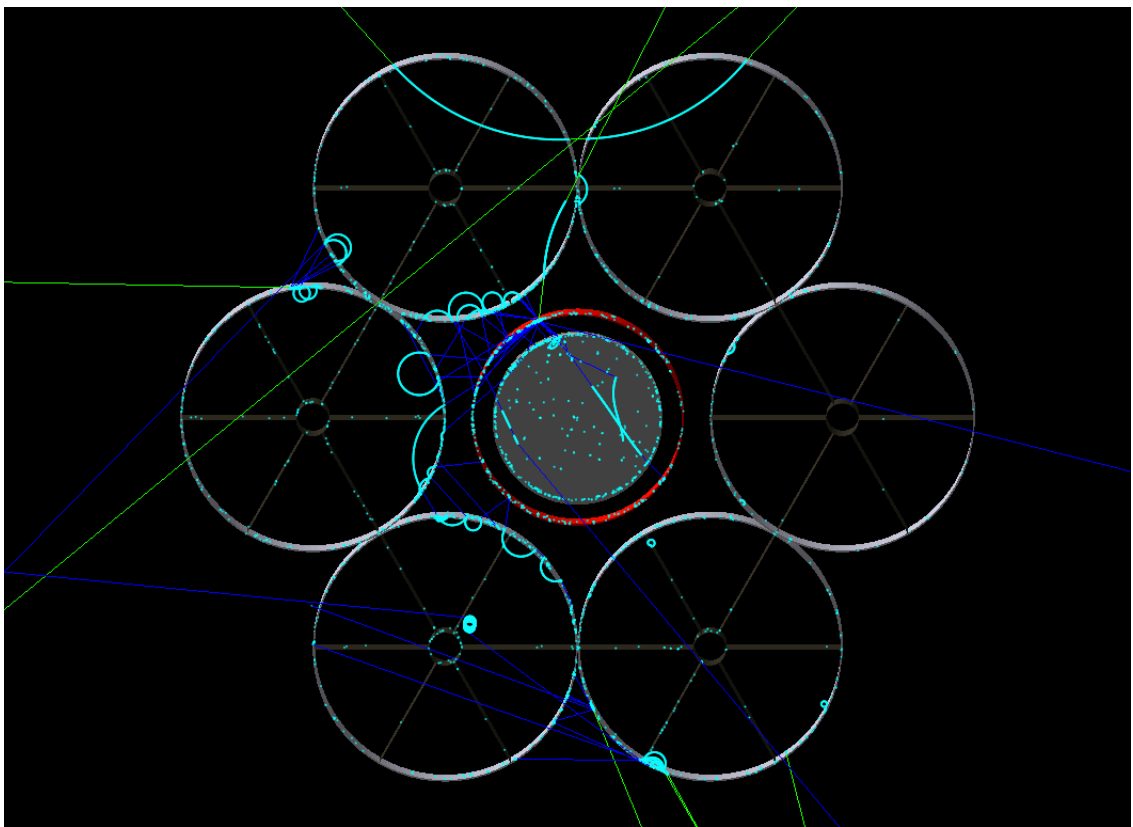


Figure 67: 7 T Solenoidal Magnetic Shielding in GCR Environment

### 3.2.4 Race Track Magnetic Shielding

Our third experimental magnetic shielding case is a race track shielding configuration. In this configuration, radiation shielding is provided to the astronauts inside the spacecraft from 120 20 m length, 3.6 m width superconducting race track loops and a titanium forming cylinder. Since the race track loops are based on toroidal magnets, the magnetic fields are confined and this configuration includes negligible fringe field acting at a distance. Details on the geometry of this shielding configuration and how we modeled it in GEANT4 can be found under Shielding Configurations in Section 2.3.3.5.

Results from the race track shielding cases are provided in Figure 88, Figure 89, Figure 90, and Figure 91. From these figures, it is apparent that the effectiveness of the race track magnetic shielding configuration is highly dependent on the magnetic field. None of the race track magnetic shielding cases meet the strictest effective dose limit (30-year-old female astronaut). For older and/or male astronauts, the total mission effective dose is within permissible limits for the 7 T toroidal shielding case. The error bars in these figures represent  $\pm 2\sigma$ , or a 95% CI, as required by NASA (Cucinotta et al. 2012b).

Screenshots from the simulations of the race track shielding case under both SPE and GCR incident radiation are provided in Figure 68, Figure 69, Figure 70, Figure 71, and Figure 72. Here we can see that SPE protons (blue) are almost entirely stopped by the passive shielding effect of the magnetic shielding structure, but GCR particles (blue, green, yellow, magenta, red) are still able pass through and can deliver dose to the astronaut depending on the magnetic field. As the magnetic field increases, the radius of curvature of the particle deflection decreases and particles of increasingly higher rigidity are turned away from the habitable volume.

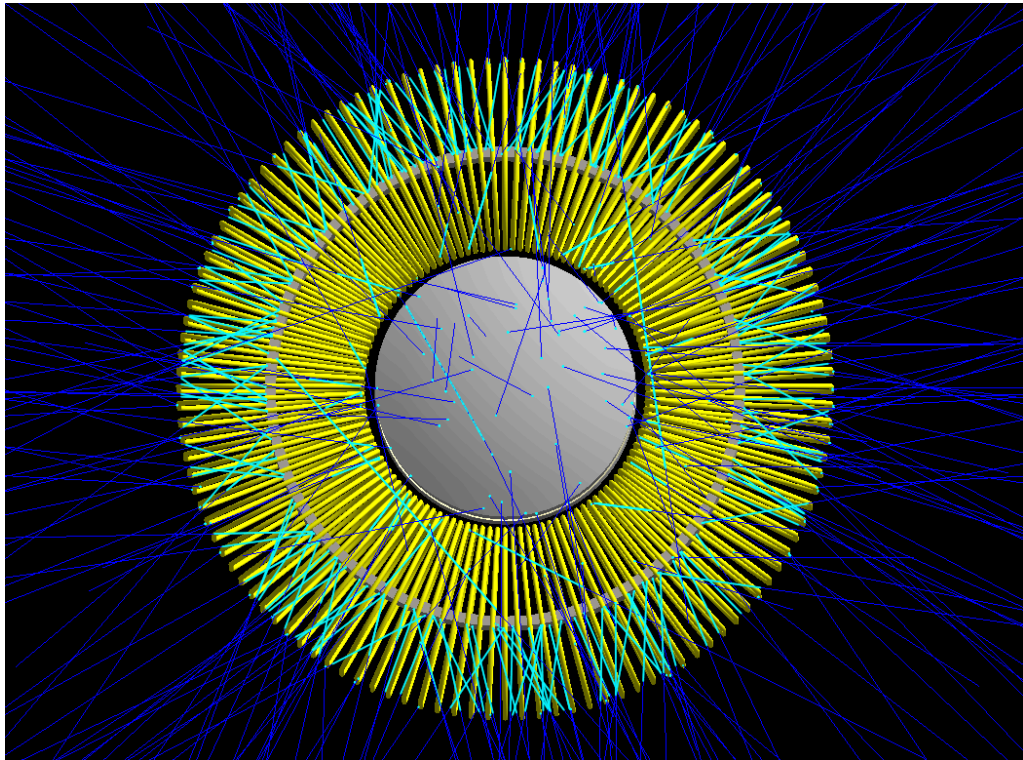


Figure 68: 0 T Race Track Magnetic Shielding in SPE Environment

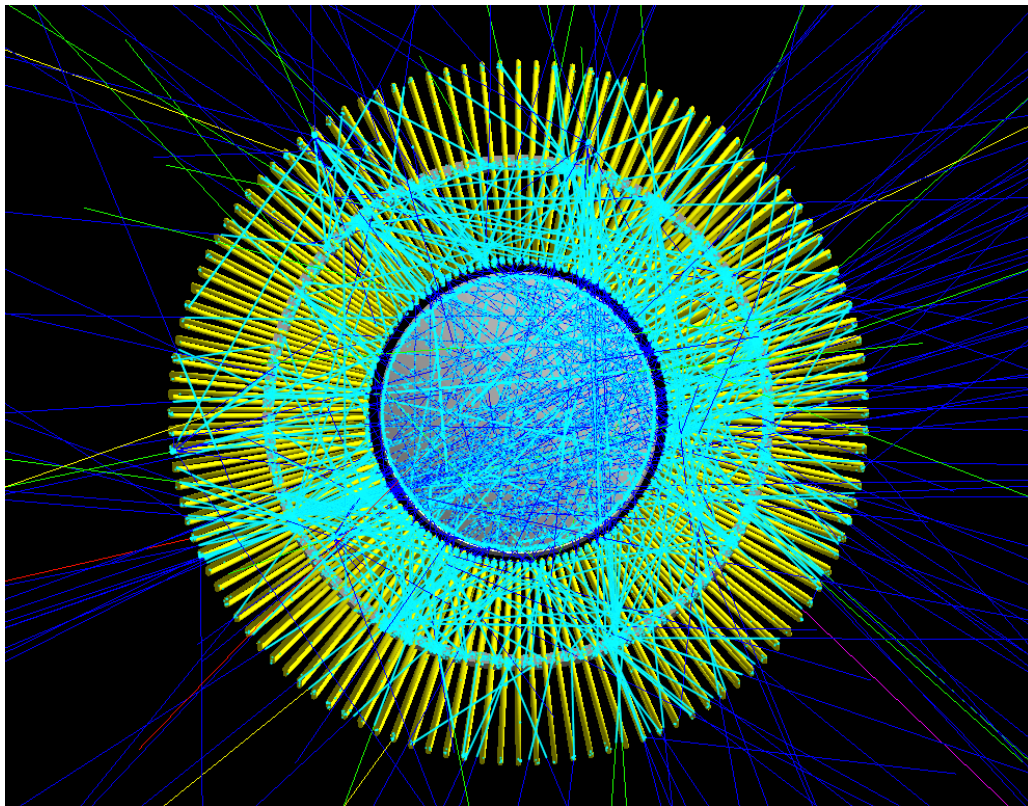


Figure 69: 0 T Race Track Magnetic Shielding in GCR Environment

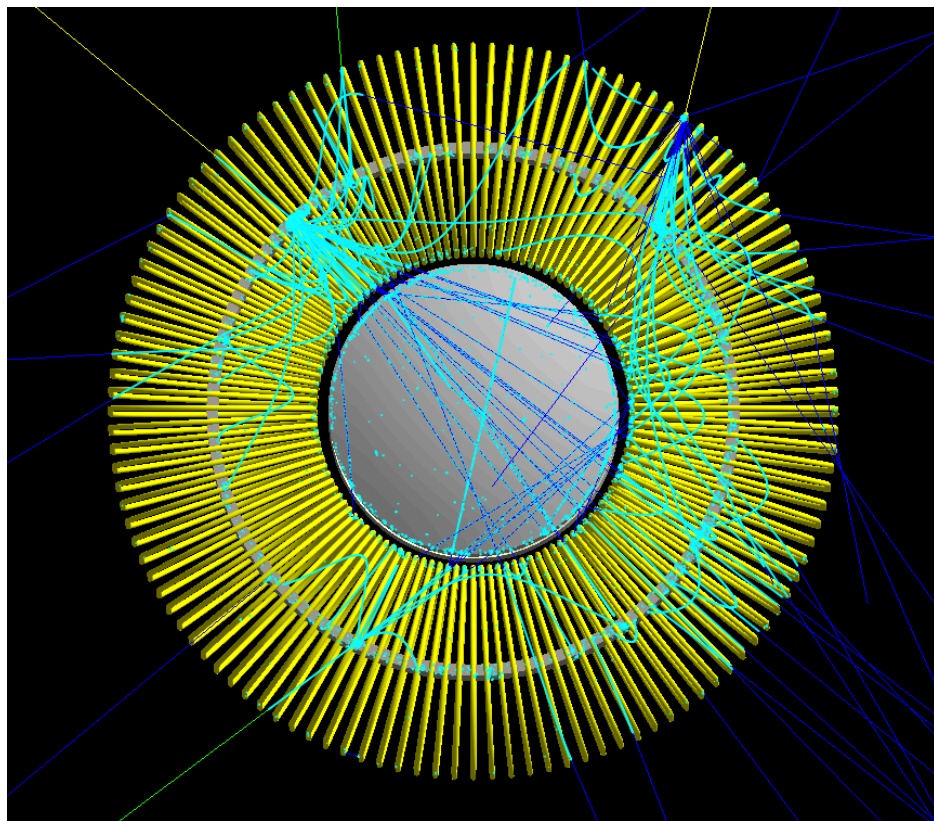


Figure 70: 1.5 T Race Track Magnetic Shielding in GCR Environment

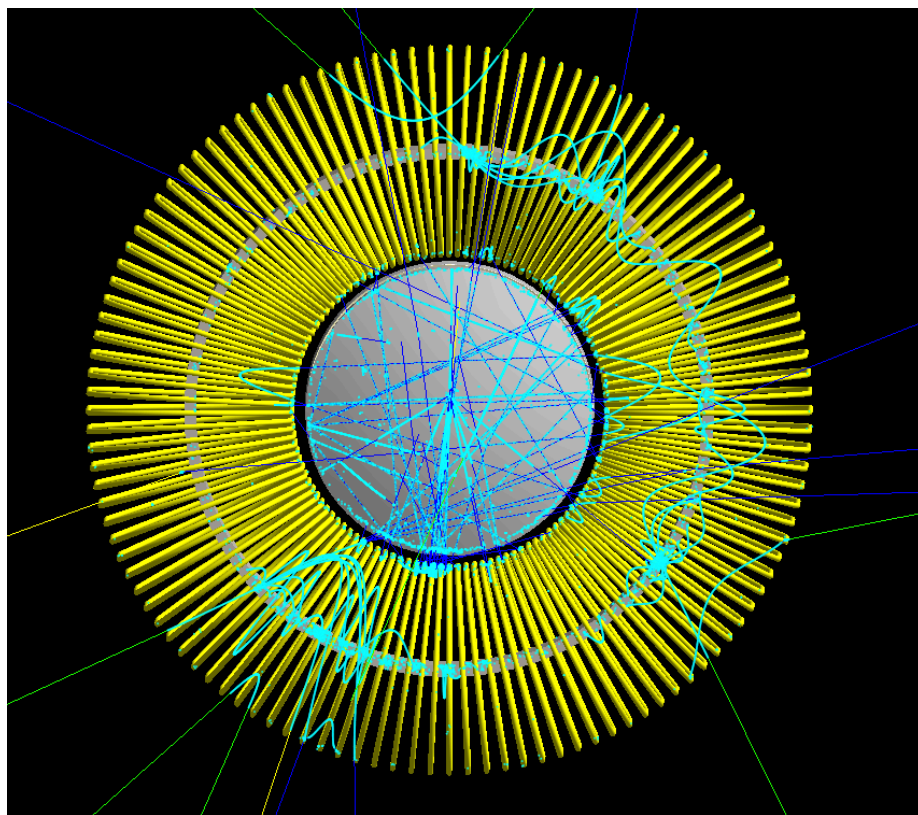


Figure 71: 3 T Race Track Magnetic Shielding in GCR Environment

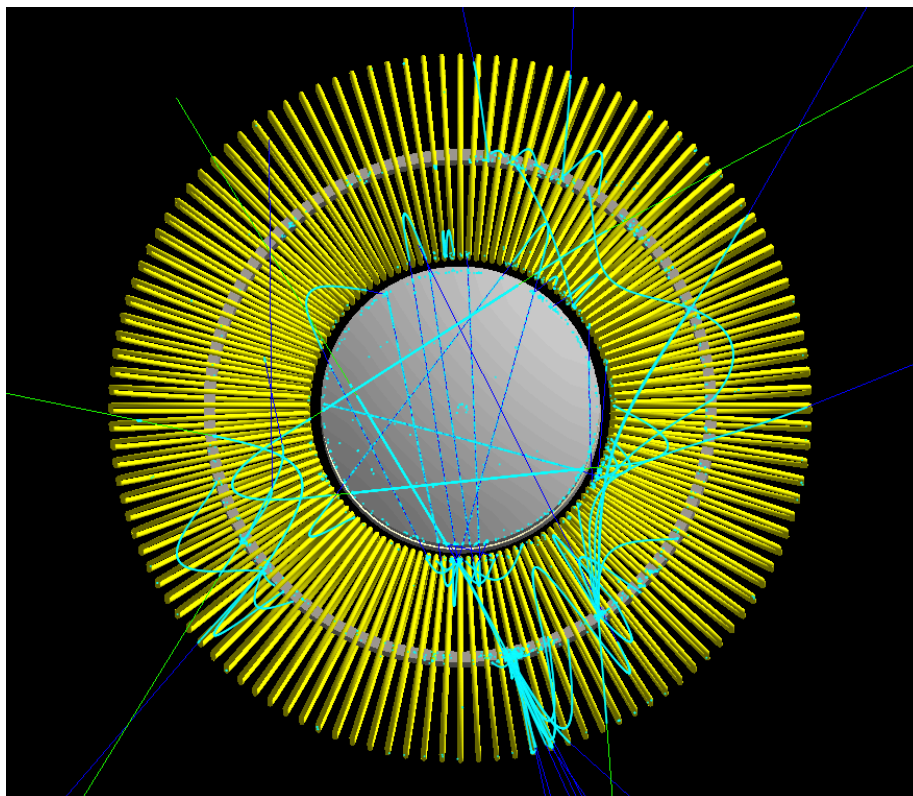


Figure 72: 7 T Race Track Magnetic Shielding in GCR Environment

### 3.3 Shielding Effectiveness by Scenario and Dosimetry Approach

#### 3.3.1 Effective Dose vs. Magnetic Field by Scenario

Figure 73 displays effective dose rate vs. magnetic field strength for the **female astronaut, solar max** scenario. With only four data points in each series, any fit curve would be questionable; more data points at higher hypothetical magnetic fields would be required to more accurately identify the characteristics of each relationship. However, simulation data using higher magnetic fields would not be valid because the superconducting magnet configurations selected in this study would not feasibly support magnetic forces at fields greater than 7-8 T (Westover 2014). In the absence of additional data points at these higher fields, our team decided to apply an exponential fit because it provided the strongest  $R^2$  as well as a physical argument based on exponential attenuation of radiation. Exponential fit curves were added to the data points for this scenario with equations and  $R^2$  values displayed in Figure 73;  $R^2$  for all fit curves in this figure were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

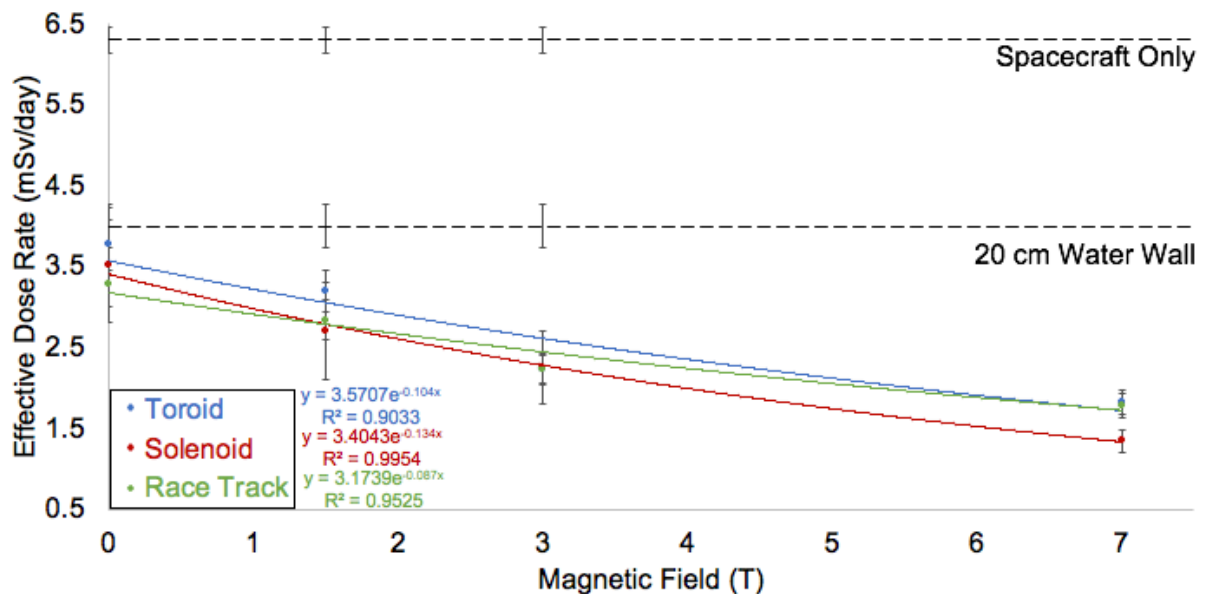


Figure 73: Effective Dose Rate vs. Magnetic Field (Female, Solar Max)

Figure 74 displays effective dose rate vs. magnetic field strength for the **female astronaut, solar min** scenario. Similar to the prior case, we selected exponential fit curves for the data points in this scenario with equations and  $R^2$  values displayed in Figure 74;  $R^2$  for all fit curves in this figure were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

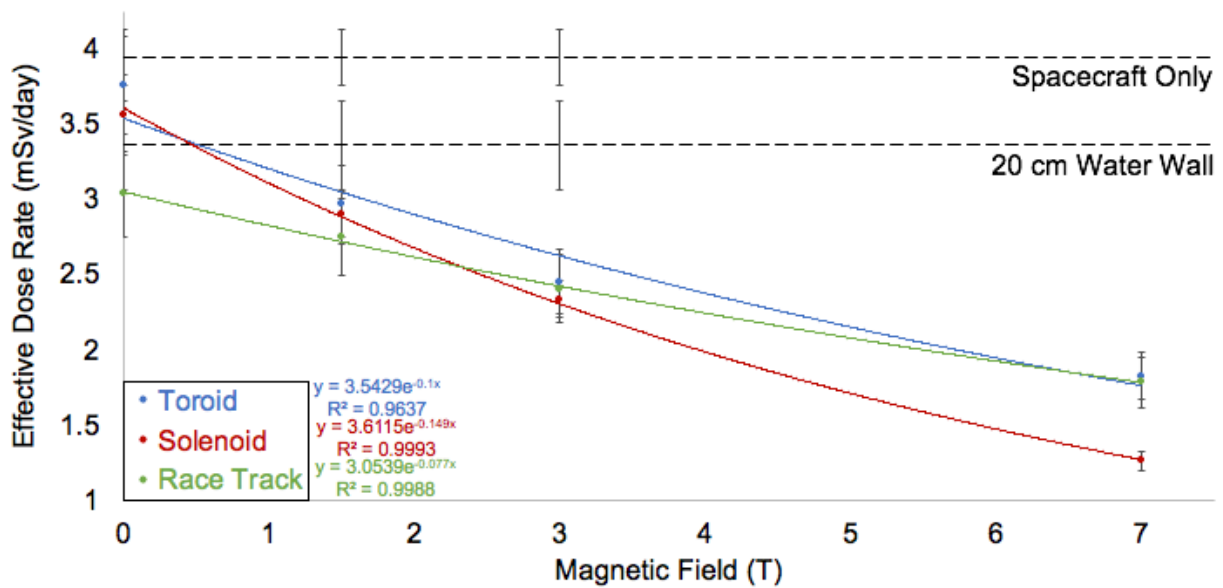


Figure 74: Effective Dose Rate vs. Magnetic Field (Female, Solar Min)

Figure 75 displays effective dose rate vs. magnetic field strength for the **male astronaut, solar max** scenarios. Similar to the female astronaut cases, we selected exponential fit curves for the data points in this scenario with equations and  $R^2$  values displayed in Figure 75;  $R^2$  for all fit curves in this figure were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

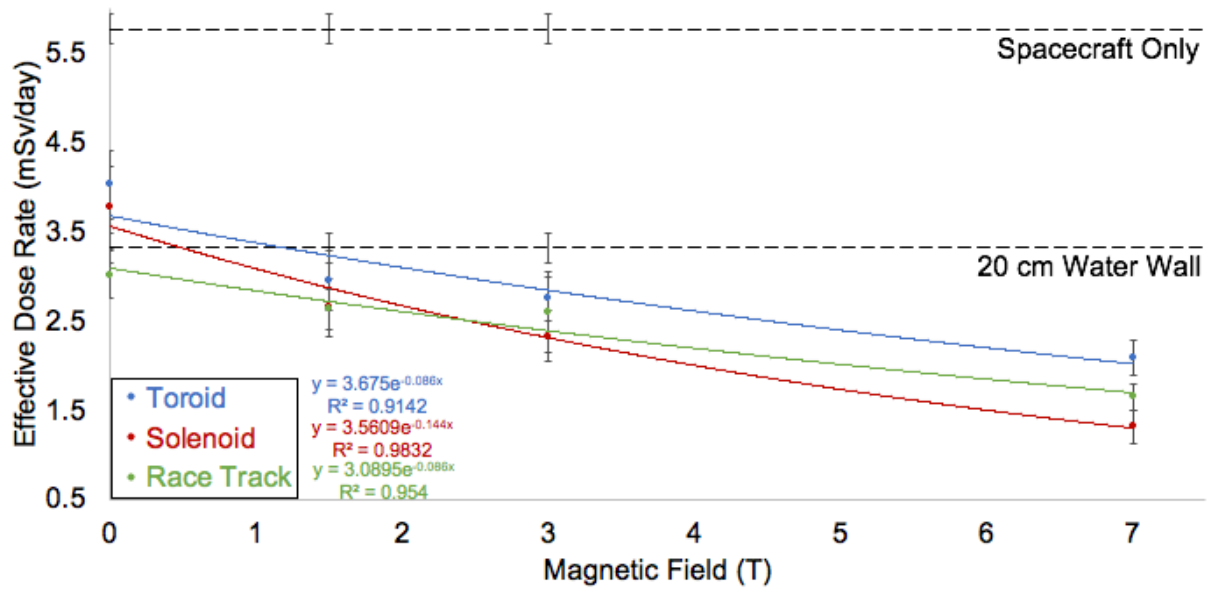


Figure 75: Effective Dose Rate vs. Magnetic Field (Male, Solar Max)

Figure 76 displays effective dose rate vs. magnetic field strength for the **male astronaut, solar min** scenarios. Similar to the prior cases, we selected exponential fit curves for the data points in this scenario with equations and  $R^2$  values displayed in Figure 76;  $R^2$  for all fit curves in this figure were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

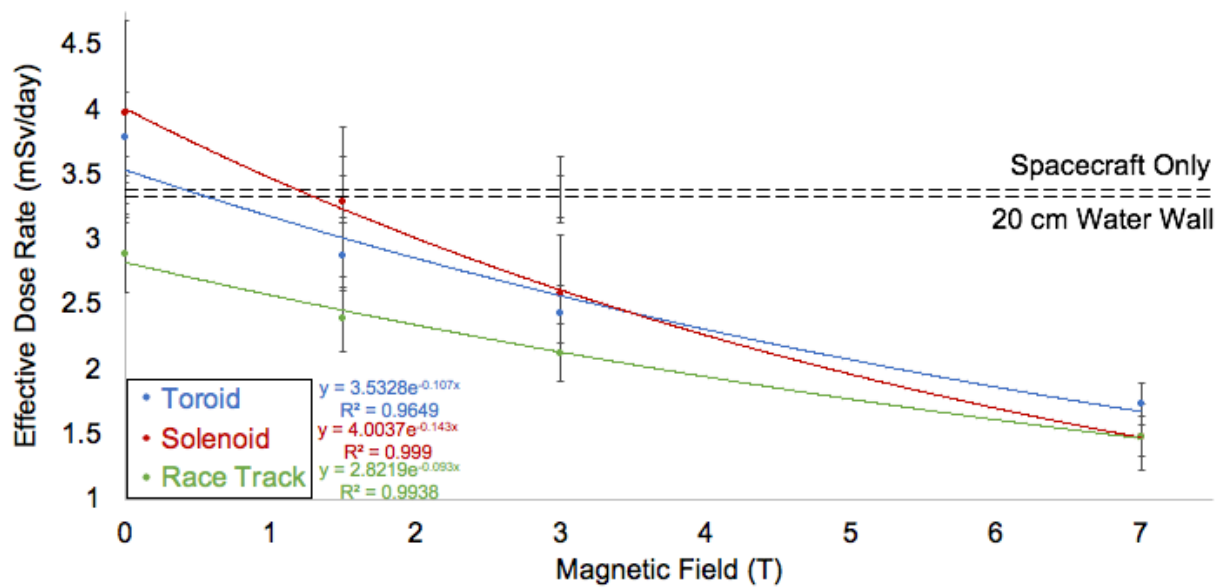


Figure 76: Effective Dose Rate vs. Magnetic Field (Male, Solar Min)

From these data (summarized in Table 12), we observed that the toroid and solenoid configurations provided a similar passive shielding effect, as evidenced by similar intercepts in the exponential fit curves ( $3.61 \pm 0.03$  Sv), while the race track configuration provided slightly better passive shielding effect (3.03 Sv). The toroid and race track configurations provided a similar active shielding effect, as evidenced by similar exponents in the exponential fit curves ( $-0.09 \pm 0.01$ ), while the solenoid configuration provided slightly better active shielding effect ( $-0.14$ ). Also from these data in Table 12, we observed that the standard deviation of the exponential fit curve parameters was  $\leq 10\%$  across the astronaut sex and solar cycle parameters.

**Table 12: Effective Dose vs. Magnetic Field Exponential Fit Intercept and Exponent Summary**

	Toroid		Solenoid		Race Track	
	Intercept	Exponent	Intercept	Exponent	Intercept	Exponent
<b>Female, Solar Max</b>	3.57	-0.104	3.40	-0.134	3.17	-0.087
<b>Female, Solar Min</b>	3.54	-0.100	3.61	-0.149	3.05	-0.077
<b>Male, Solar Max</b>	3.68	-0.086	3.56	-0.144	3.09	-0.086
<b>Male, Solar Min</b>	3.53	-0.107	4.00	-0.143	2.82	-0.093
<b>Average</b>	3.580	-0.099	3.643	-0.143	3.033	-0.086
<b>St. Dev.</b>	0.07	0.01	0.25	0.01	0.15	0.01
<b>% St. Dev.</b>	1.9%	9.4%	7.0%	4.4%	5.0%	7.7%

The selection of exponential fit allowed us to analyze the projected magnetic field strength that reduced the total effective dose rate by 50% using exponential decay parameters. The exponential fit curves followed the form:

$$E = E_0 e^{-kB}$$

where  $E$  is the total effective dose rate (mSv/day),  $E_0$  is the intercept or effective dose rate at 0 T (mSv/day),  $k$  is the exponent from the fit curve, and  $B$  is the magnetic field strength (T). The

projected magnetic field strength that reduced the total effective dose rate by 50% is then given by:

$$B_{1/2} = \frac{\ln 2}{k} = \frac{0.693}{k}$$

where  $B_{1/2}$  is the projected magnetic field strength and  $k$  is the exponent from the exponential fit curve. A summary of these projected magnetic field strengths is given in Table 13 below. With these data, we isolated the active shielding effect from the passive shielding effect of each magnetic shielding configuration and estimated the magnetic field required to reduce the effective dose rate by 50% vs. the 0 T cases. We observed that the solenoid configuration had a stronger active shielding effect, requiring an average of 4.9 T to reduce the effective dose rate by 50% vs. the 0 T case, where the toroid and race track configurations required an average of 7-8 T to produce the same effectiveness. Note: The field strengths in Table 13 are higher than what was required to reduce the effective dose rate by 50% vs. the minimal shielding case (our hypothesis), but this information served to characterize specifically the active shielding effect.

**Table 13: Projected Magnetic Field Strengths to Reduce Effective Dose Rate by 50% vs. 0 T**

	<b>Toroid (T)</b>	<b>Solenoid (T)</b>	<b>Race Track (T)</b>
<b>Female, Solar Max</b>	6.66	5.17	7.97
<b>Female, Solar Min</b>	6.93	4.65	9.00
<b>Male, Solar Max</b>	8.06	4.81	8.06
<b>Male, Solar Min</b>	6.48	4.85	7.45
<b>Average</b>	7.03	4.87	8.12
<b>St. Dev.</b>	0.71	0.22	0.65
<b>% St. Dev.</b>	10.1%	4.5%	7.9%

Further study is warranted to investigate potential differences in effectiveness across the three magnetic shielding configurations. However, the average difference in total effective dose

rate by magnetic shielding configuration was  $\leq 10\%$  across all the scenarios we tested (see Table 14). For the purposes of this study, we concluded that the selection of magnetic shielding configuration produced minimal difference in effectiveness at the magnetic fields tested. We confirmed this was a reasonable conclusion within the objectives of this study via practical application with our REID vs. age analysis in Section 3.11, Figure 126 through Figure 129 and Figure 134 through Figure 137.

**Table 14: Average Difference in Effective Dose Rate by Magnetic Shielding Configuration**

	Average Difference			
	Female, Max	Female, Min	Male, Max	Male, Min
<b>Toroid/Solenoid</b>	5%	4%	7%	6%
<b>Toroid/Race Track</b>	6%	7%	8%	7%
<b>Solenoid/Race Track</b>	6%	6%	7%	10%

The minimum age a **female astronaut** could safely embark on a 700-day mission at **solar maximum** is summarized in Table 15 below. From this table, the minimum safe age is within two years for all three magnetic shielding configurations at all magnetic fields, with exception of the solenoid configuration at 7 T. This is primarily due to the flatness of the REID vs. astronaut age curves throughout the typical range of ages such that a small difference in effective dose can produce a large shift in the minimum safe age for a specific mission scenario. Further study may be warranted to further investigate possible differences in magnetic shielding configurations at intermediate magnetic field strengths, but this study found minimal difference in the shielding effectiveness of the three magnetic shielding configurations at the field strengths tested.

**Table 15: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration (Female, Solar Max)**

	<b>Minimum Safe Age for 700-day Mission (Female)</b>			
<b>Magnetic Field</b>	<b>Toroid</b>	<b>Solenoid</b>	<b>Race Track</b>	<b>Average</b>
<b>0 T</b>	67	67	67	67
<b>1.5 T</b>	66	65	65	66
<b>3 T</b>	62	62	62	62
<b>7 T</b>	49	36	48	43

The minimum age a **female astronaut** could safely embark on a 700-day mission at **solar minimum** is summarized in Table 16 below. From this table, the minimum safe age is within two years for all three magnetic shielding configurations at all magnetic fields, with exception of the solenoid configuration at 7 T. Similar to the prior case, this is primarily due to the flatness of the REID vs. astronaut age curves throughout the typical range of ages such that a small difference in effective dose can produce a large shift in the minimum safe age for a specific mission scenario. Further study may be warranted to further investigate possible differences in magnetic shielding configurations at intermediate magnetic field strengths, but this study found minimal difference in the shielding effectiveness of the three magnetic shielding configurations at the field strengths tested.

**Table 16: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration (Female, Solar Min)**

	<b>Minimum Safe Age for 700-day Mission (Female)</b>			
<b>Magnetic Field</b>	<b>Toroid</b>	<b>Solenoid</b>	<b>Race Track</b>	<b>Average</b>
<b>0 T</b>	67	67	67	67
<b>1.5 T</b>	66	65	65	66
<b>3 T</b>	62	62	62	62
<b>7 T</b>	49	36	48	43

The minimum age a **male astronaut** could safely embark on a 700-day mission at **solar maximum** is summarized in Table 17 below. From this table, the minimum safe age is within five years for all three magnetic shielding configurations at all magnetic fields. The end result for all configurations is that the minimum safe crew age is greater than a practical astronaut age for all magnetic fields except 7 T. For only the 7 T scenarios, the minimum safe age is within the typical range of astronaut ages (35-55) and therefore supports the feasibility of a 700-day Mars flyby mission. Similar to the prior cases, this is primarily due to the flatness of the REID vs. astronaut age curves throughout the typical range of ages such that a small difference in effective dose can produce a large shift in the minimum safe age for a specific mission scenario. Further study may be warranted to further investigate possible differences in magnetic shielding configurations at intermediate magnetic field strengths, but this study found minimal difference in the shielding effectiveness of the three magnetic shielding configurations at the field strengths tested.

**Table 17: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration (Male, Solar Max)**

<b>Magnetic Field</b>	<b>Minimum Safe Age for 700-day Mission (Male)</b>			
	<b>Toroid</b>	<b>Solenoid</b>	<b>Race Track</b>	<b>Average</b>
<b>0 T</b>	65	65	62	64
<b>1.5 T</b>	62	63	58	61
<b>3 T</b>	59	59	56	58
<b>7 T</b>	36	31	31	33

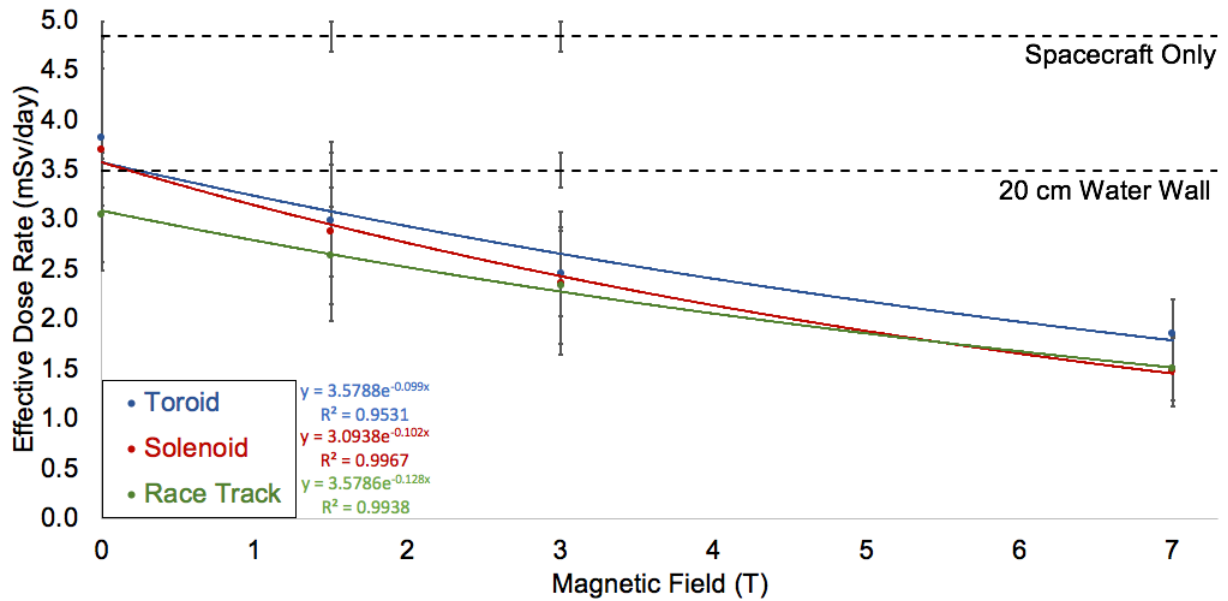
The minimum age a **male astronaut** could safely embark on a 700-day mission at **solar minimum** is summarized in Table 18 below. From this table, the minimum safe age is within five years for all three magnetic shielding configurations at all magnetic fields. The end result for all configurations is that the minimum safe crew age is greater than a practical astronaut age for all magnetic fields except 7 T. For only the 7 T scenarios, the minimum safe age is within the typical range of astronaut ages (35-55) and therefore supports the feasibility of a 700-day Mars flyby

mission. Similar to the prior cases, this is primarily due to the flatness of the REID vs. astronaut age curves throughout the typical range of ages such that a small difference in effective dose can produce a large shift in the minimum safe age for a specific mission scenario. Further study may be warranted to further investigate possible differences in magnetic shielding configurations at intermediate magnetic field strengths, but this study found minimal difference in the shielding effectiveness of the three magnetic shielding configurations at the field strengths tested.

**Table 18: Minimum Safe Age for 700-Day Mission by Magnetic Shielding Configuration (Male, Solar Min)**

<b>Magnetic Field</b>	<b>Minimum Safe Age for 700-day Mission (Male)</b>			
	<b>Toroid</b>	<b>Solenoid</b>	<b>Race Track</b>	<b>Average</b>
<b>0 T</b>	65	65	62	64
<b>1.5 T</b>	62	63	57	61
<b>3 T</b>	59	59	56	58
<b>7 T</b>	36	31	31	33

Figure 77 displays effective dose rate vs. magnetic field strength, averaging across astronaut sex and solar cycle parameters. We selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 77;  $R^2$  for all fit curves were  $>0.9$ . The error bars in this figure represent the standard error of the mean from the Monte Carlo simulations adjusted for the standard deviation in averaging the dose rates across astronaut sex and solar cycle parameters.



**Figure 77: Effective Dose Rate vs. Magnetic Field by Shielding Configuration  
(Average Astronaut Sex, Solar Cycle)**

In summary, the average difference between the effective dose rate vs. magnetic field by shielding configuration data points was 5-10% across all magnetic field strengths (Table 14), the intercepts and exponents of the exponential fit curves vary by less than 10% (Table 12), and the minimum safe age or a 700-day mission varies by <5 years in most cases (Table 15 through Table 18). Averaging across all astronaut sex, solar cycle, and magnetic shielding configuration parameters produced an exponential intercept of  $3.42 \pm 0.03$  (1% standard deviation) and exponent of  $-0.11 \pm 0.01$  (1% standard deviation).

The results of this section support the conclusion that the astronaut sex, solar cycle, and magnetic shielding configuration parameters produce minimal differences in shielding effectiveness within the objectives of this study and can be pooled for the remainder of the analysis.

### 3.3.2 Absorbed Dose vs. Effective Dose Comparison

Our team wished to further investigate the results in the previous section by analyzing absorbed dose vs. magnetic field strength. The purpose of this analysis was to determine if the translation from absorbed dose to effective dose masked trends in shielding effectiveness vs. magnetic field strength.

Figure 78 displays absorbed dose rate vs. magnetic field strength for the **female astronaut, solar max** scenario. Similar to the effective dose vs. magnetic field strength scenarios in the prior section, we selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 78;  $R^2$  for the toroid and solenoid fit curves were  $>0.9$ ,  $R^2$  for the race track fit curve was 0.89. Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

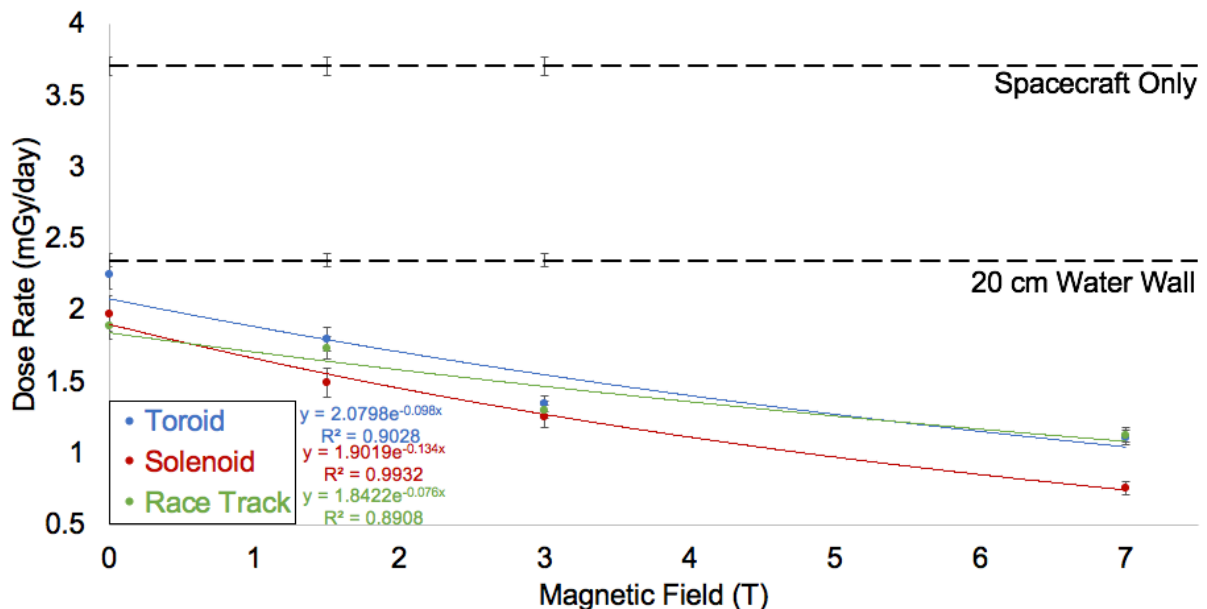


Figure 78: Absorbed Dose Rate vs. Magnetic Field (Female, Solar Max)

Figure 79 displays absorbed dose rate vs. magnetic field strength for the **female astronaut, solar min** scenario. Similar to the prior case, we selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 79;  $R^2$  for all fit curves were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

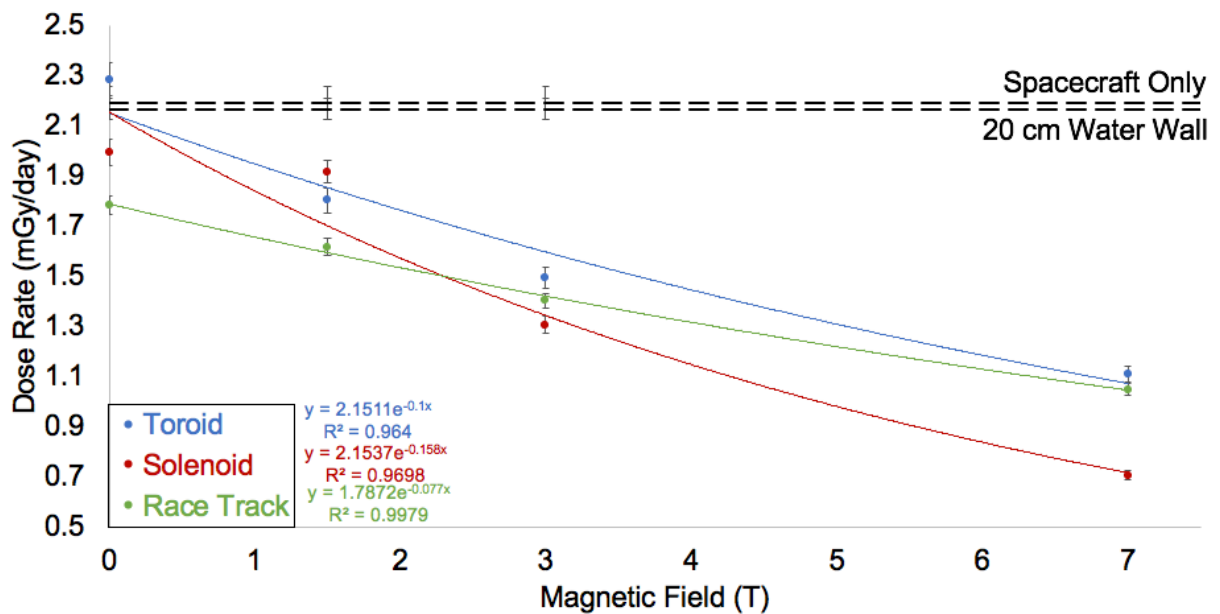


Figure 79: Absorbed Dose Rate vs. Magnetic Field (Female, Solar Min)

Figure 80 displays absorbed dose rate vs. magnetic field strength for the **male astronaut, solar max** scenario. Similar to the prior cases, we selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 80;  $R^2$  for the solenoid and race track fit curves were  $>0.9$ , and  $R^2$  for the toroid fit curve was 0.77. Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

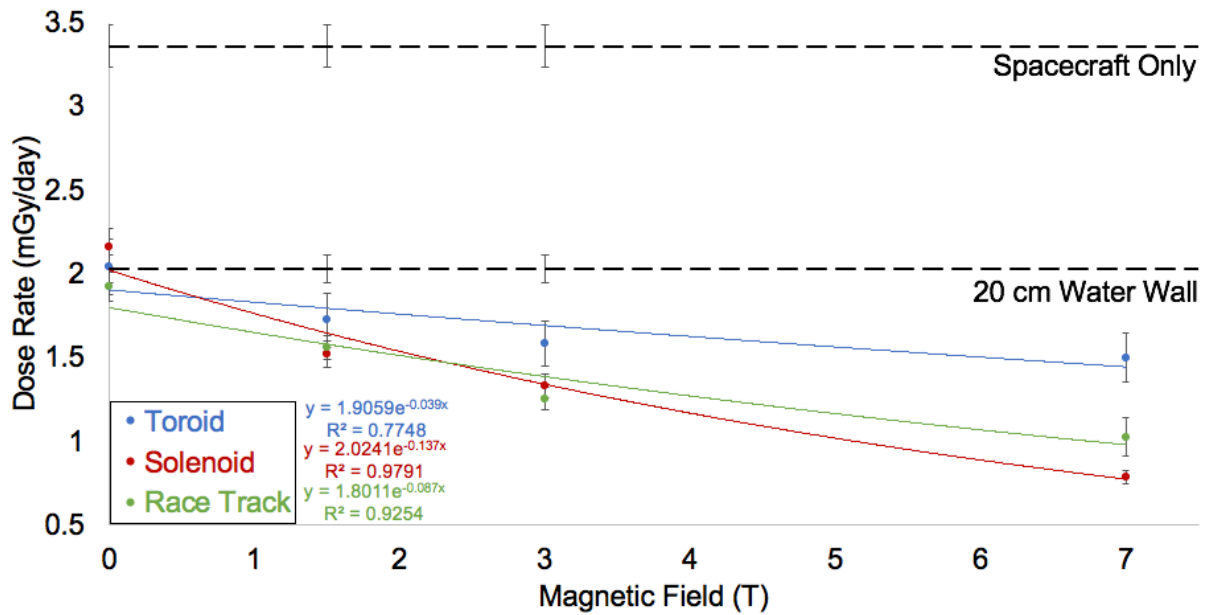


Figure 80: Absorbed Dose Rate vs. Magnetic Field (Male, Solar Max)

Figure 81 displays absorbed dose rate vs. magnetic field strength for the **male astronaut, solar min** scenario. Similar to the prior cases, we selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 81;  $R^2$  for all the fit curves were  $>0.9$ . Dashed lines were included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and 20 cm water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations.

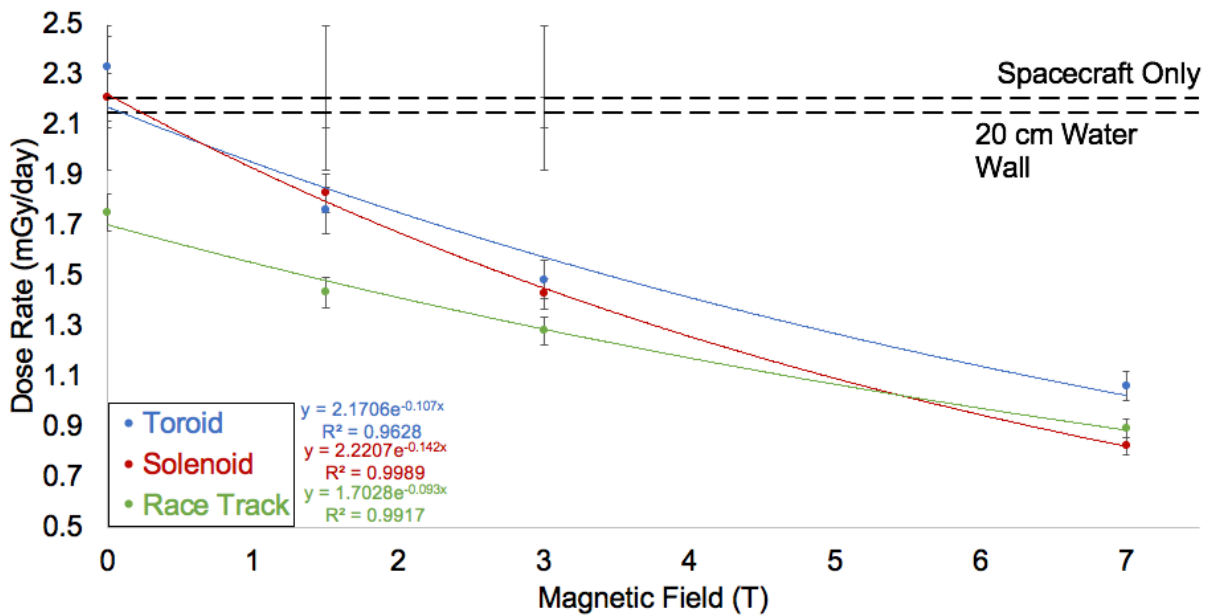


Figure 81: Absorbed Dose Rate vs. Magnetic Field (Male, Solar Min)

From these data (summarized in Table 19), we observed that as in the effective dose case, the toroid and solenoid configurations provided a similar passive shielding effect, as evidenced by similar intercepts in the exponential fit curves ( $2.075 \pm 0.003$  Gy), while the race track configuration provided slightly better passive shielding effect (1.78 Gy). Also as in the effective dose case, the toroid and race track configurations provided a similar active shielding effect, as evidenced by similar exponents in the exponential fit curves ( $-0.085 \pm 0.003$ ), while the solenoid configuration provided slightly better active shielding effect (-0.14).

**Table 19: Absorbed Dose vs. Magnetic Field Exponential Fit Intercept and Exponent Summary**

	Toroid		Solenoid		Race Track	
	Intercept	Exponent	Intercept	Exponent	Intercept	Exponent
<b>Female, Solar Max</b>	2.08	-0.098	1.90	-0.134	1.84	-0.076
<b>Female, Solar Min</b>	2.15	-0.100	2.15	-0.158	1.79	-0.077
<b>Male, Solar Max</b>	1.91	-0.039	2.02	-0.137	1.80	-0.087
<b>Male, Solar Min</b>	2.17	-0.107	2.22	-0.142	1.70	-0.093
<b>Average</b>	2.078	-0.086	2.073	-0.143	1.783	-0.083
<b>St. Dev.</b>	0.12	0.03	0.14	0.01	0.06	0.01
<b>% St. Dev.</b>	5.7%	36.7%	6.8%	7.5%	3.3%	9.8%

Also from these data in Table 19, we observed that the standard deviation of the exponential fit curve parameters was  $\leq 10\%$  across the astronaut sex and solar cycle parameters, with exception of the toroid exponent (36.7%). This deviation was due to one outlier scenario (toroid configuration, male astronaut, solar maximum); if the outlier is removed the standard deviation for the toroid exponent dropped to 0.5%.

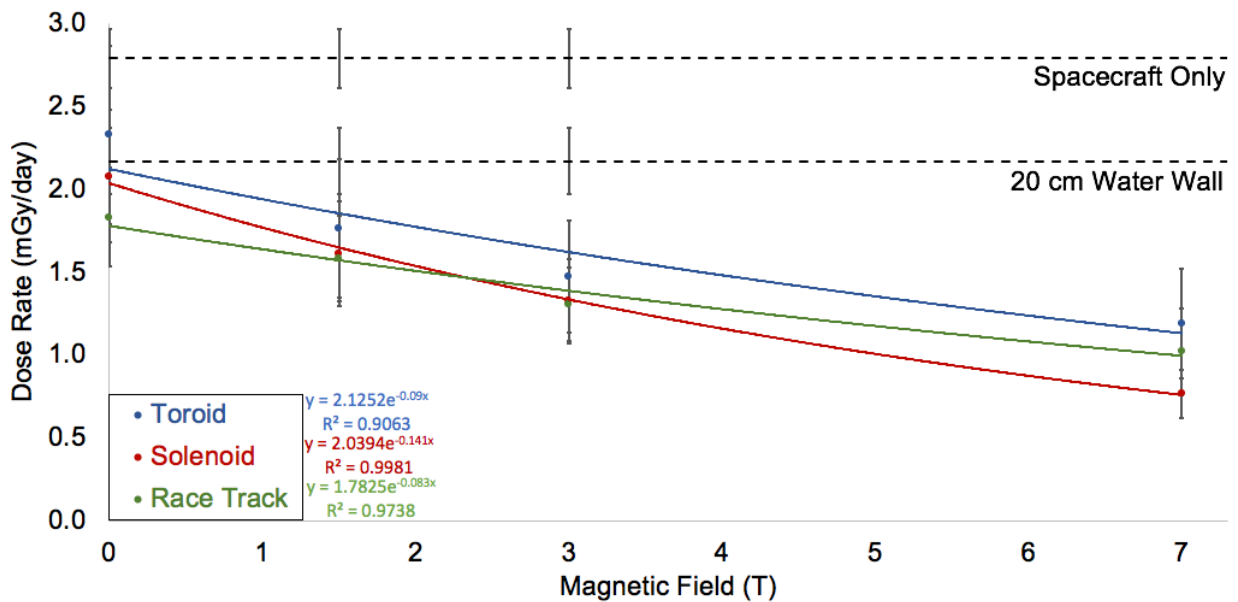
Similarly to the effective dose case in the previous section, we used these data to determine the magnetic field strength required to reduce the dose rate by 50% vs. the 0 T cases (see Table 20). From these data, we observed that the solenoid configuration had a stronger active shielding

effect, requiring an average of 4.9 T to reduce the effective dose rate by 50% vs. the 0 T case, where the toroid and race track configurations required an average of 8-9 T to produce the same effectiveness. The toroid configuration, male astronaut, solar max case again appears as an outlier (17.8 T). Note: The field strengths in Table 20 are higher than what was required to reduce the dose rate by 50% vs. the minimal shielding case (our hypothesis), but this information served to characterize specifically the active shielding effect.

**Table 20: Projected Magnetic Field Strengths to Reduce Dose Rate by 50% vs. 0 T**

	<b>Toroid (T)</b>	<b>Solenoid (T)</b>	<b>Race Track (T)</b>
<b>Female, Solar Max</b>	7.07	5.17	9.12
<b>Female, Solar Min</b>	6.93	4.39	9.00
<b>Male, Solar Max</b>	17.77	5.06	7.97
<b>Male, Solar Min</b>	6.48	4.88	7.45
<b>Average</b>	9.56	4.87	8.38
<b>St. Dev.</b>	5.48	0.35	0.81
<b>% St. Dev.</b>	57.3%	7.1%	9.6%

Figure 82 displays absorbed dose rate vs. magnetic field strength, averaged over astronaut sex and solar cycle parameters. Similar to the effective dose vs. magnetic field strength scenario in the prior section, we selected exponential fit curves for the data points for this scenario with equations and  $R^2$  values displayed in Figure 82;  $R^2$  for all fit curves were  $>0.9$ . The error bars in this figure represent the standard error of the mean from the Monte Carlo simulations adjusted for the standard deviation in averaging the dose rates across astronaut sex and solar cycle parameters.



**Figure 82: Dose Rate vs. Magnetic Field by Shielding Configuration (Average Astronaut Sex, Solar Cycle)**

In summary, the average difference between the absorbed dose rate vs. magnetic field by shielding type data points was 5-10% across all magnetic field strengths (see Table 21) and the intercepts and exponents of the exponential fit curves vary by less than 10%, with one exception (see Table 19). Averaging across all astronaut sex, solar cycle, and magnetic shielding configuration parameters produced an intercept of  $1.98 \pm 0.6$  (9.1% standard deviation) and exponent of  $-0.10 \pm 0.1$  (10% standard deviation).

These results support the conclusion from the prior section that the astronaut sex, solar cycle, and magnetic shielding configuration parameters produce minimal differences in shielding effectiveness within the objectives of this study and can be pooled for the remainder of the analysis.

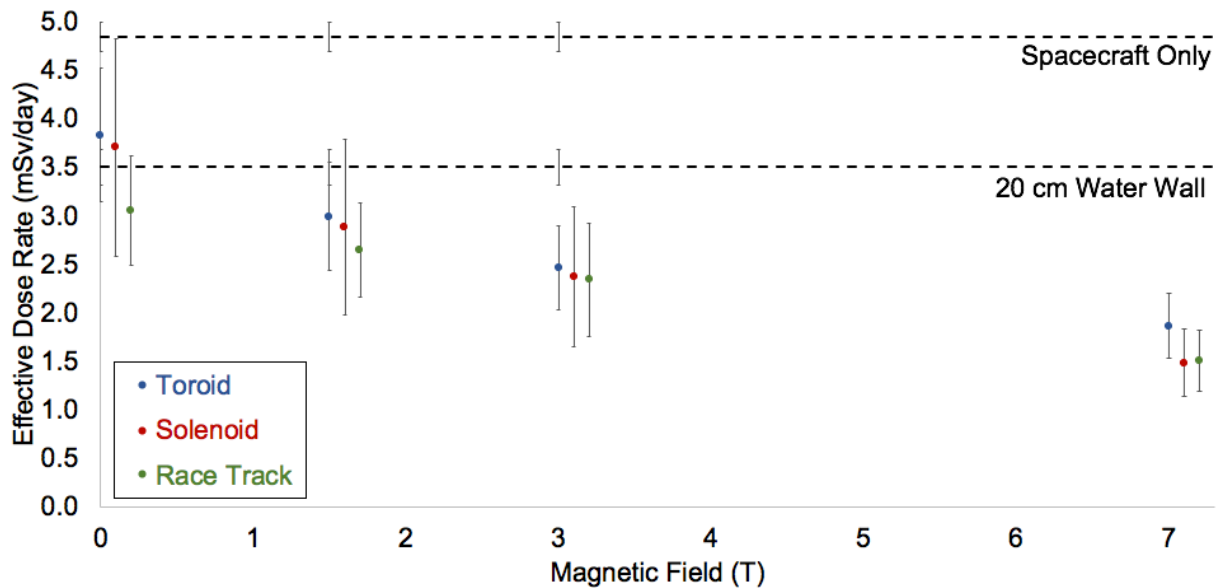
**Table 21: Average Difference in Absorbed Dose Rate by Magnetic Shielding Type**

	Average Difference in Absorbed Dose Rate			
	Female, Max	Female, Min	Male, Max	Male, Min
<b>Toroid/Solenoid</b>	8%	7%	8%	5%
<b>Toroid/Race Track</b>	5%	9%	9%	8%
<b>Solenoid/Race Track</b>	9%	8%	5%	8%

The results of this section indicate that the three magnetic shielding configurations analyzed in this study are similarly effective at reducing absorbed dose and in reducing effective dose. This confirms that the process of translating from absorbed dose (average energy deposited per unit mass across all organs) to effective dose (weighted sum of dose equivalent to specific organs using tissue weighting factors and radiation quality factors) in our calculations did not affect the overall trend in shielding effectiveness by shielding configuration. However, absorbed dose rate is not a practical measure of radiation risk in the space environment due to high radiation quality factors and relative biological effectiveness of HZE ions. Dose equivalent or effective dose are preferred quantities for space radiation doses, and we chose to use our effective dose data for the remainder of the analysis for this project.

### *3.4 Relative Shielding Effectiveness*

Figure 83 displays the relative effectiveness of each shielding configuration in reducing daily effective dose on an interplanetary mission based on magnetic field and averaged across astronaut sex and solar cycle parameters. Dashed lines are included to indicate the daily effective dose for the control cases, both minimal (aluminum spacecraft only) and passive water shielding. Error bars in this figure represent the standard error of the mean from the Monte Carlo simulations combined with the standard deviation between the specific male, female, solar max, solar min scenarios that were averaged to produce each data point.



**Figure 83: Effective Dose Rate vs. Magnetic Field by Shielding Configuration  
(Average Astronaut Sex, Solar Cycle)**

**Note:** Clusters of data points are artificially separated along the x-axis by a small amount to discern error bars.

In this figure, we observed that the differences in effective dose rate vs. magnetic field strength were within uncertainty when averaged over the astronaut sex, solar cycle, and magnetic shielding configuration parameters. Our sensitivity analysis results (see Section 3.10, Table 26) confirmed that variation in the total mission effective dose was 3% due to astronaut sex, 7% due to solar cycle, and 6% due to magnetic shielding type, with the remainder of the variation due to magnetic field. These results are consistent with the results of the prior two sections, that the specific magnetic shielding configuration contributes a small component of the variation in total shielding effectiveness when compared to the contribution of the magnetic field strength.

The minimal difference between the dose vs. magnetic field data in this study has several possible explanations. First, it is possible that there really is no true significant difference between these data. Second, it is possible that the uncertainty in our data is obscures a true difference. Third, it is possible that a true difference between the data may be too small to be detected with our current simulation methods. The complete explanation may be a combination of these. Future

study to increase the precision of the simulation data is warranted to further investigate potential differences in the effectiveness of the three magnetic shielding configurations.

### *3.5 Volume, Mass, and Shielding Thickness Comparison*

To further investigate our findings, we calculated the total volume of magnetized space, total structural passive shielding mass, and effective passive shielding thickness of each magnetic shielding configuration. A summary of the results of these calculations is provided in Figure 84.

#### *3.5.1 Total Volume of Magnetized Space*

For the toroid configuration, the total volume of magnetized space was provided by four 2 cm thickness YBCO shielding toroids. The interior volume of the four toroids constitutes the magnetized space and is calculated by a toroid volume calculation  $(\pi r^2)(2\pi R)$  where  $r$  = the cross-sectional radius and  $R$  = the toroid radius. The total magnetized volume for the toroid configuration is then calculated by:

$$4 \times \pi (1.25 \text{ m})^2 \times 2 \pi (3.85 \text{ m}) = 475 \text{ m}^3$$

For the solenoid configuration, the total volume of magnetized space was provided by six 5 cm thickness YBCO shielding solenoids. For simplicity, we assumed the magnetized volume due to the correction solenoid and the superimposed fringe field of all 7 solenoids is negligible. Therefore, the interior volume of the six solenoids constitutes the magnetized space and is calculated by a cylinder volume calculation  $(\pi r^2 L)$  where  $r$  = the solenoid radius and  $L$  = the solenoid length. The total magnetized volume for the solenoid configuration is then calculated by:

$$6 \times \pi (4 \text{ m})^2 \times 20 \text{ m} = 6032 \text{ m}^3$$

For the race track configuration, the total volume of magnetized space was provided by 120 YBCO shielding “race track” elongated toroids. The interior volume of the 120 elongated toroids constitutes the magnetized space. This volume is a fairly complex shape due to the rounded ends of the 120 elongated toroids (see Figure 33). Therefore, we estimated the volume of magnetized space for this configuration with the summation of a cylinder subtraction  $(\pi L)(R^2 - r^2)$  and a full toroid (representing the upper and lower half-toroids)  $(\pi r_1^2)(2\pi r_2)$ , where  $L$  = the height of the cylinder,  $R$  = the distance from the z-axis to the outer edge of the shielding elongated toroids,  $r$  = the inner radius of the race track,  $r_1$  is the cross-sectional radius, and  $r_2$  is the toroid radius. The total magnetized volume for the race track configuration is then estimated by:

$$\pi \times 10 \text{ m} \times ((6.4 \text{ m})^2 - (2.8 \text{ m})^2) + \pi \times (1.8 \text{ m})^2 \times 2\pi \times 4.6 = 1335 \text{ m}^3$$

The trend in total magnetized volume Solenoid > Race Track > Toroid is noted and the data is included for comparison in Figure 84. The solenoid configuration demonstrated the steepest slope in effective dose and absorbed dose vs. magnetic field in many cases, which tends to agree with this result. This is also an important finding in that the larger volume structures would be more difficult to launch given launch vehicle fairing limitations. The larger volume structures may require more complex deployment or assembly on-orbit than would a shielding structure of more compact volume.

### 3.5.2 Total Structural Passive Shielding Mass

For the toroid configuration, the total structural passive shielding mass was provided by four 2 cm thickness YBCO shielding toroids. The volume of the toroids was calculated by a subtraction toroid  $(2\pi R)\pi(r_1^2 - r_2^2)$  where  $r_1$  is the outer cross-sectional radius,  $r_2$  is the inner cross-sectional radius,  $r_1 - r_2$  is the toroid wall thickness, and  $R$  is the radius of the toroid. This volume was then

multiplied by the density of YBCO ( $\rho = 6.3 \text{ g/cm}^3$  or  $6300 \text{ kg/m}^3$ ) to obtain the total mass. The total structural passive shielding mass for the toroid configuration is then calculated by:

$$4 \times 2\pi (3.85 \text{ m}) \times \pi ((1.25 \text{ m})^2 - (1.23 \text{ m})^2) \times 6300 \frac{\text{kg}}{\text{m}^3} = 95,000 \text{ kg} = 95 \text{ t}$$

For the solenoid configuration, the total structural passive shielding mass was provided by six 5 cm thickness YBCO shielding solenoids, six 1 cm thickness graphite support cylinders, 36 2.5 mm thickness graphite support plates, and a 5 cm thickness YBCO correction solenoid. The volumes of the solenoids were calculated by subtraction cylinders  $(\pi L)(r_1^2 - r_2^2)$  where  $L$  = the length of the solenoid,  $r_1$  = the outer radius of the solenoid,  $r_2$  = the inner radius of the solenoid, and  $r_1 - r_2$  is the solenoid wall thickness. These volumes were then multiplied by the density of YBCO ( $\rho = 6.3 \text{ g/cm}^3$  or  $6300 \text{ kg/m}^3$ ) for the shielding and correction solenoids or the density of graphite ( $\rho = 2.2 \text{ g/cm}^3$  or  $2200 \text{ kg/m}^3$ ) for the support cylinders to obtain the mass. The volumes of the graphite support plates were calculated as box volumes (length x width x height), and these volumes were multiplied by the density of graphite ( $\rho = 2.2 \text{ g/cm}^3$  or  $2200 \text{ kg/m}^3$ ) to obtain the mass. The total structural passive shielding mass for the solenoid configuration is then calculated by:

$$\begin{aligned} & \left( (6 \times \pi (20 \text{ m}) \times ((4 \text{ m})^2 - (3.95 \text{ m})^2) + \pi (20 \text{ m}) \times \pi ((3.2 \text{ m})^2 - (3.15 \text{ m})^2)) \times 6300 \frac{\text{kg}}{\text{m}^3} \right) \\ & + \left( (6 \times \pi (20 \text{ m}) \times ((0.5 \text{ m})^2 - (0.49 \text{ m})^2) + 36 \times 1.72 \text{ m} \times 0.00125 \text{ m} \times 20 \text{ m}) \right. \\ & \left. \times 2200 \frac{\text{kg}}{\text{m}^3} \right) = 1,100,000 \text{ kg} = 1100 \text{ t} \end{aligned}$$

For the race track configuration, the total structural passive shielding mass was provided by 120 YBCO “race track” elongated toroids and a 20 cm wall thickness titanium forming cylinder. Each elongated toroid is composed of two cylinders and two half-toroids. The volume of the elongated toroids was approximated by the summation of  $2\pi Lr^2$  and  $(2\pi R)\pi r^2$  where  $r$  is the radius of the cylinder and inner cross-sectional radius of the toroid,  $L$  is the height of cylinder, and  $R$  is the radius of the toroid. This volume was then multiplied by the density of YBCO ( $\rho = 6.3 \text{ g/cm}^3$  or  $6300 \text{ kg/m}^3$ ) to obtain the mass. The volume of the titanium forming cylinder was calculated by a subtraction cylinder  $(\pi L)(r_1^2 - r_2^2)$  where  $L$  = the height of the forming cylinder,  $r_1$  is the outer cross-sectional radius,  $r_2$  is the inner cross-sectional radius,  $r_1 - r_2$  is the cylinder wall thickness (20 cm). This volume was then multiplied by the density of titanium ( $\rho = 4.5 \text{ g/cm}^3$  or  $4500 \text{ kg/m}^3$ ) to obtain the mass. The total structural passive shielding mass for the solenoid configuration is then estimated by:

$$\begin{aligned} & \left( 120 \times (2\pi \times 10 \text{ m} \times 0.05 \text{ m}^2 + 2\pi \times 1.8 \text{ m} \times \pi \times (0.05 \text{ m})^2) \times 6300 \frac{\text{kg}}{\text{m}^3} \right) \\ & + \left( 10 \text{ m} \times \pi ((4.7 \text{ m})^2 - (4.5 \text{ m})^2) \times 4500 \frac{\text{kg}}{\text{m}^3} \right) = 2,702,320 \text{ kg} = 2702 \text{ t} \end{aligned}$$

The wide spread in these total structural passive shielding mass values indicated that this parameter did not affect the overall trend in shielding effectiveness by shielding configuration. Total structural passive shielding mass is included for comparison in Figure 84. This is also an important finding in that the higher mass structures would be more difficult to launch given launch vehicle capability limitations. The higher mass structures may require more complex deployment or assembly on-orbit than would a shielding structure of a smaller mass.

### 3.5.3 Effective Passive Shielding Thickness

We also calculated the effective shielding thickness for each magnetic shielding configuration for a particle incident along the x-axis (orthogonal to the spacecraft and shielding structure, avoiding the end caps).

For the toroid configuration, the effective passive shielding thickness is provided by 2 cm thickness YBCO ( $\rho = 6.3 \text{ g/cm}^3$ ) shielding toroids. The example incident particle would pass through two thicknesses of the toroid wall ( $2 \times 2 \text{ cm} = 4 \text{ cm}$ ) before striking the spacecraft in the passive shielding scenario. The effective passive shielding thickness for the toroid configuration is then estimated by:

$$4 \text{ cm} \times 6.3 \frac{\text{g}}{\text{cm}^3} = 25.2 \frac{\text{g}}{\text{cm}^2}$$

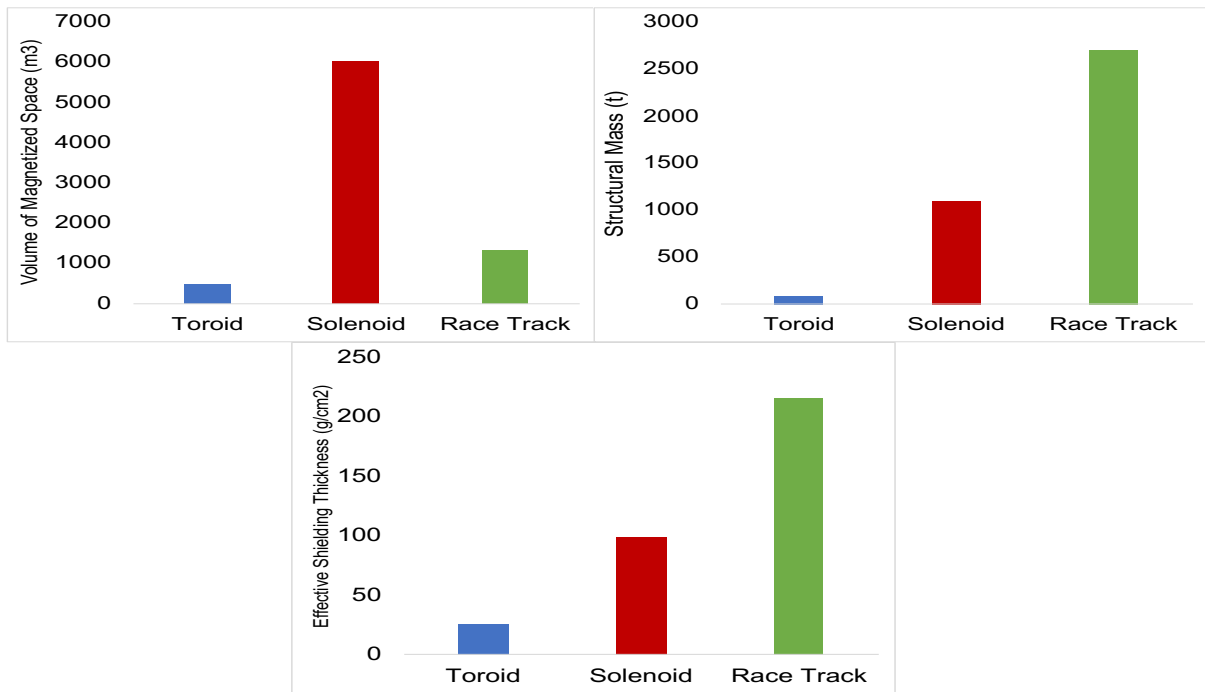
For the solenoid configuration, the effective passive shielding thickness is provided by 5 cm thickness YBCO ( $\rho = 6.3 \text{ g/cm}^3$ ) shielding solenoids, 1 cm graphite thickness ( $\rho = 2.2 \text{ g/cm}^3$ ) support structure, and a 5 cm thickness YBCO correction solenoid. For simplicity, we assumed the additional effective passive shielding thickness provided by the 2.5 mm thickness graphite support plates was negligible. The example incident particle would pass through two thicknesses of the solenoid wall (10 cm), two thicknesses of the graphite support structure ( $2 \times 1 \text{ cm} = 2 \text{ cm}$ ), and one thickness of the correction solenoid wall ( $2 \times 5 \text{ cm} = 10 \text{ cm}$ ) before striking the spacecraft in the passive shielding scenario. The effective passive shielding thickness for the solenoid configuration is then estimated by:

$$15 \text{ cm} \times 6.3 \frac{\text{g}}{\text{cm}^3} + 2 \text{ cm} \times 2.2 \frac{\text{g}}{\text{cm}^3} = 99 \frac{\text{g}}{\text{cm}^2}$$

For the race track configuration, the effective passive shielding thickness was provided by two 5 cm wall thickness YBCO ( $\rho = 6.3 \text{ g/cm}^3$ ) shielding elongated toroids and a 20 cm wall thickness titanium ( $\rho = 4.5 \text{ g/cm}^3$ ) forming cylinder. The example incident particle would pass through two thicknesses of the shielding elongated toroid ( $2 \times 10 \text{ cm} = 20 \text{ cm}$ ) and the titanium forming cylinder ( $1 \times 20 \text{ cm} = 20 \text{ cm}$ ) before striking the spacecraft in the passive shielding scenario. The effective passive shielding thickness for the race track configuration is then estimated by:

$$20 \text{ cm} \times 6.3 \frac{\text{g}}{\text{cm}^3} + 20 \text{ cm} \times 4.5 \frac{\text{g}}{\text{cm}^3} = 216 \frac{\text{g}}{\text{cm}^2}$$

The results of the effective passive shielding thickness calculations above indicate that the shielding structure of Toroid < Solenoid < Race Track. These data are included for comparison in Figure 84.



**Figure 84: Comparison of Total Volume of Magnetized Space, Total Structural Passive Shielding Mass, and Effective Shielding Thickness for Three Magnetic Shielding Configurations**

#### *3.5.4 Interpretation of Passive and Active Shielding Effects*

Effective dose results for the three magnetic shielding configurations can be interpreted by evaluating the setup parameters in the Monte Carlo simulations, such as the spatial and angular distributions of the particle source, the effective thickness of the passive shielding structure, the spatial distribution of the magnetic field, and the magnetic field strength.

First, we analyzed why the GCR dose from the three different magnetic field configurations are similar at the same magnetic field strength. In our Monte Carlo simulations, the charged particle source was defined as isotropic, which means particles can be emitted from the source point in any direction into the world volume with a uniform angular distribution. Consequently, the magnetic field received incident particles from any direction with a uniform angular distribution. Therefore, for a magnetic field vector along any direction, the incident particles were always isotropic and incident particles not parallel to the field vector were deflected. This isotropic characteristic makes the active shielding effect independent of the direction of the magnetic field vector. This helps explain why at the same magnetic field strength, the direction of the magnetic field vector based on the shielding geometry does not contribute to a large difference in shielding effectiveness.

Second, we analyzed why SPE effective dose (solar min/max, female/male, GCR/SPE) does not depend on the magnetic field configurations. Theoretically, because the low energy SPE protons are more easily deflected by the magnetic field, the active shielding effect should increase with the magnetic field strength. However, this was not observed in our SPE results (see Figures 85-88). For all three magnetic field configurations, we observed the SPE dose was almost constant for each setup, independent of the magnetic field strength. This can be explained as follows. As we know, although the fluence rate of SPE protons can be very high, the energy is relatively low and can be easily stopped by the passive structure. Therefore, the thickness of

the passive structure of the magnetic field also plays a role in the shielding effect. The effective shielding thickness of these three configurations follows this relationship: Toroid ( $30 \text{ g/cm}^2$ ) < Solenoid ( $99 \text{ g/cm}^2$ ) < Race Track ( $216 \text{ g/cm}^2$ ). All of these are greater than the penetration range of SPE protons ( $\sim 20 \text{ g/cm}^2$ ). Therefore, SPE protons incident on the magnetic field structure are mostly stopped via passive shielding. However, we can still see SPE dose in the toroid and race track configurations. These doses are contributed by the SPE protons incident on the end caps of the spacecraft where there is no magnetic field and the end cap thickness is thin ( $\sim 5 \text{ g/cm}^2$ ). We also observed that the SPE dose in the race track configuration is lower than in the toroid configuration. This is due to the additional passive structure (a series of YBCO half-toroids) of the race track in the end cap region of the spacecraft. For solenoid configuration, the SPE dose is almost zero. This is because there is a fringe magnetic field and passive structure further beyond the length of the spacecraft, both of which have greatly reduced the SPE dose. Note that the end cap conditions also apply to GCR particles, but the observed effect is smaller because the total GCR dose remains higher in all passive and magnetic shielding cases.

Third, we evaluated if increasing the passive structure thickness always reduces the effective dose. The general trend is that the GCR dose decreases with increasing passive structure thickness. At zero and low magnetic field values, we observed the magnitude of the effective dose followed Toroid > Solenoid > Race Track, though uncertainties were large enough to statistically obscure this difference. At high magnetic field (7 T), we found the magnitude of the effective dose using the solenoid configuration was lower than effective dose in the race track configuration, though uncertainties statistically obscured this difference as well. These potential differences may be explained using knowledge in particle physics. At higher magnetic field strengths, the dose to astronauts is mainly from high-energy GCR particles. One of the very important dosimetric characteristics of HZE ions is that the dose increases in the medium and then a Bragg peak of dose occurs at the end of the penetration range. This phenomenon makes

the shielding of GCR particles more complex. For example, ions energetic enough to completely penetrate the passive shielding structure may deposit the Bragg peak dose to the astronaut. This means increasing the effective thickness of the passive structure may result in the increase of dose to the astronaut inside the spacecraft. However, if the passive structure is further increased until it is thick enough to completely absorb the primary incident ions, the dose to the astronaut will be decreased. In addition, the primary incident ions interact with the passive structure to generate more secondary particles with strong penetration ability such as neutrons and nuclear fragments, which can complicate the effect of shielding. Therefore, the combination of passive structural shielding effect and active magnetic shielding effect makes it challenging to find the optimal shielding design.

Geng et al. observed that at a very high magnetic field (6 T), for GCR particles, increasing the thickness of the aluminum layer of the habitat is not an appropriate way for “shielding” radiation because it actually increased the effective dose to the astronaut inside the spacecraft (Geng et al. 2015). We also observed a similar result. For example, the effective thickness of our race track configuration was greater than the effective thickness of our solenoid configuration. At 7 T, we found that the race track effective dose was higher than the solenoid effective dose, as predicted (see Figure 82). However, at low magnetic fields, the thicker passive structure is still more effective in reducing dose because it can prevent the low-energy ions from entering the spacecraft. For magnetic fields of 1.5 T and 3 T, our results reflect the shielding effect of the passive structure.

### *3.6 Validation from Magnetic Shielding Literature*

Several prior studies on active magnetic shielding have been conducted by various teams within the space community. The relative effectiveness of the magnetic shielding designs depends on several factors including the choice of superconductor material, magnetic field,

simulation package used for radiation transport, dosimetry method employed, and phantom selection. Direct comparison of our results to prior studies is complicated by differences in these factors, however general comparisons may be made.

We attempted to make specific comparisons of our results with three prior magnetic shielding studies. We compared our results for the toroid configuration to a previous study by a Chinese research team (Geng et al. 2015), results for the solenoid configuration to the 2014 NASA Magnet Architectures and Active Radiation Shielding (MAARS) study (Westover 2014), and results for the race track configuration to the 2016 ESA Space Radiation Superconducting Shielding (SR2S) study (Vuolo et al. 2016). A summary of the designs from each of these studies and our own is provided in Table 22.

**Table 22: Summary of Key Parameters from This and Prior Magnetic Shielding Studies**

	Geng et al. 2015	Westover 2014	Vuolo et al. 2016	Ferrone 2020
<b>Magnetic Shielding Configuration</b>	Toroid	Solenoid	Race Track	Toroid Solenoid Race Track
<b>Human Phantom</b>	ICRP Reference Male	Water Cylinder w/BFO	Water Sphere	MIRD Male and Female (modified)
<b>Radiation Transport Code</b>	GEANT4	FLUKA	GRAS (GEANT4 variant)	GEANT4
<b>GCR Model</b>	Badhwar-O'Neill 2010	Cosmic Rays Effects on Micro Electronics (CRÈME) 2009	ISO15390	Badhwar-O'Neill 2014
<b>Solar Cycle</b>	Solar Min	Solar Min Solar Max	Solar Min	Solar Min Solar Max
<b>Magnetic Field Strength</b>	0, 2, 4, 6 T	1, 2.5 T	0, 2 T	0, 1.5, 3, 7 T
<b>Spacecraft Composition</b>	10 g/cm <sup>2</sup> Al	5 g/cm <sup>2</sup> Al	5 g/cm <sup>2</sup> Al	5 g/cm <sup>2</sup> Al
<b>Shielding Structure</b>	None	Multiple layers of mechanical, thermal, and electrical protection	MgB2 and titanium	Toroid: 25 g/cm <sup>2</sup> YBCO Solenoid: 94 g/cm <sup>2</sup> YBCO, graphite Race Track: 211 g/cm <sup>2</sup> YBCO, titanium
<b>Dosimetric Units Reported</b>	Effective Dose, E	Dose Equivalent, H	% Decrease in Effective Dose, E	Effective Dose, E Dose Equivalent, H

Geng et al. performed a study on active magnetic shielding using GEANT4 (Geng et al. 2015). They modeled the ICRP reference male phantom in a spacecraft with a 10 g/cm<sup>2</sup> aluminum layer as the passive structure and a toroidal magnetic field as the active shielding component (2 T, 4 T, 6 T). The GCR spectrum at solar min and the September 1989 SPE event (close to the worst-case SPE spectrum in our simulations) were simulated separately. Although they also defined an isotropic particle source, they assumed that the incident particles to the end caps of the

spacecraft were completely shielded by the spacecraft structures without depositing dose to the phantom. This assumption could underestimate the dose. For the worst-case SPE at 0 T magnetic field, using the male phantom, the effective dose was 0.262 Sv in the Geng study (Geng et al. 2015), while we calculated an effective dose of 0.346 Sv for the similar configuration in our study, a difference of 32%. This difference likely occurs because our aluminum spacecraft layer is 5 g/cm<sup>2</sup> while Geng's was 10 g/cm<sup>2</sup>, and we considered dose contributions from SPE particles incident on the spacecraft end caps while Geng did not.

At solar min, using the male phantom and zero magnetic field, the 700-day mission GCR effective dose was 1.11 Sv in the Geng study (Geng et al. 2015), while we calculated an effective dose of 2.24 Sv for a similar design in our study, a factor of two difference. This comparison confirms our result that the GCR effective dose is also reasonable compared to prior studies. The GCR effective dose in the Geng study is lower in part because they did not consider the dose from the particles incident on the end caps of the spacecraft.

Geng's results included the observation that the SPE effective dose was reduced with increasing magnetic field, while our SPE effective dose does not vary considerably with increasing magnetic field. The Geng study used only a 10 g/cm<sup>2</sup> aluminum layer without considering the passive structure of the magnetic field, while our total passive structure effective thickness was 30 g/cm<sup>2</sup> (5 g/cm<sup>2</sup> aluminum spacecraft + 25 g/cm<sup>2</sup> YBCO) in the toroid shielding configuration. An aluminum layer of 10 g/cm<sup>2</sup> is not thick enough to completely absorb the SPE protons, while an effective passive shielding thickness of 30 g/cm<sup>2</sup> structure is thick enough to stop SPE protons. Therefore, the magnetic field will play a role in deflecting the SPE protons in Geng's design, but not in ours. The SPE effective dose in our study is mainly contributed by the protons incident on the end caps of the spacecraft where there is no magnetic field for the toroid

configuration and the passive structure is thin, i.e.  $12 \text{ g/cm}^2$  on the end with the Orion capsule and  $5 \text{ g/cm}^2$  on the other end.

In the Geng study, total mission GCR effective dose at solar min was reduced to 95% (2 T), 85% (4 T), and 70% (6 T) (Geng et al. 2015). In our study, GCR effective dose was reduced to 80% (2 T interpolated), 66% (4 T interpolated) and 54% (6 T interpolated), differences of 15%, 19%, and 16% respectively. See Figure 85 below for details of the exponential fit used for interpolation. These results are resonable given the differences in geometry and radiation environment model. Our design is more effective in reducing GCR effective dose because the passive structure of our magnetic shielding configuration ( $30 \text{ g/cm}^2$  thickness) also contributes to shielding some particles whereas the Geng study did not model the magnetic shielding structure at all.

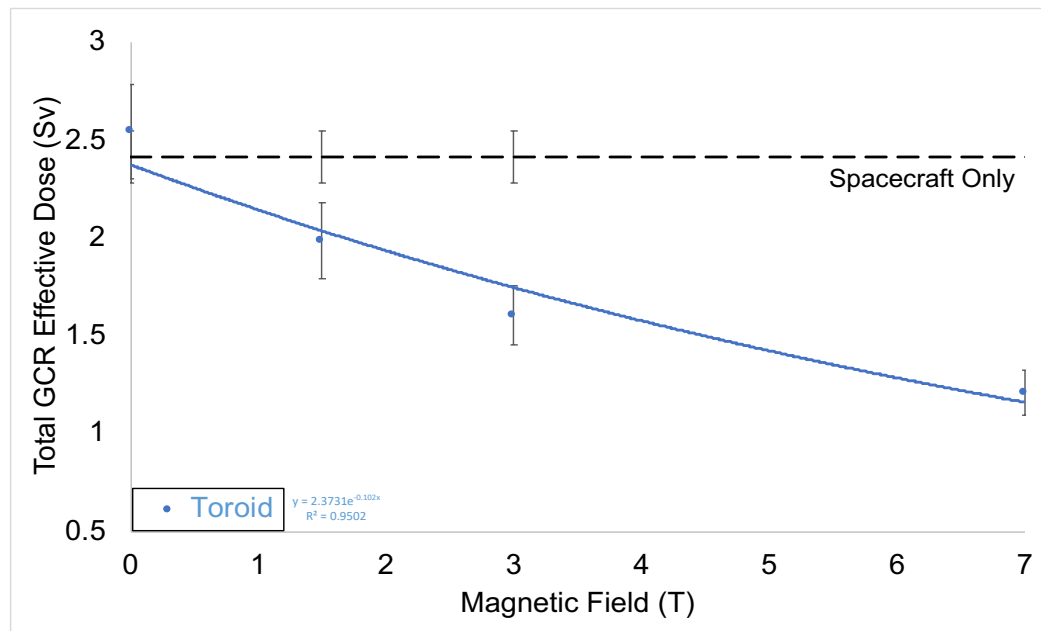


Figure 85: Total GCR Effective Dose vs. Magnetic Field (Toroid)

Westover also performed a study of active magnetic shielding in FLUKA (a Monte Carlo-based space radiation transport code). They pioneered the “6 + 1” solenoid configuration that we have reproduced in our study. However, there are several key differences between their setup and ours. First, the Westover study had an engineering focus and therefore selected a more detailed superconductor material design consisting of several layers of materials each performing different mechanical, thermal, and electrical functions. In contrast, in our study we represented the structure by only the superconducting material and did not simulate the numerous other layers. This may affect the effectiveness of passive shielding between the designs. Also, the Westover study assumed the superimposed fringe field of the 6 + 1 solenoid configuration to be negligible, where this field is modeled in our simulations. This should have a small effect on the overall results because the fringe field is typically less than 1% of the magnetic field strength within the shielding solenoids. Finally, Westover used a simple cylindrical phantom to simulate the astronaut rather than an organ-based human phantom such as the modified MIRD phantom used in our study (Westover 2014). The cylindrical phantom had a separate interior scoring volume to simulate the blood-forming organs (BFO) but was otherwise a water phantom. With a lower fidelity human phantom, approximations must be made to calculate effective dose from dose equivalent, and we would expect the effective dose results from the Westover study may differ from our results. Also, Westover uses a water phantom to represent the entire body mass (~70 kg), while we tallied dose only to the organs used to calculate effective dose (~10 kg). A smaller target volume and mass would lead to higher overall dose uncertainties in our study (assuming a similar number of initiated particles per simulation), though our method produces an overall higher fidelity to the human body.

The Westover study reported total annual dose equivalent to the cylindrical phantom in free space of 0.3 to 1.2 Sv depending on solar cycle (Westover 2014). For our minimal shielding case, we calculated an annual total effective dose of 1.8 to 3.1 Sv depending on solar cycle, averaged

across astronaut sex. Our effective dose calculations vary from and should not be directly compared to the Westover study's dose equivalent calculations because we calculated effective dose using organ doses rather than calculating total body dose equivalent. Also, our calculated dose is substantially higher because we used the worst-case SPE environment while Westover used an average SPE environment, and SPE dose contributes a large amount to total dose in minimal shielding cases.

The Westover study also calculated annual dose equivalent to the phantom blood-forming organs (BFO) of 0.34 Sv at 1 T (reported as 8 Tm) and 0.25 Sv at 2.5 T (reported as 20 Tm). For our solenoid configuration, we calculated an annual dose equivalent to active bone marrow of 0.48 Sv at 1 T (interpolated), and 0.39 Sv at 2.5 T (interpolated), averaged across astronaut sex, a consistent difference of 0.14 Sv. See Figure 86 below for details of the exponential fit used for interpolation. Despite substantial differences in design and dosimetry, the trend in effective dose reduction with magnetic field strength in our study is consistent with the Westover study between 1 and 2.5 T.

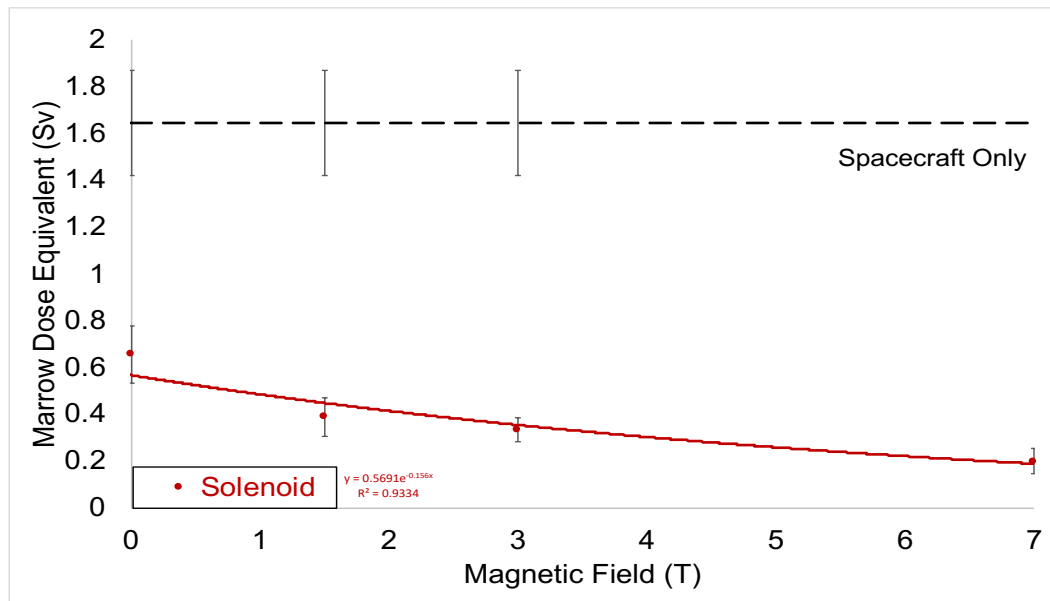


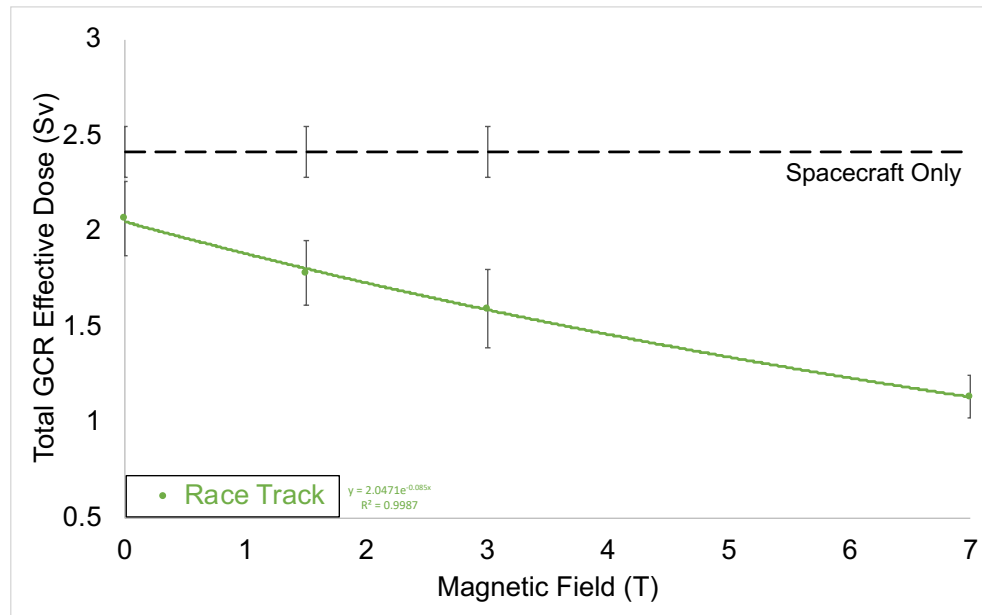
Figure 86: Annual Active Bone Marrow Dose Equivalent vs. Magnetic Field (Solenoid)

Vuolo et al. also performed a study of active magnetic shielding in GRAS (a GEANT4-based space radiation transport code only available within the European Space Agency) (Vuolo et al. 2016). They pioneered the “race track” shielding configuration that we have reproduced in our study. However, there are several key differences between their design and ours. First, Vuolo et al. chose magnesium diboride ( $\text{MgB}_2$ ,  $\rho = 2.57 \text{ g/cm}^3$ ) as their high temperature superconducting material, where we used YBCO ( $\rho = 6.3 \text{ g/cm}^3$ ). This means the effective passive shielding structural mass of our design is greater than that of the structure in the Vuolo study (141 vs.  $216 \frac{\text{g}}{\text{cm}^2}$ ). However, the Vuolo study included end cap structures to simulate propulsion elements and hydrogenous passive end cap shielding, thereby increasing the passive shielding structural mass in the end cap area.

Vuolo et al. also chose to focus only on GCRs, assuming what our study has confirmed, that SPE particles are adequately managed via passive shielding methods. Also, the Vuolo study used the ISO model for the GCR environment rather than the NASA Badhwar-O'Neill model used in our study. We anticipate this would introduce a small variation because the models are based on common data from actual measurements of the radiation environment in space.

The Vuolo study for GCRs reported a reduction in effective dose of 12% at 0 T and 45% at ~2 T (reported as 7.9 Tm) for the similar race track configuration (Vuolo et al. 2016). Our results for the race track configuration showed a reduction in total mission GCR effective dose of 15% at 0 T and 30% at 2 T (interpolated), averaged across astronaut sex, differences of 3% and 15%, respectively. See Figure 87 below for details of the exponential fit used for interpolation. Our race track design using YBCO instead of  $\text{MgB}_2$  provides a higher amount of passive shielding, which may explain why our dose calculations are closer to Vuolo's at 0 T than at 2 T. Further, there may be considerable error introduced in the translation of bending power (Tm) to magnetic field (T) because the conversion methods used are not typically described in detail and must be

estimated. Despite substantial differences in design and dosimetry, the trend in effective dose reduction with magnetic field strength in our study is consistent with the Vuolo study between 0 and 2 T.



**Figure 87: Total GCR Effective Dose vs. Magnetic Field (Race Track)**

From the above analyses, we have learned that the active magnetic shielding problem is complex, and multiple factors should be considered together in interpreting the calculation results (see Table 23). Factors such as the total volume of magnetized space, total structural passive shielding mass, and equivalent passive shielding thickness warrant further study to determine if there is a break point where passive shielding structure becomes the dominant factor rather than the magnetic field strength. Further, the complex relationship between incident particle energy and passive shielding material and thickness compounds the problem and makes direct comparisons of different shielding configurations difficult. Future studies should strive to use as many common parameters as possible, such as space environment model, human phantom, and dosimetry method to enable strong comparisons on the utility of different magnetic shielding designs.

**Table 23: Summary of Results Comparison with Literature**

	<b>Geng et al. 2015</b>		<b>Westover 2014</b>		<b>Vuolo et al. 2016</b>
<b>Dose Quantity</b>	Annual GCR Effective Dose	SPE Effective Dose	Annual BFO Dose Equivalent	Annual Dose Equivalent	GCR Effective Dose Reduction %
<b>Value</b>	1.11 Sv, reduced to 95% (2T), 85% (4 T), 70% (6 T)	0.262 Sv, minimal shielding	0.34 Sv (1 T), 0.25 Sv (2.5 T)	0.3 to 1.2 Sv, free space	reduction of 12% (0 T), 45% (2 T)
<b>Our Comparable Result</b>	2.24 Sv reduced to 80% (2 T), 66% (4 T), 54% (6 T)	0.346 Sv, minimal shielding	0.48 Sv (1 T), 0.39 Sv (2.5 T)	1.8 to 3.1 Sv, minimal shielding	reduction of 15% (0 T), 30% (2 T)
<b>Confounding Factors</b>	No magnetic shielding structure No source over end caps Different spacecraft aluminum thickness Different phantom		Different phantom Different SPE model		Uncertain conversion from Tm to T Different GCR model Different superconductor material Different phantom

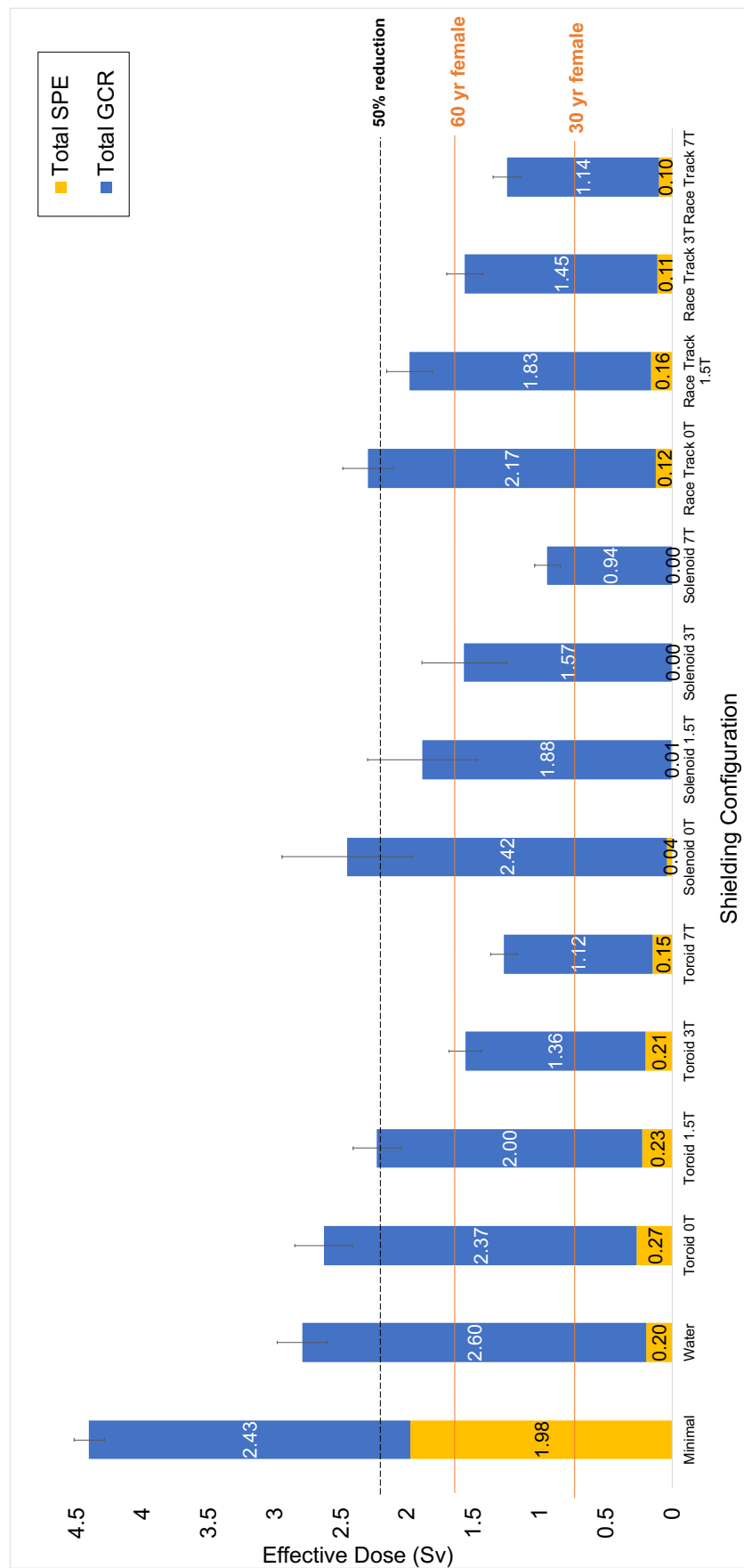
### 3.7 Results Summary

Figure 88, Figure 89, Figure 90, and Figure 91 summarize the results of hundreds of GEANT4 simulation runs, comprising over 1,000 hours of computing time. NASA's age- and sex-based effective dose limits are indicated by the horizontal lines on the graphs. The error bars in these figures represent  $\pm 2\sigma$  based on the standard error of the mean from the Monte Carlo simulations, or a 95% CI, as required by NASA (Cucinotta et al. 2012b), after the completion of error propagation calculations. Relative effectiveness of all shielding configurations as a percentage versus no shielding (spacecraft only case) is shown in Table 24.

In Figure 88, Figure 89, Figure 90, and Figure 91, we observed that no combination of solar cycle, astronaut sex, or astronaut age satisfied established limits for total mission effective dose. The solar particle dose alone at solar max exceeded the effective dose limits for the mission. Fortunately, all of the shielding configurations we tested, including passive water shielding, were effective at reducing solar particle dose by 90%+. However, the cosmic ray dose contributions remained high at magnetic fields of 1.5 to 3 T. This suggests that active magnetic shielding may only makes sense at higher magnetic fields; it might not be worth the investment to develop operational magnetic shielding technology at magnetic fields lower than 7 T.

In these figures and from these data, we also observed that the magnitude of the uncertainty of the total mission effective dose for certain scenarios was relatively large when compared to the magnitude of the uncertainty of the majority of scenarios we tested. Specifically, in the minimal shielding case using the male astronaut phantom, the percent error (derived from SEM from the GEANT4 code) was 7.4% whereas the percent error for the other minimal shielding scenarios ranged from 2.6% to 4.7%. The relatively larger uncertainty for one of our control cases may raise questions regarding the precision of the control case dose values which provided the underpinning of the remainder of our analysis. However, this variation in uncertainty was within the expected range based on the SEM calculation methodology, and the magnitude of the total mission effective dose for the minimal shielding, male astronaut scenario is consistent with the total mission effective doses for the other minimal shielding scenarios.

Additionally, it is interesting to note the effective dose delivered to male vs. female astronauts varies by less than 5%, but the effective dose *limits* imposed by NASA and other space agencies serve to magnify the apparent risk of a similar effective dose by 50% or more. This raises the question of whether male/female differences in the current effective dose calculation and/or dose limits are truly valid or if they are based on flawed radiation carcinogenesis data and should be reinvestigated. Further study is warranted on this topic.



**Figure 88: Total Mission (700 d) Effective Dose (Max, Female)**

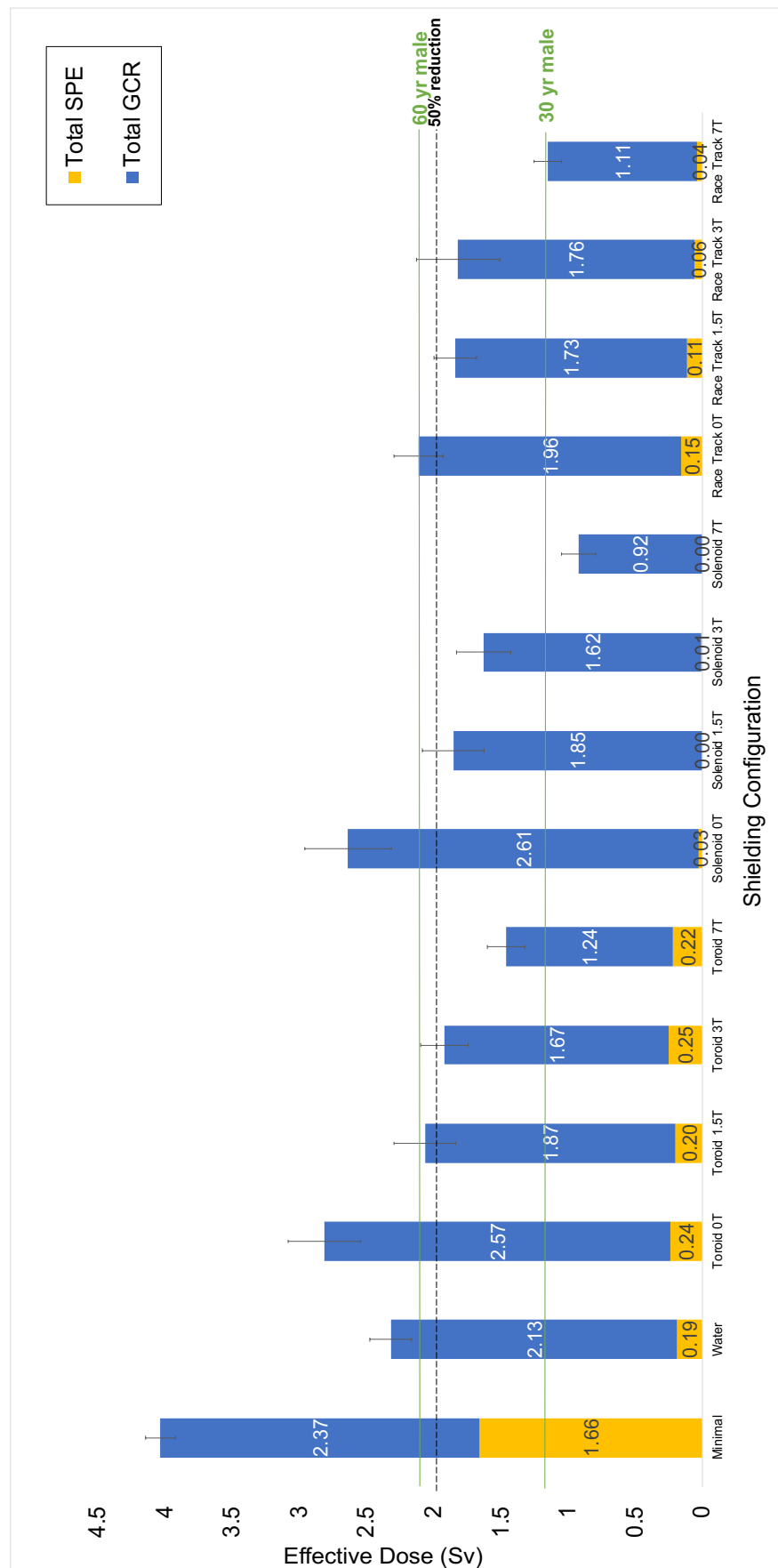


Figure 89: Total Mission (700 d) Effective Dose (Max, Male)

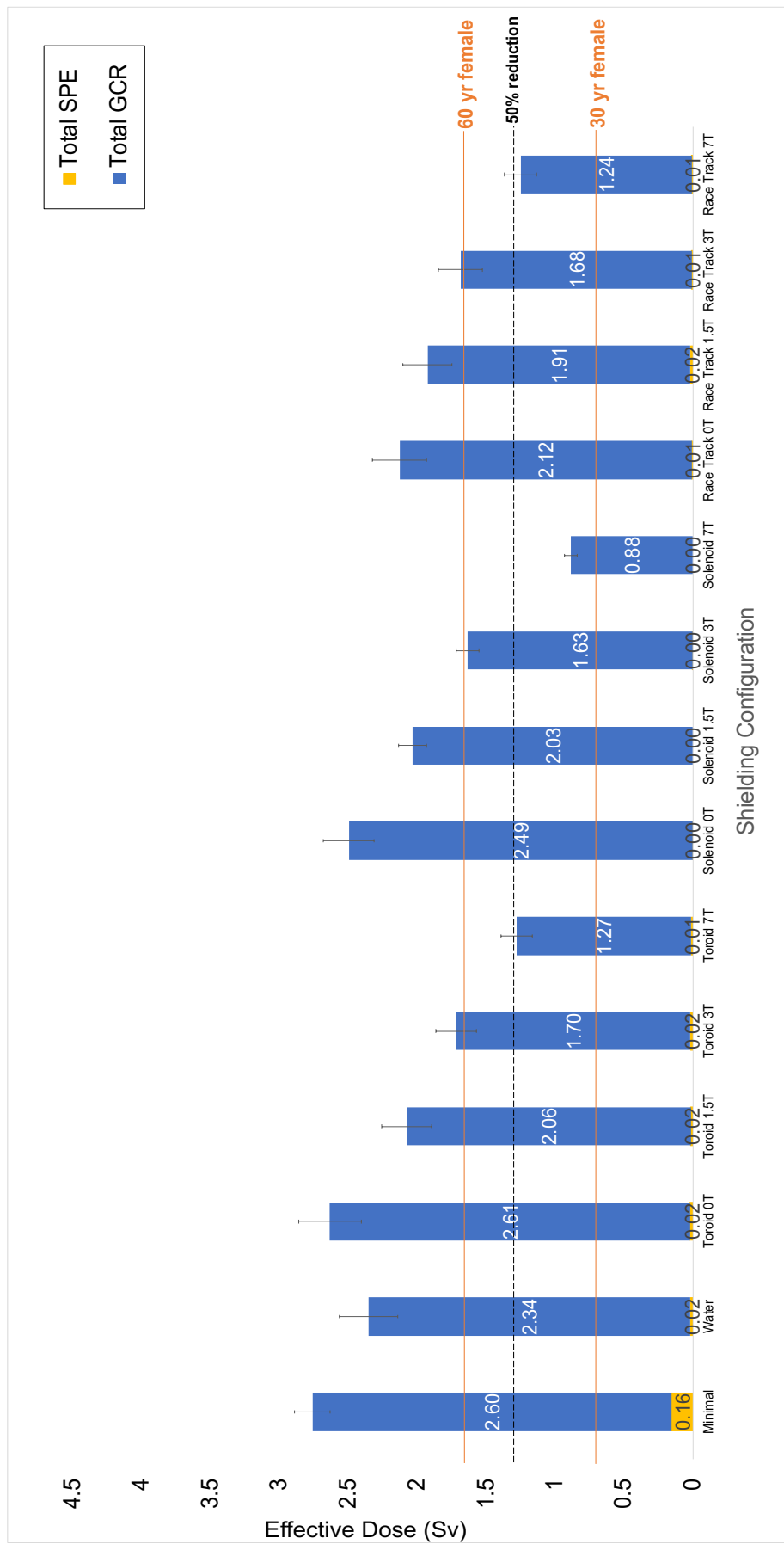
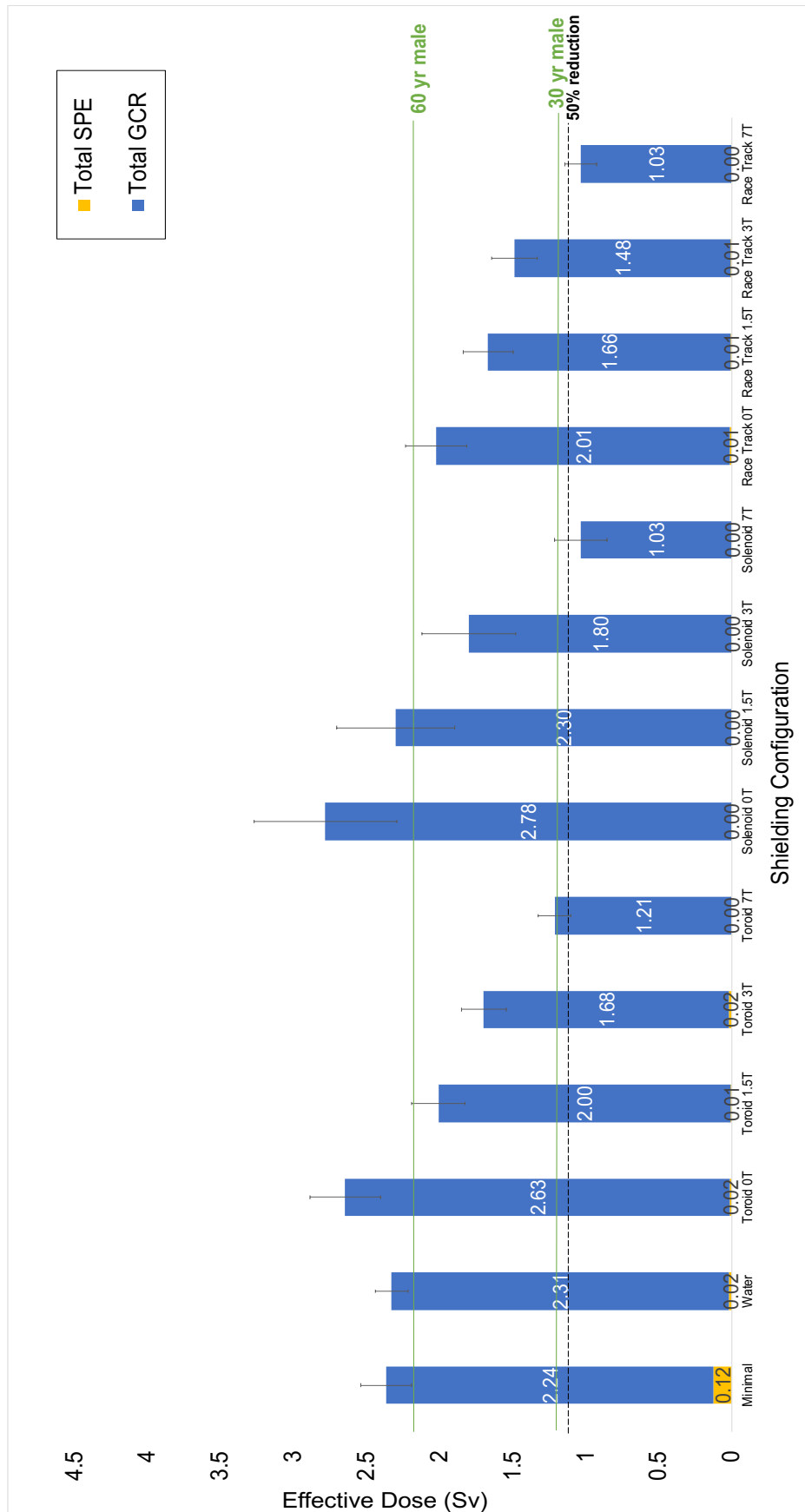


Figure 90: Total Mission (700 d) Effective Dose (Min, Female)



**Figure 91: Total Mission (700 d) Effective Dose (Min, Male)**

Table 24: Relative Effectiveness of Shielding Configurations

SCENARIO DEFINITION						TOTAL EFFECTIVE DOSE (700 d)		RELATIVE EFFECTIVENESS		95% CI [2 $\sigma$ ]	
Name	Mission Type	Solar Cycle	Shielding Type	Phantom	Particle Histories	E (Sv)	2 $\sigma$ (Sv) [95% CI]	% Effectiveness	$\sigma$	Low	High
Solar Max Female	Mars Flyby	Max	Minimal	MIRD Female	1.E+09	4.4070	0.1138	-	-	-	-
Solar Max Male	Mars Flyby	Max	Minimal	MIRD Male	1.E+09	4.0345	0.1115	-	-	-	-
Solar Min Female	Mars Flyby	Min	Minimal	MIRD Female	1.E+09	2.7587	0.1297	-	-	-	-
Solar Min Male	Mars Flyby	Min	Minimal	MIRD Male	1.E+09	2.3647	0.1749	-	-	-	-
Solar Max Female	Mars Flyby	Max	Water	MIRD Female	1.E+09	2.7963	0.1908	36.55%	1.23%	34.08%	39.02%
Solar Max Male	Mars Flyby	Max	Water	MIRD Male	1.E+09	2.3196	0.1549	42.51%	0.91%	40.68%	44.33%
Solar Min Female	Mars Flyby	Min	Water	MIRD Female	1.E+09	2.3523	0.2100	14.73%	1.52%	11.69%	17.78%
Solar Min Male	Mars Flyby	Min	Water	MIRD Male	1.E+09	2.3275	0.1122	1.57%	1.08%	-0.59%	3.73%
Solar Max Female	Mars Flyby	Max	Toroid 0T	MIRD Female	1.E+09	2.6344	0.2175	40.22%	1.51%	37.21%	43.24%
Solar Max Male	Mars Flyby	Max	Toroid 0T	MIRD Male	1.E+09	2.8138	0.2721	30.26%	2.16%	25.93%	34.58%
Solar Min Female	Mars Flyby	Min	Toroid 0T	MIRD Female	1.E+09	2.6304	0.2292	4.65%	1.73%	1.18%	8.12%
Solar Min Male	Mars Flyby	Min	Toroid 0T	MIRD Male	1.E+09	2.6468	0.2406	-11.93%	2.21%	-16.35%	-7.50%
Solar Max Female	Mars Flyby	Max	Toroid 1.5T	MIRD Female	1.E+07	2.2308	0.1815	49.38%	1.15%	47.09%	51.68%
Solar Max Male	Mars Flyby	Max	Toroid 1.5T	MIRD Male	1.E+07	2.0655	0.2287	48.80%	1.62%	45.57%	52.04%
Solar Min Female	Mars Flyby	Min	Toroid 1.5T	MIRD Female	1.E+07	2.0777	0.1810	24.69%	1.24%	22.21%	27.16%
Solar Min Male	Mars Flyby	Min	Toroid 1.5T	MIRD Male	1.E+07	2.0078	0.1828	15.09%	1.60%	11.89%	18.29%
Solar Max Female	Mars Flyby	Max	Toroid 3T	MIRD Female	1.E+07	1.5652	0.1205	64.48%	0.69%	63.11%	65.86%
Solar Max Male	Mars Flyby	Max	Toroid 3T	MIRD Male	1.E+07	1.9195	0.1768	52.42%	1.09%	50.24%	54.61%
Solar Min Female	Mars Flyby	Min	Toroid 3T	MIRD Female	1.E+07	1.7166	0.1490	37.78%	0.98%	35.82%	39.73%
Solar Min Male	Mars Flyby	Min	Toroid 3T	MIRD Male	1.E+07	1.6976	0.1538	28.21%	1.36%	25.50%	30.92%
Solar Max Female	Mars Flyby	Max	Toroid 7T	MIRD Female	1.E+07	1.2727	0.1025	71.12%	0.59%	69.95%	72.29%
Solar Max Male	Mars Flyby	Max	Toroid 7T	MIRD Male	1.E+07	1.4602	0.1403	63.81%	0.80%	62.20%	65.41%
Solar Min Female	Mars Flyby	Min	Toroid 7T	MIRD Female	1.E+07	1.2776	0.1112	53.69%	0.73%	52.23%	55.15%
Solar Min Male	Mars Flyby	Min	Toroid 7T	MIRD Male	1.E+07	1.2118	0.1106	48.75%	1.07%	46.61%	50.90%
Solar Max Female	Mars Flyby	Max	Solenoid 0T	MIRD Female	1.E+09	2.4564	0.2910	44.26%	2.44%	39.38%	49.14%
Solar Max Male	Mars Flyby	Max	Solenoid 0T	MIRD Male	1.E+09	2.6372	0.2244	34.63%	1.57%	31.49%	37.77%
Solar Min Female	Mars Flyby	Min	Solenoid 0T	MIRD Female	1.E+09	2.4950	0.1842	9.56%	1.27%	7.02%	12.10%
Solar Min Male	Mars Flyby	Min	Solenoid 0T	MIRD Male	1.E+09	2.7814	0.2276	-17.62%	2.06%	-21.74%	-13.50%
Solar Max Female	Mars Flyby	Max	Solenoid 1.5T	MIRD Female	1.E+07	1.8880	0.2142	57.16%	1.47%	54.22%	60.10%
Solar Max Male	Mars Flyby	Max	Solenoid 1.5T	MIRD Male	1.E+07	1.8546	0.2282	54.03%	1.61%	50.81%	57.26%
Solar Min Female	Mars Flyby	Min	Solenoid 1.5T	MIRD Female	1.E+07	2.0311	0.1032	26.37%	0.69%	25.00%	27.75%
Solar Min Male	Mars Flyby	Min	Solenoid 1.5T	MIRD Male	1.E+07	2.3020	0.2046	2.65%	1.81%	-0.97%	6.27%
Solar Max Female	Mars Flyby	Max	Solenoid 3T	MIRD Female	1.E+07	1.5735	0.2192	64.30%	1.53%	61.24%	67.35%
Solar Max Male	Mars Flyby	Max	Solenoid 3T	MIRD Male	1.E+07	1.6275	0.2014	59.66%	1.32%	57.01%	62.31%
Solar Min Female	Mars Flyby	Min	Solenoid 3T	MIRD Female	1.E+07	1.6351	0.0830	40.73%	0.59%	39.54%	41.91%
Solar Min Male	Mars Flyby	Min	Solenoid 3T	MIRD Male	1.E+07	1.8002	0.2206	23.87%	1.98%	19.91%	27.84%
Solar Max Female	Mars Flyby	Max	Solenoid 7T	MIRD Female	1.E+07	0.9434	0.0968	78.59%	0.56%	77.48%	79.71%
Solar Max Male	Mars Flyby	Max	Solenoid 7T	MIRD Male	1.E+07	0.9199	0.1294	77.20%	0.73%	75.74%	78.66%
Solar Min Female	Mars Flyby	Min	Solenoid 7T	MIRD Female	1.E+07	0.8823	0.0452	68.02%	0.47%	67.07%	68.96%
Solar Min Male	Mars Flyby	Min	Solenoid 7T	MIRD Male	1.E+07	1.0332	0.1806	56.31%	1.58%	53.15%	59.47%
Solar Max Female	Mars Flyby	Max	Race Track 0T	MIRD Female	1.E+09	2.2981	0.1925	47.85%	1.25%	45.35%	50.35%
Solar Max Male	Mars Flyby	Max	Race Track 0T	MIRD Male	1.E+09	2.1106	0.1822	47.69%	1.14%	45.40%	49.97%
Solar Min Female	Mars Flyby	Min	Race Track 0T	MIRD Female	1.E+09	2.1269	0.1982	22.90%	1.40%	20.10%	25.71%
Solar Min Male	Mars Flyby	Min	Race Track 0T	MIRD Male	1.E+09	2.0218	0.2104	14.50%	1.87%	10.76%	18.24%
Solar Max Female	Mars Flyby	Max	Race Track 1.5T	MIRD Female	1.E+07	1.9852	0.1712	54.95%	1.06%	52.84%	57.07%
Solar Max Male	Mars Flyby	Max	Race Track 1.5T	MIRD Male	1.E+07	1.8397	0.1606	54.40%	0.96%	52.49%	56.31%
Solar Min Female	Mars Flyby	Min	Race Track 1.5T	MIRD Female	1.E+07	1.9239	0.1784	30.26%	1.22%	27.83%	32.69%
Solar Min Male	Mars Flyby	Min	Race Track 1.5T	MIRD Male	1.E+07	1.6675	0.1728	29.48%	1.51%	26.46%	32.51%
Solar Max Female	Mars Flyby	Max	Race Track 3T	MIRD Female	1.E+07	1.5659	0.1357	64.47%	0.78%	62.90%	66.04%
Solar Max Male	Mars Flyby	Max	Race Track 3T	MIRD Male	1.E+07	1.8180	0.2124	54.94%	1.44%	52.06%	57.82%
Solar Min Female	Mars Flyby	Min	Race Track 3T	MIRD Female	1.E+07	1.6832	0.1584	38.99%	1.05%	36.89%	41.08%
Solar Min Male	Mars Flyby	Min	Race Track 3T	MIRD Male	1.E+07	1.4865	0.1570	37.14%	1.38%	34.38%	39.90%
Solar Max Female	Mars Flyby	Max	Race Track 7T	MIRD Female	1.E+07	1.2464	0.1067	71.72%	0.61%	70.50%	72.93%
Solar Max Male	Mars Flyby	Max	Race Track 7T	MIRD Male	1.E+07	1.1503	0.1054	71.49%	0.59%	70.31%	72.67%
Solar Min Female	Mars Flyby	Min	Race Track 7T	MIRD Female	1.E+07	1.2489	0.1186	54.73%	0.77%	53.18%	56.27%
Solar Min Male	Mars Flyby	Min	Race Track 7T	MIRD Male	1.E+07	1.0347	0.1074	56.24%	1.05%	54.14%	58.35%

### 3.8 Relative Contribution by Incident Radiation Types

Our team was interested in determining the relative effective dose contribution of the different types of primary space radiation particle: SPE, GCR proton, GCR alpha, GCR heavy ions. The results of this analysis in Figure 92 through Figure 99 show that GCR protons dominate the effective dose contribution in all cases except the unshielded solar max case, where the solar protons dominate the effective dose contribution. The error bars in these figures represent the standard error of the mean from the Monte Carlo simulations.

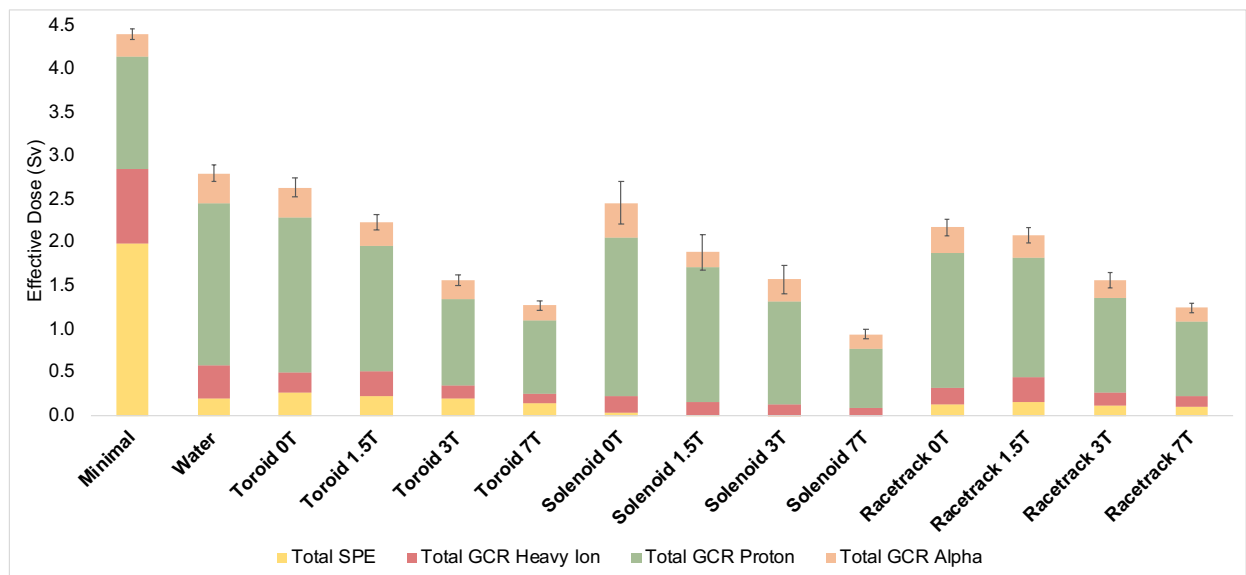


Figure 92: Total Mission (700 d) Effective Dose Contributions (Max, Female)

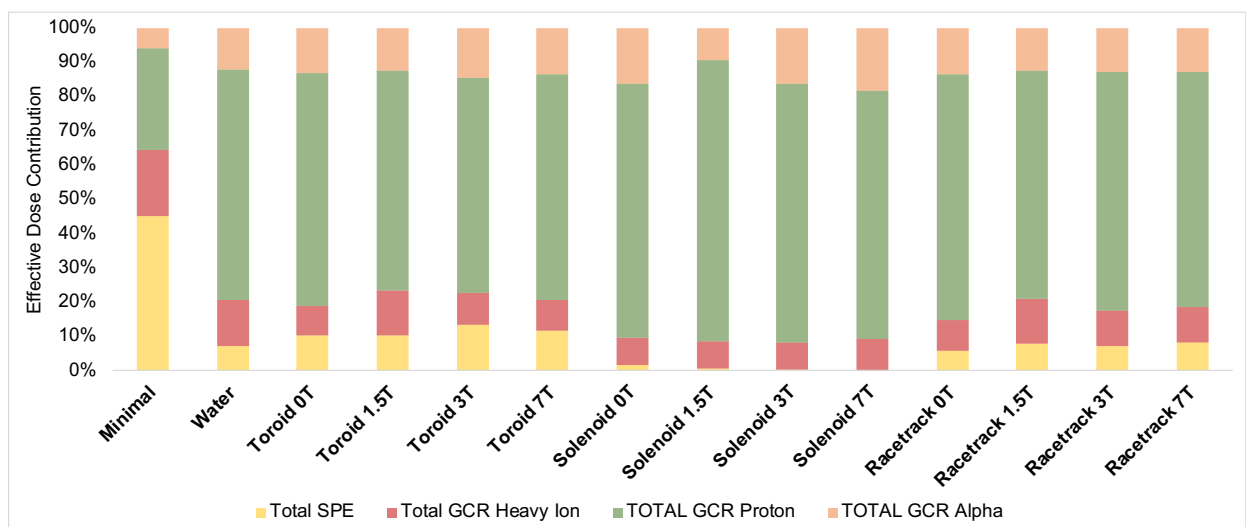
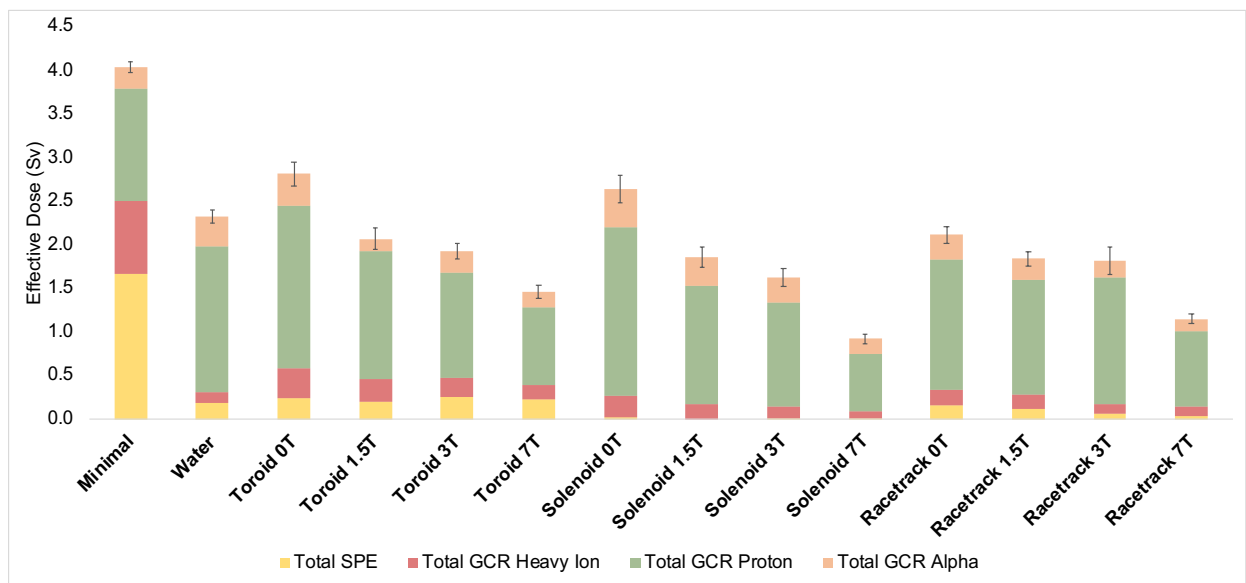
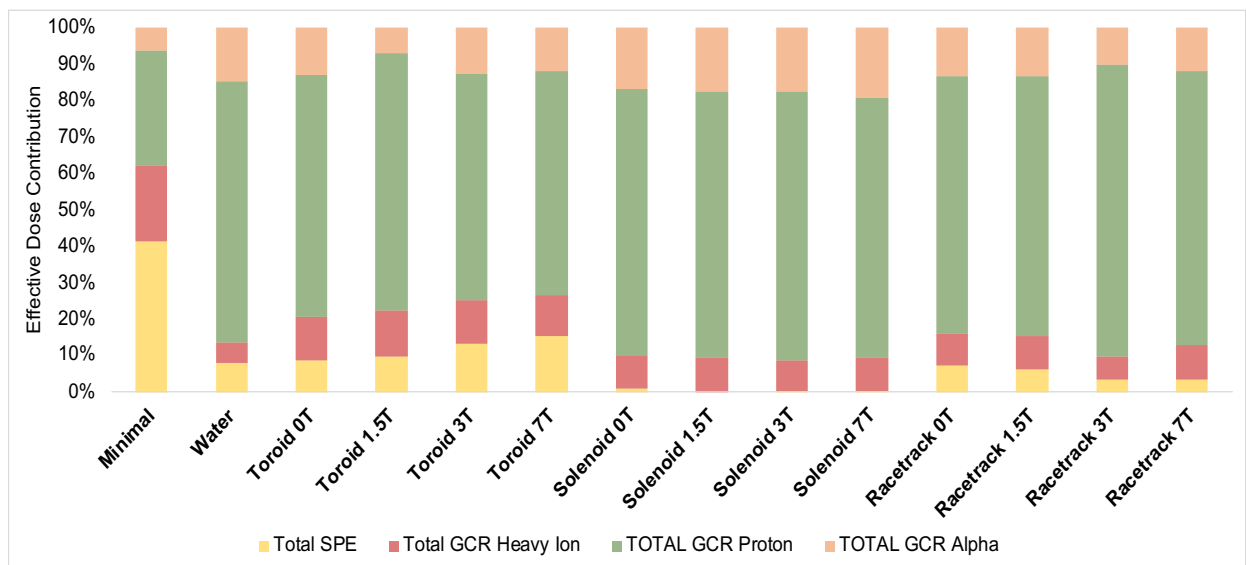


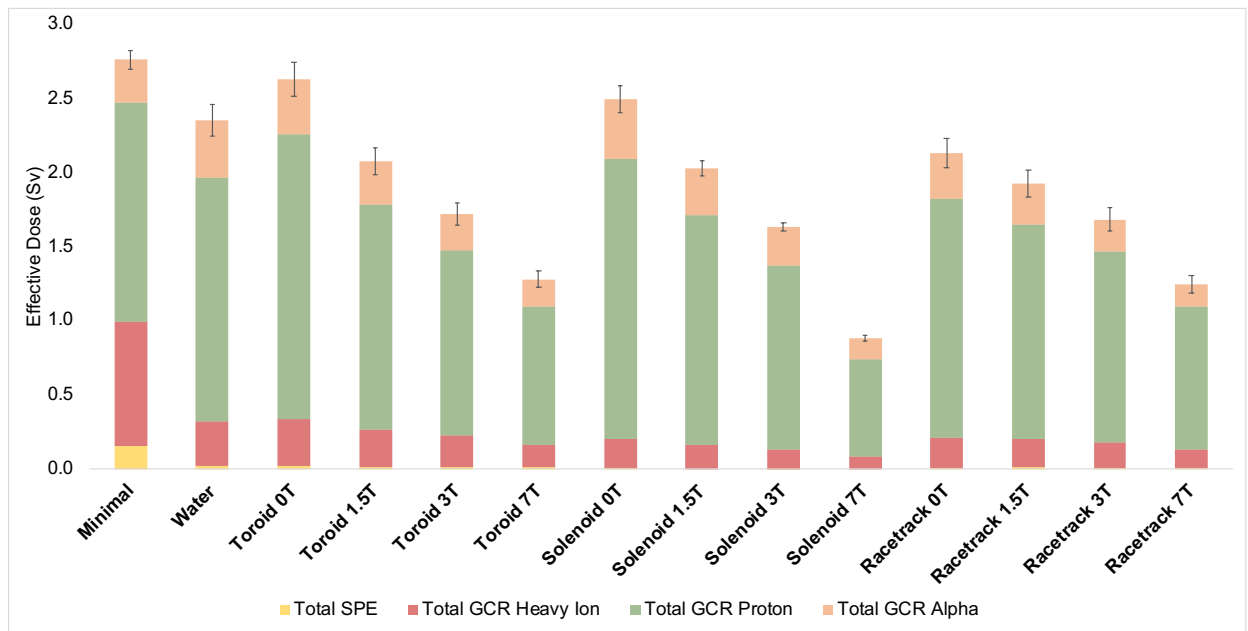
Figure 93: Total Mission (700 d) Effective Dose Normalized Contributions (Max, Female)



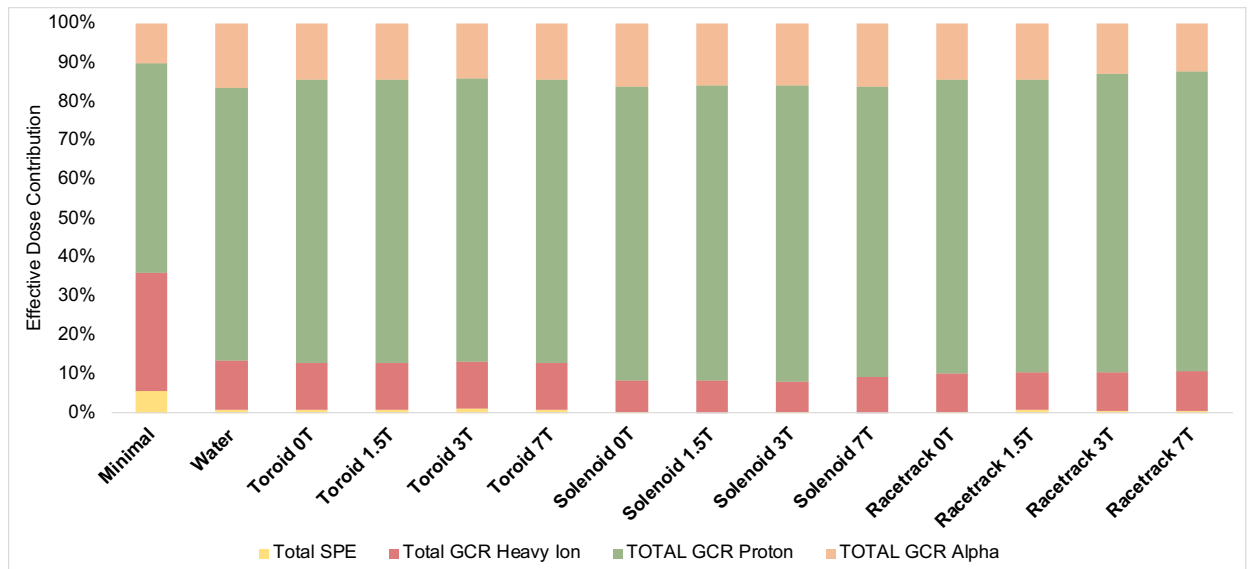
**Figure 94: Total Mission (700 d) Effective Dose Contributions (Max, Male)**



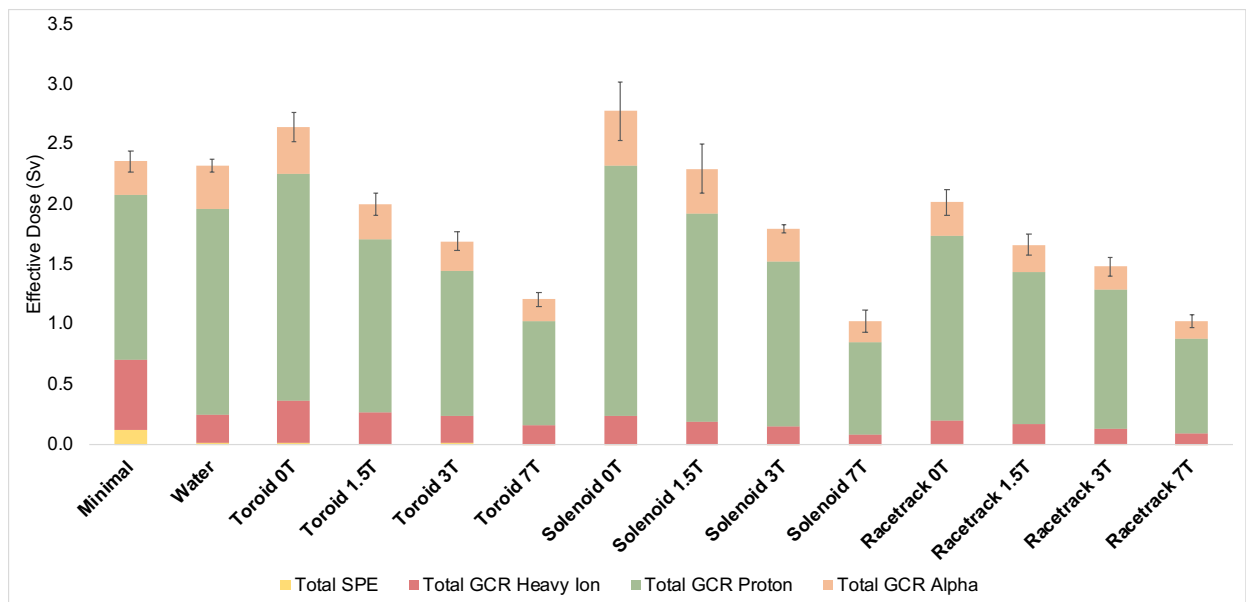
**Figure 95: Total Mission (700 d) Effective Dose Normalized Contributions (Max, Male)**



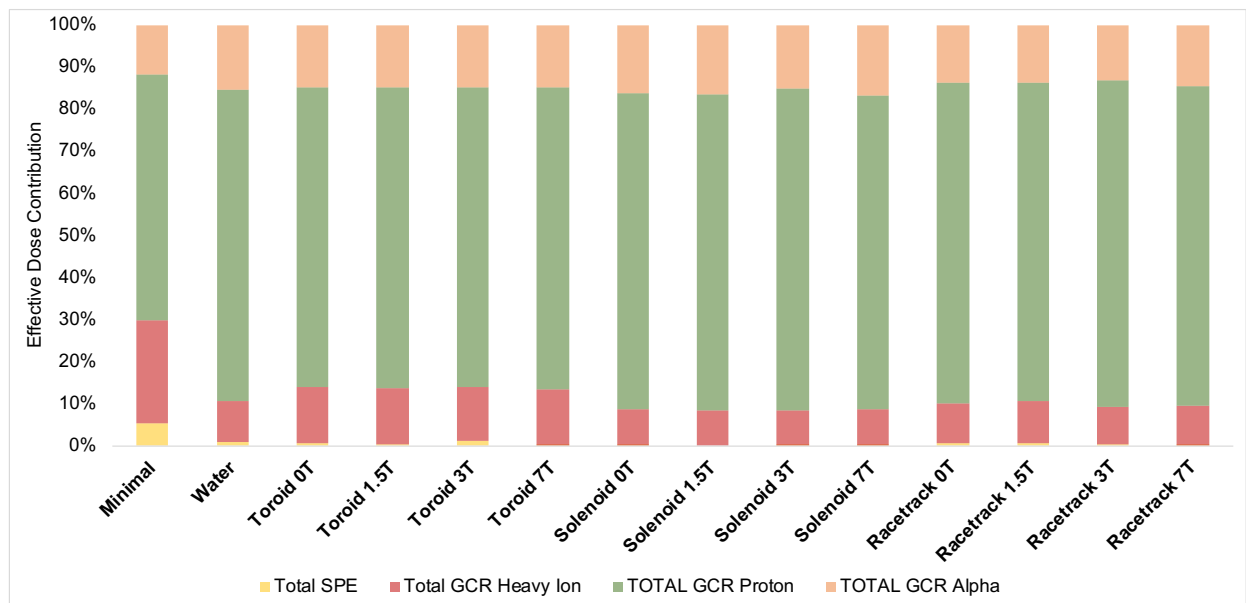
**Figure 96: Total Mission (700 d) Effective Dose Contributions (Min, Female)**



**Figure 97: Total Mission (700 d) Effective Dose Normalized Contributions (Min, Female)**



**Figure 98: Total Mission (700 d) Effective Dose Contributions (Min, Male)**



**Figure 99: Total Mission (700 d) Effective Dose Normalized Contributions (Min, Male)**

### 3.9 Organ Dose Equivalent Analysis

Our team was also interested in investigating the relative dose equivalent delivered to the individual organs in our phantom in each scenario. For both the male and female phantoms, we tallied dose for active bone marrow, brain, thyroid, lungs, heart, thymus, stomach, large intestine, small intestine, liver, kidneys, adrenals, pancreas, spleen, and bladder. For the female phantom, we added uterus, ovaries, and breasts. For the male phantom, we added testes.

The data are mostly well-behaved, that is the magnitude of the individual organ dose equivalents decrease with increasing field strength as expected. However, we observed a few exceptions in organ dose equivalents, where the dose equivalent delivered to an individual organ volume is higher than we expected based on the overall trends. The majority of these relatively high individual organ doses applied to the smaller organ volumes (adrenals, ovaries, thymus, etc.). This variation is likely due to the fact that there is a larger impact of any high LET particle on the smaller volumes' dose equivalents due to the small relative masses. Further, we simulated a large number of particles for each simulation in this study ( $10^7$  -  $10^9$ ), but that corresponds to a short real time duration (~1 second), so any relatively large organ dose equivalent from a single simulation is magnified when we scale the dose out to 700 days or even 1 day.

Because of the variation in the individual organ dose equivalents due to the reasons described above, we performed a qualitative analysis of the organ doses and used average dose equivalents across several simulations. In this way, the organ doses are presented as a set of data, and often weighted and/or averaged over different simulations, to mitigate the impact of any single high organ dose equivalent that may appear to be an outlier. In all other cases, our data is presented as total dose or total effective dose, which also mitigates the effect of any single high organ dose equivalent. The individual organ dose equivalent data are provided in Table 30 to Table 87 in Appendix A10.

Figure 100, Figure 101, and Figure 102 depict several visualizations of the organ dose equivalents delivered in each shielding scenario for the solar max, female astronaut case. These figures are repeated for the solar max, male case in Figure 103, Figure 104, and Figure 105, the solar min, female astronaut case in Figure 106, Figure 107, and Figure 108, and the solar min, male astronaut case in Figure 109, Figure 110, and Figure 111. Because the solar cycle position only provides a significant difference in dose in the minimal shielding case, it is reasonable to average the solar max and solar min cases for the male and female. Averages for the female astronaut case are displayed in Figure 112, Figure 113, and Figure 114, and averages for the male astronaut case are displayed in Figure 115, Figure 116, and Figure 117.

For each case, the first plot is a stacked column chart showing the normalized organ dose equivalents for each organ in each shielding configuration. While we can see that active bone marrow and brain receive a slightly larger normalized dose equivalent in many cases, these figures show that the organ dose equivalents are fairly uniform across the board; no one scenario, shielding configuration, or organ immediately stands out as dramatically different than its peers. The second plot for each is a simple bar graph showing the average inverse rank of each organ across each shielding configuration. The ranks were calculated for each shielding configuration by sorting the organ dose equivalents, assigning 1 to the lowest/least important through 16 or 18 for the highest/most important. The bar graph then plots the average inverse rank for each organ across all the shielding configuration options.

The bubble plots for each case display the relative organ dose equivalent rank for each organ analyzed and the size of each bubble represents the frequency that organ received that rank. From these plots we can infer the relative importance of each organ based on the relative organ dose equivalent from scenarios using many different shielding configurations.

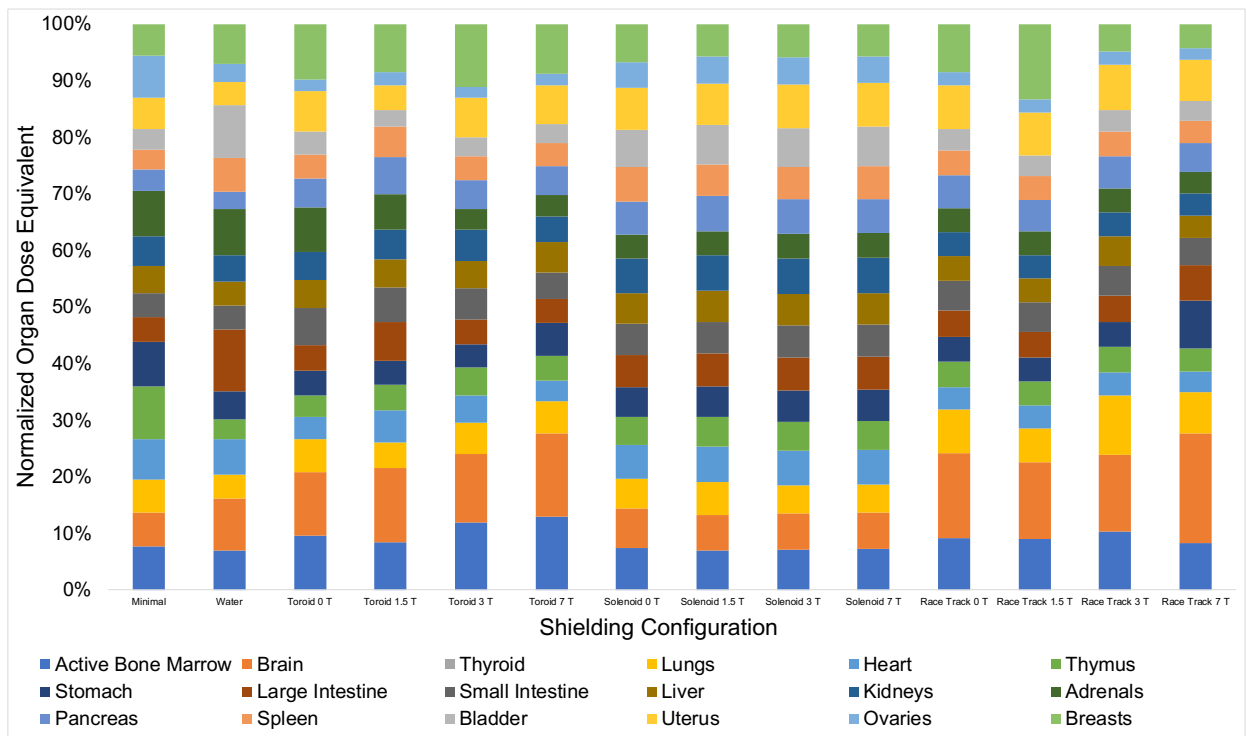


Figure 100: Organ Dose Equivalent, Normalized (Max, Female)

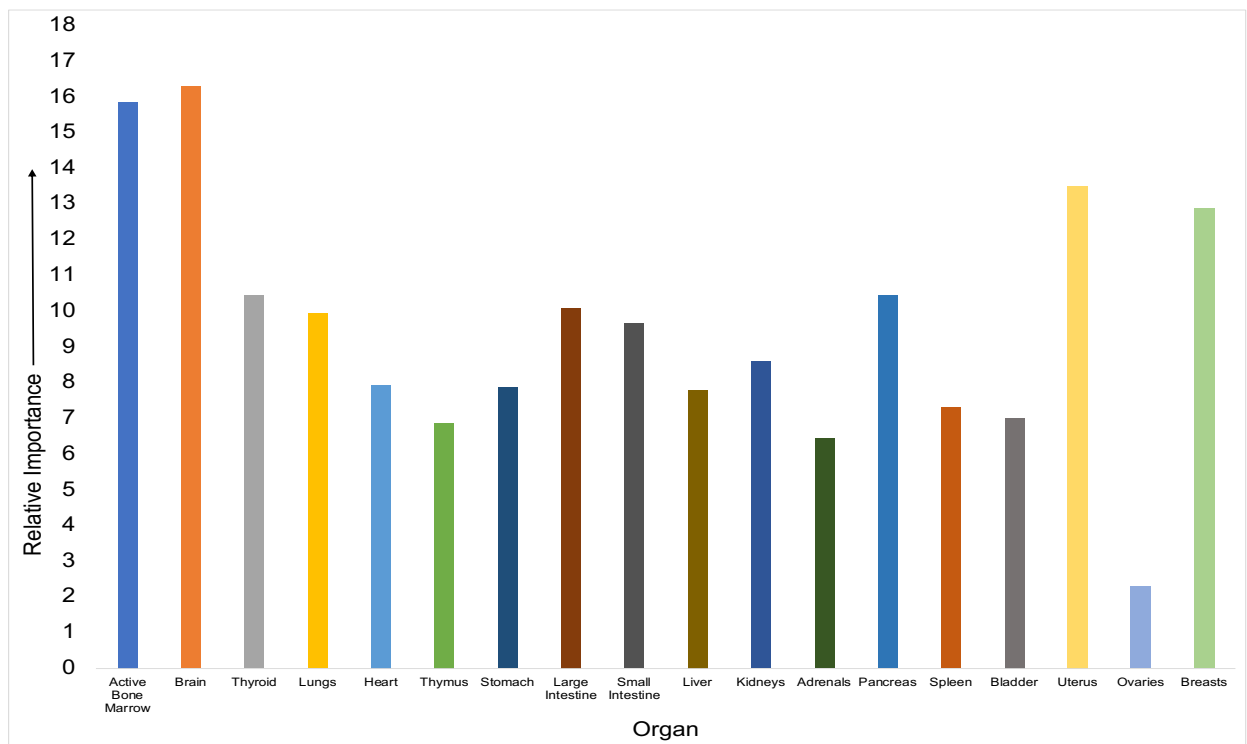


Figure 101: Organ Dose Equivalent, Average Inverse Rank (Max, Female)

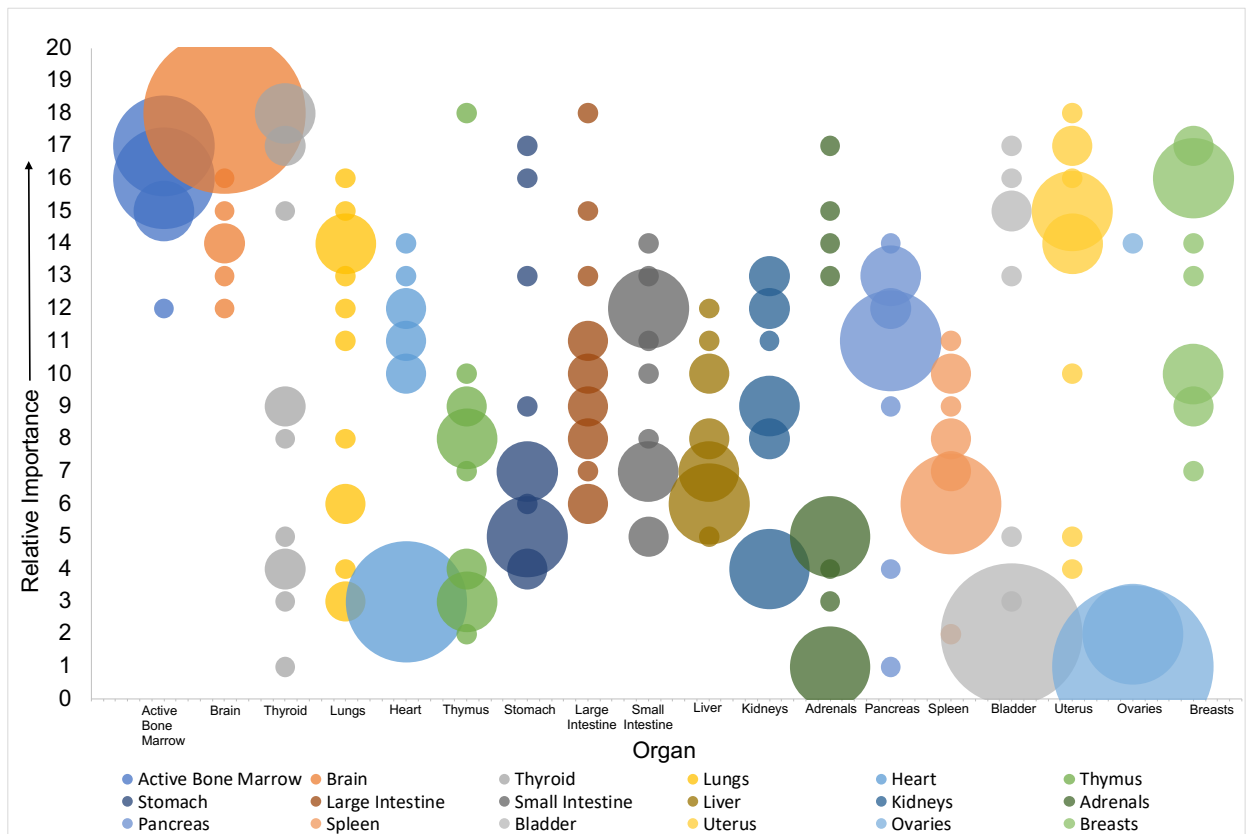


Figure 102: Organ Dose Equivalent, Frequency of Inverse Rank (Max, Female)

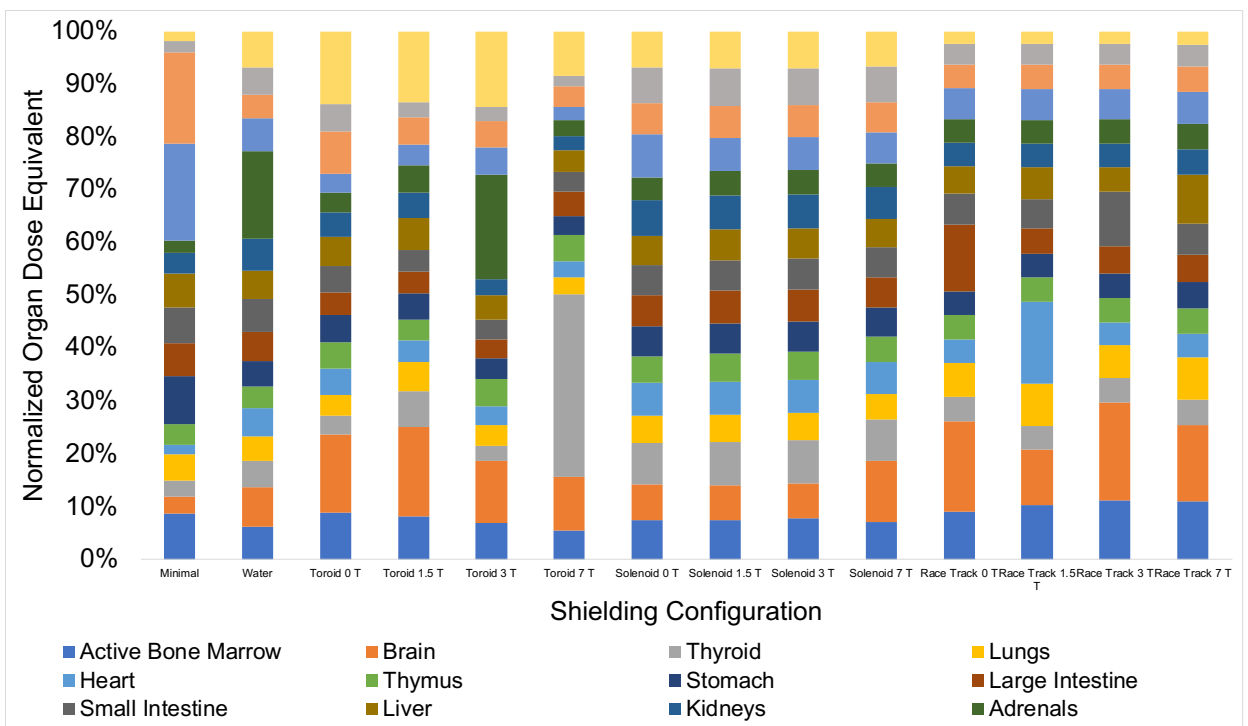


Figure 103: Organ Dose Equivalent, Normalized (Max, Male)

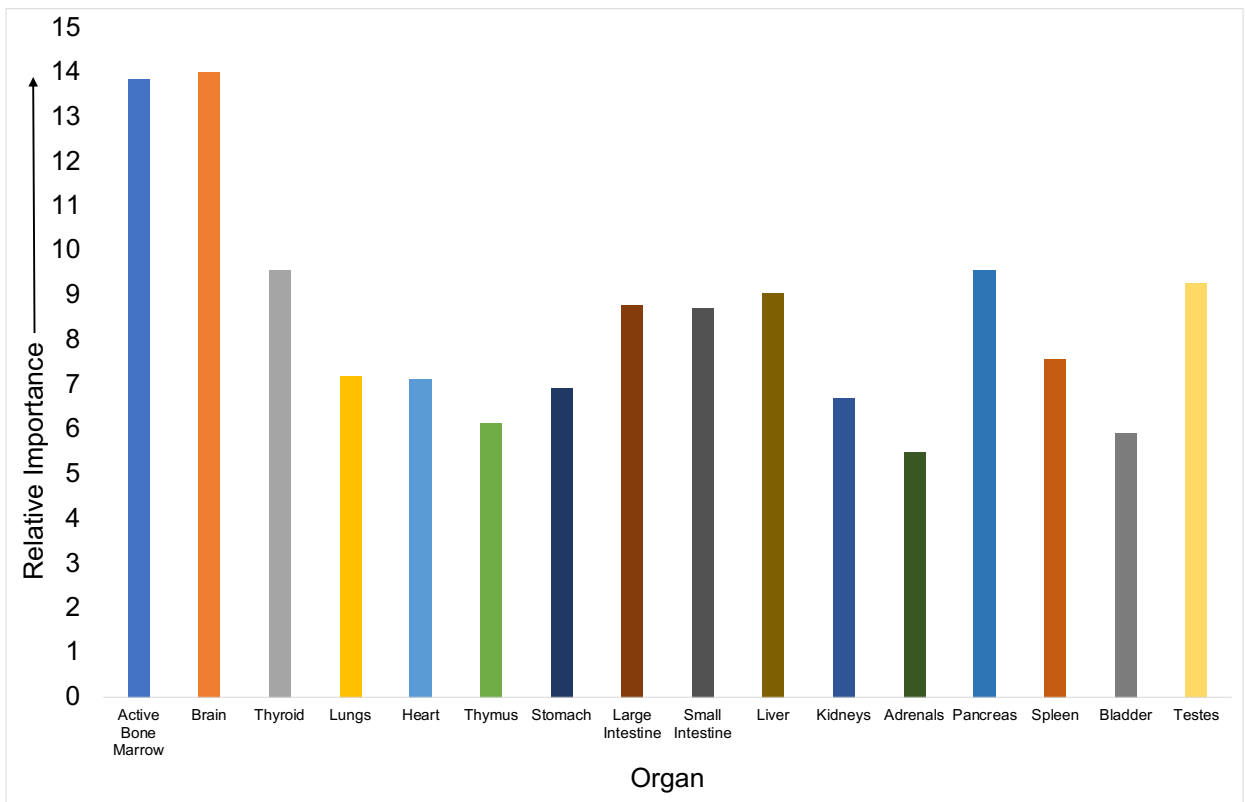


Figure 104: Organ Dose Equivalent, Average Inverse Rank (Max, Male)

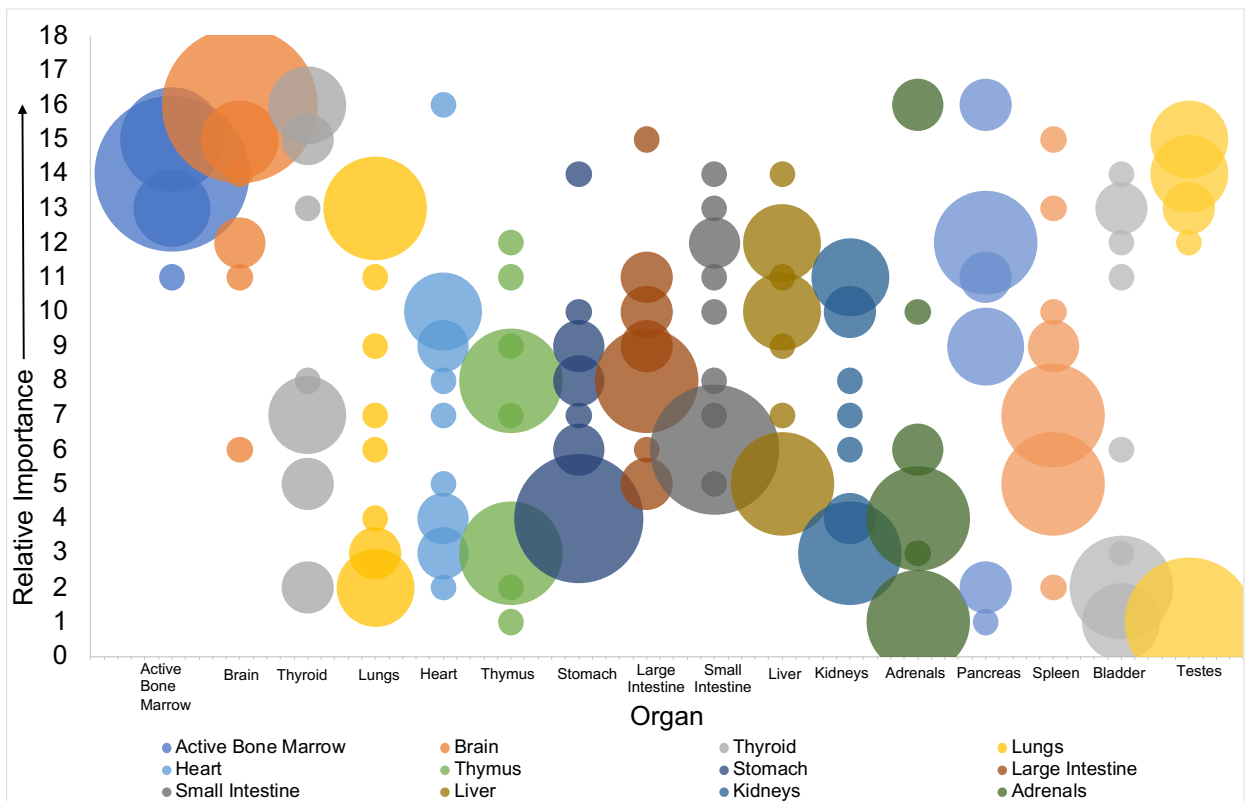


Figure 105: Organ Dose Equivalents, Frequency of Inverse Rank (Max, Male)

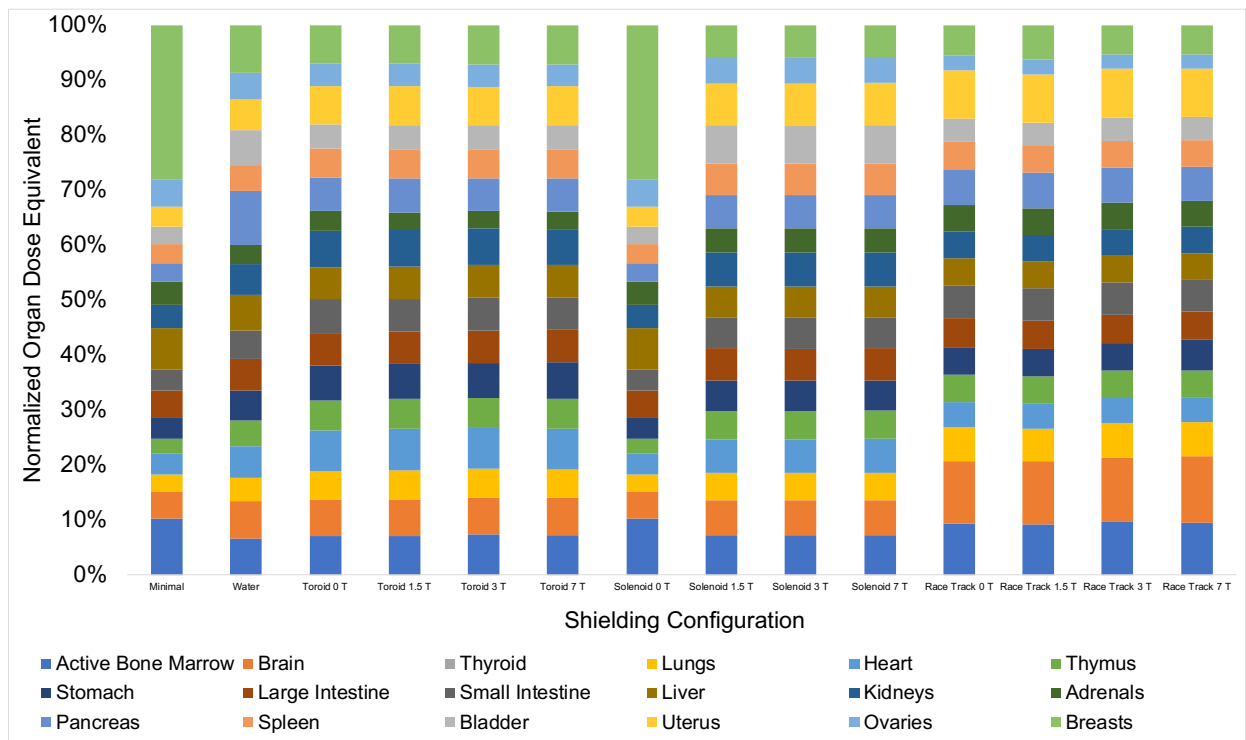


Figure 106: Organ Dose Equivalent, Normalized (Min, Female)

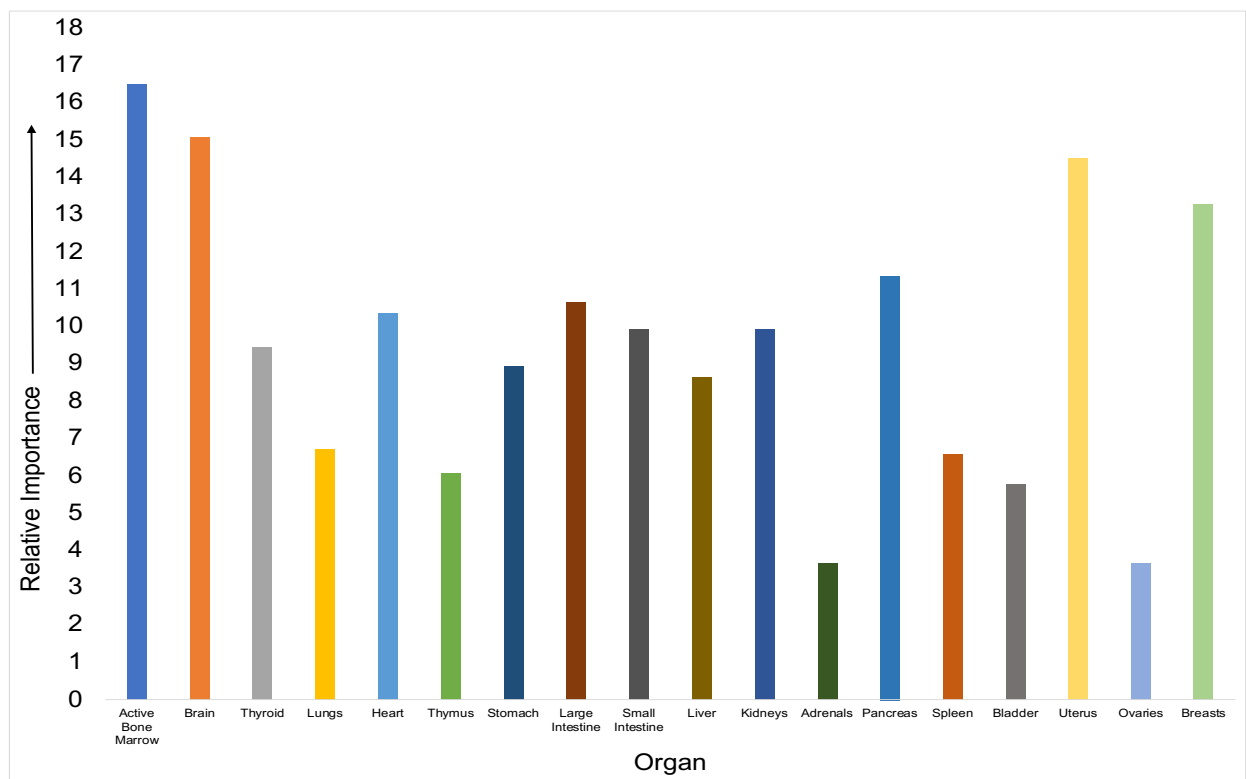


Figure 107: Organ Dose Equivalent, Average Inverse Rank (Min, Female)

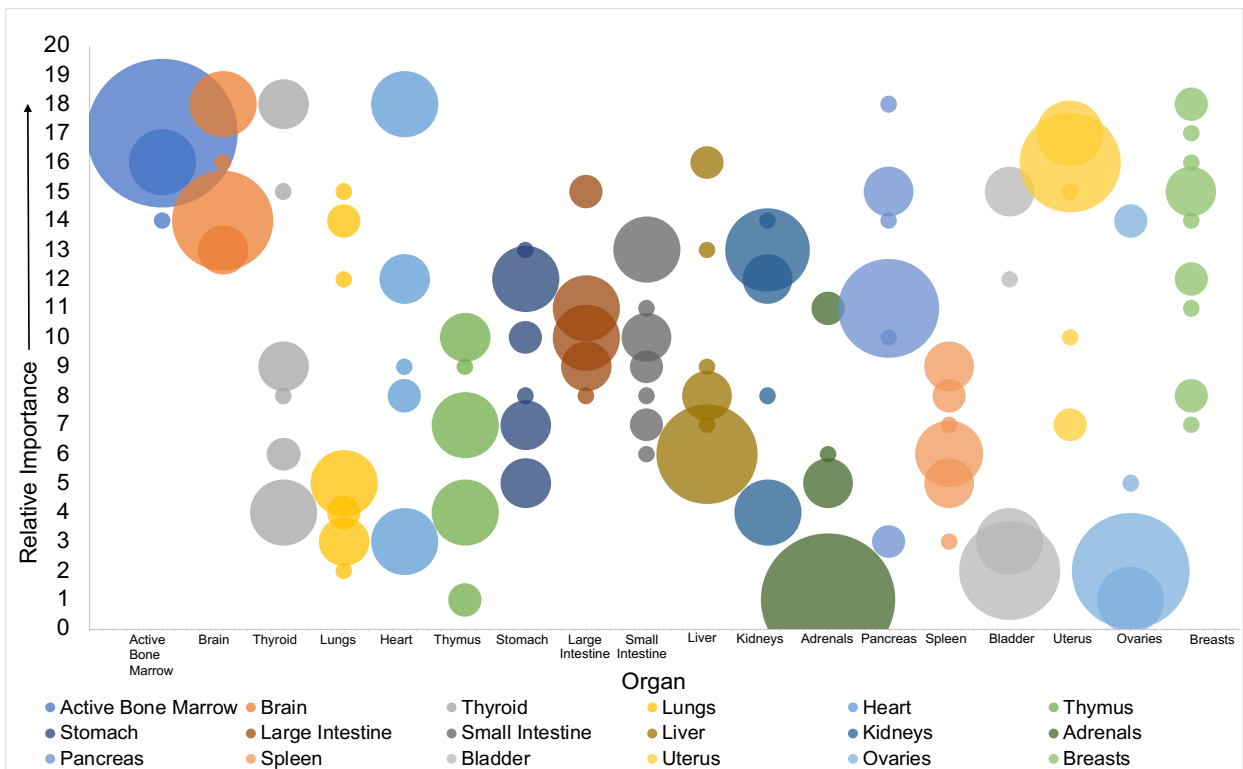


Figure 108: Organ Dose Equivalent, Frequency of Inverse Rank (Min, Female)

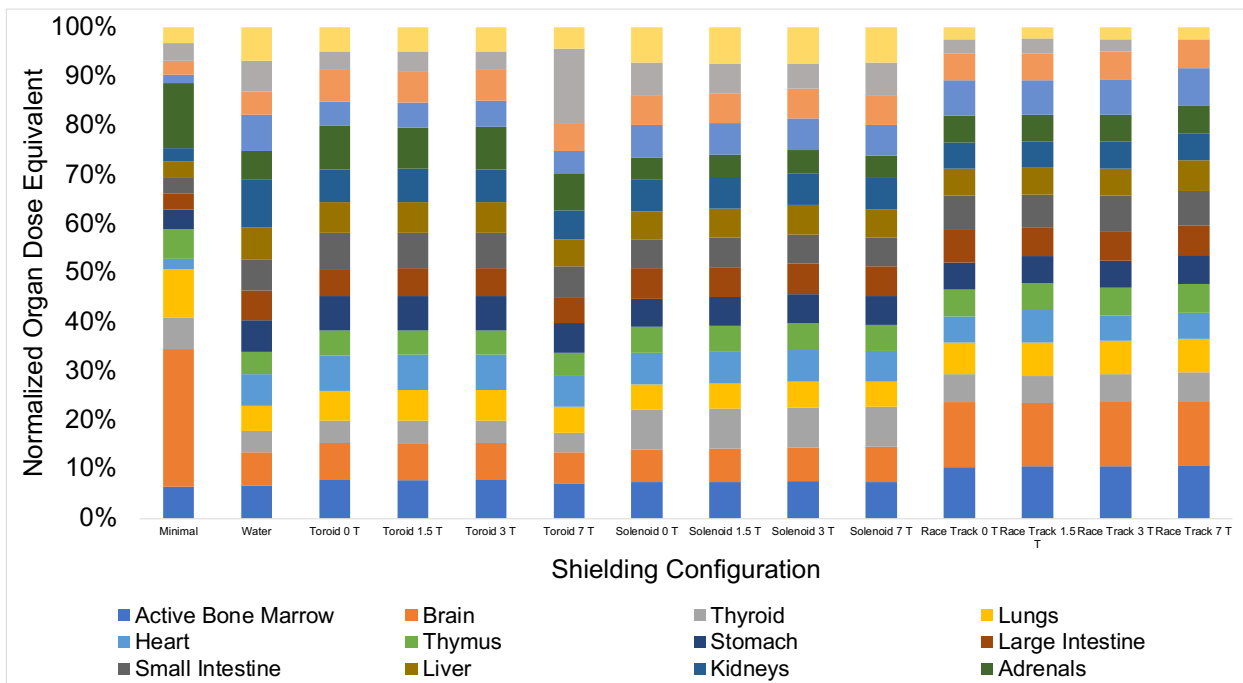


Figure 109: Organ Dose Equivalent, Normalized (Min, Male)

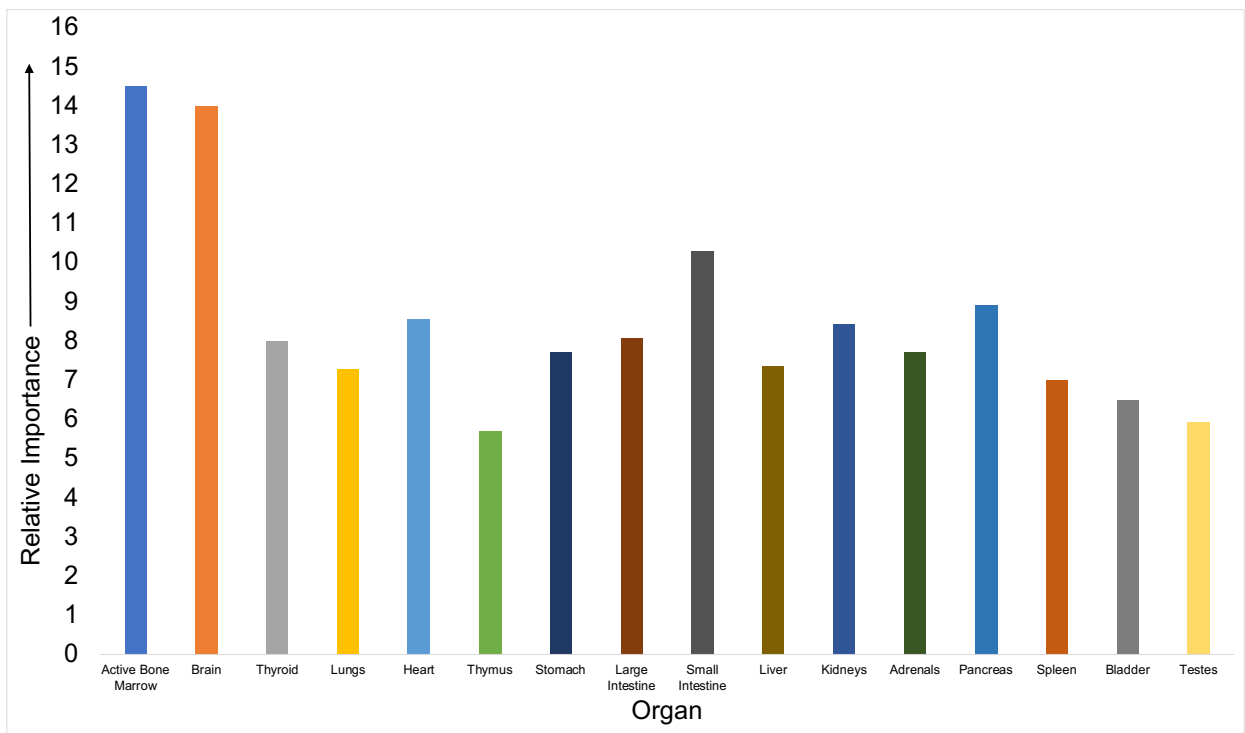


Figure 110: Organ Dose Equivalent, Average Rank (Min, Male)

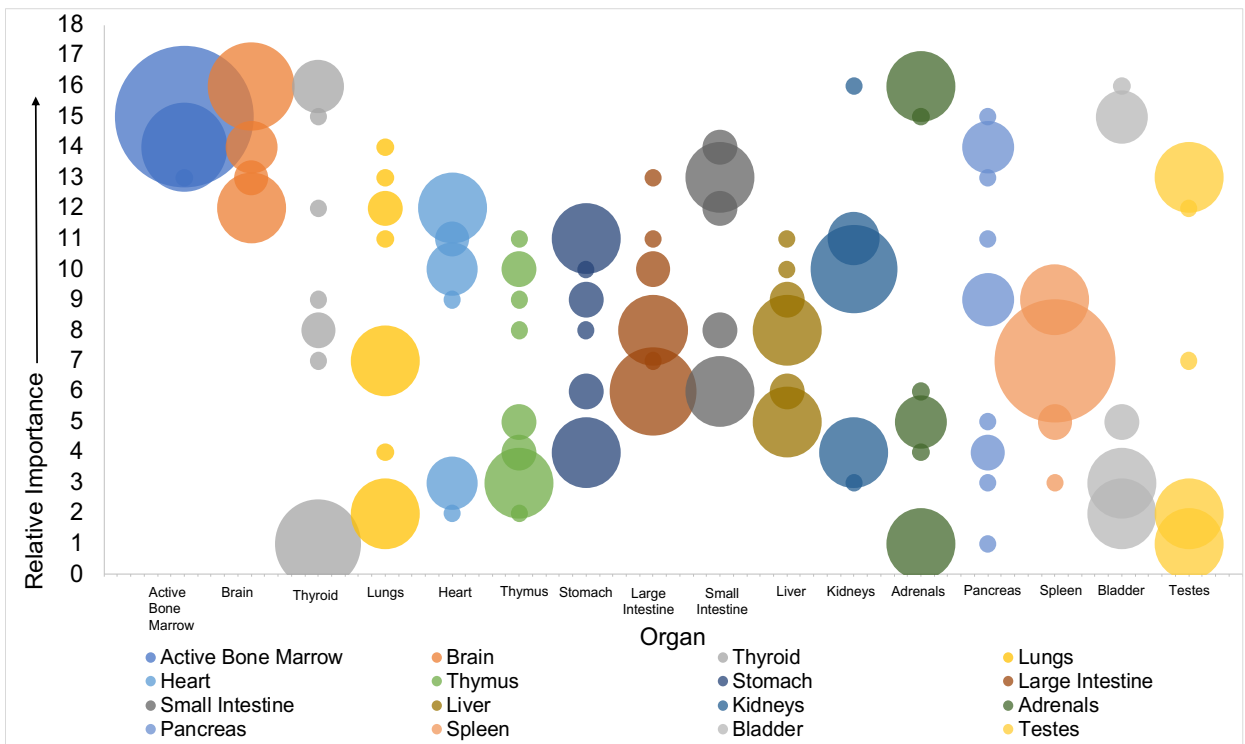


Figure 111: Organ Dose Equivalents, Frequency of Rank (Min, Male)

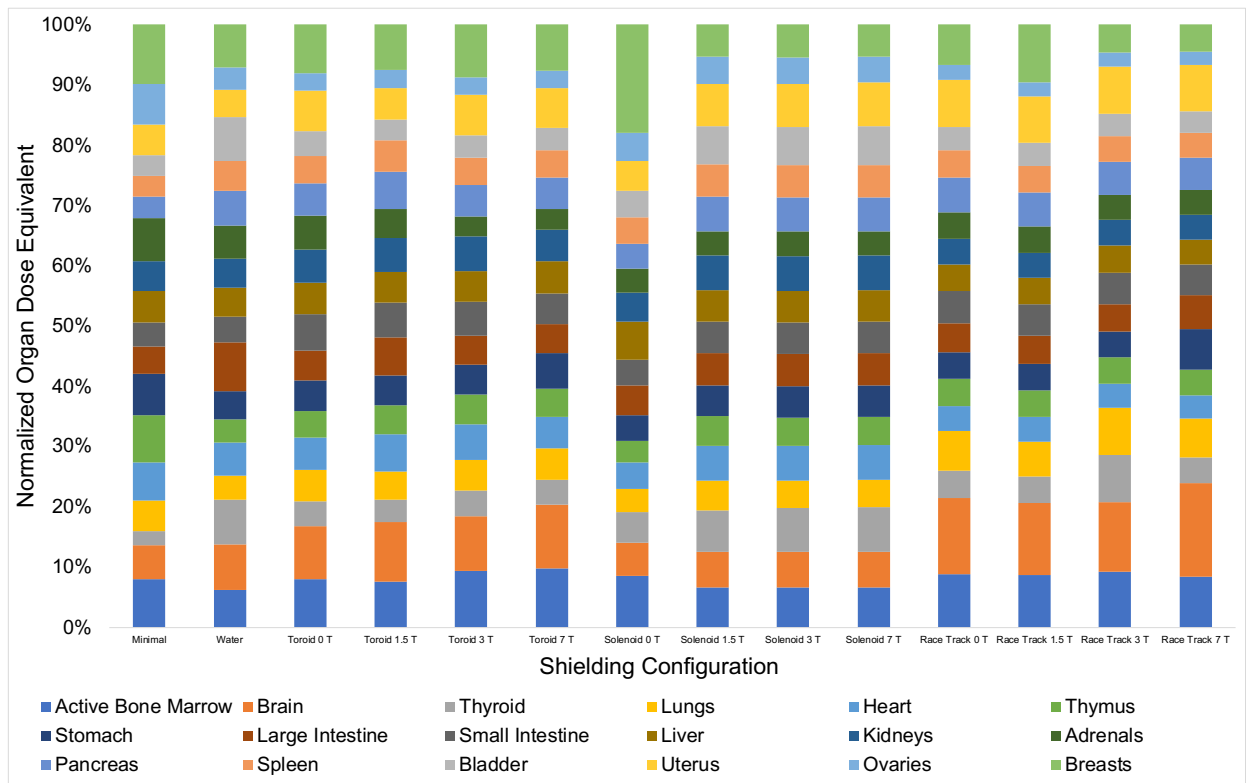


Figure 112: Organ Dose Equivalent, Normalized (Average, Female)

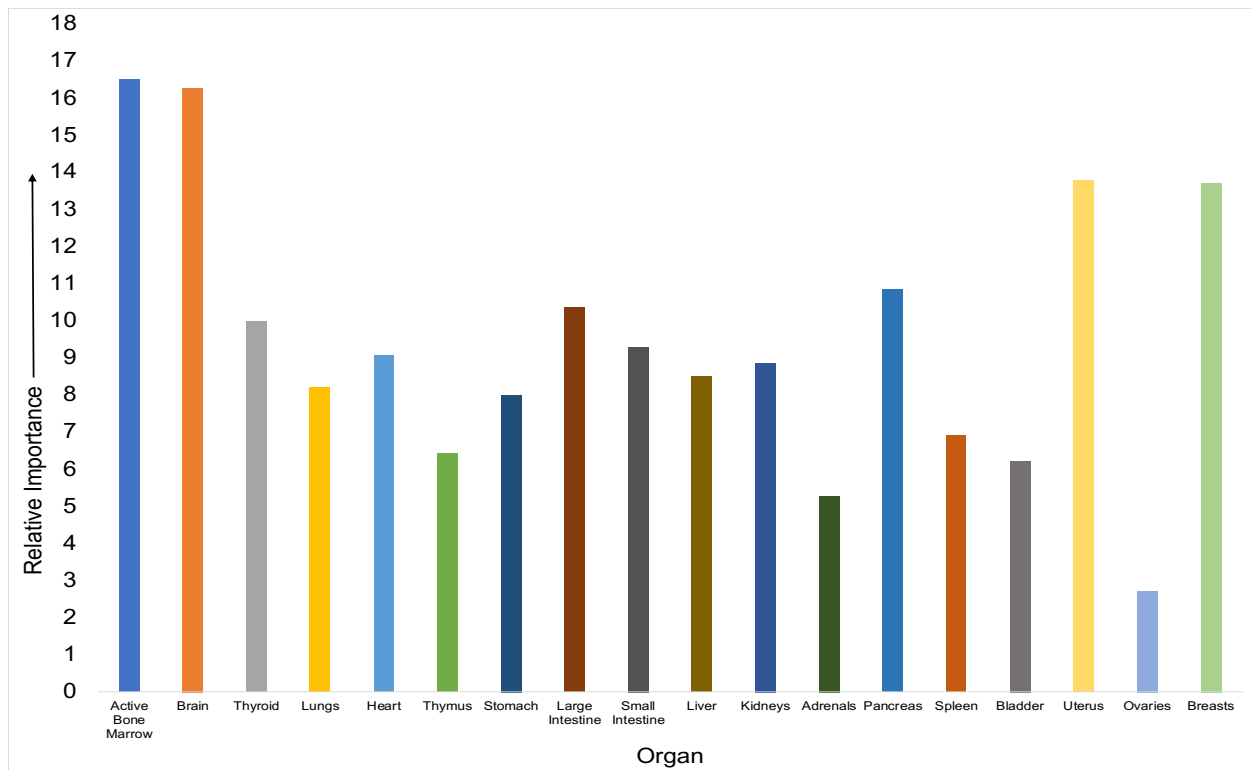
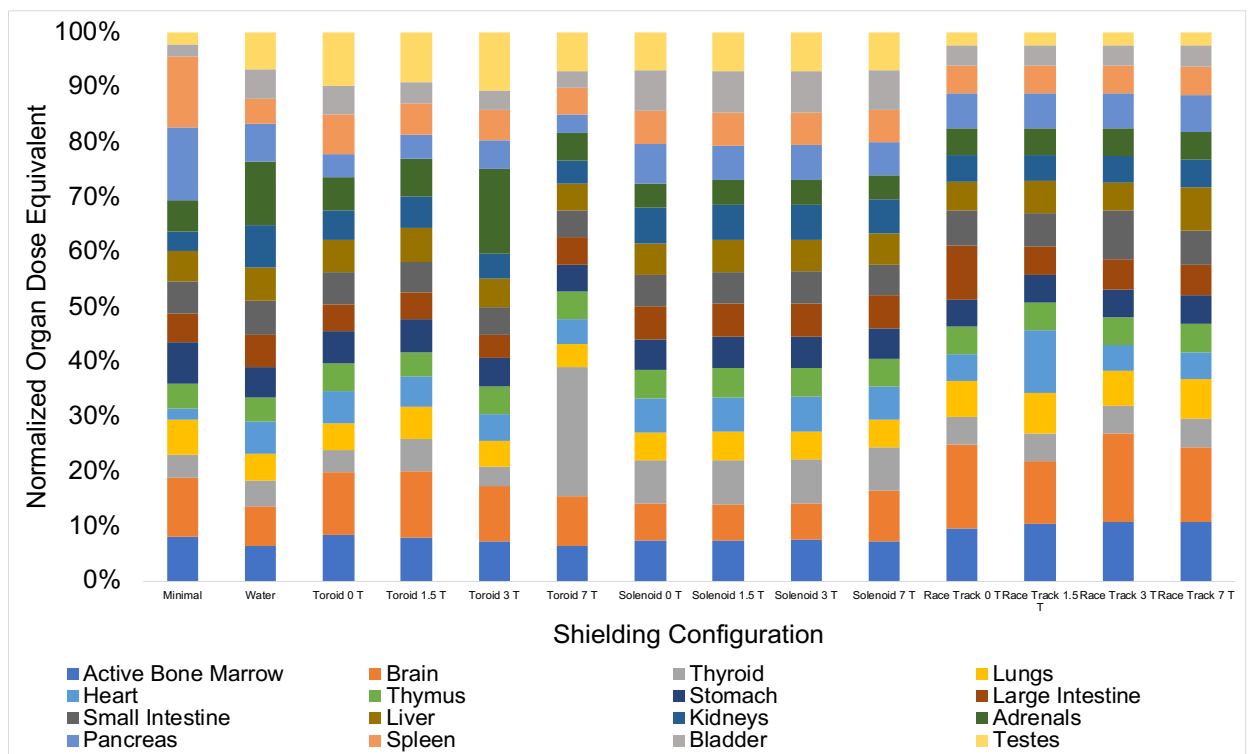
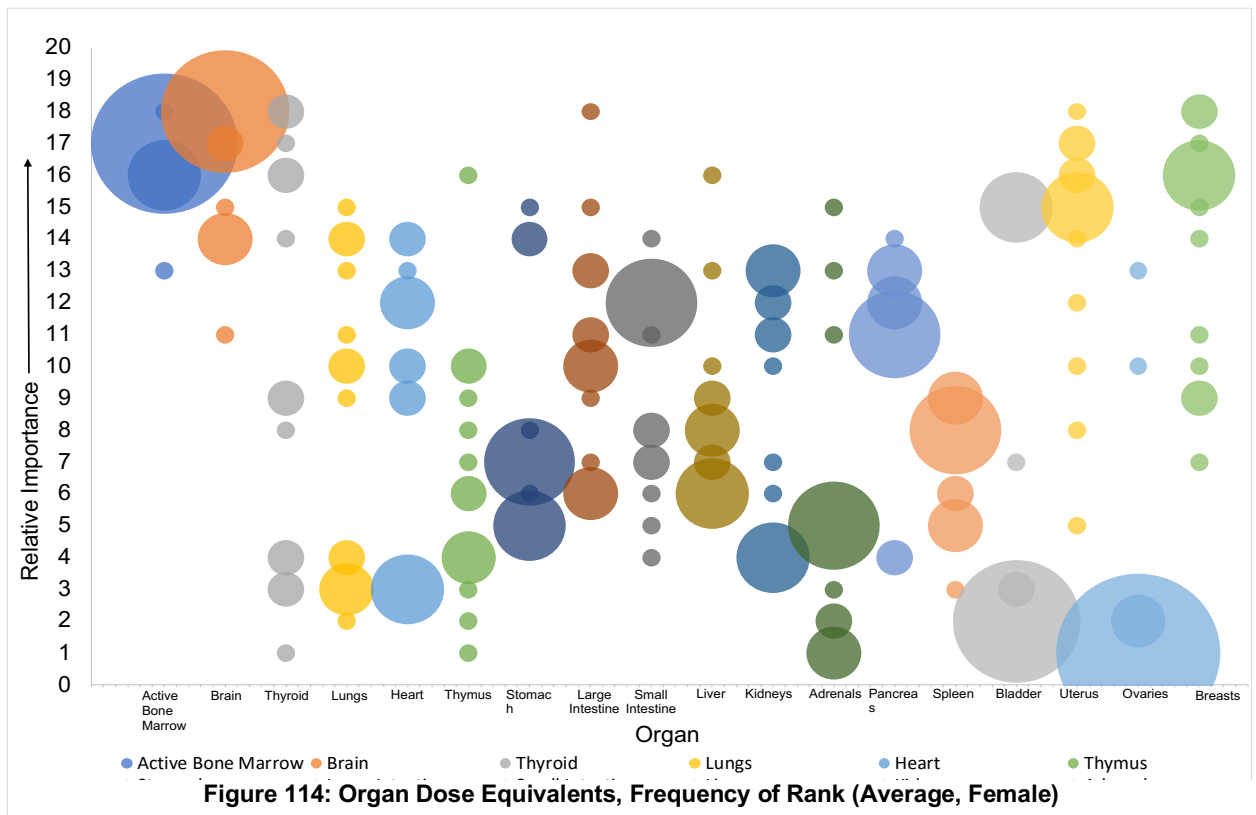


Figure 113: Organ Dose Equivalent, Average Rank (Average, Female)



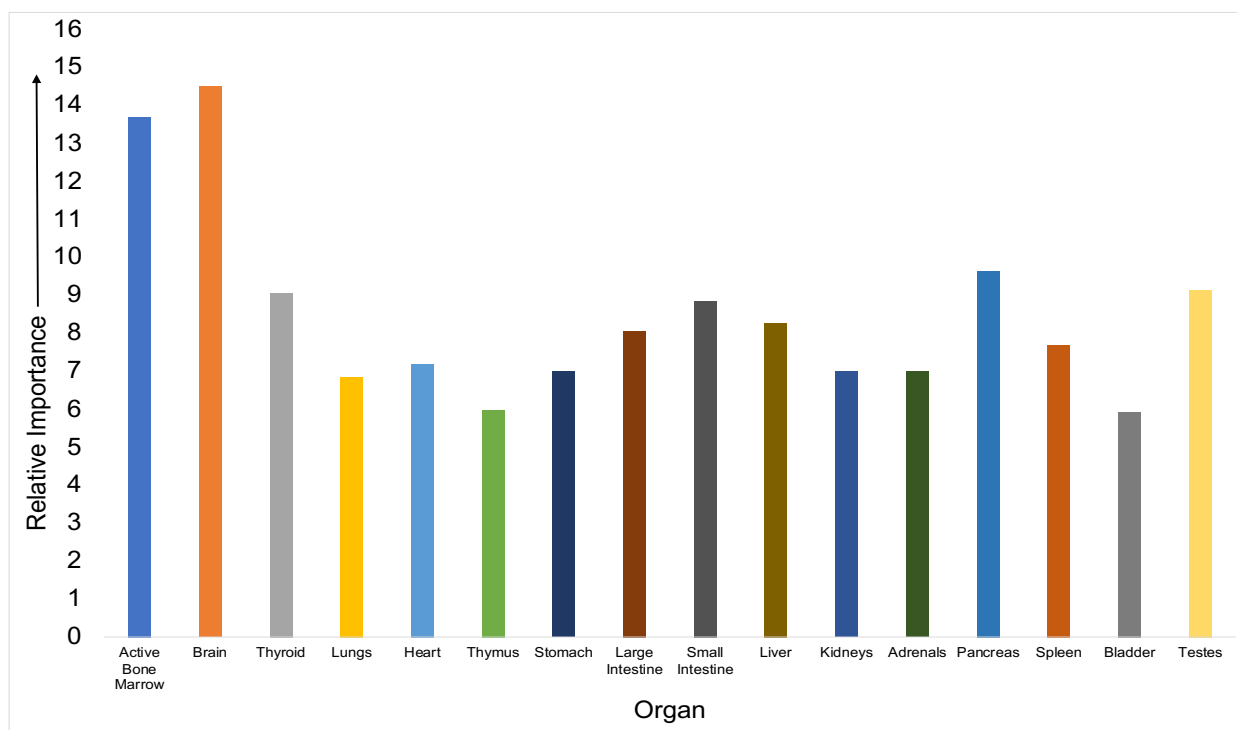


Figure 116: Organ Dose Equivalent, Average Rank (Average, Male)

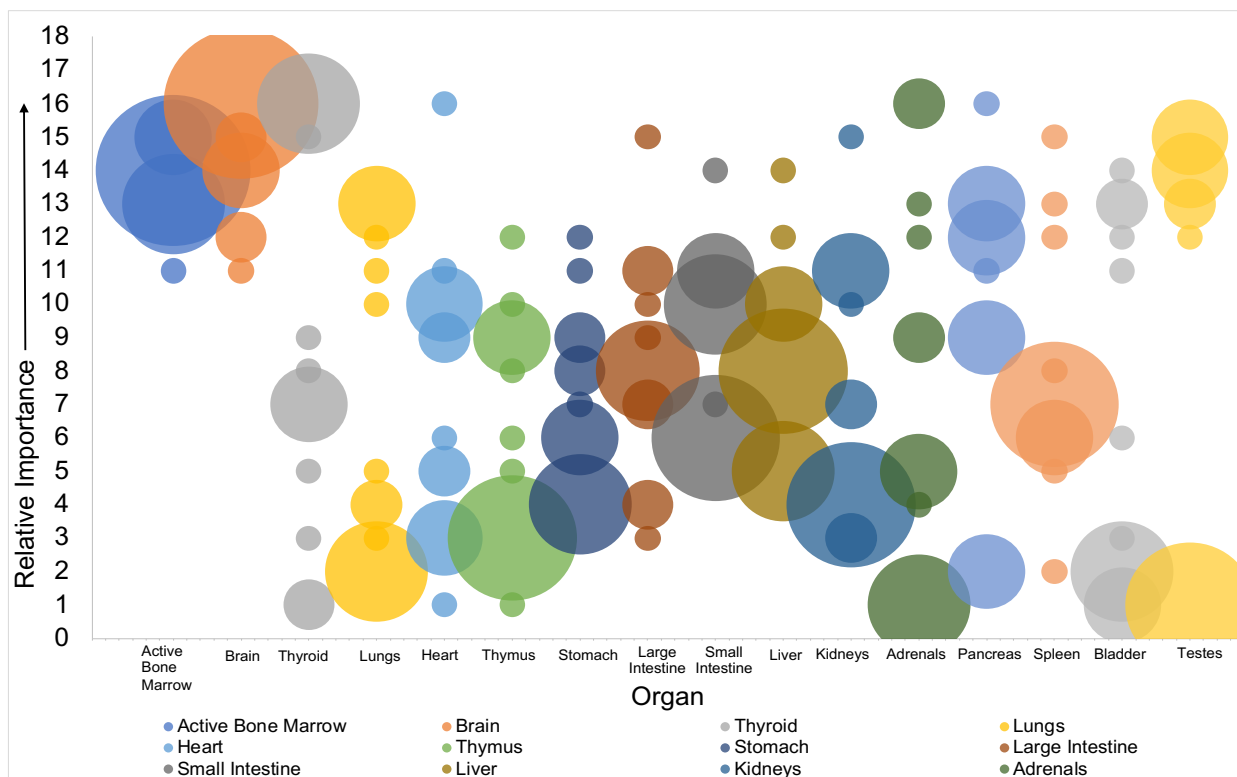


Figure 117: Organ Dose Equivalents, Frequency of Rank (Average, Male)

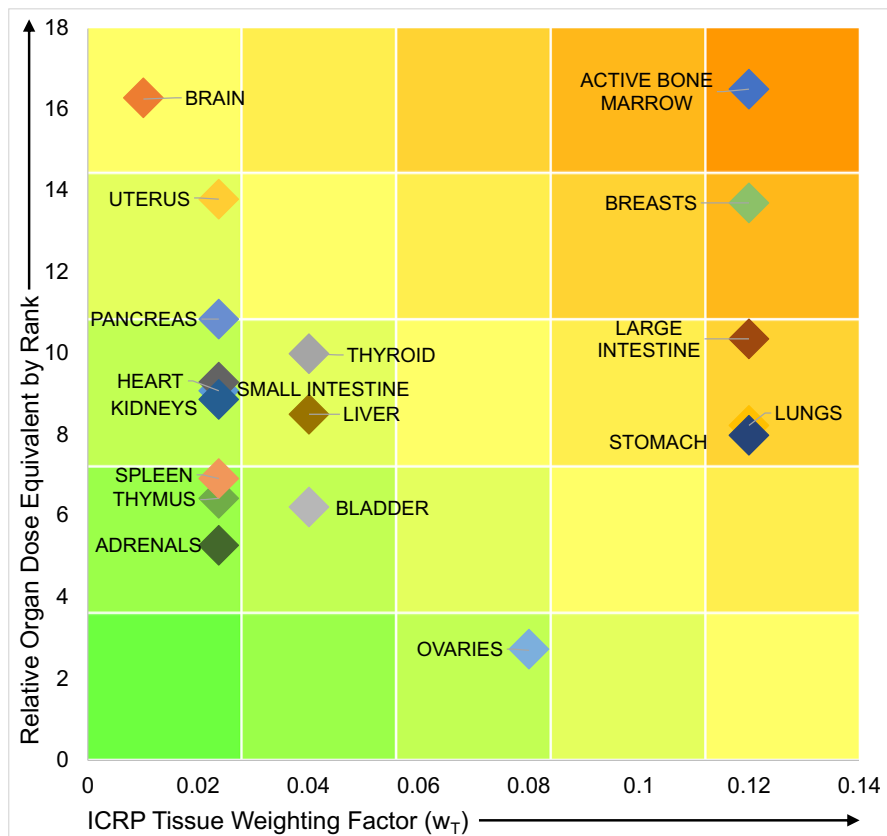
From these figures, it is notable that active bone marrow and brain have a consistently higher average rank (translating to a higher relative organ dose equivalent) across the board. This finding is confirmed with the third plot for each case, a bubble plot ranking the organ doses frequency of inverse rank across each shielding configuration. The larger a bubble on this plot, the more frequently the organ was assigned that rank for a given shielding configuration. Again, we see that active bone marrow and brain have consistently higher ranks (translating to a higher relative organ dose equivalent) across the board. While this is an interesting finding with many possible radiation protection applications (e.g. designing head-specific shielding for the astronauts to provide extra protection for the brain), further investigation involving a complete analysis of uncertainties for the organ dose equivalents will determine whether or not these differences are statistically significant.

Our team was also interested in determining the relative risk to each organ based on the relative dose equivalent paired with the ICRP tissue weighting factors (ICRP 2007). Our male and female modified MIRD phantoms have different remainder organs than the 14 identified by ICRP. In calculating overall effective dose, we followed the ICRP recommendation to take the mean equivalent dose to all the remainder organs and apply the overall weighting factor of 0.12. It is difficult, however, to assign individual tissue weighting factors to organs in the remainder with this method. Thus, to estimate the tissue weighting factors for our remainder organs for this portion of the analysis, we used a prior ICRP method of “splitting” the weighting factor across each of the organs in the remainder (ICRP 1991, 2007).

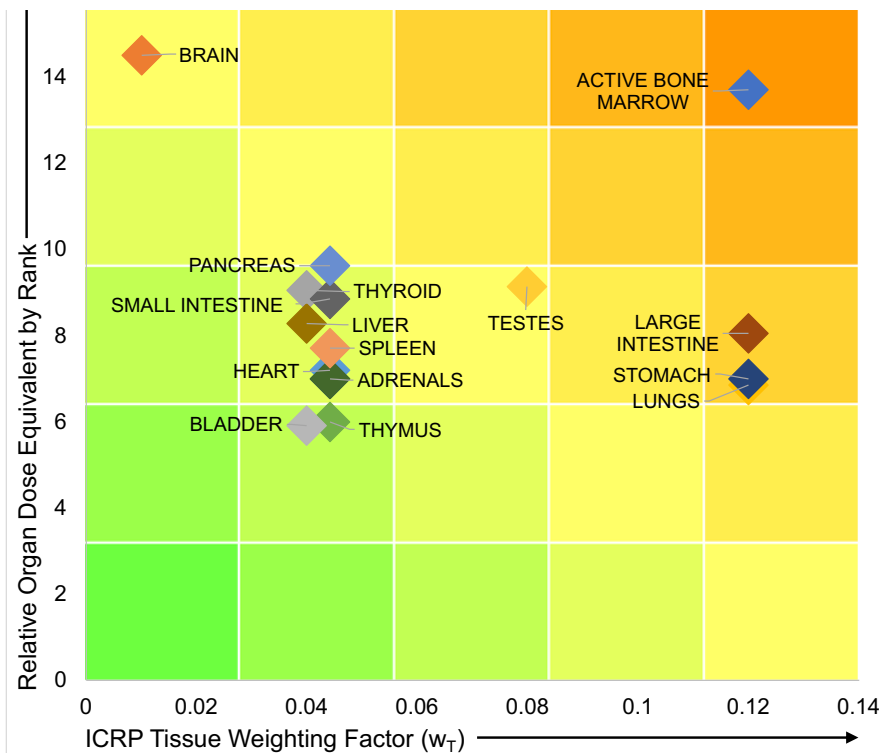
To display these data, we created traditional 5 x 5 risk matrices plotting the relative dose equivalent by inverse rank versus the tissue weighting factor. For organs appearing in the remainder, we assigned an equal portion of the remainder weighting factor to those organs. The results of this analysis are provided in Table 25 and depicted in Figure 118 and Figure 119.

**Table 25: Organ Dose Ranks and Tissue Weighting Factors**

Organ	FEMALE		MALE	
	Average Inverse Rank of Organ Dose Equivalent	Tissue Weighting Factor	Average Inverse Rank of Organ Dose Equivalent	Tissue Weighting Factor
Active Bone Marrow	16.50	0.120	13.71	0.120
Brain	16.29	0.010	14.50	0.010
Thyroid	10.00	0.040	9.07	0.040
Lungs	8.21	0.120	6.86	0.120
Heart	9.07	0.024	7.21	0.044
Thymus	6.43	0.024	6.00	0.044
Stomach	8.00	0.120	7.00	0.120
Large Intestine	10.36	0.120	8.07	0.120
Small Intestine	9.29	0.024	8.86	0.044
Liver	8.50	0.040	8.29	0.040
Kidneys	8.86	0.024	7.00	0.044
Adrenals	5.29	0.024	7.00	0.044
Pancreas	10.86	0.024	9.64	0.044
Spleen	6.93	0.024	7.71	0.044
Bladder	6.21	0.040	5.93	0.040
Uterus	13.79	0.024	-	-
Ovaries	2.71	0.080	-	-
Breasts	13.71	0.120	-	-
Testes	-	-	9.14	0.080



**Figure 118: Matrix of Organs at Risk (Female)**  
Tissue weighting factors for remainder organs per Table 25



**Figure 119: Matrix of Organs at Risk (Male)**  
Tissue weighting factors for remainder organs per Table 25

From these figures we again see that the brain and active bone marrow have the highest relative dose equivalent by rank. However, with the tissue weighting factors applied, we observe that the overall risk associated with the brain dose is moderate. It is important to note that the latest research into the biological effects of space radiation indicate that track effects in the brain from GCR particles are particularly damaging (Parihar et al. 2015; Nelson et al. 2016). While NASA's Space Cancer Risk Model (NSCR) in development is working to update radiation quality factors to incorporate additional effects of GCRs (Cucinotta et al. 2013; NRC 2012), it is important to note that many of the biological effects to the central nervous system are noncancerous. The current standard ICRP tissue weighting factors focus the risk of radiation-induced cancer and do not take these track effects into account (ICRP 2007).

For both male and female astronauts, we observe that active bone marrow is the organ at highest risk in this analysis due to its high relative organ dose equivalent by rank and its high tissue weighting factor. For the female astronauts, we observe that breast is also at relatively high risk in this analysis due to its moderately high relative organ dose equivalent by rank and its high tissue weighting factor. With this information, space agencies can begin to focus on specific organs at highest risk for radiation-induced cancer.

### *3.10 Sensitivity Analysis*

In this project, several variables influence the effective dose calculation including the solar cycle, shielding type, magnetic field, and astronaut sex. Astronaut age is also a highly influential variable, but it only affects the limits applied once effective dose is calculated. In a multivariable scenario such as this, it can be difficult to determine the direct effect of each individual variable. One solution is to conduct a sensitivity analysis on all the variables. This involves holding all variables constant except the variable being measured and recording the effect to the final calculation as the variable is modified. We conducted the sensitivity analysis on each of these

variables manually in a spreadsheet, though there are several commercially available applications that assist with multivariate sensitivity analysis. Figure 120, Figure 121, and Figure 122 display these data showing the relative impact of each variable, while Figure 121, Figure 123, and Figure 125 display these data in the form of radar charts. Radar charts show the relative contribution of several variables on one plot, which is useful to determine the overall sensitivity to each variable. Table 26 shows the calculations measuring the effect of each variable in the scenario.

### *3.10. SPE Effective Dose*

Figure 120 shows that changing between female and male astronauts has a less than 10% effect on the SPE effective dose. This figure also shows that the other variables (shielding type, magnetic field, and solar cycle) all have a large impact on the SPE effective dose. However, the data from the various magnetic fields and shielding types is clustered indicating that any of the shielding configurations or magnetic fields analyzed in this project produced a similar, significant decrease in SPE effective dose. In the radar chart in Figure 121 we again see that the selection of any shielding configuration and magnetic field will produce a significant decrease in SPE effective dose. The figure also confirms that astronaut sex has a small impact on the SPE effective dose.

### *3.10. GCR Effective Dose*

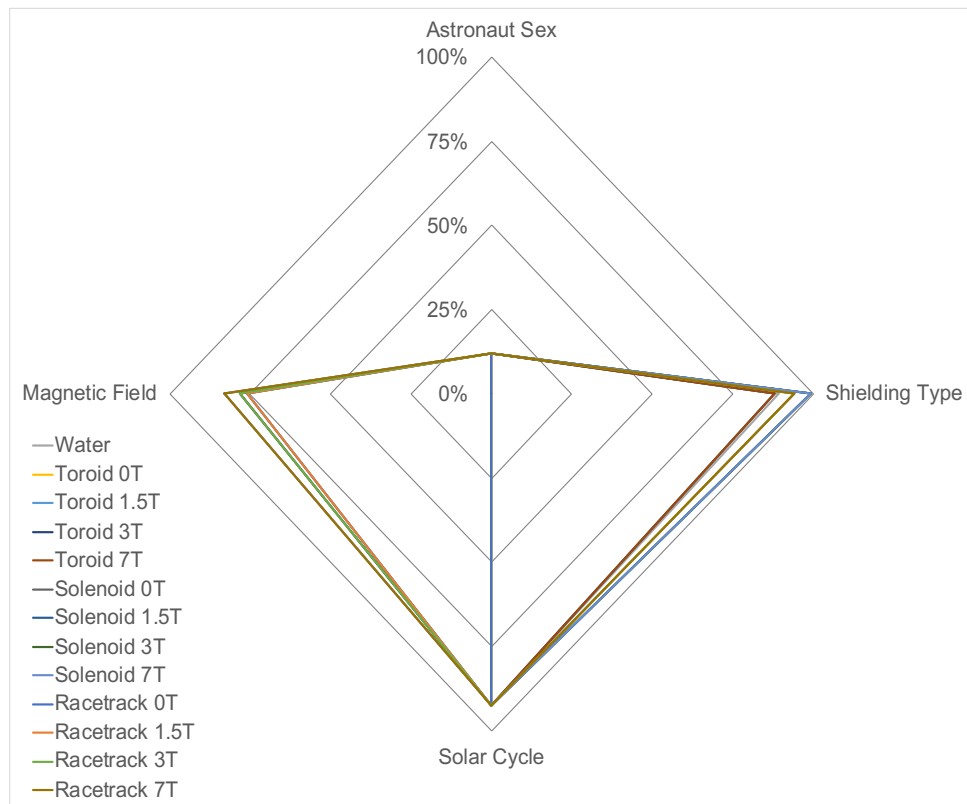
Figure 122 shows that the only variables that have a large impact on the GCR effective dose are shielding type and magnetic field. Varying astronaut sex or solar cycle has a less than 10% effect on the GCR effective dose. In the radar chart in Figure 123 we again see that the selection of shielding type and magnetic field has the largest impact on the GCR effective dose. The figure also confirms that astronaut sex and solar cycle have a small impact on the GCR effective dose.

### 3.10. Total Effective Dose

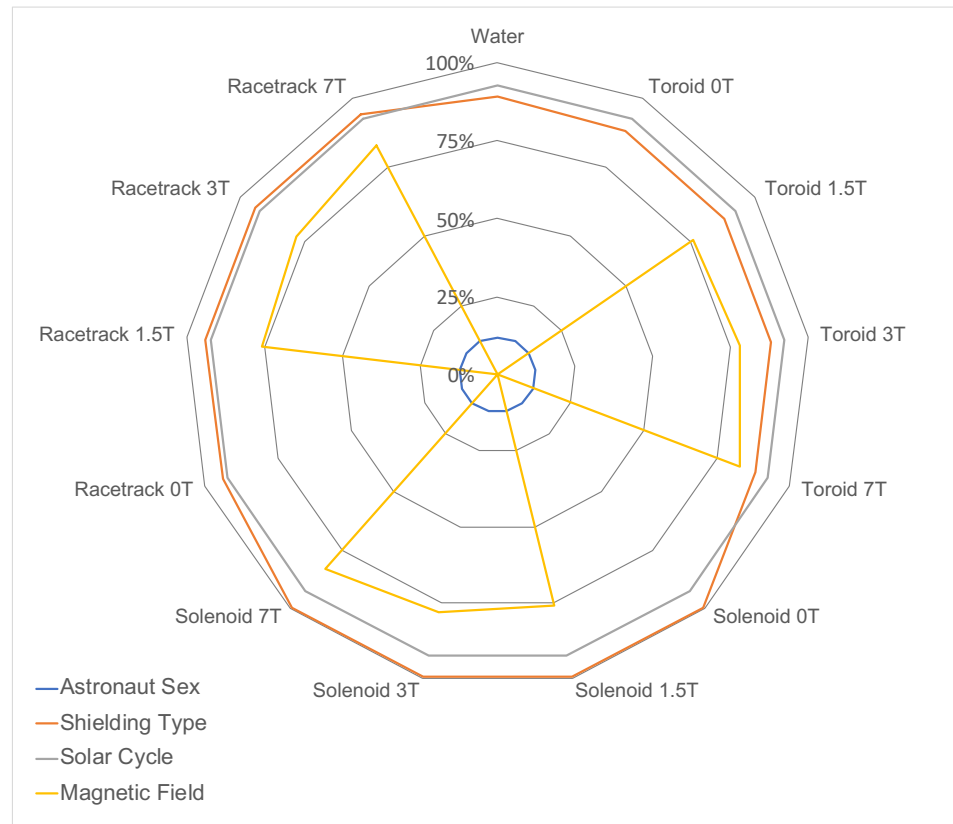
Figure 124 shows that total effective dose for the mission depends most strongly on shielding type and magnetic field. This is similar to the sensitivity of GCR effective dose, because GCR effective dose is the dominant component of total effective dose. In the radar chart in Figure 125 we again see that the selection of shielding type and magnetic field has the largest impact on total effective dose. The figure also confirms once again that astronaut sex and solar cycle have a small impact on the total effective dose.

**Table 26: Sensitivity Analysis Calculations**

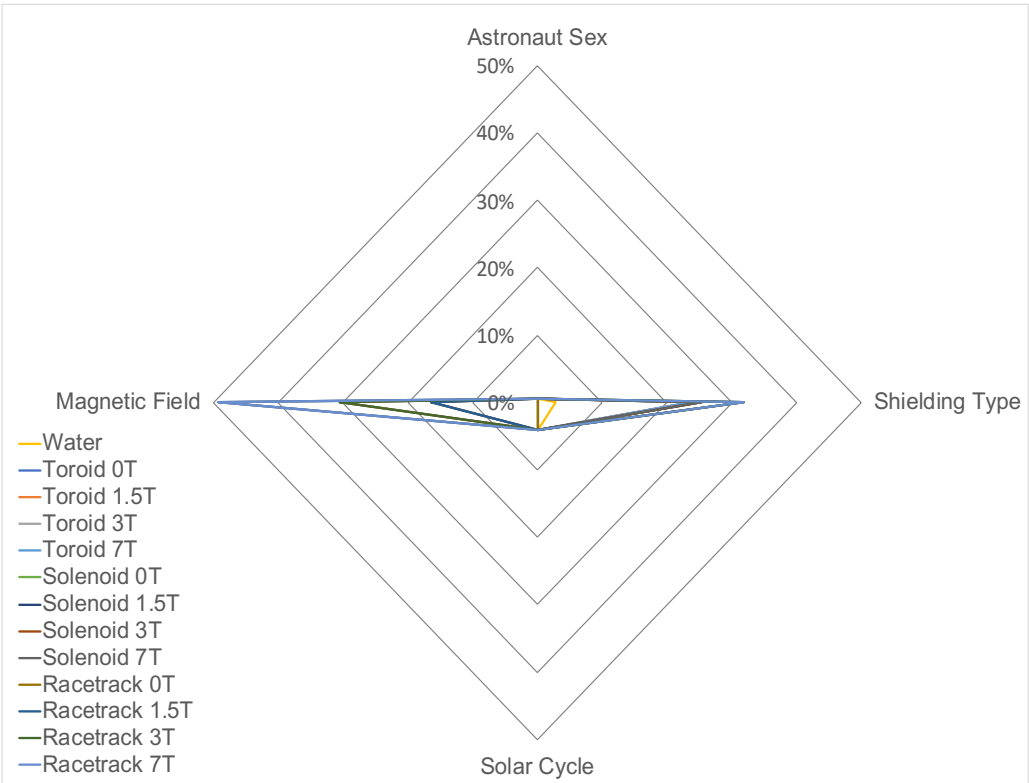
<b>Total Mean Eff Dose (Sv)</b> 2.0068					
<b>Variable</b>	<b>Value 1</b>	<b>Value 2</b>	<b>Value 3</b>	<b>Value 4</b>	<b>Value 5</b>
Solar Cycle	Max	Min			
Shielding Type	Aluminum	Water	Toroid	Solenoid	Racetrack
B Field	0	1.5	3	7	
Crew Gender	Female	Male			
<b>Solar Cycle</b> <b>Total Mean Eff Dose (Sv)</b> Max Min 2.08023571 1.93338571					
Changing the solar cycle variable (max to min) changes the total mean effective dose by:					
					-7.06%
<b>Solar Cycle</b> <b>SPE Mean Eff Dose (Sv)</b> Max Min 0.24083214 0.01792143					
Changing the solar cycle variable (max to min) changes the SPE mean effective dose by:					
					-92.56%
<b>Solar Cycle</b> <b>GCR Mean Eff Dose (Sv)</b> Max Min 1.81038929 1.88290714					
Changing the solar cycle variable (max to min) changes the GCR mean effective dose by:					
					4.01%
<b>Shielding Type</b> <b>Total Mean Eff Dose (Sv)</b> Aluminum Water Toroid Solenoid Racetrack 3.489725 2.458225 2.02150625 1.8149625 1.70038125					
Changing the shielding config variable (Al to water) changes the total mean effective dose by:					
					-29.56%
Changing the shielding config variable (Al to Toroid) changes the total mean effective dose by:					
					-42.07%
Changing the shielding config variable (Al to Solenoid) changes the total mean effective dose by:					
					-47.99%
Changing the shielding config variable (Al to Racetrack) changes the total mean effective dose by:					
					-51.27%
<b>Shielding Type</b> <b>SPE Mean Eff Dose (Sv)</b> Aluminum Water Toroid Solenoid Racetrack 0.98105 0.105975 0.11694375 0.00564375 0.058475					
Changing the shielding config variable (Al to water) changes the SPE mean effective dose by:					
					-89.20%
Changing the shielding config variable (Al to Toroid) changes the SPE mean effective dose by:					
					-88.08%
Changing the shielding config variable (Al to Solenoid) changes the SPE mean effective dose by:					
					-99.42%
Changing the shielding config variable (Al to Racetrack) changes the SPE mean effective dose by:					
					-94.04%
<b>Shielding Type</b> <b>GCR Mean Eff Dose (Sv)</b> Aluminum Water Toroid Solenoid Racetrack 2.410175 2.34295 1.83483125 1.79815625 1.642					
Changing the shielding config variable (Al to water) changes the GCR mean effective dose by:					
					-2.79%
Changing the shielding config variable (Al to Toroid) changes the GCR mean effective dose by:					
					-23.87%
Changing the shielding config variable (Al to Solenoid) changes the GCR mean effective dose by:					
					-25.39%
Changing the shielding config variable (Al to Racetrack) changes the GCR mean effective dose by:					
					-31.87%
<b>B field (T)</b> <b>Total Mean Eff Dose (Sv)</b> 0 1.5 3 7 2.528355 2.11220833 1.75681667 1.28216667					
Changing the B field variable (0 to 1.5 T) changes the total mean eff dose by:					
					-16.46%
Changing the B field variable (0 to 3 T) changes the total mean eff dose by:					
					-30.52%
Changing the B field variable (0 to 7 T) changes the total mean eff dose by:					
					-49.29%
<b>B field (T)</b> <b>SPE Mean Eff Dose (Sv)</b> 0 1.5 3 7 0.26302 0.06335 0.057275 0.04476667					
Changing the B field variable (0 to 1.5 T) changes the SPE mean eff dose by:					
					-75.91%
Changing the B field variable (0 to 3 T) changes the SPE mean eff dose by:					
					-78.22%
Changing the B field variable (0 to 7 T) changes the SPE mean eff dose by:					
					-82.98%
<b>B field (T)</b> <b>GCR Mean Eff Dose (Sv)</b> 0 1.5 3 7 2.38765 1.92613333 1.61679167 1.09535					
Changing the B field variable (0 to 1.5 T) changes the GCR mean eff dose by:					
					-19.33%
Changing the B field variable (0 to 3 T) changes the GCR mean eff dose by:					
					-32.29%
Changing the B field variable (0 to 7 T) changes the GCR mean eff dose by:					
					-54.12%
<b>Astronaut Sex</b> <b>Total Mean Eff Dose (Sv)</b> Female Male 2.03753214 1.97608929					
Changing the crew gender variable (female to male) changes the total mean effective dose by:					
					-3.02%
<b>Astronaut Sex</b> <b>SPE Mean Eff Dose (Sv)</b> Female Male 0.13768571 0.12106786					
Changing the crew gender variable (female to male) changes the SPE mean effective dose by:					
					-12.07%
<b>Astronaut Sex</b> <b>GCR Mean Eff Dose (Sv)</b> Female Male 1.85170714 1.84158929					
Changing the crew gender variable (female to male) changes the GCR mean effective dose by:					
					-0.55%



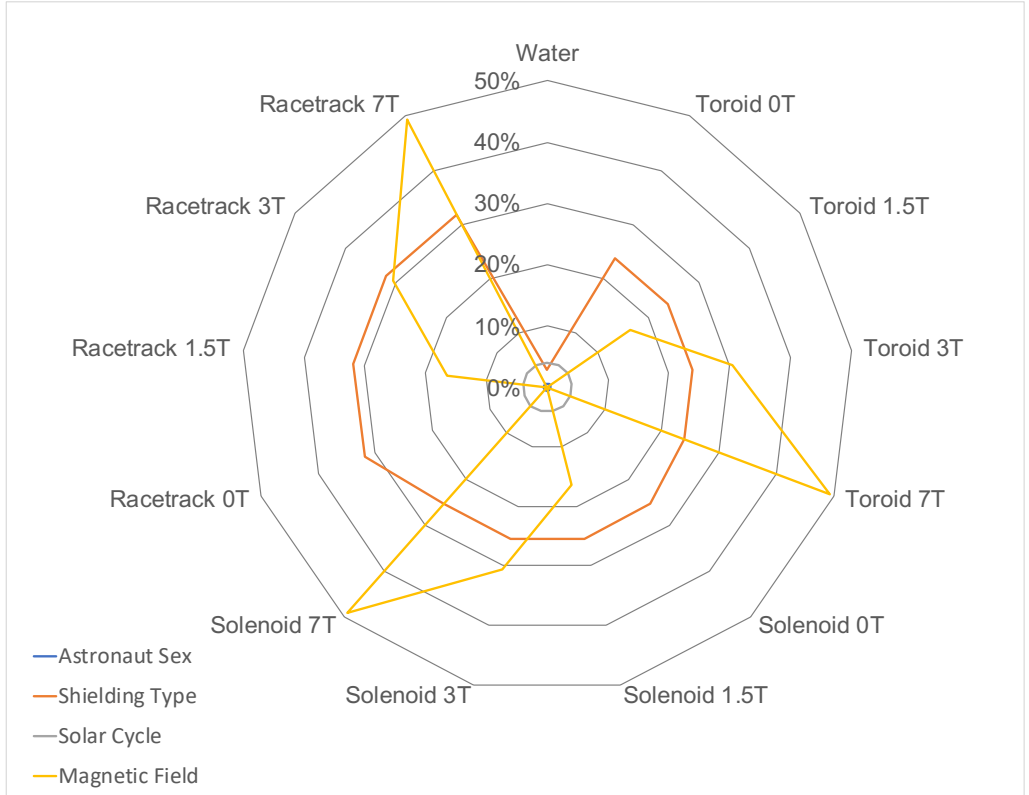
**Figure 120: Sensitivity Analysis for Total Mission (700 d) SPE Effective Dose**



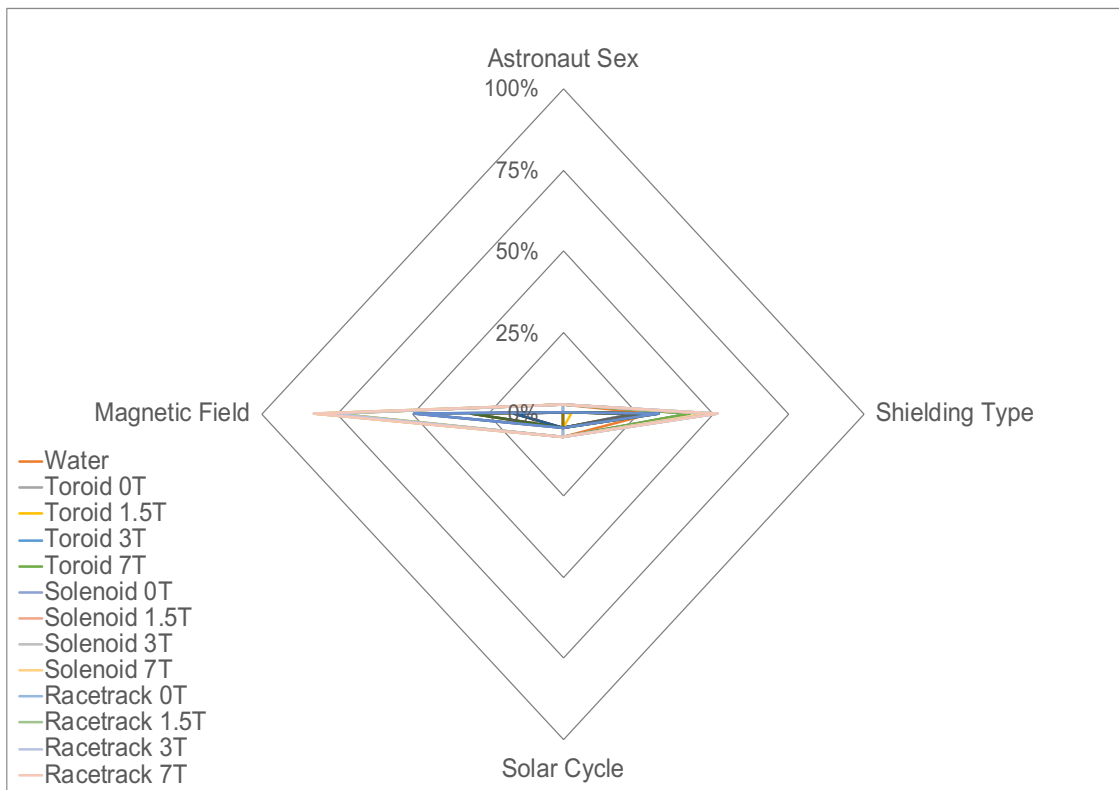
**Figure 121: Radar Chart for Total Mission (700 d) SPE Effective Dose**



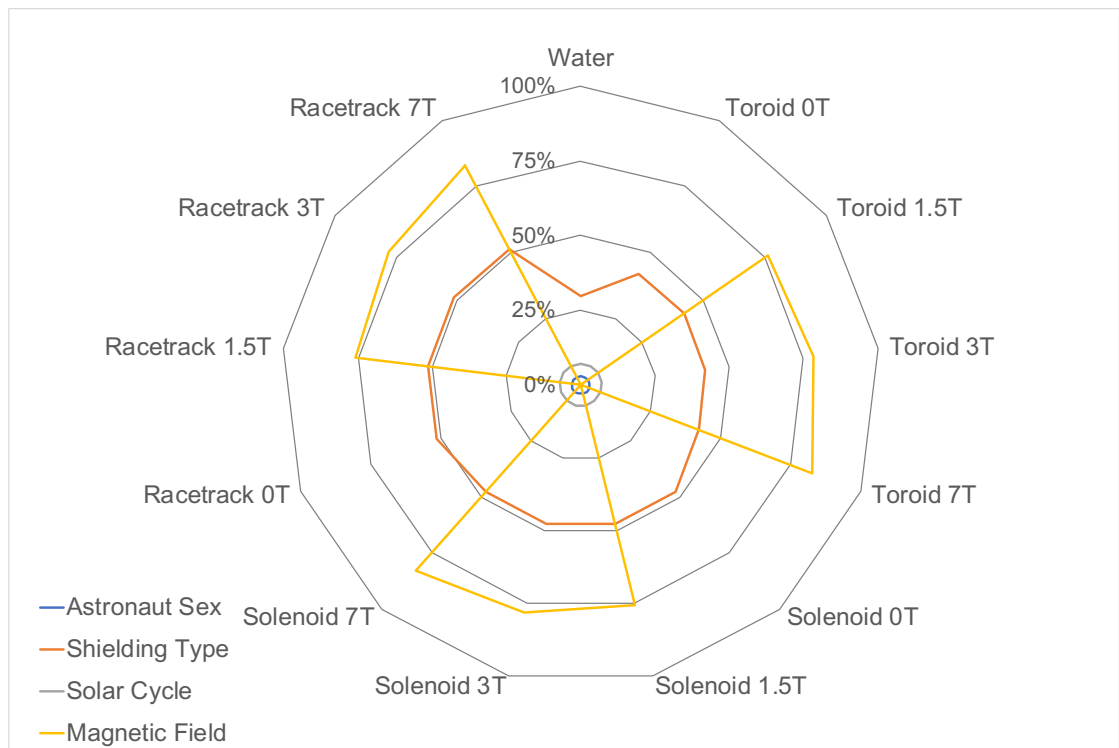
**Figure 122: Sensitivity Analysis for Total Mission (700 d) GCR Effective Dose**



**Figure 123: Radar Chart for Total Mission (700 d) GCR Effective Dose**



**Figure 124: Sensitivity Analysis for Total Mission (700 d) Total Effective Dose**



**Figure 125: Radar Chart for Total Mission (700 d) Total Effective Dose**

### 3.11 Cancer Risk and Safe Days in Space Analysis

Current guidance from NASA prescribes dose limits that vary based on astronaut age and sex (NASA 2015). In this study, the least of the annual dose limit extrapolated to mission duration (500 or 700 days) and the career dose limit were used. These limits represent a 3% risk of exposure induced death (REID) at a 95% confidence interval and were applied to the effective dose from the GEANT4 Monte Carlo simulations in this study. Linear interpolation was used to determine the corresponding REID of any effective dose value at a 95% confidence interval ( $+2\sigma$ ).

NASA has coined the term “safe days in space” (SDS) to refer to “the mission length where risk limits are not exceeded, for several mission scenarios at different acceptable levels of uncertainty” (Cucinotta et al. 2015). SDS is particularly useful in comparing several mission profiles to determine which are feasible under the current risk tolerances. REID and safe days in space, both at 95% CI, are plotted versus astronaut age for each shielding configuration and polynomial fit curves calculated for each.  $R^2$  values for all fit curves exceed 0.95. Coefficients for all fit curves for REID and SDS versus astronaut age are provided in Table 27 and Table 28.

For female astronauts on a 700 day mission, the REID versus crew age data (see Figure 126) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9990$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000006A^3 + 0.0009A^2 - 0.0498A + 0.9720$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0373A + 0.7280$$

Toroid 0 T

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0384A + 0.7493$$

Toroid 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0314A + 0.6128$$

Toroid 3 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0238A + 0.4659$$

Toroid 7 T

$$REID = -0.000002A^3 + 0.0004A^2 - 0.0186A + 0.3626$$

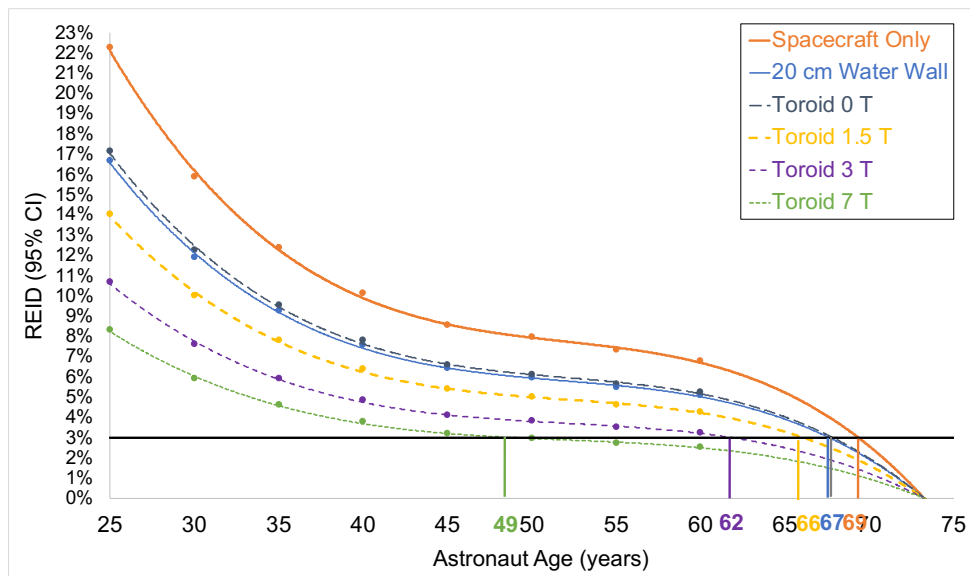


Figure 126: REID vs. Age (Female, 700 d, Toroid)

For female astronauts on a 700 day mission, the REID versus crew age data (see Figure 127) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9990$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000006A^3 + 0.0009A^2 - 0.0948A + 0.9720$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0373A + 0.7280$$

Solenoid 0 T

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0378A + 0.7382$$

Solenoid 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0298A + 0.5821$$

Solenoid 3 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0242A + 0.4737$$

Solenoid 7 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0132A + 0.2581$$

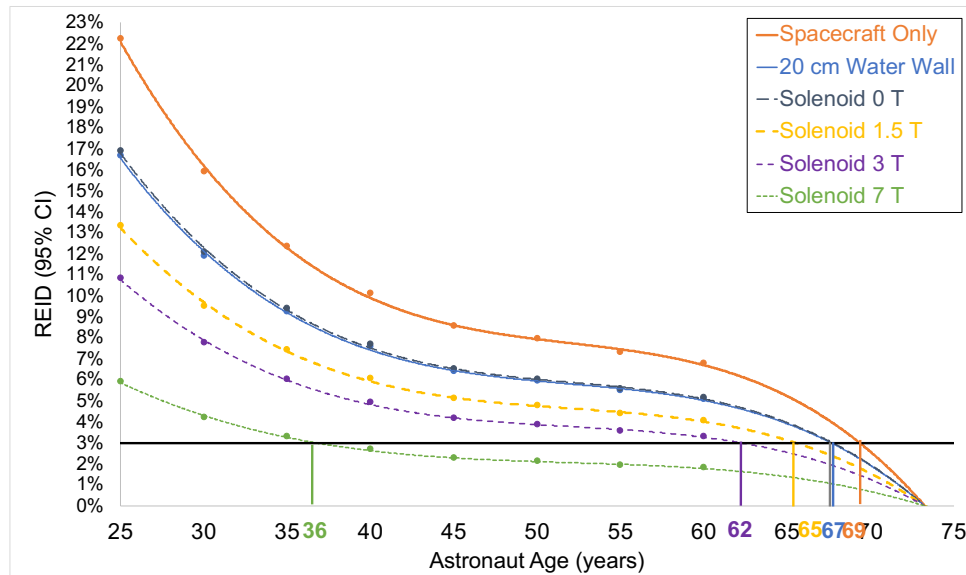


Figure 127: REID vs. Age (Female, 700 d, Solenoid)

For female astronauts on a 700 day mission, the REID versus crew age data (see Figure 128) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9990$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000006A^3 + 0.0009A^2 - 0.0948A + 0.9720$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0373A + 0.7280$$

Race Track 0 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0323A + 0.6318$$

Race Track 1.5 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0286A + 0.5587$$

Race Track 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0238A + 0.4648$$

Race Track 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0183A + 0.3569$$

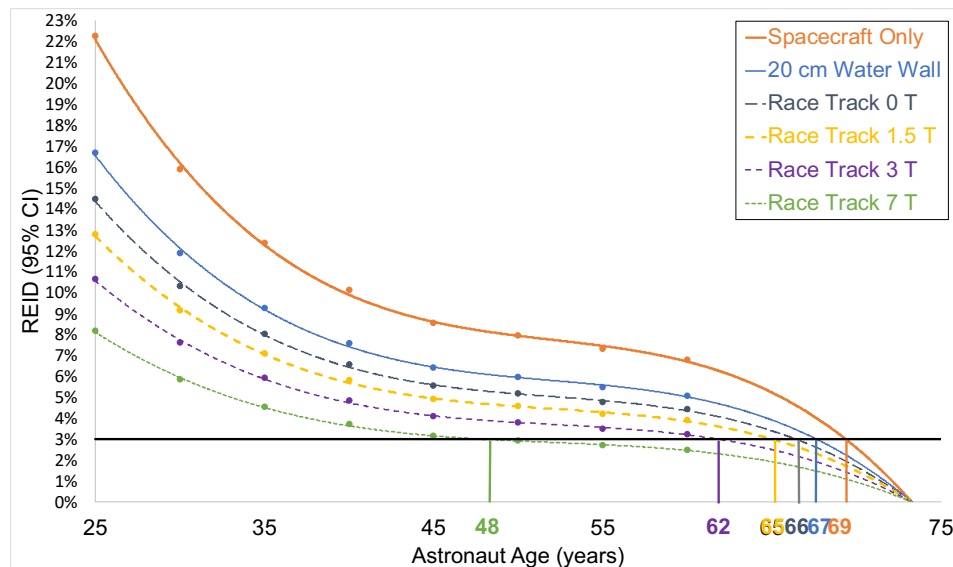


Figure 128: REID vs. Age (Female, 700 d, Race Track)

For female astronauts on a 700 day mission, the REID versus crew age data (see Figure 129) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (excluding zero magnetic field) best fit third-order polynomials ( $R^2 = 0.9990$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000006A^3 + 0.0009A^2 - 0.0498A + 0.9720$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0373A + 0.7280$$

Magnetic 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0299A + 0.5846$$

Magnetic 3 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0240A + 0.4682$$

Magnetic 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0167A + 0.3259$$

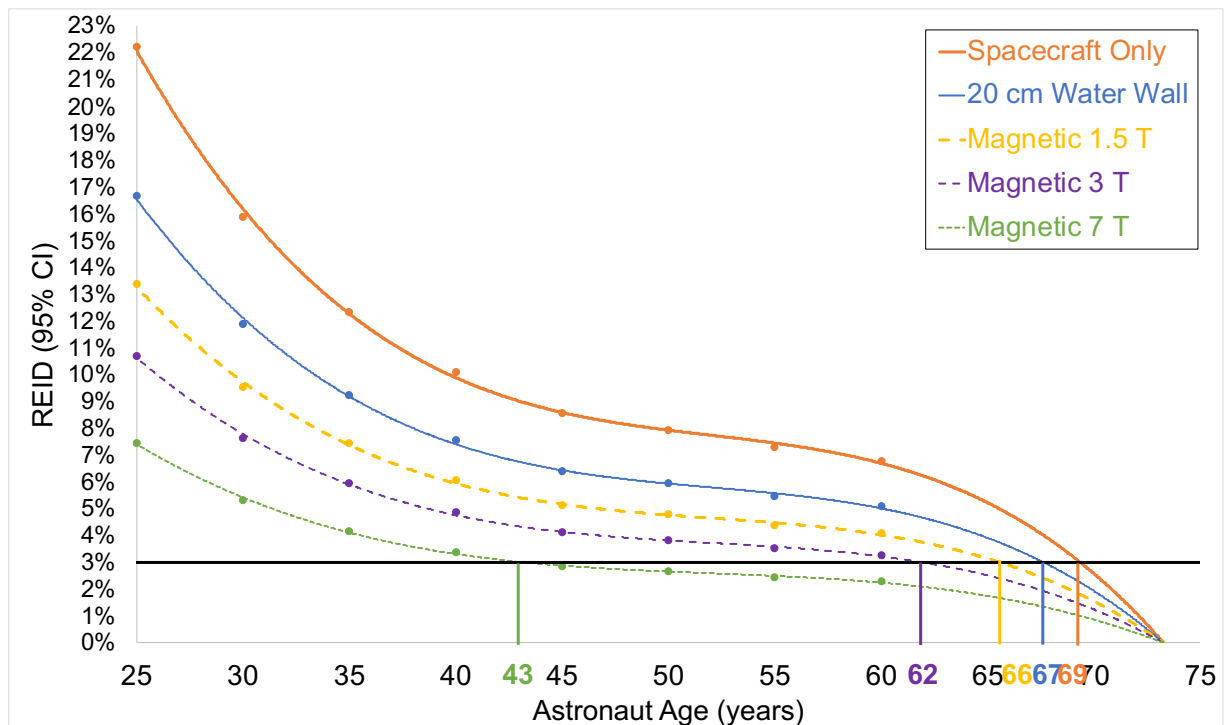


Figure 129: REID vs. Age (Female, 700 d, Average Magnetic)

For female astronauts on a 500 day mission, the REID versus crew age data (see Figure 130) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9869$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000007A^3 + 0.0009A^2 - 0.0440A + 0.7776$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0329A + 0.5824$$

Toroid 0 T

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0339A + 0.5994$$

Toroid 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0277A + 0.4902$$

Toroid 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0211A + 0.3727$$

Toroid 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0164A + 0.2901$$

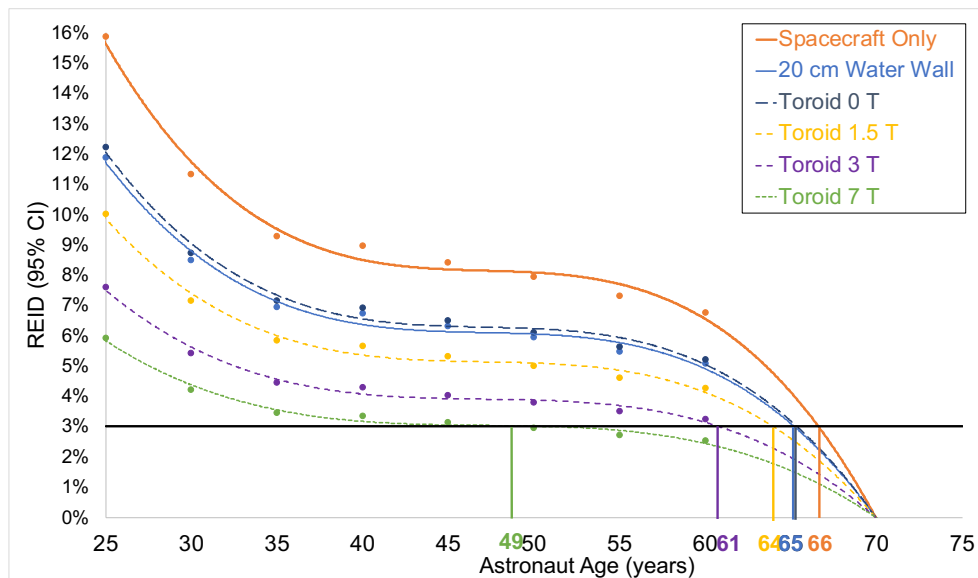


Figure 130: REID vs. Age (Female, 500 d, Toroid)

For female astronauts on a 500 day mission, the REID versus crew age data (see Figure 131) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9869$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000007A^3 + 0.0009A^2 - 0.0440A + 0.7776$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0329A + 0.5824$$

Solenoid 0 T

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0334A + 0.5905$$

Solenoid 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0263A + 0.4657$$

Solenoid 3 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0214A + 0.3790$$

Solenoid 7 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0117A + 0.2065$$

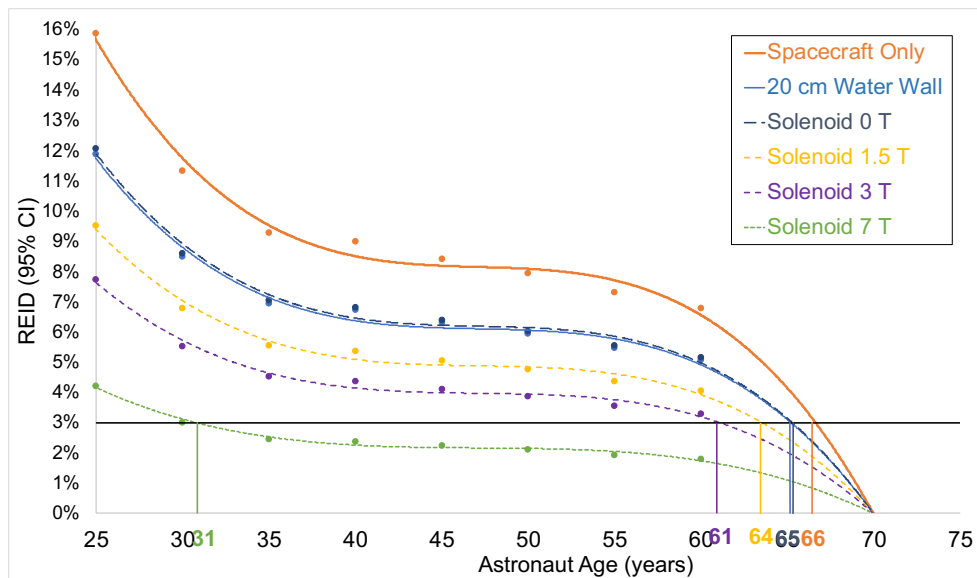


Figure 131: REID vs. Age (Female, 500 d, Solenoid)

For female astronauts on a 500 day mission, the REID versus crew age data (see Figure 132) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9869$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000007A^3 + 0.0009A^2 - 0.0440A + 0.7776$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0329A + 0.5824$$

Race Track 0 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0286A + 0.5054$$

Race Track 1.5 T

$$REID = -0.000004A^3 + 0.0005A^2 - 0.0253A + 0.4469$$

Race Track 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0210A + 0.3719$$

Race Track 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0161A + 0.2855$$

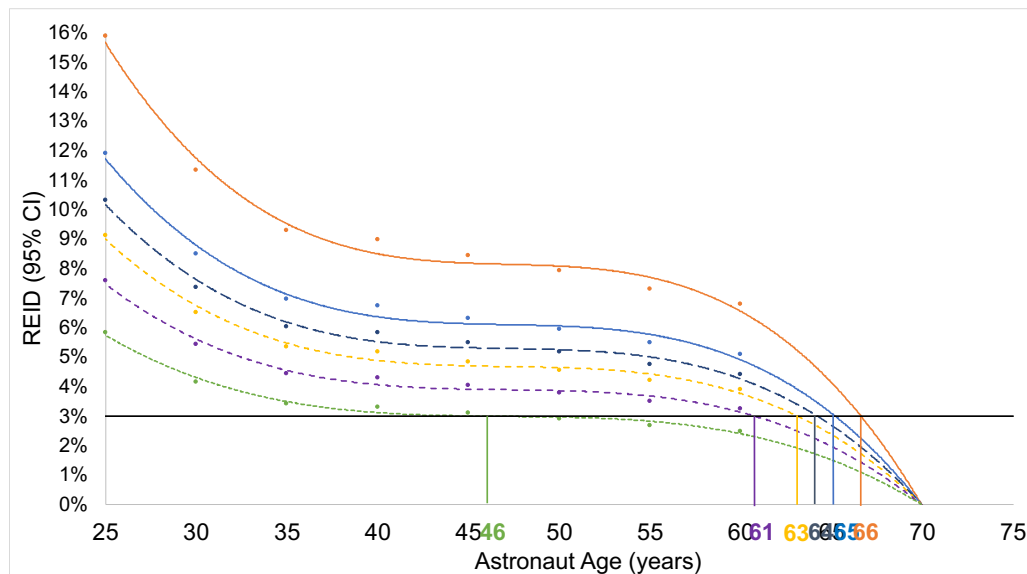


Figure 132: REID vs. Age (Female, 500 d, Race Track)

For female astronauts on a 500 day mission, the REID versus crew age data (see Figure 133) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (excluding zero magnetic field) best fit third-order polynomials ( $R^2 = 0.9869$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000007A^3 + 0.0009A^2 - 0.0440A + 0.7776$$

20 cm Water Wall

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0329A + 0.5824$$

Magnetic 1.5 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0264A + 0.4676$$

Magnetic 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0212A + 0.3745$$

Magnetic 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0147A + 0.2607$$

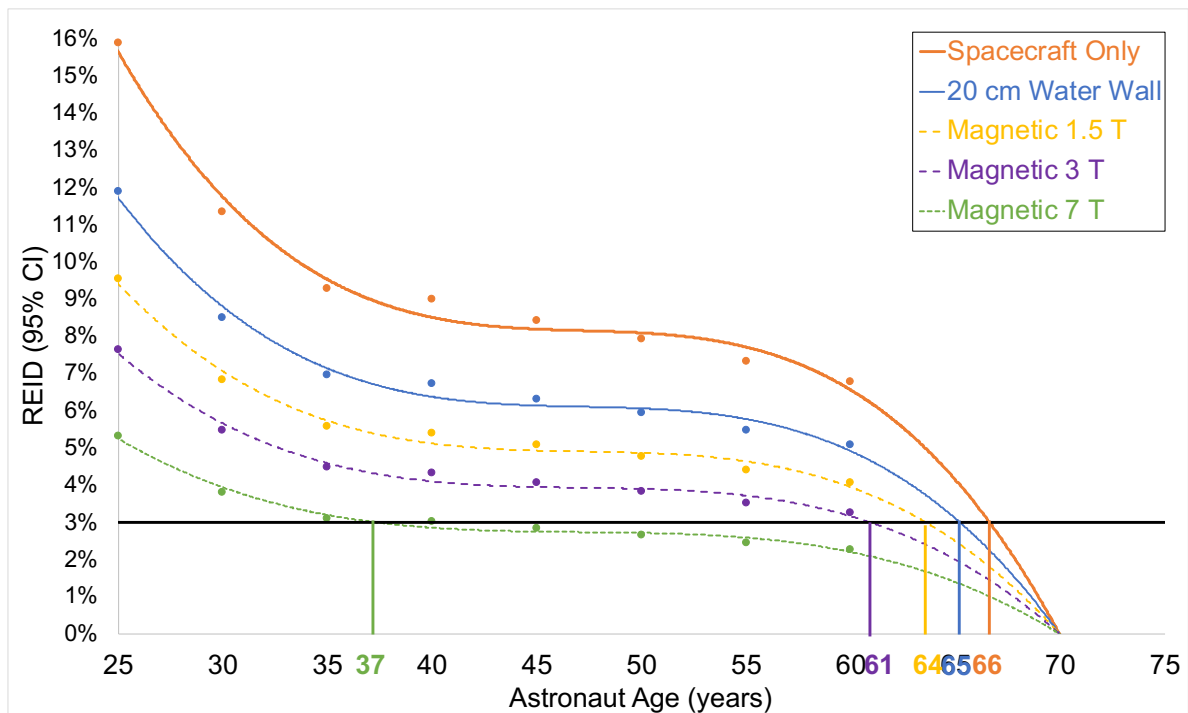


Figure 133: REID vs. Age (Female, 500 d, Average Magnetic)

For male astronauts on a 700 day mission, the REID versus crew age data (see Figure 134) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9983$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0336A + 0.6099$$

20 cm Water Wall

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0247A + 0.4483$$

Toroid 0 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0300A + 0.5450$$

Toroid 1.5 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0225A + 0.4092$$

Toroid 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0198A + 0.3602$$

Toroid 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0147A + 0.2667$$

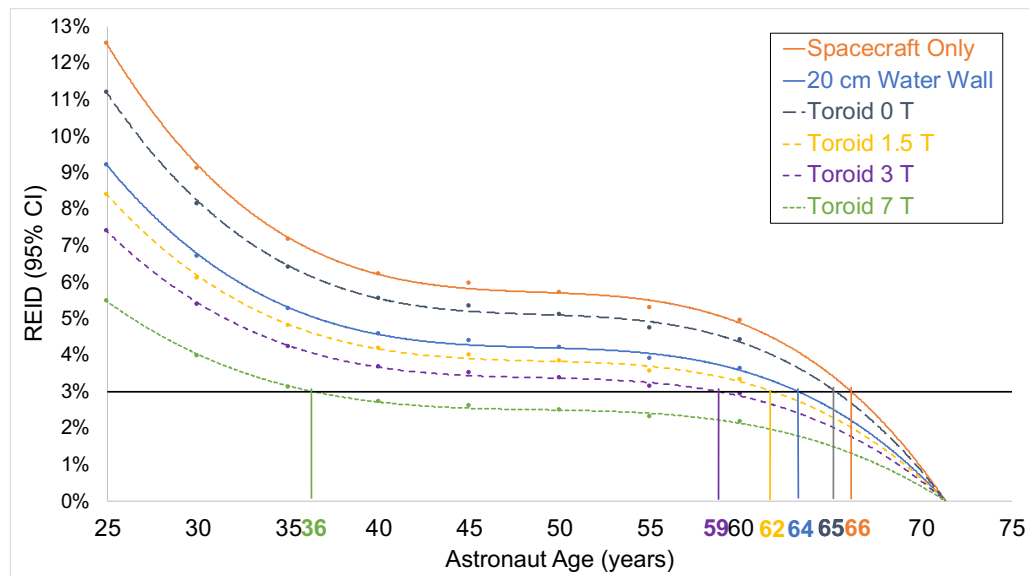


Figure 134: REID vs. Age (Male, 700 d, Toroid)

For male astronauts on a 700 day mission, the REID versus crew age data (see Figure 135) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9983$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0336A + 0.6099$$

20 cm Water Wall

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0247A + 0.4483$$

Solenoid 0 T

$$REID = -0.000004A^3 + 0.0006A^2 - 0.0313A + 0.5684$$

Solenoid 1.5 T

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0241A + 0.4370$$

Solenoid 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0198A + 0.3603$$

Solenoid 7 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0114A + 0.2065$$

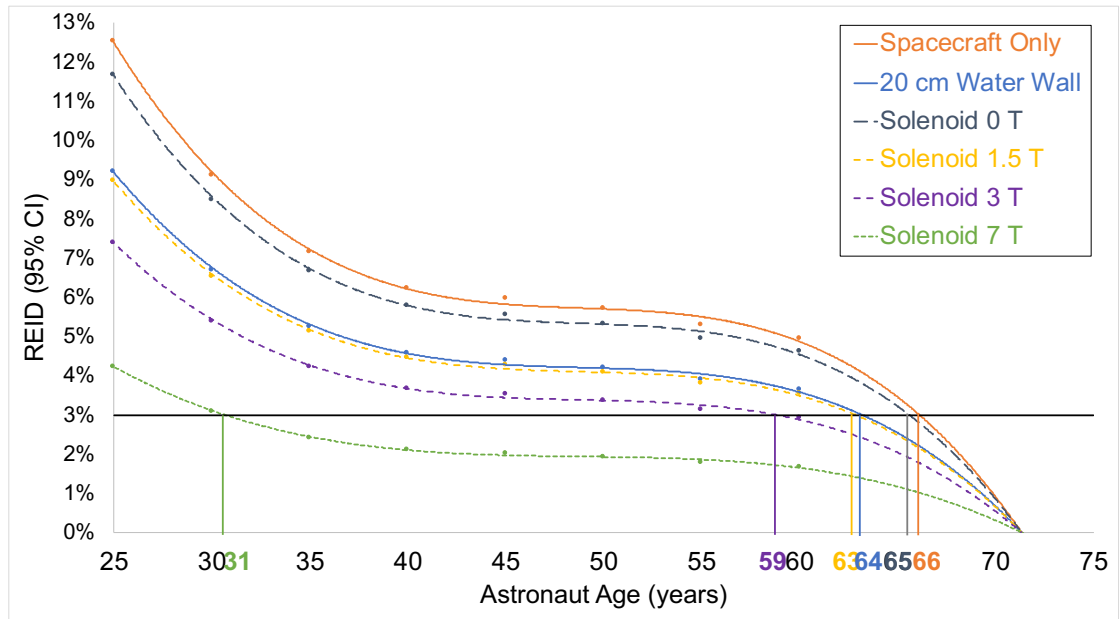


Figure 135: REID vs. Age (Male, 700 d, Solenoid)

For male astronauts on a 700 day mission, the REID versus crew age data (see Figure 136) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9983$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0336A + 0.6099$$

20 cm Water Wall

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0247A + 0.4483$$

Race Track 0 T

$$REID = -0.000003A^3 + 0.0006A^2 - 0.0227A + 0.4128$$

Race Track 1.5 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0193A + 0.3504$$

Race Track 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0190A + 0.3443$$

Race Track 7 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0120A + 0.2188$$

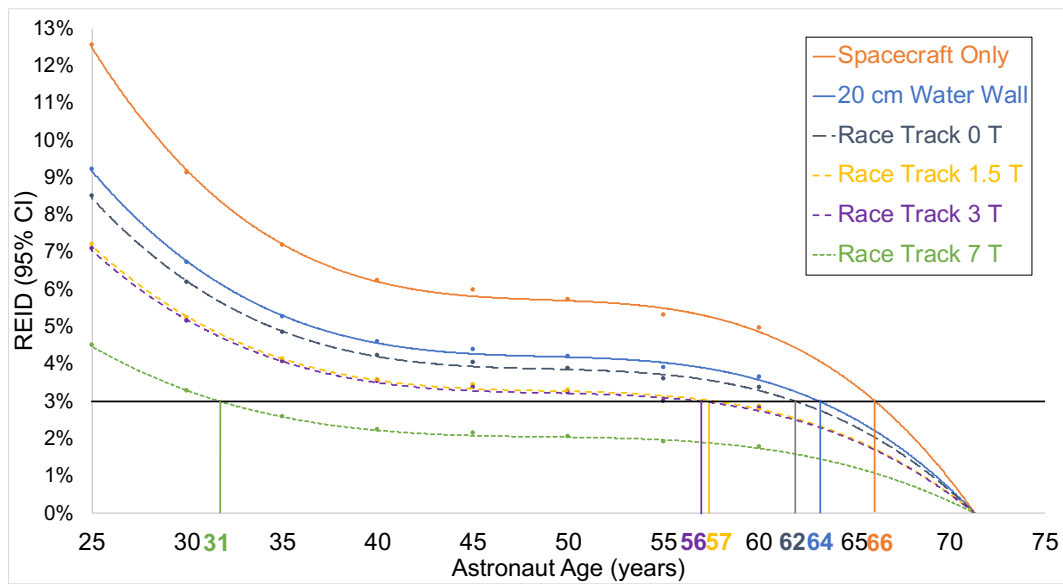


Figure 136: REID vs. Age (Male, 700 d, Race Track)

For male astronauts on a 700 day mission, the REID versus crew age data (see Figure 137) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (except zero magnetic field) best fit third-order polynomials ( $R^2 = 0.9983$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000005A^3 + 0.0007A^2 - 0.0336A + 0.6099$$

20 cm Water Wall

$$REID = -0.000003A^3 + 0.0005A^2 - 0.0247A + 0.4483$$

Magnetic 1.5 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0220A + 0.3988$$

Magnetic 3 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0195A + 0.3549$$

Magnetic 7 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0127A + 0.2306$$

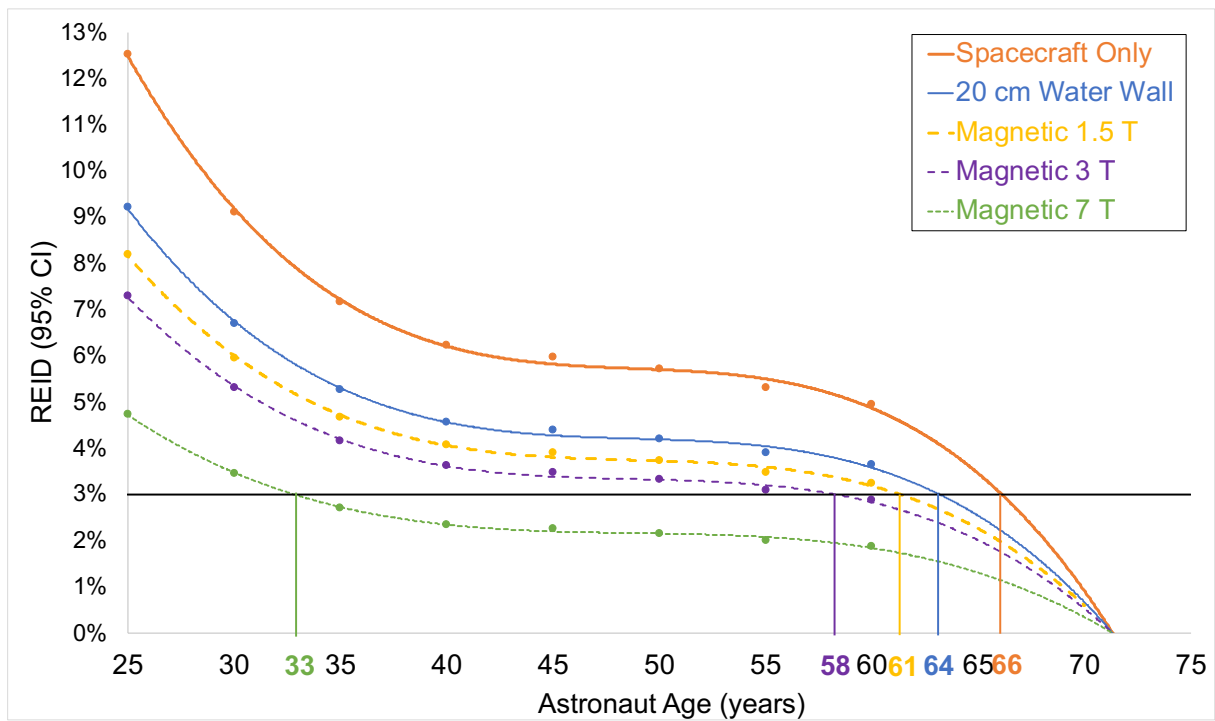


Figure 137: REID vs. Age (Male, 700 d, Average Magnetic)

For male astronauts on a 500 day mission, the REID versus crew age data (see Figure 138) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9554$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0194A + 0.3560$$

20 cm Water Wall

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0143A + 0.2617$$

Toroid 0 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0173A + 0.3181$$

Toroid 1.5 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0130A + 0.2388$$

Toroid 3 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0115A + 0.2102$$

Toroid 7 T

$$REID = -0.000001A^3 + 0.0002A^2 - 0.0085A + 0.1556$$

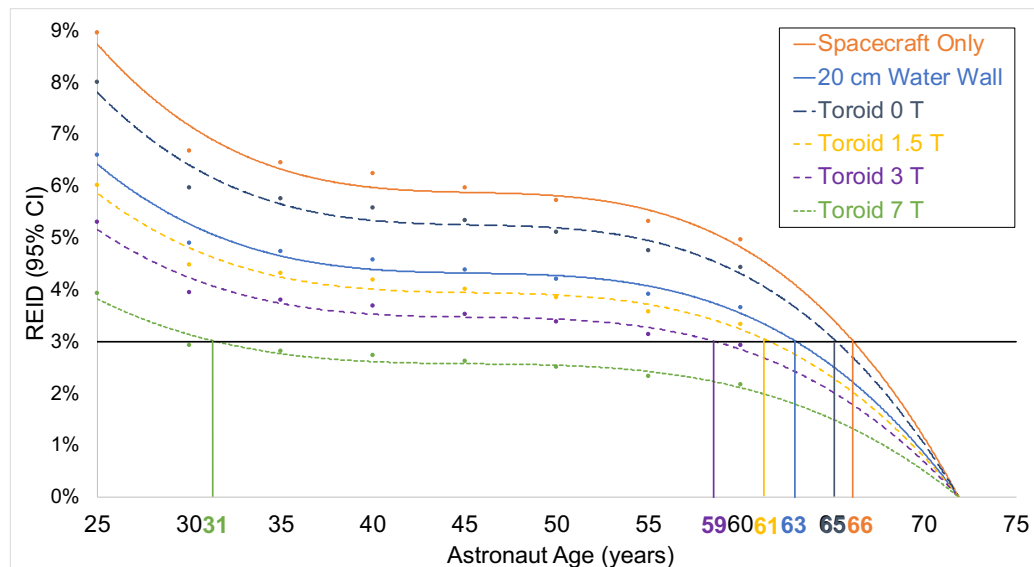


Figure 138: REID vs. Age (Male, 500 d, Toroid)

For male astronauts on a 500 day mission, the REID versus crew age data (see Figure 139) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9554$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0194A + 0.3560$$

20 cm Water Wall

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0143A + 0.2617$$

Solenoid 0 T

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0181A + 0.3318$$

Solenoid 1.5 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0139A + 0.2550$$

Solenoid 3 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0115A + 0.2103$$

Solenoid 7 T

$$REID = -0.000001A^3 + 0.0001A^2 - 0.0066A + 0.1205$$

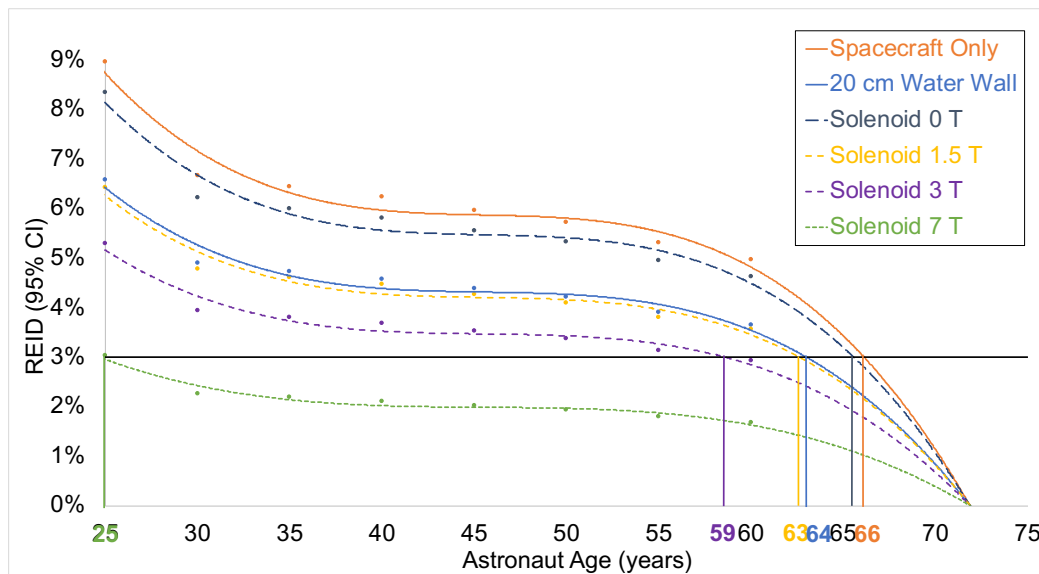


Figure 139: REID vs. Age (Male, 500 d, Solenoid)

For male astronauts on a 500 day mission, the REID versus crew age data (see Figure 140) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit third-order polynomials ( $R^2 = 0.9554$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0194A + 0.3560$$

20 cm Water Wall

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0143A + 0.2617$$

Race Track 0 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0131A + 0.2410$$

Race Track 1.5 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0112A + 0.2045$$

Race Track 3 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0110A + 0.2010$$

Race Track 7 T

$$REID = -0.000001A^3 + 0.0002A^2 - 0.0070A + 0.1277$$

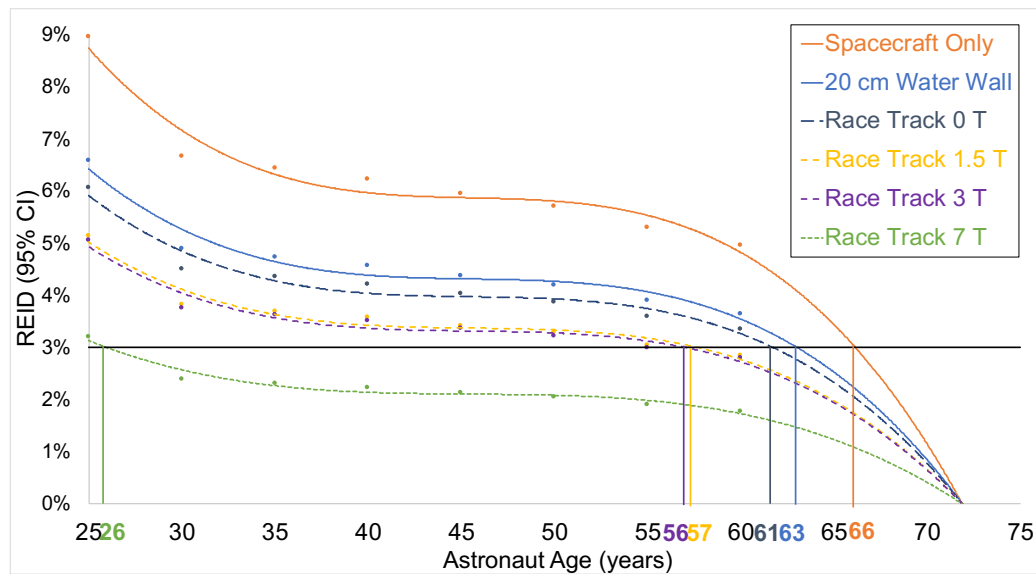


Figure 140: REID vs. Age (Male, 500 d, Race Track)

For male astronauts on a 500 day mission, the REID versus crew age data (see Figure 141) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (except zero magnetic field) best fit third-order polynomials ( $R^2 = 0.9554$ ) with fit equations calculated by Microsoft Excel where A is the astronaut age in years:

Aluminum spacecraft only

$$REID = -0.000003A^3 + 0.0004A^2 - 0.0194A + 0.3560$$

20 cm Water Wall

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0143A + 0.2617$$

Magnetic 1.5 T

$$REID = -0.000002A^3 + 0.0003A^2 - 0.0127A + 0.2328$$

Magnetic 3 T

$$REID = -0.000002A^3 + 0.0002A^2 - 0.0113A + 0.2072$$

Magnetic 7 T

$$REID = -0.000001A^3 + 0.0001A^2 - 0.0073A + 0.1346$$

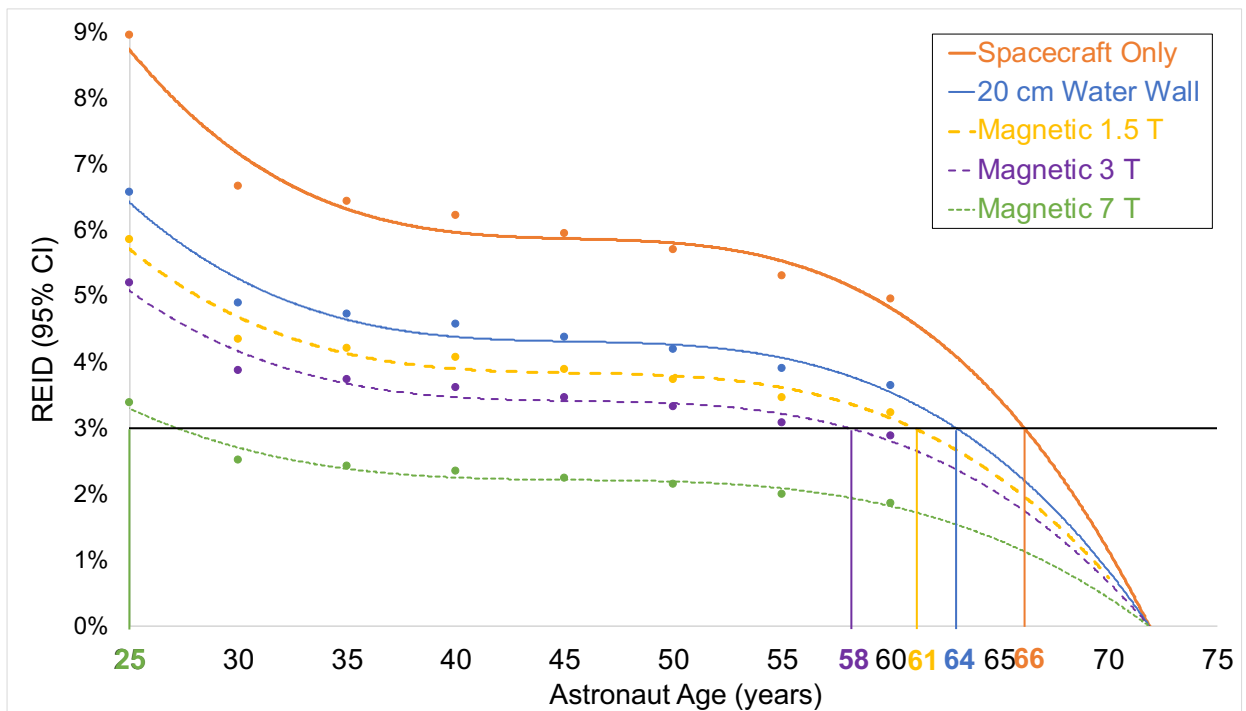


Figure 141: REID vs. Age (Male, 500 d, Average Magnetic)

For female astronauts, the safe days in space versus crew age data (Figure 142) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9993$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0002A^4 - 0.0407A^3 + 2.498A^2 - 57.324A + 511.39$$

20 cm Water Wall

$$SDS = 0.0003A^4 - 0.0544A^3 + 3.3352A^2 - 76.536A + 682.77$$

Toroid 0 T

$$SDS = 0.0003A^4 - 0.0528A^3 + 3.2406A^2 - 74.364A + 663.39$$

Toroid 1.5 T

$$SDS = 0.0003A^4 - 0.0646A^3 + 3.9624A^2 - 90.929A + 811.17$$

Toroid 3 T

$$SDS = 0.0005A^4 - 0.0849A^3 + 5.2118A^2 - 119.6A + 1066.9$$

Toroid 7 T

$$SDS = 0.0006A^4 - 0.1091A^3 + 6.6963A^2 - 153.66A + 1370.8$$

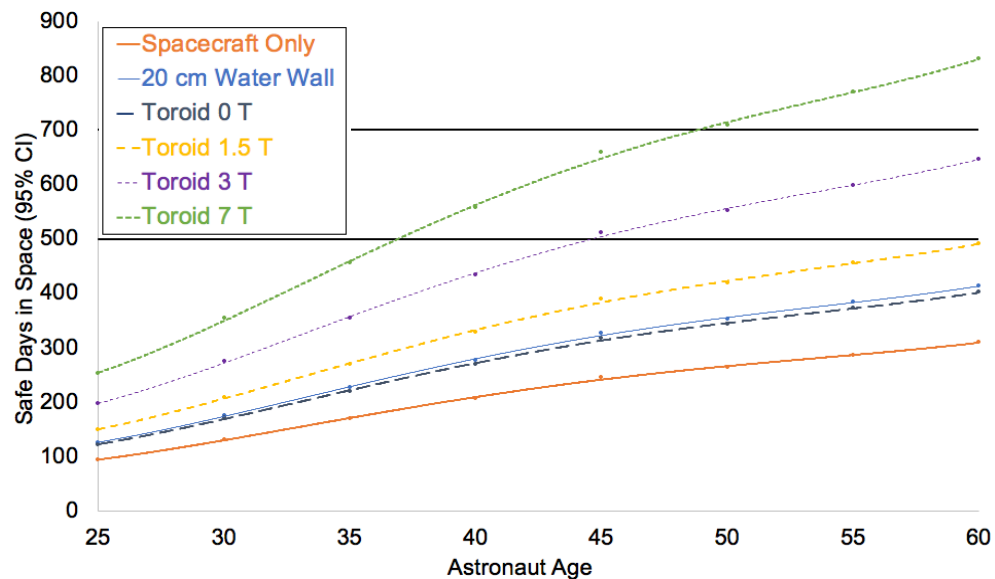


Figure 142: Safe Days in Space vs. Age (Female, Toroid)

For female astronauts, the safe days in space versus crew age data (Figure 143) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9993$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0002A^4 - 0.0407A^3 + 2.498A^2 - 57.324A + 511.39$$

20 cm Water Wall

$$SDS = 0.0003A^4 - 0.0544A^3 + 3.3352A^2 - 76.536A + 682.77$$

Solenoid 0 T

$$SDS = 0.0003A^4 - 0.0536A^3 + 3.2895A^2 - 75.486A + 673.4$$

Solenoid 1.5 T

$$SDS = 0.0004A^4 - 0.0680A^3 + 4.1711A^2 - 95.718A + 853.89$$

Solenoid 3 T

$$SDS = 0.0005A^4 - 0.0835A^3 + 5.1259A^2 - 117.63A + 1049.3$$

Solenoid 7 T

$$SDS = 0.0009A^4 - 0.1533A^3 + 9.4062A^2 - 215.85A + 1925.6$$

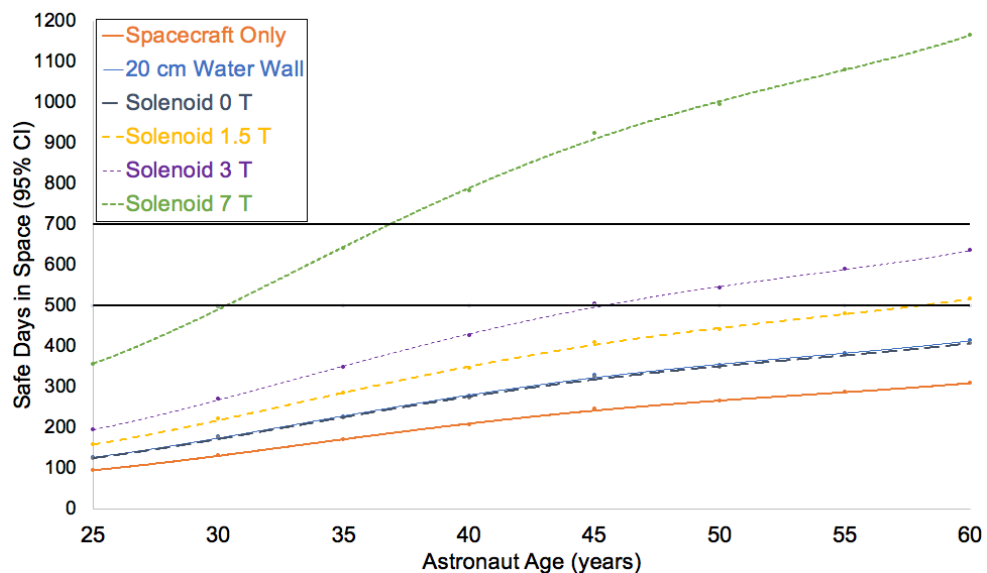


Figure 143: Safe Days in Space vs. Age (Female, Solenoid)

For female astronauts, the safe days in space versus crew age data (Figure 144) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9993$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0002A^4 - 0.0407A^3 + 2.498A^2 - 57.324A + 511.39$$

20 cm Water Wall

$$SDS = 0.0003A^4 - 0.0544A^3 + 3.3352A^2 - 76.536A + 682.77$$

Race Track 0 T

$$SDS = 0.0004A^4 - 0.0626A^3 + 3.3834A^2 - 88.196A + 786.79$$

Race Track 1.5 T

$$SDS = 0.0004A^4 - 0.0708A^3 + 4.3461A^2 - 99.732A + 889.7$$

Race Track 3 T

$$SDS = 0.0005A^4 - 0.0851A^3 + 5.2237A^2 - 119.87A + 1069.4$$

Race Track 7 T

$$SDS = 0.0006A^4 - 0.1109A^3 + 6.8031A^2 - 156.12A + 1392.7$$

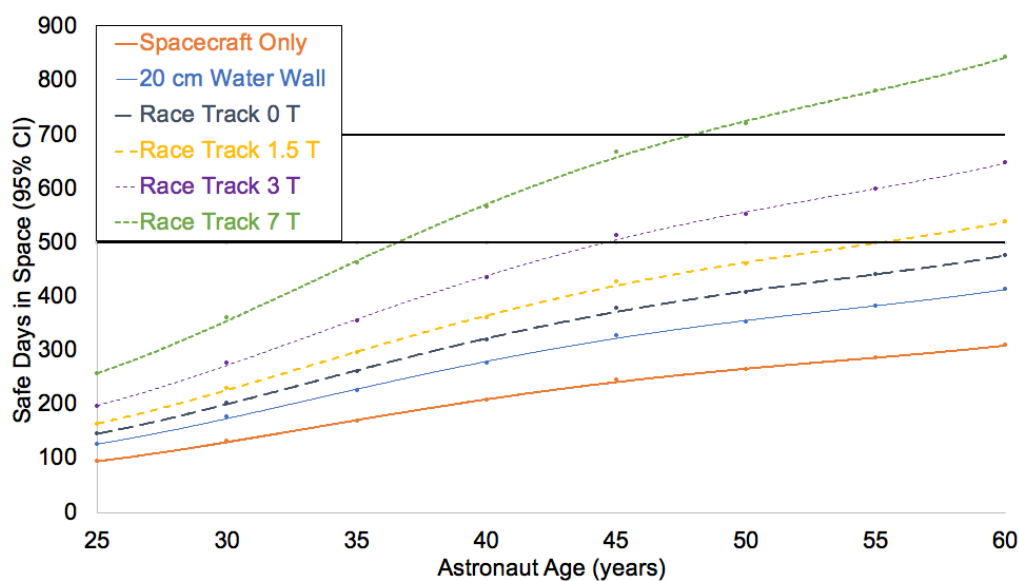


Figure 144: Safe Days in Space vs. Age (Female, Race Track)

For female astronauts, the safe days in space versus crew age data (Figure 145) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (except zero magnetic field) best fit fourth order polynomial functions ( $R^2 = 0.9993$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0002A^4 - 0.0407A^3 + 2.498A^2 - 57.324A + 511.39$$

20 cm Water Wall

$$SDS = 0.0003A^4 - 0.0544A^3 + 3.3352A^2 - 76.536A + 682.77$$

Magnetic 1.5 T

$$SDS = 0.0004A^4 - 0.0677A^3 + 4.1539A^2 - 95.323A + 850.37$$

Magnetic 3 T

$$SSDS = 0.0005A^4 - 0.0845A^3 + 5.1867A^2 - 119.02A + 1061.8$$

Magnetic 7 T

$$SDS = 0.0007A^4 - 0.1214A^3 + 7.4508A^2 - 170.98A + 1525.3$$

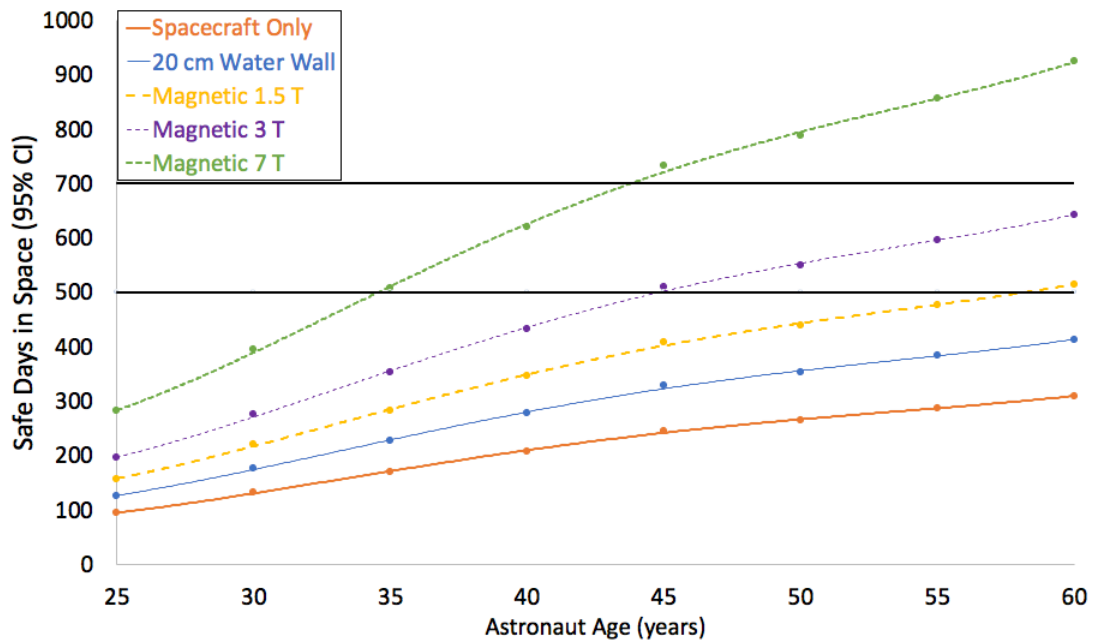


Figure 145: Safe Days in Space vs. Age (Female, Average Magnetic)

For male astronauts, the safe days in space versus crew age data (Figure 146) for the aluminum spacecraft only, 20 cm water wall, and all four toroidal magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9973$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0003A^4 - 0.0491A^3 + 2.3421A^2 - 31.671A + 129.05$$

20 cm Water Wall

$$SDS = 0.0005A^4 - 0.0667A^3 + 3.1864A^2 - 43.088A + 175.57$$

Toroid 0 T

$$SDS = 0.0004A^4 - 0.0549A^3 + 2.6214A^2 - 35.448A + 144.44$$

Toroid 1.5 T

$$SDS = 0.0005A^4 - 0.0731A^3 + 3.4914A^2 - 47.213A + 192.38$$

Toroid 3 T

$$SDS = 0.0006A^4 - 0.0831A^3 + 3.9665A^2 - 53.637A + 218.56$$

Toroid 7 T

$$SDS = 0.0008A^4 - 0.1122A^3 + 5.3571A^2 - 72.442A + 295.18$$

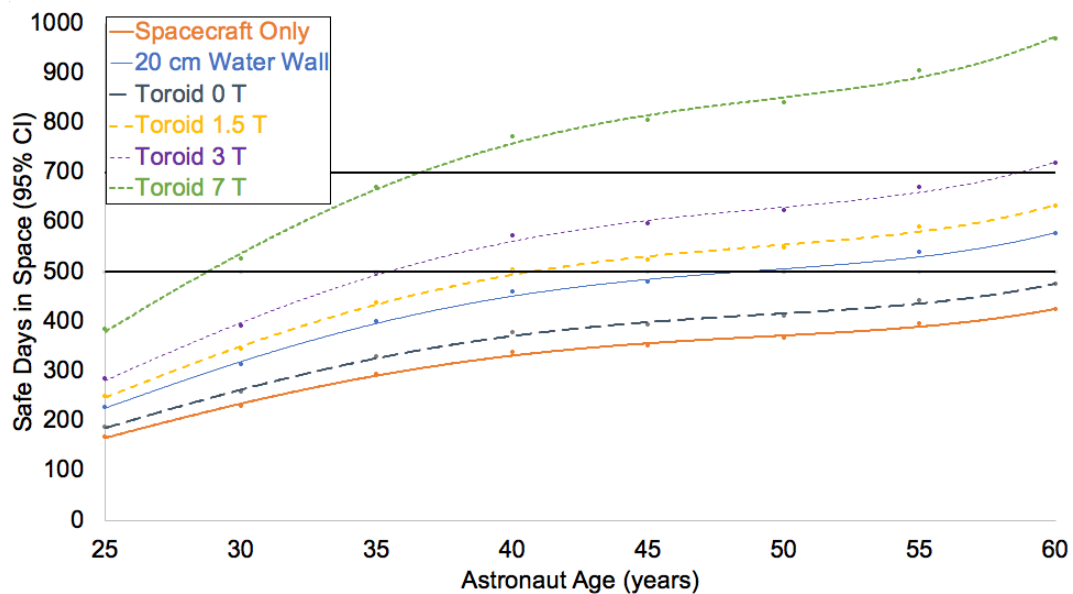


Figure 146: Safe Days in Space vs. Age (Male, Toroid)

For male astronauts, the safe days in space versus crew age data (Figure 147) for the aluminum spacecraft only, 20 cm water wall, and all four solenoidal magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9973$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0003A^4 - 0.0491A^3 + 2.3421A^2 - 31.671A + 129.05$$

20 cm Water Wall

$$SDS = 0.0005A^4 - 0.0667A^3 + 3.1864A^2 - 43.088A + 175.57$$

Solenoid 0 T

$$SDS = 0.0004A^4 - 0.0526A^3 + 2.5131A^2 - 33.984A + 138.48$$

Solenoid 1.5 T

$$SDS = 0.0005A^4 - 0.0685A^3 + 3.2694A^2 - 44.210A + 180.15$$

Solenoid 3 T

$$SDS = 0.0006A^4 - 0.0083A^3 + 3.9645A^2 - 53.609A + 218.44$$

Solenoid 7 T

$$SDS = 0.0010A^4 - 0.1449A^3 + 6.919A^2 - 93.562A + 381.24$$

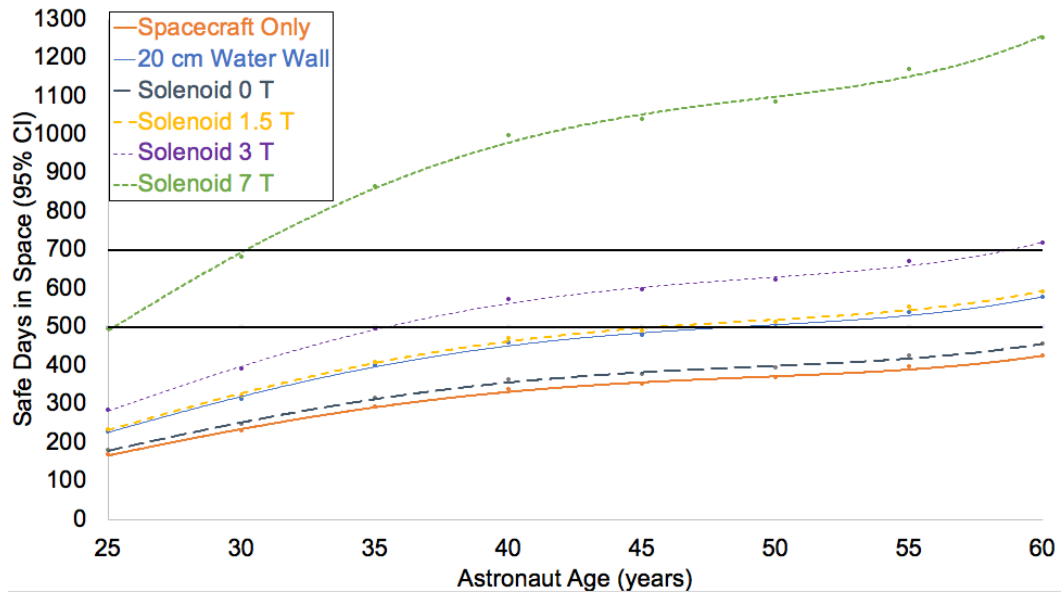


Figure 147: Safe Days in Space vs. Age (Male, Solenoid)

For male astronauts, the safe days in space versus crew age data (Figure 148) for the aluminum spacecraft only, 20 cm water wall, and all four race track magnetic shielding configurations best fit fourth order polynomial functions ( $R^2 = 0.9973$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SDS = 0.0003A^4 - 0.0491A^3 + 2.3421A^2 - 31.671A + 129.05$$

20 cm Water Wall

$$SDS = 0.0005A^4 - 0.0667A^3 + 3.1864A^2 - 43.088A + 175.57$$

Race Track 0 T

$$SDS = 0.0005A^4 - 0.0725A^3 + 3.4604A^2 - 46.794A + 190.67$$

Race Track 1.5 T

$$SDS = 0.0006A^4 - 0.0854A^3 + 4.0771A^2 - 55.132A + 224.65$$

Race Track 3 T

$$SDS = 0.0006A^4 - 0.0869A^3 + 4.1491A^2 - 56.107A + 228.62$$

Race Track 7 T

$$SDS = 0.0009A^4 - 0.1368A^3 + 6.5303A^2 - 88.306A + 359.83$$

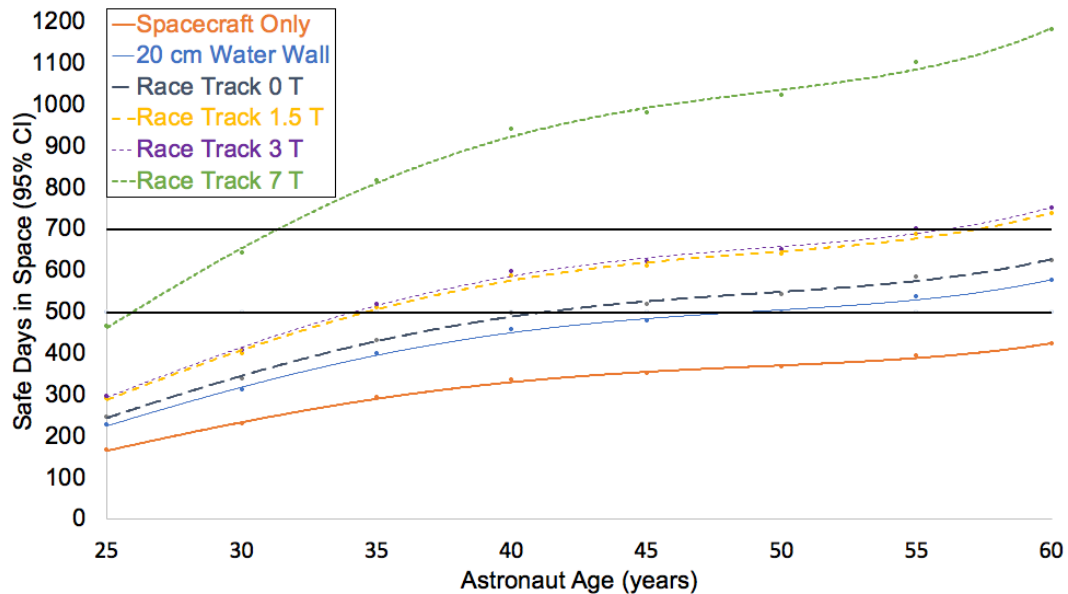


Figure 148: Safe Days in Space vs. Age (Male, Race Track)

For male astronauts, the safe days in space versus crew age data (Figure 149) for the aluminum spacecraft only, 20 cm water wall, and the average of all magnetic shielding configurations (except zero magnetic field) best fit fourth order polynomial functions ( $R^2 = 0.9973$ ) with fit equations calculated by Microsoft Excel where SDS is safe days in space and A is the astronaut age in years:

Aluminum spacecraft only

$$SSDS = 0.0003A^4 - 0.0491A^3 + 2.3421A^2 - 31.671A + 129.05$$

20 cm Water Wall

$$SDS = 0.0005A^4 - 0.0667A^3 + 3.1864A^2 - 43.088A + 175.57$$

Magnetic 1.5 T

$$SDS = 0.0004A^4 - 0.0677A^3 + 4.1539A^2 - 95.323A + 850.37$$

Magnetic 3 T

$$SDS = 0.0005A^4 - 0.0845A^3 + 5.1867A^2 - 119.02A + 1061.8$$

Magnetic 7 T

$$SDS = 0.0007A^4 - 0.1214A^3 + 7.4508A^2 - 170.98A + 1525.3$$

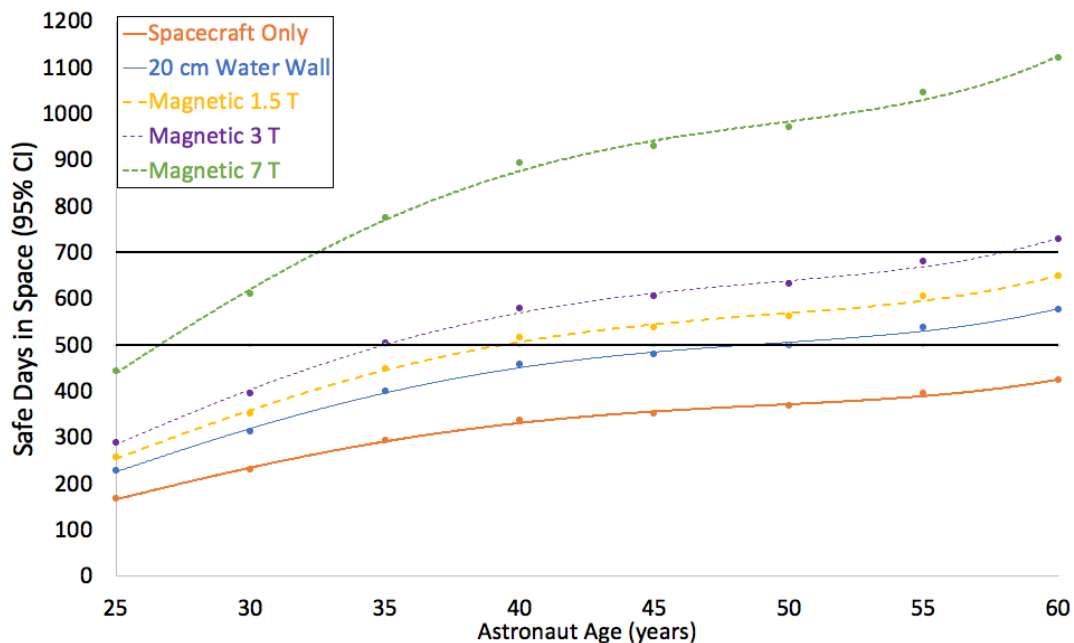


Figure 149: Safe Days in Space vs. Age (Male, Average Magnetic)

**Table 27: 3<sup>rd</sup> Order Polynomial Coefficients for REID vs. Astronaut Age**

Scenario	Shielding Type	A3	A2	A	C
Female, 700 d	Spacecraft Only	-0.000006	0.0009	-0.0498	0.9720
	20 cm Water Wall	-0.000005	0.0007	-0.0373	0.7280
	Toroid 0 T	-0.000005	0.0007	-0.0384	0.7493
	Toroid 1.5 T	-0.000004	0.0006	-0.0314	0.6128
	Toroid 3 T	-0.000003	0.0005	-0.0238	0.4659
	Toroid 7 T	-0.000002	0.0004	-0.0186	0.3626
	Solenoid 0 T	-0.000005	0.0007	-0.0378	0.7382
	Solenoid 1.5 T	-0.000004	0.0006	-0.0298	0.5821
	Solenoid 3 T	-0.000003	0.0005	-0.0242	0.4737
	Solenoid 7 T	-0.000002	0.0002	-0.0132	0.2581
	Race Track 0 T	-0.000004	0.0006	-0.0323	0.6318
	Race Track 1.5 T	-0.000003	0.0004	-0.0286	0.5587
	Race Track 3 T	-0.000003	0.0004	-0.0238	0.4648
	Race Track 7 T	-0.000002	0.0003	-0.0183	0.3569
	Magnetic 1.5 T	-0.000004	0.0006	-0.0299	0.5846
	Magnetic 3 T	-0.000003	0.0005	-0.0240	0.4682
	Magnetic 7 T	-0.000002	0.0003	-0.0167	0.3259
Female, 500 d	Spacecraft Only	-0.000007	0.0009	-0.0440	0.7776
	20 cm Water Wall	-0.000005	0.0007	-0.0329	0.5824
	Toroid 0 T	-0.000005	0.0007	-0.0339	0.5994
	Toroid 1.5 T	-0.000004	0.0006	-0.0277	0.4902
	Toroid 3 T	-0.000003	0.0004	-0.0211	0.3727
	Toroid 7 T	-0.000002	0.0003	-0.0164	0.2901
	Solenoid 0 T	-0.000005	0.0007	-0.0334	0.5905
	Solenoid 1.5 T	-0.000004	0.0006	-0.0263	0.4657
	Solenoid 3 T	-0.000003	0.0005	-0.0214	0.3790
	Solenoid 7 T	-0.000002	0.0002	-0.0117	0.2065
	Race Track 0 T	-0.000004	0.0006	-0.0286	0.5054
	Race Track 1.5 T	-0.000004	0.0005	-0.0253	0.4469
	Race Track 3 T	-0.000003	0.0004	-0.0210	0.3719
	Race Track 7 T	-0.000002	0.0003	-0.0161	0.2855
	Magnetic 1.5 T	-0.000004	0.0006	-0.0264	0.4676
	Magnetic 3 T	-0.000003	0.0004	-0.0212	0.3745
	Magnetic 7 T	-0.000002	0.0003	-0.0147	0.2607
	Spacecraft Only	-0.000005	0.0007	-0.0336	0.6099
	20 cm Water Wall	-0.000003	0.0005	-0.0247	0.4483
	Toroid 0 T	-0.000004	0.0006	-0.0300	0.5450

Male, 700 d	Toroid 1.5 T	-0.000003	0.0005	-0.0225	0.4092
	Toroid 3 T	-0.000003	0.0004	-0.0198	0.3602
	Toroid 7 T	-0.000002	0.0003	-0.0147	0.2667
	Solenoid 0 T	-0.000004	0.0006	-0.0313	0.5684
	Solenoid 1.5 T	-0.000003	0.0005	-0.0241	0.4370
	Solenoid 3 T	-0.000003	0.0004	-0.0198	0.3603
	Solenoid 7 T	-0.000002	0.0002	-0.0114	0.2065
	Race Track 0 T	-0.000003	0.0006	-0.0227	0.4128
	Race Track 1.5 T	-0.000003	0.0004	-0.0193	0.3504
	Race Track 3 T	-0.000003	0.0004	-0.0190	0.3443
	Race Track 7 T	-0.000002	0.0002	-0.0120	0.2188
	Magnetic 1.5 T	-0.000003	0.0004	-0.0220	0.3988
	Magnetic 3 T	-0.000003	0.0004	-0.0195	0.3549
	Magnetic 7 T	-0.000002	0.0003	-0.0127	0.2306
Male, 500 d	Spacecraft Only	-0.000003	0.0004	-0.0194	0.3560
	20 cm Water Wall	-0.000002	0.0003	-0.0143	0.2617
	Toroid 0 T	-0.000003	0.0004	-0.0173	0.3181
	Toroid 1.5 T	-0.000002	0.0003	-0.0130	0.2388
	Toroid 3 T	-0.000002	0.0003	-0.0115	0.2102
	Toroid 7 T	-0.000001	0.0002	-0.0085	0.1556
	Solenoid 0 T	-0.000003	0.0004	-0.0181	0.3318
	Solenoid 1.5 T	-0.000002	0.0003	-0.0139	0.2550
	Solenoid 3 T	-0.000002	0.0003	-0.0115	0.2103
	Solenoid 7 T	-0.000001	0.0001	-0.0066	0.1205
	Race Track 0 T	-0.000002	0.0003	-0.0131	0.2410
	Race Track 1.5 T	-0.000002	0.0002	-0.0112	0.2045
	Race Track 3 T	-0.000002	0.0002	-0.0110	0.2010
	Race Track 7 T	-0.000001	0.0002	-0.0070	0.1277
	Magnetic 1.5 T	-0.000002	0.0003	-0.0127	0.2328
	Magnetic 3 T	-0.000002	0.0002	-0.0113	0.2072
	Magnetic 7 T	-0.000001	0.0001	-0.0073	0.1346

**Table 28: 4th Order Polynomial Coefficients for SDS vs. Astronaut Age**

Scenario	Shielding Type	A4	A3	A2	A	C
Female	Spacecraft Only	0.0002	-0.0407	2.498	-57.32	511.39
	20 cm Water Wall	0.0003	-0.0544	3.3352	-76.54	682.77
	Toroid 0 T	0.0003	-0.0528	3.2406	-74.36	663.39
	Toroid 1.5 T	0.0003	-0.0646	3.9624	-90.93	811.17
	Toroid 3 T	0.0005	-0.8490	5.2118	-119.60	1066.90
	Toroid 7 T	0.0006	-0.1091	6.6963	-153.66	1370.80
	Solenoid 0 T	0.0003	-0.0536	3.2895	-75.49	673.40
	Solenoid 1.5 T	0.0004	-0.0680	4.1711	-95.72	853.89
	Solenoid 3 T	0.0005	-0.0835	5.1259	-117.63	1049.30
	Solenoid 7 T	0.0009	-0.1533	9.4062	-215.85	1925.60
	Race Track 0 T	0.0004	-0.0626	3.3834	-88.20	786.79
	Race Track 1.5 T	0.0004	-0.0708	4.3461	-99.73	889.70
	Race Track 3 T	0.0005	-0.0851	5.2237	-119.87	1069.40
	Race Track 7 T	0.0006	-0.1109	6.8031	-156.12	1392.70
	Magnetic 1.5 T	0.0004	-0.0677	4.1539	-95.32	850.37
	Magnetic 3 T	0.0005	-0.0845	5.1867	-119.02	1061.80
	Magnetic 7 T	0.0007	-0.1214	7.4508	-170.98	1525.30
Male	Spacecraft Only	0.0003	-0.0491	2.3421	-31.67	129.05
	20 cm Water Wall	0.0005	-0.0667	3.1864	-43.09	175.57
	Toroid 0 T	0.0004	-0.0549	2.6214	-35.45	144.44
	Toroid 1.5 T	0.0005	-0.0731	3.4914	-47.21	192.38
	Toroid 3 T	0.0006	-0.0831	3.9665	-53.64	218.56
	Toroid 7 T	0.0008	-0.1122	5.3571	-72.44	295.18
	Solenoid 0 T	0.0004	-0.0526	2.5131	-33.98	138.48
	Solenoid 1.5 T	0.0005	-0.0685	3.2694	-44.21	180.15
	Solenoid 3 T	0.0006	-0.0083	3.9645	-53.61	218.44
	Solenoid 7 T	0.0010	-0.1449	6.9190	-93.56	381.24
	Race Track 0 T	0.0005	-0.0725	3.4604	-46.79	190.67
	Race Track 1.5 T	0.0006	-0.0854	4.0771	-55.13	224.65
	Race Track 3 T	0.0006	-0.0869	4.1491	-56.11	228.62
	Race Track 7 T	0.0009	-0.0137	6.5303	-88.31	359.83
	Magnetic 1.5 T	0.0004	-0.0677	4.1539	-95.32	850.37
	Magnetic 3 T	0.0005	-0.0845	5.1867	-119.02	1061.80
	Magnetic 7 T	0.0007	-0.1214	7.4508	-170.98	1525.30

It is interesting to note that the third order polynomial curves for REID converge at minimal risk for both female and male astronauts at ages in the 70s, which tracks with statistics on average life expectancy and timeline for cancer induction. Safe days in space curves indicate even the shorter mission duration of 500 days is imprudent for female or male astronauts of any reasonable age with minimal (aluminum spacecraft only) shielding. Passive water shielding may support the shorter mission duration for male but not females in the typical astronaut age range. Magnetic shielding allows longer mission durations for male and female astronauts of all ages and facilitates the 500- and 700-day missions at ages in the 30s.

Current radiation shielding options for interplanetary flight are not adequate give current risk tolerances. Active magnetic shielding may be a solution based on the results of this study, but a high magnetic field is required to obtain maximum benefit. Radiation-induced cancer risk is still not eliminated in these scenarios but is reduced dramatically to within current policy guidelines. The data from the safe days in space curves may also be used to estimate safe mission durations for any human mission outside of Earth orbit. For example, from Figure 145 we can estimate that a 45-year-old female astronaut could safely stay at a lunar space station with water shielding for approximately 275 days before reaching current radiation dose limits. This data can also help determine the benefit of investment in advanced propulsion technology to shorten mission duration.

It is important to note that the REID values presented represent a 95% confidence interval, which current NASA policy requires. This introduces a conservative element to the estimates of the effective dose and REID because of the uncertainties present in our models. On the other hand, REID focuses on *mortality* due to radiation-induced cancers so therefore does not examine the *incidence* of radiation-induced cancers. REID also does not include risks associated with cardiac toxicity or CNS effects. Therefore, these risk estimates may actually underestimate the

total risk to the astronauts. Future work to reduce uncertainties in these types of simulations could improve the accuracy of these estimates and should be sensitive to any changes in modeling high LET effects in determining radiation and/or tissue weighting factors.

Overall, REID due to space radiation is only part of the risk profile for an interplanetary mission. Current estimates indicate that the overall loss of crew risk on this type of mission will approach or exceed 10% (Radcliffe et al. 2016). Therefore, it is possible that space policy will be modified to accept higher REID values for high-value missions. Even with a more relaxed radiation risk posture, advanced shielding technologies are likely to be required eventually for missions further into the solar system when younger astronauts will be required and when mission durations will be even longer.

### *3.9 Validation of Results*

All space technology development suffers from limited options for validation of experimental results due to the inability to replicate space environment conditions on Earth. Even in the highest fidelity space radiation simulator at the NASA Space Radiation Laboratory (NSRL) at Brookhaven National Laboratory, only certain ions and energies comparable to galactic cosmic rays are available, the dose rate is orders of magnitude larger, and the radiation is delivered by a beam geometry rather than an isotropic geometry. NSRL provides useful insight into the effects of space radiation on equipment and live tissues; however, the studied effects are isolated, and results must be interpolated and extrapolated along several dimensions to draw conclusions on the effects of the full space radiation environment.

Given these limitations, Earth-based experiments involving space technology are typically conducted using simulations of the space radiation environment, quality of vacuum, and other conditions available only in space. This is the case for our experiments, in that it is very difficult

and expensive if not impossible to validate many elements of our simulations without flying prototypes in space. In particular, even flying a prototype to the International Space Station in low Earth orbit would not be adequate to validate the conditions in our experiments since our mission of interest involves the interplanetary environment which is vastly different than the environment in low Earth orbit.

Due to the limited ability to validate our experiments to the level expected of other types of Monte Carlo simulations, we look to other means to validate pieces of our experiment. First, we chose to conduct our simulations using the GEANT4 toolkit, which has the privilege of validation in the actual space environment across many examples of space instruments. Further, we leveraged the example codes packaged with the GEANT4 installation as much as possible in building our simulations, which further increases our confidence that our baseline code is reliable.

For additional validation, we can compare dose values we calculated to values from similar studies for some of the scenarios we studied. First, we are able to validate our minimal shielding results by matching our total mission effective dose values with those expected by NASA (~1 Sv/yr) (Simonsen 2017). Second, we are able to compare the results of our magnetic shielding experiments to the results of prior studies as described in Section 3.6. Comparison of our results to the 2015 Chinese study, the 2014 NASA MAARS study, and the 2016 ESA SR2S study (Geng et al. 2015; Westover 2014; Vuolo et al. 2016) indicate that despite differences in shielding configuration geometry, superconductor material, magnetic fields, spacecraft, environment model, phantom selection, and dosimetry method that the results agree within an order of magnitude, showing GCR and SPE dose reduction between 50-80% at magnetic fields between 2 and 8 T.

Regarding the organ dose component of our study, our result that active bone marrow has the highest relative radiation-induced cancer risk of all organs overall agrees with findings from the Nagasaki and Hiroshima atomic bomb survivor data showing leukemia to be one of the most common radiation-induced cancers (Peterson & Kovyrshina 2015).

Further, our safe days in space (SDS) analysis is consistent with NASA projections of SDS in that SDS for a 45-year-old female astronaut is approximately 200 days and for a 45-year-old male astronaut is approximately 250 days at a 95% CI (Cucinotta et al. 2015).

### *3.13 Engineering Challenges*

While several active shielding configurations have the potential to offer mass savings over equivalent performance passive shielding configurations, there are several engineering challenges associated with the advanced technology of active shielding. In the specific case of magnetic shielding using superconducting magnets, the core systems of interest are the superconductor material, thermal regulation, and power consumption (Washburn et al. 2015). The Active Radiation for Space Exploration Missions (ARSSEM) project listed ten critical technologies for magnetic shielding using superconductors:

1. High performance intermediate temperature superconductor (ITS) and high temperature superconductor (HTS) cables, specifically  $\text{MgB}_2$  and YBCO
2. Double helix superconducting coil design and assembly
3. Cryogenically stable, light mechanics
4. Gas-based recirculating cooling systems
5. Cryocoolers operating a low temperature
6. Magnetic field flux charging devices
7. Quench protection for HTS coils

8. Large cryogenic cases
9. Superinsulation, radiation shielding, heat removal
10. Deployable superconducting coils (Battiston et al. 2011)

In addition to the above technologies, Spillantini suggested superconducting system models and validation of prototypes are also critical path items (Spillantini 2010).

One of the largest open questions with respect to magnetic shielding systems is current and force management. Superconducting materials can only carry up to a characteristic maximum current per unit area, which limits the magnetic field that can be produced with a given geometry and material selection (Musenich et al. 2018; Papini & Spillantini 2014). Magnetic fields require very high currents and produce large forces via magnetic pressure that must be managed with support structures as well (S. Shepherd & J. Shepherd 2009; Washburn et al. 2015; Westover 2014; Westover et al. 2012) These specifics were beyond the objectives of our study but should be studied in further detail for each shielding configuration presented.

Despite the relatively high critical temperature of superconducting materials such as yttrium-barium-copper-oxide (YBCO), magnesium diboride ( $\text{MgB}_2$ ), and other bismuth-strontium-calcium-copper-oxides (BSCCOs) and rare-earth-barium-copper-oxides (REBCOs), the superconducting coils will still require protective cooling systems to dissipate radiant heat incoming from the sun and from the spacecraft. Placement of solar panels in front of the coils could largely reduce solar radiation (Kervendal et al. 2009). Passive cooling systems such as the V-groove sunshield or solid hydrogen reservoir and active cooling systems such as pulse tube cryocoolers have been suggested (Bruce & Baudouy 2015; Musenich et al. 2014; Papini & Spillantini 2014) but require further study.

Additionally, most active magnetic shielding systems require a large launch mass and volume. While there is some evidence that baseline systems could be configured to launch on a single NASA Space Launch System (SLS) vehicle (Singleterry et al. 2015), further engineering will be required to determine the details of the manifesting and launch performance. Much of this mass penalty is due to the superconductor material itself, but it is also due in large part to the required support structures for managing magnetic forces, power, and thermal dissipation. Previous studies have suggested that a semi-rigid mixed material structural component along with convention cryocoolers and transformer rectifiers is the ideal combination of support structure, keeping mass in mind along with performance (Washburn et al. 2015).

There is evidence that chronic radiation exposure will degrade certain magnetic materials (Bowman 2013). Further investigation into the mechanism for damage to superconducting materials due to space radiation and the estimated degradation in performance over the course of a typical mission profile is needed in order to characterize and assess the risk.

Ultimately, the selection of a superconducting magnetic shielding configuration will require analysis of the benefits and risks of each candidate configuration. Westover suggests the parameters by which to perform an objective comparison of magnetic shielding configurations:

1. Shielding efficiency
2. Residual magnetic field within habitable volume
3. Mechanical stability/magnetic pressure on individual coil
4. Expandability
5. Peak field enhancement
6. Coil-to-coil forces
7. Magnetic forces on habitat

8. Scalability to higher magnetic fields
9. Quench Safety
10. Amount (kA\*m) of required superconductor
11. Ease of construction (Westover 2014)

Whether one of the configurations presented in this study is chosen or not will depend on many of the above factors, in addition to the shielding efficiency analysis provided here.

## **4 Conclusion**

### *4.1 Project Summary*

Currently operational space radiation shielding techniques are incompatible with NASA's radiation risk tolerances for an interplanetary mission. Many strategic options exist for mitigating space radiation risk, from advanced propulsion to reduce mission duration, to astronaut selection criteria based on genetic susceptibility to radiation damage, to advanced active and passive shielding methods.

Over the last several decades, magnetic shielding has consistently been identified as a preferable solution to passive shielding due to its ability to deflect rather than absorb or fragment the incoming radiation. Additionally, within the last two decades, the discovery of high temperature superconductors that require minimal cooling in space has caused active magnetic shielding to again rise to the top of the list of shielding options.

Several studies on active magnetic shielding have been conducted over the past 5-7 years; however, these studies have used varying models of the environment, spacecraft, shielding designs, simulation environments, and dosimetry methods. By selecting the most promising active magnetic shielding configurations and comparing them on a common platform, our project

has presented the first analysis of the relative effectiveness of different magnetic shielding options.

In creating the Monte Carlo simulations for this project, we have brought together well-established models of the space environment, likely interplanetary spacecraft designs, several proposed magnetic shielding designs, multiple representations of the human body, and standard dosimetry methods within the common framework of a well-validated simulation toolkit in GEANT4.

The results of these simulations suggest that space agencies must either tolerate higher risks due to space radiation dose or must develop advanced shielding technologies. Active magnetic shielding technology can support human interplanetary missions within current radiation dose limits provided a magnetic field of 7 T is achieved, regardless of the specific magnetic shielding configuration. With this information, space agencies can take strategic steps to manage the radiation risk to astronauts based on technological advancements and/or policy modifications, thereby facilitating interplanetary travel.

#### *4.2 Response to Hypothesis*

Our central hypothesis that active magnetic shielding reduces effective radiation dose to astronauts on a Mars flyby mission by 50% versus no shielding was confirmed in several mission and shielding scenarios studied. In general, our hypothesis is confirmed at a 95% confidence interval (CI) for magnetic shielding scenarios involving fields of 7 T and for scenarios involving magnetic fields of 3 T and 1.5 T under certain conditions. The conditions for confirmation of our hypothesis are listed below.

At periods of *solar maximum* activity, effective dose delivered to the *female astronaut* was reduced by at least 50% at a 95% CI versus no shielding for the following shielding configurations. Shielding effectiveness is given as the reduction in total mission effective dose versus the spacecraft only case and the 95% CI values are given in brackets ([ ]).

- Toroid 3 T: 64.48% [63.11%, 65.86%]
- Toroid 7 T: 71.12% [69.95%, 72.29%]
- Solenoid 1.5 T: 57.16% [54.22%, 60.10%]
- Solenoid 3 T: 64.30% [61.24%, 67.35%]
- Solenoid 7 T: 78.5% [77.48%, 79.71%]
- Race Track 1.5 T: 54.95% [52.84%, 57.07%]
- Race Track 3 T: 64.47% [62.90%, 66.04%]
- Race Track 7 T: 71.72% [70.50%, 72.93%]

At periods of *solar minimum* activity, effective dose delivered to the *female astronaut* was reduced by at least 50% at a 95% CI versus no shielding for the following shielding configurations. Because the total effective dose in the unshielded control case was lower by a factor of two in the solar minimum scenarios due to dramatically lower SPE dose, the effective dose reduction at 1.5 – 3 T did not reach 50%. However, the GCR effective dose reduction in these scenarios was similar to that of the solar maximum cases (~15% with 1.5 T, ~30% with 3 T, ~55% with 7 T). Shielding effectiveness is given as the reduction in total mission effective dose versus the spacecraft only case and the 95% CI values are given in brackets ([ ]).

- Toroid 7 T: 53.69% [52.23%, 55.15%]
- Solenoid 7 T: 68.02% [67.07%, 68.96%]
- Race Track 7 T: 54.73% [53.18%, 56.27%]

At periods of *solar maximum* activity, effective dose delivered to the *male astronaut* was reduced by at least 50% at a 95% CI versus no shielding for the following shielding configurations. Shielding effectiveness is given as the reduction in total mission effective dose versus the spacecraft only case and the 95% CI values are given in brackets ([]).

- Toroid 3 T: 52.52% [50.24%, 54.61%]
- Toroid 7 T: 63.81% [62.20%, 65.41%]
- Solenoid 1.5 T: 54.03% [50.81%, 57.26%]
- Solenoid 3 T: 59.66% [57.01%, 62.31%]
- Solenoid 7 T: 77.20% [75.74%, 78.66%]
- Race Track 1.5 T: 54.40% [52.49%, 56.31%]
- Race Track 3 T: 54.94% [52.06%, 57.82%]
- Race Track 7 T: 71.49% [70.31%, 72.67%]

At periods of *solar minimum* activity, effective dose delivered to the *male astronaut* was reduced by at least 50% at a 95% CI versus no shielding for the following shielding configurations. Because the total effective dose in the unshielded control case was lower by a factor of two in the solar minimum scenarios due to dramatically lower SPE dose, the effective dose reduction at 1.5 to 3 T did not reach 50%. However, the GCR effective dose reduction in these scenarios was similar to that of the solar maximum cases (~15% with 1.5 T, ~30% with 3 T, ~55% with 7 T). This scenario was also the only case where Toroid 7 T shielding did not confirm the hypothesis (48.75% [46.61%, 50.90%]). Shielding effectiveness is given as the reduction in total mission effective dose versus the spacecraft only case and the 95% CI values are given in brackets ([]).

- Solenoid 7 T: 56.31% [53.15%, 59.47%]
- Race Track 7 T: 56.24% [54.14%, 58.35%]

#### *4.3 Significance*

The results of this project inform the human spaceflight community on the utility of magnetic shielding as compared to passive or minimal shielding, based upon an end-to-end model. Effective dose versus astronaut age and sex, magnetic field, and shielding configuration provide recommendations on future investment in magnetic shielding technology and addressing engineering challenges.

The results of the risk of exposure-induced death (REID) and safe days in space (SDS) analysis in this project extend beyond the magnetic shielding application to overall interplanetary exploration questions. Using our data on REID, space policy executives at governmental agencies and commercial spaceflight firms can make informed decisions to improve the probability of mission success such as investing in the development of advanced propulsion to shorten mission duration or investing in the development of active magnetic shielding to allow deeper exploration of the solar system. Based on our SDS data, mission designers can make informed decisions at the system and sub-system levels regarding mission destination, trajectory, duration, and objectives. In general, leaders in the human spaceflight industry can use our data to tailor the type, complexity, and cost of radiation shielding required for a specific mission.

Further, our results indicate that the choice of magnetic shielding configuration and geometry does not have a significant impact on shielding effectiveness given that the magnetic field strength is high. This unexpected finding has the benefit of creating an open trade space for engineering solutions for magnetic shielding. The selection of a specific magnetic shielding

design can thus be based on factors such as logistics, cost, power, and other feasibility questions given a strong magnetic field can be achieved.

Finally, our results indicate that active bone marrow is the organ at highest relative risk compared to the other organs studied in this project. This information can allow biomedical scientists in the human spaceflight community to focus prevention and countermeasure development on this specific organ.

#### *4.4 Future Work*

Future work on this topic is warranted to reduce uncertainty in simulation data. This can be accomplished with larger computing power, allowing for a higher number of particles to be initiated per simulation. This will induce a higher number of hits per organ volume, particularly in the high magnetic field cases where fewer of the incident particles reach the habitable volume and astronaut. With a higher number of hits per organ volume, statistical uncertainty calculated within the simulations and propagated through the effective dose calculations may be decreased. Baseline simulations involving minimal shielding may also be repeated with a larger number of particles to confirm the control cases and reduce uncertainty in comparisons. By reducing uncertainty in the simulation data, possible differences in the effectiveness of different magnetic shielding configuration may be investigated.

Further studies should be conducted to advance the technological maturity of magnetic shielding. Specifically, a detailed engineering analysis is needed including evaluation of launch mass and volume, on-orbit assembly, electrical power, and cryocooling. Further, there is a need for prototyping of superconducting magnetic shields in relevant laboratory and space-based environments. These studies should include Earth-based experimental validation of simulation results via beam experiments and in-space testing of prototype magnetic shields.

## Appendix A1: GEANT4 GCR Source Macro (Solar Max, Z=1)

(/macro/source/shoot\_Z\_01\_max.mac)

```
# K.Ferrone 2019-04-26 (modified from F.Guan)
# H1 (proton), solar max
/gps/particle proton
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Epectrum_Z_01_max.dat
/gps/hist/inter Lin
```

## Appendix A2: GEANT4 GCR Source Macro (Solar Max, Z=2)

(/macro/source/shoot\_Z\_02\_max.mac)

```
# K.Ferrone 2019-04-26
# He4 (alpha), solar max
/gps/particle alpha
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Epectrum_Z_02_max.dat
/gps/hist/inter Lin
```

## Appendix A3: GEANT4 GCR Source Macro (Solar Max, Z=3 to Z=26)

(/macro/source/shoot\_Z\_ALL\_EXCEPT\_H\_He\_max.mac)

```
# K.Ferrone 2019-04-26
# All GCR (EXCEPT He and He)

# Li7, solar max
/gps/source/intensity 0.04458
/gps/particle ion
/gps/ion 3 7 3 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_03_max.dat
/gps/hist/inter Lin

# Be9, solar max
/gps/source/add 0.03699
/gps/particle ion
/gps/ion 4 9 4 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_04_max.dat
/gps/hist/inter Lin

# B11, solar max
/gps/source/add 0.09070
/gps/particle ion
/gps/ion 5 11 5 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_05_max.dat
/gps/hist/inter Lin

# C12, solar max
/gps/source/add 0.2546
/gps/particle ion
/gps/ion 6 12 6 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_06_max.dat
/gps/hist/inter Lin

# N14, solar max
/gps/source/add 0.08658
```

```

/gps/particle ion
/gps/ion 7 14 7 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_07_max.dat
/gps/hist/inter Lin

```

```

# 016, solar max
/gps/source/add 0.2600
/gps/particle ion
/gps/ion 8 16 8 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_08_max.dat
/gps/hist/inter Lin

```

```

# F19, solar max
/gps/source/add 0.007783
/gps/particle ion
/gps/ion 9 19 9 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_09_max.dat
/gps/hist/inter Lin

```

```

# Ne20, solar max
/gps/source/add 0.04313
/gps/particle ion
/gps/ion 10 20 10 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_10_max.dat
/gps/hist/inter Lin

```

```

# Na23, solar max
/gps/source/add 0.007974
/gps/particle ion
/gps/ion 11 23 11 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb

```

```

/gps/hist/file ./macro/source/GCR/Espectrum_Z_11_max.dat
/gps/hist/inter Lin

# Mg24, solar max
/gps/source/add 0.04822
/gps/particle ion
/gps/ion 12 24 12 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_12_max.dat
/gps/hist/inter Lin

# Al27, solar max
/gps/source/add 0.009727
/gps/particle ion
/gps/ion 13 27 13 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_13_max.dat
/gps/hist/inter Lin

# Si28, solar max
/gps/source/add 0.03660
/gps/particle ion
/gps/ion 14 28 14 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_14_max.dat
/gps/hist/inter Lin

# P31, solar max
/gps/source/add 0.003331
/gps/particle ion
/gps/ion 15 31 15 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_15_max.dat
/gps/hist/inter Lin

# S32, solar max
/gps/source/add 0.009199
/gps/particle ion
/gps/ion 16 32 16 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm

```

```
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_16_max.dat
/gps/hist/inter Lin
```

```
# Cl35, solar max
/gps/source/add 0.003312
/gps/particle ion
/gps/ion 17 35 17 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_17_max.dat
/gps/hist/inter Lin
```

```
# Ar40, solar max
/gps/source/add 0.004748
/gps/particle ion
/gps/ion 18 40 18 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_18_max.dat
/gps/hist/inter Lin
```

```
# K39, solar max
/gps/source/add 0.004369
/gps/particle ion
/gps/ion 19 39 19 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_19_max.dat
/gps/hist/inter Lin
```

```
# Ca40, solar max
/gps/source/add 0.005928
/gps/particle ion
/gps/ion 20 40 20 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_20_max.dat
/gps/hist/inter Lin
```

```
# Sc45, solar max
/gps/source/add 0.002053
```

```

/gps/particle ion
/gps/ion 21 45 21 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_21_max.dat
/gps/hist/inter Lin

```

```

# Ti48, solar max
/gps/source/add 0.005442
/gps/particle ion
/gps/ion 22 48 22 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_22_max.dat
/gps/hist/inter Lin

```

```

# V51, solar max
/gps/source/add 0.002698
/gps/particle ion
/gps/ion 23 51 23 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_23_max.dat
/gps/hist/inter Lin

```

```

# Cr52, solar max
/gps/source/add 0.005030
/gps/particle ion
/gps/ion 24 52 24 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_24_max.dat
/gps/hist/inter Lin

```

```

# Mn55, solar max
/gps/source/add 0.002944
/gps/particle ion
/gps/ion 25 55 25 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb

```

```
/gps/hist/file ./macro/source/GCR/Espectrum_Z_25_max.dat
/gps/hist/inter Lin

# Fe56, solar max
/gps/source/add 0.02399
/gps/particle ion
/gps/ion 26 56 26 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_26_max.dat
/gps/hist/inter Lin
```

## Appendix A4: GEANT4 GCR Source Macro (Solar Min, Z=1)

(/macro/source/shoot\_Z\_01\_min.mac)

```
# K.Ferrone 2019-04-26 (modified from F. Guan)
# H1 (proton), solar min
/gps/particle proton
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_01_min.dat
/gps/hist/inter Lin
```

## Appendix A5: GEANT4 GCR Source Macro (Solar Min, Z=2)

(/macro/source/shoot\_Z\_02\_min.mac)

```
# K.Ferrone 2019-04-26
# He4 (alpha), solar min
/gps/particle alpha
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_02_min.dat
/gps/hist/inter Lin
```

## Appendix A6: GEANT4 GCR Source Macro (Solar Min, Z=3 to Z=26)

(/macro/source/shoot\_Z\_ALL\_EXCEPT\_H\_He\_min.mac)

```
# K.Ferrone 2019-04-26
# All GCR (EXCEPT H and He)

# Li7, solar min
/gps/source/intensity 0.04537
/gps/particle ion
/gps/ion 3 7 3 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_03_min.dat
/gps/hist/inter Lin

# Be9, solar min
/gps/source/add 0.03719
/gps/particle ion
/gps/ion 4 9 4 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_04_min.dat
/gps/hist/inter Lin

# B11, solar min
/gps/source/add 0.09144
/gps/particle ion
/gps/ion 5 11 5 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_05_min.dat
/gps/hist/inter Lin

#C12, solar min
/gps/source/add 0.2536
/gps/particle ion
/gps/ion 6 12 6 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_06_min.dat
/gps/hist/inter Lin

# N14, solar min
/gps/source/add 0.08710
```

```

/gps/particle ion
/gps/ion 7 14 7 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_07_min.dat
/gps/hist/inter Lin

```

```

#016, solar min
/gps/source/add 0.2592
/gps/particle ion
/gps/ion 8 16 8 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_08_min.dat
/gps/hist/inter Lin

```

```

# F19, solar min
/gps/source/add 0.007874
/gps/particle ion
/gps/ion 9 19 9 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_09_min.dat
/gps/hist/inter Lin

```

```

# Ne20, solar min
/gps/source/add 0.04304
/gps/particle ion
/gps/ion 10 20 10 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_10_min.dat
/gps/hist/inter Lin

```

```

# Na23, solar min
/gps/source/add 0.007950
/gps/particle ion
/gps/ion 11 23 11 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb

```

```

/gps/hist/file ./macro/source/GCR/Espectrum_Z_11_min.dat
/gps/hist/inter Lin

# Mg24, solar min
/gps/source/add 0.04802
/gps/particle ion
/gps/ion 12 24 12 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_12_min.dat
/gps/hist/inter Lin

# Al27, solar min
/gps/source/add 0.009706
/gps/particle ion
/gps/ion 13 27 13 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_13_min.dat
/gps/hist/inter Lin

# Si28, solar min
/gps/source/add 0.03638
/gps/particle ion
/gps/ion 14 28 14 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_14_min.dat
/gps/hist/inter Lin

# P31, solar min
/gps/source/add 0.003381
/gps/particle ion
/gps/ion 15 31 15 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_15_min.dat
/gps/hist/inter Lin

# S32, solar min
/gps/source/add 0.009161
/gps/particle ion
/gps/ion 16 32 16 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm

```

```
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_16_min.dat
/gps/hist/inter Lin
```

```
# Cl35, solar min
/gps/source/add 0.003359
/gps/particle ion
/gps/ion 17 35 17 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_17_min.dat
/gps/hist/inter Lin
```

```
# Ar40, solar min
/gps/source/add 0.004767
/gps/particle ion
/gps/ion 18 40 18 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_18_min.dat
/gps/hist/inter Lin
```

```
# K39, solar min
/gps/source/add 0.004430
/gps/particle ion
/gps/ion 19 39 19 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_19_min.dat
/gps/hist/inter Lin
```

```
# Ca40, solar min
/gps/source/add 0.005915
/gps/particle ion
/gps/ion 20 40 20 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_20_min.dat
/gps/hist/inter Lin
```

```
# Sc45, solar min
/gps/source/add 0.002083
```

```

/gps/particle ion
/gps/ion 21 45 21 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_21_min.dat
/gps/hist/inter Lin

```

```

# Ti48, solar min
/gps/source/add 0.005487
/gps/particle ion
/gps/ion 22 48 22 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_22_min.dat
/gps/hist/inter Lin

```

```

# V51, solar min
/gps/source/add 0.002720
/gps/particle ion
/gps/ion 23 51 23 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_23_min.dat
/gps/hist/inter Lin

```

```

# Cr52, solar min
/gps/source/add 0.005050
/gps/particle ion
/gps/ion 24 52 24 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_24_min.dat
/gps/hist/inter Lin

```

```

# Mn55, solar min
/gps/source/add 0.002944
/gps/particle ion
/gps/ion 25 55 25 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb

```

```
/gps/hist/file ./macro/source/GCR/Espectrum_Z_25_min.dat
/gps/hist/inter Lin

# Fe56, solar min
/gps/source/add 0.02380
/gps/particle ion
/gps/ion 26 56 26 0
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre 0 0 0 mm
/gps/pos/radius 40 m
/gps/ang/type iso
/gps/ang/mintheta 0 deg
/gps/ang/maxtheta 5 deg
/gps/ene/type Arb
/gps/hist/file ./macro/source/GCR/Espectrum_Z_26_min.dat
/gps/hist/inter Lin
```

## Appendix A7: GEANT4 Detector Construction Code

(src/DetectorConstruction.cc)

```
// K.Ferrone, 2019-06-01 (modified from F. Guan, S. Guatelli, M.G. Pia, & F. Ambroglini)
// define global maps for MIRDO organ or tissue in UserConstant.hh
// these maps are used to store organ name, ID, mass, volume, density, Edep, and dose

#include "DetectorConstruction.hh"

// geant4
#include "globals.hh"
#include "G4GeometryManager.hh"
#include "G4VisAttributes.hh"
#include "G4SDManager.hh"
#include "G4UserLimits.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4SDParticleFilter.hh"
#include "G4TransportationManager.hh"
#include "G4UIcommand.hh"
#include "G4VSensitiveDetector.hh"
#include "G4RunManager.hh"

// materials
#include "G4Material.hh"
#include "G4MaterialTable.hh"
#include "G4NistManager.hh"

// geometry
#include "G4Box.hh"
#include "G4Colour.hh"
#include "G4Cons.hh"
#include "G4Ellipsoid.hh"
#include "G4EllipticalTube.hh"
#include "G4IntersectionSolid.hh"
#include "G4LogicalVolume.hh"
#include "G4PVParameterised.hh"
#include "G4ProductionCuts.hh"
#include "G4RegionStore.hh"
#include "G4PVPlacement.hh"
#include "G4PVReplica.hh"
#include "G4Region.hh"
#include "G4RotationMatrix.hh"
#include "G4Sphere.hh"
#include "G4SubtractionSolid.hh"
#include "G4Torus.hh"
#include "G4Trap.hh"
#include "G4Tubs.hh"
#include "G4Types.hh"
#include "G4UnionSolid.hh"
#include "G4VPhysicalVolume.hh"
#include "G4VSolid.hh"

// magnetic field
#include "G4AutoDelete.hh"
#include "G4GlobalMagFieldMessenger.hh"
#include "G4UniformMagField.hh"
#include "G4MagneticField.hh"
#include "ConfinedMagneticField.hh"
#include "FieldSetup.hh"

// constants
#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"
#include "UserConstant.hh"
#include "G4UserLimits.hh"
#include "G4ios.hh"
```

```

// energy deposited
#include "EdepCalculationInMIRD.hh"
#include "G4LogicalVolumeStore.hh"

DetectorConstruction::DetectorConstruction()
:WorldLogical(NULL),WorldPhysical(NULL),

// spacecraft components
MylarBlanketLogical(NULL), MylarBlanketPhysical(NULL),
DacronBlanketLogical(NULL), DacronBlanketPhysical(NULL),
KevlarBlanketLogical(NULL), KevlarBlanketPhysical(NULL),
AluminumShellLogical(NULL), AluminumShellPhysical(NULL),
SpacecraftEndCapMylar1Logical(NULL), SpacecraftEndCapMylar1Physical(NULL),
SpacecraftEndCapMylar2Logical(NULL), SpacecraftEndCapMylar2Physical(NULL),
SpacecraftEndCapDacron1Logical(NULL), SpacecraftEndCapDacron1Physical(NULL),
SpacecraftEndCapDacron2Logical(NULL), SpacecraftEndCapDacron2Physical(NULL),
SpacecraftEndCapKevlar1Logical(NULL), SpacecraftEndCapKevlar1Physical(NULL),
SpacecraftEndCapKevlar2Logical(NULL), SpacecraftEndCapKevlar2Physical(NULL),
SpacecraftEndCapAl1Logical(NULL), SpacecraftEndCapAl1Physical(NULL),
SpacecraftEndCapAl2Logical(NULL), SpacecraftEndCapAl2Physical(NULL),
SpacecraftAirLogical(NULL), SpacecraftAirPhysical(NULL),
OrionHatchLogical(NULL), OrionHatchPhysical(NULL),
AirCylinderLogical(NULL), AirCylinderPhysical(NULL),
OrionMainLogical(NULL), OrionMainPhysical(NULL),
OrionTopLogical(NULL), OrionTopPhysical(NULL),
OrionBottomLogical(NULL), OrionBottomPhysical(NULL),
OrionAirLogical(NULL), OrionAirPhysical(NULL),
OrionHeatShieldLogical(NULL), OrionHeatShieldPhysical(NULL),

// water shielding configuration
WaterWallShieldLogical(NULL), WaterWallShieldPhysical(NULL),

// 2.5m toroid shielding configuration
ShieldingToroidLogical(NULL),
ShieldingToroid1Physical(NULL), ShieldingToroid2Physical(NULL),
ShieldingToroid3Physical(NULL), ShieldingToroid4Physical(NULL),

ShieldingToroidInteriorLogical(NULL),
ShieldingToroidInterior1Physical(NULL), ShieldingToroidInterior2Physical(NULL),
ShieldingToroidInterior3Physical(NULL), ShieldingToroidInterior4Physical(NULL),

// NIAC 6+1 solenoid shielding configuration
ShieldingSolenoidLogical(NULL),
ShieldingSolenoid1Physical(NULL), ShieldingSolenoid2Physical(NULL),
ShieldingSolenoid3Physical(NULL), ShieldingSolenoid4Physical(NULL),
ShieldingSolenoid5Physical(NULL), ShieldingSolenoid6Physical(NULL),

ShieldingSolenoidInteriorLogical(NULL),
ShieldingSolenoidInterior1Physical(NULL), ShieldingSolenoidInterior2Physical(NULL),
ShieldingSolenoidInterior3Physical(NULL), ShieldingSolenoidInterior4Physical(NULL),
ShieldingSolenoidInterior5Physical(NULL), ShieldingSolenoidInterior6Physical(NULL),

ShieldingSolenoidSupportCylinderLogical(NULL), ShieldingSolenoidSupportCylinderPhysical(NULL),

ShieldingSolenoidSupportPlateLogical(NULL),
ShieldingSolenoidSupportPlate1Physical(NULL), ShieldingSolenoidSupportPlate2Physical(NULL),
ShieldingSolenoidSupportPlate3Physical(NULL), ShieldingSolenoidSupportPlate4Physical(NULL),
ShieldingSolenoidSupportPlate5Physical(NULL), ShieldingSolenoidSupportPlate6Physical(NULL),

CorrectionSolenoidLogical(NULL), CorrectionSolenoidPhysical(NULL),

// ESA racetrack configuration
InnerShieldingCylinderLogical(NULL),
InnerShieldingCylinder1Physical(NULL), InnerShieldingCylinder2Physical(NULL),
InnerShieldingCylinder3Physical(NULL), InnerShieldingCylinder4Physical(NULL),
InnerShieldingCylinder5Physical(NULL), InnerShieldingCylinder6Physical(NULL),
InnerShieldingCylinder7Physical(NULL), InnerShieldingCylinder8Physical(NULL),
InnerShieldingCylinder9Physical(NULL), InnerShieldingCylinder10Physical(NULL),
InnerShieldingCylinder11Physical(NULL), InnerShieldingCylinder12Physical(NULL),

```







```

LowerShieldingTorus1aPhysical(NULL), LowerShieldingTorus2aPhysical(NULL),
LowerShieldingTorus3aPhysical(NULL), LowerShieldingTorus4aPhysical(NULL),
LowerShieldingTorus5aPhysical(NULL), LowerShieldingTorus6aPhysical(NULL),
LowerShieldingTorus7aPhysical(NULL), LowerShieldingTorus8aPhysical(NULL),
LowerShieldingTorus9aPhysical(NULL), LowerShieldingTorus10aPhysical(NULL),
LowerShieldingTorus11aPhysical(NULL), LowerShieldingTorus12aPhysical(NULL),
LowerShieldingTorus13aPhysical(NULL), LowerShieldingTorus14aPhysical(NULL),
LowerShieldingTorus15aPhysical(NULL), LowerShieldingTorus16aPhysical(NULL),
LowerShieldingTorus17aPhysical(NULL), LowerShieldingTorus18aPhysical(NULL),
LowerShieldingTorus19aPhysical(NULL), LowerShieldingTorus20aPhysical(NULL),
LowerShieldingTorus21aPhysical(NULL), LowerShieldingTorus22aPhysical(NULL),
LowerShieldingTorus23aPhysical(NULL), LowerShieldingTorus24aPhysical(NULL),
LowerShieldingTorus25aPhysical(NULL), LowerShieldingTorus26aPhysical(NULL),
LowerShieldingTorus27aPhysical(NULL), LowerShieldingTorus28aPhysical(NULL),
LowerShieldingTorus29aPhysical(NULL), LowerShieldingTorus30aPhysical(NULL),
LowerShieldingTorus31aPhysical(NULL), LowerShieldingTorus32aPhysical(NULL),
LowerShieldingTorus33aPhysical(NULL), LowerShieldingTorus34aPhysical(NULL),
LowerShieldingTorus35aPhysical(NULL), LowerShieldingTorus36aPhysical(NULL),
LowerShieldingTorus37aPhysical(NULL), LowerShieldingTorus38aPhysical(NULL),
LowerShieldingTorus39aPhysical(NULL), LowerShieldingTorus40aPhysical(NULL),
LowerShieldingTorus41aPhysical(NULL), LowerShieldingTorus42aPhysical(NULL),
LowerShieldingTorus43aPhysical(NULL), LowerShieldingTorus44aPhysical(NULL),
LowerShieldingTorus45aPhysical(NULL), LowerShieldingTorus46aPhysical(NULL),
LowerShieldingTorus47aPhysical(NULL), LowerShieldingTorus48aPhysical(NULL),
LowerShieldingTorus49aPhysical(NULL), LowerShieldingTorus50aPhysical(NULL),
LowerShieldingTorus51aPhysical(NULL), LowerShieldingTorus52aPhysical(NULL),
LowerShieldingTorus53aPhysical(NULL), LowerShieldingTorus54aPhysical(NULL),
LowerShieldingTorus55aPhysical(NULL), LowerShieldingTorus56aPhysical(NULL),
LowerShieldingTorus57aPhysical(NULL), LowerShieldingTorus58aPhysical(NULL),
LowerShieldingTorus59aPhysical(NULL), LowerShieldingTorus60aPhysical(NULL),

ShieldingInteriorLogical(NULL), ShieldingInteriorPhysical(NULL),
FormerCylinderLogical(NULL), FormerCylinderPhysical(NULL),

// water phantom
WaterPhantomLogical(NULL), WaterPhantomPhysical(NULL),

// MIRD phantom organs (plus eye lens, active bone marrow, bone, muscle, and skin volumes)
MIRDMotherLogical(NULL), MIRDMotherPhysical(NULL),
HeadLogical(NULL), HeadPhysical(NULL),
CraniumLogical(NULL), CraniumPhysical(NULL),
CraniumMarrowLogical(NULL), CraniumMarrowPhysical(NULL),
LeftEyeLensLogical(NULL), LeftEyeLensPhysical(NULL),
RightEyeLensLogical(NULL), RightEyeLensPhysical(NULL),
BrainLogical(NULL), BrainPhysical(NULL),
UpperSpineLogical(NULL), UpperSpinePhysical(NULL),
UpperSpineMarrowLogical(NULL), UpperSpineMarrowPhysical(NULL),
ThyroidLogical(NULL), ThyroidPhysical(NULL),
TrunkLogical(NULL), TrunkPhysical(NULL),
HeartLogical(NULL), HeartPhysical(NULL),
LeftLungLogical(NULL), LeftLungPhysical(NULL),
RightLungLogical(NULL), RightLungPhysical(NULL),
LeftKidneyLogical(NULL), LeftKidneyPhysical(NULL),
RightKidneyLogical(NULL), RightKidneyPhysical(NULL),
LeftAdrenalLogical(NULL), LeftAdrenalPhysical(NULL),
RightAdrenalLogical(NULL), RightAdrenalPhysical(NULL),
RibCageLogical(NULL), RibCagePhysical(NULL),
Rib1Logical(NULL), Rib1Physical(NULL),
Rib1MarrowLogical(NULL), Rib1MarrowPhysical(NULL),
Rib2Logical(NULL), Rib2Physical(NULL),
Rib2MarrowLogical(NULL), Rib2MarrowPhysical(NULL),
Rib3Logical(NULL), Rib3Physical(NULL),
Rib3MarrowLogical(NULL), Rib3MarrowPhysical(NULL),
Rib4Logical(NULL), Rib4Physical(NULL),
Rib4MarrowLogical(NULL), Rib4MarrowPhysical(NULL),
Rib5Logical(NULL), Rib5Physical(NULL),
Rib5MarrowLogical(NULL), Rib5MarrowPhysical(NULL),
Rib6Logical(NULL), Rib6Physical(NULL),
Rib6MarrowLogical(NULL), Rib6MarrowPhysical(NULL),
Rib7Logical(NULL), Rib7Physical(NULL),

```

```

Rib7MarrowLogical(NULL), Rib7MarrowPhysical(NULL),
Rib8Logical(NULL), Rib8Physical(NULL),
Rib8MarrowLogical(NULL), Rib8MarrowPhysical(NULL),
Rib9Logical(NULL), Rib9Physical(NULL),
Rib9MarrowLogical(NULL), Rib9MarrowPhysical(NULL),
Rib10Logical(NULL), Rib10Physical(NULL),
Rib10MarrowLogical(NULL), Rib10MarrowPhysical(NULL),
Rib11Logical(NULL), Rib11Physical(NULL),
Rib11MarrowLogical(NULL), Rib11MarrowPhysical(NULL),
Rib12Logical(NULL), Rib12Physical(NULL),
Rib12MarrowLogical(NULL), Rib12MarrowPhysical(NULL),
LeftClavicleLogical(NULL), LeftClaviclePhysical(NULL),
LeftClavicleMarrowLogical(NULL), LeftClavicleMarrowPhysical(NULL),
RightClavicleLogical(NULL), RightClaviclePhysical(NULL),
RightClavicleMarrowLogical(NULL), RightClavicleMarrowPhysical(NULL),
LeftScapulaLogical(NULL), LeftScapulaPhysical(NULL),
LeftScapulaMarrowLogical(NULL), LeftScapulaMarrowPhysical(NULL),
RightScapulaLogical(NULL), RightScapulaPhysical(NULL),
RightScapulaMarrowLogical(NULL), RightScapulaMarrowPhysical(NULL),
SmallIntestineLogical(NULL), SmallIntestinePhysical(NULL),
LowerLargeIntestineLogical(NULL), LowerLargeIntestinePhysical(NULL),
UpperLargeIntestineLogical(NULL), UpperLargeIntestinePhysical(NULL),
LiverLogical(NULL), LiverPhysical(NULL),
StomachLogical(NULL), StomachPhysical(NULL),
PancreasLogical(NULL), PancreasPhysical(NULL),
SpleenLogical(NULL), SpleenPhysical(NULL),
BladderLogical(NULL), BladderPhysical(NULL),
ThymusLogical(NULL), ThymusPhysical(NULL),
MiddleLowerSpineLogical(NULL), MiddleLowerSpinePhysical(NULL),
MiddleLowerSpineMarrowLogical(NULL), MiddleLowerSpineMarrowPhysical(NULL),
PelvisLogical(NULL), PelvisPhysical(NULL),
PelvisMarrowLogical(NULL), PelvisMarrowPhysical(NULL),
LeftLegLogical(NULL), LeftLegPhysical(NULL),
RightLegLogical(NULL), RightLegPhysical(NULL),
LeftLegBoneLogical(NULL), LeftLegBonePhysical(NULL),
LeftLegBoneMarrowLogical(NULL), LeftLegBoneMarrowPhysical(NULL),
RightLegBoneLogical(NULL), RightLegBonePhysical(NULL),
RightLegBoneMarrowLogical(NULL), RightLegBoneMarrowPhysical(NULL),
LeftArmBoneLogical(NULL), LeftArmBonePhysical(NULL),
LeftArmBoneMarrowLogical(NULL), LeftArmBoneMarrowPhysical(NULL),
RightArmBoneLogical(NULL), RightArmBonePhysical(NULL),
RightArmBoneMarrowLogical(NULL), RightArmBoneMarrowPhysical(NULL),

// MIRD female organs
UterusLogical(NULL), UterusPhysical(NULL),
LeftOvaryLogical(NULL), LeftOvaryPhysical(NULL),
RightOvaryLogical(NULL), RightOvaryPhysical(NULL),
LeftBreastLogical(NULL), LeftBreastPhysical(NULL),
RightBreastLogical(NULL), RightBreastPhysical(NULL),
// MIRD male organs
MaleGenitaliaLogical(NULL), MaleGenitaliaPhysical(NULL),
LeftTesticleLogical(NULL), LeftTesticlePhysical(NULL),
RightTesticleLogical(NULL), RightTesticlePhysical(NULL)

{
//
}

DetectorConstruction::~DetectorConstruction()
{
//
}

G4VPhysicalVolume * DetectorConstruction::Construct()
{
// materials definition
DefineMaterials();

// geometry definition
SetupGeometry();

```

```

    // return world volume
    return WorldPhysical;
}

void DetectorConstruction::ConstructSDandField()
{
    // define scorer and detector for water phantom or MIRD organs
    G4SDManager::GetSDMpointer()->SetVerboseLevel(1);

    if(phantomName=="water_phantom")
    {
        EdepCalculationInWater *EdepScorer = new EdepCalculationInWater("EdepScorerInWater",
                                                                           gDetectorHalfX, gDetectorHalfY, gDetectorHalfZ,
                                                                           gdX_pixel, gdY_pixel, gdZ_pixel);

        G4String detectorName = "EdepDetectorInWater";

        SetupScoring(WaterPhantomLogical,detectorName,EdepScorer);

        detectorNameVector.push_back(detectorName);
    }

    if(phantomName=="MIRD_female" || phantomName=="MIRD_male")
    {
        G4LogicalVolumeStore* logicalVolumeVector = G4LogicalVolumeStore::GetInstance();
        // define scorer and detector for each organ of interest
        for(std::map<G4String, G4int>::const_iterator it = Map_OrganNameID.begin();
            it!=Map_OrganNameID.end(); ++it)//
        {
            G4String organName = it->first;
            G4String organLogicalName = organName+"_Logical";

            EdepCalculationInMIRD *EdepScorer = new
            EdepCalculationInMIRD("EdepScorerInMIRD_"+organName);

            G4String detectorName = "EdepDetectorInMIRD_"+organName;

            G4LogicalVolume* organLogical = logicalVolumeVector->GetVolume(organLogicalName);

            SetupScoring(organLogical,detectorName,EdepScorer);

            detectorNameVector.push_back(detectorName);
        }
    }
}

// _____//

// materials definition

void DetectorConstruction::DefineMaterials()
{
    G4String name, symbol;
    G4double z, a, density;
    G4int natoms, ncomponents;

    // define elements

    // elements for YBCO superconductor
    G4Element *elY = new G4Element(name="Yttrium", symbol="Y", z=39.0, a=88.906*g/mole);
    G4Element *elBa = new G4Element(name="Barium", symbol="Ba", z=56.0, a=137.327*g/mole);
    G4Element *elCu = new G4Element(name="Copper", symbol="Cu", z=29.0, a=63.546*g/mole);
    G4Element *elO = new G4Element(name = "Oxygen", symbol="O", z=8.0, a=16.00*g/mole);

    // additional elements for Orion AVC0AT heat shield
    G4Element *elC = new G4Element(name="Carbon", symbol="C", z=6.0, a=12.011*g/mole);
    G4Element *elH = new G4Element(name="Hydrogen", symbol="H", z=1.0, a=1.01*g/mole);

```

```

// additional elements for spacecraft
G4Element *elAl = new G4Element(name="Aluminum", symbol="Al", z=13.0, a=26.98*g/mole);
G4Element *elLi = new G4Element(name="Lithium", symbol="Li", z=3.0, a=6.941*g/mole);
G4Element *elAg = new G4Element(name="Silver", symbol="Ag", z=47.0, a=107.87*g/mole);
G4Element *elMg = new G4Element(name="Magnesium", symbol="Mg", z=12.0, a=24.31*g/mole);

// titanium for former cylinder
G4Element *elTi = new G4Element(name="Titanium", symbol="Ti", z=22.0, a=47.867*g/mole);

// additional elements for soft tissue
G4Element *elN = new G4Element(name="Nitrogen", symbol="N", z=7.0, a=14.007*g/mole);
G4Element *elNa = new G4Element(name="Sodium", symbol="Na", z=11.0, a=22.99*g/mole);
G4Element *elP = new G4Element(name="Phosphorus", symbol="P", z=15.0, a=30.973762*g/mole);
G4Element *elS = new G4Element(name="Sulfur", symbol="S", z=16.0, a=32.065*g/mole);
G4Element *elCl = new G4Element(name="Chlorine", symbol="Cl", z=17.0, a=35.45*g/mole);
G4Element *elK = new G4Element(name="Potassium", symbol="K", z=19.0, a=39.09*g/mole);
G4Element *elFe = new G4Element(name="Iron", symbol="Fe", z=26.0, a=55.845*g/mole);
G4Element *elZn = new G4Element(name="Zinc", symbol="Zn", z=30.0, a=65.38*g/mole);
G4Element *elRb = new G4Element(name="Rubidium", symbol="Rb", z=37.0, a=85.47*g/mole);
G4Element *elSr = new G4Element(name="Strontium", symbol="Sr", z=38.0, a=87.62*g/mole);
G4Element *elZr = new G4Element(name="Zirconium", symbol="Zr", z=40.0, a=91.22*g/mole);
G4Element *elPb = new G4Element(name="Lead", symbol="Pb", z=82.0, a=207.2*g/mole);

// additional elements for bone
G4Element *elCa = new G4Element(name="Calcium", symbol="Ca", z=20.0, a=40.08*g/mole);

// define NIST materials

// water
G4Material* g4water = G4NistManager::Instance()->FindOrBuildMaterial("G4_WATER");

// air
G4Material* g4air = G4NistManager::Instance()->FindOrBuildMaterial("G4_AIR");

// vacuum
G4Material *g4galactic = G4NistManager::Instance()->FindOrBuildMaterial("G4_Galactic");

// mylar
G4Material *g4mylar = G4NistManager::Instance()->FindOrBuildMaterial("G4_MYLAR");
// density = 1.39 g/cm3

// dacron
G4Material *g4dacron = G4NistManager::Instance()->FindOrBuildMaterial("G4_DACRON");
// density = 1.39 g/cm3

// kevlar
G4Material *g4kevlar = G4NistManager::Instance()->FindOrBuildMaterial("G4_KEVLAR");
// density = 1.44 g/cm3

// graphite
G4Material *g4graphite = G4NistManager::Instance()->FindOrBuildMaterial("G4_GRAPHITE");
// density = 2.21 g/cm3

// aluminum
G4Material *g4Al = G4NistManager::Instance()->FindOrBuildMaterial("G4_Al");
// z = 13, density = 2.699 g/cm3

// titanium
G4Material *g4Ti = G4NistManager::Instance()->FindOrBuildMaterial("G4_Ti");
// z = 22, density = 4.506 g/cm3

// skeletal muscle
G4Material *g4muscle = G4NistManager::Instance()-
>FindOrBuildMaterial("G4_MUSCLE_SKELETAL_ICRP"); // density = 1.05 g/cm3

// adipose tissue
G4Material *g4adipose = G4NistManager::Instance()-
>FindOrBuildMaterial("G4_ADIPOSE_TISSUE_ICRP"); // density = 0.95 g/cm3

```

```

// blood
G4Material *g4blood = G4NistManager::Instance()->FindOrBuildMaterial("G4_BLOOD_ICRP");
// density = 1.06 g/cm3

// cortical bone
G4Material *g4corticalbone = G4NistManager::Instance()-
>FindOrBuildMaterial("G4_BONE_CORTICAL_ICRP"); // density = 1.92 g/cm3

// compact bone
G4Material *g4compactbone = G4NistManager::Instance()-
>FindOrBuildMaterial("G4_BONE_COMPACT_ICRP"); // density = 1.85 g/cm3

// brain
G4Material *g4brain = G4NistManager::Instance()->FindOrBuildMaterial("G4_BRAIN_ICRP");
// density = 1.04 g/cm3

// eye lens
G4Material *g4eyelens = G4NistManager::Instance()->FindOrBuildMaterial("G4_EYE_LENS_ICRP");
// density = 1.07 g/cm3

// lung
G4Material *g4lung = G4NistManager::Instance()->FindOrBuildMaterial("G4_LUNG_ICRP");
// density = 1.04 g/cm3 (how is ICRP lung density so high?)

// skin
G4Material *g4skin = G4NistManager::Instance()->FindOrBuildMaterial("G4_SKIN_ICRP");
// density = 1.09 g/cm3

// soft tissue
G4Material *g4soft = G4NistManager::Instance()->FindOrBuildMaterial("G4_TISSUE_SOFT_ICRP");
// density = 1.03 g/cm3

// define compounds

// YBCO superconductor material (YBa2Cu3O7)
density = 6.3*kg/cm3;
G4Material *YBCO = new G4Material(name="YBCO", density, ncomponents=4);
YBCO->AddElement(elY, natoms=1);
YBCO->AddElement(elBa, natoms=2);
YBCO->AddElement(elCu, natoms=3);
YBCO->AddElement(elO, natoms=7);

// AVCOAT 5026-39 Orion heat shield material (C47H82O10)
density = 0.51*kg/cm3;
G4Material *AVCOAT = new G4Material(name="AVCOAT", density, ncomponents=3);
AVCOAT->AddElement(elC, 33.8*perCent);
AVCOAT->AddElement(elH, 59.0*perCent);
AVCOAT->AddElement(elO, 7.2*perCent);

// aluminum-lithium alloy 2195 for Orion
density = 2.71*kg/cm3;
G4Material *Alli_2195 = new G4Material(name="Alli_2195", density, ncomponents=5);
Alli_2195->AddElement(elAl, 94.2*perCent);
Alli_2195->AddElement(elCu, 4.0*perCent);
Alli_2195->AddElement(elLi, 1.0*perCent);
Alli_2195->AddElement(elAg, 0.4*perCent);
Alli_2195->AddElement(elMg, 0.4*perCent);

// MIRD soft tissue
density = 0.9869*kg/cm3;
G4Material *soft_tissue = new G4Material(name="soft_tissue", density, ncomponents=16);
soft_tissue->AddElement(elH, 0.1047);
soft_tissue->AddElement(elC, 0.2302);
soft_tissue->AddElement(elN, 0.0234);
soft_tissue->AddElement(elO, 0.6321);
soft_tissue->AddElement(elNa, 0.0013);
soft_tissue->AddElement(elMg, 0.00015);
soft_tissue->AddElement(elP, 0.0024);
soft_tissue->AddElement(elS, 0.0022);

```

```

soft_tissue->AddElement(eICl, 0.0014);
soft_tissue->AddElement(eIK, 0.0021);
soft_tissue->AddElement(eIFe, 0.000063);
soft_tissue->AddElement(eIZn, 0.000032);
soft_tissue->AddElement(eIRb, 0.0000057);
soft_tissue->AddElement(eISr, 0.00000034);
soft_tissue->AddElement(eIZr, 0.000008);
soft_tissue->AddElement(eIPb, 0.00000016);

// MIRD skeletal bone
density = 1.4862*g/cm3;
G4Material *bone = new G4Material(name="bone", density, ncomponents=15);
bone->AddElement(eIH, 0.0704);
bone->AddElement(eIC, 0.2279);
bone->AddElement(eIN, 0.0387);
bone->AddElement(eIO, 0.4856);
bone->AddElement(eINa, 0.0032);
bone->AddElement(eIMg, 0.0011);
bone->AddElement(eIP, 0.0694);
bone->AddElement(eIS, 0.0017);
bone->AddElement(eICl, 0.0014);
bone->AddElement(eIK, 0.0015);
bone->AddElement(eICa, 0.0991);
bone->AddElement(eIFe, 0.00008);
bone->AddElement(eIZn, 0.000048);
bone->AddElement(eISr, 0.000032);
bone->AddElement(eIPb, 0.000011);

// MIRD lung tissue
density = 0.2958*g/cm3;
G4Material *lung = new G4Material(name="lung", density, ncomponents=16);
lung->AddElement(eIH, 0.1021);
lung->AddElement(eIC, 0.1001);
lung->AddElement(eIN, 0.028);
lung->AddElement(eIO, 0.7596);
lung->AddElement(eINa, 0.0019);
lung->AddElement(eIMg, 0.000074);
lung->AddElement(eIP, 0.00081);
lung->AddElement(eIS, 0.0023);
lung->AddElement(eICl, 0.0027);
lung->AddElement(eIK, 0.0020);
lung->AddElement(eICa, 0.00007);
lung->AddElement(eIFe, 0.00037);
lung->AddElement(eIZn, 0.000011);
lung->AddElement(eIRb, 0.0000037);
lung->AddElement(eISr, 0.00000059);
lung->AddElement(eIPb, 0.00000041);

// active marrow
density = 1.06*g/cm3;
G4Material *red_marrow = new G4Material(name="red_marrow", density, ncomponents=9);
red_marrow->AddElement(eIH, 0.102);
red_marrow->AddElement(eIC, 0.143);
red_marrow->AddElement(eIN, 0.034);
red_marrow->AddElement(eIO, 0.708);
red_marrow->AddElement(eINa, 0.002);
red_marrow->AddElement(eIP, 0.003);
red_marrow->AddElement(eIS, 0.003);
red_marrow->AddElement(eICl, 0.002);
red_marrow->AddElement(eIK, 0.003);

// dump material information
G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}

// _____//

// geometry definition

void DetectorConstruction::SetupGeometry()

```

```

{
    // define world
    WorldPhysical = ConstructWorld();
    WorldLogical = WorldPhysical->GetLogicalVolume();

    // define habitat components
    // define Mylar blanket, thermal reflective layer (5 mm)
    MylarBlanketPhysical = ConstructMylarBlanket(WorldLogical);
    // define Dacron blanket, thermal insulation layer (5 mm)
    DacronBlanketPhysical = ConstructDacronBlanket(WorldLogical);
    // define Kevlar blanket, thermal/MMOD protective layer (5 mm)
    KevlarBlanketPhysical = ConstructKevlarBlanket(WorldLogical);
    // define Aluminum shell (1.8 cm thick cylinder of Al)
    AluminumShellPhysical = ConstructAluminumShell(WorldLogical);
    // define Mylar end caps (5 mm thick rounded end caps of Mylar)
    SpacecraftEndCapMylar1Physical = ConstructEndCapMylar1(WorldLogical);
    SpacecraftEndCapMylar2Physical = ConstructEndCapMylar2(WorldLogical);
    // define Dacron end caps (5 mm thick rounded end caps of Dacron)
    SpacecraftEndCapDacron1Physical = ConstructEndCapDacron1(WorldLogical);
    SpacecraftEndCapDacron2Physical = ConstructEndCapDacron2(WorldLogical);
    // define Kevlar end caps (10 cm thick rounded end caps of Kevlar)
    SpacecraftEndCapKevlar1Physical = ConstructEndCapKevlar1(WorldLogical);
    SpacecraftEndCapKevlar2Physical = ConstructEndCapKevlar2(WorldLogical);
    // define Aluminum end caps (1.8 cm thick rounded end caps of Al)
    SpacecraftEndCapAl1Physical = ConstructEndCapAl1(WorldLogical);
    SpacecraftEndCapAl2Physical = ConstructEndCapAl2(WorldLogical);
    // define spacecraft air (air cylinder)
    SpacecraftAirPhysical = ConstructSpacecraftAir(WorldLogical);

    // define Orion components
    // define Orion hatch (cylinder)
    OrionHatchPhysical = ConstructOrionHatch(WorldLogical);
    // define Orion hatch air (cylinder)
    AirCylinderPhysical = ConstructAirCylinder(WorldLogical);
    // define Orion main volume
    OrionMainPhysical = ConstructOrionMain(WorldLogical);
    // define Orion top layer
    OrionTopPhysical = ConstructOrionTop(WorldLogical);
    // define Orion bottom layer
    OrionBottomPhysical = ConstructOrionBottom(WorldLogical);
    // define Orion air
    OrionAirPhysical = ConstructOrionAir(WorldLogical);
    // define Orion heat shield
    OrionHeatShieldPhysical = ConstructOrionHeatShield(WorldLogical);

    // define shielding configuration
    if(shieldingType=="H20")
    {
        // define water wall
        WaterWallShieldPhysical = ConstructWaterWallShield(SpacecraftAirLogical);
    }
    if(shieldingType=="toroid_2.5m")
    {
        // define shielding toroid magnet volumes
        ShieldingToroid1Physical = ConstructShieldingToroid(WorldLogical);

        // define shielding magnet interior volumes
        ShieldingToroidInterior1Physical = ConstructShieldingToroidInterior(WorldLogical);
    }
    if(shieldingType=="NIAC_6plus1")
    {
        // define shielding solenoid magnets
        ShieldingSolenoid1Physical = ConstructShieldingSolenoid(WorldLogical);

        // define shielding solenoid interior volumes
        ShieldingSolenoidInterior1Physical = ConstructShieldingSolenoidInterior(WorldLogical);

        // define shielding solenoid graphite support cylinders
        ShieldingSolenoidSupportCylinderPhysical =
            ConstructShieldingSolenoidSupportCylinder(ShieldingSolenoidInteriorLogical);
    }
}

```

```

    // define shielding solenoid radial graphite support plates
    ShieldingSolenoidSupportPlate1Physical =
    ConstructShieldingSolenoidSupportPlate(ShieldingSolenoidInteriorLogical);

    // define correction magnet volume
    CorrectionSolenoidPhysical = ConstructCorrectionSolenoid(WorldLogical);
}
if(shieldingType=="ESA_racetrack")
{
    // define inner shielding cylinders for racetrack
    InnerShieldingCylinder1Physical = ConstructInnerShieldingCylinder(WorldLogical);

    // define outer shielding cylinders for racetrack
    OuterShieldingCylinder1Physical = ConstructOuterShieldingCylinder(WorldLogical);

    // define upper torus volumes for racetrack
    UpperShieldingTorus1Physical = ConstructUpperShieldingTorus(WorldLogical);

    // define lower torus volumes for racetrack
    LowerShieldingTorus1Physical = ConstructLowerShieldingTorus(WorldLogical);

    // define shielding interior volume for racetrack
    ShieldingInteriorPhysical = ConstructShieldingInterior(WorldLogical);

    // define titanium former volume for racetrack
    FormerCylinderPhysical = ConstructFormerCylinder(ShieldingInteriorLogical);
}

// define water phantom
if(phantomName=="water_phantom")
{
    WaterPhantomPhysical = ConstructWaterPhantom(SpacecraftAirLogical);
}
// define MIRD phantom
if(phantomName=="MIRD_female" || phantomName=="MIRD_male")
{
    // define MIRD Mother volume
    MIRDMotherPhysical = ConstructMIRDMotherVolume(SpacecraftAirLogical);
    // define MIRD head (mother to cranium, brain, upper spine)
    HeadPhysical = ConstructMIRDHead(MIRDMotherLogical);
    // define MIRD cranium
    CraniumPhysical = ConstructMIRDCranium(HeadLogical);
    // define cranium active marrow
    CraniumMarrowPhysical = ConstructCraniumMarrow(CraniumLogical);
    // define left eye lens
    LeftEyeLensPhysical = ConstructLeftEyeLens(MIRDMotherLogical);
    // define right eye lens
    RightEyeLensPhysical = ConstructRightEyeLens(MIRDMotherLogical);
    // define MIRD brain
    BrainPhysical = ConstructMIRDBrain(HeadLogical);
    // define MIRD upper spine
    UpperSpinePhysical = ConstructMIRDUpperSpine(HeadLogical);
    // define upper spine active marrow
    UpperSpineMarrowPhysical = ConstructUpperSpineMarrow(UpperSpineLogical);
    // define MIRD thyroid
    ThyroidPhysical = ConstructMIRDThyroid(HeadLogical);
    // define MIRD trunk (mother to organs)
    TrunkPhysical = ConstructMIRDTrunk(MIRDMotherLogical);
    // define MIRD heart
    HeartPhysical = ConstructMIRDHeart(TrunkLogical);
    // define MIRD left lung
    LeftLungPhysical = ConstructMIRDLeftLung(TrunkLogical);
    // define MIRD right lung
    RightLungPhysical = ConstructMIRDRightLung(TrunkLogical);
    // define MIRD left kidney
    LeftKidneyPhysical = ConstructMIRDLeftKidney(TrunkLogical);
    // define MIRD right kidney
    RightKidneyPhysical = ConstructMIRDRightKidney(TrunkLogical);
}

```

```

// define MIRD left adrenal
LeftAdrenalPhysical = ConstructMIRDLeftAdrenal(TrunkLogical);
// define MIRD right adrenal
RightAdrenalPhysical = ConstructMIRDRightAdrenal(TrunkLogical);
// define MIRD rib cage
RibCagePhysical = ConstructMIRD RibCage(TrunkLogical);
// define MIRD rib 1
Rib1Physical = ConstructMIRD Rib1(RibCageLogical);
// define rib 1 active marrow
Rib1MarrowPhysical = ConstructRib1Marrow(Rib1Logical);
// define MIRD rib 2
Rib2Physical = ConstructMIRD Rib2(RibCageLogical);
// define rib 2 active marrow
Rib2MarrowPhysical = ConstructRib2Marrow(Rib2Logical);
// define MIRD rib 3
Rib3Physical = ConstructMIRD Rib3(RibCageLogical);
// define rib 3 active marrow
Rib3MarrowPhysical = ConstructRib3Marrow(Rib3Logical);
// define MIRD rib 4
Rib4Physical = ConstructMIRD Rib4(RibCageLogical);
// define rib 4 active marrow
Rib4MarrowPhysical = ConstructRib4Marrow(Rib4Logical);
// define MIRD rib 5
Rib5Physical = ConstructMIRD Rib5(RibCageLogical);
// define rib 5 active marrow
Rib5MarrowPhysical = ConstructRib5Marrow(Rib5Logical);
// define MIRD rib 6
Rib6Physical = ConstructMIRD Rib6(RibCageLogical);
// define rib 6 active marrow
Rib6MarrowPhysical = ConstructRib6Marrow(Rib6Logical);
// define MIRD rib 7
Rib7Physical = ConstructMIRD Rib7(RibCageLogical);
// define rib 7 active marrow
Rib7MarrowPhysical = ConstructRib7Marrow(Rib7Logical);
// define MIRD rib 8
Rib8Physical = ConstructMIRD Rib8(RibCageLogical);
// define rib 8 active marrow
Rib8MarrowPhysical = ConstructRib8Marrow(Rib8Logical);
// define MIRD rib 9
Rib9Physical = ConstructMIRD Rib9(RibCageLogical);
// define rib 9 active marrow
Rib9MarrowPhysical = ConstructRib9Marrow(Rib9Logical);
// define MIRD rib 10
Rib10Physical = ConstructMIRD Rib10(RibCageLogical);
// define rib 10 active marrow
Rib10MarrowPhysical = ConstructRib10Marrow(Rib10Logical);
// define MIRD rib 11
Rib11Physical = ConstructMIRD Rib11(RibCageLogical);
// define rib 11 active marrow
Rib11MarrowPhysical = ConstructRib11Marrow(Rib11Logical);
// define MIRD rib 12
Rib12Physical = ConstructMIRD Rib12(RibCageLogical);
// define rib 12 active marrow
Rib12MarrowPhysical = ConstructRib12Marrow(Rib12Logical);
// define MIRD left clavicle
LeftClaviclePhysical = ConstructMIRDLeftClavicle(TrunkLogical);
// define left clavicle active marrow
LeftClavicleMarrowPhysical = ConstructLeftClavicleMarrow(LeftClavicleLogical);
// define MIRD right clavicle
RightClaviclePhysical = ConstructMIRDRightClavicle(TrunkLogical);
// define right clavicle active marrow
RightClavicleMarrowPhysical = ConstructRightClavicleMarrow(RightClavicleLogical);
// define MIRD left scapula
LeftScapulaPhysical = ConstructMIRDLeftScapula(TrunkLogical);
// define left scapula active marrow
LeftScapulaMarrowPhysical = ConstructLeftScapulaMarrow(LeftScapulaLogical);
// define MIRD right scapula
RightScapulaPhysical = ConstructMIRDRightScapula(TrunkLogical);
// define right scapula active marrow
RightScapulaMarrowPhysical = ConstructRightScapulaMarrow(RightScapulaLogical);

```

```

// define MIRD small intestine
SmallIntestinePhysical = ConstructMIRDSmallIntestine(TrunkLogical);
// define MIRD lower large intestine
LowerLargeIntestinePhysical = ConstructMIRDLowerLargeIntestine(TrunkLogical);
// define MIRD upper large intestine
UpperLargeIntestinePhysical = ConstructMIRDUpperLargeIntestine(TrunkLogical);
// define MIRD liver
LiverPhysical = ConstructMIRDLiver(TrunkLogical);
// define MIRD stomach
StomachPhysical = ConstructMIRDStomach(TrunkLogical);
// define MIRD pancreas
PancreasPhysical = ConstructMIRDPancreas(TrunkLogical);
// define MIRD spleen
SpleenPhysical = ConstructMIRDSpleen(TrunkLogical);
// define MIRD bladder
BladderPhysical = ConstructMIRDBladder(TrunkLogical);
// define MIRD thymus
ThymusPhysical = ConstructMIRDThymus(TrunkLogical);
// define MIRD middle lower spine
MiddleLowerSpinePhysical = ConstructMIRDMiddleLowerSpine(TrunkLogical);
// define middle lower spine active marrow
MiddleLowerSpineMarrowPhysical = ConstructMiddleLowerSpineMarrow(MiddleLowerSpineLogical);
// define MIRD pelvis
PelvisPhysical = ConstructMIRDPelvis(TrunkLogical);
// define pelvis active marrow
PelvisMarrowPhysical = ConstructPelvisMarrow(PelvisLogical);
// define MIRD left leg
LeftLegPhysical = ConstructMIRDLeftLeg(MIRDMotherLogical);
// define MIRD right leg
RightLegPhysical = ConstructMIRDRightLeg(MIRDMotherLogical);
// define MIRD left leg bone
LeftLegBonePhysical = ConstructMIRDLeftLegBone(LeftLegLogical);
// define left leg bone active marrow
LeftLegBoneMarrowPhysical = ConstructLeftLegBoneMarrow(LeftLegBoneLogical);
// define right leg bone
RightLegBonePhysical = ConstructMIRDRightLegBone(RightLegLogical);
// define right leg bone active marrow
RightLegBoneMarrowPhysical = ConstructRightLegBoneMarrow(RightLegBoneLogical);
// define MIRD left arm bone
LeftArmBonePhysical = ConstructMIRDLeftArmBone(TrunkLogical);
// define left arm bone active marrow
LeftArmBoneMarrowPhysical = ConstructLeftArmBoneMarrow(LeftArmBoneLogical);
// define right arm bone
RightArmBonePhysical = ConstructMIRDRightArmBone(TrunkLogical);
// define right arm bone active marrow
RightArmBoneMarrowPhysical = ConstructRightArmBoneMarrow(RightArmBoneLogical);

} //MIRD female or male

if(phantomName=="MIRD_female")
{
    // MIRD female organs
    // define MIRD uterus
    UterusPhysical = ConstructMIRDUterus(TrunkLogical);
    // define MIRD left ovary
    LeftOvaryPhysical = ConstructMIRDLeftOvary(TrunkLogical);
    // define MIRD right ovary
    RightOvaryPhysical = ConstructMIRDRightOvary(TrunkLogical);
    // define MIRD left breast
    LeftBreastPhysical = ConstructMIRDLeftBreast(MIRDMotherLogical);
    // define MIRD right breast
    RightBreastPhysical = ConstructMIRDRightBreast(MIRDMotherLogical);
}
else if(phantomName=="MIRD_male")
{
    // MIRD male organs
    // define MIRD male genitalia
    MaleGenitaliaPhysical = ConstructMIRDMaleGenitalia(MIRDMotherLogical);
}

```

```

    // define MIRD left testicle
    LeftTesticlePhysical = ConstructMIRDLeftTesticle(MaleGenitaliaLogical);
    // define MIRD right testicle
    RightTesticlePhysical = ConstructMIRDRightTesticle(MaleGenitaliaLogical);
}

}

// _____//

// construct world
G4VPhysicalVolume * DetectorConstruction::ConstructWorld()
{
    // get materials
    G4Material *g4galactic = G4Material::GetMaterial("G4_Galactic");

    if(shieldingType=="NIAC_6plus1") {

        // need a cube world volume for tabulated field
        G4Box *World_Solid = new G4Box("World_Solid",    // name
                                       40.*m,             // half x
                                       40.*m,             // half y
                                       40.*m);            // half z

        WorldLogical = new G4LogicalVolume(World_Solid,    // solid volume
                                           g4galactic,      // material
                                           "World_Logical"); // name

        // define and apply tabulated magnetic field
        fFieldSetup3D = new FieldSetup();
        WorldLogical->SetFieldManager(fFieldSetup3D->GetLocalFieldManager(), true);
        // apply magnetic field manager

        // manually set max step size for world volume
        // G4double maxStep = 1.*m;
        // fStepLimit = new G4UserLimits(maxStep);
        // WorldLogical->SetUserLimits(fStepLimit);

    } else {

        // set 0 magnetic fringe field for world in all other cases besides solenoid
        G4MagneticField *global_field = new G4UniformMagField(G4ThreeVector(0,0,0));
        G4FieldManager *global_fieldManager =
        G4TransportationManager::GetTransportationManager()->GetFieldManager();
        global_fieldManager->SetDetectorField(global_field);
        global_fieldManager->CreateChordFinder(global_field);

        // spherical world volume for all other cases
        G4Sphere *World_Solid = new G4Sphere("World_Solid",    // name
                                             0,                  // inner radius
                                             40.*m,              // outer radius
                                             0,                  // start phi
                                             360.*deg,            // delta phi
                                             0,                  // start theta
                                             180.*deg);           // delta theta

        WorldLogical = new G4LogicalVolume(World_Solid,    // solid volume
                                           g4galactic,      // material
                                           "World_Logical"); // name

        WorldLogical->SetFieldManager(global_fieldManager, true); // apply magnetic field manager

        // register magnetic field and its manager for delete
        G4AutoDelete::Register(global_field);
        G4AutoDelete::Register(global_fieldManager);

        // manually set max step size for world volume
        // G4double maxStep = 1.*m;
        // fStepLimit = new G4UserLimits(maxStep);
    }
}

```

```

//      WorldLogical->SetUserLimits(fStepLimit);
}

// set visibility to invisible
WorldLogical->SetVisAttributes(G4VisAttributes::Invisible);

WorldPhysical = new G4PVPlacement(0,          // rotation matrix
                                   G4ThreeVector(), // translation vector
                                   WorldLogical, // logical volume
                                   "World_Physical", // name
                                   0,           // mother volume
                                   false,      // unused boolean
                                   0);         // copy number

return WorldPhysical;
}

//_____//

// construct mylar blanket (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructMylarBlanket(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4mylar = G4Material::GetMaterial("G4_MYLAR");

    // set mylar blanket volume
    G4VSolid *MylarBlanket_Solid = new G4Tubs("MylarBlanket_Solid", // name
                                                2.51*m,                // inner radius
                                                2.5149*m,              // outer radius
                                                5.*m,                  // z half length
                                                0,                    // start phi
                                                360.*deg);             // delta phi

    MylarBlanketLogical = new G4LogicalVolume(MylarBlanket_Solid,    // solid volume
                                                g4mylar,               // material
                                                "MylarBlanket_Logical"); // name

    // set mylar color
    MylarBlanketLogical->SetVisAttributes(G4Colour(0.7529, 0.7529, 0.7529));
    //MylarBlanketLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    MylarBlanketLogical->SetUserLimits(fStepLimit);

    MylarBlanketPhysical = new G4PVPlacement(0,          // rotation matrix
                                                G4ThreeVector(), // translation vector
                                                MylarBlanketLogical, // logical volume
                                                "MylarBlanket_Physical", // name
                                                pMotherLogical, // mother volume
                                                false,      // unused boolean
                                                0);         // copy number

    return MylarBlanketPhysical;
}

//_____//

// construct dacron blanket (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructDacronBlanket(G4LogicalVolume
*pMotherLogical)
{
    // get materials

```

```

G4Material *g4dacron = G4Material::GetMaterial("G4_DACRON");

// set dacron blanket volume
G4VSolid *DacronBlanket_Solid = new G4Tubs("DacronBlanket_Solid", // name
                                           2.505*m,                // inner radius
                                           2.5099*m,                // outer radius
                                           5.*m,                    // z half length
                                           0,                       // start phi
                                           360.*deg);                // delta phi

DacronBlanketLogical = new G4LogicalVolume(DacronBlanket_Solid,    // solid volume
                                           g4dacron,                // material
                                           "DacronBlanket_Logical"); // name

// set dacron color
DacronBlanketLogical->SetVisAttributes(G4Colour(1., 1., 1.));
//DacronBlanketLogical->SetVisAttributes(G4VisAttributes::Invisible);

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
DacronBlanketLogical->SetUserLimits(fStepLimit);

DacronBlanketPhysical = new G4PVPlacement(0,                       // rotation matrix
                                           G4ThreeVector(),          // translation vector
                                           DacronBlanketLogical,      // logical volume
                                           "DacronBlanket_Physical",  // name
                                           pMotherLogical,            // mother volume
                                           false,                     // unused boolean
                                           0);                         // copy number

return DacronBlanketPhysical;
}

// _____//

// construct kevlar blanket (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructKevlarBlanket(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4kevlar = G4Material::GetMaterial("G4_KEVLAR");

    // set kevlar blanket volume
    G4VSolid *KevlarBlanket_Solid = new G4Tubs("KevlarBlanket_Solid", // name
                                                2.5*m,                  // inner radius
                                                2.5049*m,                // outer radius
                                                5.*m,                    // z half length
                                                0,                       // start phi
                                                360.*deg);                // delta phi

    KevlarBlanketLogical = new G4LogicalVolume(KevlarBlanket_Solid,    // solid volume
                                                g4kevlar,                // material
                                                "KevlarBlanket_Logical"); // name

    // set kevlar color
    KevlarBlanketLogical->SetVisAttributes(G4Colour(1., 0.980, 0.804));
    //KevlarBlanketLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    KevlarBlanketLogical->SetUserLimits(fStepLimit);

    KevlarBlanketPhysical = new G4PVPlacement(0,                       // rotation matrix
                                                G4ThreeVector(),          // translation vector
                                                KevlarBlanketLogical,      // logical volume

```

```

        "KevlarBlanket_Physical", // name
        pMotherLogical,          // mother volume
        false,                   // unused boolean
        0);                      // copy number

    return KevlarBlanketPhysical;
}

// _____//

// construct spacecraft aluminum shell (1.8 cm)

G4VPhysicalVolume * DetectorConstruction::ConstructAluminumShell(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4Al = G4Material::GetMaterial("G4_Al");

    // set aluminum shell volume
    G4VSolid *AlShell_Solid = new G4Tubs("AlShell_Solid",           // name
        2.4820*m,           // inner radius
        2.4999*m,           // outer radius
        5.*m,               // z half length
        0,                  // start phi
        360.*deg);          // delta phi

    AluminumShellLogical = new G4LogicalVolume(AlShell_Solid,      // solid volume
        g4Al,               // material
        "AlShell_Logical"); // name

    // set aluminum color
    AluminumShellLogical->SetVisAttributes(G4Colour(0.518, 0.529, 0.537));
    //AluminumShellLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    AluminumShellLogical->SetUserLimits(fStepLimit);

    AluminumShellPhysical = new G4PVPlacement(0,                   // rotation matrix
        G4ThreeVector(),    // translation vector
        AluminumShellLogical, // logical volume
        "AlShell_Physical",  // name
        pMotherLogical,      // mother volume
        false,               // unused boolean
        0);                  // copy number

    return AluminumShellPhysical;
}

// _____//

// construct mylar end cap 1 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapMylar1(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4mylar = G4Material::GetMaterial("G4_MYLAR");

    // set mylar end cap volume 1
    G4VSolid *EndCapMylar1_Solid = new G4Sphere("EndCapMylar1_Solid", // sphere segment
        6.0281*m,           // inner radius
        6.0331*m,           // outer radius
        0,                  // start phi
        360.*deg,           // delta phi
        0,                  // start theta
        0,                  // end theta
        0);                 // end phi

```

```

                25.*deg); // delta theta

SpacecraftEndCapMylar1Logical = new G4LogicalVolume(EndCapMylar1_Solid, // solid volume
                                                    g4mylar, // material
                                                    "EndCapMylar1_Logical"); // name

// set mylar color
SpacecraftEndCapMylar1Logical->SetVisAttributes(G4Colour(0.7529, 0.7529, 0.7529));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
SpacecraftEndCapMylar1Logical->SetUserLimits(fStepLimit);

SpacecraftEndCapMylar1Physical = new G4PVPlacement(0, // rotation matrix
                                                    G4ThreeVector(0,0,-0.45*m), // translation vector
                                                    SpacecraftEndCapMylar1Logical, // logical volume
                                                    "EndCapMylar1_Physical", // name
                                                    pMotherLogical, // mother volume
                                                    false, // unused boolean
                                                    0); // copy number

return SpacecraftEndCapMylar1Physical;
}

// _____//

// construct mylar end cap 2 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapMylar2(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4mylar = G4Material::GetMaterial("G4_MYLAR");

    // set mylar end cap volume 1
    G4VSolid *EndCapMylar2_Solid = new G4Sphere("EndCapMylar2_Solid", // sphere segment
                                                6.0281*m, // inner radius
                                                6.0331*m, // outer radius
                                                0, // start phi
                                                360.*deg, // delta phi
                                                0, // start theta
                                                25.*deg); // delta theta

    SpacecraftEndCapMylar2Logical = new G4LogicalVolume(EndCapMylar2_Solid, // solid volume
                                                        g4mylar, // material
                                                        "EndCapMylar2_Logical"); // name

    // set mylar color
    SpacecraftEndCapMylar2Logical->SetVisAttributes(G4Colour(0.7529, 0.7529, 0.7529));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftEndCapMylar2Logical->SetUserLimits(fStepLimit);

    // rotation matrix to place second end cap on opposite side of habitat
    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateY(180.*deg);
    SpacecraftEndCapMylar2Physical = new G4PVPlacement(rm, // rotation matrix
                                                        G4ThreeVector(0,0,0.45*m), // translation vector
                                                        SpacecraftEndCapMylar2Logical, // logical volume
                                                        "EndCapMylar2_Physical", // name
                                                        pMotherLogical, // mother volume
                                                        false, // unused boolean
                                                        0); // copy number

    return SpacecraftEndCapMylar2Physical;
}

```

```

}

// _____//

// construct dacron end cap 1 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapDacron1(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4dacron = G4Material::GetMaterial("G4_DACRON");

    // set dacron end cap volume 1
    G4VSolid *EndCapDacron1_Solid = new G4Sphere("EndCapDacron1_Solid",    // sphere segment
        6.0231*m,                    // inner radius
        6.0280*m,                    // outer radius
        0,                            // start phi
        360.*deg,                     // delta phi
        0,                            // start theta
        25.*deg);                    // delta theta

    SpacecraftEndCapDacron1Logical = new G4LogicalVolume(EndCapDacron1_Solid, // solid volume
        g4dacron,                    // material
        "EndCapDacron1_Logical"); // name

    // set dacron color
    SpacecraftEndCapDacron1Logical->SetVisAttributes(G4Colour(1., 1., 1.));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftEndCapDacron1Logical->SetUserLimits(fStepLimit);

    SpacecraftEndCapDacron1Physical = new G4PVPlacement(0,                    // rotation matrix
        G4ThreeVector(0,0,-0.45*m),    // translation vector
        SpacecraftEndCapDacron1Logical, // logical volume
        "EndCapDacron1_Physical",       // name
        pMotherLogical,                // mother volume
        false,                          // unused boolean
        0);                            // copy number

    return SpacecraftEndCapDacron1Physical;
}

// _____//

// construct dacron end cap 2 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapDacron2(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4dacron = G4Material::GetMaterial("G4_DACRON");

    // set dacron end cap volume 2
    G4VSolid *EndCapDacron2_Solid = new G4Sphere("EndCapDacron2_Solid",    // sphere segment
        6.0231*m,                    // inner radius
        6.0280*m,                    // outer radius
        0,                            // start phi
        360.*deg,                     // delta phi
        0,                            // start theta
        25.*deg);                    // delta theta

    SpacecraftEndCapDacron2Logical = new G4LogicalVolume(EndCapDacron2_Solid, // solid volume
        g4dacron,                    // material
        "EndCapDacron2_Logical"); // name

```

```

// set dacron color
SpacecraftEndCapDacron2Logical->SetVisAttributes(G4Colour(1., 1., 1.));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
SpacecraftEndCapDacron2Logical->SetUserLimits(fStepLimit);

// rotation matrix to place second end cap on opposite side of habitat
G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateY(180.*deg);
SpacecraftEndCapDacron2Physical = new G4PVPlacement(rm, // rotation matrix
G4ThreeVector(0,0,0.45*m), // translation vector
SpacecraftEndCapDacron2Logical, // logical volume
"EndCapDacron2_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return SpacecraftEndCapDacron2Physical;
}

// _____//

// construct kevlar end cap 1 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapKevlar1(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4kevlar = G4Material::GetMaterial("G4_KEVLAR");

    // set kevlar end cap volume 1
    G4VSolid *EndCapKevlar1_Solid = new G4Sphere("EndCapKevlar1_Solid", // sphere segment
6.0181*m, // inner radius
6.0230*m, // outer radius
0, // start phi
360.*deg, // delta phi
0, // start theta
25.*deg); // delta theta

    SpacecraftEndCapKevlar1Logical = new G4LogicalVolume(EndCapKevlar1_Solid, // solid volume
g4kevlar, // material
"EndCapKevlar1_Logical"); // name

    // set kevlar color
    SpacecraftEndCapKevlar1Logical->SetVisAttributes(G4Colour(1., 0.980, 0.804));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftEndCapKevlar1Logical->SetUserLimits(fStepLimit);

    SpacecraftEndCapKevlar1Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,-0.45*m), // translation vector
SpacecraftEndCapKevlar1Logical, // logical volume
"EndCapKevlar1_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

    return SpacecraftEndCapKevlar1Physical;
}

// _____//

```

```

// construct kevlar end cap 2 (5 mm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapKevlar2(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4kevlar = G4Material::GetMaterial("G4_KEVLAR");

    // set kevlar end cap volume 2
    G4VSolid *EndCapKevlar2_Solid = new G4Sphere("EndCapKevlar2_Solid",           // sphere segment
        6.0181*m,                        // inner radius
        6.0230*m,                        // outer radius
        0,                               // start phi
        360.*deg,                        // delta phi
        0,                               // start theta
        25.*deg);                        // delta theta

    SpacecraftEndCapKevlar2Logical = new G4LogicalVolume(EndCapKevlar2_Solid, // solid volume
        g4kevlar,                          // material
        "EndCapKevlar2_Logical"); // name

    // set kevlar color
    SpacecraftEndCapKevlar2Logical->SetVisAttributes(G4Colour(1., 0.980, 0.804));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftEndCapKevlar2Logical->SetUserLimits(fStepLimit);

    // rotation matrix to place second end cap on opposite side of habitat
    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateY(180.*deg);
    SpacecraftEndCapKevlar2Physical = new G4PVPlacement(rm,                       // rotation matrix
        G4ThreeVector(0,0,0.45*m),        // translation vector
        SpacecraftEndCapKevlar2Logical,    // logical volume
        "EndCapKevlar2_Physical",          // name
        pMotherLogical,                   // mother volume
        false,                             // unused boolean
        0);                               // copy number

    return SpacecraftEndCapKevlar2Physical;
}

// _____//

// construct aluminum end cap 1 (1.8 cm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapAl1(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4Al = G4Material::GetMaterial("G4_Al");

    // set aluminum end cap volume 1
    G4VSolid *EndCapAl1_Solid = new G4Sphere("EndCapAl1_Solid",           // sphere segment
        6.001*m,                        // inner radius
        6.018*m,                        // outer radius
        0,                               // start phi
        360.*deg,                        // delta phi
        0,                               // start theta
        24.7*deg);                        // delta theta

    SpacecraftEndCapAl1Logical = new G4LogicalVolume(EndCapAl1_Solid,        // solid volume
        g4Al,                          // material
        "EndCapAl1_Logical"); // name

```

```

// set aluminum color
SpacecraftEndCapAl1Logical->SetVisAttributes(G4Colour(0.518, 0.529, 0.537));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
SpacecraftEndCapAl1Logical->SetUserLimits(fStepLimit);

SpacecraftEndCapAl1Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,-0.45*m), // translation vector
SpacecraftEndCapAl1Logical, // logical volume
"EndCapAl1_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return SpacecraftEndCapAl1Physical;
}

// _____//

// construct aluminum end cap 2 (1.8 cm)

G4VPhysicalVolume * DetectorConstruction::ConstructEndCapAl2(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4Al = G4Material::GetMaterial("G4_Al");

    // set aluminum end cap volume 2
    G4VSolid *EndCapAl2_Solid = new G4Sphere("EndCapAl2_Solid", // sphere segment
6.001*m, // inner radius
6.018*m, // outer radius
0, // start phi
360.*deg, // delta phi
0, // start theta
24.7*deg); // delta theta

    SpacecraftEndCapAl2Logical = new G4LogicalVolume(EndCapAl2_Solid, // solid volume
g4Al, // material
"EndCapAl2_Logical"); // name

    // set aluminum color
    SpacecraftEndCapAl2Logical->SetVisAttributes(G4Colour(0.518, 0.529, 0.537));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftEndCapAl2Logical->SetUserLimits(fStepLimit);

    // rotation matrix to place second end cap on opposite side of habitat
    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateY(180.0*deg);
    SpacecraftEndCapAl2Physical = new G4PVPlacement(rm, // rotation matrix
G4ThreeVector(0,0,0.45*m), // translation vector
SpacecraftEndCapAl2Logical, // logical volume
"EndCapAl2_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

    return SpacecraftEndCapAl2Physical;
}

// _____//

// construct spacecraft air

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructSpacecraftAir(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4air = G4Material::GetMaterial("G4_AIR");

    // set spacecraft air cylinder
    G4VSolid *SpacecraftAir_Solid = new G4Tubs("SpacecraftAir_Solid",    // name
                                                0,                        // inner radius
                                                2.4819*m,                // outer radius
                                                5.*m,                    // z half length
                                                0,                        // start phi
                                                360.*deg);               // segment angle

    SpacecraftAirLogical = new G4LogicalVolume(SpacecraftAir_Solid,      // solid volume
                                                g4air,                  // material
                                                "SpacecraftAir_Logical"); // name

    // set visibility to invisible
    SpacecraftAirLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 10.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpacecraftAirLogical->SetUserLimits(fStepLimit);

    SpacecraftAirPhysical = new G4PVPlacement(0,                        // rotation matrix
                                                G4ThreeVector(),        // translation vector
                                                SpacecraftAirLogical,    // logical volume
                                                "SpacecraftAir_Physical", // name
                                                pMotherLogical,          // mother volume
                                                false,                  // unused boolean
                                                0);                      // copy number

    return SpacecraftAirPhysical;
}

// _____//

// construct Orion hatch

G4VPhysicalVolume * DetectorConstruction::ConstructOrionHatch(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4Al = G4Material::GetMaterial("G4_Al");

    // set hatch volume
    G4VSolid *OrionHatch_Solid = new G4Tubs("OrionHatch_Solid",        // name
                                                0.432*m,                // inner radius
                                                0.449*m,                // outer radius
                                                0.25*m,                  // z half length
                                                0,                        // starting phi
                                                360.*deg);               // ending phi

    OrionHatchLogical = new G4LogicalVolume(OrionHatch_Solid,          // solid volume
                                                g4Al,                    // material
                                                "OrionHatch_Logical");    // name

    // set aluminum color
    OrionHatchLogical->SetVisAttributes(G4Colour(0.518, 0.529, 0.537));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OrionHatchLogical->SetUserLimits(fStepLimit);

    OrionHatchPhysical = new G4PVPlacement(0,                        // rotation matrix

```

```

        G4ThreeVector(0,0,-5.835*m), // translation vector
        OrionHatchLogical,           // logical volume
        "OrionHatch_Physical",        // name
        pMotherLogical,               // mother volume
        false,                        // unused boolean
        0);                           // copy number

    return OrionHatchPhysical;
}

// _____//

// construct air inside Orion hatch

G4VPhysicalVolume * DetectorConstruction::ConstructAirCylinder(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4air = G4Material::GetMaterial("G4_AIR");

    // set air cylinder volume
    G4VSolid *AirCylinder_Solid = new G4Tubs("AirCylinder_Solid", // name
        0, // inner radius
        0.4319*m, // outer radius
        0.25*m, // z half length
        0, // starting phi
        360.*deg); // ending phi

    AirCylinderLogical = new G4LogicalVolume(AirCylinder_Solid, // solid volume
        g4air, // material
        "AirCylinder_Logical"); // name

    // set visibility to invisible
    AirCylinderLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 10.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    AirCylinderLogical->SetUserLimits(fStepLimit);

    AirCylinderPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0,0,-5.835*m), // translation vector
        AirCylinderLogical, // logical volume
        "AirCylinder_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    return AirCylinderPhysical;
}

// _____//

// construct Orion main volume

G4VPhysicalVolume * DetectorConstruction::ConstructOrionMain(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *ALi_2195 = G4Material::GetMaterial("ALi_2195");

    // set Orion main body volume
    G4VSolid *OrionMain_Solid = new G4Cons("OrionMain_Solid", // name
        2.482*m, // rmin
        2.5*m, // rmax
        0.982*m, // Rmin
        1.0*m, // Rmax
        1.65*m, // half height
        0, // starting phi
        360.*deg); // ending phi

```

```

OrionMainLogical = new G4LogicalVolume(OrionMain_Solid,           // solid volume
                                       AllLi_2195,                 // material
                                       "OrionMain_Logical");       // name

// set Orion color to white
OrionMainLogical->SetVisAttributes(G4Colour(1.,1.,1.));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
OrionMainLogical->SetUserLimits(fStepLimit);

OrionMainPhysical = new G4PVPlacement(0,                          // rotation matrix
                                       G4ThreeVector(0,0,-7.735*m), // translation vector
                                       OrionMainLogical,           // logical volume
                                       "OrionMain_Physical",        // name
                                       pMotherLogical,              // mother volume
                                       false,                       // unused boolean
                                       0);                          // copy number

    return OrionMainPhysical;
}

// _____//

// construct Orion top

G4VPhysicalVolume * DetectorConstruction::ConstructOrionTop(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *AllLi_2195 = G4Material::GetMaterial("AllLi_2195");

    // set Orion top layer volume
    G4VSolid *OrionTop_Solid = new G4Tubs("OrionTop_Solid",       // name
                                           0.45*m,                 // inner radius
                                           1.*m,                   // outer radius
                                           0.009*m,                // z half length
                                           0,                      // starting phi
                                           360.*deg);              // ending phi

    OrionTopLogical = new G4LogicalVolume(OrionTop_Solid,          // solid volume
                                           AllLi_2195,              // material
                                           "OrionTop_Logical");      // name

    // set Orion color to white
    OrionTopLogical->SetVisAttributes(G4Colour(1.,1.,1.));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OrionTopLogical->SetUserLimits(fStepLimit);

    OrionTopPhysical = new G4PVPlacement(0,                        // rotation matrix
                                           G4ThreeVector(0,0,-6.073*m), // translation vector
                                           OrionTopLogical,           // logical volume
                                           "OrionTop_Physical",        // name
                                           pMotherLogical,              // mother volume
                                           false,                       // unused boolean
                                           0);                          // copy number

    return OrionTopPhysical;
}

// _____//

// construct Orion bottom

G4VPhysicalVolume * DetectorConstruction::ConstructOrionBottom(G4LogicalVolume *pMotherLogical)

```

```

{
    // get materials
    G4Material *Alli_2195 = G4Material::GetMaterial("Alli_2195");

    // set Orion bottom layer volume
    G4VSolid *OrionBottom_Solid = new G4Sphere("OrionBottom_Solid", // name
                                                5.84000*m, // inner radius
                                                5.88400*m, // outer radius
                                                0, // start phi
                                                360.*deg, // delta phi
                                                0, // start theta
                                                25.2*deg); // delta theta

    OrionBottomLogical = new G4LogicalVolume(OrionBottom_Solid, // solid volume
                                             Alli_2195, // material
                                             "OrionBottom_Logical"); // name

    // set Orion color to white
    OrionBottomLogical->SetVisAttributes(G4Colour(1.,1.,1.));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OrionBottomLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(180.*degree);
    OrionBottomPhysical = new G4PVPlacement(rm, // rotation matrix
                                             G4ThreeVector(0,0,-4.105*m), // translation vector
                                             OrionBottomLogical, // logical volume
                                             "OrionBottom_Physical", // name
                                             pMotherLogical, // mother volume
                                             false, // unused boolean
                                             0); // copy number

    return OrionBottomPhysical;
}

// _____//

// construct Orion heat shield

G4VPhysicalVolume * DetectorConstruction::ConstructOrionHeatShield(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *AVCOAT = G4Material::GetMaterial("AVCOAT");

    // set Orion heat shield volume
    G4VSolid *OrionHeatShield_Solid = new G4Sphere("OrionHeatShield_Solid", // name
                                                    5.885*m, // inner radius
                                                    6.035*m, // outer radius
                                                    0, // start phi
                                                    360.*deg, // delta phi
                                                    0, // start theta
                                                    25.2*deg); // delta theta

    OrionHeatShieldLogical = new G4LogicalVolume(OrionHeatShield_Solid, // solid volume
                                                  AVCOAT, // material
                                                  "OrionHeatShield_Logical"); // name

    // set AVCOAT color
    OrionHeatShieldLogical->SetVisAttributes(G4Colour(0.6627, 0.6627, 0.6627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OrionHeatShieldLogical->SetUserLimits(fStepLimit);
}

```

```

G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateX(180.*degree);
OrionHeatShieldPhysical = new G4PVPlacement(rm,                                // rotation matrix
      G4ThreeVector(0,0,-4.105*m),      // translation vector
      OrionHeatShieldLogical,           // logical volume
      "OrionHeatShield_Physical",       // name
      pMotherLogical,                  // mother volume
      false,                           // unused boolean
      0);                              // copy number

    return OrionHeatShieldPhysical;
}

// _____//

// construct air inside Orion

G4VPhysicalVolume * DetectorConstruction::ConstructOrionAir(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4air = G4Material::GetMaterial("G4_AIR");

    // set Orion air volume
    G4VSolid *OrionAir_Solid = new G4Cons("OrionAir_Solid",           // name
      0,                        // rmin
      2.4819*m,                 // rmax
      0,                        // Rmin
      0.9819*m,                 // Rmax
      1.65*m,                   // half height
      0,                        // starting phi
      360.*deg);                // ending phi

    OrionAirLogical = new G4LogicalVolume(OrionAir_Solid,             // solid volume
      g4air,                   // material
      "OrionAir_Logical");     // name

    // set visibility to invisible
    OrionAirLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OrionAirLogical->SetUserLimits(fStepLimit);

    OrionAirPhysical = new G4PVPlacement(0,                            // rotation matrix
      G4ThreeVector(0,0,-7.735*m), // translation vector
      OrionAirLogical,          // logical volume
      "OrionAir_Physical",      // name
      pMotherLogical,           // mother volume
      false,                    // unused boolean
      0);                       // copy number

    return OrionAirPhysical;
}

// _____//

// construct water wall shield (20 cm)

G4VPhysicalVolume * DetectorConstruction::ConstructWaterWallShield(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4water = G4Material::GetMaterial("G4_WATER");

    // set water shield volume
    G4VSolid *WaterWallShield_Solid = new G4Tubs("WaterWallShield_Solid", // name
      2.2820*m,                    // inner radius

```

```

                2.4819*m,           // outer radius
                5.*m,             // z half length
                0,                // start phi
                360.*deg);        // segment angle

WaterWallShieldLogical = new G4LogicalVolume(WaterWallShield_Solid, // solid volume
                g4water,         // material
                "WaterWallShield_Logical"); // name

// set water color
WaterWallShieldLogical->SetVisAttributes(G4Colour(0.529, 0.808, 0.980));

// manually set step size for volume
G4double maxStep = 10.*cm;
fStepLimit = new G4UserLimits(maxStep);
WaterWallShieldLogical->SetUserLimits(fStepLimit);

WaterWallShieldPhysical= new G4PVPlacement(0,           // rotation matrix
                G4ThreeVector(),           // translation vector
                WaterWallShieldLogical,     // logical volume
                "WaterWallShield_Physical", // name
                pMotherLogical,            // mother volume
                false,                     // unused boolean
                0);                          // copy number

return WaterWallShieldPhysical;
}

// _____//

// construct 2.5 m shielding toroid magnets

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingToroid(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set shielding 2.5 m toroid magnet volume
    G4VSolid *ShieldingToroid_Solid = new G4Torus("ShieldingToroid_Solid", // name
                1.23*m, // inner toroid cross-sectional radius
                1.25*m, // outer toroid cross-sectional radius
                3.85*m, // radius of toroid
                0,      // start phi
                360.*deg); // delta phi

    ShieldingToroidLogical = new G4LogicalVolume(ShieldingToroid_Solid, // solid volume
                YBCO,           // material
                "ShieldingToroid_Logical"); // name

    // set color to white
    ShieldingToroidLogical->SetVisAttributes(G4Colour(1., 1., 1.));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    ShieldingToroidLogical->SetUserLimits(fStepLimit);

    ShieldingToroid1Physical = new G4PVPlacement(0,           // rotation matrix
                G4ThreeVector(0,0,3.75*m), // translation vector
                ShieldingToroidLogical,     // logical volume
                "ShieldingToroid1_Physical", // name
                pMotherLogical,            // mother volume
                false,                     // unused boolean
                0);                          // copy number

    ShieldingToroid2Physical = new G4PVPlacement(0,           // rotation matrix
                G4ThreeVector(0,0,1.25*m), // translation vector

```

```

        ShieldingToroidLogical,           // logical volume
        "ShieldingToroid2_Physical",      // name
        pMotherLogical,                   // mother volume
        false,                             // unused boolean
        0);                               // copy number

    ShieldingToroid3Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(0,0,-1.25*m),       // translation vector
        ShieldingToroidLogical,           // logical volume
        "ShieldingToroid3_Physical",      // name
        pMotherLogical,                   // mother volume
        false,                             // unused boolean
        0);                               // copy number

    ShieldingToroid4Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(0,0,-3.75*m),       // translation vector
        ShieldingToroidLogical,           // logical volume
        "ShieldingToroid4_Physical",      // name
        pMotherLogical,                   // mother volume
        false,                             // unused boolean
        0);                               // copy number

    // calculate mass
    G4double ToroidVol = ShieldingToroidLogical->GetSolid()->GetCubicVolume();
    G4String ToroidMat = ShieldingToroidLogical->GetMaterial()->GetName();
    G4double ToroidDensity = ShieldingToroidLogical->GetMaterial()->GetDensity();
    G4double ToroidMass = (ToroidVol)*ToroidDensity;
    G4cout << "Mass of Shielding Toroid = " << ToroidMass/gram << " g" << G4endl;
    G4cout << "Mass of Shielding Toroid = " << ToroidMass/kilogram << " kg" << G4endl;
    G4cout << "Mass of Shielding Toroids = " << ToroidMass/kilogram * 4 << " kg" << G4endl;

    return ShieldingToroid1Physical;
}

// _____//

// construct shielding 2.5 m toroid magnet interiors

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingToroidInterior(G4LogicalVolume
*pMotherLogical)
{
    // magnetic field
    G4MagneticField *CMField = new ConfinedMagneticField(Bo);
    G4FieldManager *CMFieldManager = new G4FieldManager;
    CMFieldManager->SetDetectorField(CMField);
    CMFieldManager->CreateChordFinder(CMField);

    // get materials
    G4Material *g4galactic = G4Material::GetMaterial("G4_Galactic");

    // set shielding 2.5 m toroid magnet interior volume
    G4VSolid *ShieldingToroidInterior_Solid = new G4Torus("ShieldingToroidInterior_Solid", //name
        0, // inner toroid cross-sectional radius
        1.2299*m, // outer toroid cross-sectional radius
        3.85*m, // radius of toroid
        0, // start phi
        360.*deg); // delta phi

    ShieldingToroidInteriorLogical = new G4LogicalVolume(ShieldingToroidInterior_Solid, // solid
        g4galactic, // material
        "ShieldingToroidInterior_Logical"); // name

    ShieldingToroidInteriorLogical->SetFieldManager(CMFieldManager, false);
    // apply magnetic field manager

    // register magnetic field and its manager for delete
    G4AutoDelete::Register(CMField);
    G4AutoDelete::Register(CMFieldManager);
}

```

```

// set visibility to invisible
ShieldingToroidInteriorLogical->SetVisAttributes(G4VisAttributes::Invisible);

// manually set step size for volume
G4double maxStep = 10.*cm;
fStepLimit = new G4UserLimits(maxStep);
ShieldingToroidInteriorLogical->SetUserLimits(fStepLimit);

ShieldingToroidInterior1Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,3.75*m), // translation vector
ShieldingToroidInteriorLogical, // logical volume
"ShieldingToroidInterior1_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingToroidInterior2Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,1.25*m), // translation vector
ShieldingToroidInteriorLogical, // logical volume
"ShieldingToroidInterior2_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingToroidInterior3Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,-1.25*m), // translation vector
ShieldingToroidInteriorLogical, // logical volume
"ShieldingToroidInterior3_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingToroidInterior4Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,-3.75*m), // translation vector
ShieldingToroidInteriorLogical, // logical volume
"ShieldingToroidInterior4_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return ShieldingToroidInterior1Physical;
}

// _____//

// construct shielding solenoid magnets

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingSolenoid(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set shielding solenoid magnet volume
    G4VSolid *ShieldingSolenoid_Solid = new G4Tubs("ShieldingSolenoid_Solid", // name
3.95*m, // inner radius
3.999*m, // outer radius
10.*m, // z half length
0, // start phi
360.*deg); // delta phi

    ShieldingSolenoidLogical = new G4LogicalVolume(ShieldingSolenoid_Solid, // solid volume
YBCO, // material
"ShieldingSolenoid_Logical"); // name

    // set color to gray to match NIAC
    ShieldingSolenoidLogical->SetVisAttributes(G4Colour(0.7843, 0.7843, 0.8431));
}

```

```

// // manually set step size for volume
// G4double maxStep = 1.*cm;
// fStepLimit = new G4UserLimits(maxStep);
// ShieldingSolenoidLogical->SetUserLimits(fStepLimit);

ShieldingSolenoid1Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(8.*m,0,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid1_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingSolenoid2Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(4.*m,-6.928*m,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid2_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingSolenoid3Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-4.*m,-6.928*m,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid3_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingSolenoid4Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-8.*m,0,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid4_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingSolenoid5Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-4.*m,6.928*m,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid5_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

ShieldingSolenoid6Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(4.*m,6.928*m,0), // translation vector
ShieldingSolenoidLogical, // logical volume
"ShieldingSolenoid6_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return ShieldingSolenoid1Physical;
}

// _____//

// construct shielding solenoid magnet interiors

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingSolenoidInterior(G4LogicalVolume
*pMotherLogical)
{
    // define uniform magnetic field inside solenoids
    G4MagneticField *Bfield = new G4UniformMagField(G4ThreeVector(0,0,Bo));
    G4FieldManager *BfieldManager = new G4FieldManager;
    BfieldManager->SetDetectorField(Bfield);
}

```

```

BfieldManager->CreateChordFinder(Bfield);

// get materials
G4Material *g4galactic = G4Material::GetMaterial("G4_Galactic");

// set shielding solenoid magnet interior volume
G4VSolid *ShieldingSolenoidInterior_Solid = new G4Tubs("ShieldingSolenoidInterior_Solid",
    0, // inner radius
    3.9499*m, // outer radius
    10.*m, // z half length
    0, // start phi
    360.*deg); // delta phi

ShieldingSolenoidInteriorLogical = new G4LogicalVolume(ShieldingSolenoidInterior_Solid,
    g4galactic, // material
    "ShieldingSolenoidInterior_Logical"); // name

ShieldingSolenoidInteriorLogical->SetFieldManager(BfieldManager, false);
// to apply magnetic field manager

// register magnetic field and its manager for delete
G4AutoDelete::Register(Bfield);
G4AutoDelete::Register(BfieldManager);

// set visibility to invisible
ShieldingSolenoidInteriorLogical->SetVisAttributes(G4VisAttributes::Invisible);

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
ShieldingSolenoidInteriorLogical->SetUserLimits(fStepLimit);

ShieldingSolenoidInterior1Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(8.*m,0,0), // translation vector
    ShieldingSolenoidInteriorLogical, // logical volume
    "ShieldingSolenoidInterior1_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

ShieldingSolenoidInterior2Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.*m,-6.928*m,0), // translation vector
    ShieldingSolenoidInteriorLogical, // logical volume
    "ShieldingSolenoidInterior2_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

ShieldingSolenoidInterior3Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-4.*m,-6.928*m,0), // translation vector
    ShieldingSolenoidInteriorLogical, // logical volume
    "ShieldingSolenoidInterior3_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

ShieldingSolenoidInterior4Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-8.*m,0,0), // translation vector
    ShieldingSolenoidInteriorLogical, // logical volume
    "ShieldingSolenoidInterior4_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

ShieldingSolenoidInterior5Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-4.*m,6.928*m,0), // translation vector
    ShieldingSolenoidInteriorLogical, // logical volume
    "ShieldingSolenoidInterior5_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

```

```

        0); // copy number

ShieldingSolenoidInterior6Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(4.*m,6.928*m,0), // translation vector
ShieldingSolenoidInteriorLogical, // logical volume
"ShieldingSolenoidInterior6_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return ShieldingSolenoidInterior1Physical;
}

// _____//

// construct shielding solenoid support cylinders

G4VPhysicalVolume *
DetectorConstruction::ConstructShieldingSolenoidSupportCylinder(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4graphite = G4Material::GetMaterial("G4_GRAPHITE");

    // set shielding solenoid support cylinder volume
    G4VSolid *ShieldingSolenoidSupportCylinder_Solid = new
    G4Tubs("ShieldingSolenoidSupportCylinder_Solid", // name
        0.49*m, // inner radius
        0.5*m, // outer radius
        10.*m, // z half length
        0, // start phi
        360.*deg); // delta phi

    ShieldingSolenoidSupportCylinderLogical = new
    G4LogicalVolume(ShieldingSolenoidSupportCylinder_Solid, // solid volume
        g4graphite, // material
        "ShieldingSolenoidSupportCylinder_Logical"); // name

    // set graphite color
    ShieldingSolenoidSupportCylinderLogical->SetVisAttributes(G4Colour(0.2196, 0.2039, 0.1569));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    ShieldingSolenoidSupportCylinderLogical->SetUserLimits(fStepLimit);

    ShieldingSolenoidSupportCylinderPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(), // translation vector
        ShieldingSolenoidSupportCylinderLogical, // logical volume
        "ShieldingSolenoidSupportCylinder_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    return ShieldingSolenoidSupportCylinderPhysical;
}

// _____//

// construct shielding solenoid support plates

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingSolenoidSupportPlate(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4C = G4Material::GetMaterial("G4_GRAPHITE");

```

```

// set shielding solenoid support plate volume
G4VSolid *ShieldingSolenoidSupportPlate_Solid = new
G4Box("ShieldingSolenoidSupportPlate_Solid",          // name
      1.7244*m,          // half x
      1.25*mm,           // half y
      10.*m);           // half z

ShieldingSolenoidSupportPlateLogical = new
G4LogicalVolume(ShieldingSolenoidSupportPlate_Solid,    // solid volume
                g4C,          // material
                "ShieldingSolenoidSupportPlate_Logical");// name

// set graphite color
ShieldingSolenoidSupportPlateLogical->SetVisAttributes(G4Colour(0.2196, 0.2039, 0.1569));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
ShieldingSolenoidSupportPlateLogical->SetUserLimits(fStepLimit);

ShieldingSolenoidSupportPlate1Physical = new G4PVPlacement(0,          // rotation matrix
G4ThreeVector(2.2254*m,0,0),    // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate1_Physical", // name
pMotherLogical,          // mother volume
false,                   // unused boolean
0);                       // copy number

G4RotationMatrix *rm2= new G4RotationMatrix();
rm2->rotateZ(-60.0*deg);
ShieldingSolenoidSupportPlate2Physical = new G4PVPlacement(rm2,          // rotation matrix
G4ThreeVector(1.1127*m,1.9273*m,0), // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate2_Physical", // name
pMotherLogical,          // mother volume
false,                   // unused boolean
0);                       // copy number

G4RotationMatrix *rm3 = new G4RotationMatrix();
rm3->rotateZ(-120.0*deg);
ShieldingSolenoidSupportPlate3Physical = new G4PVPlacement(rm3,          // rotation matrix
G4ThreeVector(-1.1127*m,1.9273*m,0), // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate3_Physical", // name
pMotherLogical,          // mother volume
false,                   // unused boolean
0);                       // copy number

G4RotationMatrix *rm4 = new G4RotationMatrix();
rm4->rotateZ(-180.0*deg);
ShieldingSolenoidSupportPlate4Physical = new G4PVPlacement(rm4,          // rotation matrix
G4ThreeVector(-2.2254*m,0,0),    // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate4_Physical", // name
pMotherLogical,          // mother volume
false,                   // unused boolean
0);                       // copy number

G4RotationMatrix *rm5 = new G4RotationMatrix();
rm5->rotateZ(-240.0*deg);

ShieldingSolenoidSupportPlate5Physical = new G4PVPlacement(rm5,          // rotation matrix
G4ThreeVector(-1.1127*m,-1.9273*m,0), // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate5_Physical", // name
pMotherLogical,          // mother volume
false,                   // unused boolean
0);                       // copy number

```

```

G4RotationMatrix *rm6 = new G4RotationMatrix();
rm6->rotateZ(-300.0*deg);
ShieldingSolenoidSupportPlate6Physical = new G4PVPlacement(rm6,          // rotation matrix
G4ThreeVector(1.1127*m,-1.9273*m,0), // translation vector
ShieldingSolenoidSupportPlateLogical, // logical volume
"ShieldingSolenoidSupportPlate6_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return ShieldingSolenoidSupportPlate1Physical;
}

// _____//

// construct correction solenoid magnet

G4VPhysicalVolume * DetectorConstruction::ConstructCorrectionSolenoid(G4LogicalVolume
*pMotherLogical)
{
    // define uniform magnetic field if not using world tabulated field
    G4MagneticField *Cfield = new G4UniformMagField(G4ThreeVector(0,0,Bcorr));
    G4FieldManager *CfieldManager = new G4FieldManager;
    CfieldManager->SetDetectorField(Cfield);
    CfieldManager->CreateChordFinder(Cfield);

    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set correction solenoid magnet volume
    G4VSolid *CorrectionSolenoid_Solid = new G4Tubs("CorrectionSolenoid_Solid", // name
3.15*m, // inner radius
3.1999*m, // outer radius
9.9999*m, // z half length
0, // start phi
360.*deg); // delta phi

    CorrectionSolenoidLogical = new G4LogicalVolume(CorrectionSolenoid_Solid, // solid volume
YBCO, // material
"CorrectionSolenoid_Logical");// name

    CorrectionSolenoidLogical->SetFieldManager(CfieldManager, false);
    // to apply magnetic field manager

    // register magnetic field and its manager for delete
    G4AutoDelete::Register(Cfield);
    G4AutoDelete::Register(CfieldManager);

    // set color to red to match NIAC
    CorrectionSolenoidLogical->SetVisAttributes(G4Colour(1., 0, 0));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    CorrectionSolenoidLogical->SetUserLimits(fStepLimit);

    CorrectionSolenoidPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector
CorrectionSolenoidLogical, // logical volume
"CorrectionSolenoid_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return CorrectionSolenoidPhysical;
}

```

```

// _____//

// construct inner shielding cylinders

G4VPhysicalVolume * DetectorConstruction::ConstructInnerShieldingCylinder(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set inner shielding cylinder volume
    G4VSolid *InnerShieldingCylinder_Solid = new G4Tubs("InnerShieldingCylinder_Solid", // name
                                                         0.*m, // inner radius
                                                         0.049*m, // outer radius
                                                         5*m, // z half length
                                                         0, // start phi
                                                         360.*deg); // delta phi

    InnerShieldingCylinderLogical = new G4LogicalVolume(InnerShieldingCylinder_Solid, // solid
                                                         YBCO, // material
                                                         "InnerShieldingCylinder_Logical");//name

    // set color to yellow to match ESA S2RS
    InnerShieldingCylinderLogical->SetVisAttributes(G4Colour(1., 1., 0));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    InnerShieldingCylinderLogical->SetUserLimits(fStepLimit);

    // place cylinder physical volumes
    InnerShieldingCylinder1Physical = new G4PVPlacement(0, // rotation matrix
                                                         G4ThreeVector(2.8*m,0,0), // translation vector (0 deg)
                                                         InnerShieldingCylinderLogical, // logical volume
                                                         "InnerShieldingCylinder1_Physical", // name
                                                         pMotherLogical, // mother volume
                                                         false, // unused boolean
                                                         0); // copy number

    InnerShieldingCylinder2Physical = new G4PVPlacement(0, // rotation matrix
                                                         G4ThreeVector(2.796*m,0.146*m,0), // translation vector (3 deg)
                                                         InnerShieldingCylinderLogical, // logical volume
                                                         "InnerShieldingCylinder2_Physical", // name
                                                         pMotherLogical, // mother volume
                                                         false, // unused boolean
                                                         0, // copy number
                                                         false); // surface check

    InnerShieldingCylinder3Physical = new G4PVPlacement(0, // rotation matrix
                                                         G4ThreeVector(2.785*m,0.293*m,0), // translation vector (6 deg)
                                                         InnerShieldingCylinderLogical, // logical volume
                                                         "InnerShieldingCylinder3_Physical", // name
                                                         pMotherLogical, // mother volume
                                                         false, // unused boolean
                                                         0, // copy number
                                                         false); // surface check

    InnerShieldingCylinder4Physical = new G4PVPlacement(0, // rotation matrix
                                                         G4ThreeVector(2.7655*m,0.438*m,0), // translation vector (9 deg)
                                                         InnerShieldingCylinderLogical, // logical volume
                                                         "InnerShieldingCylinder4_Physical", // name
                                                         pMotherLogical, // mother volume
                                                         false, // unused boolean
                                                         0, // copy number
                                                         false); // surface check

    InnerShieldingCylinder5Physical = new G4PVPlacement(0, // rotation matrix
                                                         G4ThreeVector(2.739*m,0.582*m,0), // translation vector (12 deg)

```

```

        InnerShieldingCylinderLogical,           // logical volume
        "InnerShieldingCylinder5_Physical",      // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder6Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(2.705*m,0.725*m,0),// translation vector (15 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder6_Physical",     // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder7Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(2.663*m,0.865*m,0),// translation vector (18 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder7_Physical",     // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder8Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(2.614*m,1.003*m,0),// translation vector (21 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder8_Physical",     // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder9Physical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(2.558*m,1.139*m,0),// translation vector (24 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder9_Physical",     // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder10Physical = new G4PVPlacement(0,          // rotation matrix
        G4ThreeVector(2.495*m,1.271*m,0),// translation vector (27 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder10_Physical",    // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder11Physical = new G4PVPlacement(0,          // rotation matrix
        G4ThreeVector(2.425*m,1.4*m,0),// translation vector (30 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder11_Physical",    // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder12Physical = new G4PVPlacement(0,          // rotation matrix
        G4ThreeVector(2.348*m,1.525*m,0),// translation vector (33 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder12_Physical",    // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

```

```

InnerShieldingCylinder13Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(2.265*m,1.646*m,0), // translation vector (36 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder13_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder14Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(2.176*m,1.762*m,0), // translation vector (39 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder14_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder15Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(2.081*m,1.874*m,0), // translation vector (42 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder15_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder16Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.98*m,1.98*m,0), // translation vector (45 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder16_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder17Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.8735*m,2.08*m,0), // translation vector (48 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder17_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder18Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.762*m,2.176*m,0), // translation vector (51 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder18_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder19Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.646*m,2.265*m,0), // translation vector (54 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder19_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder20Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.525*m,2.348*m,0), // translation vector (57 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder20_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

```

```

        false); // surface check

InnerShieldingCylinder21Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(1.4*m,2.425*m,0), // translation vector (60 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder21_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder22Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(1.271*m,2.495*m,0), // translation vector (63 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder22_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder23Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(1.139*m,2.558*m,0), // translation vector (66 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder23_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder24Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(1.003*m,2.614*m,0), // translation vector (69 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder24_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder25Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.865*m,2.663*m,0), // translation vector (72 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder25_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder26Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.725*m,2.705*m,0), // translation vector (75 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder26_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder27Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.582*m,2.739*m,0), // translation vector (78 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder27_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder28Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.438*m,2.766*m,0), // translation vector (81 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder28_Physical", // name
    pMotherLogical, // mother volume

```

```

        false, // unused boolean
        0, // copy number
        false); // surface check

InnerShieldingCylinder29Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0.293*m,2.785*m,0), // translation vector (84 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder29_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder30Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0.1465*m,2.796*m,0), // translation vector (87 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder30_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder31Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,2.8*m,0), // translation vector (90 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder31_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder32Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-0.1465*m,2.796*m,0), // translation vector (93 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder32_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder33Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-0.2927*m,2.785*m,0), // translation vector (96 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder33_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder34Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-0.4380*m,2.766*m,0), // translation vector (99 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder34_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder35Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-0.5822*m,2.739*m,0), // translation vector (102 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder35_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder36Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-0.7247*m,2.705*m,0), // translation vector (105 deg)
InnerShieldingCylinderLogical, // logical volume

```

```

        "InnerShieldingCylinder36_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0, // copy number
        false); // surface check

InnerShieldingCylinder37Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-0.8652*m,2.663*m,0), // translation vector (108 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder37_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder38Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.003*m,2.614*m,0), // translation vector (111 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder38_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder39Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.139*m,2.558*m,0), // translation vector (114 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder39_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder40Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.271*m,2.495*m,0), // translation vector (117 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder40_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder41Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.4*m,2.425*m,0), // translation vector (120 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder41_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder42Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.525*m,2.348*m,0), // translation vector (123 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder42_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder43Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.646*m,2.265*m,0), // translation vector (126 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder43_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder44Physical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(-1.762*m,2.176*m,0),// translation vector (129 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder44_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder45Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-1.874*m,2.081*m,0),// translation vector (132 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder45_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder46Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-1.980*m,1.980*m,0),// translation vector (135 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder46_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder47Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-2.081*m,1.874*m,0),// translation vector (138 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder47_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder48Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-2.176*m,1.762*m,0),// translation vector (141 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder48_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder49Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-2.265*m,1.646*m,0),// translation vector (144 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder49_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder50Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-2.348*m,1.525*m,0),// translation vector (147 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder50_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

InnerShieldingCylinder51Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-2.425*m,1.4*m,0),// translation vector (150 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder51_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0,                                  // copy number
        false);                             // surface check

```

```

InnerShieldingCylinder52Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.495*m,1.271*m,0),// translation vector (153 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder52_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder53Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.558*m,1.139*m,0),// translation vector (156 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder53_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder54Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.614*m,1.003*m,0),// translation vector (159 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder54_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder55Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.663*m,0.865*m,0),// translation vector (162 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder55_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder56Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.705*m,0.725*m,0),// translation vector (165 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder56_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder57Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.739*m,0.582*m,0),// translation vector (168 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder57_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder58Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.766*m,0.438*m,0),// translation vector (171 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder58_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder59Physical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-2.785*m,0.293*m,0),// translation vector (174 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder59_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

```

```

0, // copy number
false); // surface check

InnerShieldingCylinder60Physical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.796*m,0.147*m,0), // translation vector (177 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder60_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder1aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.8*m,0,0), // translation vector (180 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder1a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder2aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.796*m,-0.146*m,0), // translation vector (183 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder2a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder3aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.785*m,-0.293*m,0), // translation vector (186 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder3a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder4aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.7655*m,-0.438*m,0), // translation vector (189 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder4a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder5aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.739*m,-0.582*m,0), // translation vector (192 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder5a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder6aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.705*m,-0.725*m,0), // translation vector (195 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder6a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder7aPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(-2.663*m,-0.865*m,0), // translation vector (198 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder7a_Physical", // name

```

```

        pMotherLogical,                // mother volume
        false,                        // unused boolean
        0,                            // copy number
        false);                       // surface check

InnerShieldingCylinder8aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.614*m,-1.003*m,0), // translation vector (201 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder8a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder9aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.558*m,-1.139*m,0), // translation vector (204 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder9a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder10aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.495*m,-1.271*m,0), // translation vector (207 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder10a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder11aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.425*m,-1.4*m,0), // translation vector (210 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder11a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder12aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.348*m,-1.525*m,0), // translation vector (213 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder12a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder13aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.265*m,-1.646*m,0), // translation vector (216 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder13a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder14aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.176*m,-1.762*m,0), // translation vector (219 deg)
    InnerShieldingCylinderLogical,    // logical volume
    "InnerShieldingCylinder14a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0,                               // copy number
    false);                          // surface check

InnerShieldingCylinder15aPhysical = new G4PVPlacement(0,                // rotation matrix
    G4ThreeVector(-2.081*m,-1.874*m,0), // translation vector (222 deg)

```

```

        InnerShieldingCylinderLogical,           // logical volume
        "InnerShieldingCylinder15a_Physical",    // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder16aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.98*m,-1.98*m,0), // translation vector (225 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder16a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder17aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.8735*m,-2.08*m,0), // translation vector (228 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder17a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder18aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.762*m,-2.176*m,0), // translation vector (231 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder18a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder19aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.646*m,-2.265*m,0), // translation vector (234 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder19a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder20aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.525*m,-2.348*m,0), // translation vector (237 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder20a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder21aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.4*m,-2.425*m,0), // translation vector (240 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder21a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

InnerShieldingCylinder22aPhysical = new G4PVPlacement(0,           // rotation matrix
        G4ThreeVector(-1.271*m,-2.495*m,0), // translation vector (243 deg)
        InnerShieldingCylinderLogical,          // logical volume
        "InnerShieldingCylinder22a_Physical",   // name
        pMotherLogical,                         // mother volume
        false,                                  // unused boolean
        0,                                       // copy number
        false);                                // surface check

```

```

InnerShieldingCylinder23aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-1.139*m,-2.558*m,0),// translation vector (246 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder23a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder24aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-1.003*m,-2.614*m,0),// translation vector (249 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder24a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder25aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.865*m,-2.663*m,0),// translation vector (252 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder25a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder26aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.725*m,-2.705*m,0),// translation vector (255 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder26a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder27aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.582*m,-2.739*m,0),// translation vector (258 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder27a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder28aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.438*m,-2.766*m,0),// translation vector (261 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder28a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder29aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.293*m,-2.785*m,0),// translation vector (264 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder29a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder30aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(-0.1465*m,-2.796*m,0),// translation vector (267 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder30a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

```

```

        false); // surface check

InnerShieldingCylinder31aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0,-2.8*m,0), // translation vector (270 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder31a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder32aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.1465*m,-2.796*m,0), // translation vector (273 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder32a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder33aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.2927*m,-2.785*m,0), // translation vector (276 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder33a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder34aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.4380*m,-2.766*m,0), // translation vector (279 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder34a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder35aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.5822*m,-2.739*m,0), // translation vector (282 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder35a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder36aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.7247*m,-2.705*m,0), // translation vector (285 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder36a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder37aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0.8652*m,-2.663*m,0), // translation vector (288 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder37a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder38aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(1.003*m,-2.614*m,0), // translation vector (291 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder38a_Physical", // name
    pMotherLogical, // mother volume

```

```

        false,                                // unused boolean
        0,                                    // copy number
        false);                               // surface check

InnerShieldingCylinder39aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.139*m,-2.558*m,0), // translation vector (294 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder39a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder40aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.271*m,-2.495*m,0), // translation vector (297 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder40a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder41aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.4*m,-2.425*m,0), // translation vector (300 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder41a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder42aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.525*m,-2.348*m,0), // translation vector (303 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder42a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder43aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.646*m,-2.265*m,0), // translation vector (306 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder43a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder44aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.762*m,-2.176*m,0), // translation vector (309 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder44a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder45aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.874*m,-2.081*m,0), // translation vector (312 deg)
InnerShieldingCylinderLogical, // logical volume
"InnerShieldingCylinder45a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0, // copy number
false); // surface check

InnerShieldingCylinder46aPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.980*m,-1.980*m,0), // translation vector (315 deg)
InnerShieldingCylinderLogical, // logical volume

```

```

        "InnerShieldingCylinder46a_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0, // copy number
        false); // surface check

InnerShieldingCylinder47aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.081*m,-1.874*m,0), // translation vector (318 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder47a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder48aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.176*m,-1.762*m,0), // translation vector (321 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder48a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder49aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.265*m,-1.646*m,0), // translation vector (324 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder49a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder50aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.348*m,-1.525*m,0), // translation vector (327 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder50a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder51aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.425*m,-1.4*m,0), // translation vector (330 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder51a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder52aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.495*m,-1.271*m,0), // translation vector (333 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder52a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder53aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(2.558*m,-1.139*m,0), // translation vector (336 deg)
    InnerShieldingCylinderLogical, // logical volume
    "InnerShieldingCylinder53a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0, // copy number
    false); // surface check

InnerShieldingCylinder54aPhysical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(2.614*m,-1.003*m,0), // translation vector (339 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder54a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder55aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.663*m,-0.865*m,0), // translation vector (342 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder55a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder56aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.705*m,-0.725*m,0), // translation vector (345 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder56a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder57aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.739*m,-0.582*m,0), // translation vector (348 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder57a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder58aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.766*m,-0.438*m,0), // translation vector (351 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder58a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder59aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.785*m,-0.293*m,0), // translation vector (354 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder59a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    InnerShieldingCylinder60aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.796*m,-0.147*m,0), // translation vector (357 deg)
        InnerShieldingCylinderLogical,      // logical volume
        "InnerShieldingCylinder60a_Physical", // name
        pMotherLogical,                    // mother volume
        false,                             // unused boolean
        0,                                 // copy number
        false);                             // surface check

    return InnerShieldingCylinder1Physical;
}

// _____//
// construct outer shielding cylinders

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructOuterShieldingCylinder(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set outer shielding cylinder volume
    G4VSolid *OuterShieldingCylinder_Solid = new G4Tubs("OuterShieldingCylinder_Solid", // name
        0.*m, // inner radius
        0.049*m, // outer radius
        5.*m, // z half length
        0, // start phi
        360.*deg); // delta phi

    OuterShieldingCylinderLogical = new G4LogicalVolume(OuterShieldingCylinder_Solid, // solid
        YBCO, // material
        "OuterShieldingCylinder_Logical");//name

    // set color to yellow to match ESA S2RS
    OuterShieldingCylinderLogical->SetVisAttributes(G4Colour(1., 1., 0));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    OuterShieldingCylinderLogical->SetUserLimits(fStepLimit);

    OuterShieldingCylinder1Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.4*m,0,0), // translation vector (0 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder1_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    OuterShieldingCylinder2Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.391*m,0.335*m,0), // translation vector (3 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder2_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    OuterShieldingCylinder3Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.365*m,0.669*m,0), // translation vector (6 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder3_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    OuterShieldingCylinder4Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.321*m,1.001*m,0), // translation vector (9 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder4_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    OuterShieldingCylinder5Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.26*m,1.331*m,0), // translation vector (12 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder5_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    OuterShieldingCylinder6Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(6.182*m,1.656*m,0), // translation vector (15 deg)
        OuterShieldingCylinderLogical, // logical volume

```

```

        "OuterShieldingCylinder6_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder7Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.087*m,1.978*m,0), // translation vector (18 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder7_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder8Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.975*m,2.294*m,0), // translation vector (21 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder8_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder9Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.847*m,2.603*m,0), // translation vector (24 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder9_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder10Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.702*m,2.906*m,0), // translation vector (27 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder10_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder11Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.543*m,3.2*m,0), // translation vector (30 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder11_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder12Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.367*m,3.486*m,0), // translation vector (33 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder12_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder13Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.178*m,3.762*m,0), // translation vector (36 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder13_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder14Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.974*m,4.028*m,0), // translation vector (39 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder14_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder15Physical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(4.756*m,4.282*m,0),// translation vector (42 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder15_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder16Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(4.525*m,4.525*m,0),// translation vector (45 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder16_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder17Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(4.282*m,4.756*m,0),// translation vector (48 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder17_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder18Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(4.028*m,4.974*m,0),// translation vector (51 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder18_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder19Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(3.762*m,5.178*m,0),// translation vector (54 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder19_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder20Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(3.486*m,5.367*m,0),// translation vector (57 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder20_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder21Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(3.2*m,5.543*m,0), // translation vector (60 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder21_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder22Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.906*m,5.702*m,0),// translation vector (63 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder22_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder23Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(2.603*m,5.847*m,0),// translation vector (66 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder23_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

```

```

OuterShieldingCylinder24Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(2.294*m,5.975*m,0), // translation vector (69 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder24_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder25Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.978*m,6.087*m,0), // translation vector (72 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder25_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder26Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.656*m,6.182*m,0), // translation vector (75 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder26_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder27Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.331*m,6.26*m,0), // translation vector (78 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder27_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder28Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(1.001*m,6.321*m,0), // translation vector (81 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder28_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder29Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(0.669*m,6.365*m,0), // translation vector (84 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder29_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder30Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(0.335*m,6.391*m,0), // translation vector (87 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder30_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder31Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(0,6.4*m,0), // translation vector (90 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder31_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder32Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-0.335*m,6.391*m,0), // translation vector (93 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder32_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

```

```

        false, // unused boolean
        0); // copy number

OuterShieldingCylinder33Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-0.669*m,6.365*m,0), // translation vector (96 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder33_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder34Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.001*m,6.321*m,0), // translation vector (99 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder34_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder35Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.331*m,6.260*m,0), // translation vector (102 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder35_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder36Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.656*m,6.182*m,0), // translation vector (105 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder36_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder37Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-1.978*m,6.087*m,0), // translation vector (108 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder37_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder38Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.294*m,5.975*m,0), // translation vector (111 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder38_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder39Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.603*m,5.847*m,0), // translation vector (114 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder39_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder40Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.906*m,5.702*m,0), // translation vector (117 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder40_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder41Physical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-3.2*m,5.543*m,0), // translation vector (120 deg)
    OuterShieldingCylinderLogical, // logical volume

```

```

        "OuterShieldingCylinder41_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder42Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-3.486*m,5.367*m,0), // translation vector (123 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder42_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder43Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-3.762*m,5.178*m,0), // translation vector (126 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder43_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder44Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-4.028*m,4.974*m,0), // translation vector (129 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder44_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder45Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-4.282*m,4.756*m,0), // translation vector (132 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder45_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder46Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-4.525*m,4.525*m,0), // translation vector (135 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder46_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder47Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-4.756*m,4.282*m,0), // translation vector (138 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder47_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder48Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-4.974*m,4.028*m,0), // translation vector (141 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder47_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder49Physical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-5.178*m,3.762*m,0), // translation vector (144 deg)
        OuterShieldingCylinderLogical, // logical volume
        "OuterShieldingCylinder49_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

OuterShieldingCylinder50Physical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(-5.367*m,3.486*m,0),// translation vector (147 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder50_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder51Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-5.543*m,3.2*m,0),// translation vector (150 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder51_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder52Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-5.702*m,2.906*m,0),// translation vector (153 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder52_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder53Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-5.847*m,2.603*m,0),// translation vector (156 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder53_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder54Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-5.975*m,2.294*m,0),// translation vector (159 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder54_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder55Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-6.087*m,1.978*m,0),// translation vector (162 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder55_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder56Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-6.182*m,1.656*m,0),// translation vector (165 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder56_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder57Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-6.260*m,1.331*m,0),// translation vector (168 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder57_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder58Physical = new G4PVPlacement(0,      // rotation matrix
        G4ThreeVector(-6.321*m,1.001*m,0),// translation vector (171 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder58_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

```

```

OuterShieldingCylinder59Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.365*m,0.669*m,0), // translation vector (174 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder59_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder60Physical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.391*m,0.335*m,0), // translation vector (177 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder60_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder1aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.4*m,0,0), // translation vector (180 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder1a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder2aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.391*m,-0.335*m,0), // translation vector (183 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder2a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder3aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.365*m,-0.669*m,0), // translation vector (186 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder3a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder4aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.321*m,-1.001*m,0), // translation vector (189 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder4a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder5aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.26*m,-1.331*m,0), // translation vector (192 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder5a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder6aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.182*m,-1.656*m,0), // translation vector (195 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder6a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

OuterShieldingCylinder7aPhysical = new G4PVPlacement(0,           // rotation matrix
    G4ThreeVector(-6.087*m,-1.978*m,0), // translation vector (198 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder7a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                               // copy number

```

```

        false,                                // unused boolean
        0);                                  // copy number

OuterShieldingCylinder8aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(-5.975*m,-2.294*m,0), // translation vector (201 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder8a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder9aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(-5.847*m,-2.603*m,0), // translation vector (204 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder9a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder10aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-5.702*m,-2.906*m,0), // translation vector (207 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder10a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder11aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-5.543*m,-3.2*m,0), // translation vector (210 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder11a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder12aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-5.367*m,-3.486*m,0), // translation vector (213 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder12a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder13aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-5.178*m,-3.762*m,0), // translation vector (216 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder13a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder14aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-4.974*m,-4.028*m,0), // translation vector (219 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder14a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder15aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-4.756*m,-4.282*m,0), // translation vector (222 deg)
    OuterShieldingCylinderLogical,      // logical volume
    "OuterShieldingCylinder15a_Physical", // name
    pMotherLogical,                    // mother volume
    false,                              // unused boolean
    0);                                // copy number

OuterShieldingCylinder16aPhysical = new G4PVPlacement(0,         // rotation matrix
    G4ThreeVector(-4.525*m,-4.525*m,0), // translation vector (225 deg)
    OuterShieldingCylinderLogical,      // logical volume

```

```

        "OuterShieldingCylinder16a_Physical", // name
        pMotherLogical,                      // mother volume
        false,                               // unused boolean
        0);                                  // copy number

OuterShieldingCylinder17aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-4.282*m,-4.756*m,0), // translation vector (228 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder17a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder18aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-4.028*m,-4.974*m,0), // translation vector (231 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder18a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder19aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-3.762*m,-5.178*m,0), // translation vector (234 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder19a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder20aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-3.486*m,-5.367*m,0), // translation vector (237 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder20a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder21aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-3.2*m,-5.543*m,0), // translation vector (240 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder21a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder22aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.906*m,-5.702*m,0), // translation vector (243 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder22a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder23aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.603*m,-5.847*m,0), // translation vector (246 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder23a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder24aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-2.294*m,-5.975*m,0), // translation vector (249 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder24a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder25aPhysical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(-1.978*m,-6.087*m,0),// translation vector (252 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder25a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder26aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-1.656*m,-6.182*m,0),// translation vector (255 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder26a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder27aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-1.331*m,-6.26*m,0),// translation vector (258 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder27a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder28aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-1.001*m,-6.321*m,0),// translation vector (261 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder28a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder29aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-0.669*m,-6.365*m,0),// translation vector (264 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder29a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder30aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(-0.335*m,-6.391*m,0),// translation vector (267 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder30a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder31aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0,-6.4*m,0),          // translation vector (270 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder31a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder32aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0.335*m,-6.391*m,0),// translation vector (273 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder32a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

OuterShieldingCylinder33aPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0.669*m,-6.365*m,0),// translation vector (276 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder33a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

```

```

OuterShieldingCylinder34aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(1.001*m,-6.321*m,0), // translation vector (279 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder34a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder35aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(1.331*m,-6.260*m,0), // translation vector (282 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder35a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder36aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(1.656*m,-6.182*m,0), // translation vector (285 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder36a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder37aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(1.978*m,-6.087*m,0), // translation vector (288 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder37a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder38aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(2.294*m,-5.975*m,0), // translation vector (291 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder38a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder39aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(2.603*m,-5.847*m,0), // translation vector (294 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder39a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder40aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(2.906*m,-5.702*m,0), // translation vector (297 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder40a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder41aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(3.2*m,-5.543*m,0), // translation vector (300 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder41a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

OuterShieldingCylinder42aPhysical = new G4PVPlacement(0,          // rotation matrix
    G4ThreeVector(3.486*m,-5.367*m,0), // translation vector (303 deg)
    OuterShieldingCylinderLogical,    // logical volume
    "OuterShieldingCylinder42a_Physical", // name
    pMotherLogical,                  // mother volume
    false,                           // unused boolean
    0);                              // copy number

```

```

        false, // unused boolean
        0); // copy number

OuterShieldingCylinder43aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(3.762*m,-5.178*m,0), // translation vector (306 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder43a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder44aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.028*m,-4.974*m,0), // translation vector (309 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder44a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder45aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.282*m,-4.756*m,0), // translation vector (312 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder45a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder46aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.525*m,-4.525*m,0), // translation vector (315 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder46a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder47aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.756*m,-4.282*m,0), // translation vector (318 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder47a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder48aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(4.974*m,-4.028*m,0), // translation vector (321 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder47a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder49aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.178*m,-3.762*m,0), // translation vector (324 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder49a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder50aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.367*m,-3.486*m,0), // translation vector (327 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder50a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder51aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.543*m,-3.2*m,0), // translation vector (330 deg)
    OuterShieldingCylinderLogical, // logical volume

```

```

        "OuterShieldingCylinder51a_Physical", // name
        pMotherLogical,                       // mother volume
        false,                                // unused boolean
        0);                                   // copy number

OuterShieldingCylinder52aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.702*m,-2.906*m,0), // translation vector (333 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder52a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder53aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.847*m,-2.603*m,0), // translation vector (336 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder53a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder54aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(5.975*m,-2.294*m,0), // translation vector (339 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder54a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder55aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.087*m,-1.978*m,0), // translation vector (342 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder55a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder56aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.182*m,-1.656*m,0), // translation vector (345 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder56a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder57aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.260*m,-1.331*m,0), // translation vector (348 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder57a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder58aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.321*m,-1.001*m,0), // translation vector (351 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder58a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder59aPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(6.365*m,-0.669*m,0), // translation vector (354 deg)
    OuterShieldingCylinderLogical, // logical volume
    "OuterShieldingCylinder59a_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

OuterShieldingCylinder60aPhysical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(6.391*m,-0.335*m,0), // translation vector (357 deg)
        OuterShieldingCylinderLogical,      // logical volume
        "OuterShieldingCylinder60a_Physical", // name
        pMotherLogical,                     // mother volume
        false,                              // unused boolean
        0);                                // copy number

    return OuterShieldingCylinder1Physical;
}

// _____//

// construct upper shielding toroids

G4VPhysicalVolume * DetectorConstruction::ConstructUpperShieldingTorus(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set upper shielding torus volume
    G4VSolid *UpperShieldingTorus_Solid = new G4Torus("UpperShieldingTorus_Solid", // name
        0.*m, // inner toroid cross-sectional radius
        0.049*m, // outer toroid cross-sectional radius
        1.8*m, // radius of toroid
        0, // start phi
        180.*deg); // delta phi

    UpperShieldingTorusLogical = new G4LogicalVolume(UpperShieldingTorus_Solid, // solid volume
        YBCO, // material
        "UpperShieldingTorus_Logical"); // name

    // set color to yellow to match ESA S2RS
    UpperShieldingTorusLogical->SetVisAttributes(G4Colour(1., 1., 0));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    UpperShieldingTorusLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix* rm1 = new G4RotationMatrix();
    G4RotationMatrix* rm2 = new G4RotationMatrix();
    G4RotationMatrix* rm3 = new G4RotationMatrix();
    G4RotationMatrix* rm4 = new G4RotationMatrix();
    G4RotationMatrix* rm5 = new G4RotationMatrix();
    G4RotationMatrix* rm6 = new G4RotationMatrix();
    G4RotationMatrix* rm7 = new G4RotationMatrix();
    G4RotationMatrix* rm8 = new G4RotationMatrix();
    G4RotationMatrix* rm9 = new G4RotationMatrix();
    G4RotationMatrix* rm10 = new G4RotationMatrix();
    G4RotationMatrix* rm11 = new G4RotationMatrix();
    G4RotationMatrix* rm12 = new G4RotationMatrix();
    G4RotationMatrix* rm13 = new G4RotationMatrix();
    G4RotationMatrix* rm14 = new G4RotationMatrix();
    G4RotationMatrix* rm15 = new G4RotationMatrix();
    G4RotationMatrix* rm16 = new G4RotationMatrix();
    G4RotationMatrix* rm17 = new G4RotationMatrix();
    G4RotationMatrix* rm18 = new G4RotationMatrix();
    G4RotationMatrix* rm19 = new G4RotationMatrix();
    G4RotationMatrix* rm20 = new G4RotationMatrix();
    G4RotationMatrix* rm21 = new G4RotationMatrix();
    G4RotationMatrix* rm22 = new G4RotationMatrix();
    G4RotationMatrix* rm23 = new G4RotationMatrix();
    G4RotationMatrix* rm24 = new G4RotationMatrix();
    G4RotationMatrix* rm25 = new G4RotationMatrix();
    G4RotationMatrix* rm26 = new G4RotationMatrix();
    G4RotationMatrix* rm27 = new G4RotationMatrix();
    G4RotationMatrix* rm28 = new G4RotationMatrix();

```

[illegible]

```

G4RotationMatrix* rm39a = new G4RotationMatrix();
G4RotationMatrix* rm40a = new G4RotationMatrix();
G4RotationMatrix* rm41a = new G4RotationMatrix();
G4RotationMatrix* rm42a = new G4RotationMatrix();
G4RotationMatrix* rm43a = new G4RotationMatrix();
G4RotationMatrix* rm44a = new G4RotationMatrix();
G4RotationMatrix* rm45a = new G4RotationMatrix();
G4RotationMatrix* rm46a = new G4RotationMatrix();
G4RotationMatrix* rm47a = new G4RotationMatrix();
G4RotationMatrix* rm48a = new G4RotationMatrix();
G4RotationMatrix* rm49a = new G4RotationMatrix();
G4RotationMatrix* rm50a = new G4RotationMatrix();
G4RotationMatrix* rm51a = new G4RotationMatrix();
G4RotationMatrix* rm52a = new G4RotationMatrix();
G4RotationMatrix* rm53a = new G4RotationMatrix();
G4RotationMatrix* rm54a = new G4RotationMatrix();
G4RotationMatrix* rm55a = new G4RotationMatrix();
G4RotationMatrix* rm56a = new G4RotationMatrix();
G4RotationMatrix* rm57a = new G4RotationMatrix();
G4RotationMatrix* rm58a = new G4RotationMatrix();
G4RotationMatrix* rm59a = new G4RotationMatrix();
G4RotationMatrix* rm60a = new G4RotationMatrix();

rm1-> rotateX(-90*deg);
rm1-> rotateY(0*deg);
UpperShieldingTorus1Physical = new G4PVPlacement(rm1,           // rotation matrix
G4ThreeVector(4.6*m,0,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus1_Physical", // name
pMotherLogical,           // mother volume
false,                    // unused boolean
0);                        // copy number

rm2-> rotateX(-90*deg);
rm2-> rotateY(-3*deg);
UpperShieldingTorus2Physical = new G4PVPlacement(rm2,           // rotation matrix
G4ThreeVector(4.594*m,0.241*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus2_Physical", // name
pMotherLogical,           // mother volume
false,                    // unused boolean
0);                        // copy number

rm3-> rotateX(-90*deg);
rm3-> rotateY(-6*deg);
UpperShieldingTorus3Physical = new G4PVPlacement(rm3,           // rotation matrix
G4ThreeVector(4.575*m,0.481*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus3_Physical", // name
pMotherLogical,           // mother volume
false,                    // unused boolean
0);                        // copy number

rm4-> rotateX(-90*deg);
rm4-> rotateY(-9*deg);
UpperShieldingTorus4Physical = new G4PVPlacement(rm4,           // rotation matrix
G4ThreeVector(4.543*m,0.720*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus4_Physical", // name
pMotherLogical,           // mother volume
false,                    // unused boolean
0);                        // copy number

rm5-> rotateX(-90*deg);
rm5-> rotateY(-12*deg);
UpperShieldingTorus5Physical = new G4PVPlacement(rm5,           // rotation matrix
G4ThreeVector(4.499*m,0.956*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus5_Physical", // name
pMotherLogical,           // mother volume

```

```

        false,                                // unused boolean
        0);                                  // copy number

rm6-> rotateX(-90*deg);
rm6-> rotateY(-15.*deg);
UpperShieldingTorus6Physical = new G4PVPlacement(rm6,                // rotation matrix
G4ThreeVector(4.443*m,1.191*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus6_Physical",       // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm7-> rotateX(-90*deg);
rm7-> rotateY(-18*deg);
UpperShieldingTorus7Physical = new G4PVPlacement(rm7,                // rotation matrix
G4ThreeVector(4.375*m,1.421*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus7_Physical",       // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm8-> rotateX(-90*deg);
rm8-> rotateY(-21*deg);
UpperShieldingTorus8Physical = new G4PVPlacement(rm8,                // rotation matrix
G4ThreeVector(4.294*m,1.648*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus8_Physical",       // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm9-> rotateX(-90*deg);
rm9-> rotateY(-24*deg);
UpperShieldingTorus9Physical = new G4PVPlacement(rm9,                // rotation matrix
G4ThreeVector(4.202*m,1.871*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus9_Physical",       // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm10-> rotateX(-90*deg);
rm10-> rotateY(-27*deg);
UpperShieldingTorus10Physical = new G4PVPlacement(rm10,              // rotation matrix
G4ThreeVector(4.099*m,2.088*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus10_Physical",      // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm11-> rotateX(-90*deg);
rm11-> rotateY(-30*deg);
UpperShieldingTorus11Physical = new G4PVPlacement(rm11,              // rotation matrix
G4ThreeVector(3.984*m,2.3*m,5.*m),      // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus11_Physical",      // name
pMotherLogical,                        // mother volume
false,                                // unused boolean
0);                                  // copy number

rm12-> rotateX(-90*deg);
rm12-> rotateY(-33*deg);
UpperShieldingTorus12Physical = new G4PVPlacement(rm12,              // rotation matrix
G4ThreeVector(3.858*m,2.505*m,5.*m),    // translation vector
UpperShieldingTorusLogical,            // logical volume
"UpperShieldingTorus12_Physical",      // name
pMotherLogical,                        // mother volume

```

```

                                false,                                // unused boolean
                                0);                                // copy number

rm13-> rotateX(-90*deg);
rm13-> rotateY(-36*deg);
UpperShieldingTorus13Physical = new G4PVPlacement(rm13,                // rotation matrix
G4ThreeVector(3.721*m,2.704*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus13_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm14-> rotateX(-90*deg);
rm14-> rotateY(-39*deg);
UpperShieldingTorus14Physical = new G4PVPlacement(rm14,                // rotation matrix
G4ThreeVector(3.575*m,2.895*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus14_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm15-> rotateX(-90*deg);
rm15-> rotateY(-42*deg);
UpperShieldingTorus15Physical = new G4PVPlacement(rm15,                // rotation matrix
G4ThreeVector(3.418*m,3.078*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus15_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm16-> rotateX(-90*deg);
rm16-> rotateY(-45.*deg);
UpperShieldingTorus16Physical = new G4PVPlacement(rm16,                // rotation matrix
G4ThreeVector(3.253*m,3.253*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus16_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm17-> rotateX(-90*deg);
rm17-> rotateY(-48*deg);
UpperShieldingTorus17Physical = new G4PVPlacement(rm17,                // rotation matrix
G4ThreeVector(3.078*m,3.418*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus17_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm18-> rotateX(-90*deg);
rm18-> rotateY(-51*deg);
UpperShieldingTorus18Physical = new G4PVPlacement(rm18,                // rotation matrix
G4ThreeVector(2.895*m,3.575*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus18_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                                // copy number

rm19-> rotateX(-90*deg);
rm19-> rotateY(-54*deg);
UpperShieldingTorus19Physical = new G4PVPlacement(rm19,                // rotation matrix
G4ThreeVector(2.704*m,3.721*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus19_Physical", // name
pMotherLogical,                  // mother volume

```

```

false, // unused boolean
0);    // copy number

rm20-> rotateX(-90*deg);
rm20-> rotateY(-57*deg);
UpperShieldingTorus20Physical = new G4PVPlacement(rm20, // rotation matrix
G4ThreeVector(2.505*m,3.858*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus20_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm21-> rotateX(-90*deg);
rm21-> rotateY(-60*deg);
UpperShieldingTorus21Physical = new G4PVPlacement(rm21, // rotation matrix
G4ThreeVector(2.3*m,3.984*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus21_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm22-> rotateX(-90*deg);
rm22-> rotateY(-63*deg);
UpperShieldingTorus22Physical = new G4PVPlacement(rm22, // rotation matrix
G4ThreeVector(2.088*m,4.099*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus22_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm23-> rotateX(-90*deg);
rm23-> rotateY(-66*deg);
UpperShieldingTorus23Physical = new G4PVPlacement(rm23, // rotation matrix
G4ThreeVector(1.871*m,4.202*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus23_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm24-> rotateX(-90*deg);
rm24-> rotateY(-69*deg);
UpperShieldingTorus24Physical = new G4PVPlacement(rm24, // rotation matrix
G4ThreeVector(1.648*m,4.294*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus24_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm25-> rotateX(-90*deg);
rm25-> rotateY(-72*deg);
UpperShieldingTorus25Physical = new G4PVPlacement(rm25, // rotation matrix
G4ThreeVector(1.421*m,4.375*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus25_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm26-> rotateX(-90*deg);
rm26-> rotateY(-75*deg);
UpperShieldingTorus26Physical = new G4PVPlacement(rm26, // rotation matrix
G4ThreeVector(1.191*m,4.443*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus26_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm27-> rotateX(-90*deg);
rm27-> rotateY(-78*deg);
UpperShieldingTorus27Physical = new G4PVPlacement(rm27, // rotation matrix
G4ThreeVector(0.956*m,4.499*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus27_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm28-> rotateX(-90*deg);
rm28-> rotateY(-81*deg);
UpperShieldingTorus28Physical = new G4PVPlacement(rm28, // rotation matrix
G4ThreeVector(0.72*m,4.543*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus28_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm29-> rotateX(-90*deg);
rm29-> rotateY(-84*deg);
UpperShieldingTorus29Physical = new G4PVPlacement(rm29, // rotation matrix
G4ThreeVector(0.481*m,4.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus29_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm30-> rotateX(-90*deg);
rm30-> rotateY(-87*deg);
UpperShieldingTorus30Physical = new G4PVPlacement(rm30, // rotation matrix
G4ThreeVector(0.241*m,4.594*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus30_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm31-> rotateX(-90*deg);
rm31-> rotateY(-90*deg);
UpperShieldingTorus31Physical = new G4PVPlacement(rm31, // rotation matrix
G4ThreeVector(0,4.6*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus31_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm32-> rotateX(-90*deg);
rm32-> rotateY(-93*deg);
UpperShieldingTorus32Physical = new G4PVPlacement(rm32, // rotation matrix
G4ThreeVector(-0.241*m,4.594*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus32_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm33-> rotateX(-90*deg);
rm33-> rotateY(-96*deg);
UpperShieldingTorus33Physical = new G4PVPlacement(rm33, // rotation matrix
G4ThreeVector(-0.481*m,4.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus33_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm34-> rotateX(-90*deg);
rm34-> rotateY(-99*deg);
UpperShieldingTorus34Physical = new G4PVPlacement(rm34, // rotation matrix
G4ThreeVector(-0.720*m,4.543*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus34_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm35-> rotateX(-90*deg);
rm35-> rotateY(-102*deg);
UpperShieldingTorus35Physical = new G4PVPlacement(rm35, // rotation matrix
G4ThreeVector(-0.956*m,4.499*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus35_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm36-> rotateX(-90*deg);
rm36-> rotateY(-105*deg);
UpperShieldingTorus36Physical = new G4PVPlacement(rm36, // rotation matrix
G4ThreeVector(-1.191*m,4.443*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus36_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm37-> rotateX(-90*deg);
rm37-> rotateY(-108*deg);
UpperShieldingTorus37Physical = new G4PVPlacement(rm37, // rotation matrix
G4ThreeVector(-1.421*m,4.375*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus37_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm38-> rotateX(-90*deg);
rm38-> rotateY(-111*deg);
UpperShieldingTorus38Physical = new G4PVPlacement(rm38, // rotation matrix
G4ThreeVector(-1.648*m,4.294*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus38_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm39-> rotateX(-90*deg);
rm39-> rotateY(-114*deg);
UpperShieldingTorus39Physical = new G4PVPlacement(rm39, // rotation matrix
G4ThreeVector(-1.871*m,4.202*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus39_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm40-> rotateX(-90*deg);
rm40-> rotateY(-117*deg);
UpperShieldingTorus40Physical = new G4PVPlacement(rm40, // rotation matrix
G4ThreeVector(-2.089*m,4.099*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus40_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm41-> rotateX(-90*deg);
rm41-> rotateY(-120*deg);
UpperShieldingTorus41Physical = new G4PVPlacement(rm41, // rotation matrix
G4ThreeVector(-2.3*m,3.984*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus41_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm42-> rotateX(-90*deg);
rm42-> rotateY(-123*deg);
UpperShieldingTorus42Physical = new G4PVPlacement(rm42, // rotation matrix
G4ThreeVector(-2.505*m,3.858*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus42_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm43-> rotateX(-90*deg);
rm43-> rotateY(-126*deg);
UpperShieldingTorus43Physical = new G4PVPlacement(rm43, // rotation matrix
G4ThreeVector(-2.704*m,3.721*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus43_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm44-> rotateX(-90*deg);
rm44-> rotateY(-129*deg);
UpperShieldingTorus44Physical = new G4PVPlacement(rm44, // rotation matrix
G4ThreeVector(-2.895*m,3.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus44_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm45-> rotateX(-90*deg);
rm45-> rotateY(-132*deg);
UpperShieldingTorus45Physical = new G4PVPlacement(rm45, // rotation matrix
G4ThreeVector(-3.078*m,3.418*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus45_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm46-> rotateX(-90*deg);
rm46-> rotateY(-135*deg);
UpperShieldingTorus46Physical = new G4PVPlacement(rm46, // rotation matrix
G4ThreeVector(-3.253*m,3.253*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus46_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0);    // copy number

rm47-> rotateX(-90*deg);
rm47-> rotateY(-138*deg);
UpperShieldingTorus47Physical = new G4PVPlacement(rm47, // rotation matrix
G4ThreeVector(-3.418*m,3.078*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus47_Physical", // name
pMotherLogical, // mother volume

```

```

        false,                                // unused boolean
        0);                                    // copy number

rm48-> rotateX(-90*deg);
rm48-> rotateY(-141*deg);
UpperShieldingTorus48Physical = new G4PVPlacement(rm48,                // rotation matrix
G4ThreeVector(-3.575*m,2.895*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus48_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm49-> rotateX(-90*deg);
rm49-> rotateY(-144*deg);
UpperShieldingTorus49Physical = new G4PVPlacement(rm49,                // rotation matrix
G4ThreeVector(-3.721*m,2.704*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus49_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm50-> rotateX(-90*deg);
rm50-> rotateY(-147*deg);
UpperShieldingTorus50Physical = new G4PVPlacement(rm50,                // rotation matrix
G4ThreeVector(-3.858*m,2.505*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus50_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm51-> rotateX(-90*deg);
rm51-> rotateY(-150*deg);
UpperShieldingTorus51Physical = new G4PVPlacement(rm51,                // rotation matrix
G4ThreeVector(-3.984*m,2.3*m,5.*m),    // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus51_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm52-> rotateX(-90*deg);
rm52-> rotateY(-153*deg);
UpperShieldingTorus52Physical = new G4PVPlacement(rm52,                // rotation matrix
G4ThreeVector(-4.099*m,2.088*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus52_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm53-> rotateX(-90*deg);
rm53-> rotateY(-156*deg);
UpperShieldingTorus53Physical = new G4PVPlacement(rm53,                // rotation matrix
G4ThreeVector(-4.202*m,1.871*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus53_Physical", // name
pMotherLogical,                  // mother volume
false,                            // unused boolean
0);                               // copy number

rm54-> rotateX(-90*deg);
rm54-> rotateY(-159*deg);
UpperShieldingTorus54Physical = new G4PVPlacement(rm54,                // rotation matrix
G4ThreeVector(-4.294*m,1.648*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus54_Physical", // name
pMotherLogical,                  // mother volume

```

```

        false,                                // unused boolean
        0);                                    // copy number

rm55-> rotateX(-90*deg);
rm55-> rotateY(-162*deg);
UpperShieldingTorus55Physical = new G4PVPlacement(rm55,           // rotation matrix
G4ThreeVector(-4.375*m,1.421*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus55_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm56-> rotateX(-90*deg);
rm56-> rotateY(-165*deg);
UpperShieldingTorus56Physical = new G4PVPlacement(rm56,           // rotation matrix
G4ThreeVector(-4.443*m,1.191*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus56_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm57-> rotateX(-90*deg);
rm57-> rotateY(-168*deg);
UpperShieldingTorus57Physical = new G4PVPlacement(rm57,           // rotation matrix
G4ThreeVector(-4.499*m,0.956*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus57_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm58-> rotateX(-90*deg);
rm58-> rotateY(-171*deg);
UpperShieldingTorus58Physical = new G4PVPlacement(rm58,           // rotation matrix
G4ThreeVector(-4.543*m,0.720*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus58_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm59-> rotateX(-90*deg);
rm59-> rotateY(-174*deg);
UpperShieldingTorus59Physical = new G4PVPlacement(rm59,           // rotation matrix
G4ThreeVector(-4.575*m,0.481*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus59_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm60-> rotateX(-90*deg);
rm60-> rotateY(-177*deg);
UpperShieldingTorus60Physical = new G4PVPlacement(rm60,           // rotation matrix
G4ThreeVector(-4.594*m,0.241*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus60_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm1a-> rotateX(-90*deg);
rm1a-> rotateY(-180*deg);
UpperShieldingTorus1aPhysical = new G4PVPlacement(rm1a,           // rotation matrix
G4ThreeVector(-4.6*m,0,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus1a_Physical", // name
pMotherLogical, // mother volume

```

```

        false, // unused boolean
        0); // copy number

rm2a-> rotateX(-90*deg);
rm2a-> rotateY(-183*deg);
UpperShieldingTorus2aPhysical = new G4PVPlacement(rm2a, // rotation matrix
G4ThreeVector(-4.594*m,-0.241*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus2a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm3a-> rotateX(-90*deg);
rm3a-> rotateY(-186*deg);
UpperShieldingTorus3aPhysical = new G4PVPlacement(rm3a, // rotation matrix
G4ThreeVector(-4.575*m,-0.481*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus3a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm4a-> rotateX(-90*deg);
rm4a-> rotateY(-189*deg);
UpperShieldingTorus4aPhysical = new G4PVPlacement(rm4a, // rotation matrix
G4ThreeVector(-4.543*m,-0.720*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus4a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm5a-> rotateX(-90*deg);
rm5a-> rotateY(-192*deg);
UpperShieldingTorus5aPhysical = new G4PVPlacement(rm5a, // rotation matrix
G4ThreeVector(-4.499*m,-0.956*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus5a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm6a-> rotateX(-90*deg);
rm6a-> rotateY(-195*deg);
UpperShieldingTorus6aPhysical = new G4PVPlacement(rm6a, // rotation matrix
G4ThreeVector(-4.443*m,-1.191*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus6a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm7a-> rotateX(-90*deg);
rm7a-> rotateY(-198*deg);
UpperShieldingTorus7aPhysical = new G4PVPlacement(rm7a, // rotation matrix
G4ThreeVector(-4.375*m,-1.421*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus7a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm8a-> rotateX(-90*deg);
rm8a-> rotateY(-201*deg);
UpperShieldingTorus8aPhysical = new G4PVPlacement(rm8a, // rotation matrix
G4ThreeVector(-4.294*m,-1.648*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus8a_Physical", // name
pMotherLogical, // mother volume

```

```

                                false,                                // unused boolean
                                0);                                // copy number

rm9a-> rotateX(-90*deg);
rm9a-> rotateY(-204*deg);
UpperShieldingTorus9aPhysical = new G4PVPlacement(rm9a,                // rotation matrix
G4ThreeVector(-4.202*m,-1.871*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus9a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm10a-> rotateX(-90*deg);
rm10a-> rotateY(-207*deg);
UpperShieldingTorus10aPhysical = new G4PVPlacement(rm10a,             // rotation matrix
G4ThreeVector(-4.099*m,-2.088*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus10a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm11a-> rotateX(-90*deg);
rm11a-> rotateY(-210*deg);
UpperShieldingTorus11aPhysical = new G4PVPlacement(rm11a,             // rotation matrix
G4ThreeVector(-3.984*m,-2.3*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus11a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm12a-> rotateX(-90*deg);
rm12a-> rotateY(-213*deg);
UpperShieldingTorus12aPhysical = new G4PVPlacement(rm12a,             // rotation matrix
G4ThreeVector(-3.858*m,-2.505*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus12a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm13a-> rotateX(-90*deg);
rm13a-> rotateY(-216*deg);
UpperShieldingTorus13aPhysical = new G4PVPlacement(rm13a,             // rotation matrix
G4ThreeVector(-3.721*m,-2.704*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus13a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm14a-> rotateX(-90*deg);
rm14a-> rotateY(-219*deg);
UpperShieldingTorus14aPhysical = new G4PVPlacement(rm14a,             // rotation matrix
G4ThreeVector(-3.575*m,-2.895*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus14a_Physical", // name
pMotherLogical,                 // mother volume
false,                          // unused boolean
0);                              // copy number

rm15a-> rotateX(-90*deg);
rm15a-> rotateY(-222*deg);
UpperShieldingTorus15aPhysical = new G4PVPlacement(rm15a,             // rotation matrix
G4ThreeVector(-3.418*m,-3.078*m,5.*m), // translation vector
UpperShieldingTorusLogical,      // logical volume
"UpperShieldingTorus15a_Physical", // name
pMotherLogical,                 // mother volume

```

```

        false, // unused boolean
        0); // copy number

rm16a-> rotateX(-90*deg);
rm16a-> rotateY(-225*deg);
UpperShieldingTorus16aPhysical = new G4PVPlacement(rm16a, // rotation matrix
G4ThreeVector(-3.253*m,-3.253*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus16a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm17a-> rotateX(-90*deg);
rm17a-> rotateY(-228*deg);
UpperShieldingTorus17aPhysical = new G4PVPlacement(rm17a, // rotation matrix
G4ThreeVector(-3.078*m,-3.418*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus17a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm18a-> rotateX(-90*deg);
rm18a-> rotateY(-231*deg);
UpperShieldingTorus18aPhysical = new G4PVPlacement(rm18a, // rotation matrix
G4ThreeVector(-2.895*m,-3.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus18a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm19a-> rotateX(-90*deg);
rm19a-> rotateY(-234*deg);
UpperShieldingTorus19aPhysical = new G4PVPlacement(rm19a, // rotation matrix
G4ThreeVector(-2.704*m,-3.721*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus19a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm20a-> rotateX(-90*deg);
rm20a-> rotateY(-237*deg);
UpperShieldingTorus20aPhysical = new G4PVPlacement(rm20a, // rotation matrix
G4ThreeVector(-2.505*m,-3.858*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus20a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm21a-> rotateX(-90*deg);
rm21a-> rotateY(-240*deg);
UpperShieldingTorus21aPhysical = new G4PVPlacement(rm21a, // rotation matrix
G4ThreeVector(-2.3*m,-3.984*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus21a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm22a-> rotateX(-90*deg);
rm22a-> rotateY(-243*deg);
UpperShieldingTorus22aPhysical = new G4PVPlacement(rm22a, // rotation matrix
G4ThreeVector(-2.088*m,-4.099*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus22a_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm23a-> rotateX(-90*deg);
rm23a-> rotateY(-246*deg);
UpperShieldingTorus23aPhysical = new G4PVPlacement(rm23a, // rotation matrix
G4ThreeVector(-1.871*m,-4.202*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus23a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm24a-> rotateX(-90*deg);
rm24a-> rotateY(-249*deg);
UpperShieldingTorus24aPhysical = new G4PVPlacement(rm24a, // rotation matrix
G4ThreeVector(-1.648*m,-4.294*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus24a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm25a-> rotateX(-90*deg);
rm25a-> rotateY(-252*deg);
UpperShieldingTorus25aPhysical = new G4PVPlacement(rm25a, // rotation matrix
G4ThreeVector(-1.421*m,-4.375*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus25a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm26a-> rotateX(-90*deg);
rm26a-> rotateY(-255*deg);
UpperShieldingTorus26aPhysical = new G4PVPlacement(rm26a, // rotation matrix
G4ThreeVector(-1.191*m,-4.443*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus26a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm27a-> rotateX(-90*deg);
rm27a-> rotateY(-258*deg);
UpperShieldingTorus27aPhysical = new G4PVPlacement(rm27a, // rotation matrix
G4ThreeVector(-0.956*m,-4.499*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus27a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm28a-> rotateX(-90*deg);
rm28a-> rotateY(-261*deg);
UpperShieldingTorus28aPhysical = new G4PVPlacement(rm28a, // rotation matrix
G4ThreeVector(-0.72*m,-4.543*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus28a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm29a-> rotateX(-90*deg);
rm29a-> rotateY(-264*deg);
UpperShieldingTorus29aPhysical = new G4PVPlacement(rm29a, // rotation matrix
G4ThreeVector(-0.481*m,-4.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus29a_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm30a-> rotateX(-90*deg);
rm30a-> rotateY(-267*deg);
UpperShieldingTorus30aPhysical = new G4PVPlacement(rm30a, // rotation matrix
G4ThreeVector(-0.241*m,-4.594*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus30a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm31a-> rotateX(-90*deg);
rm31a-> rotateY(-270*deg);
UpperShieldingTorus31aPhysical = new G4PVPlacement(rm31a, // rotation matrix
G4ThreeVector(0,-4.6*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus31a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm32a-> rotateX(-90*deg);
rm32a-> rotateY(-273*deg);
UpperShieldingTorus32aPhysical = new G4PVPlacement(rm32a, // rotation matrix
G4ThreeVector(0.741*m,-4.594*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus32a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm33a-> rotateX(-90*deg);
rm33a-> rotateY(-276*deg);
UpperShieldingTorus33aPhysical = new G4PVPlacement(rm33a, // rotation matrix
G4ThreeVector(0.481*m,-4.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus33a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm34a-> rotateX(-90*deg);
rm34a-> rotateY(-279*deg);
UpperShieldingTorus34aPhysical = new G4PVPlacement(rm34a, // rotation matrix
G4ThreeVector(0.720*m,-4.543*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus34a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm35a-> rotateX(-90*deg);
rm35a-> rotateY(-282*deg);
UpperShieldingTorus35aPhysical = new G4PVPlacement(rm35a, // rotation matrix
G4ThreeVector(0.956*m,-4.499*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus35a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm36a-> rotateX(-90*deg);
rm36a-> rotateY(-285*deg);
UpperShieldingTorus36aPhysical = new G4PVPlacement(rm36a, // rotation matrix
G4ThreeVector(1.191*m,-4.443*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus36a_Physical", // name
pMotherLogical, // mother volume

```

```

        false, // unused boolean
        0); // copy number

rm37a-> rotateX(-90*deg);
rm37a-> rotateY(-288*deg);
UpperShieldingTorus37aPhysical = new G4PVPlacement(rm37a, // rotation matrix
G4ThreeVector(1.421*m,-4.375*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus37a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm38a-> rotateX(-90*deg);
rm38a-> rotateY(-291*deg);
UpperShieldingTorus38aPhysical = new G4PVPlacement(rm38a, // rotation matrix
G4ThreeVector(1.648*m,-4.294*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus38a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm39a-> rotateX(-90*deg);
rm39a-> rotateY(-294*deg);
UpperShieldingTorus39aPhysical = new G4PVPlacement(rm39a, // rotation matrix
G4ThreeVector(1.871*m,-4.202*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus39a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm40a-> rotateX(-90*deg);
rm40a-> rotateY(-297*deg);
UpperShieldingTorus40aPhysical = new G4PVPlacement(rm40a, // rotation matrix
G4ThreeVector(2.089*m,-4.099*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus40a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm41a-> rotateX(-90*deg);
rm41a-> rotateY(-300*deg);
UpperShieldingTorus41aPhysical = new G4PVPlacement(rm41a, // rotation matrix
G4ThreeVector(2.3*m,-3.984*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus41a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm42a-> rotateX(-90*deg);
rm42a-> rotateY(-303*deg);
UpperShieldingTorus42aPhysical = new G4PVPlacement(rm42a, // rotation matrix
G4ThreeVector(2.505*m,-3.858*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus42a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm43a-> rotateX(-90*deg);
rm43a-> rotateY(-306*deg);
UpperShieldingTorus43aPhysical = new G4PVPlacement(rm43a, // rotation matrix
G4ThreeVector(2.704*m,-3.721*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus43a_Physical", // name
pMotherLogical, // mother volume

```

```

false, // unused boolean
0);    // copy number

rm44a-> rotateX(-90*deg);
rm44a-> rotateY(-309*deg);
UpperShieldingTorus44aPhysical = new G4PVPlacement(rm44a, // rotation matrix
G4ThreeVector(2.895*m,-3.575*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus44a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm45a-> rotateX(-90*deg);
rm45a-> rotateY(-312*deg);
UpperShieldingTorus45aPhysical = new G4PVPlacement(rm45a, // rotation matrix
G4ThreeVector(3.078*m,-3.418*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus45a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm46a-> rotateX(-90*deg);
rm46a-> rotateY(-315*deg);
UpperShieldingTorus46aPhysical = new G4PVPlacement(rm46a, // rotation matrix
G4ThreeVector(3.253*m,-3.253*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus46a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm47a-> rotateX(-90*deg);
rm47a-> rotateY(-318*deg);
UpperShieldingTorus47aPhysical = new G4PVPlacement(rm47a, // rotation matrix
G4ThreeVector(3.418*m,-3.078*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus47a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm48a-> rotateX(-90*deg);
rm48a-> rotateY(-321*deg);
UpperShieldingTorus48aPhysical = new G4PVPlacement(rm48a, // rotation matrix
G4ThreeVector(3.575*m,-2.895*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus48a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm49a-> rotateX(-90*deg);
rm49a-> rotateY(-324*deg);
UpperShieldingTorus49aPhysical = new G4PVPlacement(rm49a, // rotation matrix
G4ThreeVector(3.721*m,-2.704*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus49a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm50a-> rotateX(-90*deg);
rm50a-> rotateY(-327*deg);
UpperShieldingTorus50aPhysical = new G4PVPlacement(rm50a, // rotation matrix
G4ThreeVector(3.858*m,-2.505*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus50a_Physical", // name
pMotherLogical, // mother volume

```

```

        false, // unused boolean
        0); // copy number

rm51a-> rotateX(-90*deg);
rm51a-> rotateY(-330*deg);
UpperShieldingTorus51aPhysical = new G4PVPlacement(rm51a, // rotation matrix
G4ThreeVector(3.984*m,-2.3*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus51a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm52a-> rotateX(-90*deg);
rm52a-> rotateY(-333*deg);
UpperShieldingTorus52aPhysical = new G4PVPlacement(rm52a, // rotation matrix
G4ThreeVector(4.099*m,-2.088*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus52a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm53a-> rotateX(-90*deg);
rm53a-> rotateY(-336*deg);
UpperShieldingTorus53aPhysical = new G4PVPlacement(rm53a, // rotation matrix
G4ThreeVector(4.202*m,-1.871*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus53a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm54a-> rotateX(-90*deg);
rm54a-> rotateY(-339*deg);
UpperShieldingTorus54aPhysical = new G4PVPlacement(rm54a, // rotation matrix
G4ThreeVector(4.294*m,-1.648*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus54a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm55a-> rotateX(-90*deg);
rm55a-> rotateY(-342*deg);
UpperShieldingTorus55aPhysical = new G4PVPlacement(rm55a, // rotation matrix
G4ThreeVector(4.375*m,-1.421*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus55a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm56a-> rotateX(-90*deg);
rm56a-> rotateY(-345*deg);
UpperShieldingTorus56aPhysical = new G4PVPlacement(rm56a, // rotation matrix
G4ThreeVector(4.443*m,-1.191*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus56a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm57a-> rotateX(-90*deg);
rm57a-> rotateY(-348*deg);
UpperShieldingTorus57aPhysical = new G4PVPlacement(rm57a, // rotation matrix
G4ThreeVector(4.499*m,-0.956*m,5.*m), // translation vector
UpperShieldingTorusLogical, // logical volume
"UpperShieldingTorus57a_Physical", // name

```

```

        pMotherLogical,                // mother volume
        false,                        // unused boolean
        0);                          // copy number

rm58a-> rotateX(-90*deg);
rm58a-> rotateY(-351*deg);
UpperShieldingTorus58aPhysical = new G4PVPlacement(rm58a,                // rotation matrix
    G4ThreeVector(4.543*m,-0.720*m,5.*m), // translation vector
    UpperShieldingTorusLogical,          // logical volume
    "UpperShieldingTorus58a_Physical",    // name
    pMotherLogical,                     // mother volume
    false,                              // unused boolean
    0);                                // copy number

rm59a-> rotateX(-90*deg);
rm59a-> rotateY(-354*deg);
UpperShieldingTorus59aPhysical = new G4PVPlacement(rm59a,                // rotation matrix
    G4ThreeVector(4.575*m,-0.481*m,5.*m), // translation vector
    UpperShieldingTorusLogical,          // logical volume
    "UpperShieldingTorus59a_Physical",    // name
    pMotherLogical,                     // mother volume
    false,                              // unused boolean
    0);                                // copy number

rm60a-> rotateX(-90*deg);
rm60a-> rotateY(-357*deg);
UpperShieldingTorus60aPhysical = new G4PVPlacement(rm60a,                // rotation matrix
    G4ThreeVector(4.594*m,-0.241*m,5.*m), // translation vector
    UpperShieldingTorusLogical,          // logical volume
    "UpperShieldingTorus60a_Physical",    // name
    pMotherLogical,                     // mother volume
    false,                              // unused boolean
    0);                                // copy number

return UpperShieldingTorus1Physical;
}

// _____//
// construct lower shielding toroids

G4VPhysicalVolume * DetectorConstruction::ConstructLowerShieldingTorus(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *YBCO = G4Material::GetMaterial("YBCO");

    // set lower shielding torus volume
    G4VSolid *LowerShieldingTorus_Solid = new G4Torus("LowerShieldingTorus_Solid", // name
        0.*m, // inner toroid cross-sectional radius
        0.049*m, // outer toroid cross-sectional radius
        1.8*m, // radius of toroid
        0, // start phi
        180.*deg); // delta phi

    LowerShieldingTorusLogical = new G4LogicalVolume(LowerShieldingTorus_Solid, // solid volume
        YBCO, // material
        "LowerShieldingTorus_Logical"); // name

    // set color to yellow to match ESA S2RS
    LowerShieldingTorusLogical->SetVisAttributes(G4Colour(1., 1., 0));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LowerShieldingTorusLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix* rm1 = new G4RotationMatrix();

```

[illegible]

```

G4RotationMatrix* rm12a = new G4RotationMatrix();
G4RotationMatrix* rm13a = new G4RotationMatrix();
G4RotationMatrix* rm14a = new G4RotationMatrix();
G4RotationMatrix* rm15a = new G4RotationMatrix();
G4RotationMatrix* rm16a = new G4RotationMatrix();
G4RotationMatrix* rm17a = new G4RotationMatrix();
G4RotationMatrix* rm18a = new G4RotationMatrix();
G4RotationMatrix* rm19a = new G4RotationMatrix();
G4RotationMatrix* rm20a = new G4RotationMatrix();
G4RotationMatrix* rm21a = new G4RotationMatrix();
G4RotationMatrix* rm22a = new G4RotationMatrix();
G4RotationMatrix* rm23a = new G4RotationMatrix();
G4RotationMatrix* rm24a = new G4RotationMatrix();
G4RotationMatrix* rm25a = new G4RotationMatrix();
G4RotationMatrix* rm26a = new G4RotationMatrix();
G4RotationMatrix* rm27a = new G4RotationMatrix();
G4RotationMatrix* rm28a = new G4RotationMatrix();
G4RotationMatrix* rm29a = new G4RotationMatrix();
G4RotationMatrix* rm30a = new G4RotationMatrix();
G4RotationMatrix* rm31a = new G4RotationMatrix();
G4RotationMatrix* rm32a = new G4RotationMatrix();
G4RotationMatrix* rm33a = new G4RotationMatrix();
G4RotationMatrix* rm34a = new G4RotationMatrix();
G4RotationMatrix* rm35a = new G4RotationMatrix();
G4RotationMatrix* rm36a = new G4RotationMatrix();
G4RotationMatrix* rm37a = new G4RotationMatrix();
G4RotationMatrix* rm38a = new G4RotationMatrix();
G4RotationMatrix* rm39a = new G4RotationMatrix();
G4RotationMatrix* rm40a = new G4RotationMatrix();
G4RotationMatrix* rm41a = new G4RotationMatrix();
G4RotationMatrix* rm42a = new G4RotationMatrix();
G4RotationMatrix* rm43a = new G4RotationMatrix();
G4RotationMatrix* rm44a = new G4RotationMatrix();
G4RotationMatrix* rm45a = new G4RotationMatrix();
G4RotationMatrix* rm46a = new G4RotationMatrix();
G4RotationMatrix* rm47a = new G4RotationMatrix();
G4RotationMatrix* rm48a = new G4RotationMatrix();
G4RotationMatrix* rm49a = new G4RotationMatrix();
G4RotationMatrix* rm50a = new G4RotationMatrix();
G4RotationMatrix* rm51a = new G4RotationMatrix();
G4RotationMatrix* rm52a = new G4RotationMatrix();
G4RotationMatrix* rm53a = new G4RotationMatrix();
G4RotationMatrix* rm54a = new G4RotationMatrix();
G4RotationMatrix* rm55a = new G4RotationMatrix();
G4RotationMatrix* rm56a = new G4RotationMatrix();
G4RotationMatrix* rm57a = new G4RotationMatrix();
G4RotationMatrix* rm58a = new G4RotationMatrix();
G4RotationMatrix* rm59a = new G4RotationMatrix();
G4RotationMatrix* rm60a = new G4RotationMatrix();

rm1-> rotateX(90*deg);
rm1-> rotateY(0*deg);
LowerShieldingTorus1Physical = new G4PVPlacement(rm1,
    G4ThreeVector(4.6*m,0,-5.*m),
    LowerShieldingTorusLogical,
    "LowerShieldingTorus1_Physical",
    pMotherLogical,
    false,
    0);
// rotation matrix
// translation vector
// logical volume
// name
// mother volume
// unused boolean
// copy number

rm2-> rotateX(90*deg);
rm2-> rotateY(3*deg);
LowerShieldingTorus2Physical = new G4PVPlacement(rm2,
    G4ThreeVector(4.594*m,0.241*m,-5.*m),
    LowerShieldingTorusLogical,
    "LowerShieldingTorus2_Physical",
    pMotherLogical,
    false,
    0);
// rotation matrix
// translation vector
// logical volume
// name
// mother volume
// unused boolean
// copy number

```

```

rm3-> rotateX(90*deg);
rm3-> rotateY(6*deg);
LowerShieldingTorus3Physical = new G4PVPlacement(rm3, // rotation matrix
G4ThreeVector(4.575*m,0.481*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus3_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm4-> rotateX(90*deg);
rm4-> rotateY(9*deg);
LowerShieldingTorus4Physical = new G4PVPlacement(rm4, // rotation matrix
G4ThreeVector(4.543*m,0.720*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus4_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm5-> rotateX(90*deg);
rm5-> rotateY(12*deg);
LowerShieldingTorus5Physical = new G4PVPlacement(rm5, // rotation matrix
G4ThreeVector(4.499*m,0.956*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus5_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm6-> rotateX(90*deg);
rm6-> rotateY(15.*deg);
LowerShieldingTorus6Physical = new G4PVPlacement(rm6, // rotation matrix
G4ThreeVector(4.443*m,1.191*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus6_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm7-> rotateX(90*deg);
rm7-> rotateY(18*deg);
LowerShieldingTorus7Physical = new G4PVPlacement(rm7, // rotation matrix
G4ThreeVector(4.375*m,1.421*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus7_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm8-> rotateX(90*deg);
rm8-> rotateY(21*deg);
LowerShieldingTorus8Physical = new G4PVPlacement(rm8, // rotation matrix
G4ThreeVector(4.294*m,1.648*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus8_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm9-> rotateX(90*deg);
rm9-> rotateY(24*deg);
LowerShieldingTorus9Physical = new G4PVPlacement(rm9, // rotation matrix
G4ThreeVector(4.202*m,1.871*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus9_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm10-> rotateX(90*deg);
rm10-> rotateY(27*deg);
LowerShieldingTorus10Physical = new G4PVPlacement(rm10, // rotation matrix
G4ThreeVector(4.099*m,2.088*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus10_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm11-> rotateX(90*deg);
rm11-> rotateY(30*deg);
LowerShieldingTorus11Physical = new G4PVPlacement(rm11, // rotation matrix
G4ThreeVector(3.984*m,2.3*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus11_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm12-> rotateX(90*deg);
rm12-> rotateY(33*deg);
LowerShieldingTorus12Physical = new G4PVPlacement(rm12, // rotation matrix
G4ThreeVector(3.858*m,2.505*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus12_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm13-> rotateX(90*deg);
rm13-> rotateY(36*deg);
LowerShieldingTorus13Physical = new G4PVPlacement(rm13, // rotation matrix
G4ThreeVector(3.721*m,2.704*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus13_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm14-> rotateX(90*deg);
rm14-> rotateY(39*deg);
LowerShieldingTorus14Physical = new G4PVPlacement(rm14, // rotation matrix
G4ThreeVector(3.575*m,2.895*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus14_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm15-> rotateX(90*deg);
rm15-> rotateY(42*deg);
LowerShieldingTorus15Physical = new G4PVPlacement(rm15, // rotation matrix
G4ThreeVector(3.418*m,3.078*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus15_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm16-> rotateX(90*deg);
rm16-> rotateY(45*deg);
LowerShieldingTorus16Physical = new G4PVPlacement(rm16, // rotation matrix
G4ThreeVector(3.253*m,3.253*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus16_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm17-> rotateX(90*deg);
rm17-> rotateY(48*deg);
LowerShieldingTorus17Physical = new G4PVPlacement(rm17, // rotation matrix
G4ThreeVector(3.078*m,3.418*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus17_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm18-> rotateX(90*deg);
rm18-> rotateY(51*deg);
LowerShieldingTorus18Physical = new G4PVPlacement(rm18, // rotation matrix
G4ThreeVector(2.895*m,3.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus18_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm19-> rotateX(90*deg);
rm19-> rotateY(54*deg);
LowerShieldingTorus19Physical = new G4PVPlacement(rm19, // rotation matrix
G4ThreeVector(2.704*m,3.721*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus19_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm20-> rotateX(90*deg);
rm20-> rotateY(57*deg);
LowerShieldingTorus20Physical = new G4PVPlacement(rm20, // rotation matrix
G4ThreeVector(2.505*m,3.858*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus20_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm21-> rotateX(90*deg);
rm21-> rotateY(60*deg);
LowerShieldingTorus21Physical = new G4PVPlacement(rm21, // rotation matrix
G4ThreeVector(2.3*m,3.984*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus21_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm22-> rotateX(90*deg);
rm22-> rotateY(63*deg);
LowerShieldingTorus22Physical = new G4PVPlacement(rm22, // rotation matrix
G4ThreeVector(2.088*m,4.099*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus22_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm23-> rotateX(90*deg);
rm23-> rotateY(66*deg);
LowerShieldingTorus23Physical = new G4PVPlacement(rm23, // rotation matrix
G4ThreeVector(1.871*m,4.202*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus23_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm24-> rotateX(90*deg);
rm24-> rotateY(69*deg);
LowerShieldingTorus24Physical = new G4PVPlacement(rm24, // rotation matrix
G4ThreeVector(1.648*m,4.294*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus24_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm25-> rotateX(90*deg);
rm25-> rotateY(72*deg);
LowerShieldingTorus25Physical = new G4PVPlacement(rm25, // rotation matrix
G4ThreeVector(1.421*m,4.375*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus25_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm26-> rotateX(90*deg);
rm26-> rotateY(75*deg);
LowerShieldingTorus26Physical = new G4PVPlacement(rm26, // rotation matrix
G4ThreeVector(1.191*m,4.443*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus26_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm27-> rotateX(90*deg);
rm27-> rotateY(78*deg);
LowerShieldingTorus27Physical = new G4PVPlacement(rm27, // rotation matrix
G4ThreeVector(0.956*m,4.499*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus27_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm28-> rotateX(90*deg);
rm28-> rotateY(81*deg);
LowerShieldingTorus28Physical = new G4PVPlacement(rm28, // rotation matrix
G4ThreeVector(0.72*m,4.543*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus28_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm29-> rotateX(90*deg);
rm29-> rotateY(84*deg);
LowerShieldingTorus29Physical = new G4PVPlacement(rm29, // rotation matrix
G4ThreeVector(0.481*m,4.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus29_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm30-> rotateX(90*deg);
rm30-> rotateY(87*deg);
LowerShieldingTorus30Physical = new G4PVPlacement(rm30, // rotation matrix
G4ThreeVector(0.241*m,4.594*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus30_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm31-> rotateX(90*deg);
rm31-> rotateY(90*deg);
LowerShieldingTorus31Physical = new G4PVPlacement(rm31,           // rotation matrix
G4ThreeVector(0,4.6*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus31_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm32-> rotateX(90*deg);
rm32-> rotateY(93*deg);
LowerShieldingTorus32Physical = new G4PVPlacement(rm32,           // rotation matrix
G4ThreeVector(-0.241*m,4.594*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus32_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm33-> rotateX(90*deg);
rm33-> rotateY(96*deg);
LowerShieldingTorus33Physical = new G4PVPlacement(rm33,           // rotation matrix
G4ThreeVector(-0.481*m,4.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus33_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm34-> rotateX(90*deg);
rm34-> rotateY(99*deg);
LowerShieldingTorus34Physical = new G4PVPlacement(rm34,           // rotation matrix
G4ThreeVector(-0.720*m,4.543*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus34_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm35-> rotateX(90*deg);
rm35-> rotateY(102*deg);
LowerShieldingTorus35Physical = new G4PVPlacement(rm35,           // rotation matrix
G4ThreeVector(-0.956*m,4.499*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus35_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm36-> rotateX(90*deg);
rm36-> rotateY(105*deg);
LowerShieldingTorus36Physical = new G4PVPlacement(rm36,           // rotation matrix
G4ThreeVector(-1.191*m,4.443*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus36_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm37-> rotateX(90*deg);
rm37-> rotateY(108*deg);
LowerShieldingTorus37Physical = new G4PVPlacement(rm37,           // rotation matrix
G4ThreeVector(-1.421*m,4.375*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus37_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm38-> rotateX(90*deg);
rm38-> rotateY(111*deg);
LowerShieldingTorus38Physical = new G4PVPlacement(rm38,           // rotation matrix
G4ThreeVector(-1.648*m,4.294*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus38_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm39-> rotateX(90*deg);
rm39-> rotateY(114*deg);
LowerShieldingTorus39Physical = new G4PVPlacement(rm39,           // rotation matrix
G4ThreeVector(-1.871*m,4.202*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus39_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm40-> rotateX(90*deg);
rm40-> rotateY(117*deg);
LowerShieldingTorus40Physical = new G4PVPlacement(rm40,           // rotation matrix
G4ThreeVector(-2.089*m,4.099*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus40_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm41-> rotateX(90*deg);
rm41-> rotateY(120*deg);
LowerShieldingTorus41Physical = new G4PVPlacement(rm41,           // rotation matrix
G4ThreeVector(-2.3*m,3.984*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus41_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm42-> rotateX(90*deg);
rm42-> rotateY(123*deg);
LowerShieldingTorus42Physical = new G4PVPlacement(rm42,           // rotation matrix
G4ThreeVector(-2.505*m,3.858*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus42_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm43-> rotateX(90*deg);
rm43-> rotateY(126*deg);
LowerShieldingTorus43Physical = new G4PVPlacement(rm43,           // rotation matrix
G4ThreeVector(-2.704*m,3.721*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus43_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm44-> rotateX(90*deg);
rm44-> rotateY(129*deg);
LowerShieldingTorus44Physical = new G4PVPlacement(rm44,           // rotation matrix
G4ThreeVector(-2.895*m,3.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus44_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm45-> rotateX(90*deg);
rm45-> rotateY(132*deg);
LowerShieldingTorus45Physical = new G4PVPlacement(rm45, // rotation matrix
G4ThreeVector(-3.078*m,3.418*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus45_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm46-> rotateX(90*deg);
rm46-> rotateY(135*deg);
LowerShieldingTorus46Physical = new G4PVPlacement(rm46, // rotation matrix
G4ThreeVector(-3.253*m,3.253*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus46_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm47-> rotateX(90*deg);
rm47-> rotateY(138*deg);
LowerShieldingTorus47Physical = new G4PVPlacement(rm47, // rotation matrix
G4ThreeVector(-3.418*m,3.078*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus47_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm48-> rotateX(90*deg);
rm48-> rotateY(141*deg);
LowerShieldingTorus48Physical = new G4PVPlacement(rm48, // rotation matrix
G4ThreeVector(-3.575*m,2.895*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus48_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm49-> rotateX(90*deg);
rm49-> rotateY(144*deg);
LowerShieldingTorus49Physical = new G4PVPlacement(rm49, // rotation matrix
G4ThreeVector(-3.721*m,2.704*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus49_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm50-> rotateX(90*deg);
rm50-> rotateY(147*deg);
LowerShieldingTorus50Physical = new G4PVPlacement(rm50, // rotation matrix
G4ThreeVector(-3.858*m,2.505*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus50_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm51-> rotateX(90*deg);
rm51-> rotateY(150*deg);
LowerShieldingTorus51Physical = new G4PVPlacement(rm51, // rotation matrix
G4ThreeVector(-3.984*m,2.3*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus51_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm52-> rotateX(90*deg);
rm52-> rotateY(153*deg);
LowerShieldingTorus52Physical = new G4PVPlacement(rm52,           // rotation matrix
G4ThreeVector(-4.099*m,2.088*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus52_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm53-> rotateX(90*deg);
rm53-> rotateY(156*deg);
LowerShieldingTorus53Physical = new G4PVPlacement(rm53,           // rotation matrix
G4ThreeVector(-4.202*m,1.871*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus53_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm54-> rotateX(90*deg);
rm54-> rotateY(159*deg);
LowerShieldingTorus54Physical = new G4PVPlacement(rm54,           // rotation matrix
G4ThreeVector(-4.294*m,1.648*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus54_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm55-> rotateX(90*deg);
rm55-> rotateY(162*deg);
LowerShieldingTorus55Physical = new G4PVPlacement(rm55,           // rotation matrix
G4ThreeVector(-4.375*m,1.421*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus55_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm56-> rotateX(90*deg);
rm56-> rotateY(165*deg);
LowerShieldingTorus56Physical = new G4PVPlacement(rm56,           // rotation matrix
G4ThreeVector(-4.443*m,1.191*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus56_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm57-> rotateX(90*deg);
rm57-> rotateY(168*deg);
LowerShieldingTorus57Physical = new G4PVPlacement(rm57,           // rotation matrix
G4ThreeVector(-4.499*m,0.956*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus57_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm58-> rotateX(90*deg);
rm58-> rotateY(171*deg);
LowerShieldingTorus58Physical = new G4PVPlacement(rm58,           // rotation matrix
G4ThreeVector(-4.543*m,0.720*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus58_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm59-> rotateX(90*deg);
rm59-> rotateY(174*deg);
LowerShieldingTorus59Physical = new G4PVPlacement(rm59,           // rotation matrix
G4ThreeVector(-4.575*m,0.481*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus59_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm60-> rotateX(90*deg);
rm60-> rotateY(177*deg);
LowerShieldingTorus60Physical = new G4PVPlacement(rm60,           // rotation matrix
G4ThreeVector(-4.594*m,0.241*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus60_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm1a-> rotateX(90*deg);
rm1a-> rotateY(180*deg);
LowerShieldingTorus1aPhysical = new G4PVPlacement(rm1a,           // rotation matrix
G4ThreeVector(-4.6*m,0,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus1a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm2a-> rotateX(90*deg);
rm2a-> rotateY(183*deg);
LowerShieldingTorus2aPhysical = new G4PVPlacement(rm2a,           // rotation matrix
G4ThreeVector(-4.594*m,-0.241*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus2a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm3a-> rotateX(90*deg);
rm3a-> rotateY(186*deg);
LowerShieldingTorus3aPhysical = new G4PVPlacement(rm3a,           // rotation matrix
G4ThreeVector(-4.575*m,-0.481*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus3a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm4a-> rotateX(90*deg);
rm4a-> rotateY(189*deg);
LowerShieldingTorus4aPhysical = new G4PVPlacement(rm4a,           // rotation matrix
G4ThreeVector(-4.543*m,-0.720*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus4a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm5a-> rotateX(90*deg);
rm5a-> rotateY(192*deg);
LowerShieldingTorus5aPhysical = new G4PVPlacement(rm5a,           // rotation matrix
G4ThreeVector(-4.499*m,-0.956*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus5a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm6a-> rotateX(90*deg);
rm6a-> rotateY(195*deg);
LowerShieldingTorus6aPhysical = new G4PVPlacement(rm6a,           // rotation matrix
G4ThreeVector(-4.443*m,-1.191*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus6a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm7a-> rotateX(90*deg);
rm7a-> rotateY(198*deg);
LowerShieldingTorus7aPhysical = new G4PVPlacement(rm7a,           // rotation matrix
G4ThreeVector(-4.375*m,-1.421*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus7a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm8a-> rotateX(90*deg);
rm8a-> rotateY(201*deg);
LowerShieldingTorus8aPhysical = new G4PVPlacement(rm8a,           // rotation matrix
G4ThreeVector(-4.294*m,-1.648*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus8a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm9a-> rotateX(90*deg);
rm9a-> rotateY(204*deg);
LowerShieldingTorus9aPhysical = new G4PVPlacement(rm9a,           // rotation matrix
G4ThreeVector(-4.202*m,-1.871*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus9a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm10a-> rotateX(90*deg);
rm10a-> rotateY(207*deg);
LowerShieldingTorus10aPhysical = new G4PVPlacement(rm10a,         // rotation matrix
G4ThreeVector(-4.099*m,-2.088*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus10a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm11a-> rotateX(90*deg);
rm11a-> rotateY(210*deg);
LowerShieldingTorus11aPhysical = new G4PVPlacement(rm11a,         // rotation matrix
G4ThreeVector(-3.984*m,-2.3*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus11a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm12a-> rotateX(90*deg);
rm12a-> rotateY(213*deg);
LowerShieldingTorus12aPhysical = new G4PVPlacement(rm12a,         // rotation matrix
G4ThreeVector(-3.858*m,-2.505*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus12a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm13a-> rotateX(90*deg);
rm13a-> rotateY(216*deg);
LowerShieldingTorus13aPhysical = new G4PVPlacement(rm13a,           // rotation matrix
G4ThreeVector(-3.721*m,-2.704*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus13a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm14a-> rotateX(90*deg);
rm14a-> rotateY(219*deg);
LowerShieldingTorus14aPhysical = new G4PVPlacement(rm14a,           // rotation matrix
G4ThreeVector(-3.575*m,-2.895*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus14a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm15a-> rotateX(90*deg);
rm15a-> rotateY(222*deg);
LowerShieldingTorus15aPhysical = new G4PVPlacement(rm15a,           // rotation matrix
G4ThreeVector(-3.418*m,-3.078*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus15a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm16a-> rotateX(90*deg);
rm16a-> rotateY(225*deg);
LowerShieldingTorus16aPhysical = new G4PVPlacement(rm16a,           // rotation matrix
G4ThreeVector(-3.253*m,-3.253*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus16a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm17a-> rotateX(90*deg);
rm17a-> rotateY(228*deg);
LowerShieldingTorus17aPhysical = new G4PVPlacement(rm17a,           // rotation matrix
G4ThreeVector(-3.078*m,-3.418*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus17a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm18a-> rotateX(90*deg);
rm18a-> rotateY(231*deg);
LowerShieldingTorus18aPhysical = new G4PVPlacement(rm18a,           // rotation matrix
G4ThreeVector(-2.895*m,-3.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus18a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm19a-> rotateX(90*deg);
rm19a-> rotateY(234*deg);
LowerShieldingTorus19aPhysical = new G4PVPlacement(rm19a,           // rotation matrix
G4ThreeVector(-2.704*m,-3.721*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus19a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm20a-> rotateX(90*deg);
rm20a-> rotateY(237*deg);
LowerShieldingTorus20aPhysical = new G4PVPlacement(rm20a,           // rotation matrix
G4ThreeVector(-2.505*m,-3.858*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus20a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm21a-> rotateX(90*deg);
rm21a-> rotateY(240*deg);
LowerShieldingTorus21aPhysical = new G4PVPlacement(rm21a,           // rotation matrix
G4ThreeVector(-2.3*m,-3.984*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus21a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm22a-> rotateX(90*deg);
rm22a-> rotateY(243*deg);
LowerShieldingTorus22aPhysical = new G4PVPlacement(rm22a,           // rotation matrix
G4ThreeVector(-2.088*m,-4.099*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus22a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm23a-> rotateX(90*deg);
rm23a-> rotateY(246*deg);
LowerShieldingTorus23aPhysical = new G4PVPlacement(rm23a,           // rotation matrix
G4ThreeVector(-1.871*m,-4.202*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus23a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm24a-> rotateX(90*deg);
rm24a-> rotateY(249*deg);
LowerShieldingTorus24aPhysical = new G4PVPlacement(rm24a,           // rotation matrix
G4ThreeVector(-1.648*m,-4.294*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus24a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm25a-> rotateX(90*deg);
rm25a-> rotateY(252*deg);
LowerShieldingTorus25aPhysical = new G4PVPlacement(rm25a,           // rotation matrix
G4ThreeVector(-1.421*m,-4.375*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus25a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm26a-> rotateX(90*deg);
rm26a-> rotateY(255*deg);
LowerShieldingTorus26aPhysical = new G4PVPlacement(rm26a,           // rotation matrix
G4ThreeVector(-1.191*m,-4.443*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus26a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm27a-> rotateX(90*deg);
rm27a-> rotateY(258*deg);
LowerShieldingTorus27aPhysical = new G4PVPlacement(rm27a,           // rotation matrix
G4ThreeVector(-0.956*m,-4.499*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus27a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm28a-> rotateX(90*deg);
rm28a-> rotateY(261*deg);
LowerShieldingTorus28aPhysical = new G4PVPlacement(rm28a,           // rotation matrix
G4ThreeVector(-0.72*m,-4.543*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus28a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm29a-> rotateX(90*deg);
rm29a-> rotateY(264*deg);
LowerShieldingTorus29aPhysical = new G4PVPlacement(rm29a,           // rotation matrix
G4ThreeVector(-0.481*m,-4.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus29a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm30a-> rotateX(90*deg);
rm30a-> rotateY(267*deg);
LowerShieldingTorus30aPhysical = new G4PVPlacement(rm30a,           // rotation matrix
G4ThreeVector(-0.241*m,-4.594*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus30a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm31a-> rotateX(90*deg);
rm31a-> rotateY(270*deg);
LowerShieldingTorus31aPhysical = new G4PVPlacement(rm31a,           // rotation matrix
G4ThreeVector(0,-4.6*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus31a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm32a-> rotateX(90*deg);
rm32a-> rotateY(273*deg);
LowerShieldingTorus32aPhysical = new G4PVPlacement(rm32a,           // rotation matrix
G4ThreeVector(0.241*m,-4.594*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus32a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm33a-> rotateX(90*deg);
rm33a-> rotateY(276*deg);
LowerShieldingTorus33aPhysical = new G4PVPlacement(rm33a,           // rotation matrix
G4ThreeVector(0.481*m,-4.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus33a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm34a-> rotateX(90*deg);
rm34a-> rotateY(279*deg);
LowerShieldingTorus34aPhysical = new G4PVPlacement(rm34a, // rotation matrix
G4ThreeVector(0.720*m,-4.543*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus34a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm35a-> rotateX(90*deg);
rm35a-> rotateY(282*deg);
LowerShieldingTorus35aPhysical = new G4PVPlacement(rm35a, // rotation matrix
G4ThreeVector(0.956*m,-4.499*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus35a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm36a-> rotateX(90*deg);
rm36a-> rotateY(285*deg);
LowerShieldingTorus36aPhysical = new G4PVPlacement(rm36a, // rotation matrix
G4ThreeVector(1.191*m,-4.443*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus36a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm37a-> rotateX(90*deg);
rm37a-> rotateY(288*deg);
LowerShieldingTorus37aPhysical = new G4PVPlacement(rm37a, // rotation matrix
G4ThreeVector(1.421*m,-4.375*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus37a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm38a-> rotateX(90*deg);
rm38a-> rotateY(291*deg);
LowerShieldingTorus38aPhysical = new G4PVPlacement(rm38a, // rotation matrix
G4ThreeVector(1.648*m,-4.294*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus38a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm39a-> rotateX(90*deg);
rm39a-> rotateY(294*deg);
LowerShieldingTorus39aPhysical = new G4PVPlacement(rm39a, // rotation matrix
G4ThreeVector(1.871*m,-4.202*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus39a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm40a-> rotateX(90*deg);
rm40a-> rotateY(297*deg);
LowerShieldingTorus40aPhysical = new G4PVPlacement(rm40a, // rotation matrix
G4ThreeVector(2.089*m,-4.099*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus40a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm41a-> rotateX(90*deg);
rm41a-> rotateY(300*deg);
LowerShieldingTorus41aPhysical = new G4PVPlacement(rm41a, // rotation matrix
G4ThreeVector(2.3*m,-3.984*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus41a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm42a-> rotateX(90*deg);
rm42a-> rotateY(303*deg);
LowerShieldingTorus42aPhysical = new G4PVPlacement(rm42a, // rotation matrix
G4ThreeVector(2.505*m,-3.858*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus42a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm43a-> rotateX(90*deg);
rm43a-> rotateY(306*deg);
LowerShieldingTorus43aPhysical = new G4PVPlacement(rm43a, // rotation matrix
G4ThreeVector(2.704*m,-3.721*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus43a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm44a-> rotateX(90*deg);
rm44a-> rotateY(309*deg);
LowerShieldingTorus44aPhysical = new G4PVPlacement(rm44a, // rotation matrix
G4ThreeVector(2.895*m,-3.575*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus44a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm45a-> rotateX(90*deg);
rm45a-> rotateY(312*deg);
LowerShieldingTorus45aPhysical = new G4PVPlacement(rm45a, // rotation matrix
G4ThreeVector(3.078*m,-3.418*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus45a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm46a-> rotateX(90*deg);
rm46a-> rotateY(315*deg);
LowerShieldingTorus46aPhysical = new G4PVPlacement(rm46a, // rotation matrix
G4ThreeVector(3.253*m,-3.253*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus46a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm47a-> rotateX(90*deg);
rm47a-> rotateY(318*deg);
LowerShieldingTorus47aPhysical = new G4PVPlacement(rm47a, // rotation matrix
G4ThreeVector(3.418*m,-3.078*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus47a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm48a-> rotateX(90*deg);
rm48a-> rotateY(321*deg);
LowerShieldingTorus48aPhysical = new G4PVPlacement(rm48a, // rotation matrix
G4ThreeVector(3.575*m,-2.895*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus48a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm49a-> rotateX(90*deg);
rm49a-> rotateY(324*deg);
LowerShieldingTorus49aPhysical = new G4PVPlacement(rm49a, // rotation matrix
G4ThreeVector(3.721*m,-2.704*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus49a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm50a-> rotateX(90*deg);
rm50a-> rotateY(327*deg);
LowerShieldingTorus50aPhysical = new G4PVPlacement(rm50a, // rotation matrix
G4ThreeVector(3.858*m,-2.505*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus50a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm51a-> rotateX(90*deg);
rm51a-> rotateY(330*deg);
LowerShieldingTorus51aPhysical = new G4PVPlacement(rm51a, // rotation matrix
G4ThreeVector(3.984*m,-2.3*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus51a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm52a-> rotateX(90*deg);
rm52a-> rotateY(333*deg);
LowerShieldingTorus52aPhysical = new G4PVPlacement(rm52a, // rotation matrix
G4ThreeVector(4.099*m,-2.088*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus52a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm53a-> rotateX(90*deg);
rm53a-> rotateY(336*deg);
LowerShieldingTorus53aPhysical = new G4PVPlacement(rm53a, // rotation matrix
G4ThreeVector(4.202*m,-1.871*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus53a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm54a-> rotateX(90*deg);
rm54a-> rotateY(339*deg);
LowerShieldingTorus54aPhysical = new G4PVPlacement(rm54a, // rotation matrix
G4ThreeVector(4.294*m,-1.648*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus54a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

```

```

rm55a-> rotateX(90*deg);
rm55a-> rotateY(342*deg);
LowerShieldingTorus55aPhysical = new G4PVPlacement(rm55a, // rotation matrix
G4ThreeVector(4.375*m,-1.421*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus55a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm56a-> rotateX(90*deg);
rm56a-> rotateY(345*deg);
LowerShieldingTorus56aPhysical = new G4PVPlacement(rm56a, // rotation matrix
G4ThreeVector(4.443*m,-1.191*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus56a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm57a-> rotateX(90*deg);
rm57a-> rotateY(348*deg);
LowerShieldingTorus57aPhysical = new G4PVPlacement(rm57a, // rotation matrix
G4ThreeVector(4.499*m,-0.956*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus57a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm58a-> rotateX(90*deg);
rm58a-> rotateY(351*deg);
LowerShieldingTorus58aPhysical = new G4PVPlacement(rm58a, // rotation matrix
G4ThreeVector(4.543*m,-0.720*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus58a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm59a-> rotateX(90*deg);
rm59a-> rotateY(354*deg);
LowerShieldingTorus59aPhysical = new G4PVPlacement(rm59a, // rotation matrix
G4ThreeVector(4.575*m,-0.481*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus59a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

rm60a-> rotateX(90*deg);
rm60a-> rotateY(357*deg);
LowerShieldingTorus60aPhysical = new G4PVPlacement(rm60a, // rotation matrix
G4ThreeVector(4.594*m,-0.241*m,-5.*m), // translation vector
LowerShieldingTorusLogical, // logical volume
"LowerShieldingTorus60a_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

return LowerShieldingTorus1Physical;
}

// _____//
// construct shielding interior volume

G4VPhysicalVolume * DetectorConstruction::ConstructShieldingInterior(G4LogicalVolume
*pMotherLogical)

```

```

{

    // magnetic field
    G4MagneticField *CMField = new ConfinedMagneticField(Bo);
    G4FieldManager *CMFieldManager = new G4FieldManager;
    CMFieldManager->SetDetectorField(CMField);
    CMFieldManager->CreateChordFinder(CMField);

    // get materials
    G4Material *g4galactic = G4Material::GetMaterial("G4_Galactic");

    G4VSolid *ShieldingInterior_Solid = new G4Tubs("ShieldingInterior_Solid", // name
                                                    2.8501*m, // inner radius
                                                    6.3499*m, // outer radius
                                                    4.9999*m, // half z
                                                    0, // start phi
                                                    360.*deg); // ending phi

    ShieldingInteriorLogical = new G4LogicalVolume(ShieldingInterior_Solid, // solid volume
                                                    g4galactic, // material
                                                    "ShieldingInterior_Logical");// name

    ShieldingInteriorLogical->SetFieldManager(CMFieldManager, false);
    // to apply magnetic field manager

    // register magnetic field and its manager for delete
    G4AutoDelete::Register(CMField);
    G4AutoDelete::Register(CMFieldManager);

    // set visibility to invisible
    ShieldingInteriorLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    ShieldingInteriorLogical->SetUserLimits(fStepLimit);

    ShieldingInteriorPhysical = new G4PVPlacement(0, // rotation matrix
                                                    G4ThreeVector(), // translation vector
                                                    ShieldingInteriorLogical, // logical volume
                                                    "ShieldingInterior_Physical", // name
                                                    pMotherLogical, // mother volume
                                                    false, // unused boolean
                                                    0); // copy number

    return ShieldingInteriorPhysical;
}

// _____//

// construct former cylinder

G4VPhysicalVolume * DetectorConstruction::ConstructFormerCylinder(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4Ti = G4Material::GetMaterial("G4_Ti");

    // set former cylinder volume
    G4VSolid *FormerCylinder_Solid = new G4Tubs("FormerCylinder_Solid", // name
                                                    4.5*m, // inner radius
                                                    4.7*m, // outer radius
                                                    4.99*m, // half z
                                                    0, // start phi
                                                    360.*deg); // ending phi

    FormerCylinderLogical = new G4LogicalVolume(FormerCylinder_Solid, // solid volume

```

```

        g4Ti, // material
        "FormerCylinder_Logical"); // name

// set titanium color
FormerCylinderLogical->SetVisAttributes(G4Colour(0.7137, 0.6863, 0.6627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
FormerCylinderLogical->SetUserLimits(fStepLimit);

FormerCylinderPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(), // translation vector
    FormerCylinderLogical, // logical volume
    "FormerCylinder_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

return FormerCylinderPhysical;
}

// _____//

// construct water phantom

G4VPhysicalVolume * DetectorConstruction::ConstructWaterPhantom(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4water = G4Material::GetMaterial("G4_WATER");

    // set water phantom volume (dimensions set for 70 kg mass)
    G4VSolid *WaterPhantom_Solid = new G4Box("WaterPhantom_Solid", // name
        10*cm, // half x
        10*cm, // half y
        87.5*cm); // half z

    WaterPhantomLogical = new G4LogicalVolume(WaterPhantom_Solid, // solid volume
        g4water, // material
        "WaterPhantom_Logical"); // name

    // set water color
    WaterPhantomLogical->SetVisAttributes(G4Colour(0.529, 0.808, 0.980));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    WaterPhantomLogical->SetUserLimits(fStepLimit);

    WaterPhantomPhysical= new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(), // translation vector wrt spacecraft air
        WaterPhantomLogical, // logical volume
        "WaterPhantom_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    return WaterPhantomPhysical;
}

// _____//

// construct mother volume for MIRD phantom

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDMotherVolume(G4LogicalVolume
*pMotherLogical)
{

```

```

// get materials
G4Material *g4air = G4Material::GetMaterial("G4_AIR");

// set mother volume for MIRD phantom
G4VSolid *MIRDMother_Solid = new G4Box("MIRDMother_Solid",    // name
                                     22.*cm,                  // half x
                                     22.*cm,                  // half y
                                     100.*cm);                 // half z

MIRDMotherLogical = new G4LogicalVolume(MIRDMother_Solid,      // solid volume
                                       g4air,                   // material
                                       "MIRDMother_Logical");    // name

// set visibility to invisible
MIRDMotherLogical->SetVisAttributes(G4VisAttributes::Invisible);

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
MIRDMotherLogical->SetUserLimits(fStepLimit);

MIRDMotherPhysical= new G4PVPlacement(0,                      // rotation matrix
                                     G4ThreeVector(),          // translation vector wrt spacecraft air
                                     MIRDMotherLogical,         // logical volume
                                     "MIRDMother_Physical",     // name
                                     pMotherLogical,            // mother volume
                                     false,                     // unused boolean
                                     0);                        // copy number

return MIRDMotherPhysical;
}

// _____//

// construct MIRD head (outer volume of head)

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDHead(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set head volume
    G4Ellipsoid *Head1 = new G4Ellipsoid("Head1",             // name
                                         7.*cm,                // ax
                                         10.*cm,               // by
                                         8.5*cm,              // cz
                                         0.*cm,               // zcut1
                                         8.5*cm);              // zcut2

    G4EllipticalTube *Head2 = new G4EllipticalTube("Head2",   // name
                                                    7.*cm,       // dx
                                                    10.*cm,       // dy
                                                    7.75*cm);    // dz

    G4UnionSolid *Head_Solid = new G4UnionSolid("Head_Solid", // name
                                                Head2,          // 1st volume to union
                                                Head1,          // 2nd volume to union
                                                0,              // rotation matrix
                                                G4ThreeVector(0,0,7.75*cm)); // relative position

    HeadLogical = new G4LogicalVolume(Head_Solid,             // solid volume
                                     soft_tissue,             // material
                                     "Head_Logical");           // name

    // set visibility to invisible
    HeadLogical->SetVisAttributes(G4VisAttributes::Invisible);
    // set skin color
    // HeadLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

```

```

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
HeadLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateX(180.*deg);
rm->rotateY(180.*deg);

HeadPhysical = new G4PVPlacement(rm, // rotation matrix
                                G4ThreeVector(0, 0, 77.75*cm), // translation vector wrt MIRD mother
                                HeadLogical, // logical volume
                                "Head_Physical", // name
                                pMotherLogical, // mother volume
                                false, // unused boolean
                                0); // copy number

// calculate mass
G4double HeadVol = HeadLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Head = " << HeadVol/cm3 << " cm^3" << G4endl;
G4String HeadMat = HeadLogical->GetMaterial()->GetName();
G4cout << "Material of Head = " << HeadMat << G4endl;
G4double HeadDensity = HeadLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << HeadDensity*cm3/g << " g/cm^3" << G4endl;
G4double HeadMass = (HeadVol)*HeadDensity;
G4cout << "Mass of Head = " << HeadMass/gram << " g" << G4endl;

return HeadPhysical;
}

// _____//

// construct MIRD cranium
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDCranium(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set cranium volume
    G4Ellipsoid *Cranium_Out = new G4Ellipsoid("Cranium_Out", // name
                                                6.8*cm, // ax
                                                9.8*cm, // by
                                                8.3*cm); // cz

    G4Ellipsoid *Cranium_In = new G4Ellipsoid("Cranium_In", // name
                                                6.*cm, // ax
                                                9.*cm, // by
                                                6.5*cm); // cz

    G4SubtractionSolid *Cranium_Solid = new G4SubtractionSolid("Cranium_Solid", // name
                                                                Cranium_Out, // larger volume to subtract from
                                                                Cranium_In, // smaller volume to subtract
                                                                0, // rotation matrix
                                                                G4ThreeVector(0, 0, 1.*cm)); // relative position

    CraniumLogical = new G4LogicalVolume(Cranium_Solid, // solid volume
                                         g4bone, // material
                                         "Cranium_Logical"); // name

    // set bone color
    CraniumLogical->SetVisAttributes(G4Colour(1., 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);

```

```

CraniumLogical->SetUserLimits(fStepLimit);

CraniumPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0, 0, 7.75*cm), // translation vector wrt head
    CraniumLogical, // logical volume
    "Cranium_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

// calculate mass
G4double CraniumVol = CraniumLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Cranium = " << CraniumVol/cm3 << " cm^3" << G4endl;
G4String CraniumMat = CraniumLogical->GetMaterial()->GetName();
G4cout << "Material of Cranium = " << CraniumMat << G4endl;
G4double CraniumDensity = CraniumLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << CraniumDensity*cm3/g << " g/cm^3" << G4endl;
G4double CraniumMass = (CraniumVol)*CraniumDensity;
G4cout << "Mass of Cranium = " << CraniumMass/gram << " g" << G4endl;

return CraniumPhysical;
}

// _____//

// construct cranium active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructCraniumMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set cranium active marrow volume
    G4Ellipsoid *CraniumMarrow_Out = new G4Ellipsoid("CraniumMarrow_Out", // name
        6.48*cm, // ax
        9.48*cm, // by
        7.48*cm); // cz

    G4Ellipsoid *CraniumMarrow_In = new G4Ellipsoid("CraniumMarrow_In", // name
        6.32*cm, // ax
        9.32*cm, // by
        7.32*cm); // cz

    G4SubtractionSolid *CraniumMarrow_Solid = new G4SubtractionSolid("CraniumMarrow_Solid",
        CraniumMarrow_Out, // larger volume to subtract from
        CraniumMarrow_In, // smaller volume to subtract
        0, // rotation matrix
        G4ThreeVector()); // relative position

    CraniumMarrowLogical = new G4LogicalVolume(CraniumMarrow_Solid, // solid volume
        red_marrow, // material
        "CraniumMarrow_Logical"); // name

    // set red marrow color
    CraniumMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    CraniumMarrowLogical->SetUserLimits(fStepLimit);

    CraniumMarrowPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0,0,0.5*cm), // translation vector wrt cranium
        CraniumMarrowLogical, // logical volume
        "CraniumMarrow_Physical", // name
        pMotherLogical, // mother volume

```

```

        false,                                // unused boolean
        0);                                    // copy number

// calculate mass
G4double CraniumMarrowVol = CraniumMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Cranium Marrow = " << CraniumMarrowVol/cm3 << " cm^3" << G4endl;
G4String CraniumMarrowMat = CraniumMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Cranium Marrow = " << CraniumMarrowMat << G4endl;
G4double CraniumMarrowDensity = CraniumMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << CraniumMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double CraniumMarrowMass = (CraniumMarrowVol)*CraniumMarrowDensity;
G4cout << "Mass of Cranium Marrow = " << CraniumMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = CraniumMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 1;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return CraniumMarrowPhysical;
}

// _____//

// construct left eye lens

G4VPhysicalVolume * DetectorConstruction::ConstructLeftEyeLens(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4eyelens = G4Material::GetMaterial("G4_EYE_LENS_ICRP");

    // set left eye lens volume
    G4VSolid *EyeLens_Solid = new G4Tubs("EyeLens_Solid",           // name
        0,                                                           // inner radius
        4.6*mm,                                                       // outer radius
        1.55*mm,                                                      // z half length
        0,                                                            // start phi
        360.*deg);                                                    // delta phi

    LeftEyeLensLogical = new G4LogicalVolume(EyeLens_Solid,         // solid volume
        g4eyelens,                                                   // material
        "LeftEyeLens_Logical");                                     // name

    // set eye lens color
    LeftEyeLensLogical->SetVisAttributes(G4Colour(0.529, 0.808, 0.980));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftEyeLensLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(90.*deg);
    LeftEyeLensPhysical = new G4PVPlacement(rm,                     // rotation matrix
        G4ThreeVector(4.*cm,9.*cm,88.*cm), // translation vector wrt MIRD Mother
        LeftEyeLensLogical,                                         // logical volume
        "LeftEyeLens_Physical",                                     // name
        pMotherLogical,                                             // mother volume
        false,                                                       // unused boolean

```

```

        0); // copy number

// calculate mass
G4double EyeLensVol = LeftEyeLensLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Eye Lens = " << EyeLensVol/cm3 << "cm^3" << G4endl;
G4String EyeLensMat = LeftEyeLensLogical->GetMaterial()->GetName();
G4cout << "Material of Left Eye Lens = " << EyeLensMat << G4endl;
G4double EyeLensDensity = LeftEyeLensLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << EyeLensDensity*cm3/g << " g/cm^3" << G4endl;
G4double EyeLensMass = (EyeLensVol)*EyeLensDensity;
G4cout << "Mass of Left Eye Lens = " << EyeLensMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftEyeLensLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 2;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftEyeLensPhysical;
}

// _____//

// construct right eye lens

G4VPhysicalVolume * DetectorConstruction::ConstructRightEyeLens(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4eyelens = G4Material::GetMaterial("G4_EYE_LENS_ICRP");

    // set left eye lens volume
    G4VSolid *EyeLens_Solid = new G4Tubs("EyeLens_Solid", // name
        0, // inner radius
        4.6*mm, // outer radius
        1.55*mm, // z half length
        0, // start phi
        360.*deg); // delta phi

    RightEyeLensLogical = new G4LogicalVolume(EyeLens_Solid, // solid volume
        g4eyelens, // material
        "RightEyeLens_Logical"); // name

    // set eye lens color
    RightEyeLensLogical->SetVisAttributes(G4Colour(0.529, 0.808, 0.980));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightEyeLensLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(90.*deg);

    RightEyeLensPhysical = new G4PVPlacement(rm, // rotation matrix
        G4ThreeVector(-4.*cm,9.*cm,88.*cm), // translation vector wrt MIRD Mother
        RightEyeLensLogical, // logical volume
        "RightEyeLens_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean

```

```

        0); // copy number

// calculate mass
G4double EyeLensVol = RightEyeLensLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Eye Lens = " << EyeLensVol/cm3 << "cm^3" << G4endl;
G4String EyeLensMat = RightEyeLensLogical->GetMaterial()->GetName();
G4cout << "Material of Right Eye Lens = " << EyeLensMat << G4endl;
G4double EyeLensDensity = RightEyeLensLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << EyeLensDensity*cm3/g << " g/cm^3" << G4endl;
G4double EyeLensMass = (EyeLensVol)*EyeLensDensity;
G4cout << "Mass of Right Eye Lens = " << EyeLensMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = RightEyeLensLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 3;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightEyeLensPhysical;
}

// _____//

// construct MIRD brain
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDBrain(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4brain = G4Material::GetMaterial("G4_BRAIN_ICRP");

    // set brain volume
    G4VSolid *Brain_Solid = new G4Ellipsoid("Brain_Solid", // name
                                           6.*cm,           // ax
                                           9.*cm,           // by
                                           6.5*cm);         // cz

    BrainLogical = new G4LogicalVolume(Brain_Solid, // solid volume
                                       g4brain,      // material
                                       "Brain_Logical"); // name

    // set brain color
    //BrainLogical->SetVisAttributes(G4Colour(0.41, 0.41, 0.41));
    BrainLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    BrainLogical->SetUserLimits(fStepLimit);

    BrainPhysical = new G4PVPlacement(0, // rotation matrix
                                       G4ThreeVector(0,0,8.75*cm), // translation vector wrt head
                                       BrainLogical, // logical volume
                                       "Brain_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

    // calculate mass
    G4double BrainVol = BrainLogical->GetSolid()->GetCubicVolume();

```

```

G4cout << "Volume of Brain = " << BrainVol/cm3 << " cm^3" << G4endl;
G4String BrainMat = BrainLogical->GetMaterial()->GetName();
G4cout << "Material of Brain = " << BrainMat << G4endl;
G4double BrainDensity = BrainLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << BrainDensity*cm3/g << " g/cm^3" << G4endl;
G4double BrainMass = (BrainVol)*BrainDensity;
G4cout << "Mass of Brain = " << BrainMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = BrainLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 4;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return BrainPhysical;
}

// _____//

// construct MIRD upper spine

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDUpperSpine(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set upper spine volume
    G4EllipticalTube *UpperSpine = new G4EllipticalTube("UpperSpine", // name
        2.*cm, // dx
        2.5*cm, // dy
        4.25*cm); // dz

    G4Box *SubtrUpperSpine = new G4Box("SubtrUpperSpine", // name
        10.*cm, // dx
        5.*cm, // dy
        2.5*cm); // dz

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(-25.*deg);

    G4SubtractionSolid *UpperSpine_Solid = new G4SubtractionSolid("UpperSpine_Solid", // name
        UpperSpine, // larger volume to subtract from
        SubtrUpperSpine, // smaller volume to subtract
        rm, // rotation matrix
        G4ThreeVector(0,-2.5*cm,5.5*cm)); // relative position

    UpperSpineLogical = new G4LogicalVolume(UpperSpine_Solid, // solid volume
        g4bone, // material
        "UpperSpine_Logical"); // name

    // set bone color
    UpperSpineLogical->SetVisAttributes(G4Colour(1., 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    UpperSpineLogical->SetUserLimits(fStepLimit);

    UpperSpinePhysical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(0, 5.5*cm, -3.5*cm), // translation vector wrt head
        UpperSpineLogical,                // logical volume
        "UpperSpine_Physical",            // name
        pMotherLogical,                   // mother volume
        false,                            // unused boolean
        0);                               // copy number

// calculate mass
G4double UpperSpineVol = UpperSpineLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Upper Spine = " << UpperSpineVol/cm3 << " cm^3" << G4endl;
G4String UpperSpineMat = UpperSpineLogical->GetMaterial()->GetName();
G4cout << "Material of Upper Spine = " << UpperSpineMat << G4endl;
G4double UpperSpineDensity = UpperSpineLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << UpperSpineDensity*cm3/g << " g/cm^3" << G4endl;
G4double UpperSpineMass = (UpperSpineVol)*UpperSpineDensity;
G4cout << "Mass of Upper Spine = " << UpperSpineMass/gram << " g" << G4endl;

return UpperSpinePhysical;
}

// _____//

// construct upper spine active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructUpperSpineMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set upper spine active marrow volume
    G4VSolid *UpperSpineMarrow_Solid = new G4EllipticalTube("UpperSpineMarrow_Solid", // name
                                                             1.53*cm, // dx
                                                             1.91*cm, // dy
                                                             3.*cm); // dz

    UpperSpineMarrowLogical = new G4LogicalVolume(UpperSpineMarrow_Solid, // solid volume
                                                  red_marrow, // material
                                                  "UpperSpineMarrow_Logical"); // name

    // set red marrow color
    UpperSpineMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    UpperSpineMarrowLogical->SetUserLimits(fStepLimit);

    UpperSpineMarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                                  G4ThreeVector(), // translation vector wrt upper spine
                                                  UpperSpineMarrowLogical, // logical volume
                                                  "UpperSpineMarrow_Physical", // name
                                                  pMotherLogical, // mother volume
                                                  false, // unused boolean
                                                  0); // copy number

    // calculate mass
    G4double UpperSpineMarrowVol = UpperSpineMarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Upper Spine Marrow = " << UpperSpineMarrowVol/cm3 << " cm^3" << G4endl;
    G4String UpperSpineMarrowMat = UpperSpineMarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Upper Spine Marrow = " << UpperSpineMarrowMat << G4endl;
    G4double UpperSpineMarrowDensity = UpperSpineMarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << UpperSpineMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double UpperSpineMarrowMass = (UpperSpineMarrowVol)*UpperSpineMarrowDensity;
    G4cout << "Mass of Upper Spine Marrow = " << UpperSpineMarrowMass/gram << " g" << G4endl;

    // store organ information in maps

```

```

G4LogicalVolume* currentLogical = UpperSpineMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 5;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return UpperSpineMarrowPhysical;
}

// _____//

// construct MIRD thyroid

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDThyroid(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set thyroid volume
    G4Tubs *Thyroid_Out = new G4Tubs("Thyroid_Out",    // name
                                     0,                // inner radius
                                     1.85*cm,          // outer radius
                                     2.1*cm,           // dz
                                     0,                // start phi
                                     180.*deg);        // delta phi

    G4Tubs *Thyroid_In = new G4Tubs("Thyroid_In",     // name
                                     0,                // inner radius
                                     0.83*cm,          // outer radius
                                     2.25*cm,          // dz
                                     0,                // start phi
                                     360.*deg);        // delta phi

    G4Box *SubtrThyroid = new G4Box("SubtrThyroid",   // name
                                     1.86*cm,          // dx
                                     1.86*cm,          // dy
                                     10.*cm);          // dz

    G4SubtractionSolid *Thyroid1 = new G4SubtractionSolid("Thyroid1",// name
                                                          Thyroid_Out, // larger volume to subtract from
                                                          Thyroid_In); // smaller volume to subtract

    G4RotationMatrix *rm1 = new G4RotationMatrix();
    rm1->rotateX(-50.*deg);

    G4SubtractionSolid *Thyroid2 = new G4SubtractionSolid("Thyroid2",// name
                                                          Thyroid1,    // larger volume to subtract from
                                                          SubtrThyroid, // smaller volume to subtract
                                                          rm1,        // rotation matrix
                                                          G4ThreeVector(0, 0, 4.2*cm)); // relative position

    // set rotation matrix
    G4RotationMatrix *rm2 = new G4RotationMatrix();
    rm2->rotateX(50.*deg);

    G4SubtractionSolid *Thyroid_Solid = new G4SubtractionSolid("Thyroid_Solid",// name
                                                                Thyroid2,    // larger volume to subtract from
                                                                SubtrThyroid, // smaller volume to subtract
                                                                rm2,        // rotation matrix
                                                                G4ThreeVector(0, 0, -5.4*cm)); // relative position

```

```

ThyroidLogical = new G4LogicalVolume(Thyroid_Solid,           // solid volume
                                     soft_tissue,             // material
                                     "Thyroid_Logical");       // name

// set thyroid color
ThyroidLogical->SetVisAttributes(G4Colour(0.7137, 0.8941, 1.));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
ThyroidLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm3 = new G4RotationMatrix();
rm3->rotateZ(180.*deg);
ThyroidPhysical = new G4PVPlacement(rm3,                    // rotation matrix
                                    G4ThreeVector(0., -3.91*cm, -5.65*cm), // translation vector wrt head
                                    ThyroidLogical,          // logical volume
                                    "Thyroid_Physical",       // name
                                    pMotherLogical,           // mother volume
                                    false,                    // unused boolean
                                    0);                        // copy number

// calculate mass
G4double ThyroidVol = ThyroidLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Thyroid = " << ThyroidVol/cm3 << " cm^3" << G4endl;
G4String ThyroidMat = ThyroidLogical->GetMaterial()->GetName();
G4cout << "Material of Thyroid = " << ThyroidMat << G4endl;
G4double ThyroidDensity = ThyroidLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << ThyroidDensity*cm3/g << " g/cm^3" << G4endl;
G4double ThyroidMass = (ThyroidVol)*ThyroidDensity;
G4cout << "Mass of Thyroid = " << ThyroidMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = ThyroidLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 6;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return ThyroidPhysical;
}

// _____//

// construct MIRD trunk

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDTrunk(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set trunk volume
    G4VSolid *Trunk_Solid = new G4EllipticalTube("Trunk_Solid", // name
                                                  20.*cm,          // dx
                                                  10.*cm,          // dy
                                                  35.*cm);         // dz

    TrunkLogical = new G4LogicalVolume(Trunk_Solid,             // solid volume
                                       soft_tissue,              // material
                                       "Trunk_Logical");          // name

```

```

// set visibility to invisible
TrunkLogical->SetVisAttributes(G4VisAttributes::Invisible);
// set skin color
// TrunkLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
TrunkLogical->SetUserLimits(fStepLimit);

G4RotationMatrix* rm = new G4RotationMatrix();
rm->rotateX(180.*degree);
rm->rotateY(180.*degree);
TrunkPhysical = new G4PVPlacement(rm,                // rotation matrix
    G4ThreeVector(0, 0, 35.*cm), // translation vector wrt water phantom
    TrunkLogical,                // logical volume
    "Trunk_Physical",            // name
    pMotherLogical,              // mother volume
    false,                       // unused boolean
    0);                          // copy number

// calculate mass
G4double TrunkVol = TrunkLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Trunk = " << TrunkVol/cm3 << " cm^3" << G4endl;
G4String TrunkMat = TrunkLogical->GetMaterial()->GetName();
G4cout << "Material of Trunk = " << TrunkMat << G4endl;
G4double TrunkDensity = TrunkLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << TrunkDensity*cm3/g << " g/cm^3" << G4endl;
G4double TrunkMass = (TrunkVol)*TrunkDensity;
G4cout << "Mass of Trunk = " << TrunkMass/gram << " g" << G4endl;

return TrunkPhysical;
}

// _____//

// construct MIRD heart

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDHeart(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set heart volume
    G4Ellipsoid *Heart1 = new G4Ellipsoid("Heart1",    // name
        4.*cm,    // ax
        4.*cm,    // by
        7.*cm,    // cz
        -7.*cm,   // zcut1
        0);       // zcut2

    G4Sphere *Heart2 = new G4Sphere("Heart2",         // name
        0,    // rmin
        3.99*cm, // rmax
        0,    // start phi
        360.*deg, // delta phi
        0,    // start theta
        90.*deg); // delta theta

    G4UnionSolid *Heart_Solid = new G4UnionSolid("Heart_Solid", // name
        Heart1, // 1st volume to union
        Heart2); // 2nd volume to union

    HeartLogical = new G4LogicalVolume(Heart_Solid, // solid volume
        soft_tissue, // material
        "Heart_Logical"); // name

```

```

// set heart color
HeartLogical->SetVisAttributes(G4Colour(0.8078, 0.4314, 0.3294));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
HeartLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateY(25.*deg);
HeartPhysical = new G4PVPlacement(rm,                                     // rotation matrix
    G4ThreeVector(0, -3.*cm, 15.32*cm), // translation vector wrt trunk
    HeartLogical,                      // logical volume
    "Heart_Physical",                  // name
    pMotherLogical,                    // mother volume
    false,                             // unused boolean
    0);                                // copy number

// calculate mass
G4double HeartVol = HeartLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Heart = " << HeartVol/cm3 << " cm^3" << G4endl;
G4String HeartMat = HeartLogical->GetMaterial()->GetName();
G4cout << "Material of Heart = " << HeartMat << G4endl;
G4double HeartDensity = HeartLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << HeartDensity*cm3/g << " g/cm^3" << G4endl;
G4double HeartMass = (HeartVol)*HeartDensity;
G4cout << "Mass of Heart = " << HeartMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = HeartLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 7;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return HeartPhysical;
}

// _____//

// construct MIRD left lung
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftLung(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *lung = G4Material::GetMaterial("lung");

    // set left lung volumes
    G4Ellipsoid *LungLobe = new G4Ellipsoid("LungLobe",           // name
        5.*cm,             // ax
        7.5*cm,            // by
        24.*cm,            // cz
        0,                 // zcut1
        24.*cm);           // zcut2

    G4Ellipsoid *SubtrLung = new G4Ellipsoid("SubtrLung",         // name
        5.*cm,             // ax
        7.5*cm,            // by
        24.*cm);           // cz

    G4Box *Box = new G4Box("Box",                                 // name

```

```

        5.5*cm,          // dx
        8.5*cm,          // dy
        24.*cm);        // dz

G4SubtractionSolid *Section = new G4SubtractionSolid("Box", // name
        SubtrLung, // larger volume to subtract from
        Box,       // smaller volume to subtract
        0,         // rotation matrix
        G4ThreeVector(0, 8.5*cm, 0)); // relative position

G4SubtractionSolid *LeftLung_Solid = new G4SubtractionSolid("LeftLung_Solid", // name
        LungLobe, // larger volume to subtract from
        Section,  // smaller volume to subtract
        0,        // rotation matrix
        G4ThreeVector(-6.*cm, 0, 0)); // relative position

LeftLungLogical = new G4LogicalVolume(LeftLung_Solid, // solid volume
        lung, // material
        "LeftLung_Logical"); // name

// set lung color
LeftLungLogical->SetVisAttributes(G4Colour(0.7725, 0.647, 0.5686));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftLungLogical->SetUserLimits(fStepLimit);

LeftLungPhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(8.5*cm, 0, 8.5*cm), // translation vector wrt trunk
        LeftLungLogical, // logical volume
        "LeftLung_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

// calculate mass
G4double LeftLungVol = LeftLungLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Lung = " << (LeftLungVol)/cm3 << " cm^3" << G4endl;
G4String LeftLungMat = LeftLungLogical->GetMaterial()->GetName();
G4cout << "Material of Left Lung = " << LeftLungMat << G4endl;
G4double LeftLungDensity = LeftLungLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftLungDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftLungMass = (LeftLungVol)*LeftLungDensity;
G4cout << "Mass of Left Lung = " << LeftLungMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftLungLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 8;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftLungPhysical;
}

// _____//

// construct MIRD right lung

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightLung(G4LogicalVolume
*pMotherLogical)

```

```

{

// get materials
G4Material *lung = G4Material::GetMaterial("lung");

// set right lung volumes
G4Ellipsoid *LungLobe = new G4Ellipsoid("LungLobe",      // name
                                         5.*cm,           // ax
                                         7.5*cm,          // by
                                         24.*cm,          // cz
                                         0,               // zcut1
                                         24.*cm);         // zcut2

G4Ellipsoid *SubtrLung = new G4Ellipsoid("SubtrLung",     // name
                                         5.*cm,           // ax
                                         7.5*cm,          // by
                                         24.*cm);         // cz

G4Box *Box = new G4Box("Box",                          // name
                       5.5*cm,                          // dx
                       8.5*cm,                          // dy
                       24.*cm);                         // dz

G4SubtractionSolid *Section = new G4SubtractionSolid("Box", // name
                                                     SubtrLung, // larger volume to subtract from
                                                     Box,      // smaller volume to subtract
                                                     0,        // rotation matrix
                                                     G4ThreeVector(0, 8.5*cm, 0)); // relative position

G4SubtractionSolid *RightLung_Solid = new G4SubtractionSolid("RightLung_Solid", // name
                                                             LungLobe, // larger volume to subtract from
                                                             Section, // smaller volume to subtract
                                                             0,        // rotation matrix
                                                             G4ThreeVector(6.*cm, 0, 0)); // relative position

RightLungLogical = new G4LogicalVolume(RightLung_Solid,      // solid volume
                                       lung,                  // material
                                       "RightLung_Logical");  // name

// set lung color
RightLungLogical->SetVisAttributes(G4Colour(0.7725, 0.647, 0.5686));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
RightLungLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateZ(-360.*deg);
RightLungPhysical = new G4PVPlacement(rm,                  // rotation matrix
                                       G4ThreeVector(-8.5*cm, 0, 8.5*cm), // translation vector wrt trunk
                                       RightLungLogical,    // logical volume
                                       "RightLung_Physical", // name
                                       pMotherLogical,       // mother volume
                                       false,                // unused boolean
                                       0);                   // copy number

// calculate mass
G4double RightLungVol = RightLungLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Lung = " << (RightLungVol)/cm3 << " cm^3" << G4endl;
G4String RightLungMat = RightLungLogical->GetMaterial()->GetName();
G4cout << "Material of Right Lung = " << RightLungMat << G4endl;
G4double RightLungDensity = RightLungLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightLungDensity*cm3/g << " g/cm^3" << G4endl;
G4double RightLungMass = (RightLungVol)*RightLungDensity;
G4cout << "Mass of Right Lung = " << RightLungMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = RightLungLogical;

```

```

G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 9;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return RightLungPhysical;
}

// _____//

// construct MIRD left kidney

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftKidney(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set left kidney volume
    G4Ellipsoid *Kidney = new G4Ellipsoid("Kidney", // name
                                           4.5*cm, // ax
                                           1.5*cm, // by
                                           5.5*cm); // cz

    G4Box *SubtrKidney = new G4Box("SubtrKidney", // name
                                    3.*cm, // dx
                                    6.*cm, // dy
                                    6.*cm); // dz

    G4SubtractionSolid *LeftKidney_Solid = new G4SubtractionSolid("LeftKidney_Solid", // name
                                                                    Kidney, // larger volume to subtract from
                                                                    SubtrKidney, // smaller volume to subtract
                                                                    0, // rotation matrix
                                                                    G4ThreeVector(-6.0*cm,0,0)); // relative position

    LeftKidneyLogical = new G4LogicalVolume(LeftKidney_Solid, // solid volume
                                           soft_tissue, // material
                                           "LeftKidney_Logical"); // name

    // set kidney color
    LeftKidneyLogical->SetVisAttributes(G4Colour(0.7255, 0.4, 0.3255));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftKidneyLogical->SetUserLimits(fStepLimit);

    LeftKidneyPhysical = new G4PVPlacement(0, // rotation matrix
                                           G4ThreeVector(6.*cm, 6.*cm, -2.5*cm), // translation vector wrt trunk
                                           LeftKidneyLogical, // logical volume
                                           "LeftKidney_Physical", // name
                                           pMotherLogical, // mother volume
                                           false, // unused boolean
                                           0); // copy number

    // calculate mass
    G4double LeftKidneyVol = LeftKidneyLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Kidney = " << LeftKidneyVol/cm3 << " cm^3" << G4endl;
    G4String LeftKidneyMat = LeftKidneyLogical->GetMaterial()->GetName();
    G4cout << "Material of Left Kidney = " << LeftKidneyMat << G4endl;
    G4double LeftKidneyDensity = LeftKidneyLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << LeftKidneyDensity*cm3/g << " g/cm^3" << G4endl;

```

```

G4double LeftKidneyMass = (LeftKidneyVol)*LeftKidneyDensity;
G4cout << "Mass of Left Kidney = " << LeftKidneyMass/gram << " g" << G4endl;

// set organ info in maps
G4LogicalVolume* currentLogical = LeftKidneyLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 10;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftKidneyPhysical;
}

// _____//

// construct MIRD right kidney

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightKidney(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set right kidney volume
    G4Ellipsoid *Kidney = new G4Ellipsoid("Kidney", // name
                                           4.5*cm, // ax
                                           1.5*cm, // by
                                           5.5*cm); // cz

    G4Box *SubtrKidney = new G4Box("SubtrKidney", // name
                                   3.*cm, // dx
                                   6.*cm, // dy
                                   6.*cm); // dz

    G4SubtractionSolid *RightKidney_Solid = new G4SubtractionSolid("RightKidney_Solid", // name
                                                                    Kidney, // larger volume to subtract from
                                                                    SubtrKidney, // smaller volume to subtract
                                                                    0, // rotation matrix
                                                                    G4ThreeVector(6.0*cm,0,0)); // relative position

    RightKidneyLogical = new G4LogicalVolume(RightKidney_Solid, // solid volume
                                             soft_tissue, // material
                                             "RightKidney_Logical"); // name

    // set kidney color
    RightKidneyLogical->SetVisAttributes(G4Colour(0.7255, 0.4, 0.3255));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightKidneyLogical->SetUserLimits(fStepLimit);

    RightKidneyPhysical = new G4PVPlacement(0, // rotation matrix
                                           G4ThreeVector(-6.*cm,6.*cm,-2.5*cm), // translation vector wrt trunk
                                           RightKidneyLogical, // logical volume
                                           "RightKidney_Physical", // name
                                           pMotherLogical, // mother volume
                                           false, // unused boolean
                                           0); // copy number

    // calculate mass

```

```

G4double RightKidneyVol = RightKidneyLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Kidney = " << RightKidneyVol/cm3 << " cm^3" << G4endl;
G4String RightKidneyMat = RightKidneyLogical->GetMaterial()->GetName();
G4cout << "Material of Right Kidney = " << RightKidneyMat << G4endl;
G4double RightKidneyDensity = RightKidneyLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightKidneyDensity*cm3/g << " g/cm^3" << G4endl;
G4double RightKidneyMass = (RightKidneyVol)*RightKidneyDensity;
G4cout << "Mass of RightKidney = " << RightKidneyMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = RightKidneyLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 11;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightKidneyPhysical;
}

// _____//

// construct MIRD left adrenal

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftAdrenal(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set left adrenal volume
    G4VSolid *LeftAdrenal_Solid = new G4Ellipsoid("LeftAdrenal_Solid",           // name
                                                1.5*cm,                          // ax
                                                0.5*cm,                          // by
                                                5.*cm,                          // cz
                                                0,                               // zcut1
                                                5.*cm);                          // zcut2

    LeftAdrenalLogical = new G4LogicalVolume(LeftAdrenal_Solid,                 // solid volume
                                             soft_tissue,                       // material
                                             "LeftAdrenal_Logical");             // name

    // set adrenal color
    LeftAdrenalLogical->SetVisAttributes(G4Colour(0.9765, 0.7294, 0.5882));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftAdrenalLogical->SetUserLimits(fStepLimit);

    LeftAdrenalPhysical = new G4PVPlacement(0,                                // rotation matrix
                                             G4ThreeVector(4.5*cm, 6.5*cm, 3.*cm), // translation vector wrt trunk
                                             LeftAdrenalLogical,                  // logical volume
                                             "LeftAdrenal_Physical",              // name
                                             pMotherLogical,                     // mother volume
                                             false,                             // unused boolean
                                             0);                                // copy number

    // calculate mass
    G4double LeftAdrenalVol = LeftAdrenalLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Adrenal = " << LeftAdrenalVol/cm3 << " cm^3" << G4endl;
    G4String LeftAdrenalMat = LeftAdrenalLogical->GetMaterial()->GetName();

```

```

G4cout << "Material of Left Adrenal = " << LeftAdrenalMat << G4endl;
G4double LeftAdrenalDensity = LeftAdrenalLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftAdrenalDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftAdrenalMass = (LeftAdrenalVol)*LeftAdrenalDensity;
G4cout << "Mass of Left Adrenal = " << LeftAdrenalMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftAdrenalLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 12;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftAdrenalPhysical;
}

// _____//

// construct MIRD right adrenal

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightAdrenal(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set right adrenal volume
    G4VSolid *RightAdrenal_Solid = new G4Ellipsoid("RightAdrenal_Solid",           // name
                                                    1.5*cm,                        // ax
                                                    0.5*cm,                        // by
                                                    5.*cm,                        // cz
                                                    0,                            // zcut1
                                                    5.*cm);                       // zcut2

    RightAdrenalLogical = new G4LogicalVolume(RightAdrenal_Solid,                // solid volume
                                              soft_tissue,                        // material
                                              "RightAdrenal_Logical");             // name

    // set adrenal color
    RightAdrenalLogical->SetVisAttributes(G4Colour(0.9765, 0.7294, 0.5882));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightAdrenalLogical->SetUserLimits(fStepLimit);

    RightAdrenalPhysical = new G4PVPlacement(0,                                // rotation matrix
                                              G4ThreeVector(-4.5*cm, 6.5*cm, 3.*cm), // translation vector wrt trunk
                                              RightAdrenalLogical,                // logical volume
                                              "RightAdrenal_Physical",             // name
                                              pMotherLogical,                      // mother volume
                                              false,                              // unused boolean
                                              0);                                  // copy number

    // calculate mass
    G4double RightAdrenalVol = RightAdrenalLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Adrenal = " << RightAdrenalVol/cm3 << " cm^3" << G4endl;
    G4String RightAdrenalMat = RightAdrenalLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Adrenal = " << RightAdrenalMat << G4endl;
    G4double RightAdrenalDensity = RightAdrenalLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightAdrenalDensity*cm3/g << " g/cm^3" << G4endl;

```

```

G4double RightAdrenalMass = (RightAdrenalVol)*RightAdrenalDensity;
G4cout << "Mass of Right Adrenal = " << RightAdrenalMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = RightAdrenalLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 13;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightAdrenalPhysical;
}

// _____//

// construct MIRD rib cage

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRibCage(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set rib cage volumes
    G4EllipticalTube *Outer_Cage = new G4EllipticalTube("Outer_Cage", // name
        17*cm, // dx
        9.8*cm, // dy
        16.2*cm); // dz

    G4EllipticalTube *Inner_Cage = new G4EllipticalTube("Inner_Cage", // name
        16.4*cm, // dx
        9.2*cm, // dy
        17.*cm); // dz

    G4SubtractionSolid *Ribcage = new G4SubtractionSolid("Cage", // name
        Outer_Cage, // larger volume to subtract from
        Inner_Cage); // smaller volume to subtract

    RibCageLogical = new G4LogicalVolume(Ribcage, // solid volume
        soft_tissue, // material
        "Ribcage_Logical"); // name

    // set visibility to invisible
    RibCageLogical->SetVisAttributes(G4VisAttributes::Invisible);

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RibCageLogical->SetUserLimits(fStepLimit);

    RibCagePhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(0,0,16.2*cm), // translation vector wrt trunk
        RibCageLogical, // logical volume
        "Ribcage_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    // calculate mass
    G4double RibCageVol = RibCageLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib Cage = " << RibCageVol/cm3 << " cm^3" << G4endl;
    G4String RibCageMat = RibCageLogical->GetMaterial()->GetName();

```

```

G4cout << "Material of Rib Cage = " << RibCageMat << G4endl;
G4double RibCageDensity = RibCageLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibCageDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibCageMass = (RibCageVol)*RibCageDensity;
G4cout << "Mass of Rib Cage = " << RibCageMass/gram << " g" << G4endl;

return RibCagePhysical;
}

// _____//

// construct MIRD rib 1

G4VPPhysicalVolume * DetectorConstruction::ConstructMIRDRib1(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 1 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib1Logical = new G4LogicalVolume(Rib, // solid volume
                                     g4bone, // material
                                     "Rib1_Logical"); // name

    // set bone color
    Rib1Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib1Logical->SetUserLimits(fStepLimit);

    Rib1Physical = new G4PVPlacement(0, // rotation matrix
                                     G4ThreeVector(0, 0, -15.3*cm), // translation vector wrt rib cage
                                     Rib1Logical, // logical volume
                                     "Rib1_Physical", // name
                                     pMotherLogical, // mother volume
                                     false, // unused boolean
                                     0); // copy number

    // calculate mass
    G4double RibVol = Rib1Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 1 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib1Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 1 = " << RibMat << G4endl;
    G4double RibDensity = Rib1Logical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMass = (RibVol)*RibDensity;
    G4cout << "Mass of Rib 1 = " << RibMass/gram << " g" << G4endl;

    return Rib1Physical;
}

// _____//

```

```

// construct rib 1 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib1Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 1 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib1MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                             red_marrow, // material
                                             "Rib1Marrow_Logical"); // name

    // set red marrow color
    Rib1MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib1MarrowLogical->SetUserLimits(fStepLimit);

    Rib1MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                             G4ThreeVector(), // translation vector wrt rib 1
                                             Rib1MarrowLogical, // logical volume
                                             "Rib1Marrow_Physical", // name
                                             pMotherLogical, // mother volume
                                             false, // unused boolean
                                             0); // copy number

    // calculate mass of rib 1 marrow
    G4double RibMarrowVol = Rib1MarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 1 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
    G4String RibMarrowMat = Rib1MarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Rib 1 Marrow = " << RibMarrowMat << G4endl;
    G4double RibMarrowDensity = Rib1MarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
    G4cout << "Mass of Rib 1 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

    // store organ information in maps
    G4LogicalVolume* currentLogical = Rib1MarrowLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 14;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;
}

```

```

        return Rib1MarrowPhysical;
    }

// _____//

// construct MIRD rib 2

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib2(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 2 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib2Logical = new G4LogicalVolume(Rib, // solid volume
                                      g4bone, // material
                                      "Rib2_Logical"); // name

    // set bone color
    Rib2Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib2Logical->SetUserLimits(fStepLimit);

    Rib2Physical = new G4PVPlacement(0, // rotation matrix
                                      G4ThreeVector(0, 0, -13.0*cm), // translation vector wrt rib cage
                                      Rib2Logical, // logical volume
                                      "Rib2_Physical", // name
                                      pMotherLogical, // mother volume
                                      false, // unused boolean
                                      0); // copy number

    // calculate mass
    G4double RibVol = Rib2Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 2 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib2Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 2 = " << RibMat << G4endl;
    G4double RibDensity = Rib2Logical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMass = (RibVol)*RibDensity;
    G4cout << "Mass of Rib 2 = " << RibMass/gram << " g" << G4endl;

    return Rib2Physical;
}

// _____//

// construct rib 2 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib2Marrow(G4LogicalVolume *pMotherLogical)
{

```

```

// get materials
G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

// set rib 2 active marrow volume
G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                         16.85*cm, // dx
                                                         9.65*cm, // dy
                                                         0.699*cm); // dz

G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                       16.65*cm, // dx
                                                       9.45*cm, // dy
                                                       0.75*cm); // dz

G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                       RibMarrow_Out, // larger volume to subtract from
                                                       RibMarrow_In); // smaller volume to subtract

Rib2MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                         red_marrow, // material
                                         "Rib2Marrow_Logical"); // name

// set red marrow color
Rib2MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib2MarrowLogical->SetUserLimits(fStepLimit);

Rib2MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                       G4ThreeVector(), // translation vector wrt rib 2
                                       Rib2MarrowLogical, // logical volume
                                       "Rib2Marrow_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

// calculate mass of rib 2 marrow
G4double RibMarrowVol = Rib2MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 2 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib2MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 2 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib2MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 2 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib2MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 15;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib2MarrowPhysical;
}

// _____//

// construct MIRD rib 3

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib3(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 3 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib3Logical = new G4LogicalVolume(Rib, // solid volume
                                      g4bone, // material
                                      "Rib3_Logical"); // name

    // set bone color
    Rib3Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib3Logical->SetUserLimits(fStepLimit);

    Rib3Physical = new G4PVPlacement(0, // rotation matrix
                                     G4ThreeVector(0, 0, -10.7*cm), // translation vector wrt rib cage
                                     Rib3Logical, // logical volume
                                     "Rib3_Physical", // name
                                     pMotherLogical, // mother volume
                                     false, // unused boolean
                                     0); // copy number

    // calculate mass
    G4double RibVol = Rib3Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 3 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib3Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 3 = " << RibMat << G4endl;
    G4double RibDensity = Rib3Logical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMass = (RibVol)*RibDensity;
    G4cout << "Mass of Rib 3 = " << RibMass/gram << " g" << G4endl;

    return Rib3Physical;
}

// _____//

// construct rib 3 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib3Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 3 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

```

```

G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In",      // name
                                                    16.65*cm,      // dx
                                                    9.45*cm,      // dy
                                                    0.75*cm);      // dz

G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                    RibMarrow_Out, // larger volume to subtract from
                                                    RibMarrow_In); // smaller volume to subtract

Rib3MarrowLogical = new G4LogicalVolume(RibMarrow,      // solid volume
                                        red_marrow,      // material
                                        "Rib3Marrow_Logical"); // name

// set red marrow color
Rib3MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib3MarrowLogical->SetUserLimits(fStepLimit);

Rib3MarrowPhysical = new G4PVPlacement(0,      // rotation matrix
                                        G4ThreeVector(), // translation vector wrt rib 3
                                        Rib3MarrowLogical, // logical volume
                                        "Rib3Marrow_Physical", // name
                                        pMotherLogical, // mother volume
                                        false, // unused boolean
                                        0); // copy number

// calculate mass of rib 3 marrow
G4double RibMarrowVol = Rib3MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 3 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib3MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 3 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib3MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 3 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib3MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 16;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib3MarrowPhysical;
}

// _____//

// construct MIRD rib 4

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib4(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 4 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out",      // name

```

```

        17.*cm,      // dx
        9.8*cm,     // dy
        0.7*cm);    // dz

G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In",      // name
        16.5*cm,     // dx
        9.3*cm,      // dy
        0.75*cm);    // dz

G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
        Rib_Out, // larger volume to subtract from
        Rib_In); // smaller volume to subtract

Rib4Logical = new G4LogicalVolume(Rib,                      // solid volume
        g4bone,      // material
        "Rib4_Logical"); // name

// set bone color
Rib4Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib4Logical->SetUserLimits(fStepLimit);

Rib4Physical = new G4PVPlacement(0,                        // rotation matrix
        G4ThreeVector(0, 0, -7.9*cm), // translation vector wrt rib cage
        Rib4Logical, // logical volume
        "Rib4_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

// calculate mass
G4double RibVol = Rib4Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 4 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib4Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 4 = " << RibMat << G4endl;
G4double RibDensity = Rib4Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 4 = " << RibMass/gram << " g" << G4endl;

return Rib4Physical;
}

// _____//

// construct rib 4 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib4Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 4 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
        16.85*cm, // dx
        9.65*cm,  // dy
        0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
        16.65*cm, // dx
        9.45*cm,  // dy
        0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
        RibMarrow_Out, // larger volume to subtract from

```

```

        RibMarrow_In); // smaller volume to subtract

Rib4MarrowLogical = new G4LogicalVolume(RibMarrow,           // solid volume
        red_marrow,           // material
        "Rib4Marrow_Logical"); // name

// set red marrow color
Rib4MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib4MarrowLogical->SetUserLimits(fStepLimit);

Rib4MarrowPhysical = new G4PVPlacement(0,                  // rotation matrix
        G4ThreeVector(), // translation vector wrt rib 4
        Rib4MarrowLogical, // logical volume
        "Rib4Marrow_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

// calculate mass of rib 4 marrow
G4double RibMarrowVol = Rib4MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 4 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib4MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 4 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib4MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 4 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib4MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 17;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib4MarrowPhysical;
}

// _____//

// construct MIRD rib 5

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib5(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 5 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
        17.*cm, // dx
        9.8*cm, // dy
        0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
        16.5*cm, // dx
        9.3*cm, // dy
        0.75*cm); // dz

```

```

G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                Rib_Out, // larger volume to subtract from
                                                Rib_In); // smaller volume to subtract

Rib5Logical = new G4LogicalVolume(Rib, // solid volume
                                  g4bone, // material
                                  "Rib5_Logical"); // name

// set bone color
Rib5Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib5Logical->SetUserLimits(fStepLimit);

Rib5Physical = new G4PVPlacement(0, // rotation matrix
                                G4ThreeVector(0, 0, -5.1*cm), // translation vector wrt rib cage
                                Rib5Logical, // logical volume
                                "Rib5_Physical", // name
                                pMotherLogical, // mother volume
                                false, // unused boolean
                                0); // copy number

// calculate mass
G4double RibVol = Rib5Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 5 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib5Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 5 = " << RibMat << G4endl;
G4double RibDensity = Rib5Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 5 = " << RibMass/gram << " g" << G4endl;

return Rib5Physical;
}

// _____//

// construct rib 5 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib5Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 5 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib5MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                            red_marrow, // material
                                            "Rib5Marrow_Logical"); // name

    // set red marrow color
    Rib5MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));
}

```

```

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib5MarrowLogical->SetUserLimits(fStepLimit);

Rib5MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                       G4ThreeVector(), // translation vector wrt rib 5
                                       Rib5MarrowLogical, // logical volume
                                       "Rib5Marrow_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

// calculate mass of rib 5 marrow
G4double RibMarrowVol = Rib5MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 5 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib5MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 5 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib5MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 5 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib5MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 18;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib5MarrowPhysical;
}

// _____//

// construct MIRD rib 6

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib6(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 6 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib6Logical = new G4LogicalVolume(Rib, // solid volume
                                      g4bone, // material
                                      "Rib6_Logical"); // name

```

```

// set bone color
Rib6Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib6Logical->SetUserLimits(fStepLimit);

Rib6Physical = new G4PVPlacement(0, // rotation matrix
                                G4ThreeVector(0, 0, -2.3*cm), // translation vector wrt rib cage
                                Rib6Logical, // logical volume
                                "Rib6_Physical", // name
                                pMotherLogical, // mother volume
                                false, // unused boolean
                                0); // copy number

// calculate mass
G4double RibVol = Rib6Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 6 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib6Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 6 = " << RibMat << G4endl;
G4double RibDensity = Rib6Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 6 = " << RibMass/gram << " g" << G4endl;

return Rib6Physical;
}

// _____//

// construct rib 6 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib6Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 6 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib6MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                             red_marrow, // material
                                             "Rib6Marrow_Logical"); // name

    // set red marrow color
    Rib6MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib6MarrowLogical->SetUserLimits(fStepLimit);

    Rib6MarrowPhysical = new G4PVPlacement(0, // rotation matrix

```

```

        G4ThreeVector(),      // translation vector wrt rib 6
        Rib6MarrowLogical,    // logical volume
        "Rib6Marrow_Physical", // name
        pMotherLogical,       // mother volume
        false,                // unused boolean
        0);                   // copy number

// calculate mass of rib 6 marrow
G4double RibMarrowVol = Rib6MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 6 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib6MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 6 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib6MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 6 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib6MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 19;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib6MarrowPhysical;
}

// _____//

// construct MIRD rib 7

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib7(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 7 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out",      // name
        17.*cm,              // dx
        9.8*cm,              // dy
        0.7*cm);             // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In",        // name
        16.5*cm,             // dx
        9.3*cm,              // dy
        0.75*cm);            // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
        Rib_Out, // larger volume to subtract from
        Rib_In); // smaller volume to subtract

    Rib7Logical = new G4LogicalVolume(Rib, // solid volume
        g4bone, // material
        "Rib7_Logical"); // name

    // set bone color
    Rib7Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);

```

```

Rib7Logical->SetUserLimits(fStepLimit);

Rib7Physical = new G4PVPlacement(0, // rotation matrix
                                G4ThreeVector(0, 0, 0.5*cm), // translation vector wrt rib cage
                                Rib7Logical, // logical volume
                                "Rib7_Physical", // name
                                pMotherLogical, // mother volume
                                false, // unused boolean
                                0); // copy number

// calculate mass
G4double RibVol = Rib7Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 7 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib7Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 7 = " << RibMat << G4endl;
G4double RibDensity = Rib7Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 7 = " << RibMass/gram << " g" << G4endl;

return Rib7Physical;
}

// _____//

// construct rib 7 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib7Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 7 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib7MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                            red_marrow, // material
                                            "Rib7Marrow_Logical"); // name

    // set red marrow color
    Rib7MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib7MarrowLogical->SetUserLimits(fStepLimit);

    Rib7MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                            G4ThreeVector(), // translation vector wrt rib 7
                                            Rib7MarrowLogical, // logical volume
                                            "Rib7Marrow_Physical", // name
                                            pMotherLogical, // mother volume
                                            false, // unused boolean
                                            0); // copy number

    // calculate mass of rib 7 marrow

```

```

G4double RibMarrowVol = Rib7MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 7 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib7MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 7 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib7MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 7 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib7MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 20;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib7MarrowPhysical;
}

// _____//

// construct MIRD rib 8
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib8(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 8 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib8Logical = new G4LogicalVolume(Rib, // solid volume
                                      g4bone, // material
                                      "Rib8_Logical"); // name

    // set bone color
    Rib8Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib8Logical->SetUserLimits(fStepLimit);

    Rib8Physical = new G4PVPlacement(0, // rotation matrix
                                      G4ThreeVector(0, 0, 3.3*cm), // translation vector wrt rib cage
                                      Rib8Logical, // logical volume
                                      "Rib8_Physical", // name
                                      pMotherLogical, // mother volume
                                      false, // unused boolean

```

```

        0); // copy number

// calculate mass
G4double RibVol = Rib8Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 8 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib8Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 8 = " << RibMat << G4endl;
G4double RibDensity = Rib8Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 8 = " << RibMass/gram << " g" << G4endl;

return Rib8Physical;
}

// _____//

// construct rib 8 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib8Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 8 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib8MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                             red_marrow, // material
                                             "Rib8Marrow_Logical"); // name

    // set red marrow color
    Rib8MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib8MarrowLogical->SetUserLimits(fStepLimit);

    Rib8MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                             G4ThreeVector(), // translation vector wrt rib 8
                                             Rib8MarrowLogical, // logical volume
                                             "Rib8Marrow_Physical", // name
                                             pMotherLogical, // mother volume
                                             false, // unused boolean
                                             0); // copy number

    // calculate mass of rib 8 marrow
    G4double RibMarrowVol = Rib8MarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 8 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
    G4String RibMarrowMat = Rib8MarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Rib 8 Marrow = " << RibMarrowMat << G4endl;
    G4double RibMarrowDensity = Rib8MarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
    G4cout << "Mass of Rib 8 Marrow = " << RibMarrowMass/gram << " g" << G4endl;
}

```

```

// store organ information in maps
G4LogicalVolume* currentLogical = Rib8MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 21;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib8MarrowPhysical;
}

// _____//

// construct MIRD rib 9
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib9(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 9 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out",      // name
                                                    17.*cm,           // dx
                                                    9.8*cm,           // dy
                                                    0.7*cm);          // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In",        // name
                                                    16.5*cm,          // dx
                                                    9.3*cm,           // dy
                                                    0.75*cm);         // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib",          // name
                                                    Rib_Out,          // larger volume to subtract from
                                                    Rib_In);          // smaller volume to subtract

    Rib9Logical = new G4LogicalVolume(Rib,                          // solid volume
                                      g4bone,                       // material
                                      "Rib9_Logical");               // name

    // set bone color
    Rib9Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib9Logical->SetUserLimits(fStepLimit);

    Rib9Physical = new G4PVPlacement(0,                             // rotation matrix
                                     G4ThreeVector(0, 0, 6.1*cm), // translation vector wrt rib cage
                                     Rib9Logical,                  // logical volume
                                     "Rib9_Physical",              // name
                                     pMotherLogical,               // mother volume
                                     false,                         // unused boolean
                                     0);                           // copy number

    // calculate mass
    G4double RibVol = Rib9Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 9 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib9Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 9 = " << RibMat << G4endl;
    G4double RibDensity = Rib9Logical->GetMaterial()->GetDensity();

```

```

G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 9 = " << RibMass/gram << " g" << G4endl;

return Rib9Physical;
}

// _____//
// construct rib 9 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib9Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 9 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                            16.65*cm, // dx
                                                            9.45*cm, // dy
                                                            0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                            RibMarrow_Out, // larger volume to subtract from
                                                            RibMarrow_In); // smaller volume to subtract

    Rib9MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                             red_marrow, // material
                                             "Rib9Marrow_Logical"); // name

    // set red marrow color
    Rib9MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib9MarrowLogical->SetUserLimits(fStepLimit);

    Rib9MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                             G4ThreeVector(), // translation vector wrt rib 9
                                             Rib9MarrowLogical, // logical volume
                                             "Rib9Marrow_Physical", // name
                                             pMotherLogical, // mother volume
                                             false, // unused boolean
                                             0); // copy number

    // calculate mass of rib 9 marrow
    G4double RibMarrowVol = Rib9MarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 9 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
    G4String RibMarrowMat = Rib9MarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Rib 9 Marrow = " << RibMarrowMat << G4endl;
    G4double RibMarrowDensity = Rib9MarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
    G4cout << "Mass of Rib 9 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

    // store organ information in maps
    G4LogicalVolume* currentLogical = Rib9MarrowLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 22;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
}

```

```

    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return Rib9MarrowPhysical;
}

// _____//
// construct MIRD rib 10
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib10(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 10 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out", // name
                                                    17.*cm, // dx
                                                    9.8*cm, // dy
                                                    0.7*cm); // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In", // name
                                                    16.5*cm, // dx
                                                    9.3*cm, // dy
                                                    0.75*cm); // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib10Logical = new G4LogicalVolume(Rib, // solid volume
                                       g4bone, // material
                                       "Rib10_Logical"); // name

    // set bone color
    Rib10Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib10Logical->SetUserLimits(fStepLimit);

    Rib10Physical = new G4PVPlacement(0, // rotation matrix
                                       G4ThreeVector(0, 0, 8.9*cm), // translation vector wrt rib cage
                                       Rib10Logical, // logical volume
                                       "Rib10_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

    // calculate mass
    G4double RibVol = Rib10Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 10 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib10Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 10 = " << RibMat << G4endl;
    G4double RibDensity = Rib10Logical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMass = (RibVol)*RibDensity;
    G4cout << "Mass of Rib 10 = " << RibMass/gram << " g" << G4endl;

    return Rib10Physical;
}

// _____//

```

```

// construct rib 10 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib10Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 10 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
                                                            16.85*cm, // dx
                                                            9.65*cm, // dy
                                                            0.699*cm); // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
                                                          16.65*cm, // dx
                                                          9.45*cm, // dy
                                                          0.75*cm); // dz

    G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
                                                           RibMarrow_Out, // larger volume to subtract from
                                                           RibMarrow_In); // smaller volume to subtract

    Rib10MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
                                              red_marrow, // material
                                              "Rib10Marrow_Logical"); // name

    // set red marrow color
    Rib10MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib10MarrowLogical->SetUserLimits(fStepLimit);

    Rib10MarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                              G4ThreeVector(), // translation vector wrt rib 10
                                              Rib10MarrowLogical, // logical volume
                                              "Rib10Marrow_Physical", // name
                                              pMotherLogical, // mother volume
                                              false, // unused boolean
                                              0); // copy number

    // calculate mass of rib 10 marrow
    G4double RibMarrowVol = Rib10MarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 10 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
    G4String RibMarrowMat = Rib10MarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Rib 10 Marrow = " << RibMarrowMat << G4endl;
    G4double RibMarrowDensity = Rib10MarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
    G4cout << "Mass of Rib 10 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

    // store organ information in maps
    G4LogicalVolume* currentLogical = Rib10MarrowLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 23;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return Rib10MarrowPhysical;
}

```

```

}

//_____//

// construct MIRD rib 11

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib11(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set rib 11 volume
    G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out",    // name
                                                    17.*cm,          // dx
                                                    9.8*cm,          // dy
                                                    0.7*cm);         // dz

    G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In",      // name
                                                    16.5*cm,         // dx
                                                    9.3*cm,          // dy
                                                    0.75*cm);         // dz

    G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib",        // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

    Rib11Logical = new G4LogicalVolume(Rib,                        // solid volume
                                       g4bone,                    // material
                                       "Rib11_Logical");           // name

    // set bone color
    Rib11Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    Rib11Logical->SetUserLimits(fStepLimit);

    Rib11Physical = new G4PVPlacement(0,                          // rotation matrix
                                       G4ThreeVector(0, 0, 11.7*cm), // translation vector wrt rib cage
                                       Rib11Logical,                // logical volume
                                       "Rib11_Physical",            // name
                                       pMotherLogical,              // mother volume
                                       false,                       // unused boolean
                                       0);                          // copy number

    // calculate mass
    G4double RibVol = Rib11Logical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Rib 11 = " << RibVol/cm3 << " cm^3" << G4endl;
    G4String RibMat = Rib11Logical->GetMaterial()->GetName();
    G4cout << "Material of Rib 11 = " << RibMat << G4endl;
    G4double RibDensity = Rib11Logical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RibMass = (RibVol)*RibDensity;
    G4cout << "Mass of Rib 11 = " << RibMass/gram << " g" << G4endl;

    return Rib11Physical;
}

//_____//

// construct rib 11 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib11Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

```

```

// set rib 11 active marrow volume
G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out", // name
    16.85*cm, // dx
    9.65*cm, // dy
    0.699*cm); // dz

G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In", // name
    16.65*cm, // dx
    9.45*cm, // dy
    0.75*cm); // dz

G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
    RibMarrow_Out, // larger volume to subtract from
    RibMarrow_In); // smaller volume to subtract

Rib11MarrowLogical = new G4LogicalVolume(RibMarrow, // solid volume
    red_marrow, // material
    "Rib11Marrow_Logical"); // name

// set red marrow color
Rib11MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib11MarrowLogical->SetUserLimits(fStepLimit);

Rib11MarrowPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(), // translation vector wrt rib 11
    Rib11MarrowLogical, // logical volume
    "Rib11Marrow_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

// calculate mass of rib 11 marrow
G4double RibMarrowVol = Rib11MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 11 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib11MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 11 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib11MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 11 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib11MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 24;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib11MarrowPhysical;
}

// _____//

// construct MIRD rib 12

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRib12(G4LogicalVolume *pMotherLogical)
{

```

```

// get materials
G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

// set rib 12 volume
G4EllipticalTube *Rib_Out = new G4EllipticalTube("Rib_Out",    // name
                                                    17.*cm,        // dx
                                                    9.8*cm,        // dy
                                                    0.7*cm);        // dz

G4EllipticalTube *Rib_In = new G4EllipticalTube("Rib_In",      // name
                                                    16.5*cm,       // dx
                                                    9.3*cm,        // dy
                                                    0.75*cm);       // dz

G4SubtractionSolid *Rib = new G4SubtractionSolid("Rib", // name
                                                    Rib_Out, // larger volume to subtract from
                                                    Rib_In); // smaller volume to subtract

Rib12Logical = new G4LogicalVolume(Rib,                // solid volume
                                    g4bone,             // material
                                    "Rib12_Logical");    // name

// set bone color
Rib12Logical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib12Logical->SetUserLimits(fStepLimit);

Rib12Physical = new G4PVPlacement(0,                  // rotation matrix
                                   G4ThreeVector(0, 0, 14.5*cm), // translation vector wrt rib cage
                                   Rib12Logical,        // logical volume
                                   "Rib12_Physical",    // name
                                   pMotherLogical,      // mother volume
                                   false,               // unused boolean
                                   0);                  // copy number

// calculate mass
G4double RibVol = Rib12Logical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 12 = " << RibVol/cm3 << " cm^3" << G4endl;
G4String RibMat = Rib12Logical->GetMaterial()->GetName();
G4cout << "Material of Rib 12 = " << RibMat << G4endl;
G4double RibDensity = Rib12Logical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMass = (RibVol)*RibDensity;
G4cout << "Mass of Rib 12 = " << RibMass/gram << " g" << G4endl;

return Rib12Physical;
}

// _____//

// construct rib 12 active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRib12Marrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set rib 12 active marrow volume
    G4EllipticalTube *RibMarrow_Out = new G4EllipticalTube("RibMarrow_Out",    // name
                                                            16.85*cm,        // dx
                                                            9.65*cm,        // dy
                                                            0.699*cm);       // dz

    G4EllipticalTube *RibMarrow_In = new G4EllipticalTube("RibMarrow_In",      // name

```

```

        16.65*cm,          // dx
        9.45*cm,          // dy
        0.75*cm);         // dz

G4SubtractionSolid *RibMarrow = new G4SubtractionSolid("RibMarrow", // name
    RibMarrow_Out, // larger volume to subtract from
    RibMarrow_In); // smaller volume to subtract

Rib12MarrowLogical = new G4LogicalVolume(RibMarrow,          // solid volume
    red_marrow,      // material
    "Rib12Marrow_Logical"); // name

// set red marrow color
Rib12MarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
Rib12MarrowLogical->SetUserLimits(fStepLimit);

Rib12MarrowPhysical = new G4PVPlacement(0,                  // rotation matrix
    G4ThreeVector(), // translation vector wrt rib 12
    Rib12MarrowLogical, // logical volume
    "Rib12Marrow_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

// calculate mass of rib 12 marrow
G4double RibMarrowVol = Rib12MarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Rib 12 Marrow = " << RibMarrowVol/cm3 << " cm^3" << G4endl;
G4String RibMarrowMat = Rib12MarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Rib 12 Marrow = " << RibMarrowMat << G4endl;
G4double RibMarrowDensity = Rib12MarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RibMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double RibMarrowMass = (RibMarrowVol)*RibMarrowDensity;
G4cout << "Mass of Rib 12 Marrow = " << RibMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = Rib12MarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 25;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return Rib12MarrowPhysical;
}

// _____//

// construct MIRD left clavicle

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftClavicle(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set left clavicle volume
    G4VSolid *LeftClavicle_Solid = new G4Torus("LeftClavicle_Solid", // name
        0, // inner radius of cross-sectional volume

```

```

                                0.7883*cm, // outer radius of cross-sectional volume
                                10.*cm,      // radius of toroid
                                298.15*deg,   // start phi
                                0.7*rad);      // delta phi

LeftClavicleLogical = new G4LogicalVolume(LeftClavicle_Solid,          // solid volume
                                           g4bone,                     // material
                                           "LeftClavicle_Logical");      // name

// set bone color
LeftClavicleLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftClavicleLogical->SetUserLimits(fStepLimit);

LeftClaviclePhysical = new G4PVPlacement(0,                          // rotation matrix
                                           G4ThreeVector(0, 0, 33.25*cm), // translation vector wrt trunk
                                           LeftClavicleLogical,          // logical volume
                                           "LeftClavicle_Physical",       // name
                                           pMotherLogical,                // mother volume
                                           false,                         // unused boolean
                                           0);                             // copy number

// calculate mass
G4double LeftClavicleVol = LeftClavicleLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Clavicle = " << LeftClavicleVol/cm3 << " cm^3" << G4endl;
G4String LeftClavicleMat = LeftClavicleLogical->GetMaterial()->GetName();
G4cout << "Material of Left Clavicle = " << LeftClavicleMat << G4endl;
G4double LeftClavicleDensity = LeftClavicleLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftClavicleDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftClavicleMass = (LeftClavicleVol)*LeftClavicleDensity;
G4cout << "Mass of Left Clavicle = " << LeftClavicleMass/gram << " g" << G4endl;

return LeftClaviclePhysical;
}

// _____//

// construct left clavicle active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructLeftClavicleMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set left clavicle active marrow volume
    G4VSolid *LeftClavicleMarrow_Solid = new G4Torus("LeftClavicleMarrow_Solid", // name
                                                       0, // inner radius of cross-sectional volume
                                                       0.5*cm, // outer radius of cross-sectional volume
                                                       10.*cm, // radius of toroid
                                                       298.16*deg, // start phi
                                                       0.69*rad); // delta phi

    LeftClavicleMarrowLogical = new G4LogicalVolume(LeftClavicleMarrow_Solid, // solid volume
                                                       red_marrow, // material
                                                       "LeftClavicleMarrow_Logical"); // name

    // set red marrow color
    LeftClavicleMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftClavicleMarrowLogical->SetUserLimits(fStepLimit);
}

```

```

LeftClavicleMarrowPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector wrt left clavicle
LeftClavicleMarrowLogical, // logical volume
"LeftClavicleMarrow_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double LeftClavicleMarrowVol = LeftClavicleMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Clavicle Marrow = " << LeftClavicleMarrowVol/cm3 << " cm^3"
<< G4endl;
G4String LeftClavicleMarrowMat = LeftClavicleMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Left Clavicle Marrow = " << LeftClavicleMarrowMat << G4endl;
G4double LeftClavicleMarrowDensity = LeftClavicleMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftClavicleMarrowDensity*cm3/g << " g/cm^3"
<< G4endl;
G4double LeftClavicleMarrowMass = (LeftClavicleMarrowVol)*LeftClavicleMarrowDensity;
G4cout << "Mass of Left Clavicle Marrow = " << LeftClavicleMarrowMass/gram << " g"
<< G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = LeftClavicleMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 26;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftClavicleMarrowPhysical;
}

// _____//

// construct MIRD right clavicle

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightClavicle(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set right clavicle volume
    G4VSolid *RightClavicle_Solid = new G4Torus("RightClavicle_Solid", // name
0, // inner radius of cross-sectional volume
0.7883*cm, // outer radius of cross-sectional volume
10.*cm, // radius of toroid
201.75*deg, // starting phi
0.7*rad); // delta phi

    RightClavicleLogical = new G4LogicalVolume(RightClavicle_Solid, // solid volume
g4bone, // material
"RightClavicle_Logical"); // name

    // set bone color
    RightClavicleLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightClavicleLogical->SetUserLimits(fStepLimit);

```

```

RightClaviclePhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0, 0, 33.25*cm), // translation vector wrt trunk
RightClavicleLogical, // logical volume
"RightClavicle_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double RightClavicleVol = RightClavicleLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Clavicle = " << RightClavicleVol/cm3 << " cm^3" << G4endl;
G4String RightClavicleMat = RightClavicleLogical->GetMaterial()->GetName();
G4cout << "Material of Right Clavicle = " << RightClavicleMat << G4endl;
G4double RightClavicleDensity = RightClavicleLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightClavicleDensity*cm3/g << " g/cm^3" << G4endl;
G4double RightClavicleMass = (RightClavicleVol)*RightClavicleDensity;
G4cout << "Mass of Right Clavicle = " << RightClavicleMass/gram << " g" << G4endl;

return RightClaviclePhysical;
}

// _____//

// construct right clavicle active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRightClavicleMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set right clavicle active marrow volume
    G4VSolid *RightClavicleMarrow_Solid = new G4Torus("RightClavicleMarrow_Solid", // name
0, // inner radius of cross-sectional volume
0.5*cm, // outer radius of cross-sectional volume
10.*cm, // radius of toroid
201.76*deg, // starting phi
0.69*rad); // delta phi

    RightClavicleMarrowLogical = new G4LogicalVolume(RightClavicleMarrow_Solid, // solid volume
red_marrow, // material
"RightClavicleMarrow_Logical"); // name

    // set red marrow color
    RightClavicleMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightClavicleMarrowLogical->SetUserLimits(fStepLimit);

    RightClavicleMarrowPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector wrt right clavicle
RightClavicleMarrowLogical, // logical volume
"RightClavicleMarrow_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

    // calculate mass
    G4double RightClavicleMarrowVol = RightClavicleMarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Clavicle Marrow = " << RightClavicleMarrowVol/cm3 << " cm^3"
<< G4endl;
    G4String RightClavicleMarrowMat = RightClavicleMarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Clavicle Marrow = " << RightClavicleMarrowMat << G4endl;
    G4double RightClavicleMarrowDensity = RightClavicleMarrowLogical->GetMaterial()->
GetDensity();
    G4cout << "Density of Material = " << RightClavicleMarrowDensity*cm3/g << " g/cm^3"

```

```

<< G4endl;
G4double RightClavicleMarrowMass = (RightClavicleMarrowVol)*RightClavicleMarrowDensity;
G4cout << "Mass of Right Clavicle Marrow = " << RightClavicleMarrowMass/gram << " g"
<< G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = RightClavicleMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 27;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightClavicleMarrowPhysical;
}

// _____//

// construct MIRD left scapula
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftScapula(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set left scapula volume
    G4EllipticalTube *OuterScapula = new G4EllipticalTube("OuterScapula", // name
                                                            19.*cm, // ax
                                                            9.8*cm, // by
                                                            8.2*cm); // dz

    G4EllipticalTube *InnerScapula = new G4EllipticalTube("InnerScapula", // name
                                                            17.*cm, // ax
                                                            9.8*cm, // by
                                                            8.7*cm); // dz

    G4Box *SubtrScapula = new G4Box("SubtrScapula", // name
                                     19.*cm, // half x
                                     19.*cm, // half y
                                     19.*cm); // half z

    G4RotationMatrix *rm1 = new G4RotationMatrix();
    rm1->rotateZ(-14.*deg);
    G4SubtractionSolid *LeftScapula1_Solid = new G4SubtractionSolid("LeftScapula1_Solid", // name
                                                                    OuterScapula, // larger volume to subtract from
                                                                    SubtrScapula, // smaller volume to subtract
                                                                    rm1, // rotation matrix
                                                                    G4ThreeVector(4.598*cm,-18.43*cm,0)); // relative position

    G4RotationMatrix *rm3 = new G4RotationMatrix();
    rm3->rotateZ(-38.6598*deg);
    G4SubtractionSolid *LeftScapulaBone_Solid = new G4SubtractionSolid("LeftScapulaBone_Solid",
                                                                    LeftScapula1_Solid, // larger volume to subtract from
                                                                    SubtrScapula, // smaller volume to subtract
                                                                    rm3, // rotation matrix
                                                                    G4ThreeVector(-11.8693*cm,14.8365*cm,0)); // relative position

    G4SubtractionSolid *LeftScapula_Solid = new G4SubtractionSolid("LeftScapula_Solid", // name
                                                                    LeftScapulaBone_Solid, // larger volume to subtract from
                                                                    InnerScapula); // smaller volume to subtract

```

```

LeftScapulaLogical = new G4LogicalVolume(LeftScapula_Solid,           // solid volume
                                         g4bone,                     // material
                                         "LeftScapula_Logical");      // name

// set bone color
LeftScapulaLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftScapulaLogical->SetUserLimits(fStepLimit);

LeftScapulaPhysical = new G4PVPlacement(0,                          // rotation matrix
                                         G4ThreeVector(0, 0, 24.1*cm), // translation vector wrt trunk
                                         LeftScapulaLogical,          // logical volume
                                         "LeftScapula_Physical",      // name
                                         pMotherLogical,              // mother volume
                                         false,                       // unused boolean
                                         0);                          // copy number

// calculate mass
G4double LeftScapulaVol = LeftScapulaLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Scapula = " << LeftScapulaVol/cm3 << " cm^3" << G4endl;
G4String LeftScapulaMat = LeftScapulaLogical->GetMaterial()->GetName();
G4cout << "Material of Left Scapula = " << LeftScapulaMat << G4endl;
G4double LeftScapulaDensity = LeftScapulaLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftScapulaDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftScapulaMass = (LeftScapulaVol)*LeftScapulaDensity;
G4cout << "Mass of Left Scapula = " << LeftScapulaMass/gram << " g" << G4endl;

return LeftScapulaPhysical;
}

// _____//

// construct left scapula active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructLeftScapulaMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set left scapula active marrow volume
    G4EllipticalTube *OuterScapulaMarrow = new G4EllipticalTube("OuterScapulaMarrow", // name
                                                                18.2*cm, // ax
                                                                9.8*cm, // by
                                                                8.199*cm); // dz

    G4EllipticalTube *InnerScapulaMarrow = new G4EllipticalTube("InnerScapulaMarrow", // name
                                                                17.8*cm, // ax
                                                                9.8*cm, // by
                                                                8.7*cm); // dz

    G4Box *SubtrScapulaMarrow = new G4Box("SubtrScapulaMarrow", // name
                                          19.*cm, // half x
                                          19.*cm, // half y
                                          19.*cm); // half z

    G4RotationMatrix *rm1 = new G4RotationMatrix();
    rm1->rotateZ(-14.*deg);
    G4SubtractionSolid *LeftScapulaMarrow1_Solid = new
    G4SubtractionSolid("LeftScapulaMarrow1_Solid", // name
                      OuterScapulaMarrow, // larger volume to subtract from
                      SubtrScapulaMarrow, // smaller volume to subtract
                      rm1, // rotation matrix
                      G4ThreeVector(4.598*cm,-18.43*cm,0)); // relative position

```

```

G4RotationMatrix *rm3 = new G4RotationMatrix();
rm3->rotateZ(-38.6598*deg);
G4SubtractionSolid *LeftScapulaBoneMarrow_Solid = new
G4SubtractionSolid("LeftScapulaBoneMarrow_Solid",      // name
                  LeftScapulaMarrow1_Solid,           // larger volume to subtract from
                  SubtrScapulaMarrow,                 // smaller volume to subtract
                  rm3,                                 // rotation matrix
                  G4ThreeVector(-11.8693*cm,14.5*cm,0)); // relative position

G4SubtractionSolid *LeftScapulaMarrow_Solid = new
G4SubtractionSolid("LeftScapulaMarrow_Solid",          // name
                  LeftScapulaBoneMarrow_Solid,        // larger volume to subtract from
                  InnerScapulaMarrow);                // smaller volume to subtract

LeftScapulaMarrowLogical = new G4LogicalVolume(LeftScapulaMarrow_Solid, // solid volume
                                              red_marrow,                // material
                                              "LeftScapulaMarrow_Logical"); // name

// set red marrow color
LeftScapulaMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftScapulaMarrowLogical->SetUserLimits(fStepLimit);

LeftScapulaMarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                              G4ThreeVector(), // translation vector wrt left scapula
                                              LeftScapulaMarrowLogical, // logical volume
                                              "LeftScapulaMarrow_Physical", // name
                                              pMotherLogical, // mother volume
                                              false, // unused boolean
                                              0); // copy number

// calculate mass
G4double LeftScapulaMarrowVol = LeftScapulaMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Scapula Marrow = " << LeftScapulaMarrowVol/cm3 << " cm^3" <<
G4endl;
G4String LeftScapulaMarrowMat = LeftScapulaMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Left Scapula Marrow = " << LeftScapulaMarrowMat << G4endl;
G4double LeftScapulaMarrowDensity = LeftScapulaMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftScapulaMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftScapulaMarrowMass = (LeftScapulaMarrowVol)*LeftScapulaMarrowDensity;
G4cout << "Mass of Left Scapula Marrow = " << LeftScapulaMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = LeftScapulaMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 28;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftScapulaMarrowPhysical;
}

// _____//

// construct MIRD right scapula

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightScapula(G4LogicalVolume
*pMotherLogical)
{

```

```

// get materials
G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

// set right scapula volume
G4EllipticalTube *OuterScapula = new G4EllipticalTube("OuterScapula", // name
    19.*cm, // ax
    9.8*cm, // by
    8.2*cm); // dz

G4EllipticalTube *InnerScapula = new G4EllipticalTube("InnerScapula", // name
    17.*cm, // ax
    9.8*cm, // by
    8.7*cm); // dz

G4Box *SubtrScapula = new G4Box("SubtrScapula", // name
    19.*cm, // half x
    19.*cm, // half y
    19.*cm); // half z

G4RotationMatrix *rm2 = new G4RotationMatrix();
rm2->rotateZ(14.*deg);

G4SubtractionSolid *RightScapula1_Solid = new G4SubtractionSolid("RightScapula1_Solid",
    OuterScapula, // larger volume to subtract from
    SubtrScapula, // smaller volume to subtract
    rm2, // rotation matrix
    G4ThreeVector(-4.598*cm,-18.43*cm,0)); // relative position

G4RotationMatrix *rm4 = new G4RotationMatrix();
rm4->rotateZ(38.6598*deg);

G4SubtractionSolid *RightScapulaBone_Solid = new
G4SubtractionSolid("RightScapulaBone_Solid", // name
    RightScapula1_Solid, // larger volume to subtract from
    SubtrScapula, // smaller volume to subtract
    rm4, // rotation matrix
    G4ThreeVector(11.8693*cm,14.8365*cm,0)); // relative position

G4SubtractionSolid *RightScapula_Solid = new G4SubtractionSolid("RightScapula_Solid", // name
    RightScapulaBone_Solid, // larger volume to subtract from
    InnerScapula); // smaller volume to subtract

RightScapulaLogical = new G4LogicalVolume(RightScapula_Solid, // solid volume
    g4bone, // material
    "RightScapula_Logical"); // name

// set bone color
RightScapulaLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
RightScapulaLogical->SetUserLimits(fStepLimit);

RightScapulaPhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(0, 0, 24.1*cm), // translation vector wrt trunk
    RightScapulaLogical, // logical volume
    "RightScapula_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

// calculate mass
G4double RightScapulaVol = RightScapulaLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Scapula = " << RightScapulaVol/cm3 << " cm^3" << G4endl;
G4String RightScapulaMat = RightScapulaLogical->GetMaterial()->GetName();
G4cout << "Material of Right Scapula = " << RightScapulaMat << G4endl;
G4double RightScapulaDensity = RightScapulaLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightScapulaDensity*cm3/g << " g/cm^3" << G4endl;

```

```

G4double RightScapulaMass = (RightScapulaVol)*RightScapulaDensity;
G4cout << "Mass of Right Scapula = " << RightScapulaMass/gram << " g" << G4endl;

return RightScapulaPhysical;
}

// _____//

// construct right scapula active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRightScapulaMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set right scapula active marrow volume
    G4EllipticalTube *OuterScapulaMarrow = new G4EllipticalTube("OuterScapulaMarrow", // name
                                                                18.2*cm, // ax
                                                                9.8*cm, // by
                                                                8.199*cm); // dz

    G4EllipticalTube *InnerScapulaMarrow = new G4EllipticalTube("InnerScapulaMarrow", // name
                                                                17.8*cm, // ax
                                                                9.8*cm, // by
                                                                8.7*cm); // dz

    G4Box *SubtrScapulaMarrow = new G4Box("SubtrScapulaMarrow", // name
                                           19.*cm, // half x
                                           19.*cm, // half y
                                           19.*cm); // half z

    G4RotationMatrix *rm2 = new G4RotationMatrix();
    rm2->rotateZ(14.*deg);

    G4SubtractionSolid *RightScapulaMarrow1_Solid = new
    G4SubtractionSolid("RightScapulaMarrow1_Solid", // name
                      OuterScapulaMarrow, // larger volume to subtract from
                      SubtrScapulaMarrow, // smaller volume to subtract
                      rm2, // rotation matrix
                      G4ThreeVector(-4.598*cm,-18.43*cm,0)); // relative position

    G4RotationMatrix *rm4 = new G4RotationMatrix();
    rm4->rotateZ(38.6598*deg);

    G4SubtractionSolid *RightScapulaBoneMarrow_Solid = new
    G4SubtractionSolid("RightScapulaBoneMarrow_Solid", // name
                      RightScapulaMarrow1_Solid, // larger volume to subtract from
                      SubtrScapulaMarrow, // smaller volume to subtract
                      rm4, // rotation matrix
                      G4ThreeVector(11.8693*cm, 14.5*cm, 0)); // relative position

    G4SubtractionSolid *RightScapulaMarrow_Solid = new
    G4SubtractionSolid("RightScapulaMarrow_Solid", // name
                      RightScapulaBoneMarrow_Solid, // larger volume to subtract from
                      InnerScapulaMarrow); // smaller volume to subtract

    RightScapulaMarrowLogical = new G4LogicalVolume(RightScapulaMarrow_Solid, // solid volume
                                                    red_marrow, // material
                                                    "RightScapulaMarrow_Logical");// name

    // set red marrow color
    RightScapulaMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);

```

```

RightScapulaMarrowLogical->SetUserLimits(fStepLimit);

RightScapulaMarrowPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector wrt right scapula
RightScapulaMarrowLogical, // logical volume
"RightScapulaMarrow_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double RightScapulaMarrowVol = RightScapulaMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Scapula Marrow = " << RightScapulaMarrowVol/cm3 << " cm^3" <<
G4endl;
G4String RightScapulaMarrowMat = RightScapulaMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Right Scapula Marrow = " << RightScapulaMarrowMat << G4endl;
G4double RightScapulaMarrowDensity = RightScapulaMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightScapulaMarrowDensity*cm3/g << " g/cm^3" <<
G4endl;
G4double RightScapulaMarrowMass = (RightScapulaMarrowVol)*RightScapulaMarrowDensity;
G4cout << "Mass of Right Scapula Marrow = " << RightScapulaMarrowMass/gram << " g" <<
G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = RightScapulaMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 29;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightScapulaMarrowPhysical;
}

// _____//

// construct MIRD small intestine

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDSmallIntestine(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set small intestine volume
    G4Box *SmallIntestineBox = new G4Box("SmallIntestineBox", // name
11.*cm, // dx
3.53*cm, // dy
5.*cm); // dz

    G4Tubs *SmallIntestineTubs = new G4Tubs("SmallIntestineTubs", // name
0., // rmin
11.*cm, // rmax
5.*cm, // dz
0, // start phi
360.*deg); // delta phi

    G4IntersectionSolid *FilledSmallIntestine1 = new
G4IntersectionSolid("FilledSmallIntestine1", // name
SmallIntestineTubs, // 1st solid to intersect
SmallIntestineBox, // 2nd solid to intersect
0, // rotation matrix

```

```

        G4ThreeVector(0,-1.33*cm,0)); // relative position

G4IntersectionSolid *FilledSmallIntestine = new G4IntersectionSolid("FilledSmallIntestine",
        FilledSmallIntestine1, // 1st solid to intersect
        SmallIntestineTubs, // 2nd solid to intersect
        0, // rotation matrix
        G4ThreeVector(0,0.8*cm,0)); // relative position

G4VSolid *AscendingColon = new G4EllipticalTube("AscendingColon", // name
        2.5*cm, // ax
        2.5*cm, // by
        4.775*cm); // dz

G4VSolid *TransverseColon = new G4EllipticalTube("TransverseColon", // name
        2.5*cm, // ax
        1.5*cm, // by
        10.5*cm); // dz

G4RotationMatrix *rm1 = new G4RotationMatrix();
rm1->rotateX(90.*deg);
rm1->rotateY(90.*deg);
G4UnionSolid *UpperLargeIntestine = new G4UnionSolid("UpperLargeIntestine", // name
        AscendingColon, // 1st volume to intersect
        TransverseColon, // 2nd volume to intersect
        rm1, // rotation matrix
        G4ThreeVector(-8.*cm, 0, 6.275*cm)); // relative position

G4EllipticalTube *DescendingColon = new G4EllipticalTube("DescendingColon", // name
        1.88*cm, // ax
        2.13*cm, // by
        7.64*cm); // dz

G4UnionSolid *UpperLowerLargeIntestine = new G4UnionSolid("UpperLowerLargeIntestine", // name
        UpperLargeIntestine, // 1st volume to intersect
        DescendingColon, // 2nd volume to intersect
        0, // rotation matrix
        G4ThreeVector(-16.72*cm,0,-2.865*cm)); // relative position

G4SubtractionSolid *SmallIntestine_Solid = new G4SubtractionSolid("SmallIntestine", // name
        FilledSmallIntestine, // 1st volume to intersect
        UpperLowerLargeIntestine, // 2nd volume to intersect
        0, // rotation matrix
        G4ThreeVector(8.*cm,-0.3*cm,-2.775*cm)); // relative position

SmallIntestineLogical = new G4LogicalVolume(SmallIntestine_Solid, // solid volume
        soft_tissue, // material
        "SmallIntestine_Logical"); // name

// set small intestine color
SmallIntestineLogical->SetVisAttributes(G4Colour(0.8039, 0.6549, 0.5569));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
SmallIntestineLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm2 = new G4RotationMatrix();
rm2->rotateX(180.*deg);
rm2->rotateY(180.*deg);
SmallIntestinePhysical = new G4PVPlacement(rm2, // rotation matrix
        G4ThreeVector(0,-2.66*cm,-13.*cm), // translation vector wrt trunk
        SmallIntestineLogical, // logical volume
        "SmallIntestine_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

// calculate mass
G4double SmallIntestineVol = SmallIntestineLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Small Intestine = " << SmallIntestineVol/cm3 << " cm^3" << G4endl;

```

```

G4String SmallIntestineMat = SmallIntestineLogical->GetMaterial()->GetName();
G4cout << "Material of Small Intestine = " << SmallIntestineMat << G4endl;
G4double SmallIntestineDensity = SmallIntestineLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << SmallIntestineDensity*cm3/g << " g/cm^3" << G4endl;
G4double SmallIntestineMass = (SmallIntestineVol)*SmallIntestineDensity;
G4cout << "Mass of Small Intestine = " << SmallIntestineMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = SmallIntestineLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 30;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return SmallIntestinePhysical;
}

// _____//

// construct MIRD lower large intestine

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLowerLargeIntestine(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set lower large intestine volume
    G4EllipticalTube *DescendingColon2 = new G4EllipticalTube("DescendingColon", // name
                                                                1.88*cm, // dx
                                                                2.13*cm, // dy
                                                                7.64*cm); // dz

    G4Torus *SigmoidColonUp = new G4Torus("SigmoidColonUp", // name
                                           0, // inner radius
                                           1.88*cm, // outer radius
                                           5.72*cm, // radius of toroid
                                           0, // start phi
                                           90.*deg); // delta phi

    G4VSolid *SigmoidColonDown = new G4Torus("SigmoidColonDown", // name
                                              0, // inner radius
                                              1.88*cm, // outer radius
                                              3.*cm, // radius of toroid
                                              0, // start phi
                                              90.*deg); // delta phi

    G4RotationMatrix *rm1 = new G4RotationMatrix();
    rm1->rotateY(180.*deg);
    rm1->rotateZ(90.*deg);
    G4UnionSolid *SigmoidColon = new G4UnionSolid("SigmoidColon", // name
                                                  SigmoidColonUp, // 1st volume to union
                                                  SigmoidColonDown, // 2nd volume to union
                                                  rm1, // rotation matrix
                                                  G4ThreeVector(0,8.72*cm,0)); // relative position

    G4RotationMatrix *rm2 = new G4RotationMatrix();
    rm2->rotateX(90.*deg);
    G4UnionSolid *LowerLargeIntestine_Solid = new G4UnionSolid("LowerLargeIntestine_Solid",
                                                                DescendingColon2, // 1st volume to union
                                                                SigmoidColon, // 2nd volume to union

```

```

        rm2, // rotation matrix
        G4ThreeVector(-5.72*cm,0,-7.64*cm)); // relative position

LowerLargeIntestineLogical = new G4LogicalVolume(LowerLargeIntestine_Solid, // solid volume
        soft_tissue, // material
        "LowerLargeIntestine_Logical"); // name

// set large intestine color
LowerLargeIntestineLogical->SetVisAttributes(G4Colour(0.5412, 0.3725, 0.2902));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LowerLargeIntestineLogical->SetUserLimits(fStepLimit);

LowerLargeIntestinePhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(8.72*cm,-2.36*cm,-18.64*cm), // translation vector wrt trunk
        LowerLargeIntestineLogical, // logical volume
        "LowerLargeIntestine_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

// calculate mass
G4double LowerLargeIntestineVol = LowerLargeIntestineLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Lower Large Intestine = " << LowerLargeIntestineVol/cm3 << " cm^3" <<
G4endl;
G4String LowerLargeIntestineMat = LowerLargeIntestineLogical->GetMaterial()->GetName();
G4cout << "Material of LowerLargeIntestine = " << LowerLargeIntestineMat << G4endl;
G4double LowerLargeIntestineDensity = LowerLargeIntestineLogical->GetMaterial()-
>GetDensity();
G4cout << "Density of Material = " << LowerLargeIntestineDensity*cm3/g << " g/cm^3" <<
G4endl;
G4double LowerLargeIntestineMass = (LowerLargeIntestineVol)*LowerLargeIntestineDensity;
G4cout << "Mass of Lower Large Intestine = " << LowerLargeIntestineMass/gram << " g" <<
G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LowerLargeIntestineLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 31;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LowerLargeIntestinePhysical;
}

// _____//

// construct MIRD upper large intestine

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDUpperLargeIntestine(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set upper large intestine volume
    G4VSolid *AscendingColon2 = new G4EllipticalTube("AscendingColon_Solid", // name
        2.5*cm, // dx
        2.5*cm, // dy

```

```

                                4.775*cm);                                // dz

G4VSolid *TransverseColon2 = new G4EllipticalTube("TransverseColon_Solid", // name
                                2.5*cm,                                // dx
                                1.5*cm,                                // dy
                                10.5*cm);                                // dz

G4RotationMatrix *rm = new G4RotationMatrix();
rm->rotateX(90.*deg);
rm->rotateY(90.*deg);
G4UnionSolid *UpperLargeIntestine_Solid = new G4UnionSolid("UpperLargeIntestine_Solid",
    AscendingColon2, // 1st volume to union
    TransverseColon2, // 2nd volume to union
    rm, // rotation matrix
    G4ThreeVector(8.*cm,0,6.275*cm)); // relative position

UpperLargeIntestineLogical = new G4LogicalVolume(UpperLargeIntestine_Solid, // solid volume
    soft_tissue, // material
    "UpperLargeIntestine_Logical");// name

// set large intestine color
UpperLargeIntestineLogical->SetVisAttributes(G4Colour(0.5412, 0.3725, 0.2902));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
UpperLargeIntestineLogical->SetUserLimits(fStepLimit);

UpperLargeIntestinePhysical = new G4PVPlacement(0, // rotation matrix
    G4ThreeVector(-8.*cm,-2.36*cm,-15.775*cm), // translation vector wrt trunk
    UpperLargeIntestineLogical, // logical volume
    "UpperLargeIntestine_Physical", // name
    pMotherLogical, // mother volume
    false, // unused boolean
    0); // copy number

// calculate mass
G4double UpperLargeIntestineVol = UpperLargeIntestineLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Upper Large Intestine = " << UpperLargeIntestineVol/cm3 << " cm^3" <<
G4endl;
G4String UpperLargeIntestineMat = UpperLargeIntestineLogical->GetMaterial()->GetName();
G4cout << "Material of Upper Large Intestine = " << UpperLargeIntestineMat << G4endl;
G4double UpperLargeIntestineDensity = UpperLargeIntestineLogical->GetMaterial()-
>GetDensity();
G4cout << "Density of Material = " << UpperLargeIntestineDensity*cm3/g << " g/cm^3" <<
G4endl;
G4double UpperLargeIntestineMass = (UpperLargeIntestineVol)*UpperLargeIntestineDensity;
G4cout << "Mass of Upper Large Intestine = " << UpperLargeIntestineMass/gram << " g" <<
G4endl;

//store organ info in maps
G4LogicalVolume* currentLogical = UpperLargeIntestineLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 32;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return UpperLargeIntestinePhysical;
}

//_____//

```

```

// construct MIRD liver

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLiver(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set liver volume
    G4EllipticalTube *LiverLobe = new G4EllipticalTube("LiverLobe",    // name
                                                         14.9*cm,        // dx
                                                         7.84*cm,        // dy
                                                         7.21*cm);        // dz

    G4Box *SubtrLiver = new G4Box("SubtrLiver",    // name
                                   10.*cm,        // half
                                   25.*cm,        // half
                                   25.*cm);        // half

    G4RotationMatrix* rm = new G4RotationMatrix();
    rm-> rotateY(32.*deg);
    rm-> rotateZ(40.9*deg);
    G4SubtractionSolid *Liver_Solid = new G4SubtractionSolid("Liver_Solid", // name
                                                             LiverLobe,        // larger volume to subtract from
                                                             SubtrLiver,        // smaller volume to subtract
                                                             rm,                // rotation matrix
                                                             G4ThreeVector(10.*cm,0,0)); // relative position

    LiverLogical = new G4LogicalVolume(Liver_Solid,    // solid volume
                                       soft_tissue,    // material
                                       "Liver_Logical"); // name

    // set liver color
    LiverLogical->SetVisAttributes(G4Colour(0.867, 0.5098, 0.396));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LiverLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm2 = new G4RotationMatrix();
    rm2-> rotateX(180.*deg);
    LiverPhysical = new G4PVPlacement(rm2,            // rotation matrix
                                       G4ThreeVector(), // translation vector wrt trunk
                                       LiverLogical,    // logical volume
                                       "Liver_Physical", // name
                                       pMotherLogical,  // mother volume
                                       false,           // unused boolean
                                       0);              // copy number

    // calculate mass
    G4double LiverVol = LiverLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Liver = " << LiverVol/cm3 << " cm^3" << G4endl;
    G4String LiverMat = LiverLogical->GetMaterial()->GetName();
    G4cout << "Material of Liver = " << LiverMat << G4endl;
    G4double LiverDensity = LiverLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << LiverDensity*cm3/g << " g/cm^3" << G4endl;
    G4double LiverMass = (LiverVol)*LiverDensity;
    G4cout << "Mass of Liver = " << LiverMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = LiverLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 33;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
}

```

```

    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return LiverPhysical;
}

// _____//

// construct MIRD stomach

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDStomach(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set stomach volume
    G4VSolid *Stomach_Solid = new G4Ellipsoid("Stomach_Solid", // name
                                              4.*cm,             // ax
                                              3.*cm,             // by
                                              8.*cm);            // cz

    StomachLogical = new G4LogicalVolume(Stomach_Solid,        // solid volume
                                         soft_tissue,          // material
                                         "Stomach_Logical");    // name

    // set stomach color
    StomachLogical->SetVisAttributes(G4Colour(0.847, 0.518, 0.412));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    StomachLogical->SetUserLimits(fStepLimit);

    StomachPhysical = new G4PVPlacement(0,                    // rotation matrix
                                         G4ThreeVector(8.*cm,-4.*cm,0), // translation vector wrt trunk
                                         StomachLogical,        // logical volume
                                         "Stomach_Physical",    // name
                                         pMotherLogical,        // mother volume
                                         false,                 // unused boolean
                                         0);                     // copy number

    // calculate mass
    G4double StomachVol = StomachLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Stomach = " << StomachVol/cm3 << " cm^3" << G4endl;
    G4String StomachMat = StomachLogical->GetMaterial()->GetName();
    G4cout << "Material of Stomach = " << StomachMat << G4endl;
    G4double StomachDensity = StomachLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << StomachDensity*cm3/g << " g/cm^3" << G4endl;
    G4double StomachMass = (StomachVol)*StomachDensity;
    G4cout << "Mass of Stomach = " << StomachMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = StomachLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 34;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return StomachPhysical;
}

```

```

}

// _____//

// construct MIRD pancreas

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDPancreas(G4LogicalVolume *pMotherLogical)
{

    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set pancreas volume
    G4Ellipsoid *Pancreas1 = new G4Ellipsoid("Pancreas1", // name
        3.*cm, // ax
        1.*cm, // by
        15.*cm, // cz
        -15.9*cm, // zcut1
        0); // zcut2

    G4Box *SubtrPancreas = new G4Box("SubtrPancreas", // name
        3.*cm, // dx
        1.*cm, // dy
        6.*cm); // dz

    G4SubtractionSolid *Pancreas_Solid = new G4SubtractionSolid("Pancreas_Solid", // name
        Pancreas1, // larger volume to subtract from
        SubtrPancreas, // smaller volume to subtract
        0, // rotation matrix
        G4ThreeVector(-3.*cm,0,-9.*cm)); // relative position

    PancreasLogical = new G4LogicalVolume(Pancreas_Solid, // solid volume
        soft_tissue, // material
        "Pancreas_Logical"); // name

    // set pancreas color
    PancreasLogical->SetVisAttributes(G4Colour(0.9765, 0.7059, 0.4353));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    PancreasLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateY(90.*deg);
    PancreasPhysical = new G4PVPlacement(rm, // rotation matrix
        G4ThreeVector(0,0,2.*cm), // translation vector wrt trunk
        PancreasLogical, // logical volume
        "Pancreas_Physical", // name
        pMotherLogical, // mother volume
        false, // unused boolean
        0); // copy number

    // calculate mass
    G4double PancreasVol = PancreasLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Pancreas = " << PancreasVol/cm3 << " cm^3" << G4endl;
    G4String PancreasMat = PancreasLogical->GetMaterial()->GetName();
    G4cout << "Material of Pancreas = " << PancreasMat << G4endl;
    G4double PancreasDensity = PancreasLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << PancreasDensity*cm3/g << " g/cm^3" << G4endl;
    G4double PancreasMass = (PancreasVol)*PancreasDensity;
    G4cout << "Mass of Pancreas = " << PancreasMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = PancreasLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 35;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
}

```

```

    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return PancreasPhysical;
}

// _____//

// construct MIRD spleen

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDSpleen(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set spleen volume
    G4Ellipsoid *Spleen_Solid = new G4Ellipsoid("Spleen_Solid", // name
                                                3.5*cm,           // ax
                                                2.*cm,           // by
                                                6.*cm);          // cz

    SpleenLogical = new G4LogicalVolume(Spleen_Solid,           // solid volume
                                        soft_tissue,             // material
                                        "Spleen_Logical");        // name

    // set spleen color
    SpleenLogical->SetVisAttributes(G4Colour(0.6157, 0.4235, 0.6353));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    SpleenLogical->SetUserLimits(fStepLimit);

    SpleenPhysical = new G4PVPlacement(0, // rotation matrix
                                       G4ThreeVector(11.*cm, 3.*cm, 2.*cm), // translation vector wrt trunk
                                       SpleenLogical, // logical volume
                                       "Spleen_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

    // calculate mass
    G4double SpleenVol = SpleenLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Spleen = " << SpleenVol/cm3 << " cm^3" << G4endl;
    G4String SpleenMat = SpleenLogical->GetMaterial()->GetName();
    G4cout << "Material of Spleen = " << SpleenMat << G4endl;
    G4double SpleenDensity = SpleenLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << SpleenDensity*cm3/g << " g/cm^3" << G4endl;
    G4double SpleenMass = (SpleenVol)*SpleenDensity;
    G4cout << "Mass of Spleen = " << SpleenMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = SpleenLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 36;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
}

```

```

    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return SpleenPhysical;
}

// _____//

// construct MIRD bladder

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDBladder(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set bladder volume
    G4Ellipsoid *Bladder_Out = new G4Ellipsoid("Bladder_Out",    // name
                                                4.958*cm,          // ax
                                                3.458*cm,          // by
                                                3.548*cm);         // cz

    G4Ellipsoid *Bladder_In = new G4Ellipsoid("Bladder_In",      // name
                                                4.706*cm,          // ax
                                                3.206*cm,          // by
                                                3.206*cm);         // cz

    G4SubtractionSolid *Bladder_Solid = new G4SubtractionSolid("Bladder_Solid", // name
                                                                Bladder_Out, // larger volume to subtract from
                                                                Bladder_In); // smaller volume to subtract

    BladderLogical = new G4LogicalVolume(Bladder_Solid,           // solid volume
                                         soft_tissue,             // material
                                         "Bladder_Logical");       // name

    // set bladder color
    BladderLogical->SetVisAttributes(G4Colour(0.8039, 0.7020, 0.4235));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    BladderLogical->SetUserLimits(fStepLimit);

    BladderPhysical = new G4PVPlacement(0,                       // rotation matrix
                                         G4ThreeVector(0, -4.5*cm, -27.*cm), // translation vector wrt trunk
                                         BladderLogical,          // logical volume
                                         "Bladder_Physical",      // name
                                         pMotherLogical,          // mother volume
                                         false,                   // unused boolean
                                         0);                       // copy number

    // calculate mass
    G4double BladderVol = BladderLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Bladder = " << BladderVol/cm3 << " cm^3" << G4endl;
    G4String BladderMat = BladderLogical->GetMaterial()->GetName();
    G4cout << "Material of Bladder = " << BladderMat << G4endl;
    G4double BladderDensity = BladderLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << BladderDensity*cm3/g << " g/cm^3" << G4endl;
    G4double BladderMass = (BladderVol)*BladderDensity;
    G4cout << "Mass of Bladder = " << BladderMass/gram << " g" << G4endl;

    //store organ info in maps
    G4LogicalVolume* currentLogical = BladderLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 37;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
}

```

```

    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return BladderPhysical;
}

// _____//

// construct MIRD thymus

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDThymus(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set thymus volume
    G4VSolid *Thymus_Solid = new G4Ellipsoid("Thymus_Solid", // name
                                             3.*cm,           // ax
                                             0.5*cm,          // by
                                             4.*cm);          // cz

    ThymusLogical = new G4LogicalVolume(Thymus_Solid,        // solid volume
                                         soft_tissue,         // material
                                         "Thymus_Logical");    // name

    // set thymus color
    ThymusLogical->SetVisAttributes(G4Colour(0.2431, 0.6353, 0.447));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    ThymusLogical->SetUserLimits(fStepLimit);

    ThymusPhysical = new G4PVPlacement(0,                    // rotation matrix
                                       G4ThreeVector(2.*cm,-6.*cm,25.5*cm), // translation vector wrt trunk
                                       ThymusLogical,         // logical volume
                                       "Thymus_Physical",      // name
                                       pMotherLogical,         // mother volume
                                       false,                  // unused boolean
                                       0);                      // copy number

    // calculate mass
    G4double ThymusVol = ThymusLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Thymus = " << ThymusVol/cm3 << " cm^3" << G4endl;
    G4String ThymusMat = ThymusLogical->GetMaterial()->GetName();
    G4cout << "Material of Thymus = " << ThymusMat << G4endl;
    G4double ThymusDensity = ThymusLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << ThymusDensity*cm3/g << " g/cm^3" << G4endl;
    G4double ThymusMass = (ThymusVol)*ThymusDensity;
    G4cout << "Mass of Thymus = " << ThymusMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = ThymusLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 38;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return ThymusPhysical;
}

```

```

}

//_____//

// construct MIRD middle lower spine

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDMiddleLowerSpine(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set middle lower spine volume
    G4VSolid *MiddleLowerSpine_Solid = new G4EllipticalTube("MiddleLowerSpine_Solid", // name
                                                            2.*cm, // dx
                                                            2.5*cm, // dy
                                                            24.*cm); // dz

    MiddleLowerSpineLogical = new G4LogicalVolume(MiddleLowerSpine_Solid, // solid volume
                                                  g4bone, // material
                                                  "MiddleLowerSpine_Logical"); // name

    // set bone color
    MiddleLowerSpineLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    MiddleLowerSpineLogical->SetUserLimits(fStepLimit);

    MiddleLowerSpinePhysical = new G4PVPlacement(0, // rotation matrix
                                                  G4ThreeVector(0,5.5*cm,11.*cm), // translation vector wrt trunk
                                                  MiddleLowerSpineLogical, // logical volume
                                                  "MiddleLowerSpine_Physical", // name
                                                  pMotherLogical, // mother volume
                                                  false, // unused boolean
                                                  0); // copy number

    // calculate mass
    G4double MiddleLowerSpineVol = MiddleLowerSpineLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Middle Lower Spine = " << MiddleLowerSpineVol/cm3 << " cm^3" << G4endl;
    G4String MiddleLowerSpineMat = MiddleLowerSpineLogical->GetMaterial()->GetName();
    G4cout << "Material of Middle Lower Spine = " << MiddleLowerSpineMat << G4endl;
    G4double MiddleLowerSpineDensity = MiddleLowerSpineLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << MiddleLowerSpineDensity*cm3/g << " g/cm^3" << G4endl;
    G4double MiddleLowerSpineMass = (MiddleLowerSpineVol)*MiddleLowerSpineDensity;
    G4cout << "Mass of Middle Lower Spine = " << MiddleLowerSpineMass/gram << " g" << G4endl;

    return MiddleLowerSpinePhysical;
}

//_____//

// construct middle lower spine active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructMiddleLowerSpineMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set middle lower spine active marrow volume
    G4VSolid *MiddleLowerSpineMarrow_Solid = new
    G4EllipticalTube("MiddleLowerSpineMarrow_Solid", // name
                    1.7*cm, // dx
                    2.12*cm, // dy

```

```

        23.99*cm); // dz

MiddleLowerSpineMarrowLogical = new G4LogicalVolume(MiddleLowerSpineMarrow_Solid, // solid
                                                    red_marrow, // material
                                                    "MiddleLowerSpineMarrow_Logical"); // name

// set red marrow color
MiddleLowerSpineMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
MiddleLowerSpineMarrowLogical->SetUserLimits(fStepLimit);

MiddleLowerSpineMarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                                    G4ThreeVector(), // translation vector wrt middle lower spine
                                                    MiddleLowerSpineMarrowLogical, // logical volume
                                                    "MiddleLowerSpineMarrow_Physical", // name
                                                    pMotherLogical, // mother volume
                                                    false, // unused boolean
                                                    0); // copy number

// calculate mass
G4double MiddleLowerSpineMarrowVol = MiddleLowerSpineMarrowLogical->GetSolid()-
>GetCubicVolume();
G4cout << "Volume of Middle Lower Spine Marrow = " << MiddleLowerSpineMarrowVol/cm3 << "
cm^3" << G4endl;
G4String MiddleLowerSpineMarrowMat = MiddleLowerSpineMarrowLogical->GetMaterial()-
>GetName();
G4cout << "Material of Middle Lower Spine Marrow = " << MiddleLowerSpineMarrowMat << G4endl;
G4double MiddleLowerSpineMarrowDensity = MiddleLowerSpineMarrowLogical->GetMaterial()-
>GetDensity();
G4cout << "Density of Material = " << MiddleLowerSpineMarrowDensity*cm3/g << " g/cm^3" <<
G4endl;
G4double MiddleLowerSpineMarrowMass =
(MiddleLowerSpineMarrowVol)*MiddleLowerSpineMarrowDensity;
G4cout << "Mass of Middle Lower Spine Marrow = " << MiddleLowerSpineMarrowMass/gram << " g"
<< G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = MiddleLowerSpineMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 39;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return MiddleLowerSpineMarrowPhysical;
}

// _____//

// construct MIRD pelvis

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDPelvis(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set pelvis volume
    G4EllipticalTube *Pelvis_Out = new G4EllipticalTube("Pelvis_Out", // name

```

```

12.*cm,          // dx
12.*cm,          // dy
11.*cm);        // dz

G4EllipticalTube *Pelvis_In = new G4EllipticalTube("Pelvis_In", // name
11.3*cm,         // dx
11.3*cm,         // dy
12.*cm);        // dz

G4Box *SubtrPelvis = new G4Box("SubtrPelvis", // name
14.*cm,          // dx
14.*cm,          // dy
12.*cm);        // dz

G4SubtractionSolid *Pelvis1 = new G4SubtractionSolid("Pelvis1", // name
Pelvis_Out,      // larger volume to subtract from
Pelvis_In,       // smaller volume to subtract
0,               // rotation matrix
G4ThreeVector(0,-0.8*cm,0)); // relative position

G4SubtractionSolid *Pelvis2 = new G4SubtractionSolid("Pelvis2", // name
Pelvis1,         // larger volume to subtract from
SubtrPelvis,     // smaller volume to subtract
0,               // rotation matrix
G4ThreeVector(0,-14.*cm,0)); // relative position

G4SubtractionSolid *Pelvis_Solid = new G4SubtractionSolid("Pelvis_Solid", // name
Pelvis2,         // larger volume to subtract from
SubtrPelvis,     // smaller volume to subtract
0,               // rotation matrix
G4ThreeVector(0,22.*cm,-9.*cm)); // relative position

PelvisLogical = new G4LogicalVolume(Pelvis_Solid, // solid volume
g4bone,     // material
"Pelvis_Logical"); // name

// set bone color
PelvisLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
PelvisLogical->SetUserLimits(fStepLimit);

PelvisPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0, -3.*cm, -24.*cm), // translation vector wrt trunk
PelvisLogical, // logical volume
"Pelvis_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double PelvisVol = PelvisLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Pelvis = " << PelvisVol/cm3 << " cm^3" << G4endl;
G4String PelvisMat = PelvisLogical->GetMaterial()->GetName();
G4cout << "Material of Pelvis = " << PelvisMat << G4endl;
G4double PelvisDensity = PelvisLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << PelvisDensity*cm3/g << " g/cm^3" << G4endl;
G4double PelvisMass = (PelvisVol)*PelvisDensity;
G4cout << "Mass of Pelvis = " << PelvisMass/gram << " g" << G4endl;

return PelvisPhysical;
}

// _____//
// construct pelvis active marrow

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructPelvisMarrow(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set pelvis active marrow volume
    G4EllipticalTube *PelvisMarrow_Out = new G4EllipticalTube("PelvisMarrow_Out", // name
                                                                11.68*cm, // dx
                                                                11.68*cm, // dy
                                                                10.999*cm); // dz

    G4EllipticalTube *PelvisMarrow_In = new G4EllipticalTube("PelvisMarrow_In", // name
                                                             11.62*cm, // dx
                                                             11.62*cm, // dy
                                                             12.*cm); // dz

    G4Box *SubtrPelvisMarrow = new G4Box("SubtrPelvisMarrow", // name
                                          14.*cm, // dx
                                          14.*cm, // dy
                                          12.*cm); // dz

    G4SubtractionSolid *PelvisMarrow1 = new G4SubtractionSolid("PelvisMarrow1", // name
                                                                PelvisMarrow_Out, // larger volume to subtract from
                                                                PelvisMarrow_In, // smaller volume to subtract
                                                                0, // rotation matrix
                                                                G4ThreeVector(0,-0.8*cm,0)); // relative position

    G4SubtractionSolid *PelvisMarrow2 = new G4SubtractionSolid("PelvisMarrow2", // name
                                                                PelvisMarrow1, // larger volume to subtract from
                                                                SubtrPelvisMarrow, // smaller volume to subtract
                                                                0, // rotation matrix
                                                                G4ThreeVector(0,-14.*cm,0)); // relative position

    G4SubtractionSolid *PelvisMarrow_Solid = new G4SubtractionSolid("PelvisMarrow_Solid", // name
                                                                    PelvisMarrow2, // larger volume to subtract from
                                                                    SubtrPelvisMarrow, // smaller volume to subtract
                                                                    0, // rotation matrix
                                                                    G4ThreeVector(0,22.*cm,-9.*cm)); // relative position

    PelvisMarrowLogical = new G4LogicalVolume(PelvisMarrow_Solid, // solid volume
                                              red_marrow, // material
                                              "PelvisMarrow_Logical"); // name

    // set red marrow color
    PelvisMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    PelvisMarrowLogical->SetUserLimits(fStepLimit);

    PelvisMarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                              G4ThreeVector(), // translation vector wrt pelvis
                                              PelvisMarrowLogical, // logical volume
                                              "PelvisMarrow_Physical", // name
                                              pMotherLogical, // mother volume
                                              false, // unused boolean
                                              0); // copy number

    // calculate mass
    G4double PelvisMarrowVol = PelvisMarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Pelvis Marrow = " << PelvisMarrowVol/cm3 << " cm^3" << G4endl;
    G4String PelvisMarrowMat = PelvisMarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Pelvis Marrow = " << PelvisMarrowMat << G4endl;
    G4double PelvisMarrowDensity = PelvisMarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << PelvisMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double PelvisMarrowMass = (PelvisMarrowVol)*PelvisMarrowDensity;
    G4cout << "Mass of Pelvis Marrow = " << PelvisMarrowMass/gram << " g" << G4endl;
}

```

```

// store organ information in maps
G4LogicalVolume* currentLogical = PelvisMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 40;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return PelvisMarrowPhysical;
}

// _____//

// construct MIRD left leg
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftLeg(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set left leg volume
    G4Cons *LeftLeg_Solid = new G4Cons("LeftLeg_Solid", // name
                                       0,                // rmin1
                                       2.*cm,            // rmax1
                                       0,                // rmin2
                                       10.*cm,           // rmax2
                                       40.*cm,           // dz
                                       0,                // start phi
                                       360.*deg);         // delta phi

    LeftLegLogical = new G4LogicalVolume(LeftLeg_Solid, // solid volume
                                       soft_tissue,     // material
                                       "LeftLeg_Logical"); // name

    // set visibility to invisible
    LeftLegLogical->SetVisAttributes(G4VisAttributes::Invisible);
    // set skin color
    // LeftLegLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftLegLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(180.*deg);
    rm->rotateY(180.*deg);
    LeftLegPhysical = new G4PVPlacement(rm, // rotation matrix
                                       G4ThreeVector(10.1*cm,0,-40.*cm), // translation vector wrt water phantom
                                       LeftLegLogical, // logical volume
                                       "LeftLeg_Physical", // name
                                       pMotherLogical, // mother volume
                                       false, // unused boolean
                                       0); // copy number

    // calculate mass
    G4double LeftLegVol = LeftLegLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Leg = " << LeftLegVol/cm3 << " cm^3" << G4endl;
    G4String LeftLegMat = LeftLegLogical->GetMaterial()->GetName();
    G4cout << "Material of Left Leg = " << LeftLegMat << G4endl;
    G4double LeftLegDensity = LeftLegLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << LeftLegDensity*cm3/g << " g/cm^3" << G4endl;
}

```

```

G4double LeftLegMass = (LeftLegVol)*LeftLegDensity;
G4cout << "Mass of LeftLeg = " << LeftLegMass/gram << " g" << G4endl;

return LeftLegPhysical;
}

//_____//
// construct MIRD right leg

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightLeg(G4LogicalVolume *pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set right leg volume
    G4Cons *RightLeg_Solid = new G4Cons("RightLeg_Solid", // name
                                        0, // rmin1
                                        2.*cm, // rmax1
                                        0, // rmin2
                                        10.*cm, // rmax2
                                        40.*cm, // dz
                                        0, // start phi
                                        360.*deg); // delta phi

    RightLegLogical = new G4LogicalVolume(RightLeg_Solid, // solid volume
                                        soft_tissue, // material
                                        "RightLeg_Logical"); // name

    // set visibility to invisible
    RightLegLogical->SetVisAttributes(G4VisAttributes::Invisible);
    // set skin color
    // RightLegLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightLegLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(180.*deg);
    rm->rotateY(180.*deg);
    RightLegPhysical = new G4PVPlacement(rm, // rotation matrix
                                        G4ThreeVector(-10.1*cm,0,-40.*cm), // translation vector wrt water phantom
                                        RightLegLogical, // logical volume
                                        "RightLeg_Physical", // name
                                        pMotherLogical, // mother volume
                                        false, // unused boolean
                                        0); // copy number

    // calculate mass
    G4double RightLegVol = RightLegLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Leg = " << RightLegVol/cm3 << " cm^3" << G4endl;
    G4String RightLegMat = RightLegLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Leg = " << RightLegMat << G4endl;
    G4double RightLegDensity = RightLegLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightLegDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RightLegMass = (RightLegVol)*RightLegDensity;
    G4cout << "Mass of Right Leg = " << RightLegMass/gram << " g" << G4endl;

    return RightLegPhysical;
}

//_____//
// construct MIRD left leg bone

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftLegBone(G4LogicalVolume
*pMotherLogical)
{

    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set left leg bone volume
    G4Cons *LeftLegBone_Solid = new G4Cons("LeftLegBone_Solid", // name
                                           0, // rmin1
                                           1.*cm, // rmax1
                                           0, // rmin2
                                           3.5*cm, // rmax2
                                           39.9*cm, // dz
                                           0, // start phi
                                           360.*deg); // delta phi

    LeftLegBoneLogical = new G4LogicalVolume(LeftLegBone_Solid, // solid volume
                                           g4bone, // material
                                           "LeftLegBone_Logical"); // name

    // set bone color
    LeftLegBoneLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftLegBoneLogical->SetUserLimits(fStepLimit);

    LeftLegBonePhysical = new G4PVPlacement(0, // rotation matrix
                                           G4ThreeVector(0, 0, 0.1*cm), // translation vector wrt left leg
                                           LeftLegBoneLogical, // logical volume
                                           "LeftLegBone_Physical", // name
                                           pMotherLogical, // mother volume
                                           false, // unused boolean
                                           0); // copy number

    // calculate mass
    G4double LeftLegBoneVol = LeftLegBoneLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Leg Bone = " << LeftLegBoneVol/cm3 << " cm^3" << G4endl;
    G4String LeftLegBoneMat = LeftLegBoneLogical->GetMaterial()->GetName();
    G4cout << "Material of Left Leg Bone = " << LeftLegBoneMat << G4endl;
    G4double LeftLegBoneDensity = LeftLegBoneLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << LeftLegBoneDensity*cm3/g << " g/cm^3" << G4endl;
    G4double LeftLegBoneMass = (LeftLegBoneVol)*LeftLegBoneDensity;
    G4cout << "Mass of Left Leg Bone = " << LeftLegBoneMass/gram << " g" << G4endl;

    return LeftLegBonePhysical;
}

// _____//

// construct left leg bone active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructLeftLegBoneMarrow(G4LogicalVolume
*pMotherLogical)
{

    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set left leg bone active marrow volume

    G4VSolid *LeftLegBoneMarrow_Solid = new G4Tubs("LeftLegBoneMarrow_Solid", // name
                                                    0, // inner radius
                                                    0.434*cm, // outer radius
                                                    39.899*cm, // z half length
                                                    0, // start phi

```

```

360.*deg); // delta phi

LeftLegBoneMarrowLogical = new G4LogicalVolume(LeftLegBoneMarrow_Solid, // solid volume
red_marrow, // material
"LeftLegBoneMarrow_Logical");// name

// set red marrow color
LeftLegBoneMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftLegBoneMarrowLogical->SetUserLimits(fStepLimit);

LeftLegBoneMarrowPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector wrt left leg bone
LeftLegBoneMarrowLogical, // logical volume
"LeftLegBoneMarrow_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double LeftLegBoneMarrowVol = LeftLegBoneMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Leg Bone Marrow = " << LeftLegBoneMarrowVol/cm3 << " cm^3" <<
G4endl;
G4String LeftLegBoneMarrowMat = LeftLegBoneMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Left Leg Bone Marrow = " << LeftLegBoneMarrowMat << G4endl;
G4double LeftLegBoneMarrowDensity = LeftLegBoneMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftLegBoneMarrowDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftLegBoneMarrowMass = (LeftLegBoneMarrowVol)*LeftLegBoneMarrowDensity;
G4cout << "Mass of Left Leg Bone Marrow = " << LeftLegBoneMarrowMass/gram << " g" << G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = LeftLegBoneMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 41;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftLegBoneMarrowPhysical;
}

// _____//

// construct MIRD right leg bone

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightLegBone(G4LogicalVolume
*pMotherLogical)
{

// get materials
G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

// set left leg bone volume
G4Cons *RightLegBone_Solid = new G4Cons("RightLegBone_Solid", // name
0, // rmin1
1.*cm, // rmax1
0, // rmin2
3.5*cm, // rmax2
39.9*cm, // dz
0, // start phi

```

```

360.*deg); // delta phi

RightLegBoneLogical = new G4LogicalVolume(RightLegBone_Solid, // solid volume
g4bone, // material
"RightLegBone_Logical"); // name

// set bone color
RightLegBoneLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
RightLegBoneLogical->SetUserLimits(fStepLimit);

RightLegBonePhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(0,0,0.1*cm), // translation vector wrt left leg
RightLegBoneLogical, // logical volume
"RightLegBone_Physical", // name
pMotherLogical, // mother volume
false, // unused boolean
0); // copy number

// calculate mass
G4double RightLegBoneVol = RightLegBoneLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Leg Bone = " << RightLegBoneVol/cm3 << " cm^3" << G4endl;
G4String RightLegBoneMat = RightLegBoneLogical->GetMaterial()->GetName();
G4cout << "Material of Right Leg Bone = " << RightLegBoneMat << G4endl;
G4double RightLegBoneDensity = RightLegBoneLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightLegBoneDensity*cm3/g << " g/cm^3" << G4endl;
G4double RightLegBoneMass = (RightLegBoneVol)*RightLegBoneDensity;
G4cout << "Mass of Right Leg Bone = " << RightLegBoneMass/gram << " g" << G4endl;

return RightLegBonePhysical;
}

// _____//

// construct right leg bone active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructRightLegBoneMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set right leg bone active marrow volume

    G4VSolid *RightLegBoneMarrow_Solid = new G4Tubs("RightLegBoneMarrow_Solid", // name
0, // inner radius
0.434*cm, // outer radius
39.899*cm, // z half length
0, // start phi
360.*deg); // delta phi

    RightLegBoneMarrowLogical = new G4LogicalVolume(RightLegBoneMarrow_Solid, // solid volume
red_marrow, // material
"RightLegBoneMarrow_Logical"); // name

    // set red marrow color
    RightLegBoneMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightLegBoneMarrowLogical->SetUserLimits(fStepLimit);

    RightLegBoneMarrowPhysical = new G4PVPlacement(0, // rotation matrix
G4ThreeVector(), // translation vector wrt right leg bone

```

```

        RightLegBoneMarrowLogical,    // logical volume
        "RightLegBoneMarrow_Physical", // name
        pMotherLogical,              // mother volume
        false,                       // unused boolean
        0);                          // copy number

// calculate mass
G4double RightLegBoneMarrowVol = RightLegBoneMarrowLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Leg Bone Marrow = " << RightLegBoneMarrowVol/cm3 << " cm^3" <<
G4endl;
G4String RightLegBoneMarrowMat = RightLegBoneMarrowLogical->GetMaterial()->GetName();
G4cout << "Material of Right Leg Bone Marrow = " << RightLegBoneMarrowMat << G4endl;
G4double RightLegBoneMarrowDensity = RightLegBoneMarrowLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightLegBoneMarrowDensity*cm3/g << " g/cm^3" <<
G4endl;
G4double RightLegBoneMarrowMass = (RightLegBoneMarrowVol)*RightLegBoneMarrowDensity;
G4cout << "Mass of Right Leg Bone Marrow = " << RightLegBoneMarrowMass/gram << " g" <<
G4endl;

// store organ information in maps
G4LogicalVolume* currentLogical = RightLegBoneMarrowLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 42;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightLegBoneMarrowPhysical;
}

// _____//

// construct MIRD left arm bone

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftArmBone(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set left arm bone volume
    G4VSolid *LeftArmBone_Solid = new G4EllipticalTube("LeftArmBone_Solid", // name
        1.4*cm, // dx
        2.7*cm, // dy
        34.5*cm); // dz

    LeftArmBoneLogical = new G4LogicalVolume(LeftArmBone_Solid, // solid volume
        g4bone, // material
        "LeftArmBone_Logical"); // name

    // set bone color
    LeftArmBoneLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftArmBoneLogical->SetUserLimits(fStepLimit);

    LeftArmBonePhysical = new G4PVPlacement(0, // rotation matrix
        G4ThreeVector(18.4*cm,0,-0.5*cm), // translation vector wrt trunk
        LeftArmBoneLogical, // logical volume
        "LeftArmBone_Physical", // name

```

```

        pMotherLogical,           // mother volume
        false,                   // unused boolean
        0);                      // copy number

// calculate mass
G4double LeftArmBoneVol = LeftArmBoneLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Arm Bone = " << LeftArmBoneVol/cm3 << " cm^3" << G4endl;
G4String LeftArmBoneMat = LeftArmBoneLogical->GetMaterial()->GetName();
G4cout << "Material of Left Arm Bone = " << LeftArmBoneMat << G4endl;
G4double LeftArmBoneDensity = LeftArmBoneLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftArmBoneDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftArmBoneMass = (LeftArmBoneVol)*LeftArmBoneDensity;
G4cout << "Mass of Left Arm Bone = " << LeftArmBoneMass/gram << " g" << G4endl;

return LeftArmBonePhysical;
}

// _____//

// construct left arm bone active marrow

G4VPhysicalVolume * DetectorConstruction::ConstructLeftArmBoneMarrow(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set left arm bone active marrow volume
    G4VSolid *LeftArmBoneMarrow_Solid = new G4EllipticalTube("LeftArmBoneMarrow_Solid", // name
                                                             0.197*cm,           // dx
                                                             0.38*cm,           // dy
                                                             34.5*cm);          // dz

    LeftArmBoneMarrowLogical = new G4LogicalVolume(LeftArmBoneMarrow_Solid,           // solid volume
                                                    red_marrow,                       // material
                                                    "LeftArmBoneMarrow_Logical"); // name

    // set red marrow color
    LeftArmBoneMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftArmBoneMarrowLogical->SetUserLimits(fStepLimit);

    LeftArmBoneMarrowPhysical = new G4PVPlacement(0,                               // rotation matrix
                                                    G4ThreeVector(), // translation vector wrt left arm bone
                                                    LeftArmBoneMarrowLogical, // logical volume
                                                    "LeftArmBoneMarrow_Physical", // name
                                                    pMotherLogical,           // mother volume
                                                    false,                   // unused boolean
                                                    0);                      // copy number

    // calculate mass
    G4double LeftArmBoneMarrowVol = LeftArmBoneMarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Arm Bone Marrow = " << LeftArmBoneMarrowVol/cm3 << " cm^3" <<
    G4endl;
    G4String LeftArmBoneMarrowMat = LeftArmBoneMarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Left Arm Bone Marrow = " << LeftArmBoneMarrowMat << G4endl;
    G4double LeftArmBoneMarrowDensity = LeftArmBoneMarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << LeftArmBoneMarrowDensity*cm3/g << " g/cm^3" << G4endl;
    G4double LeftArmBoneMarrowMass = (LeftArmBoneMarrowVol)*LeftArmBoneMarrowDensity;
    G4cout << "Mass of Left Arm Bone Marrow = " << LeftArmBoneMarrowMass/gram << " g" << G4endl;

    // store organ information in maps
    G4LogicalVolume* currentLogical = LeftArmBoneMarrowLogical;
    G4String OrganLogicalName = currentLogical->GetName();

```

```

G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 43;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return LeftArmBoneMarrowPhysical;
}

// _____//

// construct MIRD right arm bone

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightArmBone(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *g4bone = G4Material::GetMaterial("G4_BONE_CORTICAL_ICRP");

    // set right arm bone volume
    G4VSolid *RightArmBone_Solid = new G4EllipticalTube("RightArmBone_Solid", // name
                                                         1.4*cm,           // dx
                                                         2.7*cm,           // dy
                                                         34.5*cm);          // dz

    RightArmBoneLogical = new G4LogicalVolume(RightArmBone_Solid,           // solid volume
                                              g4bone,                       // material
                                              "RightArmBone_Logical");        // name

    // set bone color
    RightArmBoneLogical->SetVisAttributes(G4Colour(1.0, 0.9804, 0.8627));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightArmBoneLogical->SetUserLimits(fStepLimit);

    RightArmBonePhysical = new G4PVPlacement(0,                          // rotation matrix
                                              G4ThreeVector(-18.4*cm,0,-0.5*cm), // translation vector wrt trunk
                                              RightArmBoneLogical,           // logical volume
                                              "RightArmBone_Physical",         // name
                                              pMotherLogical,                 // mother volume
                                              false,                          // unused boolean
                                              0);                             // copy number

    // calculate mass
    G4double RightArmBoneVol = RightArmBoneLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Arm Bone = " << RightArmBoneVol/cm3 << " cm^3" << G4endl;
    G4String RightArmBoneMat = RightArmBoneLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Arm Bone = " << RightArmBoneMat << G4endl;
    G4double RightArmBoneDensity = RightArmBoneLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightArmBoneDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RightArmBoneMass = (RightArmBoneVol)*RightArmBoneDensity;
    G4cout << "Mass of Right Arm Bone = " << RightArmBoneMass/gram << " g" << G4endl;

    return RightArmBonePhysical;
}

// _____//

// construct right arm bone active marrow

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructRightArmBoneMarrow(G4LogicalVolume
*pMotherLogical)
{

    // get materials
    G4Material *red_marrow = G4Material::GetMaterial("red_marrow");

    // set right arm bone active marrow volume
    G4VSolid *RightArmBoneMarrow_Solid = new G4EllipticalTube("RightArmBoneMarrow_Solid", // name
                                                                0.197*cm,           // dx
                                                                0.38*cm,           // dy
                                                                34.5*cm);          // dz

    RightArmBoneMarrowLogical = new G4LogicalVolume(RightArmBoneMarrow_Solid, // solid volume
                                                    red_marrow,           // material
                                                    "RightArmBoneMarrow_Logical"); // name

    // set red marrow color
    RightArmBoneMarrowLogical->SetVisAttributes(G4Colour(0.847, 0.396, 0.3098));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightArmBoneMarrowLogical->SetUserLimits(fStepLimit);

    RightArmBoneMarrowPhysical = new G4PVPlacement(0, // rotation matrix
                                                    G4ThreeVector(), // translation vector wrt right arm bone
                                                    RightArmBoneMarrowLogical, // logical volume
                                                    "RightArmBoneMarrow_Physical", // name
                                                    pMotherLogical, // mother volume
                                                    false, // unused boolean
                                                    0); // copy number

    // calculate mass
    G4double RightArmBoneMarrowVol = RightArmBoneMarrowLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Arm Bone Marrow = " << RightArmBoneMarrowVol/cm3 << " cm^3" <<
    G4endl;
    G4String RightArmBoneMarrowMat = RightArmBoneMarrowLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Arm Bone Marrow = " << RightArmBoneMarrowMat << G4endl;
    G4double RightArmBoneMarrowDensity = RightArmBoneMarrowLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightArmBoneMarrowDensity*cm3/g << " g/cm^3" <<
    G4endl;
    G4double RightArmBoneMarrowMass = (RightArmBoneMarrowVol)*RightArmBoneMarrowDensity;
    G4cout << "Mass of Right Arm Bone Marrow = " << RightArmBoneMarrowMass/gram << " g" <<
    G4endl;

    // store organ information in maps
    G4LogicalVolume* currentLogical = RightArmBoneMarrowLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 44;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return RightArmBoneMarrowPhysical;
}

// _____//

// construct MIRD uterus

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDUterus(G4LogicalVolume *pMotherLogical)

```

```

{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set uterus volume
    G4VSolid *Uterus_Solid = new G4Ellipsoid("Uterus_Solid", // name
                                             2.5*cm,          // ax
                                             1.5*cm,          // by
                                             5.*cm,           // cz
                                             -5.*cm,          // zcut1
                                             2.5*cm);         // zcut2

    UterusLogical = new G4LogicalVolume(Uterus_Solid,        // solid volume
                                       soft_tissue,          // material
                                       "Uterus_Logical");     // name

    // set uterus color
    UterusLogical->SetVisAttributes(G4Colour(1.0, 0.7098, 0.6196));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    UterusLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm = new G4RotationMatrix();
    rm->rotateX(90.*deg);
    UterusPhysical = new G4PVPlacement(rm,                  // rotation matrix
                                       G4ThreeVector(0, 2.*cm, -21.*cm), // translation vector wrt trunk
                                       UterusLogical,        // logical volume
                                       "Uterus_Physical",    // name
                                       pMotherLogical,       // mother volume
                                       false,                // unused boolean
                                       0);                   // copy number

    // calculate mass
    G4double UterusVol = UterusLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Uterus = " << UterusVol/cm3 << "cm^3" << G4endl;
    G4String UterusMat = UterusLogical->GetMaterial()->GetName();
    G4cout << "Material of Uterus = " << UterusMat << G4endl;
    G4double UterusDensity = UterusLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << UterusDensity*cm3/g << " g/cm^3" << G4endl;
    G4double UterusMass = (UterusVol)*UterusDensity;
    G4cout << "Mass of Uterus = " << UterusMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = UterusLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0, OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 45;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return UterusPhysical;
}

// _____//

// construct MIRD left ovary

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftOvary(G4LogicalVolume
*pMotherLogical)
{

```

```

// get materials
G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

// set left ovary volume
G4VSolid *LeftOvary_Solid = new G4Ellipsoid("LeftOvary_Solid", // name
                                           1.0*cm,             // ax
                                           0.5*cm,             // by
                                           2.*cm);             // cz

LeftOvaryLogical = new G4LogicalVolume(LeftOvary_Solid,        // solid volume
                                       soft_tissue,             // material
                                       "LeftOvary_Logical");     // name

// set ovary color
LeftOvaryLogical->SetVisAttributes(G4Colour(0.8353, 0.5529, 0.4431));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftOvaryLogical->SetUserLimits(fStepLimit);

LeftOvaryPhysical = new G4PVPlacement(0,                      // rotation matrix
                                       G4ThreeVector(-6.*cm,0.5*cm,-20.*cm), // translation vector wrt trunk
                                       LeftOvaryLogical,        // logical volume
                                       "LeftOvary_Physical",     // name
                                       pMotherLogical,          // mother volume
                                       false,                   // unused boolean
                                       0);                       // copy number

// calculate mass
G4double LeftOvaryVol = LeftOvaryLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Ovary = " << LeftOvaryVol/cm3 << " cm^3" << G4endl;
G4String LeftOvaryMat = LeftOvaryLogical->GetMaterial()->GetName();
G4cout << "Material of Left Ovary = " << LeftOvaryMat << G4endl;
G4double LeftOvaryDensity = LeftOvaryLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftOvaryDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftOvaryMass = (LeftOvaryVol)*LeftOvaryDensity;
G4cout << "Mass of Left Ovary = " << LeftOvaryMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftOvaryLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 46;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftOvaryPhysical;
}

// _____//

// construct MIRD right ovary
G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightOvary(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

```

```

// set right ovary volume
G4VSolid *RightOvary_Solid = new G4Ellipsoid("RightOvary_Solid", // name
                                             1.0*cm,             // ax
                                             0.5*cm,             // by
                                             2.0*cm);            // cz

RightOvaryLogical = new G4LogicalVolume(RightOvary_Solid,        // solid volume
                                         soft_tissue,            // material
                                         "RightOvary_Logical");    // name

// set ovary color
RightOvaryLogical->SetVisAttributes(G4Colour(0.8353, 0.5529, 0.4431));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
RightOvaryLogical->SetUserLimits(fStepLimit);

RightOvaryPhysical = new G4PVPlacement(0,                        // rotation matrix
                                         G4ThreeVector(6.*cm,0.5*cm,-20.*cm), // translation vector wrt trunk
                                         RightOvaryLogical,        // logical volume
                                         "RightOvary_Physical",    // name
                                         pMotherLogical,           // mother volume
                                         false,                    // unused boolean
                                         0);                        // copy number

// calculate mass
G4double RightOvaryVol = RightOvaryLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Right Ovary = " << RightOvaryVol/cm3 << " cm^3" << G4endl;
G4String RightOvaryMat = RightOvaryLogical->GetMaterial()->GetName();
G4cout << "Material of Right Ovary = " << RightOvaryMat << G4endl;
G4double RightOvaryDensity = RightOvaryLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << RightOvaryDensity*cm3/g << " g/cm^3" << G4endl;
G4double RightOvaryMass = (RightOvaryVol)*RightOvaryDensity;
G4cout << "Mass of Right Ovary = " << RightOvaryMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = RightOvaryLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 47;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return RightOvaryPhysical;
}

// _____//

// construct MIRD left breast

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftBreast(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set left breast volume
    G4Ellipsoid *Breast1 = new G4Ellipsoid("Breast1",           // name
                                             4.95*cm,             // ax
                                             4.35*cm,             // by
                                             4.15*cm);            // cz

```

```

G4EllipticalTube *SubtrBreast = new G4EllipticalTube("SubtrBreast", // name
                                                    20.*cm,           // dx
                                                    10.*cm,           // dy
                                                    35.*cm);          // dz

G4RotationMatrix *rm1 = new G4RotationMatrix();
rm1->rotateX(90.*deg);
G4SubtractionSolid *LeftBreast_Solid = new G4SubtractionSolid("LeftBreast_Solid", // name
Breast1,           // larger volume to subtract from
SubtrBreast,       // smaller volume to subtract
rm1,               // rotation matrix
G4ThreeVector(-10.*cm,0,-8.66*cm)); // relative position

LeftBreastLogical = new G4LogicalVolume(LeftBreast_Solid,           // solid volume
soft_tissue,           // material
"LeftBreast_Logical"); // name

// set breast color
LeftBreastLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
LeftBreastLogical->SetUserLimits(fStepLimit);

G4RotationMatrix *rm2 = new G4RotationMatrix();
rm2->rotateX(90.*deg);
rm2->rotateZ(16.*deg);
LeftBreastPhysical = new G4PVPlacement(rm2,           // rotation matrix
G4ThreeVector(10.*cm,9.1*cm,52.*cm), // translation vector wrt water phantom
LeftBreastLogical, // logical volume
"LeftBreast_Physical", // name
pMotherLogical,     // mother volume
false,              // unused boolean
0);                  // copy number

// calculate mass
G4double LeftBreastVol = LeftBreastLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Left Breast = " << LeftBreastVol/cm3 << " cm^3" << G4endl;
G4String LeftBreastMat = LeftBreastLogical->GetMaterial()->GetName();
G4cout << "Material of Left Breast = " << LeftBreastMat << G4endl;
G4double LeftBreastDensity = LeftBreastLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << LeftBreastDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftBreastMass = (LeftBreastVol)*LeftBreastDensity;
G4cout << "Mass of Left Breast = " << LeftBreastMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftBreastLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 48;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftBreastPhysical;
}

// _____//
// construct MIRD right breast

```

```

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightBreast(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set right breast volume
    G4Ellipsoid *Breast1 = new G4Ellipsoid("Breast1",      // name
                                           4.95*cm,         // ax
                                           4.35*cm,         // by
                                           4.15*cm);        // cz

    G4EllipticalTube *SubtrBreast = new G4EllipticalTube("SubtrBreast", // name
                                                         20.*cm,         // dx
                                                         10.*cm,         // dy
                                                         35.*cm);        // dz

    G4RotationMatrix *rm1 = new G4RotationMatrix();
    rm1->rotateX(90.*deg);
    G4SubtractionSolid *RightBreast_Solid = new G4SubtractionSolid("RightBreast_Solid", // name
                                                                    Breast1,         // larger volume to subtract from
                                                                    SubtrBreast,     // smaller volume to subtract
                                                                    rm1,             // rotation matrix
                                                                    G4ThreeVector(10.*cm,0,-8.66*cm)); // relative position

    RightBreastLogical = new G4LogicalVolume(RightBreast_Solid,          // solid volume
                                             soft_tissue,                // material
                                             "RightBreast_Logical");      // name

    // set breast color
    RightBreastLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightBreastLogical->SetUserLimits(fStepLimit);

    G4RotationMatrix *rm3 = new G4RotationMatrix();
    rm3->rotateX(90.*deg);
    rm3->rotateZ(-16.*deg);
    RightBreastPhysical = new G4PVPlacement(rm3,                    // rotation matrix
                                           G4ThreeVector(-10.*cm,9.1*cm,52.*cm), // translation vector wrt water phantom
                                           RightBreastLogical,        // logical volume
                                           "RightBreast_Physical",    // name
                                           pMotherLogical,            // mother volume
                                           false,                     // unused boolean
                                           0);                         // copy number

    // calculate mass
    G4double RightBreastVol = RightBreastLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Breast = " << RightBreastVol/cm3 << " cm^3" << G4endl;
    G4String RightBreastMat = RightBreastLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Breast = " << RightBreastMat << G4endl;
    G4double RightBreastDensity = RightBreastLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightBreastDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RightBreastMass = (RightBreastVol)*RightBreastDensity;
    G4cout << "Mass of Right Breast = " << RightBreastMass/gram << " g" << G4endl;

    // store organ info in maps
    G4LogicalVolume* currentLogical = RightBreastLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 49;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
}

```

```

    return RightBreastPhysical;
}

// _____//

// construct MIRD male genitalia

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDMaleGenitalia(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set male genitalia volume
    G4Trap *MaleGenitalia = new G4Trap("MaleGenitalia", // name
        2.4*cm, // pDz
        0, // pTheta
        0, // pPhi
        4.76*cm, // pDy1
        9.52*cm, // pDx1
        9.52*cm, // pDx2
        0, // pAlp1
        5.*cm, // pDy2
        10.*cm, // pDx3
        10.*cm, // pDx4
        0); // pAlp2

    G4Cons *SubtrGenitaliaL = new G4Cons("SubtrGenitaliaL", // name
        0, // rmin1
        9.51*cm, // rmax1
        0, // rmin2
        10.01*cm, // rmax2
        2.5*cm, // dz
        0, // start phi
        360.*deg); // delta phi

    G4Cons *SubtrGenitaliaR = new G4Cons("SubtrGenitaliaR", // name
        0, // rmin1
        9.51*cm, // rmax1
        0, // rmin2
        10.01*cm, // rmax2
        2.5*cm, // dz
        0, // start phi
        360.*deg); // delta phi

    G4UnionSolid *SubtrGenitalia = new G4UnionSolid("SubtrGenitalia", // name
        SubtrGenitaliaL, // 1st volume to union
        SubtrGenitaliaR, // 2nd volume to union
        0, // rotation matrix
        G4ThreeVector(20.*cm,0,0)); // relative position

    G4VSolid *MaleGenitalia_Solid = new G4SubtractionSolid("MaleGenitalia_Solid", // name
        MaleGenitalia, // larger volume to subtract from
        SubtrGenitalia, // smaller volume to subtract
        0, // rotation matrix
        G4ThreeVector(-10.*cm,-5.*cm,0)); // relative position

    MaleGenitaliaLogical = new G4LogicalVolume(MaleGenitalia_Solid, // solid volume
        soft_tissue, // material
        "MaleGenitalia_Logical"); // name

    // set visibility to invisible
    MaleGenitaliaLogical->SetVisAttributes(G4VisAttributes::Invisible);
    // set skin color
    MaleGenitaliaLogical->SetVisAttributes(G4Colour(0.949, 0.8078, 0.5569));
}

```

```

// manually set step size for volume
G4double maxStep = 1.*cm;
fStepLimit = new G4UserLimits(maxStep);
MaleGenitaliaLogical->SetUserLimits(fStepLimit);

MaleGenitaliaPhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(0,5.*cm,-2.4*cm), // translation vector wrt water phantom
MaleGenitaliaLogical,           // logical volume
"MaleGenitalia_Physical",       // name
pMotherLogical,                 // mother volume
false,                           // unused boolean
0);                             // copy number

// calculate mass
G4double MaleGenitaliaVol = MaleGenitaliaLogical->GetSolid()->GetCubicVolume();
G4cout << "Volume of Male Genitalia = " << MaleGenitaliaVol/cm3 << " cm^3" << G4endl;
G4String MaleGenitaliaMat = MaleGenitaliaLogical->GetMaterial()->GetName();
G4cout << "Material of Male Genitalia = " << MaleGenitaliaMat << G4endl;
G4double MaleGenitaliaDensity = MaleGenitaliaLogical->GetMaterial()->GetDensity();
G4cout << "Density of Material = " << MaleGenitaliaDensity*cm3/g << " g/cm^3" << G4endl;
G4double MaleGenitaliaMass = (MaleGenitaliaVol)*MaleGenitaliaDensity;
G4cout << "Mass of Male Genitalia = " << MaleGenitaliaMass/gram << " g" << G4endl;

return MaleGenitaliaPhysical;
}

// _____//

// construct MIRD left testicle

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDLeftTesticle(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set left testicle volume
    G4VSolid *LeftTesticle_Solid = new G4Ellipsoid("LeftTesticle_Solid", // name
1.3*cm, // ax
1.5*cm, // by
2.3*cm); // cz

    LeftTesticleLogical = new G4LogicalVolume(LeftTesticle_Solid, // solid volume
soft_tissue, // material
"LeftTesticle_Logical"); // name

    // set testicle color
    LeftTesticleLogical->SetVisAttributes(G4Colour(0.8353, 0.5529, 0.4431));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    LeftTesticleLogical->SetUserLimits(fStepLimit);

    LeftTesticlePhysical = new G4PVPlacement(0,           // rotation matrix
G4ThreeVector(1.4*cm,3.*cm,0), // translation vector wrt male genitalia
LeftTesticleLogical,           // logical volume
"LeftTesticle_Physical",       // name
pMotherLogical,                 // mother volume
false,                           // unused boolean
0);                             // copy number

    // calculate mass
    G4double LeftTesticleVol = LeftTesticleLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Left Testicle = " << LeftTesticleVol/cm3 << " cm^3" << G4endl;
    G4String LeftTesticleMat = LeftTesticleLogical->GetMaterial()->GetName();
    G4cout << "Material of Left Testicle = " << LeftTesticleMat << G4endl;
    G4double LeftTesticleDensity = LeftTesticleLogical->GetMaterial()->GetDensity();

```

```

G4cout << "Density of Material = " << LeftTesticleDensity*cm3/g << " g/cm^3" << G4endl;
G4double LeftTesticleMass = (LeftTesticleVol)*LeftTesticleDensity;
G4cout << "Mass of Left Testicle = " << LeftTesticleMass/gram << " g" << G4endl;

// store organ info in maps
G4LogicalVolume* currentLogical = LeftTesticleLogical;
G4String OrganLogicalName = currentLogical->GetName();
G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
Map_OrganNameID[OrganName] = 50;
Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
Map_OrganNameEdep[OrganName] = 0.0;
Map_OrganNameDose[OrganName] = 0.0;
Map_OrganNameDoseSumW2[OrganName] = 0.0;
Map_OrganNameDoseEqui[OrganName] = 0.0;
Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

return LeftTesticlePhysical;
}

// _____//

// construct MIRD right testicle

G4VPhysicalVolume * DetectorConstruction::ConstructMIRDRightTesticle(G4LogicalVolume
*pMotherLogical)
{
    // get materials
    G4Material *soft_tissue = G4Material::GetMaterial("soft_tissue");

    // set right testicle volume
    G4VSolid *RightTesticle_Solid = new G4Ellipsoid("RightTesticle_Solid", // name
                                                    1.3*cm, // ax
                                                    1.5*cm, // by
                                                    2.3*cm); // cz

    RightTesticleLogical = new G4LogicalVolume(RightTesticle_Solid, // solid volume
                                                soft_tissue, // material
                                                "RightTesticle_Logical"); // name

    // set testicle color
    RightTesticleLogical->SetVisAttributes(G4Colour(0.8353, 0.5529, 0.4431));

    // manually set step size for volume
    G4double maxStep = 1.*cm;
    fStepLimit = new G4UserLimits(maxStep);
    RightTesticleLogical->SetUserLimits(fStepLimit);

    RightTesticlePhysical = new G4PVPlacement(0, // rotation matrix
                                                G4ThreeVector(-1.4*cm,3.*cm,0), // translation vector wrt male genitalia
                                                RightTesticleLogical, // logical volume
                                                "RightTesticle_Physical", // name
                                                pMotherLogical, // mother volume
                                                false, // unused boolean
                                                0); // copy number

    // calculate mass
    G4double RightTesticleVol = RightTesticleLogical->GetSolid()->GetCubicVolume();
    G4cout << "Volume of Right Testicle = " << RightTesticleVol/cm3 << " cm^3" << G4endl;
    G4String RightTesticleMat = RightTesticleLogical->GetMaterial()->GetName();
    G4cout << "Material of Right Testicle = " << RightTesticleMat << G4endl;
    G4double RightTesticleDensity = RightTesticleLogical->GetMaterial()->GetDensity();
    G4cout << "Density of Material = " << RightTesticleDensity*cm3/g << " g/cm^3" << G4endl;
    G4double RightTesticleMass = (RightTesticleVol)*RightTesticleDensity;
    G4cout << "Mass of Right Testicle = " << RightTesticleMass/gram << " g" << G4endl;

    // store organ info in maps

```

```

    G4LogicalVolume* currentLogical = RightTesticleLogical;
    G4String OrganLogicalName = currentLogical->GetName();
    G4String OrganName = OrganLogicalName.substr(0,OrganLogicalName.size()-8);
    Map_OrganNameID[OrganName] = 51;
    Map_OrganNameMass[OrganName] = currentLogical->GetMass()/g;
    Map_OrganNameVolume[OrganName] = currentLogical->GetSolid()->GetCubicVolume()/cm3;
    Map_OrganNameDensity[OrganName] = currentLogical->GetMaterial()->GetDensity()/(g/cm3);
    Map_OrganNameEdep[OrganName] = 0.0;
    Map_OrganNameDose[OrganName] = 0.0;
    Map_OrganNameDoseSumW2[OrganName] = 0.0;
    Map_OrganNameDoseEqui[OrganName] = 0.0;
    Map_OrganNameDoseEquiSumW2[OrganName] = 0.0;

    return RightTesticlePhysical;
}

// _____//

void DetectorConstruction::SetStepLimitsForVolume(G4LogicalVolume *logicalVolume, G4double
stepMax)
{
    G4UserLimits *userLimits = new G4UserLimits(stepMax);
    logicalVolume->SetUserLimits(userLimits);
}

void DetectorConstruction::SetupScoring(G4LogicalVolume *scoringVolume, const G4String
&detectorName,G4VPrimitiveScorer *aScorer)
{
    // create a new sensitive detector
    G4MultiFunctionalDetector *aDetector = new G4MultiFunctionalDetector(detectorName);
    // get pointer to detector manager
    G4SDManager *sdManager = G4SDManager::GetSDMpointer();
    // register detector with manager
    sdManager->AddNewDetector(aDetector);
    // attach detector to scoring volume
    scoringVolume->SetSensitiveDetector(aDetector);
    // register primitive scorer with detector
    aDetector->RegisterPrimitive(aScorer);
}

```

## Appendix A8: MATLAB r2018a Script for Solenoidal Magnetic Field 1.5 T (Solenoid\_6plus1\_txt.m)

```

close all
clear all

% assume world volume is a sphere of radius 40 m
% create tabulated list of coordinates at resolution of 1 m
% assign Bx, By, Bz for each coordinate on the list for import into Geant4

% set up solenoidal fields based on Callaghan & Maslen 1960 NASA publication

mu_o = (4*pi)*10^-7; % permeability of free space
mu_s = 0.27;         % relative permeability of YBCO superconductor

nx = 81;
ny = 81;
nz = 81;

xmin = -40;
xmax = 40;
ymin = -40;
ymax = 40;
zmin = -40;
zmax = 40;

t = 1; % row count

x = -40;
for i = 1:nx % stepping through x
    y = -40; % restart y for each x
    for j = 1:ny % stepping through y
        z = -40; % restart z for each y
        for m = 1:nz % stepping through z
            theta = atan(y/x); % for cartesian/cylindrical conversion

            % field from correction solenoid
            a = 3.2; % radius of correction solenoid (m)
            L = 20; % length of correction solenoid (m)
            r = sqrt(x^2 + y^2);
            Bo = 0.355; % correction solenoid field (T) for 1.5 T shielding fields,
                        % -z direction
            % Bo = 0.710; % correction solenoid field (T) for 3 T shielding fields,
                        % -z direction
            % Bo = 1.657; % correction solenoid field (T) for 7 T shielding fields,
                        % -z direction
            ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
                                % permeability
            kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
            kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
            k = [kmax^2, kmin^2];
            [K,E] = ellipke(k);
            P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) * K(2) - (E(1)/kmax));
            Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) * K(1) - (E(2)/kmin));
            Br_corr = P-Q;
            Bx_corr = Br_corr*cos(theta);
            By_corr = Br_corr*sin(theta);
            angle_max = atan(abs((z+(L/2)) / (a-r)));
            angle_min = atan(abs((z-(L/2)) / (a-r)));
            R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) * K(2) + ( ((a-
            r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambd(angle_max,kmax)));
            S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) * K(1) + ( ((a-r)*(z-
            (L/2)))/(abs((a-r)*(z-(L/2)))) * hlambd(angle_min,kmin)));
            Bz_corr = R-S;

            % field from shielding solenoid 1 (displaced +8 m in x direction)
            a = 4; % radius of shielding solenoid (m)
            L = 20; % length of shielding solenoid (m)
            Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
            % Bo = 3; % magnetic field of shielding solenoid (T), +z direction
            % Bo = 7; % magnetic field of shielding solenoid (T), +z direction
            ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
                                % permeability

```

```

displ_x = 8; % solenoid x displacement from origin (m)
displ_y = 0; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_1 = P-Q;
Bx_1 = Br_1*cos(theta);
By_1 = Br_1*sin(theta);
if x >=0 && x< 8
Bx_1 = -Bx_1;
else
end
angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambda(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambda(angle_min,kmin)));
Bz_1 = R-S;

% field from shielding solenoid 2 (displaced +4 m in x direction and -6.928 m in
y direction)
a = 4; % radius of shielding solenoid (m)
L = 20; % length of shielding solenoid (m)
Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
% Bo = 3; % magnetic field of shielding solenoid (T), +z direction
% Bo = 7; % magnetic field of shielding solenoid (T), +z direction
ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
permeability
displ_x = 4; % solenoid x displacement from origin (m)
displ_y = -6.928; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_2 = P-Q;
Bx_2 = Br_2*cos(theta);
By_2 = Br_2*sin(theta);
angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambda(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambda(angle_min,kmin)));
Bz_2 = R-S;

% field from shielding solenoid 3 (displaced -4 m in x direction and -6.928 m in
y direction)
a = 4; % radius of shielding solenoid (m)
L = 20; % length of shielding solenoid (m)
Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
% Bo = 3; % magnetic field of shielding solenoid (T), +z direction
% Bo = 7; % magnetic field of shielding solenoid (T), +z direction
ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
permeability
displ_x = -4; % solenoid x displacement from origin (m)
displ_y = -6.928; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_3 = P-Q;
Bx_3 = Br_3*cos(theta);
By_3 = Br_3*sin(theta);

```

```

angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambda(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambda(angle_min,kmin)));
Bz_3 = R-S;

% field from shielding solenoid 4 (displaced -8 m in x direction)
a = 4; % radius of shielding solenoid (m)
L = 20; % length of shielding solenoid (m)
Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
% Bo = 3; % magnetic field of shielding solenoid (T), +z direction
% Bo = 7; % magnetic field of shielding solenoid (T), +z direction
ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
permeability
displ_x = -8; % solenoid x displacement from origin (m)
displ_y = 0; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_4 = P-Q;
Bx_4 = Br_4*cos(theta);
By_4 = Br_4*sin(theta);
angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambda(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambda(angle_min,kmin)));
Bz_4 = R-S;

% field from shielding solenoid 5 (displaced -4 m in x direction and 6.928 m in
y direction)
a = 4; % radius of shielding solenoid (m)
L = 20; % length of shielding solenoid (m)
Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
% Bo = 3; % magnetic field of shielding solenoid (T), +z direction
% Bo = 7; % magnetic field of shielding solenoid (T), +z direction
ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
permeability
displ_x = -4; % solenoid x displacement from origin (m)
displ_y = 6.928; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_5 = P-Q;
Bx_5 = Br_5*cos(theta);
By_5 = Br_5*sin(theta);
angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambda(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambda(angle_min,kmin)));
Bz_5 = R-S;

% field from shielding solenoid 6 (displaced +4 m in x direction and +6.928 m in
y direction)
a = 4; % radius of shielding solenoid (m)
L = 20; % length of shielding solenoid (m)
Bo = 1.5; % magnetic field of shielding solenoid (T), +z direction
% Bo = 3; % magnetic field of shielding solenoid (T), +z direction
% Bo = 7; % magnetic field of shielding solenoid (T), +z direction
ni = Bo/(mu_o*mu_s); % turn density x current in each coil derived from Bo and
permeability

```

```

displ_x = 4; % solenoid x displacement from origin (m)
displ_y = 6.928; % solenoid y displacement from origin (m)
r = sqrt((x-displ_x)^2 + (y-displ_y)^2);
kmax = sqrt((4*a*r)/((z+(L/2))^2 + (a+r)^2));
kmin = sqrt((4*a*r)/((z-(L/2))^2 + (a+r)^2));
k = [kmax^2, kmin^2];
[K,E] = ellipke(k);
P = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmax^2)/(2*kmax)) *K(2) - (E(1)/kmax));
Q = (mu_s*mu_o*ni/pi) * sqrt(a/r) * (((2-kmin^2)/(2*kmin)) *K(1) - (E(2)/kmin));
Br_6 = P-Q;
Bx_6 = Br_6*cos(theta);
By_6 = Br_6*sin(theta);
angle_max = atan(abs((z+(L/2)) / (a-r)));
angle_min = atan(abs((z-(L/2)) / (a-r)));
R = (mu_s*mu_o*ni/4) * (((z+(L/2))*kmax)/(pi*sqrt(a*r)) *K(2) + ( ((a-
r)*(z+(L/2)))/(abs((a-r)*(z+(L/2)))) * hlambd(angle_max,kmax)));
S = (mu_s*mu_o*ni/4) * (((z-(L/2))*kmin)/(pi*sqrt(a*r)) *K(1) + ( ((a-r)*(z-
(L/2)))/(abs((a-r)*(z-(L/2)))) * hlambd(angle_min,kmin)));
Bz_6 = R-S;

% sum together Bx, By, and Bz components from all 7 solenoids
Bx = Bx_corr + Bx_1 + Bx_2 + Bx_3 + Bx_4 + Bx_5 + Bx_6;
By = By_corr + By_1 + By_2 + By_3 + By_4 + By_5 + By_6;
Bz = Bz_corr + Bz_1 + Bz_2 + Bz_3 + Bz_4 + Bz_5 + Bz_6;

B_list(t,1) = Bx;
B_list(t,2) = By;
B_list(t,3) = Bz;

% check to make sure the coordinates are in the correct order for the text file
% coord_list(t,1) = x;
% coord_list(t,2) = y;
% coord_list(t,3) = z;

t = t + 1;
z = z + 1;
m = m + 1;
end
y = y + 1;
j = j + 1;
end
x = x + 1;
i = i + 1;
end

% replace any NaN or Inf values with 0
B_list(isnan(B_list)) = 0;
B_list(isinf(B_list)) = 0;
line_1 = [nx, ny, nz];
line_2 = [xmin, xmax];
line_3 = [ymin, ymax];
line_4 = [zmin, zmax];

% open/create a text file, paste values for import into Geant4 script
filename = sprintf('Blist_%dT.txt', Bo);
fileID = fopen(filename,'w+');
fprintf(fileID, '%5d %5d %5d\n', line_1);
fprintf(fileID, '%5d %5d\n', line_2);
fprintf(fileID, '%5d %5d\n', line_3);
fprintf(fileID, '%5d %5d\n', line_4);
fprintf(fileID, '%12.8f %12.8f %12.8f\n', B_list);
fclose(fileID);

```

Appendix A9: Organ Dose Calculations in Excel

Organ Name	Organ ID	Mass (g)	Volume (L)	Density (g/cm3)	Energy Deposit (MeV)	Dose (Gy)	Dose Std Dev (Gy)	Dose Std Dev (%)	Dose Equiv. (Sv)	Dose Equiv. Std Dev (Sv)	Dose Equiv. Std Dev (%)
Cervical Marrow	1	132.6	119.469	1.06	F2	G2	H2	I2	J2	K2	L2
Left Eye Lens	2	0.221	0.260276	1.07	F3	G3	H3	I3	J3	K3	L3
Right Eye Lens	3	0.221	0.260276	1.07	F4	G4	H4	I4	J4	K4	L4
Brain	4	1329	1470.27	1.04	F5	G5	H5	I5	J5	K5	L5
Upper Spine Marrow	5	58.39	55.0841	1.06	F6	G6	H6	I6	J6	K6	L6
Thyroid	6	12.77	12.9366	0.9869	F7	G7	H7	I7	J7	K7	L7
Heart	7	362.8	367.597	0.9869	F8	G8	H8	I8	J8	K8	L8
Left Lung	8	498.5	1688.59	0.2958	F9	G9	H9	I9	J9	K9	L9
Right Lung	9	498.7	1685.94	0.2958	F10	G10	H10	I10	J10	K10	L10
Left Kidney	10	142.3	144.165	0.9869	F11	G11	H11	I11	J11	K11	L11
Right Kidney	11	141.9	143.823	0.9869	F12	G12	H12	I12	J12	K12	L12
Left Adrenal	12	7.751	7.85398	0.9869	F13	G13	H13	I13	J13	K13	L13
Right Adrenal	13	7.751	7.85398	0.9869	F14	G14	H14	I14	J14	K14	L14
Bladder	14	24.46	21.0746	1.06	F15	G15	H15	I15	J15	K15	L15
Blb1 Marrow	15	24.45	21.0655	1.06	F16	G16	H16	I16	J16	K16	L16
Blb2 Marrow	16	24.44	21.2456	1.06	F17	G17	H17	I17	J17	K17	L17
Blb3 Marrow	17	24.44	21.2483	1.06	F18	G18	H18	I18	J18	K18	L18
Blb4 Marrow	18	24.49	21.1056	1.06	F19	G19	H19	I19	J19	K19	L19
Blb5 Marrow	19	24.51	21.1201	1.06	F20	G20	H20	I20	J20	K20	L20
Blb6 Marrow	20	24.35	21.2101	1.06	F21	G21	H21	I21	J21	K21	L21
Blb7 Marrow	21	24.66	21.2638	1.06	F22	G22	H22	I22	J22	K22	L22
Blb8 Marrow	22	24.47	21.0856	1.06	F23	G23	H23	I23	J23	K23	L23
Blb10 Marrow	23	24.4	21.0164	1.06	F24	G24	H24	I24	J24	K24	L24
Blb11 Marrow	24	24.51	21.1201	1.06	F25	G25	H25	I25	J25	K25	L25
Blb12 Marrow	25	24.36	21.9764	1.06	F26	G26	H26	I26	J26	K26	L26
Left Cervical Marrow	26	5.744	5.41925	1.06	F27	G27	H27	I27	J27	K27	L27
Right Cervical Marrow	27	5.744	5.41925	1.06	F28	G28	H28	I28	J28	K28	L28
Left Scapula Marrow	28	21.01	19.8184	1.06	F29	G29	H29	I29	J29	K29	L29
Right Scapula Marrow	29	21.03	19.8418	1.06	F30	G30	H30	I30	J30	K30	L30
Small Intestine	30	1006	1018.97	0.9869	F31	G31	H31	I31	J31	K31	L31
Lower Large Intestine	31	339.5	344.054	0.9869	F32	G32	H32	I32	J32	K32	L32
Upper Large Intestine	32	429.3	435.025	0.9869	F33	G33	H33	I33	J33	K33	L33
Liver	33	1543	1563.14	0.9869	F34	G34	H34	I34	J34	K34	L34
Stomach	34	396.9	402.124	0.9869	F35	G35	H35	I35	J35	K35	L35
Pancreas	35	60.32	61.1247	0.9869	F36	G36	H36	I36	J36	K36	L36
Spleen	36	173.6	175.929	0.9869	F37	G37	H37	I37	J37	K37	L37
Bladder	37	51.49	52.1695	0.9869	F38	G38	H38	I38	J38	K38	L38
Thymus	38	575.8	543.244	1.06	F39	G39	H39	I39	J39	K39	L39
Midline Spine Marrow	39	17.71	16.1432	1.06	F40	G40	H40	I40	J40	K40	L40
Posterior Marrow	40	27.21	26.1432	1.06	F41	G41	H41	I41	J41	K41	L41
Left Leg Bone Marrow	41	50.05	47.2195	1.06	F42	G42	H42	I42	J42	K42	L42
Right Leg Bone Marrow	42	50.05	47.2195	1.06	F43	G43	H43	I43	J43	K43	L43
Left Arm Bone Marrow	43	17.2	16.2274	1.06	F44	G44	H44	I44	J44	K44	L44
Right Arm Bone Marrow	44	17.2	16.2274	1.06	F45	G45	H45	I45	J45	K45	L45
Uterus	45	65.4	66.268	0.9869	F46	G46	H46	I46	J46	K46	L46
Left Ovary	46	4.134	4.18879	0.9869	F47	G47	H47	I47	J47	K47	L47
Right Ovary	47	4.134	4.18879	0.9869	F48	G48	H48	I48	J48	K48	L48
Left Breast	48	191.3	193.806	0.9869	F49	G49	H49	I49	J49	K49	L49
Right Breast	49	191.8	194.396	0.9869	F50	G50	H50	I50	J50	K50	L50
Total Active Bone Marrow											
AVERAGE(G2:G6,G11:G16,G27:G38,G29:G30,G41:G46)						H3/G53		SQRT(H2*H5*H15*H19*H29*H39*H49*H59*H69*H79*H89*H99*H109*H119*H129*H139*H149*H159*H169*H179*H189*H199*H209*H219*H229*H239*H249*H259*H269*H279*H289*H299*H309*H319*H329*H339*H349*H359*H369*H379*H389*H399*H409*H419*H429*H439*H449*H459*H469*H479*H489*H499*H509*H519*H529*H539*H549*H559*H569*H579*H589*H599*H609*H619*H629*H639*H649*H659*H669*H679*H689*H699*H709*H719*H729*H739*H749*H759*H769*H779*H789*H799*H809*H819*H829*H839*H849*H859*H869*H879*H889*H899*H909*H919*H929*H939*H949*H959*H969*H979*H989*H999*H1009*H1019*H1029*H1039*H1049*H1059*H1069*H1079*H1089*H1099*H1109*H1119*H1129*H1139*H1149*H1159*H1169*H1179*H1189*H1199*H1209*H1219*H1229*H1239*H1249*H1259*H1269*H1279*H1289*H1299*H1309*H1319*H1329*H1339*H1349*H1359*H1369*H1379*H1389*H1399*H1409*H1419*H1429*H1439*H1449*H1459*H1469*H1479*H1489*H1499*H1509*H1519*H1529*H1539*H1549*H1559*H1569*H1579*H1589*H1599*H1609*H1619*H1629*H1639*H1649*H1659*H1669*H1679*H1689*H1699*H1709*H1719*H1729*H1739*H1749*H1759*H1769*H1779*H1789*H1799*H1809*H1819*H1829*H1839*H1849*H1859*H1869*H1879*H1889*H1899*H1909*H1919*H1929*H1939*H1949*H1959*H1969*H1979*H1989*H1999*H2009*H2019*H2029*H2039*H2049*H2059*H2069*H2079*H2089*H2099*H2109*H2119*H2129*H2139*H2149*H2159*H2169*H2179*H2189*H2199*H2209*H2219*H2229*H2239*H2249*H2259*H2269*H2279*H2289*H2299*H2309*H2319*H2329*H2339*H2349*H2359*H2369*H2379*H2389*H2399*H2409*H2419*H2429*H2439*H2449*H2459*H2469*H2479*H2489*H2499*H2509*H2519*H2529*H2539*H2549*H2559*H2569*H2579*H2589*H2599*H2609*H2619*H2629*H2639*H2649*H2659*H2669*H2679*H2689*H2699*H2709*H2719*H2729*H2739*H2749*H2759*H2769*H2779*H2789*H2799*H2809*H2819*H2829*H2839*H2849*H2859*H2869*H2879*H2889*H2899*H2909*H2919*H2929*H2939*H2949*H2959*H2969*H2979*H2989*H2999*H3009*H3019*H3029*H3039*H3049*H3059*H3069*H3079*H3089*H3099*H3109*H3119*H3129*H3139*H3149*H3159*H3169*H3179*H3189*H3199*H3209*H3219*H3229*H3239*H3249*H3259*H3269*H3279*H3289*H3299*H3309*H3319*H3329*H3339*H3349*H3359*H3369*H3379*H3389*H3399*H3409*H3419*H3429*H3439*H3449*H3459*H3469*H3479*H3489*H3499*H3509*H3519*H3529*H3539*H3549*H3559*H3569*H3579*H3589*H3599*H3609*H3619*H3629*H3639*H3649*H3659*H3669*H3679*H3689*H3699*H3709*H3719*H3729*H3739*H3749*H3759*H3769*H3779*H3789*H3799*H3809*H3819*H3829*H3839*H3849*H3859*H3869*H3879*H3889*H3899*H3909*H3919*H3929*H3939*H3949*H3959*H3969*H3979*H3989*H3999*H4009*H4019*H4029*H4039*H4049*H4059*H4069*H4079*H4089*H4099*H4109*H4119*H4129*H4139*H4149*H4159*H4169*H4179*H4189*H4199*H4209*H4219*H4229*H4239*H4249*H4259*H4269*H4279*H4289*H4299*H4309*H4319*H4329*H4339*H4349*H4359*H4369*H4379*H4389*H4399*H4409*H4419*H4429*H4439*H4449*H4459*H4469*H4479*H4489*H4499*H4509*H4519*H4529*H4539*H4549*H4559*H4569*H4579*H4589*H4599*H4609*H4619*H4629*H4639*H4649*H4659*H4669*H4679*H4689*H4699*H4709*H4719*H4729*H4739*H4749*H4759*H4769*H4779*H4789*H4799*H4809*H4819*H4829*H4839*H4849*H4859*H4869*H4879*H4889*H4899*H4909*H4919*H4929*H4939*H4949*H4959*H4969*H4979*H4989*H4999*H5009*H5019*H5029*H5039*H5049*H5059*H5069*H5079*H5089*H5099*H5109*H5119*H5129*H5139*H5149*H5159*H5169*H5179*H5189*H5199*H5209*H5219*H5229*H5239*H5249*H5259*H5269*H5279*H5289*H5299*H5309*H5319*H5329*H5339*H5349*H5359*H5369*H5379*H5389*H5399*H5409*H5419*H5429*H5439*H5449*H5459*H5469*H5479*H5489*H5499*H5509*H5519*H5529*H5539*H5549*H5559*H5569*H5579*H5589*H5599*H5609*H5619*H5629*H5639*H5649*H5659*H5669*H5679*H5689*H5699*H5709*H5719*H5729*H5739*H5749*H5759*H5769*H5779*H5789*H5799*H5809*H5819*H5829*H5839*H5849*H5859*H5869*H5879*H5889*H5899*H5909*H5919*H5929*H5939*H5949*H5959*H5969*H5979*H5989*H5999*H6009*H6019*H6029*H6039*H6049*H6059*H6069*H6079*H6089*H6099*H6109*H6119*H6129*H6139*H6149*H6159*H6169*H6179*H6189*H6199*H6209*H6219*H6229*H6239*H6249*H6259*H6269*H6279*H6289*H6299*H6309*H6319*H6329*H6339*H6349*H6359*H6369*H6379*H6389*H6399*H6409*H6419*H6429*H6439*H6449*H6459*H6469*H6479*H6489*H6499*H6509*H6519*H6529*H6539*H6549*H6559*H6569*H6579*H6589*H6599*H6609*H6619*H6629*H6639*H6649*H6659*H6669*H6679*H6689*H6699*H6709*H6719*H6729*H6739*H6749*H6759*H6769*H6779*H6789*H6799*H6809*H6819*H6829*H6839*H6849*H6859*H6869*H6879*H6889*H6899*H6909*H6919*H6929*H6939*H6949*H6959*H6969*H6979*H6989*H6999*H7009*H7019*H7029*H7039*H7049*H7059*H7069*H7079*H7089*H7099*H7109*H7119*H7129*H7139*H7149*H7159*H7169*H7179*H7189*H7199*H7209*H7219*H7229*H7239*H7249*H7259*H7269*H7279*H7289*H7299*H7309*H7319*H7329*H7339*H7349*H7359*H7369*H7379*H7389*H7399*H7409*H7419*H7429*H7439*H7449*H7459*H7469*H7479*H7489*H7499*H7509*H7519*H7529*H7539*H7549*H7559*H7569*H7579*H7589*H7599*H7609*H7619*H7629*H7639*H7649*H7659*H7669*H7679*H7689*H7699*H7709*H7719*H7729*H7739*H7749*H7759*H7769*H7779*H7789*H7799*H7809*H7819*H7829*H7839*H7849*H7859*H7869*H7879*H7889*H7899*H7909*H7919*H7929*H7939*H7949*H7959*H7969*H7979*H7989*H7999*H8009*H8019*H8029*H8039*H8049*H8059*H8069*H8079*H8089*H8099*H8109*H8119*H8129*H8139*H8149*H8159*H8169*H8179*H8189*H8199*H8209*H8219*H8229*H8239*H8249*H8259*H8269*H8279*H8289*H8299*H8309*H8319*H8329*H8339*H8349*H8359*H8369*H8379*H8389*H8399*H8409*H8419*H8429*H8439*H8449*H8459*H8469*H8479*H8489*H8499*H8509*H8519*H8529*H8539*H8549*H8559*H8569*H8579*H8589*H8599*H8609*H8619*H8629*H8639*H8649*H8659*H8669*H8679*H8689*H8699*H8709*H8719*H8729*H8739*H8749*H8759*H8769*H8779*H8789*H8799*H8809*H8819*H8829*H8839*H8849*H8859*H8869*H8879*H8889*H8899*H8909*H8919*H8929*H8939*H8949*H8959*H8969*H8979*H8989*H8999*H9009*H9019*H9029*H9039*H9049*H9059*H9069*H9079*H9089*H9099*H9109*H9119*H9129*H9139*H9149*H9159*H9169*H9179*H9189*H			

## Appendix A10: Summary Data Sheets for All Scenarios

Table 30: Max / Aluminum / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Aluminum							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
Average SPEs per Mission	38	Dose per Average SPE	0.0311	0.0008	2.72%	0.0215	0.0017	8.06%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.4687	0.0158	3.38%	0.4382	0.0713	16.27%
		Total SPE Dose for Mission	1.6621	0.0167	1.00%	1.2634	0.0721	5.71%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		GCR (EXCEPT H,He) Dose	0.8420	0.0302	3.59%	0.0965	0.0074	7.66%
		GCR H (Z=1) Dose	1.2782	0.0433	3.39%	0.8336	0.0489	5.87%
		GCR He (Z=2) Dose	0.2522	0.0066	2.61%	0.1565	0.0051	3.23%
		Total GCR Dose for Mission	2.3724	0.0532	2.24%	1.0866	0.0497	4.58%
TOTAL DOSE CALCULATION								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		Total Dose for Mission	4.0344	0.0558	1.38%	2.3500	0.0876	3.73%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			<b>GCR (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>SPE (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>TOTAL (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>
		Active Bone Marrow	3.932	0.333	4.212	0.401	8.143	0.522
		Brain	2.463	0.031	3.801	0.082	6.264	0.088
		Thyroid	1.719	0.190	0.346	0.144	2.065	0.238
		Lungs	3.189	0.101	2.923	0.103	6.111	0.144
		Heart	5.012	0.553	2.429	0.116	7.442	0.565
		Thymus	8.016	0.808	1.842	0.323	9.858	0.870
		Stomach	6.322	0.570	1.886	0.096	8.208	0.579
		Large Intestine	2.504	0.041	2.143	0.186	4.648	0.191
		Small Intestine	2.446	0.034	1.971	0.129	4.417	0.133
		Liver	2.614	0.032	2.392	0.124	5.005	0.128
		Kidneys	2.578	0.063	3.004	0.132	5.582	0.146
		Adrenals	2.410	0.282	5.952	0.819	8.362	0.866
		Pancreas	2.254	0.126	1.777	0.177	4.031	0.217
		Spleen	2.589	0.096	1.053	0.093	3.642	0.133
		Bladder	2.409	0.136	1.311	0.214	3.720	0.253
		Uterus	2.524	0.128	3.420	0.281	5.945	0.308
		Ovaries	4.006	0.602	3.662	0.925	7.668	1.104
		Breasts	2.038	0.047	3.844	0.120	5.883	0.129

Table 31: Min / Aluminum / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Aluminum							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0401	0.0011	2.81%	0.0305	0.0026	8.49%
		Total SPE Dose for Mission	0.1561	0.0044	2.81%	0.1186	0.0101	8.49%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.8373	0.0389	4.64%	0.1156	0.0086	7.48%
		GCR H (Z=1) Dose	1.4801	0.0499	3.37%	1.1248	0.0449	4.85%
		GCR He (Z=2) Dose	0.2852	0.0138	4.84%	0.1733	0.0056	3.23%
		Total GCR Dose for Mission	2.6026	0.0647	2.49%	1.4138	0.0460	3.79%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.7587	0.0649	2.35%	1.5324	0.0471	3.08%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	2.344	0.144	0.352	0.040	2.696	0.150
		Brain	0.955	0.024	0.298	0.007	1.253	0.025
		Thyroid	0.940	0.166	0.035	0.015	0.975	0.166
		Lungs	0.624	0.016	0.229	0.009	0.853	0.019
		Heart	0.790	0.032	0.195	0.011	0.986	0.034
		Thymus	0.552	0.068	0.159	0.032	0.711	0.075
		Stomach	0.880	0.035	0.131	0.008	1.011	0.036
		Large Intestine	1.143	0.035	0.174	0.019	1.317	0.040
		Small Intestine	0.862	0.021	0.133	0.005	0.995	0.021
		Liver	1.772	0.055	0.184	0.012	1.956	0.056
		Kidneys	0.869	0.036	0.259	0.013	1.128	0.038
		Adrenals	0.516	0.035	0.604	0.083	1.119	0.090
		Pancreas	0.687	0.076	0.165	0.018	0.852	0.078
		Spleen	0.854	0.043	0.073	0.009	0.927	0.044
		Bladder	0.728	0.079	0.092	0.020	0.820	0.081
		Uterus	0.728	0.079	0.258	0.027	0.986	0.083
		Ovaries	1.029	0.343	0.269	0.090	1.298	0.355
		Breasts	7.064	0.284	0.280	0.011	7.344	0.284

Table 32: Max / Aluminum / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Aluminum							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0311	0.0008	2.72%	0.0215	0.0017	8.06%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.4687	0.0158	3.38%	0.4382	0.0713	16.27%
		Total SPE Dose for Mission	1.6621	0.0167	1.00%	1.2634	0.0721	5.71%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.8420	0.0302	3.59%	0.0965	0.0074	7.66%
		GCR H (Z=1) Dose	1.2782	0.0433	3.39%	0.8336	0.0489	5.87%
		GCR He (Z=2) Dose	0.2522	0.0066	2.61%	0.1565	0.0051	3.23%
		Total GCR Dose for Mission	2.3724	0.0532	2.24%	1.0866	0.0497	4.58%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	4.0344	0.0558	1.38%	2.3500	0.0876	3.73%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	2.449	0.096	2.965	0.244	5.413	0.262
		Brain	1.424	0.109	0.603	0.058	2.027	0.124
		Thyroid	0.347	0.023	1.499	0.464	1.846	0.465
		Lungs	0.704	0.024	2.370	0.163	3.074	0.165
		Heart	0.694	0.031	0.480	0.079	1.174	0.084
		Thymus	2.370	0.313	0.000	0.000	2.370	0.313
		Stomach	0.710	0.025	4.946	0.404	5.656	0.405
		Large Intestine	0.900	0.026	2.952	0.246	3.852	0.248
		Small Intestine	1.055	0.055	3.224	0.249	4.280	0.255
		Liver	1.718	0.051	2.200	0.539	3.918	0.542
		Kidneys	0.687	0.032	1.782	0.182	2.468	0.185
		Adrenals	0.606	0.046	0.842	0.268	1.448	0.271
		Pancreas	10.819	0.857	0.643	0.133	11.462	0.867
		Spleen	5.794	0.340	4.960	0.805	10.754	0.873
		Bladder	0.422	0.016	0.861	0.171	1.284	0.172
		Testes	0.703	0.089	0.460	0.137	1.163	0.163

Table 33: Min / Aluminum / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Aluminum							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0321	0.0009	2.71%	0.0215	0.0020	9.33%
		Total SPE Dose for Mission	0.1249	0.0034	2.71%	0.0837	0.0078	9.33%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.5828	0.0447	7.67%	0.1735	0.0522	30.09%
		GCR H (Z=1) Dose	1.3797	0.0736	5.33%	1.1166	0.1932	17.30%
		GCR He (Z=2) Dose	0.2773	0.0150	5.42%	0.1695	0.0079	4.65%
		Total GCR Dose for Mission	2.2398	0.0874	3.90%	1.4596	0.2003	13.72%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.3647	0.0875	3.70%	1.5433	0.2004	12.99%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.561	0.086	0.211	0.023	1.773	0.089
		Brain	7.691	0.224	0.012	0.005	7.704	0.224
		Thyroid	1.772	0.364	0.000	0.000	1.772	0.364
		Lungs	2.542	0.061	0.189	0.016	2.731	0.063
		Heart	0.531	0.020	0.000	0.000	0.531	0.020
		Thymus	1.690	0.526	0.000	0.000	1.690	0.526
		Stomach	0.591	0.023	0.484	0.041	1.075	0.047
		Large Intestine	0.691	0.023	0.180	0.024	0.871	0.033
		Small Intestine	0.667	0.019	0.236	0.022	0.903	0.029
		Liver	0.772	0.019	0.192	0.055	0.964	0.058
		Kidneys	0.632	0.030	0.062	0.012	0.693	0.032
		Adrenals	3.221	0.564	0.438	0.081	3.658	0.570
		Pancreas	0.441	0.037	0.000	0.000	0.441	0.037
		Spleen	0.358	0.020	0.438	0.081	0.795	0.084
		Bladder	0.520	0.063	0.047	0.028	0.567	0.069
		Testes	0.782	0.048	0.094	0.014	0.876	0.050

Table 34: Max / Aluminum / Water Summary Data Sheet

SCENARIO DEFINITION					
Mission Type	Mars Flyby				
Mission Duration (days)	700				
Solar Cycle	Maximum				
Shielding Type	Aluminum				
Crew Gender	N/A (H2O)				
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION					
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission					
			H (Sv)	$\sigma$ (Sv)	$\sigma$ (%)
Average SPEs per Mission	38	Dose Equivalent (H) per Average SPE	0.0449	0.0010	2.16%
Worst-Case SPEs per Mission	1	Dose Equivalent (H) per Worst-Case SPE	0.5100	0.0079	1.55%
		Total SPE Dose Equivalent (H) for Mission	2.2320	0.0099	0.44%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION					
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency					
			H (Sv)	$\sigma$ (Sv)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose Equivalent (H)	0.8685	0.0095	1.10%
		GCR H (Z=1) Dose Equivalent (H)	1.3003	0.0084	0.65%
		GCR He (Z=2) Dose Equivalent (H)	0.2377	0.0011	0.47%
		Total GCR Dose Equivalent (H) for Mission	2.4065	0.0128	0.53%
		Total Dose Equivalent (H) for Mission	4.6385	0.0162	0.35%

Table 35: Min / Aluminum / Water Summary Data Sheet

SCENARIO DEFINITION					
Mission Type	Mars Flyby				
Mission Duration (days)	700				
Solar Cycle	Minimum				
Shielding Type	Aluminum				
Crew Gender	N/A (Water)				
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION					
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission					
			H (Sv)	$\sigma$ (Sv)	$\sigma$ (%)
Average SPEs per Mission	4	Dose Equivalent (H) per Average SPE	0.0449	0.0010	2.16%
		Total SPE Dose Equivalent (H) for Mission	0.1746	0.0038	2.16%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION					
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency					
			H (Sv)	$\sigma$ (Sv)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose Equivalent (H)	0.7155	0.0070	0.98%
		GCR H (Z=1) Dose Equivalent (H)	1.3487	0.0086	0.64%
		GCR He (Z=2) Dose Equivalent (H)	0.2419	0.0011	0.47%
		Total GCR Dose Equivalent (H) for Mission	2.3061	0.0112	0.49%
		Total Dose Equivalent (H) for Mission	2.4807	0.0118	0.48%

Table 36: Max / Water / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Water							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
		E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)	
Average SPEs per Mission	38	Dose per Average SPE	0.0044	0.0002	4.75%	0.0045	0.0009	19.05%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0322	0.0017	5.28%	0.0246	0.0019	7.54%
		Total SPE Dose for Mission	0.1997	0.0021	1.07%	0.1982	0.0057	2.85%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
		E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)	
		GCR (EXCEPT H,He) Dose	0.3787	0.0222	5.86%	0.1088	0.0030	2.77%
		GCR H (Z=1) Dose	1.8770	0.0903	4.81%	1.1239	0.0280	2.49%
		GCR He (Z=2) Dose	0.3409	0.0216	6.33%	0.2114	0.0116	5.49%
		Total GCR Dose for Mission	2.5966	0.0954	3.68%	1.4441	0.0305	2.11%
TOTAL DOSE CALCULATION								
		E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)	
		Total Effective Dose (E) for Mission	2.7963	0.0954	3.41%	1.6423	0.0310	1.89%
ORGAN DOSE EQUIVALENT CALCULATIONS								
		GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)	
		Active Bone Marrow	1.388	0.077	0.339	0.033	1.727	0.083
		Brain	1.997	0.050	0.304	0.021	2.301	0.054
		Thyroid	1.115	0.245	1.126	0.211	2.241	0.323
		Lungs	0.821	0.020	0.240	0.015	1.061	0.025
		Heart	1.279	0.041	0.309	0.036	1.588	0.055
		Thymus	0.867	0.129	0.001	0.001	0.868	0.129
		Stomach	1.008	0.040	0.210	0.033	1.218	0.051
		Large Intestine	2.706	0.108	0.042	0.010	2.748	0.108
		Small Intestine	0.967	0.021	0.089	0.012	1.055	0.024
		Liver	0.956	0.017	0.113	0.009	1.069	0.020
		Kidneys	0.977	0.042	0.183	0.075	1.160	0.086
		Adrenals	2.057	0.274	0.001	0.001	2.058	0.274
		Pancreas	0.735	0.047	0.001	0.001	0.736	0.047
		Spleen	1.356	0.065	0.144	0.051	1.500	0.083
		Bladder	1.985	0.249	0.348	0.132	2.333	0.282
		Uterus	1.026	0.084	0.006	0.002	1.032	0.084
		Ovaries	0.808	0.099	0.000	0.000	0.808	0.099
		Breasts	0.902	0.032	0.839	0.063	1.741	0.071

Table 37: Min / Water / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Water						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0044	0.0002	4.75%	0.0045	0.0009
		Total SPE Dose for Mission	0.0170	0.0008	4.75%	0.0176	0.0034
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.3012	0.0233	7.73%	0.1421	0.0091
		GCR H (Z=1) Dose	1.6481	0.1017	6.17%	1.1207	0.0278
		GCR He (Z=2) Dose	0.3860	0.0117	3.04%	0.2338	0.0057
		Total GCR Dose for Mission	2.3353	0.1050	4.50%	1.4966	0.0298
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	2.3522	0.1050	4.47%	1.5142	0.0300
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.326	0.037	0.027	0.003	1.354
		Brain	1.404	0.072	0.018	0.001	1.422
		Thyroid	1.295	0.258	0.114	0.021	1.409
		Lungs	0.858	0.015	0.017	0.001	0.875
		Heart	1.132	0.043	0.029	0.004	1.161
		Thymus	0.987	0.141	0.000	0.000	0.987
		Stomach	1.097	0.044	0.021	0.003	1.118
		Large Intestine	1.193	0.029	0.003	0.001	1.196
		Small Intestine	1.054	0.022	0.008	0.001	1.062
		Liver	1.327	0.023	0.009	0.001	1.337
		Kidneys	1.133	0.042	0.018	0.008	1.151
		Adrenals	0.702	0.043	0.000	0.000	0.702
		Pancreas	2.054	0.137	0.000	0.000	2.054
		Spleen	0.927	0.038	0.014	0.005	0.941
		Bladder	1.279	0.102	0.035	0.013	1.315
		Uterus	1.173	0.080	0.000	0.000	1.173
		Ovaries	0.993	0.225	0.000	0.000	0.993
		Breasts	1.703	0.062	0.066	0.006	1.768

Table 38: Max / Water / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Water							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0043	0.0002	5.30%	0.0029	0.0005	16.81%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0234	0.0019	8.24%	0.0387	0.0152	39.43%
		Total SPE Dose for Mission	0.1870	0.0024	1.27%	0.1485	0.0155	10.46%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1262	0.0087	6.89%	0.0770	0.0054	6.96%
		GCR H (Z=1) Dose	1.6640	0.0758	4.55%	0.9876	0.0551	5.57%
		GCR He (Z=2) Dose	0.3424	0.0132	3.87%	0.2097	0.0061	2.93%
		Total GCR Dose for Mission	2.1326	0.0774	3.63%	1.2743	0.0557	4.37%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.3195	0.0774	3.34%	1.4228	0.0578	4.06%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.932	0.028	0.278	0.031	1.210	0.042
		Brain	0.935	0.017	0.533	0.037	1.469	0.041
		Thyroid	0.582	0.035	0.394	0.110	0.977	0.115
		Lungs	0.729	0.014	0.155	0.019	0.883	0.023
		Heart	1.009	0.042	0.046	0.010	1.054	0.043
		Thymus	0.784	0.111	0.000	0.000	0.785	0.111
		Stomach	0.930	0.038	0.014	0.002	0.944	0.038
		Large Intestine	1.052	0.030	0.031	0.011	1.083	0.032
		Small Intestine	1.016	0.022	0.224	0.120	1.240	0.122
		Liver	0.859	0.016	0.180	0.015	1.039	0.022
		Kidneys	0.964	0.044	0.209	0.083	1.173	0.094
		Adrenals	0.668	0.070	2.560	0.576	3.228	0.580
		Pancreas	1.220	0.117	0.018	0.007	1.238	0.117
		Spleen	0.829	0.048	0.021	0.004	0.850	0.048
		Bladder	0.897	0.098	0.109	0.108	1.006	0.145
		Testes	0.886	0.110	0.456	0.104	1.342	0.152

Table 39: Min / Water / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Water							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
Average SPEs per Mission	4	Dose per Average SPE	0.0052	0.0003	5.03%	0.0029	0.0009	31.33%
		Total SPE Dose for Mission	0.0202	0.0010	5.03%	0.0111	0.0035	31.33%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		GCR (EXCEPT H,He) Dose	0.2293	0.0076	3.30%	0.1398	0.0202	14.42%
		GCR H (Z=1) Dose	1.7174	0.0531	3.09%	1.1317	0.0355	3.13%
		GCR He (Z=2) Dose	0.3606	0.0165	4.57%	0.2200	0.0064	2.91%
		Total GCR Dose for Mission	2.3073	0.0561	2.43%	1.4915	0.0413	2.77%
TOTAL DOSE CALCULATION								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		Total Dose for Mission	2.3274	0.0561	2.41%	1.5026	0.0414	2.76%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			<b>GCR (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>SPE (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>TOTAL (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>
		Active Bone Marrow	1.169	0.034	0.024	0.003	1.193	0.034
		Brain	1.147	0.020	0.044	0.003	1.191	0.020
		Thyroid	0.775	0.037	0.002	0.002	0.777	0.037
		Lungs	0.888	0.018	0.011	0.002	0.899	0.018
		Heart	1.161	0.039	0.001	0.001	1.162	0.039
		Thymus	0.801	0.083	0.000	0.000	0.801	0.083
		Stomach	1.130	0.036	0.001	0.000	1.132	0.036
		Large Intestine	1.091	0.028	0.001	0.001	1.093	0.028
		Small Intestine	1.084	0.021	0.022	0.012	1.106	0.025
		Liver	1.147	0.019	0.014	0.002	1.162	0.019
		Kidneys	1.697	0.074	0.019	0.008	1.715	0.074
		Adrenals	1.055	0.182	0.002	0.000	1.057	0.182
		Pancreas	1.311	0.111	0.002	0.001	1.313	0.111
		Spleen	0.830	0.035	0.002	0.000	0.832	0.035
		Bladder	0.944	0.173	0.046	0.021	0.991	0.174
		Testes	1.095	0.102	0.092	0.011	1.187	0.102

Table 40: Max / Toroid 0 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (0 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0050	0.0002	3.45%	0.0036	0.0003	7.79%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0759	0.0039	5.12%	0.0517	0.0050	9.62%
		Total SPE Dose for Mission	0.2677	0.0040	1.51%	0.1882	0.0053	2.79%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2278	0.0116	5.11%	0.0985	0.0014	1.43%
		GCR H (Z=1) Dose	1.7929	0.1068	5.96%	1.0662	0.0676	6.34%
		GCR He (Z=2) Dose	0.3460	0.0163	4.70%	0.2184	0.0131	5.98%
		Total GCR Dose for Mission	2.3667	0.1087	4.59%	1.3832	0.0689	4.98%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.6344	0.1088	4.13%	1.5714	0.0691	4.40%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.736	0.068	0.620	0.042	2.356	0.080
		Brain	2.088	0.052	0.668	0.028	2.756	0.059
		Thyroid	0.953	0.122	0.008	0.008	0.961	0.123
		Lungs	1.101	0.020	0.320	0.015	1.421	0.025
		Heart	0.872	0.031	0.082	0.025	0.954	0.040
		Thymus	0.953	0.110	0.001	0.001	0.953	0.110
		Stomach	0.939	0.031	0.129	0.027	1.068	0.041
		Large Intestine	0.988	0.023	0.089	0.017	1.077	0.029
		Small Intestine	1.130	0.022	0.481	0.137	1.611	0.139
		Liver	0.934	0.018	0.288	0.058	1.222	0.061
		Kidneys	0.909	0.040	0.310	0.033	1.218	0.052
		Adrenals	0.924	0.089	0.997	0.227	1.921	0.244
		Pancreas	1.236	0.126	0.011	0.009	1.247	0.127
		Spleen	0.949	0.045	0.109	0.061	1.058	0.076
		Bladder	0.814	0.037	0.196	0.108	1.010	0.114
		Uterus	1.660	0.115	0.083	0.056	1.742	0.128
		Ovaries	0.505	0.039	0.001	0.001	0.505	0.039
		Breasts	1.000	0.032	1.377	0.112	2.378	0.116

Table 41: Min / Toroid 0 T / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Toroid (0 T)						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0050	0.0003	5.75%	0.0036	0.0003
		Total SPE Dose for Mission	0.0194	0.0011	5.75%	0.0138	0.0011
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.3154	0.0201	6.37%	0.1966	0.0181
		GCR H (Z=1) Dose	1.9185	0.1109	5.78%	1.1631	0.0405
		GCR He (Z=2) Dose	0.3771	0.0203	5.38%	0.2239	0.0076
		Total GCR Dose for Mission	2.6110	0.1146	4.39%	1.5836	0.0450
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	2.6304	0.1146	4.36%	1.5974	0.0450
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.489	0.043	0.041	0.004	1.530
		Brain	1.392	0.031	0.056	0.003	1.448
		Thyroid	1.024	0.079	0.001	0.001	1.025
		Lungs	1.100	0.022	0.018	0.001	1.118
		Heart	1.579	0.052	0.007	0.002	1.586
		Thymus	1.187	0.160	0.000	0.000	1.187
		Stomach	1.376	0.045	0.013	0.003	1.389
		Large Intestine	1.269	0.026	0.007	0.002	1.276
		Small Intestine	1.280	0.023	0.047	0.014	1.327
		Liver	1.253	0.019	0.024	0.006	1.276
		Kidneys	1.415	0.049	0.018	0.003	1.433
		Adrenals	0.681	0.044	0.101	0.023	0.781
		Pancreas	1.299	0.112	0.000	0.000	1.299
		Spleen	1.134	0.047	0.011	0.006	1.145
		Bladder	0.945	0.054	0.011	0.011	0.956
		Uterus	1.526	0.122	0.006	0.005	1.532
		Ovaries	0.874	0.076	0.000	0.000	0.874
		Breasts	1.403	0.046	0.096	0.010	1.499

Table 42: Max / Toroid 0 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (0 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0053	0.0003	4.74%	0.0047	0.0012	25.53%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0372	0.0023	6.21%	0.0288	0.0063	21.69%
		Total SPE Dose for Mission	0.2400	0.0028	1.16%	0.2079	0.0097	4.65%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.3420	0.0182	5.31%	0.1381	0.0182	13.20%
		GCR H (Z=1) Dose	1.8659	0.1326	7.10%	1.1656	0.1435	12.31%
		GCR He (Z=2) Dose	0.3659	0.0240	6.56%	0.2176	0.0122	5.63%
		Total GCR Dose for Mission	2.5738	0.1360	5.28%	1.5212	0.1452	9.54%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.8138	0.1360	4.83%	1.7291	0.1455	8.41%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.557	0.070	0.800	0.087	2.358	0.112
		Brain	3.261	0.237	0.689	0.047	3.950	0.241
		Thyroid	0.960	0.068	0.001	0.001	0.961	0.068
		Lungs	0.915	0.023	0.132	0.031	1.047	0.038
		Heart	1.051	0.036	0.276	0.050	1.327	0.062
		Thymus	0.948	0.072	0.364	0.276	1.312	0.286
		Stomach	1.212	0.041	0.153	0.036	1.365	0.055
		Large Intestine	1.063	0.027	0.106	0.019	1.169	0.033
		Small Intestine	1.012	0.022	0.285	0.100	1.296	0.103
		Liver	1.240	0.027	0.236	0.033	1.476	0.043
		Kidneys	0.982	0.036	0.255	0.083	1.237	0.090
		Adrenals	1.010	0.161	0.001	0.001	1.010	0.161
		Pancreas	0.913	0.088	0.046	0.032	0.958	0.094
		Spleen	1.319	0.063	0.780	0.131	2.098	0.146
		Bladder	0.707	0.045	0.698	0.256	1.406	0.260
		Testes	2.871	0.263	0.797	0.246	3.668	0.360

Table 43: Min / Toroid 0 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (0 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0042	0.0002	5.51%	0.0047	0.0017	37.07%
		Total SPE Dose for Mission	0.0165	0.0009	5.51%	0.0181	0.0067	37.07%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.3495	0.0280	8.00%	0.1984	0.0306	15.44%
		GCR H (Z=1) Dose	1.8883	0.1143	6.06%	1.1817	0.0782	6.62%
		GCR He (Z=2) Dose	0.3925	0.0248	6.31%	0.2319	0.0105	4.51%
		Total GCR Dose for Mission	2.6303	0.1203	4.57%	1.6120	0.0847	5.25%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.6467	0.1203	4.54%	1.6302	0.0849	5.21%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.622	0.087	0.069	0.009	1.692	0.087
		Brain	1.528	0.022	0.056	0.005	1.584	0.023
		Thyroid	0.942	0.079	0.000	0.000	0.942	0.079
		Lungs	1.277	0.046	0.010	0.003	1.287	0.046
		Heart	1.505	0.048	0.025	0.005	1.530	0.048
		Thymus	1.071	0.081	0.027	0.027	1.098	0.086
		Stomach	1.448	0.053	0.004	0.002	1.452	0.053
		Large Intestine	1.183	0.024	0.009	0.002	1.191	0.024
		Small Intestine	1.516	0.028	0.028	0.010	1.544	0.030
		Liver	1.303	0.021	0.019	0.002	1.322	0.021
		Kidneys	1.406	0.048	0.023	0.008	1.429	0.049
		Adrenals	1.790	0.236	0.071	0.013	1.861	0.237
		Pancreas	1.046	0.062	0.001	0.001	1.047	0.062
		Spleen	1.330	0.058	0.071	0.013	1.400	0.059
		Bladder	1.033	0.110	0.081	0.050	1.113	0.120
		Testes	1.016	0.065	0.000	0.025	1.016	0.069

Table 44: Max / Toroid 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (1.5 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0039	0.0002	5.42%	0.0023	0.0004	17.11%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0428	0.0023	5.41%	0.0262	0.0040	15.28%
		Total SPE Dose for Mission	0.2295	0.0027	1.16%	0.1156	0.0047	4.07%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2902	0.0274	9.43%	0.1134	0.0025	2.16%
		GCR H (Z=1) Dose	1.4343	0.0855	5.96%	0.8530	0.0541	6.34%
		GCR He (Z=2) Dose	0.2768	0.0130	4.70%	0.1748	0.0105	5.98%
		Total GCR Dose for Mission	2.0013	0.0907	4.53%	1.1412	0.0552	4.83%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.2308	0.0907	4.07%	1.2568	0.0554	4.41%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.430	0.073	0.460	0.070	1.890	0.101
		Brain	2.493	0.207	0.439	0.034	2.931	0.210
		Thyroid	0.707	0.043	0.004	0.002	0.711	0.043
		Lungs	0.751	0.015	0.266	0.046	1.017	0.048
		Heart	0.816	0.026	0.439	0.064	1.255	0.070
		Thymus	1.020	0.134	0.000	0.000	1.020	0.134
		Stomach	0.874	0.031	0.051	0.020	0.925	0.037
		Large Intestine	1.462	0.042	0.093	0.020	1.555	0.046
		Small Intestine	1.297	0.032	0.056	0.012	1.353	0.034
		Liver	0.986	0.019	0.106	0.014	1.092	0.024
		Kidneys	0.915	0.045	0.273	0.061	1.188	0.076
		Adrenals	0.823	0.079	0.576	0.160	1.399	0.178
		Pancreas	1.172	0.110	0.287	0.114	1.460	0.158
		Spleen	0.944	0.050	0.270	0.188	1.214	0.195
		Bladder	0.657	0.031	0.007	0.003	0.664	0.031
		Uterus	0.961	0.081	0.000	0.000	0.961	0.081
		Ovaries	0.535	0.054	0.005	0.005	0.540	0.054
		Breasts	0.945	0.030	0.927	0.094	1.872	0.099

Table 45: Min / Toroid 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (1.5 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0039	0.0002	5.42%	0.0023	0.0004	17.11%
		Total SPE Dose for Mission	0.0150	0.0008	5.42%	0.0091	0.0016	17.11%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
GCR (EXCEPT H,He) Dose			0.2492	0.0159	6.37%	0.1553	0.0143	9.21%
GCR H (Z=1) Dose			1.5156	0.0876	5.78%	0.9188	0.0320	3.48%
GCR He (Z=2) Dose			0.2979	0.0160	5.38%	0.1769	0.0060	3.42%
Total GCR Dose for Mission			2.0627	0.0905	4.39%	1.2510	0.0355	2.84%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Total Dose for Mission			2.0777	0.0905	4.36%	1.2601	0.0356	2.82%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
Active Bone Marrow			1.176	0.034	0.034	0.007	1.210	0.034
Brain			1.100	0.025	0.031	0.003	1.131	0.025
Thyroid			0.809	0.063	0.000	0.000	0.809	0.063
Lungs			0.869	0.017	0.025	0.005	0.894	0.018
Heart			1.247	0.041	0.043	0.006	1.290	0.041
Thymus			0.938	0.126	0.000	0.000	0.938	0.126
Stomach			1.087	0.036	0.004	0.002	1.091	0.036
Large Intestine			1.002	0.020	0.005	0.002	1.007	0.020
Small Intestine			1.011	0.018	0.002	0.001	1.013	0.018
Liver			0.990	0.015	0.008	0.001	0.998	0.015
Kidneys			1.118	0.039	0.026	0.006	1.144	0.039
Adrenals			0.538	0.035	0.000	0.000	0.538	0.035
Pancreas			1.026	0.089	0.025	0.011	1.052	0.089
Spleen			0.896	0.037	0.025	0.019	0.921	0.041
Bladder			0.747	0.043	0.000	0.000	0.747	0.043
Uterus			1.206	0.096	0.000	0.000	1.206	0.096
Ovaries			0.691	0.060	0.000	0.000	0.691	0.060
Breasts			1.108	0.036	0.080	0.009	1.188	0.037

Table 46: Max / Toroid 1.5 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0023	0.0012	51.99%	0.0016	0.0002	15.28%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0429	0.0203	47.36%	0.0368	0.0133	36.26%
		Total SPE Dose for Mission	0.1978	0.0216	10.92%	0.0982	0.0134	13.66%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2668	0.0425	15.94%	0.1077	0.0142	13.20%
		GCR H (Z=1) Dose	1.4554	0.1034	7.10%	0.9092	0.1119	12.31%
		GCR He (Z=2) Dose	0.1455	0.0103	7.10%	0.0909	0.0075	8.21%
		Total GCR Dose for Mission	1.8677	0.1123	6.01%	1.1078	0.1131	10.21%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.0655	0.1143	5.54%	1.2060	0.1139	9.44%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.057	0.054	0.262	0.036	1.319	0.066
		Brain	2.386	0.185	0.380	0.034	2.766	0.188
		Thyroid	0.713	0.050	0.402	0.284	1.116	0.289
		Lungs	0.535	0.014	0.372	0.040	0.907	0.043
		Heart	0.649	0.026	0.003	0.002	0.652	0.026
		Thymus	0.495	0.050	0.149	0.105	0.645	0.117
		Stomach	0.600	0.025	0.213	0.054	0.814	0.059
		Large Intestine	0.599	0.018	0.079	0.015	0.678	0.023
		Small Intestine	0.560	0.014	0.094	0.016	0.653	0.021
		Liver	0.805	0.021	0.204	0.074	1.009	0.077
		Kidneys	0.605	0.026	0.155	0.043	0.760	0.050
		Adrenals	0.859	0.160	0.000	0.000	0.859	0.160
		Pancreas	0.634	0.061	0.001	0.001	0.635	0.061
		Spleen	0.782	0.039	0.070	0.037	0.852	0.054
		Bladder	0.399	0.020	0.063	0.029	0.462	0.035
		Testes	1.907	0.192	0.277	0.092	2.184	0.213

Table 47: Min / Toroid 1.5 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0023	0.0001	5.19%	0.0016	0.0002	15.35%
		Total SPE Dose for Mission	0.0088	0.0005	5.19%	0.0062	0.0010	15.35%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2656	0.0213	8.00%	0.1508	0.0233	15.44%
		GCR H (Z=1) Dose	1.4351	0.0869	6.06%	0.8981	0.0595	6.62%
		GCR He (Z=2) Dose	0.2983	0.0188	6.31%	0.1763	0.0080	4.51%
		Total GCR Dose for Mission	1.9990	0.0914	4.57%	1.2251	0.0644	5.25%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.0078	0.0914	4.55%	1.2314	0.0644	5.23%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.233	0.066	0.017	0.003	1.250	0.066
		Brain	1.161	0.017	0.030	0.003	1.190	0.017
		Thyroid	0.716	0.060	0.041	0.029	0.757	0.067
		Lungs	0.971	0.035	0.032	0.004	1.002	0.035
		Heart	1.144	0.036	0.000	0.000	1.144	0.036
		Thymus	0.814	0.062	0.000	0.000	0.814	0.062
		Stomach	1.101	0.040	0.021	0.005	1.122	0.041
		Large Intestine	0.899	0.018	0.005	0.001	0.904	0.018
		Small Intestine	1.152	0.021	0.007	0.001	1.159	0.021
		Liver	0.990	0.016	0.018	0.007	1.008	0.018
		Kidneys	1.069	0.037	0.008	0.003	1.076	0.037
		Adrenals	1.360	0.180	0.006	0.004	1.366	0.180
		Pancreas	0.795	0.047	0.000	0.000	0.795	0.047
		Spleen	1.010	0.044	0.006	0.004	1.016	0.044
		Bladder	0.785	0.083	0.001	0.001	0.786	0.083
		Testes	0.772	0.049	0.001	0.000	0.773	0.049

**Table 48: Max / Toroid 3 T / Female Summary Data Sheet**

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (3 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0042	0.0002	5.51%	0.0031	0.0005	15.46%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0456	0.0024	5.32%	0.0309	0.0047	15.28%
		Total SPE Dose for Mission	0.2061	0.0028	1.37%	0.1512	0.0056	3.70%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1481	0.0076	5.11%	0.0641	0.0009	1.43%
		GCR H (Z=1) Dose	0.9861	0.0588	5.96%	0.5864	0.0372	6.34%
		GCR He (Z=2) Dose	0.2249	0.0106	4.70%	0.1420	0.0085	5.98%
		Total GCR Dose for Mission	1.3591	0.0602	4.43%	0.7925	0.0382	4.82%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.5652	0.0602	3.85%	0.9436	0.0386	4.09%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.101	0.044	0.757	0.103	1.859	0.112
		Brain	1.323	0.033	0.569	0.039	1.892	0.052
		Thyroid	0.604	0.079	0.009	0.007	0.613	0.080
		Lungs	0.692	0.012	0.169	0.031	0.861	0.034
		Heart	0.544	0.020	0.220	0.045	0.764	0.049
		Thymus	0.599	0.071	0.164	0.080	0.763	0.107
		Stomach	0.581	0.019	0.067	0.020	0.648	0.027
		Large Intestine	0.612	0.014	0.052	0.016	0.664	0.022
		Small Intestine	0.706	0.014	0.168	0.060	0.874	0.061
		Liver	0.583	0.011	0.166	0.022	0.748	0.024
		Kidneys	0.565	0.024	0.291	0.060	0.856	0.065
		Adrenals	0.569	0.051	0.000	0.000	0.569	0.051
		Pancreas	0.777	0.080	0.019	0.010	0.796	0.081
		Spleen	0.595	0.029	0.076	0.036	0.671	0.046
		Bladder	0.510	0.023	0.000	0.000	0.510	0.023
		Uterus	1.051	0.073	0.042	0.028	1.092	0.078
		Ovaries	0.315	0.024	0.002	0.002	0.317	0.024
		Breasts	0.621	0.020	1.097	0.104	1.717	0.106

Table 49: Min / Toroid 3 T / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Toroid (3 T)						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0042	0.0002	5.51%	0.0031	0.0005
		Total SPE Dose for Mission	0.0163	0.0009	5.51%	0.0122	0.0019
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.2082	0.0133	6.37%	0.1297	0.0119
		GCR H (Z=1) Dose	1.2470	0.0721	5.78%	0.7560	0.0263
		GCR He (Z=2) Dose	0.2451	0.0132	5.38%	0.1455	0.0050
		Total GCR Dose for Mission	1.7003	0.0745	4.38%	1.0313	0.0293
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	1.7166	0.0745	4.34%	1.0435	0.0294
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	0.973	0.028	0.063	0.010	1.036
		Brain	0.910	0.020	0.048	0.004	0.958
		Thyroid	0.671	0.052	0.000	0.000	0.671
		Lungs	0.719	0.014	0.015	0.003	0.734
		Heart	1.034	0.034	0.021	0.005	1.055
		Thymus	0.774	0.104	0.000	0.000	0.774
		Stomach	0.899	0.030	0.005	0.002	0.903
		Large Intestine	0.829	0.017	0.005	0.002	0.834
		Small Intestine	0.836	0.015	0.011	0.006	0.848
		Liver	0.819	0.012	0.013	0.002	0.832
		Kidneys	0.924	0.032	0.025	0.006	0.949
		Adrenals	0.444	0.029	0.000	0.000	0.444
		Pancreas	0.848	0.073	0.001	0.001	0.849
		Spleen	0.741	0.030	0.005	0.003	0.746
		Bladder	0.617	0.035	0.000	0.000	0.617
		Uterus	0.997	0.079	0.000	0.000	0.997
		Ovaries	0.572	0.050	0.000	0.000	0.572
		Breasts	0.917	0.030	0.085	0.010	1.002

Table 50: Max / Toroid 3 T / Male Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Maximum						
Shielding Type	Toroid (3 T)						
Crew Sex	Male (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	38	Dose per Average SPE	0.0057	0.0003	4.87%	0.0027	0.0006
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0898	0.0033	3.66%	0.0226	0.0064
		Total SPE Dose for Mission	0.2510	0.0037	1.48%	0.1245	0.0073
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.2291	0.0183	7.97%	0.0925	0.0122
		GCR H (Z=1) Dose	1.1942	0.0848	7.10%	0.7460	0.0918
		GCR He (Z=2) Dose	0.2452	0.0161	6.56%	0.1458	0.0082
		Total GCR Dose for Mission	1.6685	0.0883	5.29%	0.9843	0.0930
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	1.9195	0.0883	4.60%	1.1087	0.0933
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.035	0.047	0.449	0.070	1.484
		Brain	2.177	0.159	0.390	0.033	2.567
		Thyroid	0.631	0.045	0.001	0.001	0.632
		Lungs	0.606	0.015	0.244	0.036	0.850
		Heart	0.695	0.024	0.054	0.022	0.749
		Thymus	0.631	0.048	0.505	0.136	1.136
		Stomach	0.804	0.027	0.056	0.019	0.860
		Large Intestine	0.704	0.018	0.070	0.015	0.774
		Small Intestine	0.671	0.015	0.114	0.021	0.785
		Liver	0.822	0.018	0.188	0.025	1.010
		Kidneys	0.649	0.024	0.028	0.009	0.677
		Adrenals	0.663	0.104	3.617	0.910	4.279
		Pancreas	0.602	0.058	0.543	0.159	1.145
		Spleen	0.873	0.042	0.194	0.062	1.067
		Bladder	0.469	0.030	0.122	0.079	0.591
		Testes	1.919	0.176	1.177	0.807	3.096

Table 51: Min / Toroid 3 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (3 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0055	0.0003	5.13%	0.0027	0.0005	19.48%
		Total SPE Dose for Mission	0.0212	0.0011	5.13%	0.0103	0.0020	19.48%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
GCR (EXCEPT H,He) Dose			0.2167	0.0173	8.00%	0.1230	0.0190	15.44%
GCR H (Z=1) Dose			1.2085	0.0732	6.06%	0.7563	0.0501	6.62%
GCR He (Z=2) Dose			0.2512	0.0158	6.31%	0.1484	0.0067	4.51%
Total GCR Dose for Mission			1.6764	0.0769	4.58%	1.0277	0.0540	5.25%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Total Dose for Mission			1.6976	0.0769	4.53%	1.0381	0.0540	5.20%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
Active Bone Marrow			1.023	0.054	0.041	0.007	1.064	0.054
Brain			0.967	0.014	0.033	0.003	1.000	0.015
Thyroid			0.598	0.051	0.000	0.000	0.598	0.051
Lungs			0.805	0.029	0.019	0.002	0.824	0.029
Heart			0.951	0.030	0.003	0.002	0.954	0.030
Thymus			0.678	0.052	0.000	0.000	0.678	0.052
Stomach			0.916	0.033	0.003	0.002	0.919	0.033
Large Intestine			0.750	0.015	0.003	0.001	0.752	0.015
Small Intestine			0.958	0.018	0.009	0.002	0.966	0.018
Liver			0.826	0.014	0.014	0.002	0.839	0.014
Kidneys			0.890	0.031	0.001	0.000	0.891	0.031
Adrenals			1.138	0.151	0.019	0.006	1.156	0.151
Pancreas			0.664	0.040	0.055	0.016	0.719	0.043
Spleen			0.841	0.036	0.019	0.006	0.860	0.037
Bladder			0.653	0.069	0.022	0.030	0.675	0.076
Testes			0.642	0.041	0.000	0.015	0.642	0.043

Table 52: Max / Toroid 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0025	0.0002	7.05%	0.0017	0.0007	42.26%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0535	0.0023	4.38%	0.0522	0.0121	23.25%
		Total SPE Dose for Mission	0.1489	0.0026	1.74%	0.1181	0.0130	10.96%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1116	0.0057	5.11%	0.0483	0.0007	1.43%
		GCR H (Z=1) Dose	0.8427	0.0502	5.96%	0.5011	0.0318	6.34%
		GCR He (Z=2) Dose	0.1696	0.0080	4.70%	0.1070	0.0064	5.98%
		Total GCR Dose for Mission	1.1238	0.0512	4.55%	0.6565	0.0324	4.94%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.2728	0.0512	4.02%	0.7746	0.0349	4.51%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.845	0.033	0.677	0.148	1.522	0.152
		Brain	1.016	0.025	0.713	0.114	1.730	0.117
		Thyroid	0.464	0.060	0.000	0.000	0.464	0.060
		Lungs	0.535	0.010	0.147	0.032	0.682	0.033
		Heart	0.423	0.015	0.001	0.000	0.423	0.015
		Thymus	0.463	0.054	0.048	0.041	0.511	0.068
		Stomach	0.454	0.015	0.226	0.155	0.680	0.156
		Large Intestine	0.478	0.011	0.024	0.016	0.502	0.019
		Small Intestine	0.548	0.011	0.001	0.000	0.549	0.011
		Liver	0.453	0.008	0.193	0.050	0.646	0.050
		Kidneys	0.440	0.019	0.078	0.057	0.518	0.061
		Adrenals	0.447	0.042	0.000	0.000	0.447	0.042
		Pancreas	0.600	0.062	0.001	0.001	0.601	0.062
		Spleen	0.460	0.022	0.020	0.014	0.481	0.026
		Bladder	0.395	0.018	0.000	0.000	0.395	0.018
		Uterus	0.808	0.056	0.000	0.000	0.808	0.056
		Ovaries	0.245	0.019	0.000	0.000	0.245	0.019
		Breasts	0.484	0.015	0.544	0.193	1.028	0.194

Table 53: Min / Toroid 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0025	0.0002	7.05%	0.0017	0.0007	42.26%
		Total SPE Dose for Mission	0.0097	0.0007	7.05%	0.0067	0.0028	42.26%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1546	0.0098	6.37%	0.0963	0.0089	9.21%
		GCR H (Z=1) Dose	0.9305	0.0538	5.78%	0.5641	0.0196	3.48%
		GCR He (Z=2) Dose	0.1829	0.0098	5.38%	0.1086	0.0037	3.42%
		Total GCR Dose for Mission	1.2679	0.0556	4.38%	0.7690	0.0219	2.84%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.2776	0.0556	4.35%	0.7757	0.0220	2.84%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.725	0.021	0.031	0.012	0.755	0.024
		Brain	0.678	0.015	0.042	0.010	0.720	0.018
		Thyroid	0.499	0.039	0.000	0.000	0.499	0.039
		Lungs	0.535	0.010	0.001	0.000	0.536	0.010
		Heart	0.770	0.025	0.000	0.000	0.770	0.025
		Thymus	0.577	0.077	0.005	0.004	0.582	0.078
		Stomach	0.670	0.022	0.023	0.016	0.693	0.027
		Large Intestine	0.618	0.012	0.000	0.000	0.618	0.012
		Small Intestine	0.623	0.011	0.000	0.000	0.623	0.011
		Liver	0.610	0.009	0.007	0.004	0.617	0.010
		Kidneys	0.688	0.024	0.000	0.000	0.688	0.024
		Adrenals	0.331	0.022	0.000	0.000	0.331	0.022
		Pancreas	0.632	0.055	0.000	0.000	0.632	0.055
		Spleen	0.552	0.023	0.002	0.001	0.554	0.023
		Bladder	0.460	0.026	0.000	0.000	0.460	0.026
		Uterus	0.742	0.059	0.000	0.000	0.742	0.059
		Ovaries	0.426	0.037	0.000	0.000	0.426	0.037
		Breasts	0.683	0.022	0.053	0.019	0.736	0.030

Table 54: Max / Toroid 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Toroid (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0005	0.0000	3.17%	0.0002	0.0004	163.46%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.2024	0.0049	2.43%	0.3088	0.0730	23.64%
		Total SPE Dose for Mission	0.2213	0.0049	2.22%	0.3182	0.0730	22.96%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1676	0.0267	15.94%	0.0677	0.0089	13.20%
		GCR H (Z=1) Dose	0.8956	0.0636	7.10%	0.5595	0.0689	12.31%
		GCR He (Z=2) Dose	0.1756	0.0115	6.56%	0.1044	0.0059	5.63%
		Total GCR Dose for Mission	1.2389	0.0700	5.65%	0.7316	0.0697	9.53%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.4602	0.0701	4.80%	1.0498	0.1010	9.62%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.756	0.034	0.181	0.113	0.937	0.118
		Brain	1.591	0.116	0.124	0.030	1.715	0.120
		Thyroid	0.462	0.033	0.540	1.741	1.002	1.741
		Lungs	0.441	0.011	0.093	0.027	0.534	0.030
		Heart	0.507	0.017	0.003	0.002	0.510	0.018
		Thymus	0.459	0.035	0.401	0.394	0.860	0.396
		Stomach	0.584	0.020	0.004	0.003	0.588	0.020
		Large Intestine	0.513	0.013	0.288	0.087	0.801	0.088
		Small Intestine	0.488	0.011	0.125	0.061	0.613	0.062
		Liver	0.600	0.013	0.097	0.024	0.696	0.027
		Kidneys	0.473	0.017	0.000	0.000	0.473	0.017
		Adrenals	0.487	0.077	0.000	0.000	0.487	0.077
		Pancreas	0.440	0.042	0.010	0.010	0.450	0.043
		Spleen	0.636	0.030	0.012	0.009	0.648	0.031
		Bladder	0.341	0.022	0.000	0.000	0.341	0.022
		Testes	1.400	0.128	0.026	0.021	1.426	0.130

Table 55: Min / Toroid 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Toroid (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0005	0.0001	15.86%	0.0002	0.0004	0.00%
		Total SPE Dose for Mission	0.0019	0.0003	15.86%	0.0009	0.0016	0.00%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1608	0.0129	8.00%	0.0913	0.0141	15.44%
		GCR H (Z=1) Dose	0.8686	0.0526	6.06%	0.5436	0.0360	6.62%
		GCR He (Z=2) Dose	0.1806	0.0114	6.31%	0.1067	0.0048	4.51%
		Total GCR Dose for Mission	1.2099	0.0553	4.57%	0.7415	0.0389	5.25%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.2118	0.0553	4.57%	0.7425	0.0390	5.25%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.746	0.040	0.011	0.011	0.758	0.041
		Brain	0.703	0.010	0.001	0.000	0.703	0.010
		Thyroid	0.433	0.036	0.000	0.000	0.433	0.036
		Lungs	0.588	0.021	0.000	0.000	0.588	0.021
		Heart	0.692	0.022	0.000	0.000	0.693	0.022
		Thymus	0.492	0.037	0.000	0.000	0.492	0.037
		Stomach	0.666	0.024	0.000	0.000	0.666	0.024
		Large Intestine	0.544	0.011	0.006	0.006	0.550	0.012
		Small Intestine	0.697	0.013	0.011	0.006	0.709	0.014
		Liver	0.599	0.010	0.000	0.000	0.599	0.010
		Kidneys	0.647	0.022	0.000	0.000	0.647	0.022
		Adrenals	0.823	0.109	0.001	0.001	0.825	0.109
		Pancreas	0.481	0.028	0.001	0.001	0.482	0.028
		Spleen	0.612	0.027	0.001	0.001	0.613	0.027
		Bladder	0.475	0.050	0.000	0.000	0.475	0.050
		Testes	0.467	0.030	0.000	0.000	0.467	0.030

Table 56: Max / Solenoid 0 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (0 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0008	0.0000	4.65%	0.0007	0.0002	30.76%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0073	0.0003	3.78%	0.0056	0.0014	24.73%
		Total SPE Dose for Mission	0.0383	0.0004	0.94%	0.0316	0.0019	5.99%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1955	0.0184	9.42%	0.1164	0.0027	2.33%
		GCR H (Z=1) Dose	1.8270	0.2425	13.27%	1.0045	0.0834	8.30%
		GCR He (Z=2) Dose	0.3957	0.0334	8.45%	0.2293	0.0149	6.51%
		Total GCR Dose for Mission	2.4181	0.2455	10.15%	1.3502	0.0848	6.28%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.4564	0.2455	9.99%	1.3819	0.0848	6.14%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.334	0.036	0.098	0.022	1.432	0.042
		Brain	1.185	0.018	0.176	0.022	1.360	0.028
		Thyroid	1.451	0.216	0.000	0.000	1.451	0.216
		Lungs	0.924	0.017	0.085	0.012	1.009	0.021
		Heart	1.144	0.034	0.000	0.000	1.144	0.034
		Thymus	0.948	0.080	0.002	0.002	0.950	0.080
		Stomach	1.029	0.031	0.005	0.004	1.034	0.031
		Large Intestine	1.094	0.023	0.002	0.002	1.096	0.023
		Small Intestine	1.049	0.021	0.002	0.001	1.051	0.021
		Liver	1.040	0.016	0.007	0.005	1.047	0.016
		Kidneys	1.154	0.040	0.027	0.010	1.181	0.041
		Adrenals	0.817	0.045	0.000	0.000	0.817	0.045
		Pancreas	1.126	0.081	0.000	0.000	1.126	0.081
		Spleen	1.084	0.048	0.095	0.041	1.179	0.063
		Bladder	1.276	0.107	0.000	0.000	1.276	0.107
		Uterus	1.418	0.217	0.000	0.000	1.418	0.217
		Ovaries	0.890	0.071	0.000	0.000	0.890	0.071
		Breasts	1.067	0.085	0.223	0.054	1.290	0.101

Table 57: Min / Solenoid 0 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (0 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	σ (Sv)	σ (%)	D (Gy)	σ (Gy)	σ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0008	0.0000	4.65%	0.0007	0.0002	30.76%
		Total SPE Dose for Mission	0.0032	0.0001	4.65%	0.0026	0.0008	30.76%
		GALACTIC COSMIC RAY (GCR) DOSE CALCULATION						
		At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency						
			E (Sv)	σ (Sv)	σ (%)	D (Gy)	σ (Gy)	σ (%)
		GCR (EXCEPT H,He) Dose	0.2025	0.0191	9.42%	0.1206	0.0140	11.63%
		GCR H (Z=1) Dose	1.8818	0.0833	4.42%	1.0347	0.0344	3.32%
		GCR He (Z=2) Dose	0.4075	0.0344	8.45%	0.2361	0.0062	2.60%
		Total GCR Dose for Mission	2.4918	0.0921	3.70%	1.3915	0.0376	2.70%
		TOTAL DOSE CALCULATION						
			E (Sv)	σ (Sv)	σ (%)	D (Gy)	σ (Gy)	σ (%)
		Total Dose for Mission	2.4950	0.0921	3.69%	1.3941	0.0376	2.70%
		ORGAN DOSE EQUIVALENT CALCULATIONS						
			GCR (mSv/d)	σ (mSv/d)	SPE (mSv/d)	σ (mSv/d)	TOTAL (mSv/d)	σ (mSv/d)
		Active Bone Marrow	2.344	0.144	0.352	0.040	2.696	0.150
		Brain	0.955	0.024	0.298	0.007	1.253	0.025
		Thyroid	0.940	0.166	0.035	0.015	0.975	0.166
		Lungs	0.624	0.016	0.229	0.009	0.853	0.019
		Heart	0.790	0.032	0.195	0.011	0.986	0.034
		Thymus	0.552	0.068	0.159	0.032	0.711	0.075
		Stomach	0.880	0.035	0.131	0.008	1.011	0.036
		Large Intestine	1.143	0.035	0.174	0.019	1.317	0.040
		Small Intestine	0.862	0.021	0.133	0.005	0.995	0.021
		Liver	1.772	0.055	0.184	0.012	1.956	0.056
		Kidneys	0.869	0.036	0.259	0.013	1.128	0.038
		Adrenals	0.516	0.035	0.604	0.083	1.119	0.090
		Pancreas	0.687	0.076	0.165	0.018	0.852	0.078
		Spleen	0.854	0.043	0.073	0.009	0.927	0.044
		Bladder	0.728	0.079	0.092	0.020	0.820	0.081
		Uterus	0.728	0.079	0.258	0.027	0.986	0.083
		Ovaries	1.029	0.343	0.269	0.090	1.298	0.355
		Breasts	7.064	0.284	0.280	0.011	7.344	0.284

Table 58: Max / Solenoid 0 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (0 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0005	0.0001	11.80%	0.0002	0.0001	39.54%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0063	0.0003	4.12%	0.0046	0.0010	21.44%
		Total SPE Dose for Mission	0.0266	0.0005	1.75%	0.0120	0.0011	9.16%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2364	0.0246	10.40%	0.1388	0.0151	10.90%
		GCR H (Z=1) Dose	1.9297	0.1552	8.05%	1.1029	0.0785	7.12%
		GCR He (Z=2) Dose	0.4444	0.0401	9.03%	0.2575	0.0104	4.03%
		Total GCR Dose for Mission	2.6106	0.1622	6.21%	1.4993	0.0806	5.38%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.6372	0.1622	6.15%	1.5113	0.0806	5.34%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.482	0.039	0.061	0.018	1.544	0.043
		Brain	1.316	0.020	0.083	0.013	1.400	0.024
		Thyroid	1.612	0.240	0.001	0.001	1.614	0.240
		Lungs	1.027	0.019	0.035	0.009	1.062	0.021
		Heart	1.271	0.038	0.001	0.000	1.272	0.038
		Thymus	1.053	0.089	0.000	0.000	1.053	0.089
		Stomach	1.143	0.035	0.006	0.005	1.149	0.035
		Large Intestine	1.215	0.026	0.007	0.006	1.222	0.027
		Small Intestine	1.165	0.024	0.007	0.005	1.172	0.024
		Liver	1.156	0.017	0.007	0.006	1.163	0.018
		Kidneys	1.282	0.044	0.094	0.021	1.375	0.049
		Adrenals	0.907	0.050	0.000	0.000	0.907	0.050
		Pancreas	1.252	0.090	0.415	0.415	1.667	0.425
		Spleen	1.204	0.053	0.002	0.001	1.206	0.053
		Bladder	1.418	0.118	0.002	0.001	1.420	0.118
		Testes	1.406	0.138	0.000	0.000	1.406	0.138

Table 59: Min / Solenoid 0 T / Male Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Solenoid (0 T)						
Crew Sex	Male (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0005	0.0001	11.80%	0.0002	0.0001
		Total SPE Dose for Mission	0.0021	0.0002	11.80%	0.0007	0.0003
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.2424	0.0252	10.40%	0.1423	0.0347
		GCR H (Z=1) Dose	2.0816	0.2374	11.41%	1.1360	0.0548
		GCR He (Z=2) Dose	0.4553	0.0494	10.85%	0.2653	0.0105
		Total GCR Dose for Mission	2.7793	0.2438	8.77%	1.5436	0.0657
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	2.7814	0.2438	8.77%	1.5444	0.0657
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.527	0.041	0.005	0.002	1.532
		Brain	1.356	0.020	0.005	0.001	1.361
		Thyroid	1.661	0.247	0.000	0.000	1.661
		Lungs	1.058	0.019	0.001	0.000	1.059
		Heart	1.309	0.039	0.000	0.000	1.309
		Thymus	1.085	0.092	0.000	0.000	1.085
		Stomach	1.177	0.036	0.001	0.001	1.178
		Large Intestine	1.252	0.027	0.001	0.001	1.252
		Small Intestine	1.200	0.024	0.001	0.000	1.201
		Liver	1.191	0.018	0.000	0.000	1.191
		Kidneys	1.320	0.046	0.000	0.000	1.320
		Adrenals	0.934	0.051	0.000	0.000	0.934
		Pancreas	1.289	0.093	0.042	0.042	1.331
		Spleen	1.240	0.055	0.000	0.000	1.240
		Bladder	1.658	0.261	0.000	0.000	1.658
		Testes	1.472	0.142	0.000	0.000	1.472

Table 60: Max / Solenoid 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (1.5 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0000	0.0000	1.00%	0.0000	0.0000	0.00%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0085	0.0002	1.85%	0.0006	0.0009	145.67%
		Total SPE Dose for Mission	0.0085	0.0002	1.84%	0.0006	0.0009	145.67%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1525	0.0144	9.42%	0.0908	0.0021	2.33%
		GCR H (Z=1) Dose	1.5529	0.2061	13.27%	0.8539	0.0709	8.30%
		GCR He (Z=2) Dose	0.1741	0.0147	8.45%	0.1009	0.0066	6.51%
		Total GCR Dose for Mission	1.8795	0.2071	11.02%	1.0456	0.0712	6.81%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.8880	0.2071	10.97%	1.0462	0.0713	6.81%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.822	0.023	0.000	0.000	0.822	0.023
		Brain	0.741	0.011	0.000	0.000	0.741	0.011
		Thyroid	0.831	0.102	0.000	0.000	0.831	0.102
		Lungs	0.574	0.010	0.101	0.097	0.674	0.098
		Heart	0.750	0.023	0.000	0.000	0.750	0.023
		Thymus	0.618	0.061	0.000	0.000	0.618	0.061
		Stomach	0.639	0.020	0.000	0.000	0.639	0.020
		Large Intestine	0.679	0.015	0.000	0.000	0.679	0.015
		Small Intestine	0.660	0.015	0.000	0.000	0.660	0.015
		Liver	0.652	0.010	0.000	0.000	0.652	0.010
		Kidneys	0.728	0.028	0.000	0.000	0.728	0.028
		Adrenals	0.497	0.028	0.000	0.000	0.497	0.028
		Pancreas	0.735	0.060	0.000	0.000	0.735	0.060
		Spleen	0.652	0.029	0.000	0.000	0.652	0.029
		Bladder	0.824	0.078	0.000	0.000	0.824	0.078
		Uterus	0.862	0.129	0.000	0.000	0.862	0.129
		Ovaries	0.563	0.057	0.000	0.000	0.563	0.057
		Breasts	0.675	0.069	0.000	0.000	0.675	0.069

Table 61: Min / Solenoid 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Solenoid (1.5 T)						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0000	0.0000	1.00%	0.0000	0.0000
		Total SPE Dose for Mission	0.0000	0.0000	1.00%	0.0000	0.0000
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.1661	0.0156	9.42%	0.0989	0.0115
		GCR H (Z=1) Dose	1.5431	0.0410	2.65%	0.8484	0.0282
		GCR He (Z=2) Dose	0.3219	0.0272	8.45%	0.1866	0.0049
		Total GCR Dose for Mission	2.0311	0.0516	2.54%	1.1339	0.0308
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	2.0311	0.0516	2.54%	1.1339	0.0308
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.105	0.029	0.000	0.000	1.105
		Brain	0.982	0.015	0.000	0.000	0.982
		Thyroid	1.197	0.176	0.000	0.000	1.197
		Lungs	0.766	0.014	0.000	0.000	0.766
		Heart	0.951	0.028	0.000	0.000	0.951
		Thymus	0.788	0.068	0.000	0.000	0.788
		Stomach	0.852	0.026	0.000	0.000	0.852
		Large Intestine	0.906	0.019	0.000	0.000	0.906
		Small Intestine	0.870	0.018	0.000	0.000	0.870
		Liver	0.863	0.013	0.000	0.000	0.863
		Kidneys	0.957	0.033	0.000	0.000	0.957
		Adrenals	0.676	0.037	0.000	0.000	0.676
		Pancreas	0.936	0.068	0.000	0.000	0.936
		Spleen	0.896	0.040	0.000	0.000	0.896
		Bladder	1.061	0.089	0.000	0.000	1.061
		Uterus	1.175	0.179	0.000	0.000	1.175
		Ovaries	0.737	0.060	0.000	0.000	0.737
		Breasts	0.885	0.071	0.000	0.000	0.885

Table 62: Max / Solenoid 1.5 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0000	0.0000	2.89%	0.0000	0.0000	20.00%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0000	0.0000	1.00%	0.0000	0.0000	0.00%
		Total SPE Dose for Mission	0.0000	0.0000	1.00%	0.0000	0.0000	32.29%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1750	0.0182	10.40%	0.1027	0.0112	10.90%
		GCR H (Z=1) Dose	1.3508	0.1087	8.05%	0.7721	0.0550	7.12%
		GCR He (Z=2) Dose	0.3289	0.0297	9.03%	0.1906	0.0077	4.03%
		Total GCR Dose for Mission	1.8546	0.1141	6.15%	1.0654	0.0566	5.31%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.8546	0.1141	6.15%	1.0654	0.0566	5.31%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.085	0.029	0.000	0.000	1.085	0.029
		Brain	0.962	0.014	0.000	0.000	0.962	0.014
		Thyroid	1.186	0.177	0.000	0.000	1.186	0.177
		Lungs	0.750	0.014	0.000	0.000	0.750	0.014
		Heart	0.926	0.027	0.000	0.000	0.926	0.027
		Thymus	0.771	0.066	0.000	0.000	0.771	0.066
		Stomach	0.835	0.025	0.000	0.000	0.835	0.025
		Large Intestine	0.888	0.019	0.000	0.000	0.888	0.019
		Small Intestine	0.852	0.017	0.000	0.000	0.852	0.017
		Liver	0.844	0.013	0.000	0.000	0.844	0.013
		Kidneys	0.936	0.032	0.000	0.000	0.936	0.032
		Adrenals	0.663	0.037	0.000	0.000	0.663	0.037
		Pancreas	0.910	0.065	0.000	0.000	0.910	0.065
		Spleen	0.881	0.039	0.000	0.000	0.881	0.039
		Bladder	1.040	0.087	0.000	0.000	1.040	0.087
		Testes	1.028	0.101	0.000	0.000	1.028	0.101

Table 63: Min / Solenoid 1.5 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0000	0.0000	2.89%	0.0000	0.0000	200.00%
		Total SPE Dose for Mission	0.0000	0.0000	2.89%	0.0000	0.0000	200.00%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1964	0.0204	10.40%	0.1153	0.0281	24.41%
		GCR H (Z=1) Dose	1.7277	0.1971	11.41%	0.9429	0.0455	4.82%
		GCR He (Z=2) Dose	0.3779	0.0410	10.85%	0.2202	0.0087	3.95%
		Total GCR Dose for Mission	2.3020	0.2023	8.79%	1.2784	0.0542	4.24%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.3020	0.2023	8.79%	1.2784	0.0542	4.24%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.258	0.033	0.000	0.000	1.258	0.033
		Brain	1.118	0.017	0.000	0.000	1.118	0.017
		Thyroid	1.370	0.204	0.000	0.000	1.370	0.204
		Lungs	0.872	0.016	0.000	0.000	0.872	0.016
		Heart	1.079	0.032	0.000	0.000	1.079	0.032
		Thymus	0.893	0.075	0.000	0.000	0.893	0.075
		Stomach	0.971	0.030	0.000	0.000	0.971	0.030
		Large Intestine	1.033	0.022	0.000	0.000	1.033	0.022
		Small Intestine	0.990	0.020	0.000	0.000	0.990	0.020
		Liver	0.982	0.015	0.000	0.000	0.982	0.015
		Kidneys	1.089	0.038	0.000	0.000	1.089	0.038
		Adrenals	0.772	0.042	0.000	0.000	0.772	0.042
		Pancreas	1.064	0.077	0.000	0.000	1.064	0.077
		Spleen	1.024	0.045	0.000	0.000	1.024	0.045
		Bladder	1.364	0.216	0.000	0.000	1.364	0.216
		Testes	1.214	0.117	0.000	0.000	1.214	0.117

Table 64: Max / Solenoid 3 T / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Maximum						
Shielding Type	Solenoid (3 T)						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	38	Dose per Average SPE	0.0000	0.0000	1.92%	0.0000	0.0000
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0017	0.0000	1.77%	0.0001	0.0001
		Total SPE Dose for Mission	0.0017	0.0000	1.73%	0.0001	0.0001
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.1270	0.0120	9.42%	0.0757	0.0018
		GCR H (Z=1) Dose	1.1875	0.1576	13.27%	0.6530	0.0542
		GCR He (Z=2) Dose	0.2572	0.0217	8.45%	0.1490	0.0097
		Total GCR Dose for Mission	1.5718	0.1596	10.15%	0.8777	0.0551
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	1.5735	0.1596	10.14%	0.8778	0.0551
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	0.867	0.023	0.001	0.000	0.868
		Brain	0.770	0.011	0.000	0.000	0.770
		Thyroid	0.943	0.140	0.000	0.000	0.943
		Lungs	0.601	0.011	0.000	0.000	0.601
		Heart	0.744	0.022	0.000	0.000	0.744
		Thymus	0.616	0.052	0.000	0.000	0.616
		Stomach	0.669	0.020	0.000	0.000	0.669
		Large Intestine	0.711	0.015	0.000	0.000	0.711
		Small Intestine	0.682	0.014	0.000	0.000	0.682
		Liver	0.676	0.010	0.000	0.000	0.676
		Kidneys	0.750	0.026	0.000	0.000	0.750
		Adrenals	0.531	0.029	0.000	0.000	0.531
		Pancreas	0.732	0.053	0.000	0.000	0.732
		Spleen	0.704	0.031	0.000	0.000	0.704
		Bladder	0.829	0.069	0.000	0.000	0.829
		Uterus	0.922	0.141	0.000	0.000	0.922
		Ovaries	0.578	0.046	0.000	0.000	0.578
		Breasts	0.694	0.055	0.020	0.019	0.714

Table 65: Min / Solenoid 3 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (3 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0002	0.0000	2.65%	0.0000	0.0000	119.66%
		Total SPE Dose for Mission	0.0007	0.0000	2.65%	0.0000	0.0000	119.66%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1316	0.0124	9.42%	0.0784	0.0091	11.63%
		GCR H (Z=1) Dose	1.2420	0.0330	2.65%	0.6829	0.0227	3.32%
		GCR He (Z=2) Dose	0.2608	0.0220	8.45%	0.1511	0.0039	2.60%
		Total GCR Dose for Mission	1.6344	0.0415	2.54%	0.9124	0.0248	2.71%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.6351	0.0415	2.54%	0.9125	0.0248	2.71%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.889	0.024	0.000	0.000	0.889	0.024
		Brain	0.790	0.012	0.000	0.000	0.790	0.012
		Thyroid	0.964	0.142	0.000	0.000	0.964	0.142
		Lungs	0.616	0.011	0.000	0.000	0.616	0.011
		Heart	0.764	0.023	0.000	0.000	0.764	0.023
		Thymus	0.633	0.054	0.000	0.000	0.633	0.054
		Stomach	0.686	0.021	0.000	0.000	0.686	0.021
		Large Intestine	0.729	0.016	0.000	0.000	0.729	0.016
		Small Intestine	0.699	0.014	0.000	0.000	0.699	0.014
		Liver	0.694	0.010	0.000	0.000	0.694	0.010
		Kidneys	0.769	0.027	0.000	0.000	0.769	0.027
		Adrenals	0.544	0.030	0.000	0.000	0.544	0.030
		Pancreas	0.753	0.055	0.000	0.000	0.753	0.055
		Spleen	0.721	0.032	0.000	0.000	0.721	0.032
		Bladder	0.851	0.071	0.000	0.000	0.851	0.071
		Uterus	0.943	0.144	0.000	0.000	0.943	0.144
		Ovaries	0.594	0.048	0.000	0.000	0.594	0.048
		Breasts	0.713	0.058	0.008	0.007	0.721	0.058

Table 66: Max / Solenoid 3 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (3 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0001	0.0000	2.34%	0.0000	0.0000	189.96%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0000	0.0000	1.22%	0.0000	0.0000	137.84%
		Total SPE Dose for Mission	0.0051	0.0000	0.38%	0.0006	0.0002	30.49%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1371	0.0143	10.40%	0.0805	0.0088	10.90%
		GCR H (Z=1) Dose	1.1964	0.0963	8.05%	0.6838	0.0487	7.12%
		GCR He (Z=2) Dose	0.2889	0.0261	9.03%	0.1674	0.0067	4.03%
		Total GCR Dose for Mission	1.6224	0.1007	6.21%	0.9317	0.0499	5.36%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.6275	0.1007	6.19%	0.9324	0.0499	5.35%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.925	0.024	0.060	0.049	0.985	0.054
		Brain	0.821	0.012	0.000	0.000	0.821	0.012
		Thyroid	1.013	0.155	0.000	0.000	1.013	0.155
		Lungs	0.642	0.012	0.000	0.000	0.642	0.012
		Heart	0.790	0.024	0.000	0.000	0.790	0.024
		Thymus	0.651	0.053	0.000	0.000	0.651	0.053
		Stomach	0.715	0.022	0.000	0.000	0.715	0.022
		Large Intestine	0.760	0.016	0.000	0.000	0.760	0.016
		Small Intestine	0.727	0.015	0.000	0.000	0.727	0.015
		Liver	0.723	0.011	0.000	0.000	0.723	0.011
		Kidneys	0.800	0.027	0.000	0.000	0.800	0.027
		Adrenals	0.570	0.031	0.000	0.000	0.570	0.031
		Pancreas	0.781	0.057	0.000	0.000	0.781	0.057
		Spleen	0.756	0.034	0.000	0.000	0.756	0.034
		Bladder	0.874	0.071	0.000	0.000	0.874	0.071
		Testes	0.878	0.086	0.000	0.000	0.878	0.086

Table 67: Min / Solenoid 3 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (3 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0001	0.0000	2.34%	0.0000	0.0000	189.96%
		Total SPE Dose for Mission	0.0005	0.0000	2.34%	0.0001	0.0001	189.96%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1527	0.0159	10.40%	0.0897	0.0219	24.41%
		GCR H (Z=1) Dose	1.3739	0.1567	11.41%	0.7498	0.0362	4.82%
		GCR He (Z=2) Dose	0.2732	0.0296	10.85%	0.1592	0.0063	3.95%
		Total GCR Dose for Mission	1.7997	0.1603	8.91%	0.9986	0.0427	4.28%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.8003	0.1603	8.90%	0.9987	0.0427	4.28%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.947	0.025	0.006	0.005	0.953	0.026
		Brain	0.843	0.013	0.000	0.000	0.843	0.013
		Thyroid	1.020	0.149	0.000	0.000	1.020	0.149
		Lungs	0.657	0.012	0.000	0.000	0.657	0.012
		Heart	0.819	0.024	0.000	0.000	0.819	0.024
		Thymus	0.677	0.058	0.000	0.000	0.677	0.058
		Stomach	0.732	0.022	0.000	0.000	0.732	0.022
		Large Intestine	0.778	0.017	0.000	0.000	0.778	0.017
		Small Intestine	0.747	0.015	0.000	0.000	0.747	0.015
		Liver	0.741	0.011	0.000	0.000	0.741	0.011
		Kidneys	0.822	0.029	0.000	0.000	0.822	0.029
		Adrenals	0.580	0.032	0.000	0.000	0.580	0.032
		Pancreas	0.808	0.059	0.000	0.000	0.808	0.059
		Spleen	0.768	0.034	0.000	0.000	0.768	0.034
		Bladder	1.031	0.162	0.000	0.000	1.031	0.162
		Testes	0.921	0.088	0.000	0.000	0.921	0.088

Table 68: Max / Solenoid 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0000	0.0000	1.00%	0.0000	0.0000	0.00%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0000	0.0000	1.00%	0.0000	0.0000	0.00%
		Total SPE Dose for Mission	0.0000	0.0000	0.16%	0.0000	0.0000	0.00%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.0860	0.0081	9.42%	0.0512	0.0012	2.33%
		GCR H (Z=1) Dose	0.6833	0.0453	6.64%	0.3757	0.0312	8.30%
		GCR He (Z=2) Dose	0.1741	0.0147	8.45%	0.1009	0.0066	6.51%
		Total GCR Dose for Mission	0.9434	0.0484	5.13%	0.5278	0.0319	6.05%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	0.9434	0.0484	5.13%	0.5278	0.0319	6.05%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.570	0.016	0.000	0.000	0.570	0.016
		Brain	0.504	0.008	0.000	0.000	0.504	0.008
		Thyroid	0.628	0.095	0.000	0.000	0.628	0.095
		Lungs	0.393	0.007	0.000	0.000	0.393	0.007
		Heart	0.482	0.014	0.000	0.000	0.482	0.014
		Thymus	0.404	0.035	0.000	0.000	0.404	0.035
		Stomach	0.436	0.013	0.000	0.000	0.436	0.013
		Large Intestine	0.465	0.010	0.000	0.000	0.465	0.010
		Small Intestine	0.446	0.009	0.000	0.000	0.446	0.009
		Liver	0.441	0.007	0.000	0.000	0.441	0.007
		Kidneys	0.489	0.017	0.000	0.000	0.489	0.017
		Adrenals	0.347	0.019	0.000	0.000	0.347	0.019
		Pancreas	0.471	0.033	0.000	0.000	0.471	0.033
		Spleen	0.462	0.021	0.000	0.000	0.462	0.021
		Bladder	0.548	0.046	0.000	0.000	0.548	0.046
		Uterus	0.614	0.095	0.000	0.000	0.614	0.095
		Ovaries	0.371	0.027	0.000	0.000	0.371	0.027
		Breasts	0.445	0.033	0.000	0.000	0.445	0.033

Table 69: Min / Solenoid 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0000	0.0000	0.00%	0.0000	0.0000	0.00%
		Total SPE Dose for Mission	0.0000	0.0000	0.00%	0.0000	0.0000	0.00%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.0810	0.0076	9.42%	0.0483	0.0056	11.63%
		GCR H (Z=1) Dose	0.6586	0.0175	2.65%	0.3621	0.0120	3.32%
		GCR He (Z=2) Dose	0.1426	0.0121	8.45%	0.0826	0.0022	2.60%
		Total GCR Dose for Mission	0.8823	0.0226	2.56%	0.4930	0.0134	2.73%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	0.8823	0.0226	2.56%	0.4930	0.0134	2.73%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.500	0.014	0.000	0.000	0.500	0.014
		Brain	0.444	0.007	0.000	0.000	0.444	0.007
		Thyroid	0.543	0.079	0.000	0.000	0.543	0.079
		Lungs	0.345	0.006	0.000	0.000	0.345	0.006
		Heart	0.429	0.013	0.000	0.000	0.429	0.013
		Thymus	0.360	0.033	0.000	0.000	0.360	0.033
		Stomach	0.384	0.012	0.000	0.000	0.384	0.012
		Large Intestine	0.409	0.009	0.000	0.000	0.409	0.009
		Small Intestine	0.393	0.008	0.000	0.000	0.393	0.008
		Liver	0.388	0.006	0.000	0.000	0.388	0.006
		Kidneys	0.432	0.015	0.000	0.000	0.432	0.015
		Adrenals	0.304	0.017	0.000	0.000	0.304	0.017
		Pancreas	0.419	0.030	0.000	0.000	0.419	0.030
		Spleen	0.403	0.018	0.000	0.000	0.403	0.018
		Bladder	0.487	0.043	0.000	0.000	0.487	0.043
		Uterus	0.538	0.083	0.000	0.000	0.538	0.083
		Ovaries	0.328	0.026	0.000	0.000	0.328	0.026
		Breasts	0.393	0.031	0.000	0.000	0.393	0.031

Table 70: Max / Solenoid 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Solenoid (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0001	0.0000	2.35%	0.0007	0.0004	52.85%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0000	0.0000	0.00%	0.0000	0.0000	0.00%
		Total SPE Dose for Mission	0.0033	0.0000	0.38%	0.0268	0.0023	8.53%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.0828	0.0086	10.40%	0.0486	0.0053	10.90%
		GCR H (Z=1) Dose	0.6561	0.0528	8.05%	0.3750	0.0267	7.12%
		GCR He (Z=2) Dose	0.1778	0.0161	9.03%	0.1030	0.0042	4.03%
		Total GCR Dose for Mission	0.9166	0.0558	6.09%	0.5266	0.0275	5.23%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	0.9199	0.0558	6.07%	0.5534	0.0276	4.99%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.555	0.015	0.004	0.004	0.559	0.015
		Brain	0.491	0.007	0.412	0.108	0.902	0.108
		Thyroid	0.614	0.095	0.000	0.000	0.614	0.095
		Lungs	0.384	0.007	0.000	0.000	0.384	0.007
		Heart	0.468	0.014	0.000	0.000	0.468	0.014
		Thymus	0.389	0.032	0.000	0.000	0.389	0.032
		Stomach	0.427	0.013	0.000	0.000	0.427	0.013
		Large Intestine	0.454	0.010	0.000	0.000	0.454	0.010
		Small Intestine	0.434	0.009	0.000	0.000	0.434	0.009
		Liver	0.431	0.006	0.000	0.000	0.431	0.006
		Kidneys	0.477	0.016	0.000	0.000	0.477	0.016
		Adrenals	0.341	0.019	0.000	0.000	0.341	0.019
		Pancreas	0.462	0.033	0.000	0.000	0.462	0.033
		Spleen	0.453	0.020	0.000	0.000	0.453	0.020
		Bladder	0.524	0.043	0.000	0.000	0.524	0.043
		Testes	0.525	0.052	0.000	0.000	0.525	0.052

Table 71: Min / Solenoid 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Solenoid (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
Average SPEs per Mission	4	Dose per Average SPE	0.0001	0.0000	2.35%	0.0007	0.0004	52.85%
		Total SPE Dose for Mission	0.0003	0.0000	2.35%	0.0027	0.0014	52.85%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		GCR (EXCEPT H,He) Dose	0.0897	0.0093	10.40%	0.0527	0.0129	24.41%
		GCR H (Z=1) Dose	0.7702	0.0879	11.41%	0.4203	0.0203	4.82%
		GCR He (Z=2) Dose	0.1730	0.0188	10.85%	0.1008	0.0040	3.95%
		Total GCR Dose for Mission	1.0329	0.0903	8.74%	0.5738	0.0243	4.24%
TOTAL DOSE CALCULATION								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		Total Dose for Mission	1.0332	0.0903	8.74%	0.5765	0.0244	4.23%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			<b>GCR (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>SPE (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>TOTAL (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>
		Active Bone Marrow	0.573	0.015	0.000	0.000	0.573	0.015
		Brain	0.508	0.008	0.042	0.011	0.550	0.013
		Thyroid	0.625	0.094	0.000	0.000	0.625	0.094
		Lungs	0.397	0.007	0.000	0.000	0.397	0.007
		Heart	0.490	0.015	0.000	0.000	0.490	0.015
		Thymus	0.406	0.034	0.000	0.000	0.406	0.034
		Stomach	0.442	0.013	0.000	0.000	0.442	0.013
		Large Intestine	0.470	0.010	0.000	0.000	0.470	0.010
		Small Intestine	0.450	0.009	0.000	0.000	0.450	0.009
		Liver	0.447	0.007	0.000	0.000	0.447	0.007
		Kidneys	0.495	0.017	0.000	0.000	0.495	0.017
		Adrenals	0.351	0.019	0.000	0.000	0.351	0.019
		Pancreas	0.483	0.035	0.000	0.000	0.483	0.035
		Spleen	0.466	0.021	0.000	0.000	0.466	0.021
		Bladder	0.621	0.098	0.000	0.000	0.621	0.098
		Testes	0.551	0.053	0.000	0.000	0.551	0.053

Table 72: Max / Race Track 0 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (0 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0017	0.0002	9.01%	0.0017	0.0008	47.22%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0591	0.0034	5.68%	0.0572	0.0194	33.95%
		Total SPE Dose for Mission	0.1247	0.0035	2.80%	0.1239	0.0201	16.20%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1982	0.0202	10.21%	0.0857	0.0012	1.43%
		GCR H (Z=1) Dose	1.5598	0.0929	5.96%	0.9276	0.0588	6.34%
		GCR He (Z=2) Dose	0.2907	0.0137	4.70%	0.1835	0.0110	5.98%
		Total GCR Dose for Mission	2.0487	0.0961	4.69%	1.1969	0.0599	5.00%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.1734	0.0962	4.42%	1.3207	0.0631	4.78%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.494	0.059	0.203	0.084	1.697	0.102
		Brain	1.800	0.045	0.942	0.161	2.742	0.167
		Thyroid	0.812	0.106	0.000	0.000	0.812	0.106
		Lungs	0.945	0.017	0.478	0.176	1.423	0.177
		Heart	0.745	0.027	0.000	0.000	0.745	0.027
		Thymus	0.813	0.093	0.000	0.000	0.813	0.093
		Stomach	0.804	0.027	0.000	0.000	0.804	0.027
		Large Intestine	0.844	0.020	0.009	0.008	0.853	0.021
		Small Intestine	0.966	0.019	0.000	0.000	0.966	0.019
		Liver	0.797	0.015	0.018	0.011	0.815	0.019
		Kidneys	0.777	0.035	0.000	0.000	0.777	0.035
		Adrenals	0.790	0.077	0.000	0.000	0.790	0.077
		Pancreas	1.050	0.107	0.000	0.000	1.050	0.107
		Spleen	0.807	0.038	0.000	0.000	0.807	0.038
		Bladder	0.694	0.031	0.000	0.000	0.694	0.031
		Uterus	1.423	0.099	0.000	0.000	1.423	0.099
		Ovaries	0.430	0.033	0.000	0.000	0.430	0.033
		Breasts	0.856	0.027	0.710	0.196	1.565	0.198

Table 73: Min / Race Track 0 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (0 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0017	0.0002	9.01%	0.0017	0.0008	47.22%
		Total SPE Dose for Mission	0.0067	0.0006	9.01%	0.0068	0.0032	47.22%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.2075	0.0212	10.21%	0.0898	0.0064	7.17%
		GCR H (Z=1) Dose	1.6066	0.0957	5.96%	0.9555	0.0242	2.54%
		GCR He (Z=2) Dose	0.3061	0.0144	4.70%	0.1932	0.0046	2.39%
		Total GCR Dose for Mission	2.1202	0.0991	4.67%	1.2385	0.0255	2.06%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.1269	0.0991	4.66%	1.2452	0.0257	2.06%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.558	0.061	0.010	0.007	1.567	0.062
		Brain	1.875	0.047	0.057	0.015	1.932	0.049
		Thyroid	0.849	0.110	0.000	0.000	0.849	0.110
		Lungs	0.985	0.018	0.045	0.018	1.030	0.025
		Heart	0.778	0.028	0.000	0.000	0.778	0.028
		Thymus	0.850	0.098	0.000	0.000	0.850	0.098
		Stomach	0.838	0.028	0.000	0.000	0.838	0.028
		Large Intestine	0.881	0.021	0.000	0.000	0.881	0.021
		Small Intestine	1.008	0.020	0.000	0.000	1.008	0.020
		Liver	0.832	0.016	0.000	0.000	0.832	0.016
		Kidneys	0.811	0.036	0.000	0.000	0.811	0.036
		Adrenals	0.824	0.080	0.000	0.000	0.824	0.080
		Pancreas	1.099	0.112	0.000	0.000	1.099	0.112
		Spleen	0.844	0.040	0.000	0.000	0.844	0.040
		Bladder	0.726	0.032	0.000	0.000	0.726	0.032
		Uterus	1.485	0.103	0.000	0.000	1.485	0.103
		Ovaries	0.450	0.035	0.000	0.000	0.450	0.035
		Breasts	0.893	0.029	0.020	0.015	0.913	0.032

Table 74: Max / Race Track 0 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (0 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0033	0.0002	6.07%	0.0039	0.0012	32.15%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0263	0.0013	5.00%	0.0330	0.0090	27.28%
		Total SPE Dose for Mission	0.1541	0.0018	1.18%	0.1817	0.0119	6.53%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1839	0.0197	10.70%	0.0824	0.0059	7.12%
		GCR H (Z=1) Dose	1.4930	0.0880	5.90%	0.9077	0.0570	6.28%
		GCR He (Z=2) Dose	0.2796	0.0131	4.68%	0.1748	0.0069	3.96%
		Total GCR Dose for Mission	1.9565	0.0911	4.66%	1.1649	0.0578	4.96%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	2.1106	0.0912	4.32%	1.3467	0.0590	4.38%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.451	0.057	0.112	0.042	1.563	0.071
		Brain	1.750	0.044	1.175	0.191	2.925	0.196
		Thyroid	0.782	0.104	0.000	0.000	0.782	0.104
		Lungs	0.916	0.016	0.185	0.034	1.100	0.038
		Heart	0.718	0.026	0.060	0.059	0.778	0.064
		Thymus	0.784	0.089	0.000	0.000	0.784	0.089
		Stomach	0.776	0.026	0.000	0.000	0.776	0.026
		Large Intestine	0.814	0.019	1.348	0.314	2.163	0.315
		Small Intestine	0.932	0.018	0.087	0.067	1.019	0.069
		Liver	0.768	0.014	0.104	0.025	0.872	0.029
		Kidneys	0.750	0.034	0.003	0.002	0.753	0.034
		Adrenals	0.762	0.076	0.000	0.000	0.762	0.076
		Pancreas	1.007	0.102	0.000	0.000	1.007	0.102
		Spleen	0.775	0.037	0.000	0.000	0.775	0.037
		Bladder	0.669	0.030	0.000	0.000	0.669	0.030
		Testes	0.414	0.032	0.000	0.000	0.414	0.032

Table 75: Min / Race Track 0 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (0 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
Average SPEs per Mission	4	Dose per Average SPE	0.0033	0.0002	6.07%	0.0039	0.0012	32.15%
		Total SPE Dose for Mission	0.0130	0.0008	6.07%	0.0151	0.0048	32.15%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		GCR (EXCEPT H,He) Dose	0.1914	0.0205	10.70%	0.0858	0.0161	18.78%
		GCR H (Z=1) Dose	1.5383	0.1021	6.64%	0.9440	0.0482	5.11%
		GCR He (Z=2) Dose	0.2791	0.0145	5.18%	0.1801	0.0088	4.89%
		Total GCR Dose for Mission	2.0088	0.1052	5.23%	1.2099	0.0516	4.27%
TOTAL DOSE CALCULATION								
			<b>E (Sv)</b>	<b><math>\sigma</math> (Sv)</b>	<b><math>\sigma</math> (%)</b>	<b>D (Gy)</b>	<b><math>\sigma</math> (Gy)</b>	<b><math>\sigma</math> (%)</b>
		Total Dose for Mission	2.0217	0.1052	5.20%	1.2250	0.0518	4.23%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			<b>GCR (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>SPE (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>	<b>TOTAL (mSv/d)</b>	<b><math>\sigma</math> (mSv/d)</b>
		Active Bone Marrow	1.508	0.060	0.003	0.002	1.511	0.060
		Brain	1.821	0.046	0.105	0.019	1.926	0.050
		Thyroid	0.810	0.108	0.000	0.000	0.810	0.108
		Lungs	0.951	0.017	0.000	0.000	0.951	0.017
		Heart	0.744	0.027	0.006	0.006	0.750	0.027
		Thymus	0.812	0.092	0.000	0.000	0.812	0.092
		Stomach	0.805	0.027	0.000	0.000	0.805	0.027
		Large Intestine	0.844	0.020	0.137	0.032	0.980	0.038
		Small Intestine	0.966	0.019	0.009	0.007	0.975	0.020
		Liver	0.796	0.015	0.000	0.000	0.796	0.015
		Kidneys	0.778	0.035	0.000	0.000	0.778	0.035
		Adrenals	0.790	0.078	0.000	0.000	0.790	0.078
		Pancreas	1.041	0.105	0.000	0.000	1.041	0.105
		Spleen	0.802	0.038	0.000	0.000	0.802	0.038
		Bladder	0.507	0.058	0.000	0.000	0.507	0.058
		Testes	0.337	0.033	0.000	0.000	0.337	0.033

Table 76: Max / Race Track 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (1.5 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0040	0.0003	7.95%	0.0037	0.0019	51.40%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0050	0.0004	7.09%	0.0072	0.0036	50.09%
		Total SPE Dose for Mission	0.1599	0.0020	1.26%	0.1481	0.0122	8.26%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1784	0.0182	10.21%	0.0772	0.0011	1.43%
		GCR H (Z=1) Dose	1.3882	0.0827	5.96%	0.8256	0.0524	6.34%
		GCR He (Z=2) Dose	0.2587	0.0122	4.70%	0.1633	0.0098	5.98%
		Total GCR Dose for Mission	1.8253	0.0856	4.69%	1.0661	0.0533	5.00%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.9852	0.0856	4.31%	1.2141	0.0547	4.50%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.338	0.053	0.184	0.070	1.522	0.087
		Brain	1.612	0.040	0.662	0.129	2.274	0.135
		Thyroid	0.725	0.095	0.000	0.000	0.725	0.095
		Lungs	0.845	0.015	0.169	0.046	1.014	0.048
		Heart	0.664	0.024	0.018	0.008	0.683	0.025
		Thymus	0.726	0.083	0.000	0.000	0.726	0.083
		Stomach	0.717	0.024	0.000	0.000	0.717	0.024
		Large Intestine	0.753	0.018	0.010	0.010	0.763	0.020
		Small Intestine	0.862	0.017	0.013	0.011	0.875	0.020
		Liver	0.711	0.013	0.000	0.000	0.711	0.013
		Kidneys	0.694	0.031	0.000	0.000	0.694	0.031
		Adrenals	0.704	0.069	0.000	0.000	0.704	0.069
		Pancreas	0.935	0.095	0.000	0.000	0.935	0.095
		Spleen	0.719	0.034	0.000	0.000	0.719	0.034
		Bladder	0.619	0.028	0.000	0.000	0.619	0.028
		Uterus	1.273	0.088	0.000	0.000	1.273	0.088
		Ovaries	0.384	0.030	0.000	0.000	0.384	0.030
		Breasts	0.764	0.024	1.479	0.500	2.242	0.501

Table 77: Min / Race Track 1.5 T / Female Summary Data Sheet

SCENARIO DEFINITION							
Mission Type	Mars Flyby						
Mission Duration (days)	700						
Solar Cycle	Minimum						
Shielding Type	Race Track (1.5 T)						
Crew Sex	Female (MIRD)						
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION							
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
Average SPEs per Mission	4	Dose per Average SPE	0.0040	0.0003	7.95%	0.0037	0.0019
		Total SPE Dose for Mission	0.0157	0.0012	7.95%	0.0143	0.0073
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION							
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		GCR (EXCEPT H,He) Dose	0.1868	0.0191	10.21%	0.0808	0.0058
		GCR H (Z=1) Dose	1.4460	0.0862	5.96%	0.8599	0.0218
		GCR He (Z=2) Dose	0.2755	0.0129	4.70%	0.1739	0.0042
		Total GCR Dose for Mission	1.9082	0.0892	4.67%	1.1146	0.0230
TOTAL DOSE CALCULATION							
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)
		Total Dose for Mission	1.9239	0.0892	4.64%	1.1289	0.0241
ORGAN DOSE EQUIVALENT CALCULATIONS							
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)
		Active Bone Marrow	1.402	0.055	0.014	0.006	1.416
		Brain	1.687	0.042	0.057	0.013	1.744
		Thyroid	0.765	0.099	0.000	0.000	0.765
		Lungs	0.887	0.016	0.017	0.005	0.904
		Heart	0.700	0.025	0.000	0.000	0.700
		Thymus	0.765	0.088	0.000	0.000	0.765
		Stomach	0.754	0.025	0.000	0.000	0.754
		Large Intestine	0.793	0.019	0.001	0.001	0.794
		Small Intestine	0.907	0.018	0.000	0.000	0.907
		Liver	0.749	0.014	0.000	0.000	0.749
		Kidneys	0.730	0.032	0.000	0.000	0.730
		Adrenals	0.742	0.072	0.000	0.000	0.742
		Pancreas	0.989	0.101	0.000	0.000	0.989
		Spleen	0.759	0.036	0.000	0.000	0.759
		Bladder	0.653	0.029	0.000	0.000	0.653
		Uterus	1.337	0.093	0.000	0.000	1.337
		Ovaries	0.405	0.031	0.000	0.000	0.405
		Breasts	0.803	0.026	0.150	0.051	0.953

**Table 78: Max / Race Track 1.5 T / Male Summary Data Sheet**

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0028	0.0001	4.76%	0.0015	0.0005	32.46%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0083	0.0007	8.41%	0.0062	0.0038	61.86%
		Total SPE Dose for Mission	0.1143	0.0011	0.94%	0.0644	0.0049	7.61%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1655	0.0177	10.70%	0.0742	0.0053	7.12%
		GCR H (Z=1) Dose	1.3138	0.0775	5.90%	0.7988	0.0502	6.28%
		GCR He (Z=2) Dose	0.2460	0.0115	4.68%	0.1538	0.0061	3.96%
		Total GCR Dose for Mission	1.7254	0.0803	4.65%	1.0268	0.0508	4.95%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.8396	0.0803	4.36%	1.0912	0.0511	4.68%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.292	0.051	0.265	0.141	1.557	0.150
		Brain	1.561	0.039	0.002	0.001	1.563	0.039
		Thyroid	0.692	0.093	0.000	0.000	0.692	0.093
		Lungs	0.813	0.015	0.371	0.091	1.185	0.092
		Heart	0.635	0.023	1.698	0.517	2.333	0.518
		Thymus	0.694	0.079	0.000	0.000	0.694	0.079
		Stomach	0.687	0.023	0.000	0.000	0.687	0.023
		Large Intestine	0.720	0.017	0.000	0.000	0.720	0.017
		Small Intestine	0.825	0.016	0.000	0.000	0.826	0.016
		Liver	0.679	0.013	0.230	0.062	0.909	0.063
		Kidneys	0.664	0.030	0.000	0.000	0.664	0.030
		Adrenals	0.673	0.067	0.000	0.000	0.673	0.067
		Pancreas	0.888	0.090	0.000	0.000	0.888	0.090
		Spleen	0.684	0.032	0.000	0.000	0.684	0.032
		Bladder	0.591	0.026	0.000	0.000	0.591	0.026
		Testes	0.366	0.028	0.000	0.000	0.366	0.028

Table 79: Min / Race Track 1.5 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (1.5 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0028	0.0001	4.76%	0.0015	0.0005	32.46%
		Total SPE Dose for Mission	0.0107	0.0005	4.76%	0.0059	0.0019	32.46%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1666	0.0178	10.70%	0.0747	0.0140	18.78%
		GCR H (Z=1) Dose	1.2614	0.0837	6.64%	0.7741	0.0396	5.11%
		GCR He (Z=2) Dose	0.2288	0.0119	5.18%	0.1476	0.0072	4.89%
		Total GCR Dose for Mission	1.6568	0.0864	5.22%	0.9964	0.0426	4.27%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.6675	0.0864	5.18%	1.0023	0.0426	4.25%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.277	0.051	0.023	0.014	1.300	0.053
		Brain	1.547	0.040	0.000	0.000	1.547	0.040
		Thyroid	0.674	0.093	0.000	0.000	0.674	0.093
		Lungs	0.800	0.014	0.033	0.009	0.832	0.017
		Heart	0.618	0.022	0.172	0.052	0.790	0.057
		Thymus	0.676	0.076	0.000	0.000	0.676	0.076
		Stomach	0.669	0.022	0.000	0.000	0.669	0.022
		Large Intestine	0.700	0.016	0.000	0.000	0.700	0.016
		Small Intestine	0.805	0.016	0.000	0.000	0.805	0.016
		Liver	0.660	0.012	0.021	0.006	0.681	0.014
		Kidneys	0.647	0.029	0.000	0.000	0.647	0.029
		Adrenals	0.654	0.065	0.000	0.000	0.654	0.065
		Pancreas	0.859	0.086	0.000	0.000	0.859	0.086
		Spleen	0.662	0.031	0.000	0.000	0.662	0.031
		Bladder	0.423	0.048	0.000	0.000	0.423	0.048
		Testes	0.280	0.028	0.000	0.000	0.280	0.028

Table 80: Max / Race Track 3 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (3 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0019	0.0002	10.52%	0.0010	0.0004	35.31%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0418	0.0014	3.36%	0.0274	0.0124	45.35%
		Total SPE Dose for Mission	0.1129	0.0019	1.64%	0.0659	0.0126	19.16%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1605	0.0164	10.21%	0.0694	0.0010	1.43%
		GCR H (Z=1) Dose	1.0919	0.0651	5.96%	0.6493	0.0412	6.34%
		GCR He (Z=2) Dose	0.2006	0.0094	4.70%	0.1266	0.0076	5.98%
		Total GCR Dose for Mission	1.4530	0.0678	4.66%	0.8454	0.0419	4.96%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.5658	0.0678	4.33%	0.9113	0.0438	4.80%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.129	0.046	0.229	0.194	1.358	0.199
		Brain	1.371	0.036	0.400	0.076	1.772	0.084
		Thyroid	0.586	0.084	1.002	1.002	1.589	1.006
		Lungs	0.701	0.013	0.691	0.240	1.392	0.240
		Heart	0.534	0.019	0.000	0.000	0.534	0.019
		Thymus	0.586	0.067	0.000	0.000	0.586	0.067
		Stomach	0.578	0.019	0.000	0.000	0.578	0.019
		Large Intestine	0.604	0.014	0.000	0.000	0.604	0.014
		Small Intestine	0.699	0.014	0.000	0.000	0.699	0.014
		Liver	0.571	0.011	0.108	0.027	0.679	0.029
		Kidneys	0.560	0.025	0.000	0.000	0.560	0.025
		Adrenals	0.563	0.054	0.000	0.000	0.563	0.054
		Pancreas	0.739	0.074	0.000	0.000	0.739	0.074
		Spleen	0.570	0.027	0.000	0.000	0.570	0.027
		Bladder	0.498	0.022	0.000	0.000	0.498	0.022
		Uterus	1.057	0.073	0.000	0.000	1.057	0.073
		Ovaries	0.306	0.024	0.000	0.000	0.306	0.024
		Breasts	0.616	0.019	0.020	0.014	0.636	0.024

Table 81: Min / Race Track 3 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (3 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0019	0.0002	10.52%	0.0010	0.0004	35.31%
		Total SPE Dose for Mission	0.0072	0.0008	10.52%	0.0039	0.0014	35.31%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1704	0.0174	10.21%	0.0737	0.0053	7.17%
		GCR H (Z=1) Dose	1.2853	0.0766	5.96%	0.7644	0.0194	2.54%
		GCR He (Z=2) Dose	0.2204	0.0104	4.70%	0.1391	0.0033	2.39%
		Total GCR Dose for Mission	1.6760	0.0792	4.73%	0.9772	0.0204	2.09%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.6832	0.0792	4.71%	0.9811	0.0204	2.08%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.225	0.049	0.023	0.020	1.247	0.053
		Brain	1.485	0.038	0.022	0.007	1.507	0.039
		Thyroid	0.644	0.089	0.000	0.000	0.644	0.089
		Lungs	0.767	0.014	0.061	0.024	0.829	0.028
		Heart	0.592	0.021	0.000	0.000	0.592	0.021
		Thymus	0.647	0.073	0.000	0.000	0.647	0.073
		Stomach	0.643	0.021	0.000	0.000	0.643	0.021
		Large Intestine	0.672	0.016	0.000	0.000	0.672	0.016
		Small Intestine	0.772	0.015	0.000	0.000	0.772	0.015
		Liver	0.633	0.012	0.000	0.000	0.633	0.012
		Kidneys	0.621	0.028	0.000	0.000	0.621	0.028
		Adrenals	0.629	0.063	0.000	0.000	0.629	0.063
		Pancreas	0.821	0.082	0.000	0.000	0.821	0.082
		Spleen	0.633	0.030	0.000	0.000	0.633	0.030
		Bladder	0.551	0.024	0.000	0.000	0.551	0.024
		Uterus	1.153	0.080	0.000	0.000	1.153	0.080
		Ovaries	0.340	0.026	0.000	0.000	0.340	0.026
		Breasts	0.684	0.022	0.000	0.000	0.684	0.022

Table 82: Max / Race Track 3 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (3 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0014	0.0001	5.98%	0.0007	0.0004	61.60%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0061	0.0004	6.00%	0.0122	0.0054	44.51%
		Total SPE Dose for Mission	0.0592	0.0006	1.06%	0.0393	0.0061	15.45%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1146	0.0068	5.90%	0.0697	0.0044	6.28%
		GCR H (Z=1) Dose	1.4569	0.1559	10.70%	0.6532	0.0465	7.12%
		GCR He (Z=2) Dose	0.1873	0.0088	4.68%	0.1171	0.0046	3.96%
		Total GCR Dose for Mission	1.7588	0.1562	8.88%	0.8400	0.0470	5.59%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.8180	0.1562	8.59%	0.8793	0.0474	5.38%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.089	0.044	0.280	0.119	1.368	0.127
		Brain	1.325	0.034	0.940	0.781	2.265	0.781
		Thyroid	0.561	0.080	0.000	0.000	0.561	0.080
		Lungs	0.679	0.012	0.080	0.032	0.759	0.035
		Heart	0.518	0.019	0.000	0.000	0.518	0.019
		Thymus	0.566	0.062	0.000	0.000	0.566	0.062
		Stomach	0.566	0.019	0.000	0.000	0.566	0.019
		Large Intestine	0.590	0.014	0.035	0.035	0.625	0.038
		Small Intestine	0.678	0.013	0.590	0.197	1.268	0.198
		Liver	0.554	0.010	0.019	0.005	0.573	0.012
		Kidneys	0.545	0.025	0.000	0.000	0.545	0.025
		Adrenals	0.552	0.058	0.000	0.000	0.552	0.058
		Pancreas	0.709	0.070	0.000	0.000	0.709	0.070
		Spleen	0.549	0.025	0.000	0.000	0.549	0.025
		Bladder	0.481	0.021	0.000	0.000	0.481	0.021
		Testes	0.297	0.023	0.000	0.000	0.297	0.023

Table 83: Min / Race Track 3 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (3 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0014	0.0001	7.97%	0.0007	0.0004	61.60%
		Total SPE Dose for Mission	0.0054	0.0004	7.97%	0.0027	0.0017	61.60%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1321	0.0141	10.70%	0.0592	0.0111	18.78%
		GCR H (Z=1) Dose	1.1537	0.0766	6.64%	0.7080	0.0362	5.11%
		GCR He (Z=2) Dose	0.1953	0.0101	5.18%	0.1260	0.0062	4.89%
		Total GCR Dose for Mission	1.4811	0.0785	5.30%	0.8933	0.0384	4.29%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.4865	0.0785	5.28%	0.8960	0.0384	4.28%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	1.060	0.041	0.024	0.012	1.083	0.043
		Brain	1.279	0.032	0.083	0.079	1.362	0.085
		Thyroid	0.571	0.075	0.000	0.000	0.571	0.075
		Lungs	0.672	0.012	0.007	0.003	0.679	0.013
		Heart	0.529	0.019	0.000	0.000	0.529	0.019
		Thymus	0.575	0.064	0.000	0.000	0.575	0.064
		Stomach	0.575	0.019	0.000	0.000	0.575	0.019
		Large Intestine	0.602	0.014	0.004	0.004	0.606	0.015
		Small Intestine	0.686	0.014	0.060	0.020	0.746	0.024
		Liver	0.566	0.011	0.000	0.000	0.566	0.011
		Kidneys	0.554	0.025	0.000	0.000	0.554	0.025
		Adrenals	0.565	0.058	0.000	0.000	0.565	0.058
		Pancreas	0.740	0.074	0.000	0.000	0.740	0.074
		Spleen	0.570	0.027	0.000	0.000	0.570	0.027
		Bladder	0.359	0.042	0.000	0.000	0.359	0.042
		Testes	0.241	0.024	0.000	0.000	0.241	0.024

Table 84: Max / Race Track 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0019	0.0001	6.81%	0.0022	0.0008	34.89%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0288	0.0020	7.07%	0.0381	0.0123	32.31%
		Total SPE Dose for Mission	0.1018	0.0022	2.15%	0.1217	0.0132	10.82%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1268	0.0130	10.21%	0.0549	0.0008	1.43%
		GCR H (Z=1) Dose	0.8579	0.0511	5.96%	0.5102	0.0324	6.34%
		GCR He (Z=2) Dose	0.1599	0.0075	4.70%	0.1009	0.0060	5.98%
		Total GCR Dose for Mission	1.1446	0.0533	4.65%	0.6660	0.0329	4.94%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.2464	0.0533	4.28%	0.7877	0.0355	4.50%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.894	0.037	0.060	0.028	0.954	0.046
		Brain	1.085	0.028	1.159	0.221	2.244	0.223
		Thyroid	0.465	0.067	0.000	0.000	0.465	0.067
		Lungs	0.555	0.010	0.282	0.063	0.837	0.064
		Heart	0.423	0.015	0.000	0.000	0.423	0.015
		Thymus	0.465	0.053	0.000	0.000	0.465	0.053
		Stomach	0.458	0.015	0.515	0.189	0.973	0.190
		Large Intestine	0.479	0.011	0.255	0.066	0.733	0.066
		Small Intestine	0.553	0.011	0.001	0.001	0.555	0.011
		Liver	0.452	0.008	0.005	0.005	0.458	0.010
		Kidneys	0.443	0.020	0.000	0.000	0.443	0.020
		Adrenals	0.446	0.043	0.000	0.000	0.446	0.043
		Pancreas	0.587	0.059	0.000	0.000	0.587	0.059
		Spleen	0.452	0.021	0.000	0.000	0.452	0.021
		Bladder	0.395	0.017	0.000	0.000	0.395	0.017
		Uterus	0.837	0.058	0.005	0.003	0.842	0.058
		Ovaries	0.243	0.019	0.000	0.000	0.243	0.019
		Breasts	0.488	0.015	0.000	0.000	0.488	0.015

Table 85: Min / Race Track 7 T / Female Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (7 T)							
Crew Sex	Female (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0019	0.0001	6.81%	0.0022	0.0008	34.89%
		Total SPE Dose for Mission	0.0074	0.0005	6.81%	0.0085	0.0030	34.89%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1245	0.0127	10.21%	0.0539	0.0039	7.17%
		GCR H (Z=1) Dose	0.9640	0.0574	5.96%	0.5733	0.0145	2.54%
		GCR He (Z=2) Dose	0.1530	0.0072	4.70%	0.0966	0.0023	2.39%
		Total GCR Dose for Mission	1.2415	0.0593	4.77%	0.7238	0.0152	2.10%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.2489	0.0593	4.75%	0.7322	0.0155	2.12%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.886	0.036	0.001	0.001	0.888	0.036
		Brain	1.077	0.028	0.069	0.016	1.147	0.032
		Thyroid	0.460	0.065	0.000	0.000	0.460	0.065
		Lungs	0.554	0.010	0.029	0.006	0.583	0.012
		Heart	0.425	0.015	0.000	0.000	0.425	0.015
		Thymus	0.463	0.051	0.000	0.000	0.463	0.051
		Stomach	0.464	0.016	0.052	0.019	0.516	0.025
		Large Intestine	0.484	0.011	0.000	0.000	0.484	0.011
		Small Intestine	0.555	0.011	0.000	0.000	0.555	0.011
		Liver	0.454	0.009	0.000	0.000	0.454	0.009
		Kidneys	0.447	0.020	0.000	0.000	0.447	0.020
		Adrenals	0.453	0.047	0.000	0.000	0.453	0.047
		Pancreas	0.584	0.058	0.000	0.000	0.584	0.058
		Spleen	0.452	0.021	0.000	0.000	0.452	0.021
		Bladder	0.395	0.017	0.000	0.000	0.395	0.017
		Uterus	0.830	0.057	0.000	0.000	0.831	0.057
		Ovaries	0.244	0.019	0.000	0.000	0.244	0.019
		Breasts	0.493	0.016	0.000	0.000	0.493	0.016

Table 86: Max / Race Track 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Maximum							
Shielding Type	Race Track (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar max, initiate 20 average SPEs per year and 1 worst-case SPE per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	38	Dose per Average SPE	0.0005	0.0000	8.68%	0.0009	0.0004	51.28%
Worst-Case SPEs per Mission	1	Dose per Worst-Case SPE	0.0224	0.0013	5.74%	0.0245	0.0713	291.14%
		Total SPE Dose for Mission	0.0407	0.0013	3.22%	0.0579	0.0714	123.32%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar max, include Forbush modulation; run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.1066	0.0114	10.70%	0.0478	0.0034	7.12%
		GCR H (Z=1) Dose	0.8659	0.0510	5.90%	0.5265	0.0331	6.28%
		GCR He (Z=2) Dose	0.1370	0.0064	4.68%	0.0857	0.0034	3.96%
		Total GCR Dose for Mission	1.1096	0.0527	4.75%	0.6599	0.0334	5.07%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.1502	0.0527	4.58%	0.7178	0.0788	10.98%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.802	0.033	0.146	0.086	0.948	0.092
		Brain	0.976	0.025	0.244	0.074	1.221	0.078
		Thyroid	0.413	0.059	0.000	0.000	0.413	0.059
		Lungs	0.501	0.009	0.183	0.037	0.684	0.038
		Heart	0.383	0.014	0.000	0.000	0.383	0.014
		Thymus	0.417	0.046	0.000	0.000	0.417	0.046
		Stomach	0.419	0.014	0.000	0.000	0.419	0.014
		Large Intestine	0.436	0.010	0.002	0.002	0.437	0.010
		Small Intestine	0.500	0.010	0.009	0.009	0.509	0.013
		Liver	0.409	0.008	0.387	0.099	0.795	0.099
		Kidneys	0.403	0.019	0.000	0.000	0.403	0.019
		Adrenals	0.409	0.043	0.000	0.000	0.409	0.043
		Pancreas	0.523	0.052	0.000	0.000	0.523	0.052
		Spleen	0.405	0.019	0.000	0.000	0.405	0.019
		Bladder	0.355	0.015	0.000	0.000	0.355	0.015
		Testes	0.219	0.017	0.000	0.000	0.219	0.017

Table 87: Min / Race Track 7 T / Male Summary Data Sheet

SCENARIO DEFINITION								
Mission Type	Mars Flyby							
Mission Duration (days)	700							
Solar Cycle	Minimum							
Shielding Type	Race Track (7 T)							
Crew Sex	Male (MIRD)							
SOLAR PARTICLE EVENT (SPE) DOSE CALCULATION								
At solar min, initiate 2 average SPEs per year and 0 worst-case SPEs per mission								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
Average SPEs per Mission	4	Dose per Average SPE	0.0005	0.0000	8.68%	0.0009	0.0004	51.28%
		Total SPE Dose for Mission	0.0019	0.0002	8.68%	0.0034	0.0017	51.28%
GALACTIC COSMIC RAY (GCR) DOSE CALCULATION								
At solar min, run H, He (Z=1, Z=2) GCRs separately for efficiency								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		GCR (EXCEPT H,He) Dose	0.0976	0.0104	10.70%	0.0438	0.0082	18.78%
		GCR H (Z=1) Dose	0.7845	0.0521	6.64%	0.4814	0.0246	5.11%
		GCR He (Z=2) Dose	0.1507	0.0078	5.18%	0.0972	0.0048	4.89%
		Total GCR Dose for Mission	1.0328	0.0537	5.20%	0.6224	0.0264	4.24%
TOTAL DOSE CALCULATION								
			E (Sv)	$\sigma$ (Sv)	$\sigma$ (%)	D (Gy)	$\sigma$ (Gy)	$\sigma$ (%)
		Total Dose for Mission	1.0347	0.0537	5.19%	0.6258	0.0264	4.22%
ORGAN DOSE EQUIVALENT CALCULATIONS								
			GCR (mSv/d)	$\sigma$ (mSv/d)	SPE (mSv/d)	$\sigma$ (mSv/d)	TOTAL (mSv/d)	$\sigma$ (mSv/d)
		Active Bone Marrow	0.783	0.031	0.008	0.008	0.790	0.032
		Brain	0.942	0.023	0.013	0.007	0.956	0.024
		Thyroid	0.427	0.055	0.000	0.000	0.427	0.055
		Lungs	0.495	0.009	0.000	0.000	0.495	0.009
		Heart	0.391	0.014	0.000	0.000	0.391	0.014
		Thymus	0.427	0.049	0.000	0.000	0.427	0.049
		Stomach	0.421	0.014	0.000	0.000	0.421	0.014
		Large Intestine	0.443	0.010	0.000	0.000	0.443	0.010
		Small Intestine	0.507	0.010	0.000	0.000	0.507	0.010
		Liver	0.418	0.008	0.039	0.010	0.457	0.013
		Kidneys	0.408	0.018	0.000	0.000	0.408	0.018
		Adrenals	0.414	0.040	0.000	0.000	0.414	0.040
		Pancreas	0.552	0.056	0.000	0.000	0.552	0.056
		Spleen	0.424	0.020	0.000	0.000	0.424	0.020
		Bladder	0.266	0.031	0.000	0.000	0.266	0.031
		Testes	0.176	0.017	0.000	0.000	0.176	0.017

## Appendix A11: Beam Experiment Plan

The following plan describes a set of beam experiments utilizing the MD Anderson Cancer Center Proton Therapy Center (PTC) clinical fixed beam line.

The main objective of these experiments is to conduct an end-to-end simulation of solar space radiation incident upon a spacecraft with active magnetic shielding and an astronaut dwelling inside. The goal is to determine the radiation dose reduction to the astronaut offered by active magnetic shielding versus passive shielding (water) or no shielding (only the aluminum spacecraft) in a relevant environment. The proton beam available at the PTC is a close approximation of the energy of solar flare protons (~200 MeV), and an electromagnet of relevant field (~1-2 T) to active magnetic shielding configurations is available for use within our department.

The secondary objective of these experiments is to validate our GEANT4 simulation code in a laboratory environment. The same techniques and strategies we have used to build GEANT4 models of the PTC geometry and beam characteristics have been used to build full-scale models of a spacecraft with active magnetic shielding in the space radiation environment. It is impractical to test the full-scale model in a laboratory environment due to cost, mass, size, and the complexity of the space environment; yet if we are able to accurately predict the dose delivered to our simulated astronaut in the smaller-scale PTC laboratory environment, this will increase our confidence in the data produced by our full-scale simulations.

### *Experiment Specific Aims*

These beam experiments supplement a larger project for the author's dissertation. Specific Aim 2 of the larger project is the main objective of this experiment: *Perform beam experiments to measure radiation dose behind magnetic shielding in a simulated Mars flyby mission radiation*

*environment.* We will perform beam experiments in simulated space radiation conditions to analyze magnetic shielding effectiveness across portions of the space radiation spectrum. We compare these results to radiation doses measured with no shielding or water shielding to validate our Monte Carlo simulations in GEANT4.

To support the main objective of this experiment, we will complete the following specific aims:

**Specific Aim 1:** Collect distributed dose measurements using the Matrixx™ detector behind a variety of shielding configurations. In this portion of the experiment, we will vary several experimental parameters including shielding configuration, magnetic field, and proton beam energy.

**Specific Aim 2:** Determine the energy shift of the incident radiation using the Zebra detector behind a variety of shielding configurations. In this portion of the experiment, we will vary several experimental parameters including shielding configuration, magnetic field, and proton beam energy.

**Specific Aim 3:** Compare results of laboratory experiments conducted in Specific Aims 1 and 2 to results of GEANT4 Monte Carlo simulations and determine the level of agreement. In this portion of the experiment, we will compare the data collected in Specific Aims 1 and 2 to the data collected in our Monte Carlo simulations of the experimental set up in GEANT4.

#### *Experimental Methods and Measurement Strategy*

To support our experiment specific aims, we have developed the following experimental methods and measurement strategy based on four experimental parameters. Details on the experimental set up are provided as well as technical specifications for experimental hardware.

### *Experimental Parameters*

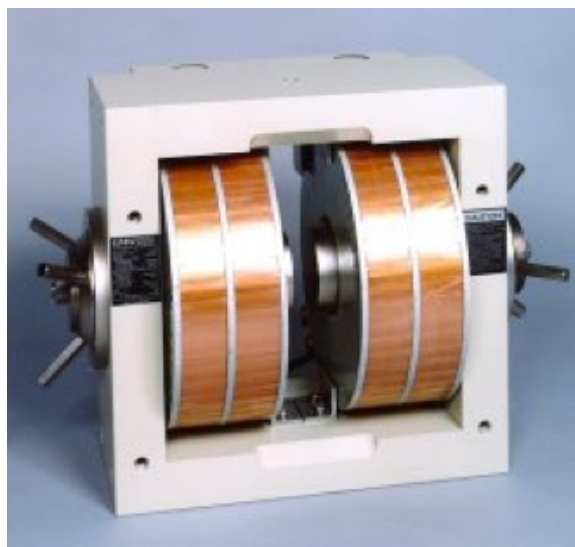
- **Output Beam Energy:** We have selected to perform experiment runs at the nominal (maximum) PTC output beam energy, which will deliver protons of  $204 \text{ MeV} \pm 2 \text{ MeV}$  to our target at a distance of 300 cm from isocenter. We also request to perform runs at a lower output beam energy, approximately 100 MeV, to cover more of the solar flare proton energy spectrum with our experiments.
- **Shielding Type:** We have selected to perform experiment runs first with a baseline configuration with no shielding, simply the proton beam and our detector. This configuration will allow us to calculate our effective d/MU ratio for our set up and serve as a basis for comparison for all subsequent runs. Next, we select to perform experiment runs with minimal shielding, represented by a slab of aluminum alloy of similar makeup and thickness of a typical spacecraft wall. This configuration will allow us to calculate dose to our detector inside a simulated spacecraft. Next, we select to perform experiment runs with passive water shielding, represented by a container of water 20 cm thick behind the aluminum slab. This configuration will allow us to calculate dose to our detector inside a simulated spacecraft with NASA's state-of-the-art water wall shielding configuration. Finally, we will perform experiment runs with active magnetic shielding, represented by a tunable electromagnet in front of the aluminum slab. This configuration will allow us to calculate dose to our detector inside a simulated spacecraft protected by our proposed new shielding configuration.
- **Magnetic Field:** We have selected to perform experiment runs with two different magnetic fields. By varying the magnet current and pole gap on a tunable electromagnet available within our department, we expect to reach magnetic fields close to 2 T. We will perform our runs at a field close to 1 T and the max field we can sustain with a reasonable pole gap distance, likely around 1.8 T. These two configurations will allow us to determine the effect of a stronger magnetic field vs. a weaker field. This information

will serve to help us validate our GEANT4 Monte Carlo simulations on the effectiveness of magnetic fields for protection against solar flare protons. Details on the electromagnet are provided in the following section.

- **Detector Selection:** We have chosen two different detectors for our experiment runs. The first is the IBA Dosimetry Matrixx™ detector, an ion chamber array. This detector will allow us to collect dose measurements across a 2-dimensional field to determine deflection and spread of the proton beam in different shielding configurations. The second detector is the IBA Dosimetry Zebra detector, a multi-layer ion chamber. This detector will allow us to collect measurements of the beam energy (via Bragg peak measurements) as it reaches the target; from this we can determine any effects of the passive or active shielding on hardening or softening the proton energy spectrum. Details on each of these detectors are provided in the following section.

### *Hardware Specifications*

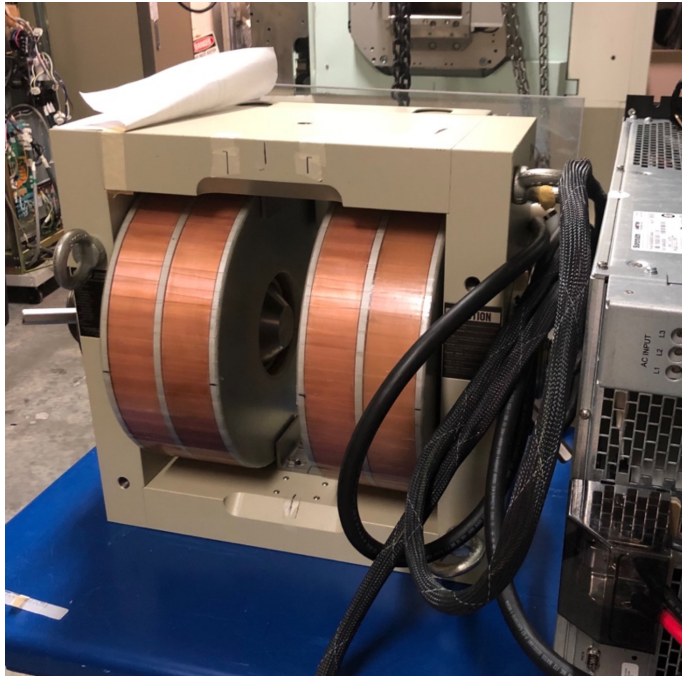
The tunable electromagnet available within our department is a GMW 3472-70 model electromagnet (see Figure 150 and Figure 152) with 100 mm pole gap. The magnetic field in the pole gap is estimated via excitation curves in the magnet specification guide and is considered accurate to within 5% (GMW 2009). By varying the magnet current and pole gap, we expect to be able to reach magnetic fields close to 2 T.



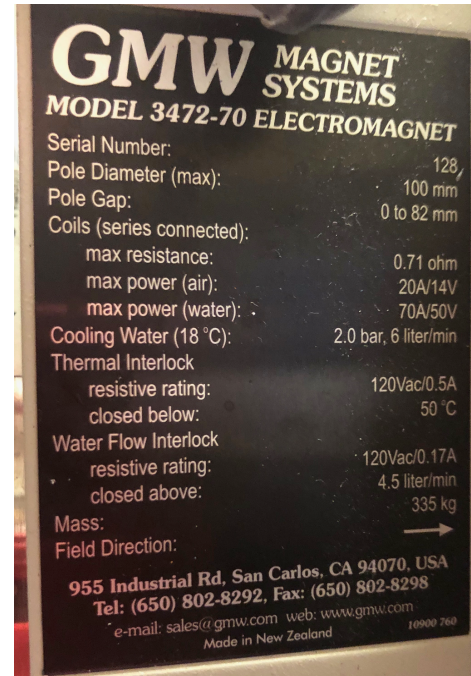
**Figure 150: GMW 3472-70 Electromagnet; courtesy of GMW**

The magnet itself is quite heavy (325 kg) and requires technician assistance to move between buildings or rotate the irradiation configuration. The magnet is currently located at the PTC facility

as of February 2019. PTC technicians assisted with rotation of the magnet to the configuration pictured in Figure 151 in November 2019, in order to be irradiated horizontally in the PTC clinical fixed beamline.



**Figure 151: GMW 3472-70 Electromagnet Set Up for Horizontal Irradiation**



**Figure 152: GMW 3472-70 Electromagnet Specifications**

The magnet also includes its own water chiller to circulate water over the poles to dissipate heat and its own power supply. The water chiller is a BV Thermal Systems Mercury MC200-A1-E1-H2-J1 air-cooled closed loop refrigerated fluid recirculator (see Figure 153) with a 2 gallon reservoir capacity and 5°-35° C temperature control (BV Thermal Systems 2014). The power supply is an Ametek Programmable Power Sorensen SGA Series DC power supply providing 4-30 kW output power (Ametek Programmable Power 2014). Power and water supply accommodations in the PTC clinical fixed target room are in-work as of January 2020.

The magnet specification guide (GMW 2009) includes excitation curves for field vs. pole gap and field vs. current (see Figure 154 and Figure 155), and a former graduate student at MD Anderson (A. Rubinstein) produced field maps of both the between pole field uniformity as well as the fringe field drop off (see Figure 157). As part of this experiment, we will use an

AlphaLab Inc. Gaussmeter Model GM2 (AlphaLab, Inc. 2012) to spot check these field measurements to ensure the field remains consistent after long-term storage of the magnet (see Figure 156).

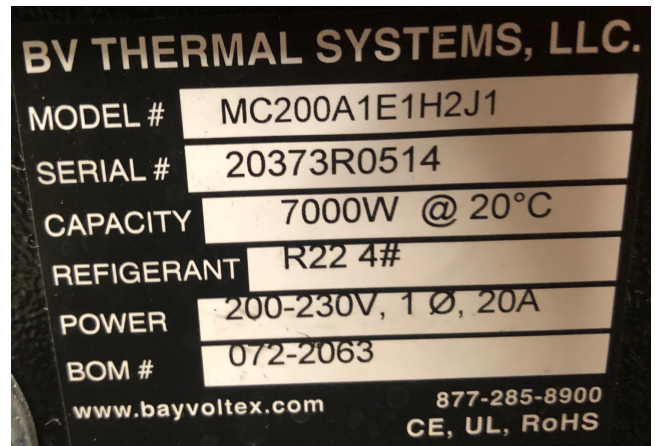


Figure 153: BV Thermal Systems Chiller Specifications

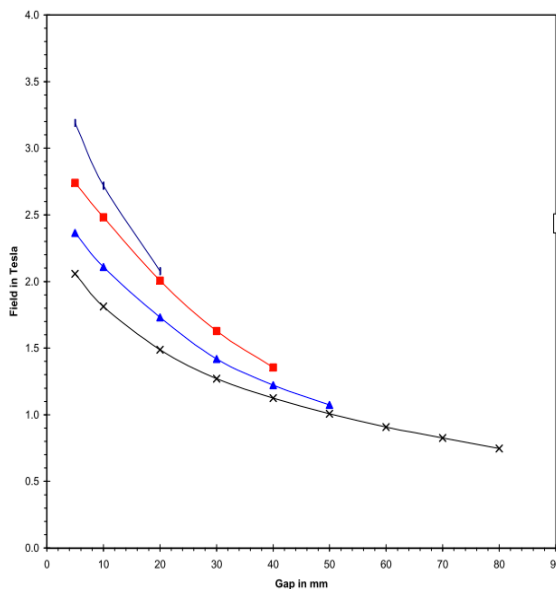


Figure 154: GMW 3472-70 Magnet Excitation Curve vs. Pole Gap (GMW 2009); courtesy of GMW

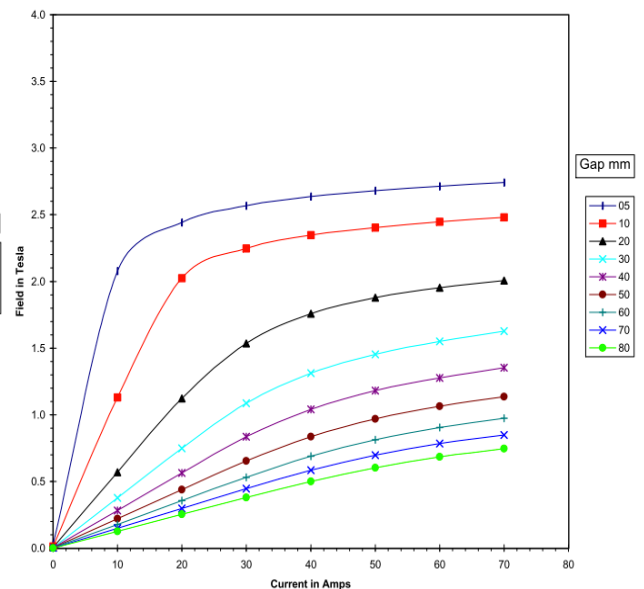
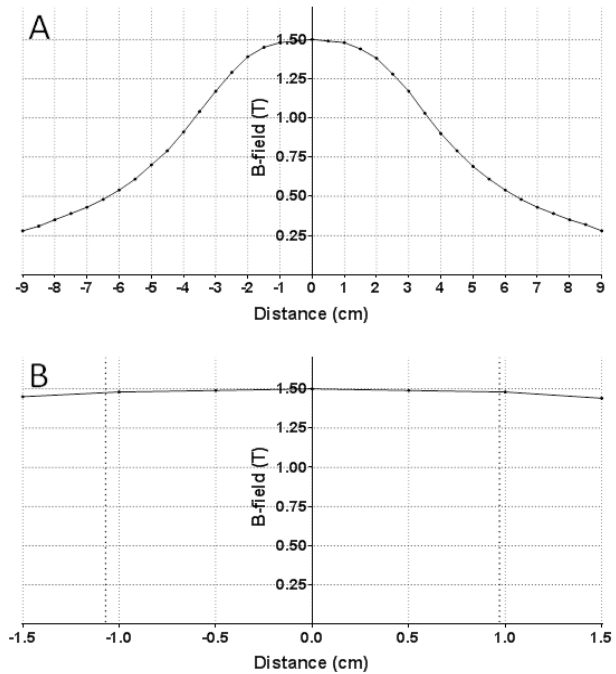


Figure 155: GMW 3472-70 Magnet Excitation Curve vs. Current (GMW 2009); courtesy of GMW



**Figure 156: Gaussmeter GM2**



**Figure 157: GMW Magnet Field Maps; courtesy of A. Rubinstein**

For beam deflection measurements, we will use the IBA Dosimetry Matrixx™ Evolution ion chamber array detector available within our department. The Matrixx™ Evolution detector (see Figure 158) provides a coplanar 1020 ion chamber grid (32 x 32 except for the 4 corner positions) with an active area of 24.4 x 24.4 cm<sup>2</sup>. The pixel spacing is 7.62 mm center-to-center, the nominal sensitivity is 2.4 nC/Gy, the dose rate range is 0.02 to 12 Gy/min, and the water equivalent depth is 3.1 m (IBA Dosimetry 2016). The Matrixx™ Evolution detector will allow us to map the dose deposition across the grid to determine the deflection of the beam resulting from our shielding configuration.

For energy shift measurements, we will use the IBA Dosimetry Zebra multi-layer ion chamber array detector available within our department. The Zebra detector (see Figure 159) provides 180 plane parallel ion chambers with an active depth of 33 cm. The spatial resolution is 2 mm, the typical sensitivity is 14.76 nC/Gy, and the dose rate range is 0.5 to 15 Gy/min (IBA Dosimetry 2013). The Zebra detector will provide us with the Bragg peak of the radiation reaching the

detector, which will allow us to determine the energy shift resulting from our shielding configuration.

For the passive shielding and no shielding configurations, we will use a 20 cm thickness solid water available via our department and a 1.8 cm thickness aluminum 6061 alloy slab.



Figure 158: Matrixx™ Evolution; courtesy of IBA Dosimetry



Figure 159: Zebra Detector; courtesy of IBA Dosimetry

### *Beam Characteristics and d/MU Ratio*

The typical experimental beam delivered to the clinical fixed beamline at the PTC is a pristine beam with the modulator wheel held fixed at the thinnest azimuth, using the small snout, with a square 10 cm x 10 cm field at isocenter. Experimental exposures are typically conducted with the target placed 300 cm downstream from isocenter. The nominal (maximum) energy of the PTC proton beam is  $205 \text{ MeV} \pm 2 \text{ MeV}$  at isocenter and  $204 \text{ MeV} \pm 2 \text{ MeV}$  at 300 cm downstream. The duty cycle of the beam on time is 25% (500 ms beam on followed by 1500 ms beam off, for a total of 2 second cycle) (Ferrone & O'Brien 2018).

The PTC clinical beamline is calibrated to deliver an output dose per MU (d/MU) of 1 cGy/MU under specific conditions. These conditions include proton beam of energy 250 MeV with passive

scattering set up for the medium snout, SOBP width of 10 cm, aperture size 10 cm x 10 cm, snout fully retracted, and a calibration depth of 23.5 cm at the center of the SOBP at isocenter with source to calibration point distance of 270 cm (Gillin et al. 2010; Sahoo et al. 2008). Modifications to this calibration would be required in order to determine the absolute dose expected for our experimental setup.

However, for the purposes of this experiment, we are concerned with the relative dose delivered to the detector in various shielding configurations, as opposed to the absolute dose. Therefore, we will use a baseline configuration with no shielding, with a starting d/MU ratio of 1 cGy/MU to determine the *effective* d/MU ratio for our unique experimental set up. We will also request a small beam size of ~1 cm x 1 cm. We will use this baseline effective d/MU to compare the dose delivered to the detector in all subsequent configurations.

#### *Description of Experiment Runs*

In order to satisfy our specific aims, the following experiment runs are proposed (see Table 88). Parameters to be varied throughout the experiments include the output beam energy, shielding configuration including magnetic field where applicable, and detector selection. With these selections, we will collect a variety of data on the effectiveness of active magnetic shielding in reducing dose to an astronaut in a solar proton radiation environment. We will also compare the results to our GEANT4 methodology to determine the level of agreement.

**Table 88: List of Experimental Runs at the PTC**

Run	Beam Energy (MeV)	MU	Shielding Configuration	Detector	Measurement
MAX BEAM ENERGY; SET UP MATRIX <sup>TM</sup> DETECTOR + FILM					
1	~200 (Max)	100	None	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
ADD ALUMINUM SLAB; SWAP FILM					
2	~200 (Max)	100	Aluminum	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
ADD SOLID WATER; SWAP FILM					
3	~200 (Max)	100	Aluminum + Water	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
REMOVE FILM, SOLID WATER, ALUMINUM SLAB; SWITCH FROM MATRIX <sup>TM</sup> TO ZEBRA DETECTOR					
4	~200 (Max)	100	None	Zebra	Zebra Bragg peak
ADD ALUMINUM SLAB					
5	~200 (Max)	100	Aluminum	Zebra	Zebra Bragg peak
ADD SOLID WATER					
6	~200 (Max)	100	Aluminum + Water	Zebra	Zebra Bragg peak
REMOVE SOLID WATER; ADD MAGNET AT 1 T					
7	~200 (Max)	100	Aluminum + Magnetic 1T	Zebra	Zebra Bragg peak
RAMP MAGNET TO MAX					
8	~200 (Max)	100	Aluminum + Magnetic 1.75T (Max)	Zebra	Zebra Bragg peak
SWITCH ZEBRA TO MATRIX <sup>TM</sup> DETECTOR; ADD FILM					
9	~200 (Max)	100	Aluminum + Magnetic 1.75T (Max)	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot deflection on film
RAMP MAGNET TO 1 T; SWAP FILM					
10	~200 (Max)	100	Aluminum + Magnetic 1T	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot deflection on film
POWER DOWN + REMOVE MAGNET; CHANGE BEAM ENERGY TO ~100 MeV; SWAP FILM					
11	~100	100	None	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
ADD ALUMINUM SLAB; SWAP FILM					
12	~100	100	Aluminum	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
ADD SOLID WATER; SWAP FILM					
13	~100	100	Aluminum + Water	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot on film
REMOVE FILM, SOLID WATER, ALUMINUM SLAB; SWITCH FROM MATRIX <sup>TM</sup> TO ZEBRA DETECTOR					
14	~100	100	None	Zebra	Zebra Bragg peak
ADD ALUMINUM SLAB					
15	~100	100	Aluminum	Zebra	Zebra Bragg peak
ADD SOLID WATER					
16	~100	100	Aluminum + Water	Zebra	Zebra Bragg peak
REMOVE SOLID WATER; ADD MAGNET AT 1 T					
17	~100	100	Aluminum + Magnetic 1T	Zebra	Zebra Bragg peak
RAMP MAGNET TO MAX					
18	~100	100	Aluminum + Magnetic 1.75T (Max)	Zebra	Zebra Bragg peak
SWITCH ZEBRA TO MATRIX <sup>TM</sup> DETECTOR; ADD FILM					
19	~100	100	Aluminum + Magnetic 1.75T (Max)	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot deflection on film
RAMP MAGNET TO 1 T; SWAP FILM					
20	~100	100	Aluminum + Magnetic 1T	Matrix <sup>TM</sup> + Film	Matrix <sup>TM</sup> dose distribution across grid; baseline dose values; beam spot deflection on film

### *Beam Time*

Per previous experimental testing, the typical beam on time to deliver 1000 MU in the PTC clinical fixed beamline was 40-50 seconds, for a total per run time of 160-200 seconds (Ferrone & O'Brien 2018). We could request 100 MU/run; therefore, we estimate the per run time to be approximately 16-20 seconds. Due to the smaller beam size we are requesting, the time to deliver 100 MU could be much longer; in this case we can request a lower MU/run to keep each run time on the order of 1 minute. We estimate the time between runs to include target room access, detector swap out, shielding configuration change, and/or magnet ramp up/down to be approximately 5 minutes per run. Thus, with 60 planned runs, our estimated total time will be approximately 6 hours, not including set up and tear down time or time to modify the output beam energy. Seeing as this estimate represents an ideal case, any issue with beam availability, technical difficulties, beam drop outs, or repeat runs will add to this time. Additionally, PTC technicians will likely require time to modify the output beam energy. Therefore, we will request a full day of experimental runs at the PTC.

### *Experiment Setup*

To determine the scale of our experiment, we first need to determine the deflection angle and distance of ~200 MeV protons acted upon by a 3-4 cm magnetic field of 1-2 T.

To determine the deflection angle, we need to know the magnetic field, the charge and mass of the incident particle, and the particle's velocity, per the equation:  $r = \frac{mv}{qB}$ .

The charge of a proton is  $1\text{ e} = 1.6 \times 10^{-19}\text{ C}$ , and the rest mass of a proton ( $m_0$ ) is  $1.67 \times 10^{-27}\text{ kg}$ .

We can determine the velocity of the particle when we know its energy. The rest energy of a proton ( $E_0$ ) is 939 MeV. The total energy of a high energy proton is equal to the rest energy ( $E_0$ ) plus the kinetic energy ( $E_K$ ), the nominal energy per nucleon reported by the accelerator. For a “100 MeV” proton, its total energy is the rest energy (939 MeV) plus its kinetic energy (100 MeV) for a total energy of 1039 MeV. For a “200 MeV” proton, the total energy is 1139 MeV, and so on. We can set the total energy equal to  $\gamma E_0$  and solve for  $\beta$  and then  $v$ .

For 100 MeV protons:

$$E = 1039 \text{ MeV} = \gamma \times 939 \text{ MeV}$$

$$1.1065 = \gamma = \frac{1}{\sqrt{1 - \beta^2}}$$

$$\sqrt{1 - \beta^2} = 0.90375$$

$$1 - \beta^2 = 0.81677$$

$$\beta^2 = 0.18323$$

$$\beta = 0.42805 = v/c$$

$$v = 0.42805 \times 3 \times 10^8 \text{ m/s} = 1.284 \times 10^8 \text{ m/s}$$

For 200 MeV protons:

$$E = 1139 \text{ MeV} = \gamma \times 939 \text{ MeV}$$

$$1.2130 = \gamma = \frac{1}{\sqrt{1 - \beta^2}}$$

$$\sqrt{1 - \beta^2} = 0.824407$$

$$1 - \beta^2 = 0.67965$$

$$\beta^2 = 0.32035$$

$$\beta = 0.56600 = v/c$$

$$v = 0.56600 \times 3 \times 10^8 \text{ m/s} = 1.700 \times 10^8 \text{ m/s}$$

For the magnetic field, we plan to use 1-2 T in our experiments.

For 100 MeV protons in 1 T magnetic field:

$$r = \frac{mv}{qB} = \frac{1.67 \times 10^{-27} \text{ kg} \times 1.284 \times 10^8 \text{ m/s}}{1.6 \times 10^{-19} \text{ C} \times 1 \text{ T}} = 1.34 \text{ m}$$

For 100 MeV protons in 2 T magnetic field:

$$r = \frac{mv}{qB} = \frac{1.67 \times 10^{-27} \text{ kg} \times 1.284 \times 10^8 \text{ m/s}}{1.6 \times 10^{-19} \text{ C} \times 2 \text{ T}} = 0.67 \text{ m}$$

For 200 MeV protons in 1 T magnetic field:

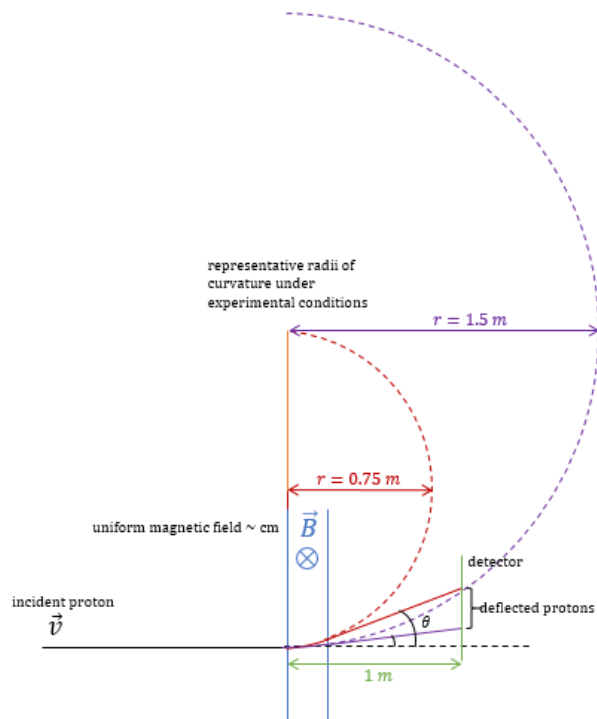
$$r = \frac{mv}{qB} = \frac{1.67 \times 10^{-27} \text{ kg} \times 1.7 \times 10^8 \text{ m/s}}{1.6 \times 10^{-19} \text{ C} \times 1 \text{ T}} = 1.77 \text{ m}$$

For 200 MeV protons in 2 T magnetic field:

$$r = \frac{mv}{qB} = \frac{1.67 \times 10^{-27} \text{ kg} \times 1.7 \times 10^8 \text{ m/s}}{1.6 \times 10^{-19} \text{ C} \times 2 \text{ T}} = 0.887 \text{ m}$$

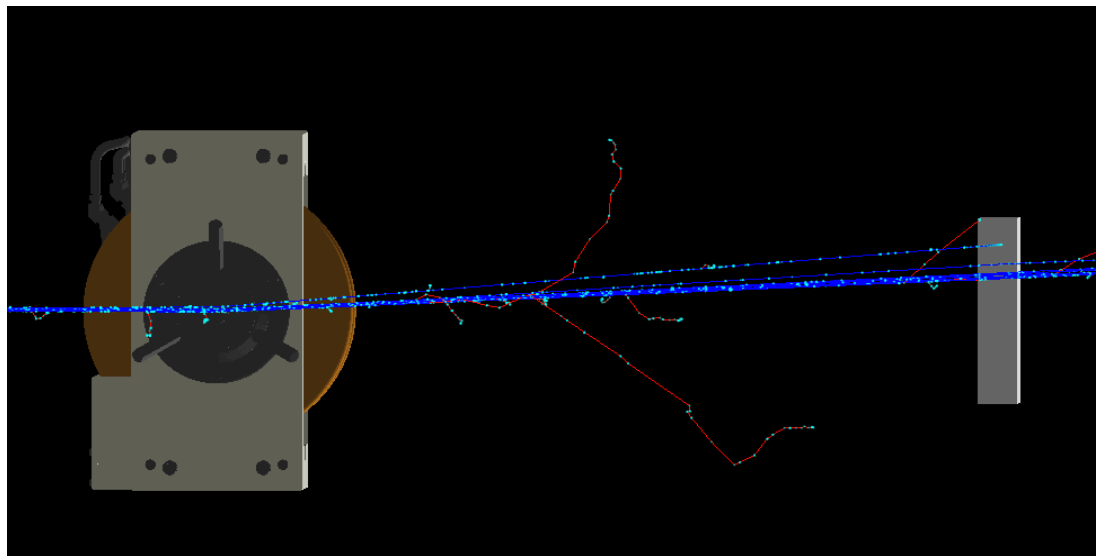
These radii indicate the distance that we would measure a 90° particle deflection in a uniform magnetic field of at least that distance. Since the experimental magnet has a uniform magnetic field of only a few cm, the deflection angle will be much less than 90°, and distance to the detector (film or ion chamber array) is depicted in Figure 160. To a first-order approximation, we should have distinguishable deflection angles for the proton energies and fields in the experiment with detector placement of ~1 m behind the magnet. This is feasible in clinical fixed beamline at the

PTC; given the typical experimental targets and size of the room. We have modeled our experimental setup in GEANT4 (see Figure 158) with a magnet to detector distance of 1 m.



**Figure 160: Schematic of Deflection Angle Using Magnetic Shielding**

As of the time of this publication, the required facilities upgrade at the PTC (power and water for the magnet within the target room) is ongoing. Therefore, results from these experiments have been delayed and will be reported at a later time.



**Figure 161: Experiment Setup Modeled in GEANT4**

## Bibliography

Adams J. 1992. Cosmic Radiation: Constraints on Space Exploration. *Nucl Tracks Radiat*

*Meas.* **20**. DOI: [10.1016/1359-0189\(92\)90023-O](https://doi.org/10.1016/1359-0189(92)90023-O).

Adams J, Dietrich W, Xapsos M. 2011. Probabilistic Solar Energetic Particle Models.

Proceedings of the 23<sup>rd</sup> International Cosmic Ray Conference, 11-18 August 2011,  
Beijing, China.

Adams J, Hathaway D, Grugel R, Watts J, Parnell T, Gregory J, Winglee R. 2005.

Revolutionary Concepts of Radiation Shielding for Human Exploration of Space. National  
Aeronautics and Space Administration Report No. NASA-TM-2005-213688.

Agostinelli S, Allison J, Amako K, Apostolakis J, Araujo H, Arce P, Asai M, Axen D, Banerjee S,  
Barrand G, Behner F, Bellagamba L, Boudreau J, Broglia L, Brunengo A, Burkhardt H,  
Chauvie S, Chuma J, Chytrcek R, Cooperman G, Cosmo G, Degtyarenko P, Dell'Acqua  
A, Depaola G, Dietrich D, Enami R, Felicello A, Ferguson C, Fesefeldt H, Folger G,  
Foppiano F, Forti A, Garelli S, Giani S, Giannitrapani R, Gibin D, Gomez Cadenas JJ,  
Gonzalez I, Gracia Abril G, Greeniaus G, Greiner W, Grichine V, Grossheim A, Guatelli  
S, Gumplinger P, Hamatsu R, Hashimoto K, Hasui H, Heikkinen A, Howard A,  
Ivanchenko V, Johnson A, Jones FW, Kallenbach J, Kanaya N, Kawabata M, Kawabata  
Y, Kawaguti M, Kelner S, Kent P, Kimura A, Kodama T, Kokoulin R, Kossov M, Kurashige  
H, Lamanna E, Lampen T, Lara V, Lefebvre V, Lei F, Liendl M, Lockman W, Longo F,  
Magni S, Maire M, Medernach E, Minamimoto K, Mora de Freitas P, Morita Y, Murakami  
K, Nagamatsu M, Nartallo R, Nieminen P, Nishimura T, Ohtsubo K, Okamura M, O'Neale  
S, Oohata Y, Paech K, Perl J, Pfeiffer A, Pia MG, Ranjard F, Rybin A, Sadilov S, Di Salvo  
E, Santin G, Sasaki T, Savvas N, Sawada Y, Scherer S, Sei S, Sirotenko V, Smith D,  
Starkov N, Stoecker H, Sulkimo J, Takahata M, Tanaka S, Tcherniaev E, Safai Tehrani E,  
Tropeano M, Truscott P, Uno H, Urban L, Urban P, Verderi M, Walkden A, Wander W,

- Weber H, Wellisch JP, Wenaus T, Williams DC, Wright D, Yamada T, Yoshida H, Zschesche D. 2003. GEANT4: A Simulation Toolkit. *Nucl Phys A*. **506**. DOI: [10.1016/S0168-9002\(03\)01368-8](https://doi.org/10.1016/S0168-9002(03)01368-8).
- Ainsbury E, Bouffler S, Dörr W, Graw J, Muirhead C, Edwards A, Cooper J. 2009. Radiation Cataractogenesis: A Review of Recent Studies. *Radiat Res*. **172**. DOI: [10.1667/RR1688.1](https://doi.org/10.1667/RR1688.1).
- AlphaLab, Inc. 2012. Gaussmeter Model GM2 Instructions. <http://www.alphalabinc.com/product/gm2> [accessed 15 January 2020].
- Ambrogini F, Battiston R, Burger W. 2016. Evaluation of Superconducting Magnet Shield Configurations for Long Duration Manned Space Missions. *Front Oncol*. **6**. DOI: [10.3389/fonc.2016.00097](https://doi.org/10.3389/fonc.2016.00097).
- Ametek Programmable Power. 2014. M550129-01 Rev AD: Sorensen SGA Series DC Power Supplies Operation Manual. <http://www.programmablepower.com> [accessed 15 January 2020].
- Antokhin E, Buzykaev A, Chupyra A, Fedorov D, Ganzhur S, Kolachev G, Litvinov A, Medvedko A, Mikerov V, Mikhailov S, Onuchin A, Singatulin S, Aleksan R, Bourgeois P, Gosset L, Graffin P, London G, Mols J, Toussaint J, Berndt M, Coombes R, Ecklund S, Jensen D, Keller L, Krebs J, Lynch H, Wolf Z. 1999. Simulation and Measurement of the Fringe Field of the 1.5 T BABAR Solenoid. *Nucl Instrum Methods Phys Res A*. **432**. DOI: [10.1016/S0168-9002\(99\)00278-8](https://doi.org/10.1016/S0168-9002(99)00278-8).
- Atomic Energy Commission. 1962. Proceedings of the 1st Symposium on the Protection Against Radiation Hazards in Space, 5-7 November 1962, Gatlinburg, Tennessee.
- Atomic Energy Commission. 1964. Proceedings of the 2nd Symposium on the Protection Against Radiation Hazards in Space, 12-14 October 1964, Gatlinburg, Tennessee.

- Atwell W, Hardy A, Peterson L. 1996. Organ Radiation Doses and Lifetime Risk of Excess Cancer for Several Space Shuttle Missions. *Adv Space Res.* **18**. DOI: [10.1016/0273-1177\(96\)00032-4](https://doi.org/10.1016/0273-1177(96)00032-4).
- Atwell W, Rojdev K, Lian D, Hill M. 2014. Metal Hydrides, MOFs, and Carbon Composites as Space Radiation Shielding Mitigators. Proceedings of the 44<sup>th</sup> International Conference on Environmental Systems, 13-17 July 2014, Tucson, Arizona.
- Augusteyn R. 2010. On the Growth and Internal Structure of the Human Lens. *Exp Eye Res.* **90**. DOI: [10.1016/j.exer.2010.01.013](https://doi.org/10.1016/j.exer.2010.01.013).
- Autsavapromporn N, De Toledo S, Buonanno M, Jay-Gerin J, Harris A, Azzam E. 2011. Intercellular Communication Amplifies Stressful Effects in High-Charge, High-Energy (HZE) Particle-Irradiated Human Cells. *Radiat Res.* **52**. DOI: [10.1269/jrr.10114](https://doi.org/10.1269/jrr.10114).
- Autsavapromporn N, Suzuki M, Plante I, Liu C, Uchihori Y, Hei T, Azzam E, Murakami T. 2013. Participation of Gap Junction Communication in Potentially Lethal Damage Repair and DNA Damage in Human Fibroblasts Exposed to Low- or High-LET Radiation. *Mutat Res.* **756**. DOI: [10.1016/j.mrgentox.2013.07.001](https://doi.org/10.1016/j.mrgentox.2013.07.001).
- AVCO-Everett Research Laboratory. 1968. Study of Plasma Radiation Shielding. National Aeronautics and Space Administration Report No. NASA-CR-61640.
- Badhwar G, Cucinotta F. 1998. Depth Dependence of Absorbed Dose, Dose Equivalent and Linear Energy Transfer Spectra of Galactic and Trapped Particles in Polyethylene and Comparison with Calculations of Models. *Radiat Res.* **149**. DOI: [10.2307/3579953](https://doi.org/10.2307/3579953).
- Baiocco G, Giruaudo M, Bocchini L. 2018. A Water-filled Garment to Protect Astronauts During Interplanetary Missions Tested On Board the ISS. *Life Sci Space Res.* **18**. DOI: [10.1016/j.lssr.2018.04.002](https://doi.org/10.1016/j.lssr.2018.04.002).

Baldwin A, Cladis J, Hawkins S, Kool C, Mattes B, Miller T, Riddle G, Weaver J. 1964.

Feasibility of Magnetic Orbital Shielding System. United States Air Force Report No. AF-04(695)-252 8-94-64-2.

Bamford R, Gibson K, Thornton A, Bradford J, Bingham R, Gargate L, Silva L, Fonseca R, Hapgood M, Norberg C, Todd T, Stamper R. 2008. The Interaction of a Flowing Plasma with a Dipole Magnetic Field: Measurements and Modeling of a Diamagnetic Cavity Relevant to Spacecraft Protection. *Plasma Phys Control Fusion*. **50**. DOI: [10.1088/0741-3335/50/12/124025](https://doi.org/10.1088/0741-3335/50/12/124025).

Barghouty A, Thibeault S. 2006. The Exploration Atmospheres Working Group's Report on Space Radiation Shielding Materials. National Aeronautics and Space Administration Report No. NASA-TM-2006-214604.

Barker J. 1950. The Magnetic Field Inside a Solenoid. *Brit J Appl Phys*. **1**. DOI: [10.1088/0508-3443/1/3/303](https://doi.org/10.1088/0508-3443/1/3/303).

Battista J. 2016. Radiation on a Voyage to Mars: All Aboard? *Adv Med Phys*. **6**.

Battiston R, Burger W, Calvelli V, Musenich R, Choutko V, Datskov V, Della Torre A, Venditti F, Gargiulo C, Laurenti G, Lucidi S, Harrison S, Meinke R. 2011. ARSSEM: Active Radiation Shield for Space Exploration Missions. Final Report ESTEC Contract No. 4200023087/10/NL/AF.

Battiston R, Burger W, Calvelli V, Datskov V, Farinon S, Musenich R. 2013. Superconducting Magnets for Astroparticle Shielding in Interplanetary Manned Missions. *IEEE Trans Appl Supercond*. **23**. DOI: [10.1109/TASC.2013.2239333](https://doi.org/10.1109/TASC.2013.2239333).

Bedingfield K, Leach R, Alexander M (Eds.). 1996. Spacecraft System Failures and Anomalies Attributed to the Natural Space Environment. National Aeronautics and Space Administration Report No. NASA-RP-1390.

- Bednorz J, Müller K. 1986. Possible High T<sub>c</sub> Superconductivity in the Ba-La-Cu-O System. *Zeitschrift Für Physik B - Condensed Matter*. **64**. DOI: [10.1007/BF01303701](https://doi.org/10.1007/BF01303701).
- Belley M, Segars W, Kapadia A. 2014. Assessment of Individual Organ Doses in a Realistic Human Phantom from Neutron and Gamma Stimulated Spectroscopy of the Breast and Liver. *Med Phys*. **41**. DOI: [10.1118/1.4873684](https://doi.org/10.1118/1.4873684).
- Bernert R, Stekly Z, Hoag E. 1964. Magnetic Radiation Shielding Systems Analysis. National Aeronautics and Space Administration Report No. NASA-CR-64915.
- Bhattacharjie A, Michael I. 1964. Mass and Magnetic Dipole Shielding Against Electrons of the Artificial Radiation Belt. *AIAA J*. **2**. DOI: [10.2514/3.2763](https://doi.org/10.2514/3.2763).
- Billings M, Yucker W. 1973. Summary Final Report: The Computerized Anatomical Man (CAM) Model. National Aeronautics and Space Administration Report No. NASA-CR-134043.
- Birch P. 1982. Radiation Shields for Ships and Settlements. *J Brit Interplanetary Soc*. **35**.
- Bolden C, Holdren J. 2010. Launching a New Era in Space Exploration. National Aeronautics and Space Administration. [https://www.nasa.gov/pdf/421063main\\_Joint\\_Statement-2-1.pdf](https://www.nasa.gov/pdf/421063main_Joint_Statement-2-1.pdf) [accessed 29 January 2020].
- Bond D, Goddard B, Singleterry R, Bilbao y León S. 2019. Evaluating the Effectiveness of Common Aerospace Materials at Lowering the Whole Body Effective Dose Equivalent in Deep Space. *Acta Astronaut*. **165**. DOI: [10.1016/j.actaastro.2019.07.022](https://doi.org/10.1016/j.actaastro.2019.07.022).
- Bonifacio D, Murata H, Moralles M. 2005. Monte Carlo Simulation of X-ray Spectra in Diagnostic Radiology and Mammography using GEANT4. Proceedings of 2005 International Nuclear Atlantic Conference, 28 August – 2 September 2005, Santo, SP, Brazil.
- Boone J, Chavez A. 1996. Comparison of X-ray Cross Sections for Diagnostic and Therapeutic Medical Physics. *Medical Physics*. **23**. DOI: [10.1118/1.597899](https://doi.org/10.1118/1.597899).

- Boone J, Siebert JA. 1997. An Accurate Method for Computer-Generating Tungsten Anode X-Ray Spectra from 30 to 140 kV. *Med Phys.* **24**. DOI: [10.1118/1.597953](https://doi.org/10.1118/1.597953).
- Borak T, Heilbronn L, Townsend L, McBeth R, de Wet W. 2014. Quality Factors for Space Radiation: A New Approach. *Life Sci Space Res.* **1**. DOI: [10.1016/j.lssr.2014.02.005](https://doi.org/10.1016/j.lssr.2014.02.005).
- Bowman C. 2013. Design Issues for Using Magnetic Materials in Radiation Environments at Elevated Temperature. Proceedings of Nuclear and Emerging Technologies for Space, 25-28 February 2013, Albuquerque, New Mexico.
- Brodin N, Vogelius I, Maraldo M, Rosenschöld P, Aznar M, Kiil-Berthelsen A, Nilsson P, Björk-Eriksson T, Specht L, Bentzen S. 2012. Life Years Lost - Comparing Potentially Fatal Late Complications After Radiotherapy for Pediatric Medulloblastoma on a Common Scale. *Cancer.* **118V.Mar**. DOI: [10.1002/cncr.27536](https://doi.org/10.1002/cncr.27536).
- Brown G. 1961. Magnetic Radiation Shielding. Proceedings of the International Conference on High-Intensity Magnetic Fields, 1-4 November 1961, Cambridge, Massachusetts.
- Bruce R, Baudouy B. 2015. Cryogenic Design of a Large Superconducting Magnet for Astroparticle Shielding on Deep Space Travel Missions. *Phys Procedia.* **67**. DOI: [10.1016/j.phpro.2015.06.085](https://doi.org/10.1016/j.phpro.2015.06.085).
- Buhler C. 2004. Analysis of a Lunar Base Electrostatic Radiation Shield Concept. National Aeronautics and Space Administration Report No. NIAC CP 04-01, Phase I.
- BV Thermal Systems. 2014. Mercury MC200-A1-E1-H2-J1 Specifications. <https://www.bvthermal.com/mercury-chillers/> [accessed 15 January 2020].
- Callaghan E, Maslen S. 1960. The Magnetic Field of a Finite Solenoid. National Aeronautics and Space Administration Report No. NASA-TN-D-365.

- Carlsen T, Peterson L, Ulsh B, Werner C, Purvis K, Sharber A. 2001. Radionuclide Contamination at Kazakhstan's Semipalatinsk Test Site: Implications on Human and Ecological Health. *Hum Ecol Risk Assess.* **7**. DOI: [10.1080/20018091094754](https://doi.org/10.1080/20018091094754).
- Carnell L, Blattnig S, Hu S, Huff J, Kim M, Norman R, Patel Z, Simonsen L, Wu H, Casey R, Cucinotta F. 2016. Evidence Report: Risk of Acute Radiation Syndromes Due to Solar Particle Events. National Aeronautics and Space Administration. <https://humanresearchroadmap.nasa.gov/Evidence/reports/Acute.pdf> [accessed 15 January 2020].
- Cassola V, de Melo Lima V, Kramer R, Khoury H. 2010. FASH and MASH: Female and Male Adult Human Phantoms Based on Polygon Mesh Surfaces: I. Development of the Anatomy. *Phys Med Biol.* **55**. DOI: [10.1088/0031-9155/55/1/009](https://doi.org/10.1088/0031-9155/55/1/009).
- Chancellor J, Scott G, Sutton J. 2014. Space Radiation: The Number One Risk to Astronaut Health beyond Low Earth Orbit. *Life.* **4**. DOI: [10.3390/life4030491](https://doi.org/10.3390/life4030491).
- Chancellor J, Blue R, Cengel K, Auñón-Chancellor S, Rubins K, Katzgraber H, Kennedy A. 2018. Limitations in Predicting the Space Radiation Health Risk for Exploration Astronauts. *NPJ Microgravity.* **4**. DOI: [10.1038/s41526-018-0043-2](https://doi.org/10.1038/s41526-018-0043-2).
- Chang-Díaz F, Squire J, Bengtson R, Breizman B, Carter M, Baity F. 2000. The Physics and Engineering of the VASIMR Engine. Proceedings of the 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, 17-19 July 2000, Huntsville, Alabama. DOI: [10.2514/6.2000-3756](https://doi.org/10.2514/6.2000-3756).
- Christofilos N. 1959. The Argus Experiment. *J Geophys Res.* **64**. DOI: [10.1029/JZ064i008p00869](https://doi.org/10.1029/JZ064i008p00869).

- Cocks F. 1991. A Deployable High Temperature Superconducting Coil (DHTSC): A Novel Concept for Producing Magnetic Shields Against Both Solar Flare and Galactic Radiation During Manned Interplanetary Missions. *J Brit Interplanetary Soc.* **44**.
- Cocks F, Watkins S. 1993. Magnetic Shielding of Interplanetary Spacecraft Against Solar Flare Radiation. National Aeronautics and Space Administration Report No. NASA-CR-195539.
- Cocks J, Watkins S, Cocks F, Sussingham C. 1997. Applications for Deployed High Temperature Superconducting Coils in Spacecraft Engineering: A Review and Analysis. *J Brit Interplanetary Soc.* **50**.
- Cohen B. 2002. Cancer Risk from Low Level Radiation. *Amer J Roentgenology.* **179**.
- Columbia Accident Investigation Board. 2003. Report of the Columbia Accident Investigation Board. [https://www.nasa.gov/columbia/home/CAIB\\_Vol1.html](https://www.nasa.gov/columbia/home/CAIB_Vol1.html) [accessed 29 January 2020].
- Cougnat C, Crosby N, Eckersley S, Foullon C, Guarnieri V, Guatelli S, Heynderickx D, Holmes-Siedle A, Lobascio C, Masiello S, Nieminen P, Parisi G, Parodi P, Perino M, Pia M, Rampini R, Spillantini P, Tamburini V, Tracino E. 2005. Radiation Exposure and Mission Strategies for Interplanetary Manned Missions (REMSIM). *Earth Moon Planets.* **94**. DOI: [10.1007/s11038-005-9014-1](https://doi.org/10.1007/s11038-005-9014-1).
- Cristy M. 1981. Active Bone Marrow Distribution as a Function of Age in Humans. *Phys Med Biol.* **26**. DOI: [10.1088/0031-9155/26/3/003](https://doi.org/10.1088/0031-9155/26/3/003).
- Cristy M, Eckerman K. 1987a. Specific Absorbed Fractions of Energy at Various Ages from Internal Photon Sources - VII Adult Male. Oak Ridge National Laboratory Report No. ORNL/TM-8381/V7.

- Cristy M, Eckerman K. 1987b. Specific Absorbed Fractions of Energy at Various Ages from Internal Photon Sources – I Methods. Oak Ridge National Laboratory Report No. ORNL/TM-8381/V1.
- Crouch R, Walberg G. 1969. An Investigation of Ablation Behavior of AVCOAT 5026/39M Over a Wide Range of Thermal Environments. National Aeronautics and Space Administration Report No. NASA-TM-X-1778.
- Cucinotta F, Manuel F, Jones T, Iszard G, Murrey J, Djojonegro B, Wear M. 2001a. Space Radiation and Cataracts in Astronauts. *Radiat Res.* **156**. DOI: [10.1667/0033-7587\(2001\)156\[0460:SRACIA\]2.0.CO;2](https://doi.org/10.1667/0033-7587(2001)156[0460:SRACIA]2.0.CO;2).
- Cucinotta F, Schimmerling W, Wilson J, Peterson L, Badhwar G, Saganti P, Dicello J. 2001b. Space Radiation Cancer Risks and Uncertainties for Mars Missions. *Radiat Res.* **156**. DOI: [10.1667/0033-7587\(2001\)156\[0682:SRCRAU\]2.0.CO;2](https://doi.org/10.1667/0033-7587(2001)156[0682:SRCRAU]2.0.CO;2).
- Cucinotta F, Schimmerling W, Wilson J, Peterson L, Saganti P, Dicello J. 2004. Uncertainties in Estimates of the Risks of Late Effects from Space Radiation. *Adv Space Res.* **34**. DOI: [10.1016/j.asr.2003.10.053](https://doi.org/10.1016/j.asr.2003.10.053).
- Cucinotta F, Kim M, Ren L. 2005. Managing Lunar and Mars Mission Radiation Risks: Part I: Cancer Risks, Uncertainties, and Shielding Effectiveness. National Aeronautics and Space Administration Report No. NASA-TP-2005-213164.
- Cucinotta F, Kim M, Chappell L. 2012a. Evaluating Shielding Approaches to Reduce Space Radiation Cancer Risks. National Aeronautics and Space Administration Report No. NASA-TM-2012-217361.
- Cucinotta F, Kim M, Chappell L. 2012b. Probability of Causation of Space Radiation Carcinogenesis Following International Space Station, Near Earth Asteroid, and Mars

Missions. National Aeronautics and Space Administration Report No. NASA-TM-2012-217357.

Cucinotta F, Kim M, Chappell L. 2013. Space Radiation Cancer Risk Projections and Uncertainties – 2012. National Aeronautics and Space Administration Report No. NASA-TP-2013-217375.

Cucinotta F, Alp M, Rowedder B, Kim M. 2015. Safe Days in Space with Acceptable Uncertainty from Space Radiation Exposure. *Life Sci Space Res.* **5**. DOI: [10.1016/j.lssr.2015.04.002](https://doi.org/10.1016/j.lssr.2015.04.002).

Dant J, Richardson R, Nie L. 2013. Monte Carlo Simulation of Age-dependent Radiation Dose from Alpha- and Beta-Emitting Radionuclides to Critical Trabecular Bone and Bone Marrow Targets. *Phys Med Biol.* **58**. DOI: [10.1088/0031-9155/58/10/3301](https://doi.org/10.1088/0031-9155/58/10/3301).

De Angelis G, Cloudsley M, Nealy J, Tripathi R, Wilson J. 2004. Radiation Analysis for Manned Missions to the Jupiter System. *Adv Space Res.* **34**. DOI: [10.1016/j.asr.2003.09.061](https://doi.org/10.1016/j.asr.2003.09.061).

de Vocht F, van Drooge H, Engels H, Kromhout H. 2006. Exposure, Health Complaints, and Cognitive Performance Among Employees of an MRI Scanners Manufacturing Department. *JMRI.* **23**. DOI: [10.1002/jmri.20485](https://doi.org/10.1002/jmri.20485).

Dow N, Shen S, Heyda J. 1962. Evaluations of Space Vehicle Shielding. GE Space Sciences Laboratory Report No. R625D31.

Durante M. 2014. Space Radiation Protection: Destination Mars. *Life Sci Space Res.* **1**. DOI: [10.1016/j.lssr.2014.01.002](https://doi.org/10.1016/j.lssr.2014.01.002).

Ehresmann B, Hassler D, Zeitlin C, Guo J, Köhler J, Wimmer-Schweingruber R, Appel J, Brinza D, Rafkin S, Böttcher S, Burmeister S, Lohf H, Martin C, Böhm E, Mattiä D, Reitz G. 2016. Charged Particle Spectra Measured During the Transit to Mars with the Mars

- Science Laboratory Radiation Assessment Detector (MSL/RAD). *Life Sci Space Res.* **10**. DOI: [10.1016/j.lssr.2016.07.001](https://doi.org/10.1016/j.lssr.2016.07.001).
- Engelberger J. 1970. Space Propulsion System. United States Patent No. 3,504,868.
- English R, Benson R, Bailey J, Barnes C. 1973. Apollo Experience Report – Protection Against Radiation. National Aeronautics and Space Administration Report No. NASA-TN-D-7080.
- Farley J, Price R. 2001. Field Just Outside a Long Solenoid. *Amer J Phys.* **69**. DOI: [10.1119/1.1362694](https://doi.org/10.1119/1.1362694).
- Ferrone K, O'Brien T.P. 2018. Proton Beam Characteristics and Experimental Conditions at MD Anderson Cancer Center. The Aerospace Corporation Report No. ATR-2018-01780.
- Finckenor M, Dooling D. 1999. Multilayer Insulation Material Guidelines. National Aeronautics and Space Administration Report No. NASA-TP-1999-209263.
- Forbush S. 1969. Variation with a Period of Two Solar Cycles in the Cosmic-Ray Diurnal Anisotropy and the Superposed Variation Correlated with Magnetic Activity. *J Geophys Res.* **74**. DOI: [10.1029/JA074i014p03451](https://doi.org/10.1029/JA074i014p03451).
- Formica D, Silvestri S. 2004. Biological Effects of Exposure to Magnetic Resonance Imaging: An Overview. *BioMed Eng Online.* **3**. DOI: [10.1186/1475-925X-3-11](https://doi.org/10.1186/1475-925X-3-11).
- Francisco D. 2015. Human Risks for Exploration Missions. Proceedings of the 2015 Aerospace Medical Association Conference, 10-14 May 2015, Orlando, Florida.
- French F. 1970. Solar Flare Radiation Protection Requirements for Passive and Active Shields. *J Spacecraft.* **7**. DOI: [10.2514/3.30043](https://doi.org/10.2514/3.30043).
- Frisina W. 1985. Optimizing Electrostatic Radiation Shielding for Manned Space Vehicles. *Acta Astronaut.* **12**. DOI: [10.1016/0094-5765\(85\)90028-1](https://doi.org/10.1016/0094-5765(85)90028-1).

- Gaza Raz, Hussein H, Patel C, Shelfer T, Murrow D, Waterman G, Milstein O, Berger T, Ackerlein J, Marsalek K, Przbyla B, Rutczynska A, Gaza Ram, Leitgab M, Lee K, Semones E, Straube U. 2018. International Science Aboard *Orion* EM-1: The Matroshka AstroRad Radiation Experiment (MARE) Payload. Proceedings of the 2018 Committee on Space Research (COSPAR), 14-22 July 2018, Pasadena, California.
- Geng C, Tang X, Gong C, Guan F, Johns J, Shu D, Chen D. 2015. A Monte Carlo-based Radiation Safety Assessment for Astronauts in an Environment with Confined Magnetic Field Shielding. *J Radiol Prot.* **35**. DOI: [10.1088/0952-4746/35/4/777](https://doi.org/10.1088/0952-4746/35/4/777).
- George K, Durante M, Wu H, Willingham V, Badhwar G, Cucinotta F. 2001. Chromosome Aberrations in the Blood Lymphocytes of Astronauts after Space Flight. *Radiat Res.* **156**. DOI: [10.1667/0033-7587\(2001\)156\[0731:CAITBL\]2.0.CO;2](https://doi.org/10.1667/0033-7587(2001)156[0731:CAITBL]2.0.CO;2).
- Gialousis G, Yakoumakis E, Dimitriadis A, Papouli Z, Yakoumakis N, Tsalafoutas I, Papadopoulou D, Georgiou E. 2008. Monte Carlo Estimation of Radiation Doses in Red Bone Marrow and Breast in Common Pediatric X-ray Examinations. *Health Phys.* **95**. DOI: [10.1097/01.HP.0000318883.79767.24](https://doi.org/10.1097/01.HP.0000318883.79767.24).
- Gieseler J, Heber B, Gómez-Herrero R, Wimmer-Schweingruber R. 2007. The Radial Gradient of Galactic Cosmic Rays: Ulysses KET and ACE CRIS Measurements. Proceedings of the 30<sup>th</sup> International Cosmic Ray Conference, 3-11 July 2007, Merida, Mexico.
- Gillin M, Sahoo N, Bues M, Ciangaru G, Sawakuchi G, Poenisch F, Arjomandy B, Martin C, Titt U, Suzuki K, Smith A, Zhu XR. 2010. Commissioning of the Discrete Spot Scanning Proton Beam Delivery System at the University of Texas M.D. Anderson Cancer Center, Proton Therapy Center, Houston. *Med Phys.* **37**. DOI: [10.1118/1.3259742](https://doi.org/10.1118/1.3259742).
- Ginsburg V. 1996. Cosmic Ray Astrophysics (History and General Review). *Physics – Uspekhi.* **39**. DOI: [10.1070/PU1996v039n02ABEH000132](https://doi.org/10.1070/PU1996v039n02ABEH000132).

GMW. 2009. User's Manual – Model 3472-50 and 3472-70 100 mm Electromagnet.

<http://www.gmw.com/product/3472> [accessed 15 January 2020].

Golge S, O'Neill P, Slaba T. 2015. NASA Galactic Cosmic Radiation Environment Model:

Badhwar-O'Neill 2014. Proceedings of the 34<sup>th</sup> International Cosmic Ray Conference, 30 July – 6 August 2015, The Hague, The Netherlands.

Good R, Shen S, Dow N. 1964. Active Shielding Concepts for the Ionizing Radiation in Space.

National Aeronautics and Space Administration Report No. NASA-CR-58950.

Gorlov T, Holmes J. 2018. Fringe Field Effect of Solenoids. Proceedings of the 9<sup>th</sup> International

Particle Accelerator Conference, 29 April – 4 May 2018, Vancouver, BC, Canada.

Grishin S, Zavadskii V, Orozodnikov S, Orlov R. 1978. Experimental Investigation of Magnetic

Shields. *Zh Tekh Fiz.* **48**.

Guthoff M, Brovchenko O, De Boer W, Dierlamm A, Müller T, Ritter A, Schmanau M, Simonis

H. 2012. GEANT4 Simulation of a Filtered X-Ray Source for Radiation Damage Studies.

*Nucl Phys A.* **675**. DOI: [10.1016/j.nima.2012.01.029](https://doi.org/10.1016/j.nima.2012.01.029).

Hamilton S, Pecaut M, Gridley D, Travis N, Bandstra E, Willey J, Nelson G, Bateman T. 2006.

A Murine Model for Bone Loss from Therapeutic and Space-Relevant Sources of Radiation. *J Appl Physiol.* **101**. DOI: [10.1152/japplphysiol.01078.2005](https://doi.org/10.1152/japplphysiol.01078.2005).

Heinbockel J, Slaba T, Blattnig S, Tripathi R, Townsend L, Handler T, Gabriel T, Pinsky L,

Reddell B, Cloudsley M, Singleterry R, Norbury J, Badavi F, Aghara S. 2009.

Comparison of Radiation Transport Codes, HZETRN, HETC and FLUKA, Using the 1956

Webber SPE Spectrum. National Aeronautics and Space Administration Report No.

NASA-TP-2009-215560.

- Heinen V, Connolly D. 1990. Aerospace Applications of High Temperature Superconductivity. Proceedings of the 41<sup>st</sup> International Astronautical Congress, 6-12 October 1990, Dresden, Germany.
- Hellweg C, Baumstark-Khan C. 2007. Getting Ready for the Manned Mission to Mars: The Astronauts' Risk from Space Radiation. *Naturwissenschaften*. **94**. DOI: [10.1007/s00114-006-0204-0](https://doi.org/10.1007/s00114-006-0204-0).
- Hendee W, Ritenour ER. 2002. Medical Imaging Physics, 4<sup>th</sup> Ed. Wiley-LISS, New York. ISBN: 0471382264.
- Herring J, Merrill B. 1991. Magnetic Shielding for Interplanetary Spacecraft. Proceedings of the 28th Space Congress, 23-26 April 1991, Cocoa Beach, Florida.
- Hilinski E, Cocks F. 1994. Deployed High-Temperature Superconducting Coil Magnetic Shield. *J Spacecraft*. **31**. DOI: [10.2514/3.26445](https://doi.org/10.2514/3.26445).
- Howell E. 2018a. *Orion* Spacecraft: Taking Astronauts Beyond Earth Orbit. Space.com, 28 November 2018. <https://www.space.com/27824-orion-spacecraft.html> [accessed 14 February 2020].
- Howell E. 2018b. NASA's *Orion* Spacecraft Gets Heat Shield for Daring Test Flight to the Moon. Space.com, 21 August 2018. <https://www.space.com/41566-nasa-orion-spacecraft-heat-shield-photo.html> [accessed 15 January 2020].
- Huff J, Carnell L, Blattnig S, Chappell L, George K, Lumpkins S, Simonsen L, Slaba T, Werneth C, Cucinotta F, Durante M. 2016. Evidence Report: Risk of Radiation Carcinogenesis. National Aeronautics and Space Administration. <https://humanresearchroadmap.nasa.gov/evidence/reports/Cancer.pdf> [accessed 15 January 2020].

Huff J, Patel Z, Simonsen L. 2017. NASA Space Radiation Protection Strategies – Risk Assessment and Permissible Exposure Limits. National Aeronautics and Space Administration.

IBA Dosimetry. 2013. Zebra with OmniPro-Incline. <http://www.iba-dosimetry.com/product/zebra> [accessed 15 January 2020].

IBA Dosimetry. 2016. Efficient Patient QA – Hardware. <http://www.iba-dosimetry.com> [accessed 15 January 2020].

ICNIRP. 1994. Guidelines on Limits of Exposure of Static Magnetic Fields. International Commission on Non-Ionizing Radiation Protection.

ICRP. 1975. Report of the Task Group on Reference Man. International Commission on Radiation Protection Publication No. 23.

ICRP. 1991. The 1990 Recommendations of the International Commission on Radiological Protection. International Commission on Radiological Protection Publication No. 60.

ICRP. 2002. Basic Anatomical and Physiological Data for Use in Radiological Protection: Reference Values. International Commission on Radiological Protection Publication No. 89.

ICRP. 2003. Relative Biological Effectiveness (RBE), Quality Factor (Q), and Radiation Weighting Factor ( $w_R$ ). International Commission on Radiological Protection Publication No. 92.

ICRP. 2007. The 2007 Recommendations of the International Commission on Radiological Protection. International Commission on Radiological Protection Publication No. 103.

ICRP. 2009. Adult Reference Computational Phantoms. International Commission on Radiological Protection Publication No. 110.

- ICRP. 2012. Compendium of Dose Coefficients Based on ICRP Publication 60. International Commission on Radiological Protection Publication No. 119.
- ICRP. 2013. Assessment of Radiation Exposure of Astronauts in Space. International Commission on Radiological Protection Publication No. 123.
- ICRU. 1992. Phantoms and Computational Models in Therapy, Diagnosis, and Protection. International Commission on Radiation Units and Measurements Report No. 48.
- ISO. 2004. Space Environment (Natural and Artificial) – Galactic Cosmic Ray Model. International Organization for Standardization Report No. ISO15390.
- Ivantchenko A, Ivantchenko V, Quesada Molina J, Incerti S. 2012. GEANT4 Hadronic Physics for Space Radiation Environment. *Int J Radiat Biol.* **88**.  
DOI: [10.3109/09553002.2011.610865](https://doi.org/10.3109/09553002.2011.610865).
- Javad Mortazavi S, Cameron J, Niroomand-rad A. 2003. Adaptive Response Studies May Help Choose Astronauts for Long-Term Space Travel. *Adv Space Res.* **31**.  
DOI: [10.1016/S0273-1177\(03\)00089-9](https://doi.org/10.1016/S0273-1177(03)00089-9).
- Johns H, Cunningham J. 1983. The Physics of Radiology, 4<sup>th</sup> Ed. Charles River Media, Hingham, Massachusetts. ISBN: 0398046697.
- Joshi R, Qiu H, Tripathi R. 2013. Evaluation of a Combined Electrostatic and Magnetostatic Configuration for Active Space-Radiation Shielding. *Adv Space Res.* **51**.  
DOI: [10.1016/j.asr.2012.12.016](https://doi.org/10.1016/j.asr.2012.12.016).
- Kamide Y, Chian A (Eds.). 2007. Handbook of the Solar-Terrestrial Environment. Springer, New York. ISBN: 9783540463153.
- Kash S. 1963. Minimum Structural Mass for a Magnetic Radiation Shield. *AIAA J.* **1**. DOI: [10.2514/3.1826](https://doi.org/10.2514/3.1826).

- Kash S. 1965. Magnetic Space Shields. Proceedings of the 6th Biennial Gas Dynamics Symposium, 25-27 August 1965, Evanston, Illinois. DOI: [10.2514/6.1965-629](https://doi.org/10.2514/6.1965-629).
- Kash S, Tooper R. 1962. Active Shielding for Manned Spacecraft. *Astronautics*. **7**.
- Kennedy A, Wan X. 2011. Countermeasures for Space Radiation Induced Adverse Biologic Effects. *Adv Space Res*. **48**. DOI: [10.1016/j.asr.2011.07.007](https://doi.org/10.1016/j.asr.2011.07.007).
- Kennedy A. 2014. Biological Effects of Space Radiation and Development of Effective Countermeasures. *Life Sci Space Res*. **1**. DOI: [10.1016/j.lssr.2014.02.004](https://doi.org/10.1016/j.lssr.2014.02.004).
- Kennedy J. 1962. Address at Rice University on the Nation's Space Effort. <https://www.jfklibrary.org/learn/about-jfk/historic-speeches/address-at-rice-university-on-the-nations-space-effort> [accessed 29 January 2020].
- Kervendal E, Kirk D, Meinke R. 2009. Spacecraft Radiation Shielding Using Ultralightweight Superconducting Magnets. *J Spacecraft Rockets*. **46**. DOI: [10.2514/1.37490](https://doi.org/10.2514/1.37490).
- Kim M, De Angelis G, Cucinotta F. 2011. Probabilistic Assessment of Radiation Risk for Astronauts in Space Missions. *Acta Astronaut*. **68**. DOI: [10.1016/j.actaastro.2010.08.035](https://doi.org/10.1016/j.actaastro.2010.08.035).
- Kinouchi Y, Yamaguchi H, Tenforde T. 1996. Theoretical Analysis of Magnetic Field Interactions with Aortic Blood Flow. *Bioelectromagnetics*. **17**. DOI: [10.1002/\(SICI\)1521-186X\(1996\)17:1<21::AID-BEM3>3.0.CO;2-8](https://doi.org/10.1002/(SICI)1521-186X(1996)17:1<21::AID-BEM3>3.0.CO;2-8).
- Kinstler G. 2012. Spacecraft Having a Magnetic Space Radiation Shield. United States Patent No. 8,210,481 B2.
- Koons H, Mazur J, Selesnick R, Blake J, Fennell J, et al. 1999. The Impact of the Space Environment on Space Systems. The Aerospace Corporation Report No. TOR-99(1670)-1.
- Koontz S, Atwell W, Rojdev K, Valle G. 2013. Evaluating the Effects of Astronaut Career Ionizing Radiation Dose Limits on Manned Interplanetary Flight Programs. Proceedings

- of the 43<sup>rd</sup> International Conference on Environmental Systems, 14-18 July 2013, Vail, Colorado. DOI: [10.2514/6.2013-3405](https://doi.org/10.2514/6.2013-3405).
- Kramer R, Vieira J, Khoury H, Lima F, Lima V, Fuelle D, Hoff G. 2003. All About MAX: A Male Adult Voxel Phantom for Monte Carlo Calculations in Radiation Protection Dosimetry. *Phys Med Biol.* **48**. DOI: [10.1088/0031-9155/48/10/301](https://doi.org/10.1088/0031-9155/48/10/301).
- Kramer R, Khoury H, Vieira J, Loureiro E, Lima V, Lima F, Hoff G. 2004. All About FAX: A Female Adult Voxel Phantom for Monte Carlo Calculation in Radiation Protection Dosimetry. *Phys Med Biol.* **49**. DOI: [10.1088/0031-9155/49/23/001](https://doi.org/10.1088/0031-9155/49/23/001).
- Kvinnslund Y, Skretting A, Bruland Ø. 2001. Radionuclide Therapy with Bone-seeking Compounds: Monte Carlo Calculations of Dose-Volume Histograms for Bone Marrow in Trabecular Bone. *Phys Med Biol.* **46**. DOI: [10.1088/0031-9155/46/4/317](https://doi.org/10.1088/0031-9155/46/4/317).
- Landis G. 1991. Magnetic Radiation Shielding: An Idea Whose Time Has Returned? Proceedings of the 10th Biennial AIAA Conference on Space Manufacturing, 15-19 May 1991, Princeton, New Jersey. DOI: [10.1063/1.42056](https://doi.org/10.1063/1.42056).
- Langell J, Jennings R, Clark J, Ward J. 2008. Pharmacological Agents for the Prevention and Treatment of Toxic Radiation Exposure in Spaceflight. *Aviat Space Environ Med.* **79**. DOI: [10.3357/ASEM.2113.2008](https://doi.org/10.3357/ASEM.2113.2008).
- Letaw J, Silberberg R, Tsao C. 1989. Radiation Hazards on Space Missions Outside the Magnetosphere. *Adv Space Res.* **9**. DOI: [10.1016/0273-1177\(89\)90451-1](https://doi.org/10.1016/0273-1177(89)90451-1).
- Levine S, Bhattacharjie A, Lepper R. 1966. Forbidden Regions Produced by Two Parallel Dipoles. *AIAA J.* **4**. DOI: [10.2514/3.3503](https://doi.org/10.2514/3.3503).
- Levine S, Lepper R. 1968a. The Quasi-Hollow Conductor Magnet As a Space Shield Against Electrons. National Aeronautics and Space Administration Report No. NASA-189-214.

- Levine S, Lepper R. 1968b. A Study of Charged Particle Motion in Magnetic Radiation Shielding Fields. National Aeronautics and Space Administration Report No. NASA-CR-98074.
- Levine S, Lepper R. 1971. An Active Radiation Shield for Cylindrically Shaped Vehicles. *J Spacecraft*. **8**. DOI: [10.2514/3.30319](https://doi.org/10.2514/3.30319).
- Levy R. 1961. Radiation Shielding of Space Vehicles By Means of Superconducting Coils. United States Air Force Report No. USAF-RP-106.
- Levy R. 1962. The Prospects for Active Shielding. AVCO-Everett Research Laboratory Report No. AMP-94.
- Levy R, French F. 1967. The Plasma Radiation Shield: Concept and Applications to Space Vehicles. National Aeronautics and Space Administration Report No. NASA-CR-61176.
- Logsdon, J (Ed.). 1995. Exploring the Unknown: Selected Documents in the History of the U.S. Civil Space Program, Vol. I: Organizing for Exploration. National Aeronautics and Space Administration Report No. NASA-SP-4407.
- Manning B, Singleterry R. 2020. Radiation Engineering Analysis of Shielding Materials to Assess Their Ability to Protect Astronauts in Deep Space from Energetic Particle Radiation. *Acta Astronaut*. **171**. DOI: [10.1016/j.actaastro.2020.02.020](https://doi.org/10.1016/j.actaastro.2020.02.020).
- Martins M, Silva R, Begalli M, Queiroz Filho P, Souza-Santos D, Pia M. 2009. Anthropomorphic Phantoms and GEANT4-Based Implementations for Dose Calculation. Proceedings of the 2009 IEEE Nuclear Science Symposium, 25-31 October 2009, Orlando, Florida. DOI: [10.1109/NSSMIC.2009.5401651](https://doi.org/10.1109/NSSMIC.2009.5401651).
- McDonald P. 1965. An Annotated Bibliography of Motion of Charged Particles in Magnetic Fields and Magnetic Shielding Against Space Radiation. National Aeronautics and Space Administration Report No. NASA-TN-R-161.

- McDonald P, Buntyn T. 1966. Space Radiation Shielding with the Magnetic Field of a Cylindrical Solenoid. National Aeronautics and Space Administration Report No. NASA-TN-R-203.
- McKenna-Lawlor S. 2014. Feasibility Study of Astronaut Standardized Career Dose Limits in LEO and the Outlook for BLEO. *Acta Astronaut.* **104**. DOI: [10.1016/j.actaastro.2014.07.011](https://doi.org/10.1016/j.actaastro.2014.07.011).
- McLauchlan K. 1981. The Effects of Magnetic Fields on Chemical Reactions. *Sci Prog (Oxford)*. **67**.
- McLaughlin M, Donoviel D, Jones J. 2017. Novel Indications for Commonly Used Medications as Radiation Protectants in Spaceflight. *Aerospace Med Human Perf.* **88**. DOI: [10.3357/AMHP.4735.2017](https://doi.org/10.3357/AMHP.4735.2017).
- Milstein O, Levitt D, Tikochinsky Y, Zur Ey, Zur Er. 2015. Radiation Protection Device and Methods Thereof. United States Patent No. 2015/0004131 A1.
- Morozov D, Riabova T, Trukhanov K, Sedin G, Tsetlin V. 1971. Some Aspects of Active Shielding Against the Radiation in Space. Proceedings of the International Congress on the Protection Against Accelerator and Space Radiation, 26-30 April 1971, Geneva, Switzerland.
- Moses R, Bushnell D, Komar D, Choi S, Singleterry R, Litchford R, Chang-Diaz F, Carter M. 2018. Maintaining Human Health for Humans-Mars. Proceedings of the AIAA 2018 Conference and Exposition, 17-19 September 2018, Orlando, Florida. DOI: [10.2514/6.2018-5360](https://doi.org/10.2514/6.2018-5360).
- Musenich R, Calvelli V, Farinon S, Battiston R, Burger W, Spillantini P. 2014. A Magnesium Diboride Superconducting Toroid for Astroparticle Shielding. *IEEE Trans Appl Supercond.* **24**. DOI: [10.1109/TASC.2013.2287047](https://doi.org/10.1109/TASC.2013.2287047).

- Musenich R, Calvelli V, Giraudo M, Vuolo M, Ambroglini F, Battiston R. 2018. The Limits of Space Radiation Magnetic Shielding: An Updated Analysis. *IEEE Trans Appl Supercond.* **28**. DOI: [10.1109/TASC.2017.2785805](https://doi.org/10.1109/TASC.2017.2785805).
- NASA. 1970. NASA Space Vehicle Design Criteria: Space Radiation Protection. National Aeronautics and Space Administration Report No. NASA-SP-8054.
- NASA. 1990. Report of the 90-Day Study on Human Exploration of the Moon and Mars. National Aeronautics and Space Administration Report No. NASA-TM-102999. [https://history.nasa.gov/90\\_day\\_study.pdf](https://history.nasa.gov/90_day_study.pdf) [accessed 29 January 2020].
- NASA. 2001. Home, Space Home. National Aeronautics and Space Administration. [https://science.nasa.gov/science-news/science-at-nasa/2001/ast14mar\\_1/](https://science.nasa.gov/science-news/science-at-nasa/2001/ast14mar_1/) [accessed 15 January 2020].
- NASA. 2004. The Vision for Space Exploration. National Aeronautics and Space Administration. [https://www.nasa.gov/pdf/55583main\\_vision\\_space\\_exploration2.pdf](https://www.nasa.gov/pdf/55583main_vision_space_exploration2.pdf) [accessed 29 January 2020].
- NASA. 2005. Super Lightweight External Tank. National Aeronautics and Space Administration Report No. NASA-FS-2005-04-025-MSFC.
- NASA. 2009. Human Exploration of Mars: Design Reference Architecture 5.0. National Aeronautics and Space Administration Report No. NASA-SP-2009-566.
- NASA. 2010. Human Integration Design Handbook (HIDH). National Aeronautics and Space Administration Report No. NASA-SP-2010-3407.
- NASA. 2011. *Orion* Quick Facts. National Aeronautics and Space Administration Report No. NASA-FS-2011-12-058-JSC.
- NASA. 2013. Badhwar-O'Neill 2011 Galactic Cosmic Ray Flux Model Description. National Aeronautics and Space Administration Report No. NASA-TP-2013-217376.

- NASA. 2015a. NASA's Journey to Mars: Pioneering Next Steps in Space Exploration. National Aeronautics and Space Administration.  
[https://www.nasa.gov/sites/default/files/atoms/files/journey-to-mars-next-steps-20151008\\_508.pdf](https://www.nasa.gov/sites/default/files/atoms/files/journey-to-mars-next-steps-20151008_508.pdf) [accessed 14 February 2020].
- NASA. 2015b. NASA Space Flight Human-System Standard Volume 1, Revision A: Crew Health. National Aeronautics and Space Administration Report No. NASA-STD-3001 Vol. 1 Rev. A w/Change 1.
- NASA. 2015c. *Destiny* Laboratory. National Aeronautics and Space Administration.  
[https://www.nasa.gov/mission\\_pages/station/structure/elements/us-destiny-laboratory/](https://www.nasa.gov/mission_pages/station/structure/elements/us-destiny-laboratory/) [accessed 15 January 2020].
- NASA. 2015d. Reference Guide to the International Space Station. National Aeronautics and Space Administration Report No. NASA-NP-2015-05-022-JSC.
- NCRP. 1989. Guidance on Radiation Received in Space Activities. National Council on Radiation Protection and Measurements Report No. 98.
- NCRP. 1993. Limitation of Exposure to Ionizing Radiation. National Council on Radiation Protection and Measurements Report No. 116.
- NCRP. 1997. Uncertainties in Fatal Cancer Risk Estimates Used in Radiation Protection. National Council on Radiation Protection and Measurements Report No. 126.
- NCRP. 2000. Radiation Protection Guidance for Activities in Low Earth Orbit. National Council on Radiation Protection and Measurements Report No. 132.
- NCRP. 2014. Radiation Protection for Space Activities: Supplement to Previous Recommendations. National Council on Radiation Protection and Measurements Commentary No. 23.

Nelson G, Simonsen L, Huff J, Cucinotta F, Kim M, Saha J, Wang M, Sulzman F. 2016.

Evidence Report: Risk of Acute and Late Central Nervous System Effects from Radiation Exposure. National Aeronautics and Space Administration.

<https://humanresearchroadmap.nasa.gov/Evidence/reports/CNS.pdf> [accessed 29 January 2020].

Neuberth G, Westphal M. 2011. Compact Superconducting Magnet Configuration with Active Shielding Having a Shielding Coil Contributing to Field Formation. United States Patent No. 7898258 B2.

Nie H, Richardson R. 2009. Radiation Dose to Trabecular Bone Marrow Stem Cells from  $^3\text{H}$ ,  $^{14}\text{C}$  and Selected  $\alpha$ -emitters Incorporated in a Bone Remodeling Compartment. *Phys Med Biol*. **54**. DOI: [10.1088/0031-9155/54/4/010](https://doi.org/10.1088/0031-9155/54/4/010).

Nixon R. 1972. Announcement on the Space Shuttle. <https://history.nasa.gov/stsnixon.htm> [accessed 29 January 2020].

Norwood J, Gibbons F. 1963. Studies of Magnetic Shielding and Superconductivity. General Dynamics Report No. AD-423178.

Nose T, Yamaguchi T, Yamamoto M, Shiraishi K, Shiraishi T, Koinuma H. 1990. Magnetic Permeability and Antiferromagnetism of YBCO Superconductors. *J Adv Sci*. **2**. DOI: [10.2978/jsas.2.108](https://doi.org/10.2978/jsas.2.108).

Novikov L, Mileev V, Voronina E, Galanina L, Makletsov A, Sinolits V. 2009. Radiation Effects on Spacecraft Materials. *J Surf Invest: X-Ray, Synchrotron, Neutron Tech*. **3**. DOI: [10.1134/S1027451009020062](https://doi.org/10.1134/S1027451009020062).

NRC. 1998. Effects of Ionizing Radiation: Atomic Bomb Survivors and Their Children (1945-1995). National Research Council. ISBN: 978-0-309-06402-6.

- NRC. 2006. Health Risks from Exposure to Low Levels of Ionizing Radiation, BEIR VII, Phase 2. National Research Council. ISBN: 978-0-309-09156-5.
- NRC. 2008. Managing Space Radiation Risk in the New Era of Space Exploration. National Research Council. ISBN: 978-0-309-11384-9.
- NRC. 2012. Technical Evaluation of the NASA Model for Cancer Risk to Astronauts Due to Space Radiation. National Research Council. ISBN: 978-0-309-25305-5.
- O'Neill P. 2006. Badhwar-O'Neill Galactic Cosmic Ray Model Update Based on Advanced Composition Explorer (ACE) Energy Spectra from 1997 to Present. *Adv Space Res.* **37**. DOI: [10.1016/j.asr.2005.02.001](https://doi.org/10.1016/j.asr.2005.02.001).
- O'Neill P, Golge S, Slaba T. 2015. Badhwar-O'Neill 2014 Galactic Cosmic Ray Flux Model Description. National Aeronautics and Space Administration Report No. NASA-TP-2015-218569.
- Onnes H. 1911. Further Experiments with Liquid Helium. *Leiden Comm.* **120b, 122b**. DOI: [10.1007/978-94-009-2079-8\\_10](https://doi.org/10.1007/978-94-009-2079-8_10).
- Paluszek M. Magnetic Radiation Shielding for Permanent Space Habitats. *Adv Astronaut Sci.* **36**.
- Papini P, Spillantini P. 2014. Toroidal Magnetic Fields for Protecting Astronauts from Ionizing Radiation in Long Duration Deep Space Missions. *Acta Astronaut.* **104**. DOI: [10.1016/j.actaastro.2014.06.011](https://doi.org/10.1016/j.actaastro.2014.06.011).
- Parihar V, Allen B, Tran K, Macaraeg T, Chu E, Kwok S, Chmielewski N, Craver B, Baulch J, Acharya M, Cucinotta F, Limoli C. 2015. What Happens to Your Brain on the Way to Mars. *Sci Adv.* **1**. DOI: [10.1126/sciadv.1400256](https://doi.org/10.1126/sciadv.1400256).
- Parker E. 2005. Shielding Space Explorers From Cosmic Rays. *Space Weather.* 18 August 2005.

- Parker E. 2006. Shielding Space Travelers. *Sci Amer.* **294**. DOI: [10.1038/scientificamerican0306-40](https://doi.org/10.1038/scientificamerican0306-40).
- Patel Z, Huff J, Saha J, Wang M, Blattnig S, Wu H, Cucinotta F. 2016. Evidence Report: Risk of Cardiovascular Disease and Other Degenerative Tissue Effects from Radiation Exposure. National Aeronautics and Space Administration. <https://humanresearchroadmap.nasa.gov/Evidence/reports/Degen.pdf> [accessed 29 January 2020].
- Peroni M. 2017. Solenoid Moon-Base Concept. Proceedings of the AIAA SPACE Conference and Exposition, 12-14 September 2017, Orlando, Florida. DOI: [10.2514/6.2017-5205](https://doi.org/10.2514/6.2017-5205).
- Peterson L, Pepper L, Hamm P, Gilbert S. 1993. Longitudinal Study of Astronaut Health: Mortality in the Years 1959-1991. *Radiat Res.* **133**. DOI: [10.2307/3578364](https://doi.org/10.2307/3578364).
- Peterson L, Cucinotta F. 1999. Monte Carlo Mixture Model of Lifetime Cancer Incidence Risk from Radiation Exposure on Shuttle and International Space Station. *Mutat Res.* **430**. DOI: [10.1016/S0027-5107\(99\)00145-1](https://doi.org/10.1016/S0027-5107(99)00145-1).
- Peterson L, Miller R. 2008. Association Between Radioactive Fallout from 1951-1962 US Nuclear Tests at the Nevada Test Site and Cancer Mortality in Midwestern US Populations. *Russian J Ecol.* **39**. DOI: [10.1134/S1067413608070060](https://doi.org/10.1134/S1067413608070060).
- Peterson L, Kovyreshina T. 2015. Adjustment of Lifetime Risks of Space Radiation-Induced Cancer by the Healthy Worker Effect and Cancer Misclassification. *Heliyon.* **1**. DOI: [10.1016/j.heliyon.2015.e00048](https://doi.org/10.1016/j.heliyon.2015.e00048).
- Portree D. 2001. Human Missions to Mars: Fifty Years of Mission Planning, 1950-2000. National Aeronautics and Space Administration Report No. NASA-SP-2001-4521.
- President's Commission on Implementation of United States Space Exploration Policy. 2004. A Journey to Inspire, Innovate, and Discover.

[https://www.nasa.gov/pdf/60736main\\_M2M\\_report\\_small.pdf](https://www.nasa.gov/pdf/60736main_M2M_report_small.pdf) [accessed 29 January 2020].

Presidential Commission on the Space Shuttle *Challenger* Accident. 1986. Report to the President. <https://history.nasa.gov/rogersrep/genindex.htm> [accessed 29 January 2020].

Radcliffe T, Bearden D, Judnick D. 2016. 2021 Human Mars Flyby Mission Independent Assessment Final Report. The Aerospace Corporation Report No. ATR-2016-0068.

Reagan R. 1983. Space Station. National Security Decision Directive 5-83. <https://www.hq.nasa.gov/office/pao/History/nsdd-83.html> [accessed 29 January 2020].

Reagan R. 1984. State of the Union Address. <https://history.nasa.gov/reagan84.htm> [accessed 29 January 2020].

Reetz A, O'Brien K (Eds.). 1968. Protection Against Space Radiation. National Aeronautics and Space Administration Report No. NASA-SP-169.

Review of U.S. Human Spaceflight Plans Committee. 2009. Seeking a Human Spaceflight Program Worthy of a Great Nation. [https://www.nasa.gov/pdf/617036main\\_396093main\\_HSF\\_Cmte\\_FinalReport.pdf](https://www.nasa.gov/pdf/617036main_396093main_HSF_Cmte_FinalReport.pdf) [accessed 29 January 2020].

Roberts D. 2011. The Evolution of the Spectrum of Solar Wind Velocity Fluctuations from 0.3 to 5 AU. National Aeronautics and Space Administration Report No. GSFC-JA-4653-2011.

Roig A, High S, Minna J, Shay J, Rusek A, Story M. 2010. DNA Damage Intensity in Fibroblasts in a 3D Collagen Matrix Correlates with the Bragg Curve Energy Distribution of a High LET Particle. *Int J Radiat Biol.* **86**. DOI: [10.3109/09553000903418603](https://doi.org/10.3109/09553000903418603).

Rojdev K, Atwell W, Wilkins R, Gersey B, Badavi F. 2009. Evaluation of Multi-Functional Materials for Deep Space Radiation Shielding. Proceedings of the National Space and Missile Materials Symposium, 22-26 June 2009, Henderson, Nevada.

- Sahoo N, Zhu X, Arjomandy B, Ciangaru G, Lii M, Amos R, Wu R, Gillin M. 2008. A Procedure for Calculation of Monitor Units for Passively Scattered Proton Radiotherapy Beams. *Med Phys*. **35**. DOI: [10.1118/1.2992055](https://doi.org/10.1118/1.2992055).
- Schenck J, Dumoulin C, Redington R, Kressel H, Elliott R, McDougall I. 1992. Human Exposure to 4.0-Tesla Magnetic Fields in a Whole-Body Scanner. *Med Phys*. **19**. DOI: [10.1118/1.596827](https://doi.org/10.1118/1.596827).
- Schenck J. 2000. Safety of Strong, Static Magnetic Fields. *JMRI*. **12**. DOI: [10.1002/1522-2586\(200007\)12:1<2::AID-JMRI2>3.0.CO;2-V](https://doi.org/10.1002/1522-2586(200007)12:1<2::AID-JMRI2>3.0.CO;2-V).
- Segars W, Tsui B. 2009. MCAT to XCAT: The Evolution of 4D Computerized Phantoms for Imaging Research. *Proceedings of the IEEE*. **97**. DOI: [10.1109/JPROC.2009.2022417](https://doi.org/10.1109/JPROC.2009.2022417).
- Semones E. 2015a. Living in a Radiation (Space) Field. Proceedings of Pumps and Pipes, 7 December 2015, Houston, Texas.
- Semones E. 2015b. Radiation Health Risk Projections. NASA briefing to the NAC HEOMD/SMD Joint Committee.
- Setlow R. 1999. The U.S. National Research Council's Views of the Radiation Hazards in Space. *Mutat Res*. **430**. DOI: [10.1016/S0027-5107\(99\)00127-X](https://doi.org/10.1016/S0027-5107(99)00127-X).
- Shea M, Smart D. 1990. A Summary of Major Solar Proton Events. *Solar Phys*. **127**. DOI: [10.1007/BF00152170](https://doi.org/10.1007/BF00152170).
- Shepherd S, Shepherd J. 2009. Toroidal Magnetic Spacecraft Shield Used to Deflect Energetic Charged Particles. *J Spacecraft Rockets*. **46**. DOI: [10.2514/1.37727](https://doi.org/10.2514/1.37727).
- Shinn J, Nealy J, Townsend L, Wilson J, Wood J. 1994. Galactic Cosmic Ray Radiation Levels in Spacecraft on Interplanetary Missions. *Adv Space Res*. **14**. DOI: [10.1016/0273-1177\(94\)90551-7](https://doi.org/10.1016/0273-1177(94)90551-7).

- Siegel R, Miller K, Jamal A. 2019. Cancer Statistics, 2019. *CA: A Cancer Journal for Clinicians*. **69**. DOI: [10.3322/caac.21551](https://doi.org/10.3322/caac.21551).
- Simonsen L. 2017. NASA's Mission and Roadmap. Proceedings of the 2017 NASA Space Radiation Summer School, Brookhaven National Laboratory, Upton, New York.
- Simonsen L, Nealy J, Townsend L, Wilson J. 1990. Radiation Exposure for Manned Mars Surface Missions. National Aeronautics and Space Administration Report No. NASA-TP-2979.
- Simonsen L, Nealy J. 1991. Radiation Protection for Human Missions to the Moon and Mars. National Aeronautics and Space Administration Report No. NASA-TP-3079.
- Sinclair W. 2000. Dose Limits for Astronauts. *Health Phys.* **79**. DOI: [10.1097/00004032-200011000-00017](https://doi.org/10.1097/00004032-200011000-00017).
- Singleterry R. 2013. Radiation Engineering Analysis of Shielding Materials to Assess Their Ability to Protect Astronauts in Deep Space from Energetic Particle Radiation. *Acta Astronaut.* **91**. DOI: [10.1016/j.actaastro.2013.04.013](https://doi.org/10.1016/j.actaastro.2013.04.013).
- Singleterry R, Bollweg K, Martin T, Westover S, Battiston R, et al. 2015. A Launch Requirements Trade Study for Active Space Radiation Shielding for Long Duration Human Missions. National Aeronautics and Space Administration Report No. NASA-TP-2015-218689.
- Slaba T, Qualls G, Cloudsley M, Blattnig S, Simonsen L, Walker S, Singleterry R. 2009. Analysis of Mass Averaged Tissue Doses in CAM, CAF, MAX, and FAX. National Aeronautics and Space Administration Report No. NASA-TP-2009-215562.
- Slaba T, Mertens C, Blattnig S. 2013. Radiation Shielding Optimization on Mars. National Aeronautics and Space Administration Report No. NASA-TP-2013-217893.

- Slaba T, Bahadori A, Reddell B, Singleterry R, Cloudsley M, Blattnig S. 2017. Optimal Shielding Thickness for Galactic Cosmic Ray Environments. *Life Sci Space Res.* **12**. DOI: [10.1016/j.lssr.2016.12.003](https://doi.org/10.1016/j.lssr.2016.12.003).
- Snyder W, Ford M, Warner G, Watson B. 1974. A Tabulation of Dose Equivalent per Microcurie-Day for Source and Target Organs of an Adult for Various Radionuclides. Oak Ridge National Laboratory Report No. ORNL-5000.
- Snyder W, Ford M, Warner G. 1978. Estimates of Specific Absorbed Fractions for Photon Sources Uniformly Distributed in Various Organs of a Heterogeneous Phantom. Medical Internal Radiation Dose (MIRD) Pamphlet No. 5.
- Space Task Group. 1969. The Post-Apollo Space Program: Directions for the Future. <https://history.nasa.gov/taskgrp.html> [accessed 29 January 2020].
- Spillantini P. 2000. Radiation Shielding of Spacecraft's in Manned Interplanetary Flights. *Nucl Phys B.* **85**. DOI: [10.1016/S0920-5632\(00\)00475-8](https://doi.org/10.1016/S0920-5632(00)00475-8).
- Spillantini P, Casolino M, Durante M, Mueller-Mellin R, Reitz G, Rossi L, Shurshakov V, Sorbi M. 2007. Shielding from Cosmic Radiation for Interplanetary Missions: Active and Passive Methods. *Radiat Meas.* **42**. DOI: [10.1016/j.radmeas.2006.04.028](https://doi.org/10.1016/j.radmeas.2006.04.028).
- Spillantini P. 2010. Active Shielding for Long Duration Interplanetary Manned Missions. *Adv Space Res.* **45**. DOI: [10.1016/j.asr.2010.01.025](https://doi.org/10.1016/j.asr.2010.01.025).
- Spillantini P. 2011. Superconducting Magnets and Mission Strategies for Protection from Ionizing Radiation in Interplanetary Manned Missions and Interplanetary Habitats. *Acta Astronaut.* **68**. DOI: [10.1016/j.actaastro.2010.07.023](https://doi.org/10.1016/j.actaastro.2010.07.023).
- Spillantini P. 2014. Manned Exploration and Exploitation of Solar System: Passive and Active Shielding for Protecting Astronauts from Ionizing Radiation - A Short Overview. *Acta Astronaut.* **104**. DOI: [10.1016/j.actaastro.2014.05.017](https://doi.org/10.1016/j.actaastro.2014.05.017).

- Sprawls, P. 1995. The Physical Principles of Medical Imaging. Medical Physics Publishing, Madison, Wisconsin. ISBN: 9780944838549.
- Sullivan D (Ed.). 1978. The Role of Superconductivity in the Space Program: An Assessment of Present Capabilities and Future Potential. National Aeronautics and Space Administration Report No. NASA-CR-158701.
- Sun S, Schillier J, Gazdar A. 2007. Lung Cancer in Never Smokers – A Different Disease. *Nature Rev Cancer*. **7**. DOI: [10.1038/nrc2190](https://doi.org/10.1038/nrc2190).
- Sussingham J, Watkins S, Cocks F. 1999. Forty Years of Development of Active Systems for Radiation Protection of Spacecraft. *J Astronaut Sci*. **47**.
- Swenberg C, Horneck G, Stassinopoulos E (Eds). 1993. Biological Effects and Physics of Solar and Galactic Cosmic Radiation. NATO ASI Series A: Life Sciences Vol. 243A-B.
- Synthesis Group on America's Space Exploration Initiative. 1991. America at the Threshold: America's Space Exploration Initiative. *Space Policy* **7**(3). DOI: [10.1016/0265-9646\(91\)90008-6](https://doi.org/10.1016/0265-9646(91)90008-6).
- Tang S, Smith D. 2010. GEANT4 Simulations of Gamma Ray Emissions from Accelerated Particles in Solar Flares. *Astrophys J*. **721**. DOI: [10.1088/0004-637X/721/2/1174](https://doi.org/10.1088/0004-637X/721/2/1174).
- Tinkham M. 1996. Introduction to Superconductivity (2nd Ed.). Mineola, New York: Dover Publications, Inc. ISBN: 0486435032.
- Togawa T, Okai O, Oshima M. 1967. Observation of Blood Flow EMF in Externally Applied Magnetic Field by Surface Electrodes. *Med Biol Eng*. **5**. DOI: [10.1007/BF02474505](https://doi.org/10.1007/BF02474505).
- Townsend L. 1983. HZE Particle Shielding Using Confined Magnetic Fields. *J Spacecraft Rockets*. **20**. DOI: [10.2514/3.8597](https://doi.org/10.2514/3.8597).
- Townsend L. 1984. Galactic Heavy-Ion Shielding Using Electrostatic Fields. National Aeronautics and Space Administration Report No. NASA-TM-86265.

- Townsend L. 2000. Overview of Active Methods for Shielding Spacecraft from Energetic Space Radiation. Proceedings of the 1st International Workshop on Space Radiation Research, 27-31 May 2000, Arona, Italy.
- Townsend L. 2005a. Critical Analysis of Active Shielding Methods of Space Radiation Protection. Proceedings of the 2005 IEEE Aerospace Conference, 5-12 March 2005, Big Sky, Montana. DOI: [10.1109/AERO.2005.1559364](https://doi.org/10.1109/AERO.2005.1559364).
- Townsend L. 2005b. Implications of the Space Radiation Environment for Human Exploration in Deep Space. *Radiat Prot Dosim.* **115**. [10.1093/rpd/nci141](https://doi.org/10.1093/rpd/nci141).
- Tripathi R, Wilson J, Youngquist R. 2008. Electrostatic Space Radiation Shielding. *Adv Space Res.* **42**. DOI: [10.1016/j.asr.2007.09.015](https://doi.org/10.1016/j.asr.2007.09.015).
- Trukhanov K, Morozov D. 1970. Optimization of a Magnetic Radiation Shield. *Soviet Phys - Technical Phys.* **15**.
- Truscott P, Lei F, Dyer C, Ferguson C, Gurriaran R, Nieminen P, Daly E, Apostolakis J, Giani S, Pia M, Urban L, Maire M. 2000. GEANT4 - A New Monte Carlo Toolkit for Simulating Space Radiation Shielding and Effects. Proceedings of the Radiation Effects Data Workshop, 24-28 July 2000, Reno, Nevada. DOI: [10.1109/REDW.2000.896281](https://doi.org/10.1109/REDW.2000.896281).
- U.S. Air Force. 1985. Application Guidelines for MIL-STD-1540B; Test Requirements for Space Vehicles. United States Air Force Report No. MIL-HDBK-340.
- U.S. Congress Office of Technology Assessment. 1991. Exploring the Moon and Mars: Choices for the Nation. <https://history.nasa.gov/32992.pdf> [accessed 29 January 2020].
- UNSCEAR. 2008. Effects of Ionizing Radiation. United Nations Scientific Committee on the Effects of Atomic Radiation Report to the General Assembly, with Scientific Annexes. [https://www.unscear.org/unscear/en/publications/2006\\_1.html](https://www.unscear.org/unscear/en/publications/2006_1.html) [accessed 14 January 2020].

- UNSCEAR. 2017. Report of the United Nations Scientific Committee on the Effects of Atomic Radiation. <https://www.unscear.org/unscear/en/publications/2017.html> [accessed 19 May 2020].
- Urban E. 1969. Superconducting Magnets for Active Shielding. In Radiation Physics Research at Marshall Space Flight Center: Research Achievements Review. National Aeronautics and Space Administration.
- Van Allen J, Ludwig G, Ray E, McIlwain C. 1958. Observation of High Intensity Radiation by Satellites 1958 Alpha and Gamma. *J Jet Propuls.* **28**. DOI: [10.2514/8.7396](https://doi.org/10.2514/8.7396).
- Vassiliev O. 2017. Monte Carlo Methods for Radiation Transport. Dover International Publishing. ISBN: 9783319441412.
- von Braun W. 1969. Will Mighty Magnets Protect Voyagers to Planets? *Pop Sci.* **1969**.
- Vuolo M, Giraudo M, Musenich R, Calvelli V, Ambroglini F, Burger W, Battiston R. 2016. Monte Carlo Simulations for the Space Radiation Superconducting Shield Project (SR2S). *Life Sci Space Res.* **8**. DOI: [10.1016/j.lssr.2015.12.003](https://doi.org/10.1016/j.lssr.2015.12.003).
- Washburn S, Blattnig S, Singleterry R, Westover S. 2014. Analytical-HZETRN Model for Rapid Assessment of Active Magnetic Radiation Shielding. *Adv Space Res.* **53**. DOI: [10.1016/j.asr.2013.09.038](https://doi.org/10.1016/j.asr.2013.09.038).
- Washburn S, Blattnig S, Singleterry R, Westover S. 2015. Active Magnetic Radiation Shielding System Analysis and Key Technologies. *Life Sci Space Res.* **4**. DOI: [10.1016/j.lssr.2014.12.004](https://doi.org/10.1016/j.lssr.2014.12.004).
- Westover S, Battiston R, Meinke R, Van Sciver S, Singleterry R. 2012. Active Radiation Shielding Utilizing High Temperature Superconductors. Proceedings of the NIAC Symposium, 27-29 March 2012, Pasadena, California.

- Westover S. 2014. Magnet Architectures and Active Radiation Shielding Study (MAARS). National Aeronautics and Space Administration Report No. NASA-TP-2014-217390.
- White E. 1963. Technique for Calculating Trajectories of Singly Charged Particles in a Static Magnetic Field for Shielding Purposes. United States Air Force Report No. ASD-TDR-63-72.
- White House National Science and Technology Council. 1996. National Space Policy. <https://fas.org/spp/military/docops/national/nstc-8.htm> [accessed 29 January 2020].
- White House. 2004. President Bush Announces New Vision for Space Exploration Program. <https://history.nasa.gov/SEP%20Press%20Release.htm> [accessed 29 January 2020].
- White House. 2017. Presidential Memorandum on Reinvigorating America's Human Space Exploration Program. <https://www.whitehouse.gov/presidential-actions/presidential-memorandum-reinvigorating-americas-human-space-exploration-program/> [accessed 29 January 2020].
- Whitmire A, Leveton L, Broughton H, Basner M, Kearney A, Ikuma L, Morris M. 2015. Minimum Acceptable Net Habitable Volume for Long-Duration Exploration Missions. National Aeronautics and Space Administration Report No. NASA-TM-2015-218564.
- Wilson J, Miller J, Konradi A, Cucinotta F. 1997. Shielding Strategies for Human Space Exploration. National Aeronautics and Space Administration Report No. NASA-CR-3360.
- Wilson J, Cucinotta F, Miller J, Shinn J, Thibeault S, Singleterry R, Simonsen L, Kim M. 1998. Materials for Shielding Astronauts from the Hazards of Space Radiation. National Aeronautics and Space Administration.
- Wilson J. 2000. Overview of Radiation Environments and Human Exposures. *Health Phys.* **79**.

- Wilson J, Cucinotta F, Miller J, Shinn J, Thibeault S, Singleterry R, Simonsen L, Kim M. 2001. Approach and Issues Relating to Shield Material Design to Protect Astronauts from Space Radiation. *Materials Design*. **22**. DOI: [10.1016/S0261-3069\(01\)00014-0](https://doi.org/10.1016/S0261-3069(01)00014-0).
- Winglee R, Slough J, Ziemba T, Goodson A. 2000. Mini-Magnetospheric Plasma Propulsion: Trapping the Energy of the Solar Wind for Spacecraft Propulsion. *J Geophys Res*. **105**. DOI: [10.1029/1999JA000334](https://doi.org/10.1029/1999JA000334).
- World Health Organization. 2006. Environmental Health Criteria 232: Static Fields.
- Wu H, Huff J, Casey R, Kim M, Cucinotta F. 2009. Risk of Acute Radiation Syndromes Due to Solar Particle Events. In *Human Health and Performance Risks of Space Exploration Missions*. McPhee J, Charles J (Eds). National Aeronautics and Space Administration.
- Xu X, Eckerman K (Eds). 2010. Handbook of Anatomical Models for Radiation Dosimetry. CRC Press, New York. ISBN: 9781420059809.
- Zang E, Wynder E. 1996. Differences in Lung Cancer Risk Between Men and Women: Examination of the Evidence. *JNCI*. **88**. DOI: [10.1093/jnci/88.11.764-a](https://doi.org/10.1093/jnci/88.11.764-a).
- Zeitlin C. 2016. The Role of Nuclear Fragmentation in Particle Therapy and Space Radiation Protection. *Front Oncol*. **6**. DOI: [10.3389/fonc.2016.00065](https://doi.org/10.3389/fonc.2016.00065).

## **Vita**

Kristine L. Ferrone is originally from Pittsburgh, Pennsylvania and graduated from Hampton High School in 2001. She completed her undergraduate studies at Carnegie Mellon University in Pittsburgh, receiving her Bachelor of Science in Physics / Astrophysics in 2004. For the next two years, she worked as a Physics Associate at Brookhaven National Laboratory in Upton, New York. She attended the University of Florida for graduate school, receiving her Master of Business Administration in 2007. From 2006 to 2011, she worked at NASA Johnson Space Center as a Mission Scientist and flight controller for the International Space Station. During this time, she completed two additional Master degrees via part-time study, a Master of Science in Space Architecture from the University of Houston and a Master of Science in Sports Medicine from the United States Sports Academy. Since 2011, she has worked for The Aerospace Corporation as a System Engineer and Project Leader, supporting NASA, U.S. Air Force, and national security space programs. In August of 2015 she entered the University of Texas MD Anderson Cancer Center UTHHealth Graduate School of Biomedical Sciences.

Permanent address:

7030 Crystal Dowels Dr.  
Pasadena, TX 77505