An Integrated Traverse Planner and Analysis Tool for Future Lunar Surface Exploration

by

AARON WILLIAM JOHNSON

B.S.E. Aerospace Engineering
University of Michigan, 2008

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Signature of Author _____
Department of Aeronautics and Astronautics
May 21, 2010

Certified by _____
Jeffrey A. Hoffman, Thesis Supervisor
Professor of the Practice of Aeronautics and Astronautics

Certified by _____
Dava J. Newman, Thesis Supervisor
Professor of Aeronautics and Astronautics & Engineering Systems
Director of Technology and Policy Program

Accepted by _____
Eytan Modiano
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Studies

An Integrated Traverse Planner and Analysis Tool for Future Lunar Surface Exploration

by

Aaron William Johnson

Submitted to the Department of Aeronautics and Astronautics on May 21, 2010 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Aeronautics and Astronautics

Abstract

This thesis discusses the Surface Exploration Traverse Analysis and Navigation Tool (SEXTANT), a system designed to help maximize productivity, scientific return, and safety on future lunar and planetary explorations,. The goal of SEXTANT is twofold: to provide engineers with a realistic simulation of traverses to assist with hardware design, and to serve as an aid for astronauts that will allow for more autonomy in traverse planning and re-planning.

SEXTANT is a MATLAB-based tool that computes the most efficient path between user-specified Activity Points across a lunar or planetary surface for a suited astronaut or transportation rover. Currently, SEXTANT uses an elevation model of the lunar south pole generated from topography data from the Lunar Orbiter Laser Altimeter instrument aboard the Lunar Reconnaissance Orbiter. The efficiency of a traverse is derived from any number of metrics: the path distance, time, or the explorer's energy consumption. Energy consumption is either the metabolic expenditure of an astronaut or the power usage of a transportation rover over the terrain. The user can select Activity Points and visualize the generated path on a 3D mapping interface. The capabilities of SEXTANT are further augmented by the Individual Mobile Agents System (iMAS) astronaut assistant, developed by NASA Ames. SEXTANT leverages iMAS's speech dialog interface to provide the explorer with real-time guidance and navigation along the most efficient path. SEXTANT can also calculate the sun position and shadowing with respect to points along the traverse and the time the explorer arrives at each of them. This data is then used to compute the thermal load on suited astronauts, or the solar power generation of rovers. Example traverses are presented for both types of explorers, showing the capabilities of SEXTANT and the dynamics of the thermal and power systems given different environmental conditions. All of its capabilities make SEXTANT the traverse planning tool with the most accurate and comprehensive representation of lunar and planetary traverses.

Thesis Supervisor: Jeffrey A. Hoffman
Professor of the Practice of Aeronautics and Astronautics

Thesis Supervisor: Dava J. Newman
Professor of Aeronautics and Astronautics & Engineering Systems
Director of Technology and Policy Program

Acknowledgments

This thesis has been a long journey, and like any voyage it has been made much easier and enjoyable by my interactions with others. There are so many people that I have to thank for their help in completing this thesis. First of all, I must thank God for all the gifts and talents with which I have been blessed. There have been a number of times when I wasn't sure if I could make it here at MIT, but He has certainly helped me through the tough times. I also want to thank my family for everything they've done for me. Mom, Dad, Adam, and Ryan – thanks for your support throughout my entire life. I'm the man I am today because of your love and guidance. And Dad, watching *From the Earth to the Moon* with you back in middle school is the reason why I'm an aerospace engineer. Thank you for introducing me to spaceflight.

To my advisors, Jeff Hoffman and Dava Newman, thank you for giving me the opportunity to work on this project and learn from you. It's been a great experience. I have not only learned about traverse planning and astronaut thermal modeling, but I've learned how to perform research and be a contributing member of the field of human spaceflight. That's been one of the best parts of my time here. I look forward to working with both of you for my Ph.D. To James Waldie, thank you for being a great sounding board for my ideas. Having someone I could trust to answer my dumb questions greatly helped with the adjustment to grad. school and research. It was always a pleasure to talk to you about non-academic subjects as well.

To everyone out at NASA Ames who assisted me during my summer out there: Jessica Marquez, thank you for all your mentoring and support. You knew exactly when to encourage me, and when to push me harder. Thank you for also taking me to and from the airport. John Dowding, thank you for all of your help getting ready to integrate iMAS and SEXTANT. It was invaluable, and the biggest disappointment of that summer is that we weren't able to work together. To Maarten Sierhuis, Ron van Hoof, and Bill Clancey, thank you for answering all of my questions about iMAS and Brahms, no matter where in the world you were.

Another group of very different people – planetary scientists – has also provided enormous assistance to my research: To Maria Zuber, thanks for your participation in this project and letting me use the LOLA elevation data. The chance to work with real lunar terrain maps has been fantastic. The Moon had always been a quarter of a million miles away to me; it's amazing to have it on my desktop now. Erwan Mazarico, thank you for all of your help in calculating the sun illumination. Your work is literally the foundation of mine, and I appreciate you taking the

time to send me all those e-mails, helping me to get things working. You saved me a great deal of time and sanity.

I was a bit nervous coming to grad. school here at MIT, wondering if I would be thrust into the middle of a group of socially inept nerds. Well, that fear turned out to be completely unfounded. I have developed so many wonderful friendships that to start naming people would inevitably leave someone out. To everyone in the MVL, you're all amazing. It's been so much fun spending the past two years with you. With Warfish, softball, squash, beer brewing, and just talking in the lab, it's a wonder that I was able to finish this thesis. (Probably because we've toned all that down this past month.) To all my other friends, it's been awesome spending time with you as well. Thanks for giving me an escape from the stresses of grad. school. You're really the reason why I've enjoyed the past two years so much. Please don't find a cooler new friend next year to replace me. And to all my friends, I'm really impressed that you're reading my thesis. I owe you a beer.

Finally, and most importantly, I would like to thank my fiancée, Gretchen Miller, for all her love and support these past two years. Gretchen, thank you for always being there for me when I needed someone to talk to. I know I can be a bit crazy, and you've helped to keep me grounded. It hasn't been easy to be apart, but looking forward to all of our weekends together has really kept me going. Spending time with you in Boston, Ann Arbor, Toledo, Pittsburgh, and San Francisco are some of the best memories I have. (Did I miss anywhere?). I'm so excited to begin the next chapter of our lives together this fall.

Biographical Note

Aaron Johnson was born in 1985 in the Chicago suburb of Naperville, IL, but grew up in Pittsburgh, PA. He attended the University of Michigan from 2004 – 2008, graduating *summa cum laude* with a bachelor's of science in engineering degree in Aerospace Engineering. In 2008 he entered graduate school at the Massachusetts Institute of Technology in the Department of Aeronautics and Astronautics. He has spent the past two years as a member of the Man Vehicle Lab performing his master's research. Aaron has been accepted to the Ph.D. program in the Department of Aeronautics and Astronautics, which he will begin in 2011 after spending a year in Ann Arbor, MI. On October 23, 2010, he will marry his girlfriend of three and a half years, Gretchen Miller.

This page intentionally left blank.

Table of Contents

Abstract 3

Acknowledgments 4

Biographical Note 5

List of Figures 9

List of Tables 13

List of Acronyms 14

1 Introduction 17
1.1 Motivation 17
1.1.1 Challenges for Future Planetary Exploration 18
1.2 Thesis Contributions 21
1.2.1 Incorporation of a Lunar Elevation Map 21
1.2.2 Development of a Transportation Rover Explorer Model 21
1.2.3 Integration of SEXTANT and iMAS Astronaut Assistant System 22
1.2.4 Determination of Shadowing, Astronaut Thermal Load, and Rover Power Generation and Usage 22
1.3 Thesis Outline 23

2 Literature Review 29
2.1 Decision Support Tools 29
2.2 Traverse Planners for Planetary Surface Exploration 32
2.3 Development of SEXTANT 34
2.4 Conclusion 38

3 Basics of SEXTANT 41
3.1 Traverse Representation in SEXTANT 41
3.1.1 Exploration Objectives 41
3.1.2 Environment Model 42
3.1.3 Explorer Model 44
3.2 Optimization Algorithm 48
3.3 SEXTANT Graphical User Interface 50
3.3.1 Initializing SEXTANT 51
3.3.2 SEXTANT Input Menus 51
3.3.3 3D Mapping Interface 53
3.4 Real-time Navigation 60
3.4.1 Individual Mobile Agents System Background 60
3.4.2 Demonstrating the Capabilities of iMAS in the Field 63
3.4.3 Integrated SEXTANT-iMAS System 64
3.5 Conclusion 68

4 Determination of Shadowing on the Lunar Surface and Computation of Astronaut Thermal Load and Transportation Rover Power Consumption and Generation 71
4.1 The Lunar Reconnaissance Orbiter 71
4.1.1 The Lunar Orbiter Laser Altimeter 72
4.2 Determining Shadowing on the LOLA-Derived Elevation Map 73
4.3 Astronaut Thermal Model 78
4.3.1 Heat Flux Into and Out of the Space Suit 79

4.3.2 Space Suit Thermal Control System ... 87
4.4 Rover Power Model ... 90
4.4.1 Rover Energy Consumption During a Traverse ... 91
4.5 Conclusion ... 92

5 Example Astronaut and Rover Traverses ... 95
5.1 Suited Astronaut EVAs ... 97
5.1.1 EVA 1 – Explanatory Traverse in Shadow and Sunlight ... 97
5.1.2 EVA 2 – Traverse Entirely in Sunlight ... 109
5.1.3 EVA 3 – Traverse Entirely in Shadow ... 119
5.2 Example Rover Traverse ... 129
5.2.1 Planning the Desired Traverse ... 129
5.2.2 Generating Return-Home Paths ... 141
5.3 Conclusion ... 148

6 Conclusions and Future Work ... 151
6.1 Motivation for SEXTANT ... 151
6.2 Thesis Contributions ... 151
6.2.1 Incorporation of a Lunar Elevation Map ... 152
6.2.2 Development of a Transportation Rover Explorer Model ... 152
6.2.3 Integration of SEXTANT and iMAS Astronaut Assistant System ... 153
6.2.4 Determination of Shadowing, Astronaut Thermal Load, and Rover Power Generation and Usage ... 154
6.2.5 Overall Contributions ... 155
6.3 Future Work ... 156
6.3.1 Integration with a Lunar Navigation System ... 156
6.3.2 Include Sun Position in Cost Functions ... 157
6.3.3 Develop a User-Centric Display ... 158
6.3.4 Measure Metabolic Costs in Real-Time ... 159
6.3.5 Optimize Scientific Return for a Traverse ... 160
6.4 Final Thoughts ... 160

7 References ... 161

Appendix A Pictorial Description of A* Graph Search Algorithm ... 169

Appendix B iMAS Language Examples ... 172
B.1 General ... 172
B.2 Activities ... 172
B.3 Navigation and Automatic Associations ... 173
B.4 Science Data ... 173
B.5 Metabolic Parameters ... 175

Appendix C Distance, Time, and Energy Cost Data for Rover Traverse ... 177
C.1 Original Rover Traverse ... 177
C.2 First Modified Rover Traverse ... 178
C.3 Second Modified Rover Traverse ... 179

Appendix D SEXTANT MATLAB Code ... 180

List of Figures

Figure 1.1. (left) Necessary EVA hours projected for the completion of the International Space Station (Gernhardt 2007) 19
Figure 1.2. (right) Necessary EVA hours projected for the establishment of a lunar base (Gernhardt 2007) 19
Figure 1.3. Original plan of the second EVA of Apollo 14 ("Surface Operations" 2010) 20
Figure 2.1. Information visualization produced by Wood and Wood's traverse planning tool (2006) 34
Figure 2.2. Planetary EVA framework (Márquez 2007) 35
Figure 3.1. Structure of explorer model within SEXTANT 44
Figure 3.2. Velocity plotted with respect to slope for suited astronauts 45
Figure 3.3. Metabolic rate plotted with respect to slope for suited astronauts 46
Figure 3.4. Energy rate plotted with respect to slope for transportation rovers 48
Figure 3.5. SEXTANT input menus 52
Figure 3.6. 3D view of lunar elevation map in sextant mapping interface 53
Figure 3.7. Activity Points for two astronauts specified on 3D mapping interface 53
Figure 3.8. Traverse paths for two astronauts on 3D mapping interface 55
Figure 3.9. Traverse path without simplification 55
Figure 3.10. Traverse path with simplification 56
Figure 3.11. Display showing metabolic cost of Astronaut 1's EVA with respect to distance and time 57
Figure 3.12. Display showing metabolic cost of Astronaut 2's EVA with respect to distance and time 57
Figure 3.13. Display showing mass of sublimated water (left) and heater power (right) of Astronaut 1's EVA with respect to distance 58
Figure 3.14. Display showing shadowing along Astronaut 1's EVA 59
Figure 3.15. Return-home path for Astronaut 1 59
Figure 3.16. Interface of the Compendium database used by iMAS 65
Figure 3.17. Determination of relative heading to requested path point 66
Figure 3.18. Flow of information within integrated SEXTANT-iMAS system 67
Figure 4.1. Artist's rendition of LRO in orbit around the moon ("Lunar Reconnaissance Orbiter" 2010) 71
Figure 4.2 (left). LOLA flight model (Ramos-Izquierdo et al. 2008) 73
Figure 4.3 (right). Ground track of LOLA (Smith et al. 2010a) 73
Figure 4.4. Elevation map of lunar south polar region 74
Figure 4.5. Names of prominent craters in lunar south polar region (Bussey and Spudis 2004, Byrne 2005) 75
Figure 4.6 (left). Lunar elevation map showing location of selected submap 76
Figure 4.7 (right). Selected submap of the lunar surface, as seen in SEXTANT 76
Figure 4.8. Mechanical efficiency as a function of the terrain slope (Adapted from Margaria 1976) 81
Figure 4.9. Mechanical efficiency as a function of the terrain slope, with curve fit (Adapted from Margaria 1976) 82
Figure 4.10. Astronaut metabolic expenditure and heat loss per meter 84
Figure 4.11. Diagram of space suit thermal control system 88

Figure 5.1. 120 km by 120 km submap for example traverses 95
Figure 5.2 (left). Submap sun illumination on June 4, 2010, at 10:00 am EDT 96
Figure 5.3 (right). Photometric rendering of submap sun illumination 96
Figure 5.4. Location of EVA 1 on 120 km by 120 km submap 98
Figure 5.5. Close-up view of EVA 1 on 25 km by 25 km map 98
Figure 5.6. Elevation profile of EVA 1 99
Figure 5.7 (left). Cumulative metabolic cost of EVA 1 with respect to distance 100
Figure 5.8 (right). Cumulative metabolic cost of EVA 1 with respect to time 100
Figure 5.9. Shadowing along EVA 1 101
Figure 5.10 (left). Mass of sublimated water during EVA 1 with respect to distance 102
Figure 5.11 (right). Heater energy during EVA 1 with respect to distance 102
Figure 5.12 (left). Mass of sublimated water during EVA 1 with respect to distance, with shadowing 102
Figure 5.13 (right). Heater energy during EVA 1 with respect to distance, with shadowing 102
Figure 5.14 (left). Mass of sublimated water during EVA 1 with respect to time 103
Figure 5.15 (right). Heater energy during EVA 1 with respect to time 103
Figure 5.16 (left). Mass of sublimated water during EVA 1 with respect to time, with shadowing 103
Figure 5.17 (right). Heater energy during EVA 1 with respect to time, with shadowing 103
Figure 5.18. Heat flux with respect to time for EVA 1 105
Figure 5.19. Heat with respect to traverse stages for EVA 1 106
Figure 5.20 (left). External space suit temperature during EVA 1 with respect to distance 107
Figure 5.21 (right). External space suit temperature during EVA 1 with respect to time 107
Figure 5.22 (left). External space suit temperature during EVA 1 with respect to distance, with shadowing 107
Figure 5.23 (right). External space suit temperature during EVA 1 with respect to time, with shadowing 107
Figure 5.24. Thermal equilibrium for varying shadowing conditions 108
Figure 5.25. Location of EVA 2 on 120 km by 120 km submap 110
Figure 5.26. Close-up view of EVA 2 on 20 km by 20 km map 110
Figure 5.27. Elevation profile of EVA 2 111
Figure 5.28 (left). Cumulative metabolic cost of EVA 2 with respect to distance 112
Figure 5.29 (right). Cumulative metabolic cost of EVA 2 with respect to time 112
Figure 5.30. Shadowing along EVA 2 113
Figure 5.31 (left). Mass of sublimated water during EVA 2 with respect to distance 114
Figure 5.32 (right). Heater energy during EVA 2 with respect to distance 114
Figure 5.33 (left). Mass of sublimated water during EVA 2 with respect to distance, with shadowing 114
Figure 5.34 (right). Heater energy during EVA 2 with respect to distance, with shadowing 114
Figure 5.35 (left). Mass of sublimated water during EVA 2 with respect to time 115
Figure 5.36 (right). Heater energy during EVA 2 with respect to time 115
Figure 5.37 (left). Mass of sublimated water during EVA 2 with respect to time, with shadowing 115
Figure 5.38 (right). Heater energy during EVA 2 with respect to time, with shadowing 115
Figure 5.39. Heat flux with respect to time for EVA 2 116
Figure 5.40. Heat flux with respect to traverse stages for EVA 2 117

Figure 5.41 (left). External space suit temperature during EVA 2 with respect to distance 118
Figure 5.42 (right). External space suit temperature during EVA 2 with respect to time 118
Figure 5.43 (left). External space suit temperature during EVA 2 with respect to distance, with shadowing 118
Figure 5.44 (right). External space suit temperature during EVA 2 with respect to time, with shadowing 118
Figure 5.45. Location of EVA 3 on 120 km by 120 km submap..... 120
Figure 5.46. Close-up view of EVA 3 on 20 km by 20 km map 120
Figure 5.47. Elevation profile of EVA 3..... 121
Figure 5.48 (left). Cumulative metabolic cost of EVA 3 with respect to distance 122
Figure 5.49 (right). Cumulative metabolic cost of EVA 3 with respect to time..... 122
Figure 5.50. Shadowing along EVA 3 123
Figure 5.51 (left). Mass of sublimated water during EVA 3 with respect to distance..... 123
Figure 5.52 (right). Heater energy during EVA 3 with respect to distance 123
Figure 5.53 (left). Mass of sublimated water during EVA 3 with respect to distance, with shadowing 124
Figure 5.54 (right). Heater energy during EVA 3 with respect to distance, with shadowing..... 124
Figure 5.55 (left). Mass of sublimated water during EVA 3 with respect to time..... 124
Figure 5.56 (right). Heater energy during EVA 3 with respect to time 124
Figure 5.57 (left). Mass of sublimated water during EVA 3 with respect to time, with shadowing 124
Figure 5.58 (right). Heater energy during EVA 3 with respect to time, with shadowing..... 124
Figure 5.59. Heat flux with respect to time for EVA 3..... 126
Figure 5.60. Heat flux with respect to traverse stages for EVA 3 127
Figure 5.61 (left). External space suit temperature during EVA 3 with respect to distance 128
Figure 5.62 (right). External space suit temperature during EVA 3 with respect to time 128
Figure 5.63 (left). External space suit temperature during EVA 3 with respect to distance, with shadowing 128
Figure 5.64 (right). External space suit temperature during EVA 3 with respect to time, with shadowing 128
Figure 5.65. Location of rover traverse on 120 km by 120 km submap..... 129
Figure 5.66. 3D view of rover traverse 130
Figure 5.67. Elevation profile of rover traverse..... 130
Figure 5.68 (left). Cumulative energy cost of rover traverse with respect to distance 132
Figure 5.69 (right). Cumulative energy cost of rover traverse with respect to time..... 132
Figure 5.70. Shadowing along rover traverse 133
Figure 5.71. Battery energy level during rover traverse with respect to distance 133
Figure 5.72. Battery energy level during rover traverse with respect to time 134
Figure 5.73. Battery energy level during rover traverse with respect to distance, with shadowing 134
Figure 5.74. Battery energy level during rover traverse with respect to time, with shadowing . 135
Figure 5.75. Battery energy level during first modified rover traverse with respect to distance 136
Figure 5.76. Battery energy level during first modified rover traverse with respect to time..... 137
Figure 5.77. Battery energy level during first modified rover traverse with respect to distance, with shadowing 137

Figure 5.78. Battery energy level during first modified rover traverse with respect to time, with shadowing 138
Figure 5.79. Battery energy level during second modified rover traverse with respect to distance 139
Figure 5.80. Battery energy level during second modified rover traverse with respect to time. 139
Figure 5.81. Battery energy level during second modified rover traverse with respect to distance, with shadowing 140
Figure 5.82. Battery energy level during second modified rover traverse with respect to time, with shadowing 140
Figure 5.83. Rover return-home path 1 from Haworth Crater to habitat 142
Figure 5.84. Shadowing along rover return-home path 1 142
Figure 5.85. Battery energy level during rover return-home path 1 with respect to distance 143
Figure 5.86. Battery energy level during rover return-home path 1 with respect to time 143
Figure 5.87. Battery energy level during rover return-home path 1 with respect to distance, with shadowing 144
Figure 5.88. Battery energy level during rover return-home path 1 with respect to time, with shadowing 144
Figure 5.89. Rover return-home path 2 from Shackleton Crater to habitat 145
Figure 5.90. Shadowing along rover return-home path 1 145
Figure 5.91. Battery energy level during rover return-home path 2 with respect to distance 146
Figure 5.92. Battery energy level during rover return-home path 2 with respect to time 147
Figure 5.93. Battery energy level during rover return-home path 2 with respect to distance, with shadowing 147
Figure 5.94. Battery energy level during rover return-home path 2with respect to time, with shadowing 148
Figure 6.1. Geologic explorations of the Arizona desert from the Lunar Electric Rover prototype (Gambino 2010) 153
Figure 6.2. Graphical representation of absolute and relative sun azimuth and elevation angles (Márquez 2008) 158

List of Tables

Table 1.1. EVA details of the Apollo lunar landing missions ("Apollo Lunar Mission Table", 2008 "Lunar Mission Timeline" 2010) 18
Table 2.1. Levels of automation (Parasuraman et al. 2000) 29
Table 3.1. Velocity equations for suited astronaut explorers (Márquez 2008) 45
Table 3.2. Metabolic rate equations for suited astronaut (Santee 2001) 46
Table 3.3. Energy rate equations for transportation rovers (Carr 2001) 47
Table 3.4. Minimum energy costs for astronauts and rovers on the Moon and Earth 50
Table 4.1. Sources of heat transfer with the astronaut's space suit 79
Table 4.2. Curve fit for mechanical efficiency energetics data 82
Table 4.3. Solar cell efficiencies (Hong 2007) 90
Table 5.1. Constraints for suited astronaut EVAs 97
Table 5.2. Cumulative traverse metrics at EVA 1 Activity Points 99
Table 5.3. Time spent at Activity Points in EVA 2 112
Table 5.4. Cumulative traverse metrics at EVA 2 Activity Points 113
Table 5.5. Time spent at Activity Points in EVA 3 121
Table 5.6. Cumulative traverse metrics at EVA 3 Activity Points 122
Table 5.7. Time spent at Activity Points in rover traverse 131

List of Acronyms

- CRaTER – Cosmic Ray Telescope for the Effects of Radiation
- DARPA – Defense Advanced Projects Research Agency
- DLRE – Diviner Lunar Radiometer Experiment
- EAPS – MIT Department of Earth and Planetary Sciences
- EMU – Extravehicular Mobility Unit
- EVA – Extravehicular activity
- FPT – Flight Planning Testbed
- GIS – Geographical information system
- GPS – Global positioning system
- GUI – Graphical user interface
- iMAS – Individual Mobile Agents System
- ISS – International Space Station
- IVA – Intravehicular activity
- LAMP – Lyman Alpha Mapping Project
- LCROSS – Lunar Crater Observation and Sensing Satellite
- LCVG – Liquid cooling and ventilation garment
- LEND – Lunar Exploration Neutron Detector
- LER – Lunar Electric Rover
- LoA – Level of automation
- LOLA – Lunar Orbiter Laser Altimeter
- LRO – Lunar Reconnaissance Orbiter
- LROC – Lunar Reconnaissance Orbiter Camera
- LRV – Lunar Roving Vehicle
- MAA – Mobile Agents Architecture
- MAPGEN – Mixed-Initiative Activity Plan Generator
- MDRS – Mars Desert Research Station
- MER – Mars Exploration Rovers
- MIT – Massachusetts Institute of Technology
- PATH – Planetary Aid for Traversing Humans
- PSI – Phoenix Science Interface
- SAP – Science Activity Planner
- SEXTANT – Surface Exploration Traverse Analysis and Navigation Tool
- TGA – Traverse Generation Assistant

For my parents,
and for Gretchen.

There once was an astronaut Ed;
Across the Moon him and Al tread.
But navigation was poor,
And their position unsure.
There must be a good way instead.

1 Introduction

1.1 Motivation

The ultimate goal of the United States' human spaceflight program is to land humans safely on the surface of Mars. This is the viewpoint of the recent report of the Review of U.S. Human Space Flight Plans Committee (also called the Augustine Committee after its chairman, Norm Augustine), which stated, "A human landing followed by an extended human presence on Mars stands prominently above all other opportunities for exploration" (2009, p. 14). Even so, it would be extremely expensive, risky, and difficult for NASA to build what the Augustine Commission called a "Mars First" program, designed to get humans to Mars as quickly as possible while bypassing all other intermediate goals. One of the other two programmatic options put forth by the Augustine Commission is the "Flexible Path", which consists of manned missions to locations such as the Earth-Sun Lagrange points, near-Earth objects, Martian orbit, and the Martian satellites, all before a Martian surface landing. The second is entitled a "Moon First" program, and involves establishing a lunar base to test Martian hardware and exploration techniques before landing humans on Mars. The Augustine Commission remarked that, "Before we explore Mars, we will likely do some of both the Flexible Path and lunar exploration – the primary decision *is one of sequence* [emphasis original]" (Review of U.S. Human Spaceflight Plans Committee 2009, p. 44). Recently, program-changing discussions have been occurring within the federal government and NASA, and the near-term goals of the human space exploration program in the United States are uncertain. However, the drive to explore beyond low-Earth orbit remains a constant.

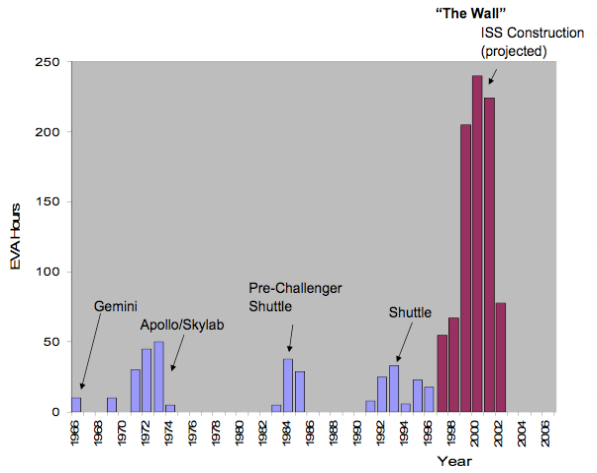
NASA is currently the only space agency in the world with experience in manned lunar and planetary exploration. Between 1969 and 1972, NASA successfully landed twelve men on the surface of the Moon and returned them safely to Earth. These missions explored six different sites spread across the near side of the Moon. Each mission had a number of extravehicular activities (EVA) where astronauts left the Lunar Module and traveled across the lunar surface, either by foot or with the use of the Lunar Roving Vehicle (LRV). With each subsequent mission, the length, distance traveled, and amount of scientific return increased. Table 1.1 shows details of each Apollo lunar mission.

Table 1.1. EVA details of the Apollo lunar landing missions ("Apollo Lunar Mission Table", 2008 "Lunar Mission Timeline" 2010)

Mission	Landing Date	Time on Surface (hr:min)	Use of LRV?	Number of EVAs	EVA Durations (hr:min)	Distance Traveled (km)	Mass of Samples Returned (kg)
Apollo 11	July 20, 1969	21:36	No	1	2:31	0.2	21.55
Apollo 12	Nov. 19, 1969	31:31	No	2	3:56 3:50	0.6 1.3	34.35
Apollo 14	Feb. 5, 1971	33:31	No	2	4:48 4:34	0.5 3.0	42.28
Apollo 15	July 30, 1971	66:55	Yes	3	6:33 7:12 4:50	10.3 12.5 5.1	77.31
Apollo 16	April 21, 1972	71:02	Yes	3	7:11 7:23 5:40	4.2 11.1 11.4	95.71
Apollo 17	Dec. 11, 1972	75:00	Yes	3	7:12 7:37 7:15	3.5 20.4 12.1	110.52

1.1.1 Challenges for Future Planetary Exploration

Future explorations of the Moon and Mars will require EVAs of a scope and scale greater than anything ever attempted. EVAs will be more frequent and longer in time and distance. Speaking to the frequency, Figure 1.1 compares the number of EVA hours performed during the Gemini, Apollo, and pre-International Space Station (ISS) Space Shuttle programs with the number of EVA hours that were projected for the construction of the ISS (Gernhardt 2009). Gernhardt terms this the "Wall of EVA." However, even the Wall of EVA is dominated by the number of EVA hours estimated for the establishment of a lunar base. This is shown as the "Mountain of EVA" in Figure 1.2.



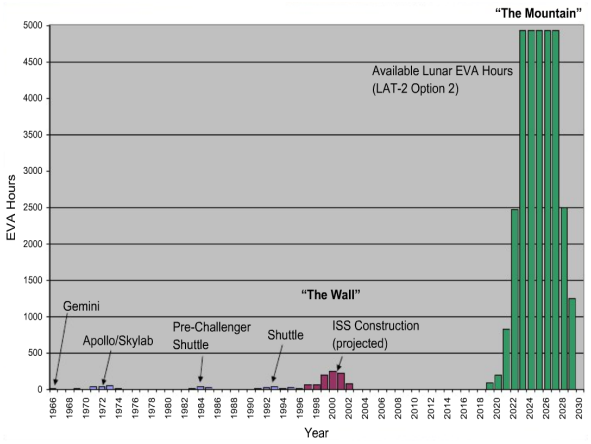


Figure 1.1. (left) Necessary EVA hours projected for the completion of the International Space Station (Gernhardt 2007)

Figure 1.2. (right) Necessary EVA hours projected for the establishment of a lunar base (Gernhardt 2007)

The length and duration of future EVAs will also be much greater than during Apollo. Under the most recent NASA lunar architecture, lunar space suits are being designed for traverses on foot of up to 8 hours (Culbert 2009). This is similar in duration to the longest EVAs undertaken during Apollo and current Space Shuttle and ISS EVAs. However, development is continuing for a two-person pressurized vehicle called the Lunar Electric Rover (LER) for longer traverses across the lunar surface. The LER is being designed to have the capability for 3-day traverses of up to 100 km, or 14-day traverses of up to 1000 km (Culbert 2009). Even the shorter end of this range is an order of magnitude farther and longer than the longest Apollo traverse with the LRV.

There are specific challenges to navigation in the lunar environment that have confounded astronauts in the past. The second EVA of the Apollo 14 mission clearly demonstrates these difficulties (Carr et al. 2003). This EVA, which occurred during the morning of Feb. 6, 1971, was the final walking EVA of the Apollo program. All subsequent Apollo missions carried the LRV. Commander Alan Shepard and Lunar Module Pilot Ed Mitchell left the Lunar Module with the goal of traveling to Cone Crater, approximately 1.5 km away (Figure 1.3).

1. Introduction

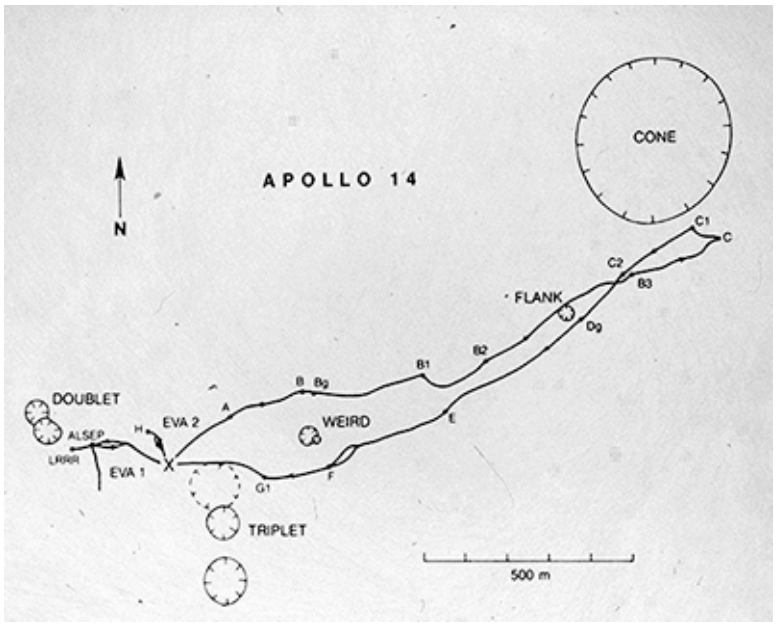


Figure 1.3. Original plan of the second EVA of Apollo 14 ("Surface Operations" 2010)

Shepard and Mitchell navigated by use of a paper map developed from photographs of the lunar surface, with craters identified as navigational landmarks. During their EVA, Shepard and Mitchell had difficulty in identifying specific craters by sight. This led to poor situational awareness, or confusion as to their exact position. Additionally, they had trouble judging distances on the lunar surface, due to the lack of atmosphere. On Earth, objects that are farther away appear "fuzzier", due to a phenomenon called "aerial perspective" (Kranz 2005). Light from farther-away objects must pass through more atmosphere to reach a person's eye; consequently, it is scattered more. Farther-away objects appear more bluish (because blue light is scattered more) and less in focus. Because there is no atmosphere on the Moon, all objects look just as clear, no matter their distance from the observer. Mitchell also commented that the helmet visor distorted his view of the lunar surface somewhat and made it more difficult to judge distances (Jones 2010). Relating to these difficulties in judging their position, Shepard and Mitchell had trouble estimating the amount of time that it would take to reach their destinations.

As the EVA progressed, Shepard and Mitchell traveled up steep slopes, necessitating high amounts of exertion and more frequent rest stops. They continued to search for the rim of Cone Crater, knowing they were close but being unable to see it. Eventually, the astronauts used up their 30-minute EVA extension time and had to continue on despite never reaching the rim of

Cone Crater. Post-mission analysis indicated that they were only approximately 20 m from the rim, but didn't know it at the time.

This EVA shows a number of difficulties with lunar navigation that must be overcome if future surface exploration is going to be a success. To help mitigate these challenges and improve the ability to carry out explorations on the lunar surface, the Massachusetts Institute of Technology (MIT) Man Vehicle Laboratory has developed an traverse planning tool called the Surface Exploration Traverse Analysis and Navigation Tool (SEXTANT). SEXTANT is designed to help users plan and optimize paths over a planetary surface. It contains an interface with a 3D terrain elevation map over which the user can place points of interest, called *Activity Points*. Terrain obstacles are areas where the explorer cannot travel due to safety constraints, and are specified by a user-inputted maximum slope. After the user has specified these, SEXTANT determines the most-efficient path between Activity Points, based on the metric of traverse distance, time, or explorer energy consumption. The roots of SEXTANT can be traced back for a decade through numerous graduate students. This thesis will focus on the author's four main contributions to SEXTANT, which greatly increases its utility and ability to model traverses.

1.2 Thesis Contributions

1.2.1 Incorporation of a Lunar Elevation Map

The first contribution of this thesis is the incorporation of a lunar elevation map into SEXTANT, which gives the user the ability to plan traverses on the surface of the Moon. The map used covers a 500 km by 500 km region around the lunar south pole, and has been created from data from the Lunar Orbiter Laser Altimeter (LOLA) instrument aboard the Lunar Reconnaissance Orbiter (LRO) spacecraft. Along with this elevation map, the equations detailing an explorer's velocity and energy consumption have been modified for the reduced lunar gravity. The incorporation of a lunar elevation map allows for the realistic simulation of lunar traverses now, years before explorers even travel to the Moon.

1.2.2 Development of a Transportation Rover Explorer Model

The second contribution of this thesis is the development of an explorer model for transportation rovers, like the LER. This allows the user to plan traverses for both suited astronauts and rovers. The rover explorer model describes the velocity of a rover and how the rover uses energy when traveling over different terrain slopes. SEXTANT can then determine the most-efficient paths for

rover traverses on the basis of distance, time, or energy cost, just as for suited astronauts. The ability to plan traverses for transportation rovers is important because they greatly extend the achievable time and distance of traverses – from 20 km in 8 hours to 1000 km in 14 days. Rovers will likely be heavily relied upon in future explorations, and incorporating them into SEXTANT increases the fidelity of the traverse model.

1.2.3 Integration of SEXTANT and iMAS Astronaut Assistant System

The third contribution of this thesis is the integration of SEXTANT and the Individual Mobile Agents System (iMAS) developed by NASA Ames Research Center. iMAS serves as an astronaut's personal assistant for planning and monitoring traverses, tracking biomedical parameters, and collecting and storing digital science data through a speech dialogue interface. The integrated SEXTANT-iMAS system is designed to take advantage of this speech dialog interface by providing the astronauts with auditory guidance along a path planned in SEXTANT. This serves as the first steps towards an implementable system that can be used for path planning and re-planning during lunar traverses. Such a system removes dependence upon guidance from Earth and gives the astronauts more flexibility and autonomy. In its current form, SEXTANT is an engineering tool not optimized for the astronaut user; however, the SEXTANT-iMAS integration is a good step in the right direction that can be experimented with and built upon in the future.

1.2.4 Determination of Shadowing, Astronaut Thermal Load, and Rover Power Generation and Usage

The final, and most important, contribution of this thesis is the ability of SEXTANT to predict the location of shadows on the lunar surface for a given time and location. This is important for many reasons. Knowing the location of shadows is critical for computing the thermal loading of an astronaut. Temperatures on the lunar surface can fluctuate from 104 K (-169° C) in shadows to 390 K (117° C) in the sunlight, on average (Heiken 1991, pp. 35-36). Temperatures in the permanently-shadowed craters on the lunar poles can dip even lower, possibly reaching 40 K (-233° C). As an astronaut travels in and out of shadows, the external thermal load on his life support system varies. The thermal load in turn directly influences the amount of consumables used, which is one of the major limiting factors of any traverse. A higher thermal load will require more metabolic expenditure, and therefore more oxygen and drinking water. Water in the

form of ice is also used for sublimation cooling, and so its consumption increases even more with a high thermal load. Similarly, the position of shadows is required for determining the power generation and usage of transportation rovers. The path that a rover can take is dependent upon whether or not it has enough power. A traverse that is in darkness for too long may not be feasible, as the available battery power may be smaller than that required. Likewise, if the rover has solar arrays, a traverse with alternating areas of sunlight and shadow may be feasible, given that there is enough time to charge the batteries while in the sun and that no shadowed portion is long enough to completely discharge the batteries.

By predicting and displaying the location of shadows and determining thermal loads for astronauts and the power generation and usage for rovers, SEXTANT is able to model a lunar traverse more accurately than it ever has before. This allows for realistic simulations of lunar explorations, a valuable tool for hardware design. When EVA suits and transportation rovers for future missions are being designed, engineers can use SEXTANT to simulate traverses and determine if the hardware systems are properly sized.

1.3 Thesis Outline

This thesis begins by discussing the relevant background literature in Chapter 2. SEXTANT can be classified as a decision support aid, which is a system designed to help the user make choices when faced with a number of options. Decision support aids have been studied for many years, in a number of forms. One of the most important questions for a decision support aid is the level of automation it should have. Prior research in this field has helped to motivate some of SEXTANT's design decisions. While there is a great deal of literature on general decision support aids, there is little literature specific too planetary path planners. This thesis discusses the few that do exist. Finally, the development of SEXTANT is explored in its numerous revisions and improvements.

Chapter 3 contains a description of SEXTANT and its functionality. First, it discusses SEXTANT's three-component representation of the traverse. This consists of the *Exploration Objectives*, which characterize the activities of the traverse, the *Environment Model*, which represents the lunar or planetary environment in which the traverse is occurring, and the *Explorer Model*, which details the energy consumption and constraints of the explorer performing the traverse. The explorer model is discussed in depth, specifically with regards to the metrics on which the most-efficient traverse can be measured – the path distance, time, and

explorer energy consumption. Energy consumption is the metabolic expenditure for a human astronaut and the power usage for transportation rovers. Both are functions of the terrain slope, path distance, explorer mass, and planetary gravity. Chapter 3 also presents SEXTANT's 3D mapping interface. SEXTANT's many capabilities are outlined, along with the underlying details of their operation. Chapter 3 concludes by discussing the integration of SEXTANT with the iMAS astronaut assistant system developed by NASA Ames Research Center. It begins with a history and background of iMAS, and then outlines the individual roles of SEXTANT and iMAS in the integrated system and how information is passed between them. SEXTANT is responsible for spatial planning of the Activity Points and computation of the most-efficient path. iMAS is then used to track an astronaut's progress during the EVA, and to provide auditory directions along the path when queried by the user. These auditory directions include the distance to a requested location along the path, as well as a relative heading computed with respect to the explorer's current heading.

Chapter 4 discusses the most important of this thesis's contributions, the prediction of shadowed locations on the lunar surface and the calculation of astronaut thermal load or rover power usage and generation. In order to accomplish this, SEXTANT has access to a 500 km by 500 km lunar elevation map acquired in partnership with the MIT Department of Earth, Atmospheric, and Planetary Sciences. The data has been captured by the Lunar Orbiter Laser Altimeter (LOLA) instrument aboard the Lunar Reconnaissance Orbiter (LRO) spacecraft. Chapter 4 first presents a history of this instrument and its technical capabilities. The map used by SEXTANT is centered about the lunar south pole, which is an area of focused interest for future exploration due to the probable presence of water ice. Determining the position of shadows on this map takes into account the sun's changing position with time, which is very important for long traverses. The sun travels approximately 0.5° across the lunar sky in one hour, meaning that over a 14-day rover traverse, the sun will be 180° different at the end than it was at the beginning. The actual calculation of the sun position is performed using the NASA Jet Propulsion Lab SPICE toolkit, which computes ephemeris data. The sun position is compared against pre-built terrain horizon databases to determine which locations along the traverse are in shadow, and which are in sunlight. The sun and shadow positions are used to calculate the thermal load on an astronaut and the power usage and generation for a transportation rover, which Chapter 4 discusses next. In the astronaut thermal model within SEXTANT, there are a

number of components to the heat flux into and out of the space suit, both external (the environment) and internal (electronics waste heat and heat from the inefficiency of astronaut metabolic work). SEXTANT speaks to each of these in depth and how they contribute to the thermal load on the astronaut. Chapter 4 then talks about the space suit thermal control system, which balances out the heat flux by sublimating ice for cooling or heating the space suit with a small heater. The amount of water required to replenish the sublimator ice and the heater power are the two consumables tracked within SEXTANT throughout a traverse. Chapter 4 finishes by presenting the rover power model. The rover has a solar array, which can provide power when in sunlight, and batteries, which are used when the rover is in shadow. The solar array can also recharge the batteries if there is excess power being produced. SEXTANT tracks the energy of the rover batteries throughout the traverse to ensure that they are never completely exhausted.

Chapter 5 describes a number of example lunar traverses with both suited astronauts and transportation rovers. These demonstrate the capability of SEXTANT and explain the dynamics of the astronaut thermal model and the rover power model. SEXTANT would be used in exactly this manner to plan and modify real traverses, making sure that they are feasible and within all constraints. All of the example traverses take place on a 120 km by 120 km submap of the lunar surface located near the south pole. At this time, approximately half of the terrain is in sunlight and half is in shadow. Chapter 5 begins by planning an EVA for a suited astronaut that travels through sunlit and shadowed regions. It shows plots of the amount of water required for sublimation cooling and the heater power required for heating. The behavior of both metrics is explained with regards to the components of heat flux to and from the space suit. In particular, the external suit temperature is discussed, which dictates the amount of heat flux through the space suit wall. This temperature changes with the environmental conditions. Chapter 5 then talks about two more EVAs, which represent extreme shadowing conditions. EVA 2 takes place entirely in sunlight, while EVA 3 takes place entirely in shadow. In each of these, the astronaut stops at certain Activity Points for different amounts of time to perform scientific activities. The results show that EVA 2 requires cooling, but not heating, and that EVA 3 requires both at different times. After discussing the astronaut EVAs, Chapter 5 presents a transportation rover traverse. It begins by identifying a desired route with stops for scientific exploration along the way. There are two issues with this traverse: it is over 50 hours long with no allocations for astronaut sleep time, and violates the rover power constraints. Both are alleviated by having the

rover stop for two seven-hour sleep periods at sunlit Activity Points along the traverse. Chapter 5 concludes by showing the rover battery energy along two return-home paths for the rover traverse. One has sufficient power to make it to the habitat, while the other does not.

Chapter 6 reiterates the specific contributions of this work and ends with a discussion of possible future areas of research relating to SEXTANT. While the current work has greatly helped to expand SEXTANT's capabilities in modeling planetary EVA, there are still multiple ways in which its abilities can be extended and improved.

This page intentionally left blank.

A number of papers abound
From researchers of world renown.
And a few about planning,
This chapter is spanning,
For this is the thesis background.

2 Literature Review

When discussing the "path planning" capabilities of SEXTANT, it is important to make the distinction between SEXTANT and autonomous systems that have the ability to plan their own paths. Many autonomous systems, like the MIT Defense Advanced Research Projects Agency (DARPA) Grand Challenge *Talos* automobile and unmanned aerial vehicles, can determine and carry out movements without the intercession of human operators. SEXTANT is similar to the path-planning component of these systems; however, SEXTANT does not automatically carry out a traverse along the planned path. Rather, it presents the information to the human user, who then must make a decision about the path's validity and whether or not it should be carried out. Such a tool is called a decision support aid, at it helps analyze and present information to assist the user.

2.1 Decision Support Tools

One of the most important questions in designing a decision support aid is the amount of automation that it employs. Parasuraman et al. define automation as "the full or partial replacement of a function previously carried out by the human operator" (2000, p. 287). The use of automation can help to increase the efficiency of carrying out a complex task and greatly decrease the amount of time required. However, too much or incorrect automation can lead to sub-optimal results in unanticipated situations, user complacency (called *automation bias*), and a loss of user situational awareness (Parasuraman and Riley 1997). Therefore, it is important to carefully choose the amount of automation in any decision support aid. To support this investigation, Sheridan and Verplank developed a 10-level classification of automation (1978), which was later refined by Parasuraman et al. (2000).

Table 2.1. Levels of automation (Parasuraman et al. 2000)

LOW	1	The computer offers no assistance: human must take all decisions and actions.
	2	The computer offers a complete set of decision/action alternatives, or
	3	narrows the selection down to a few, or
	4	suggests one alternative, and
	5	executes that suggestion if the human approves, or
	6	allows the human a restricted time to veto before automatic execution, or
	7	executes automatically, then necessarily informs the human, and
	8	informs the human only if asked, or
	9	informs the human only if it, the computer, decides to.
HIGH	10	The computer decides everything, acts autonomously, ignoring the human.

2. Literature Review

A great deal of research has investigated the effects of varying levels of automation in a decision support aid. Layton, Smith, and McCoy investigated an experimental decision support aid for re-planning commercial airplane flights around thunderstorms while en-route, called the Flight Planning Testbed (FPT) (Layton et al. 1994, Smith et al. 2007). The FPT had three different possible levels of automation. Arranged from lowest to highest, the system could:

1. Give information (fuel consumption, time, etc.) about a path chosen by the user
2. Suggest alternative routes based on high-level user-specified constraints (like maximum allowable turbulence)
3. Fully autonomously re-plan a path based on pre-set constraints whenever an issue was detected with the original route.

In two different experiments with commercial airline pilots and dispatchers, Layton, Smith, and McCoy showed that automation could greatly help the user to arrive at "optimal" solutions (minimum fuel consumption in their experiments) when there is a large solution space. It is very difficult for a human operating alone to quickly search through all of the possible solutions and find the optimal one. However, they also demonstrated that automation could be "brittle", meaning that it can fail and provide sub-optimal solutions in unanticipated situations. This can occur when the system's model of the "world" is inadequate or when it fails to consider relevant factors. The FPT exhibited brittleness because it did not account for uncertainty in the weather forecast when re-planning routes. When asked to automatically re-plan (either of the higher two levels of automation), the routes were closer to the thunderstorms than the user would place them. If the weather patterns did not behave as forecasted, these new routes could once again encounter thunderstorms. Layton, Smith, and McCoy also found that subjects did not always notice important information due to the large data space, and that subjects often did not question the validity of automatically-generated routes, which weren't always perfect due to the FPT brittleness. These lessons have all been applied to the development of SEXTANT. As with the fight re-planning task, there is a large solution space of possible traverse paths during lunar and planetary surface explorations. For this reason, SEXTANT automatically generates paths between Activity Points to help the user reach an optimal solution quickly. To prevent brittleness, one of this thesis' main contributions – the prediction of shadowing and thermal and power modeling – has been undertaken in order to improve SEXTANT's model of the "world" (a lunar traverse). SEXTANT also attempts to model the sun position as accurately as possible;

for example, it takes into account the movement of the sun with time throughout a traverse. Finally, Layton, Smith, and McCoy suggest that one way to compensate for automation brittleness is to keep the human "in the loop", or involved with the task. SEXTANT accomplishes this by making the user specify the Activity Points. This gives the user control over the destinations of the traverse, and only allows the automation to plan the route between these.

Decision support aids for planning have been extensively used during the Mars Exploration Rover (MER) *Spirit* and *Opportunity* missions (Norris et al. 2005, McCurdy 2009). The tools used during these missions are somewhat different from SEXTANT in that they have been used for temporal scheduling, and not path planning. However, the automation lessons learned are still very pertinent to SEXTANT. The MER planning suite encompasses three different tools, each with similar functions. The Science Activity Planner (SAP) is used to understand downlinked information from the rovers and to develop a list of desired science activities for the next day. Because *Spirit* and *Opportunity* execute these plans autonomously once they are received, the SAP contains simulation tools that help engineers to validate each plan before it is sent to the rovers. These simulation tools use both bottoms-up (relying on pre-flight performance equations) and top-down (based on measurements of the rover's performance) models. The science activity requests developed by the SAP are then read into the Mixed-Initiative Activity Plan Generator (MAPGEN). MAPGEN allows the user to plan rover activities in a timeline view and to access high-resolution rover thermal and power models. The scheduling of activities is done automatically, ensuring that the plan adheres to pre-specified time and resource constraints. The user then must validate the plan. Finally, the Constraint Editor is a separate tool developed in response to the difficulty in generating science constraints during the planning process. It allows users to group activities together and easily create multiple constraints at once.

While these tools have been satisfactory for completing all the major science goals during the MER missions, there have been many issues that have kept the science planning process from reaching its full potential. Most relevant to SEXTANT was the observation that the automated planning tools were not used very frequently (McCurdy 2009). McCurdy hypothesized that this was for two reasons: that it was impossible to capture all science constraints in the planning tools, and that the automatically-generated schedules were difficult to understand. In the next generation of Martian planning tools, the Phoenix Science Interface (PSI), changes were made to

involve humans more in the decision process. PSI used passive constraint checking, where a plan was automatically generated and compared to constraints. If a violation was found, only suggestions for resolution were presented. It was the user's responsibility to ensure that the plan is viable. This is the same technique used by SEXTANT. Traverse paths are not optimized around any certain constraints. Rather, the traverse information is presented with respect to user-specified constraints, which shows how much margin there is, if any. The user can then make the decision on how to best modify the traverse to fit within the constraints.

2.2 Traverse Planners for Planetary Surface Exploration

Planning EVAs for each Apollo lunar mission began with establishing locations of scientific interest. As Bill Muehlberger, chief geologist for the Apollo 16 and 17 missions, states, "The name of the game in traverse planning is maximum scientific return" (1981, p. 12). After sites of interest were selected and prioritized, traverses were planned by hand to maximize the return while remaining within all of the given constraints. The only path planning tools used were photomosaics and topographic maps produced by previous manned and unmanned missions. As a result, there was a great deal of uncertainty about how quickly and well the astronauts would be able to travel over the terrain, and the traverse planners could only speculate. For example, Muehlberger wrote of traverse planning for the Apollo 16 mission by saying, "Although many large blocks were observed in the ejecta of North Ray crater, there appeared to be relatively smooth approaches along which the astronauts could drive to the crater rim or at least to within walking distance of it" (1981, p. 17). This is only one of the numerous instances in which Muehlberger speaks of the planning team's assumptions. These were not always correct, which caused problems during some of the Apollo EVAs when the astronauts could not travel as easily as expected. Furthermore, when these challenges were encountered, the astronauts did not have the ability to re-plan their traverses by themselves. They had to radio down to the engineers and scientists on Earth, who would re-plan the EVA and direct the astronauts from the ground.

Since the Apollo program, only a few researchers have looked at improving traverse planning tools. Wilkinson developed the Traverse Generation Assistant (TGA) for long-duration pressurized rovers traverses on Mars (2004). TGA characterizes the terrain by three factors: elevation, terrain type (ranging from "easy to traverse" to "impassible") and scientific interest. Users can specify a starting location and a destination, and optimize the path between the two on many different metrics: the distance traveled, travel time, amount of terrain variation, number of

scientifically interesting sites visited, and an overall cost metric that combines the other four with relative weightings. A traverse report is generated for each planned path, which includes the total distance traveled, changed in elevation, travel time, and total traverse difficulty. While TGA is similar to SEXTANT in many ways, its model of the traverse is more limited than that of SEXTANT. TGA does not consider the position of the sun during the traverse, something that Wilkinson acknowledges. He remarks that this in less important for Martian explorations than for lunar ones because of the increased distance between the Sun and Mars. More significantly, TGA does not consider the rover's energy consumption during a traverse. Wilkinson does note that the metabolic cost of a traverse has been left out of the model because TGA is for planning rover traverses. This is true, but the rover energy cost for a traverse is a critical metric for determining whether or not the traverse is valid. A traverse often has distance or time constraints, but these are mainly designed to ensure that the explorer has enough consumables. For a rover, battery power is one of the most important consumables. SEXTANT allows the user to optimize a rover traverse on its energy cost, and also presents the energy consumption for all traverses. This allows the user to clearly see if there will be enough consumables to complete the EVA.

Wood and Wood developed a traverse planning tool of sorts that *does* take into account the metabolic cost of travel; however, the goal of their tool is quite different than that of TGA or SEXTANT (2006). Their tool is for use in archaeology on Earth, and is designed to compute geographic closeness on the metric of expended energy. In archaeology, the closeness of two locations is often thought of in terms of the straight-line distance ("as the crow flies"). However, Wood and Wood argue that a better metric of closeness is the metabolic cost that it would have taken ancient peoples to travel between the two locations. Their tool allows the user to select a digital elevation model that describes a certain terrain and optimize a path between two points on either distance or metabolic cost. Additionally, the user can create an "energetic terrain" visualization, which shows the amount of metabolic energy required to reach any point in the terrain, as is seen in Figure 2.1.

Wood and Wood's planning tool is similar to SEXTANT in that both have the ability to optimize traverses on the metric of metabolic cost. In fact, both tools use the same metabolic cost equations developed by Santee et al. (2001). Even so, the capabilities of Wood and Wood's tool are less flexible than those SEXTANT. In Wood and Wood's tool, the user can only optimize the path between two points. In SEXTANT, the user can create an entire traverse composed of

multiple intermediate points of interest. This allows for SEXTANT to capture all of the goals of a particular exploration. The differences between Wood and Wood's tool and SEXTANT can simply be explained by their different purposes.

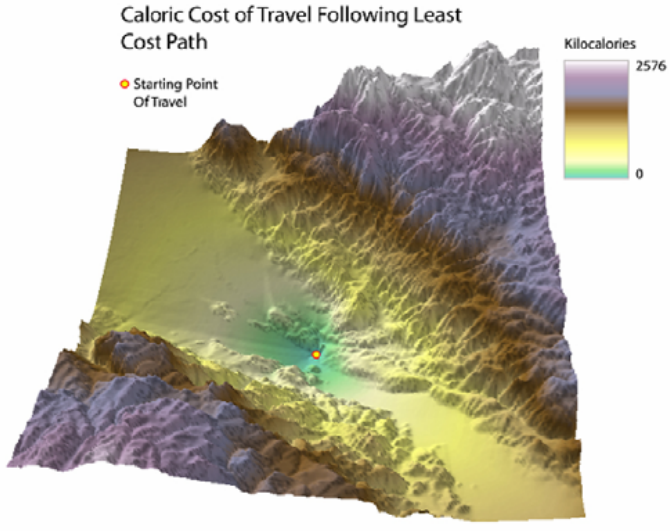


Figure 2.1. Information visualization produced by Wood and Wood's traverse planning tool (2006)

2.3 Development of SEXTANT

SEXTANT has been in development in the MIT Man Vehicle Lab since 2001. Over time its capabilities have been greatly expanded by a series of students. The first implementation of SEXTANT was called the Geologic Traverse Planner, developed by Carr (Carr 2001, Carr et al. 2003). This tool, written in MATLAB and Excel, was designed to help the user optimize traverses across a planetary surface. The Geologic Traverse Planner read in a digital elevation model of a terrain of interest, over which the user selected ordered waypoints along the traverse. After all waypoints were selected, the traverse was drawn between each set of waypoints as a straight-line path. Information about the traverse was provided as a total "exploration cost." This took into account both the metabolic cost of carrying out the traverse, and a "sun score", which quantified how good the traverse was with respect to the sun location. The sun score was 1 when the traverse was directly into or away from the sun (undesirable conditions), and was 0 when the traverse was perpendicular to the sun (the desirable condition). The Geologic Traverse Planner also determined whether the planned traverse violated any of the following constraints set by the user:

1. Maximum slope
2. Maximum allowed metabolic rate
3. Maximum average metabolic rate
4. Walk-back constraint (the maximum allowable distance to a known "safe haven" location)
5. EVA duration

With this information, the user was required to manually change the traverse waypoints in order to either remove constraint violations or further reduce the exploration cost.

Next, Márquez investigated the informational and functional requirements for an EVA path planning decision support aid (Márquez and Newman 2006, Márquez and Newman 2007, Márquez 2007, Márquez and Cummings 2008). These were developed through work-domain analyses of Apollo EVAs and Martian analog traverses at the Mars Institute's Haughton-Mars Project base on Devon Island, in the Canadian Arctic. The informational requirements detail the information that must be handled and displayed by any traverse path planning decision support aid. Márquez developed a Planetary EVA Framework, seen in Figure 2.2, which captures all of the variables that make up this information space.

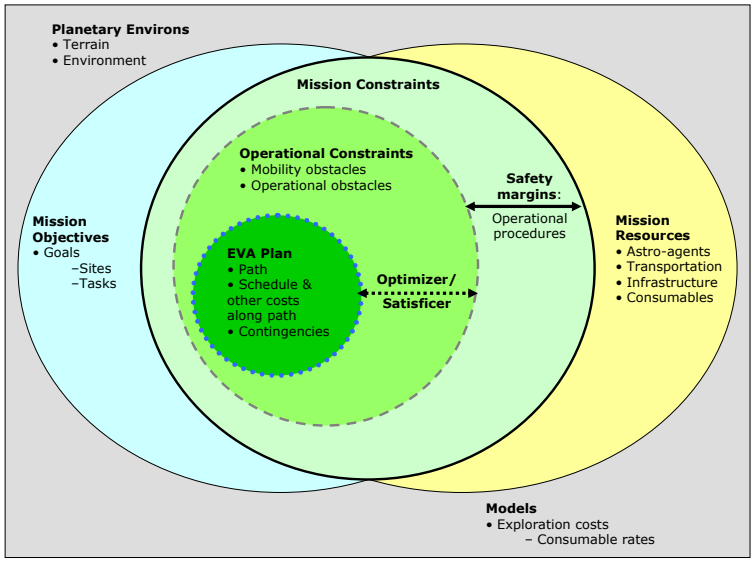


Figure 2.2. Planetary EVA framework (Márquez 2007)

Márquez recommended that the planned traverse be overlaid on a terrain map consisting of both elevation data and aerial images, as both are important in planning the path and determining exploration sites. Furthermore, the traverse schedule and metrics of time, distance,

and energy cost should be clearly displayed to the user. This allows him to make judgments about the validity of the traverse before it is undertaken, and allows the astronaut conducting the exploration to see if he is on track en-route. Finally, Márquez recommended that the amount of uncertainty in the underlying EVA models be displayed.

With these informational requirements, Márquez greatly expanded upon the capabilities of Carr's Geologic Traverse Planner. The goal of this new system, written in Java and called the Planetary Aid for Traversing Humans (PATH), was designed to experimentally investigate three different path cost visualizations and two levels of automation. PATH had the ability to display elevation contours; lines of equal cost, which showed areas on the map that could be reached from the starting point with approximately the same metabolic cost; or a combination of the two. In the lower of the two levels of autonomy (called the "passive" automation), the user was responsible for planning paths and the automation simply relayed information. This is level of autonomy (LoA) 2 on Parasuraman et al.'s scale (Table 2.1). With the higher level of autonomy (called the "active" automation), LoA 4, PATH automatically planned the traverses based on only one intermediate waypoint specified by the user. This optimization used Dijkstra's algorithm to find the minimum cost path. Márquez ran a series of experiments with PATH, asking subjects to optimize traverses on different metrics relating to the sun position. Márquez found that while users with active automation had less costly paths and were able to generate them faster than users with passive automation, they also spent less time performing manual sensitivity analyses. In other words, the users did not modify the path waypoints to see how the costs were affected. This, in turn, led to a loss of situational awareness. SEXTANT currently uses this higher level of autonomy, but also keeps the human involved during the planning process. This was one of Márquez's largest recommendations for improving situational awareness in a future version of PATH. In SEXTANT, the user specifies *all* of the major waypoints along the path (the Activity Points). The path between the Activity Points is automatically optimized, but the user still has good knowledge about where the path leads. This also encourages him to conduct a manual sensitivity analysis by moving the Activity Points around to create different paths. Having the user involved in this process serves to increase his situational awareness, more than the active automation condition of PATH. SEXTANT also models all aspects of the traverse as accurately and completely as possible, and display this information to the user. The user can see basic information about the terrain model, like the

elevation and slope. Additionally, the distance, time, energy cost, and thermal metrics or rover battery energy of the traverse is presented graphically, to show the user which portions of the traverse are most costly. All in all, SEXTANT attempts to gain the benefits of automation while compensating for the commonly-encountered problem of decreased situational awareness.

Development of PATH was continued by Lindqvist, who sought to expend its capabilities from a research tool with limited flexibility to a complete mission planner that could be used in real-world situations (2008). This was accomplished by uniting the optimization capabilities of PATH with the mapping features of the ArcGIS geographical information system (GIS) software suite. In this integrated mission planner, ArcGIS served as the interface, allowing the user to view terrain obstacles and specify waypoints directly on a terrain map. This system had the ability to take in *any* digital elevation model, expanding the domain of PATH to virtually anywhere on Earth. Once the waypoints were selected, PATH was called as a separate function to optimize the traverse on its exploration cost. This exploration cost included the metabolic expenditure and sun score. Once the traverse path was generated, the user could view it overlaid on either a two- or two-and-a-half-dimensional (2D or 3D) terrain map in ArcGIS. Lindqvist validated the integrated PATH-ArcGIS system through field tests around the MIT campus in the winter of 2007. While issues were encountered with the global positioning system (GPS) device tracking the subjects, the path planning functioned correctly.

While Lindqvist integrated PATH and ArcGIS together as two separate programs, Essenburg combined the functionality of both into one program, called Pathmaster (2008). This tool, written in MATLAB, greatly improved the user interface for the PATH algorithms, making it easier to plan traverses and view the desired information. The current SEXTANT interface is a modified version of the Pathmaster interface with more functionality added, and so details will be discussed in Chapter 3. Briefly, the main Pathmaster interface is a 3D elevation map over which the user can modify obstacles and waypoints and see the generated traverse. This map is created from a matrix of elevations, which for Earth can be exported by GIS tools like ArcGIS. Through the mapping interface, the user can also view elevation and slope information about any point in the terrain model. Pathmaster also allows for traverses to be created for multiple explorers with different physical characteristics (i.e., weight) and waypoints, all over the same terrain. After placing traverse waypoints for all explorers, the paths are optimized on the metric of metabolic cost using an A* (pronounced “A-star”) graph search algorithm (Johnson 2008).

This algorithm greatly reduced the execution time of SEXTANT over Dijkstra's algorithm used in PATH, while still returning nearly-optimal paths. The A* search is also still used by SEXTANT, and will be discussed in depth in Section 3.2. This optimization corresponds to LoA 4, which Márquez had found to cause some degradation in user situational awareness (Márquez and Cummings 2008). However, there are still multiple ways in which Pathmaster encourages sensitivity analysis, and therefore increases situational awareness. For example, multiple instances of Pathmaster can be run on the same computer at the same time. This allows the user to plan different paths and directly compare the costs without deleting and re-planning traverses. Essenburg's work concluded with two field tests of Pathmaster. The first took place at the NASA Jet Propulsion Lab (JPL) Mars Yard[1], and the second at the Haughton-Mars Project. Pathmaster was used to plan simulated EVAs for human "astronauts" in both, and also for a remote-controlled rover in the JPL tests. During the tests in the JPL Mars Yard the rover overheated and then broke down, which required the development of contingency plans. This demonstrated that Pathmaster could be used for traverse re-planning during EVA. During the tests at Haughton-Mars, Pathmaster was shown to be useful in situations where the astronaut had to travel long distances. With the conclusion of these tests, the Pathmaster project was turned over to the author of this thesis, where it became SEXTANT.

2.4 Conclusion

The use of automation for traverse planning in SEXTANT is beneficial because it helps the user to search the large solution space of possible paths and find the optimal one. However, automation has also been shown to have inherent issues, like brittleness or a loss of user situation awareness. Much research has been done, both with SEXTANT's predacessors and decision support aids in general, that provides recommendations on how best to mitigate these issues. This chapter discusses a number of examples and their lessons for SEXTANT. For example, to avoid brittleness, SEXTANT attempts to model lunar traverses as accurately and completely as possible. The incorporation of shadowing and the related astronaut thermal model and rover

[1] The author of this thesis was an intern at JPL during this time, in a completely different division. He was still deciding on potential research projects at MIT, one of which was the mission planner project. He visited the Mars Yard twice, hoping to see the Pathmaster tests, but hardware issues prevented him from doing more than standing around in the California sun, trying to help fix the rovers.

power model greatly help to do this. To prevent a decrease in user situational awareness, SEXTANT attempts to engage the user in the path planning task by relying on him to specify Activity Points on the planetary surface. These are very important considerations, which help SEXTANT to best leverage the advantages of automation. As a result, SEXTANT is the most comprehensive and accurate lunar and planetary traverse planner in the literature. Two other traverse planners discussed in this chapter have similar functions, but neither has the breath and depth of ability that SEXTANT does.

To help plan a lunar traverse
That's clear of locations adverse,
SEXTANT gives you a clue
Through its terrain map view,
And iMAS, with which you converse.

3 Basics of SEXTANT

SEXTANT is a tool that comprehensively represents traverses occurring across a planetary surface. The user can plan explorations for suited astronauts on foot and transportation rovers, and see if these traverses are feasible with respect to constraints on distance, time, energy cost, and thermal load (for astronauts) or power consumption and generation (for rovers). This chapter will present the traverse representation in SEXTANT and discuss the SEXTANT user interfaces, including the 3D mapping interface through which the user plans a path and views the related traverse data. It will also discuss SEXTANT's integration with the Individual Mobile Agents System (iMAS) developed by NASA Ames Research Center, which allows for real-time navigation along the planned path.

3.1 Traverse Representation in SEXTANT

Any traverse is represented in SEXTANT by three separate components:

1. Exploration Objectives, which characterize the activities of the traverse between which the traverse path is created
2. Environment Model, which represents the planetary environment in which the traverse is occurring
3. Explorer Model, which details the energy consumption and constraints of the explorer performing the traverse

Each of these describes a different facet of the traverse, and is equally important.

3.1.1 Exploration Objectives

Each Exploration Objective is called an *Activity Point* and consists of three parts: a specific activity, location, and duration. The activity can represent a number of tasks: sample collection, resource exploration, or scouting activities, for example. For Earth-based traverses, the activity locations can be represented as geographic coordinates in either the Universal Transverse Mercator (UTM) system, or by latitude and longitude. For Activity Points on the Moon, locations can be expressed as indices corresponding to a location on the terrain elevation map. The duration is the time spent in place at an Activity Point performing the specified tasks.

3.1.2 Environment Model

Regardless of the planet on which the traverse is occurring, the Environment Model consists of three parts. These are the:

1. Terrain elevation matrix
2. Terrain slope matrix
3. Terrain obstacle matrix

When SEXTANT is being used to plan lunar traverses, there is an additional fourth component to the Environment Model:

4. Location of sunlit and shadowed areas on the lunar surface

The fundamental component of the Environment Model is a user-inputted elevation map. This map is a matrix specifying the elevation of equally spaced points on the terrain surface. The distance between points is called the horizontal map resolution. Determining the required map resolution for a certain traverse is a trade-off between processing speed and simulation fidelity. A high resolution is desirable (on the order of meters, for example) because it gives a more precise representation of the terrain. Smaller surface features, like steep cliffs or boulders, can be captured on high-resolution maps, but may be missed on lower-resolution maps. However, the higher the resolution for a given area, the more points the terrain map will contain. This in turn leads to greater computational complexity and slower processing speed. The trade between processing speed and simulation fidelity must be determined for each specific use; there is no general solution that works for all traverses.

SEXTANT has the ability to plan traverses over both terrestrial and lunar terrain, depending upon the inputted elevation map. Earth-based elevation information can be gained from any GIS program, such as ArcGIS. The lunar elevation map currently used in SEXTANT derives from data from the Lunar Orbiter Laser Altimeter (LOLA) instrument on the Lunar Reconnaissance Orbiter (LRO) spacecraft launched in 2009 (Mazarico et al. 2009, Mazarico et al. 2010, Smith et al. 2010a, Smith et al. 2010b). This lunar data has been acquired in partnership with the MIT Department of Earth, Atmospheric, and Planetary Sciences and its chair, Professor Maria Zuber. The elevation map covers a 500 km by 500 km area centered around the lunar south pole, which has been an area of interest for NASA's future lunar exploration due to the water deposits that were predicted to be there and confirmed by the Lunar Crater Observation and Sensing Satellite (LCROSS) mission. This map has a horizontal resolution of 240 m, which

is somewhat larger than the desired resolution for future lunar EVA. Fine terrain details will be missed, giving a less accurate model of the lunar terrain. However, the ultimate horizontal resolution of LOLA is 10 m, which is an order of magnitude finer than any current elevation map of the entire lunar surface (Smith et al. 2010a).

Once the elevation map has been inputted within SEXTANT, the magnitude of the local slope at all points is calculated using a gradient operation. The terrain slope is an extremely important parameter, as the energy consumption of both human astronauts and transportation rovers directly depends on it. Like the elevation data, the slope information (a scalar value) can be stored in a matrix of the same size, where each element corresponds to the terrain slope at that particular location.

Next, SEXTANT determines areas where the explorer cannot travel due to safety constraints. These areas are called *obstacles*, and are features where an explorer could easily fall, slip, or otherwise run into problems. Obstacles are defined in part by a user-specified maximum slope. Any point with a slope magnitude greater than this limit is considered to be an obstacle. However, this does not fully describe all obstacles. There may be features below the resolution of the map (like boulders or areas of deep dust) that prohibit safe passage and are not captured on the elevation or slope maps. For this reason, the user has the ability to manually specify additional obstacles. Obstacles are represented as a matrix of identical size to the elevation map populated with Boolean values. A 1 represents that a certain point is an obstacle, and a 0 represents that it is not an obstacle. By marking multiple adjacent points as obstacles, an entire area can be rendered impassable.

The previous three components of the Environment Model – terrain elevation, terrain slope, and obstacles – are required for both Earth- and lunar-based scenarios. The fourth component of the Environment Model, the location of sunlit and shadowed areas, has only been determined for traverses on the Moon. Unlike the other three components, shadowing changes with time; therefore, it would be very time-consuming to create matrices of this information for all points in the terrain map. As a result, shadowing is only determined for points along the traverse after a path has been generated. Shadowing at a point is represented as a value from 0 to 1, which represents the percentage of the solar disk that is visible from that location. A value of 0 is complete shadow, a value of 1 is in complete sunlight, and a value between 0 and 1 is in partial shadow. Details about the determination of this metric will be discussed in Section 4.2.

In the future, the Environment Model can also take into account detailed features of the terrain, like the surface roughness. This is important because an explorer's speed and energy consumption both depend on the terrain surface, and not just the slope. Both astronauts and transportation rovers will travel slower over rocky and jagged portions of the lunar surface than over smooth, hard-packed regolith. LOLA has the capability to measure the surface roughness of the lunar surface and provide the necessary data to be incorporated into SEXTANT. Additionally, the Explorer Model can be updated to account for the varying speed and energy consumption over different terrain.

3.1.3 Explorer Model

The Explorer Model represents the explorer carrying out the traverse. It also comprises the metrics on which the most efficient path can be determined: the path distance, the travel time, or the energy cost to travel along a path. Whichever of these is being used is called the *cost function*. The efficiency of a traverse increases as the distance, time, or energy cost required to travel it decreases.

SEXTANT contains explorer models for both human astronauts and transportation rovers, both of which will be heavily involved in future planetary and lunar exploration. Both models have the same structure, which can be seen in Figure 3.1 below.

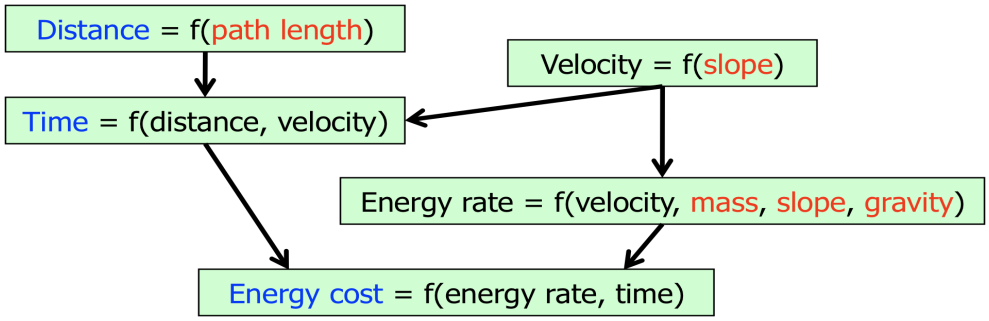


Figure 3.1. Structure of explorer model within SEXTANT

The three metrics of distance, time, and energy cost shown in blue in Figure 3.1 are all functions of four parameters, shown in red:

1. Path length
2. Terrain slope
3. Planetary gravity
4. Explorer mass

3.1.3.1 Human Astronauts

The traverse distance is determined as the length the astronaut travels along the planned path. It is not just the straight-line distance between Activity Points. The explorer velocity (m/s) parallel to the terrain surface is solely a function of the slope (º). The relation for human astronauts was developed by Márquez from data on the Apollo 14 mission compiled by Waligoria and Horrigan (Márquez 2008, Waligoria and Horrigan 1975).

Table 3.1. Velocity equations for suited astronaut explorers (Márquez 2008)

Slope, α (°)	Velocity (m/s)
$-20° \leq \alpha < -10°$	$0.095 \bullet \alpha + 1.95$
$-10° \leq \alpha < 0°$	$0.06 \bullet \alpha + 1.6$
$0° \leq \alpha < 6°$	$-0.2 \bullet \alpha + 1.6$
$6° \leq \alpha < 15°$	$-0.039 \bullet \alpha + 0.634$
$15° \leq \alpha \leq 20°$	0.05

Figure 3.2 shows the astronaut velocity from these equations plotted against the terrain slope.

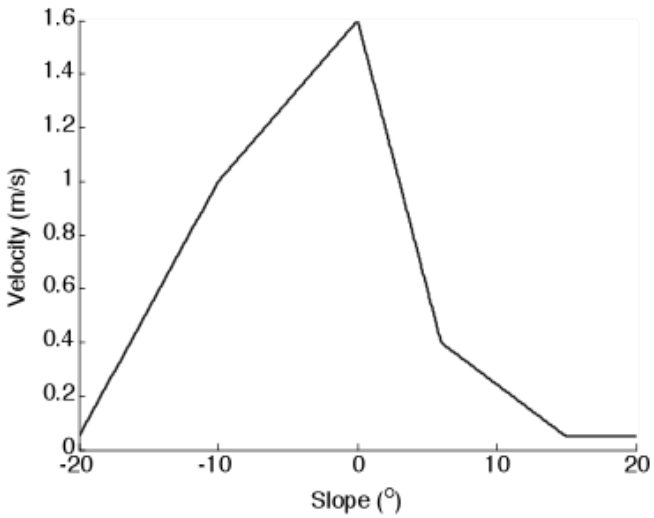


Figure 3.2. Velocity plotted with respect to slope for suited astronauts

The traverse time is then calculated as distance divided by velocity. The astronaut energy rate (W) is a function of his velocity and mass (kg), as well as the terrain slope and planetary gravitational acceleration (m/s^2) (Santee et al. 2001). Because the velocity, v, given in Table 3.1 is measured parallel to the terrain surface, it must be broken into horizontal ($v \bullet \cos(\alpha)$) and vertical ($v \bullet \cos(\alpha)$) components. The horizontal component of velocity is used to find the energy rate when traveling on a level surface, and the vertical component of velocity is used to find the

energy rate when traveling up- or downhill. Explorer mass is important because a heavier astronaut requires more energy to move. The explorer mass is the astronaut's suited mass, along with any extra equipment or samples being carried. The planetary gravity also affects the energy rate, as movement in the reduced gravity on the surface of the Moon requires less energy than on Earth. The metabolic energy rate is related to these parameters by the equations set forth in Table 3.2.

Table 3.2. Metabolic rate equations for suited astronaut (Santee 2001)

Metabolic Rate (W) = $W_{level} + W_{slope}$	
$W_{level} = [3.28 \cdot m + 71.1] \cdot [0.661 \cdot v \cdot \cos(\alpha) + 0.115]$	
Slope, α	**W_{slope}**
$\alpha = 0°$	0
$\alpha > 0°$	$3.5 \cdot m \cdot g \cdot v \cdot \sin(\alpha)$
$\alpha < 0°$	$2.4 \cdot m \cdot g \cdot v \cdot \sin(\alpha) \cdot 0.3^{\lvert\alpha\rvert/7.65}$

These equations break the energy rate into two components: the energy required for locomotion over a level surface (W_{level}), and the energy required to travel up or down a slope (W_{slope}). m is the explorer mass, v is the explorer velocity, g is the planetary gravity, and α is the terrain slope. Figure 3.3 plots this energy rate per meter, as a function of the terrain slope.

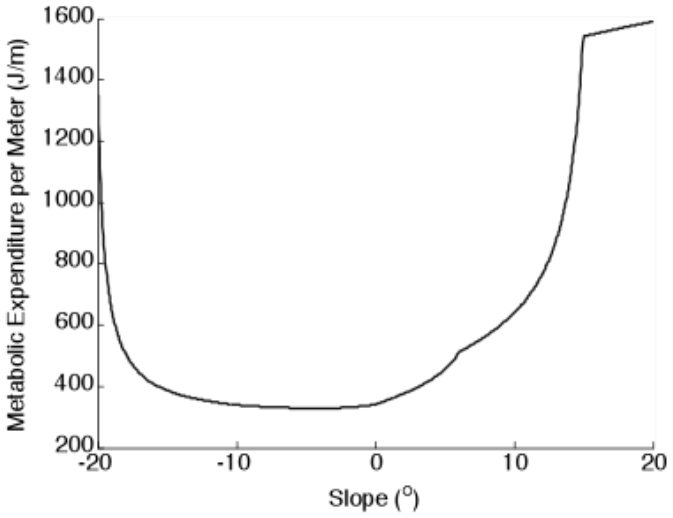


Figure 3.3. Metabolic rate plotted with respect to slope for suited astronauts

Finally, the total metabolic cost (J) of the traverse can be found by multiplying the energy rate (W) by the travel time (s).

3.1.3.2 Transportation Rovers

Unlike astronauts, the rover velocity is assumed to be a constant 15 km/hr (4.17 m/s) over all traversable terrain slopes. However, the transportation rover energy rate varies with the terrain slope. The energy rate (W) is a function of the same parameters as the astronaut's metabolic rate: the rover velocity (m/s) and mass (kg), and the terrain slope (º) and planetary gravitational acceleration (m/s^2) (Heiken 1991, Carr 2001). The rover mass consists of the rover structure and all payload, including the astronauts. Because the equations were developed from historical data on the LRV, the equations are normalized to lunar gravity (1.62 m/s^2). The energy rate is related to these parameters by the equations set forth in Table 3.3.

Table 3.3. Energy rate equations for transportation rovers (Carr 2001)

Energy Rate (W) = $W_{level} + W_{slope} + P_e$	
$W_{level} = 0.216 \bullet v \bullet m$	
Slope, α	W_{slope}
$\alpha = 0°$	0
$\alpha > 0°$	$0.02628 \bullet m \bullet \alpha \bullet (g/1.62) \bullet v$
$\alpha < 0°$	$-0.007884 \bullet m \bullet \alpha \bullet (g/1.62) \bullet v$

As with the astronaut explorer model, there is a component of the every for traveling across flat terrain (W_{level}) and up- or downhill (W_{slope}). The coefficients in the equations for these two components come from Heiken et al., which give the LRV "mass mileage" as 0.050 – 0.080 W-hr/km/kg on level terrain and an additional 0.0073 W-hr/km/kg/º when traveling up- or downhill (1991). There is also an efficiency factor that accounts for energy recovery when traveling downhill. m is the rover mass, v is the rover velocity, g is the planetary gravity, and α is the terrain slope. Unlike suited astronauts, there is an additional component of energy, the collection of all other electronic components in the rover (P_e). This exists even when the rover is stationary. This includes the avionics, communications, life support, thermal control, and scientific equipment. This energy consumption should change throughout the traverse depending upon the activities the crew is undertaking at a certain time. Nevertheless, this is not captured within SEXTANT, which does not have the ability to represent details of the activities being

performed during the exploration. As a result, this non-transportation energy consumption is constant throughout the traverse. Currently in SEXTANT, this is estimated to be 1500 W (Hong 2007). However, this can be easily changed by the user to support different rover requirements. Finally, the total metabolic cost (J) for the traverse is determined by multiplying this energy rate (W) by the travel time (s).

This energy rate per meter is plotted against the terrain slope in Figure 3.4.

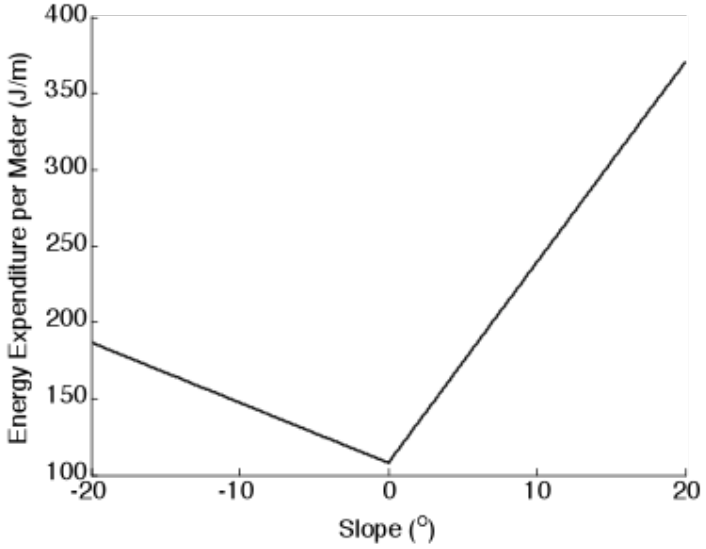


Figure 3.4. Energy rate plotted with respect to slope for transportation rovers

3.2 Optimization Algorithm

SEXTANT employs an A* (pronounced "A-star") graph search algorithm to compute the most efficient path between Activity Points, regardless of which metric is being used as the cost function. This algorithm, developed by Hart et al., optimizes the path between two points on the metric of lowest cost (1968). In the A* search all points in the terrain map are represented as nodes, and all possible paths from one node to an adjacent node – located directly laterally or diagonally – as edges. The A* search is called multiple times for a traverse, each time finding the most efficient path between two consecutive Activity Points. Once an Activity Point is reached as the goal of one A* search, it becomes the starting node for the subsequent A* search.

An instance of the A* search begins at a starting node *S* and attempts to find the least costly path to a goal node *G*. Both *S* and *G* are Activity Points that have been specified by the user. In the algorithm, the cost of moving from any node *M* to another node *N* along any edge is a combination of two factors:

1. The cumulative cost of moving from the starting node *S* to node *N* through the path already taken from *S* to *M*
2. A heuristic function that estimates the cost of moving from node *N* to the goal node *G*

The heuristic function dictates the behavior of the A* search in finding the least costly path, and is a trade-off between speed and accuracy. A low heuristic will give a more accurate result, meaning that the path cost will be closer to the true least-costly path between *S* and *G*. In fact, as long as the heuristic function is *admissible*, meaning that it underestimates the true cost of moving from *N* to *G*, the A* search is guaranteed to find the true least-costly path (Patel 2010). However, the search will explore more nodes in finding this path, and will take longer to complete. On the other hand, a high heuristic will run faster but give less accurate results.

SEXTANT's heuristic function $h(N)$ is admissible, and is described by Equation (3.1) (Patel 2010).

$$\begin{aligned} h_{diagonal}(N) &= \min\left(|N_x - G_x|, |N_y - G_y|\right) \\ h_{straight}(N) &= |N_x - G_x| + |N_y - G_y| \\ h(N) &= D_2 * h_{diagonal}(N) + D * \left(h_{straight}(N) - 2h_{diagonal}(N)\right) \end{aligned} \tag{3.1}$$

$h_{diagonal}$ is the number of diagonal steps that the explorer can take diagonally between node *N* (which has an x-coordinate of N_x and a y-coordinate of N_y) and the goal node *G* (which similarly has an x-coordinate of G_x and a y-coordinate of G_y). $h_{straight}$ is the Manhattan distance (also called the rectilinear distance) between *N* and *G*, which is the sum of the x- and y-distances between the two nodes. This is the shortest distance between *N* and *G* if the explorer could only move laterally. The quantity ($h_{straight}(N) - 2h_{diagonal}(N)$) is the number of lateral steps the explorer must take between *N* and *G* after moving $h_{diagonal}$ diagonal steps. So, the total $h_{diagonal} + (h_{straight}(N) - 2h_{diagonal}(N))$ is the minimum number of steps the explorer can take between *N* and *G*. The number of lateral and diagonal steps must each be multiplied by the cost of moving in these directions. D_2 is the minimum cost of moving from one node to another node diagonally, and *D* is the minimum cost of moving laterally between two nodes. Since the grid of nodes is regular, $D_2 = D \cdot \sqrt{2}$. Because the heuristic guarantees the minimum number of steps between *N* and *G*, these costs must also give the lowest energy possible. This will ensure that the heuristic does not overestimate the actual cost to travel from *N* to *G*, making the heuristic admissible. Figure 3.3 and Figure 3.4 in Section 3.1.3 show that there is a terrain slope value at which the energy cost

per unit distance is a minimum. Table 3.4 details these slopes and the resultant energy costs for three conditions: astronauts on Earth, astronauts on the Moon, and transportation rovers on the Moon.

Table 3.4. Minimum energy costs for astronauts and rovers on the Moon and Earth

Condition	Slope of Minimum Energy Cost (º)	Minimum Energy Cost Between Lateral Nodes, *D* (J)
Astronauts on Earth	-5.85º	$[1.504 \bullet m + 53.298] \bullet R$
Astronauts on the Moon	-4.2º	$[2.295 \bullet m + 52.936] \bullet R$
Rovers on the Moon	0º	$[0.216 \bullet m + P_e / 4.167] \bullet R$

m is the explorer mass, R is the resolution of the terrain map, and P_e is the electronics power used by the rover. The A* search begins at the starting node S and determines the total cost to move to all adjacent nodes (a process called "expanding" the nodes). These paths from S and their respective costs are then stored in a queue. The lowest-cost path is taken off the queue, and the A* search travels to this node, called N. It then expands paths to the nodes adjacent to the current node, N. These paths are added to the queue, which still contains the paths from S to its adjacent nodes. If a node is already on the queue, meaning it has been reached by another path, the algorithm determines the least-costly path to that node and stores this on the queue. The lowest-cost path is once again taken off the queue and visited. Because the queue keeps track of *all* expanded nodes, the least-costly path at a certain time may not be from the most-recently expanded node. It may in fact be less costly to "backtrack" to an earlier node and take a different path. This process continues until the algorithm reaches the goal node G, where the search terminates. A* is a best-first search, meaning that each time a node is traveled to (not just expanded) it is by the least-costly route possible. So, once a viable path is created between the starting node and the goal node, there is no need to create alternate paths to check for a better route. For a pictorial description of the A* search, please refer to Appendix A.

3.3 SEXTANT Graphical User Interface

The graphical user interface (GUI) for SEXTANT allows the user to easily interact with the traverse representation and plan explorations across the planetary surface. This GUI consists of two main parts. First, there are the input menus, where parameters of the traverse representation are specified. Secondly, there is the 3D mapping interface which allows the user

to specify Activity Points on a terrain map, run the most efficient path calculation, and view the resultant traverse path and information calculated by the A* algorithm

3.3.1 Initializing SEXTANT

There are two different methods to initialize SEXTANT, depending on the desired terrain map. To plan a new Earth-based traverse or load an existing terrestrial or lunar traverse, the user types >> `sextant` from the MATLAB command prompt to bring up the file input GUI. Here, the user can either select an elevation map (.txt file) to plan an Earth-based traverse or load a MATLAB file (.mat) with an existing traverse. SEXTANT automatically creates .mat files for all traverses. If the user would like to plan a new traverse on the Moon, he must type >> `sextant('moon')` at the MATLAB command prompt. The user is asked to enter the left, right, top, and bottom boundaries of the desired portion of the lunar elevation map. These boundaries are measured from the center of the map (the lunar south pole). A negative value is to the left and below the center, while a positive value is to the right and above. For example, if the user enters [-150 50] for the boundaries in the x-direction, the submap will stretch 150 km left of the map center and 50 km to the right. Furthermore, the user can also specify a less graphically-intense option by typing >> `sextant('lite')` or >> `sextant('moon','lite')`. This version of SEXTANT uses a simpler surface rendering, which requires less computation time and resources. Whatever type of traverse is being planned, the user is taken to the same SEXTANT input menus.

3.3.2 SEXTANT Input Menus

There are two input menus – one for the elevation map parameters and one for the traverse parameters, which influence both the Environment and Explorer Models. Figure 3.5 shows both of these. The elevation map is loaded into MATLAB as a text file containing header information and the elevation matrix. The information in this header automatically populates the fields on the elevation map information dialog box (Figure 3.5, left). Only the map resolution is relevant to both terrestrial and lunar traverses. The UTM zone and coordinates of the lower-left corner of the map are only used for Earth-based planning, and serve to correlate the elevation map with an actual location on Earth.

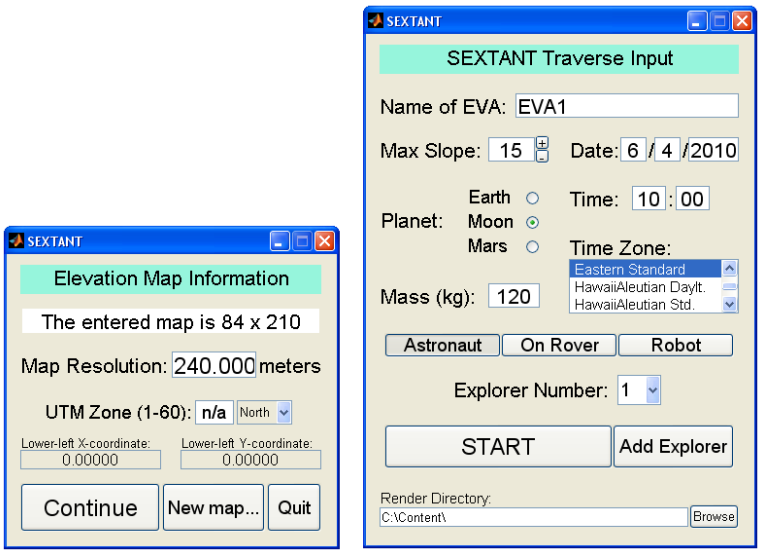


Figure 3.5. SEXTANT input menus

The traverse parameters are set by the user (Figure 3.5, right). The maximum slope defines obstacles on the terrain map. No explorer will be allowed to travel up or down a slope with a magnitude greater than this value. Two other variables important to the energy cost are specified here: the explorer mass and the planet, which dictates the gravitational acceleration. The dialog box asks for the date, time, and time zone (only required for Earth-based traverses) in order to determine the position of the sun at the beginning of the EVA. For lunar traverses, the sun position is then propagated through the traverse and used to calculate the sunlit and shadowed areas on the surface. This will be explained in detail in Section 4.2. For terrestrial EVA, the sun position is used to cosmetically display shadows on the 3D mapping interface. This information cannot be used in determining the thermal load on astronauts or the power generation of transportation rovers, as it can for lunar traverses. Multiple explorers can be added to the simulation, with any combination of astronauts on foot and transportation rovers. Additional explorers can be added with the *Add Explorer* button, and then the explorer number can be selected above to modify its mass and type. The type of a specific explorer can be changed by selecting either the *Astronaut* or *On Rover* button. The third button, *Robot*, does not currently function, but provides the structure for the future ability to plan paths for robotic explorers.

3.3.3 3D Mapping Interface

The main interface for SEXTANT is the 3D mapping interface. This GUI is used to define the traverse Activity Points, start the calculation of the traverse paths, and view information about the planned paths. Figure 3.6 shows the 3D view of a 50 km by 20 km map of the lunar surface, and Figure 3.7 shows a top-down view with Activity Points specified for two explorers.

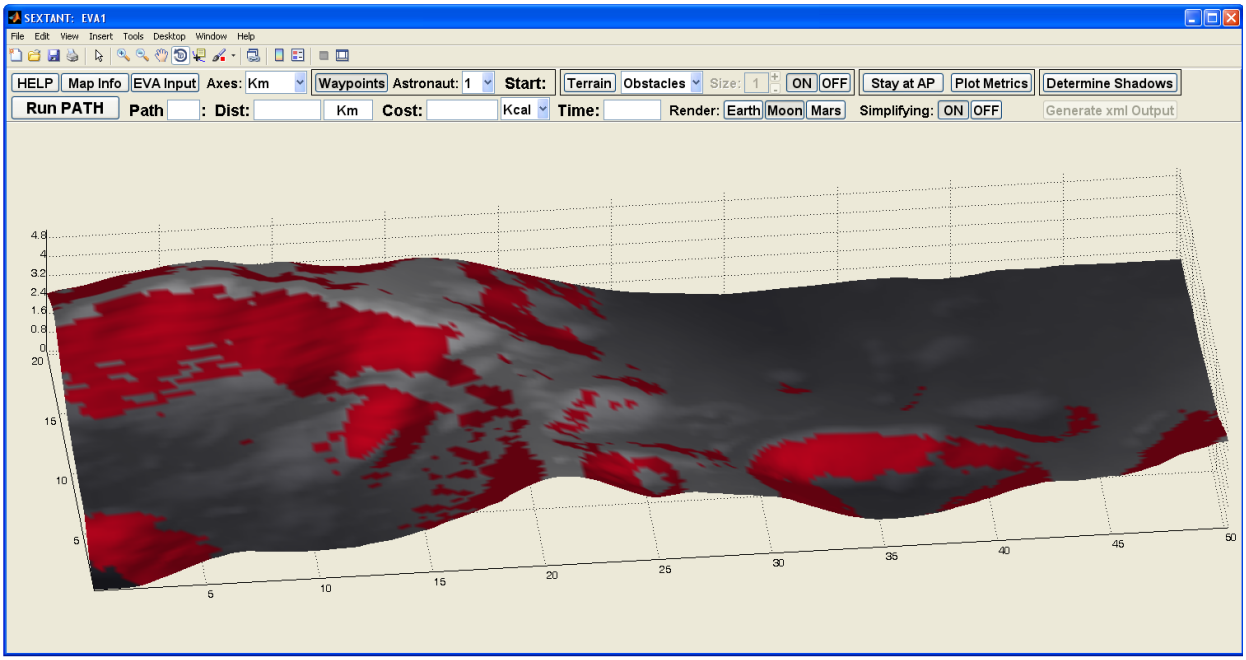


Figure 3.6. 3D view of lunar elevation map in sextant mapping interface

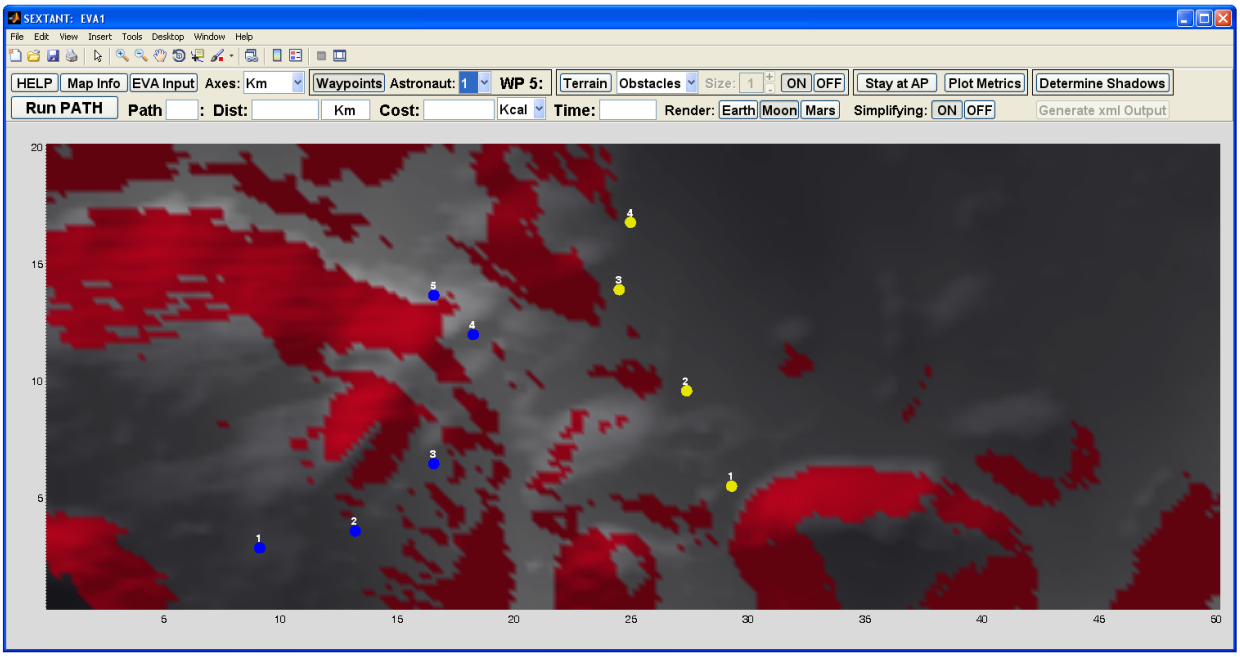


Figure 3.7. Activity Points for two astronauts specified on 3D mapping interface

This 3D view can be helpful in viewing the overall traverse, but it is very difficult to place Activity Points when in this view. In the 3D mapping interface, the accessible lunar surface is colored grey, while the red areas are obstacles as defined by the maximum slope. To adjust the obstacles, the user selects the *Terrain* button at the top of the interface, chooses a tool size between 1 and 10, and then clicks on the map to place a new obstacle or holds the shift key and clicks to remove an existing one. These areas are modified in the obstacle matrix and displayed on the terrain map. Activity Points can be placed on the terrain map simply by clicking the desired location on the terrain map, and must be specified in the order they are to be visited. If the explorer is to remain at any of the Activity Points for a period of time, the durations can be specified by clicking on the *Stay at AP* button once the traverse has been created. The user can generate unique paths for as many astronauts as have been defined in the traverse input dialog box. Each set of astronaut Activity Points is displayed in a different color to easily distinguish between them. In Figure 3.7, there are two explorers, the first marked in blue, and the second in yellow.

Once the user has completely specified the Activity Points for all explorers, the *Run PATH* button can be clicked to generate the most efficient traverse paths between them. The user also has the ability to change the metric on which this most efficient path is determined, be it the traverse distance, time, or energy cost. If any Activity Point is completely surrounded by obstacles and inaccessible by any path, SEXTANT returns an error and will not continue the path calculations. The traverse paths are displayed on the terrain map in the same color as the Activity Points. Figure 3.8 shows traverse paths for the two astronauts. The original user-selected Activity Points have been marked by underlying green circles for easier identification.

Once the traverse paths are computed, they are passed through a simplification algorithm, which reduces the number of points required to describe the traverse (Essenburg 2008). When the traverse is initially calculated, it is composed as a series of movements between adjacent nodes of the elevation map, called *Path Points* (Figure 3.9). The Path Points are marked with small colored circles in Figure 3.8. The simplification algorithm takes all of the nodes in a straight line and removes all but the end nodes, simplifying the depiction of the path. Thus, the final traverse path is displayed as a series of straight lines connected by intermediate Path Points, where the explorer changes direction (Figure 3.10).

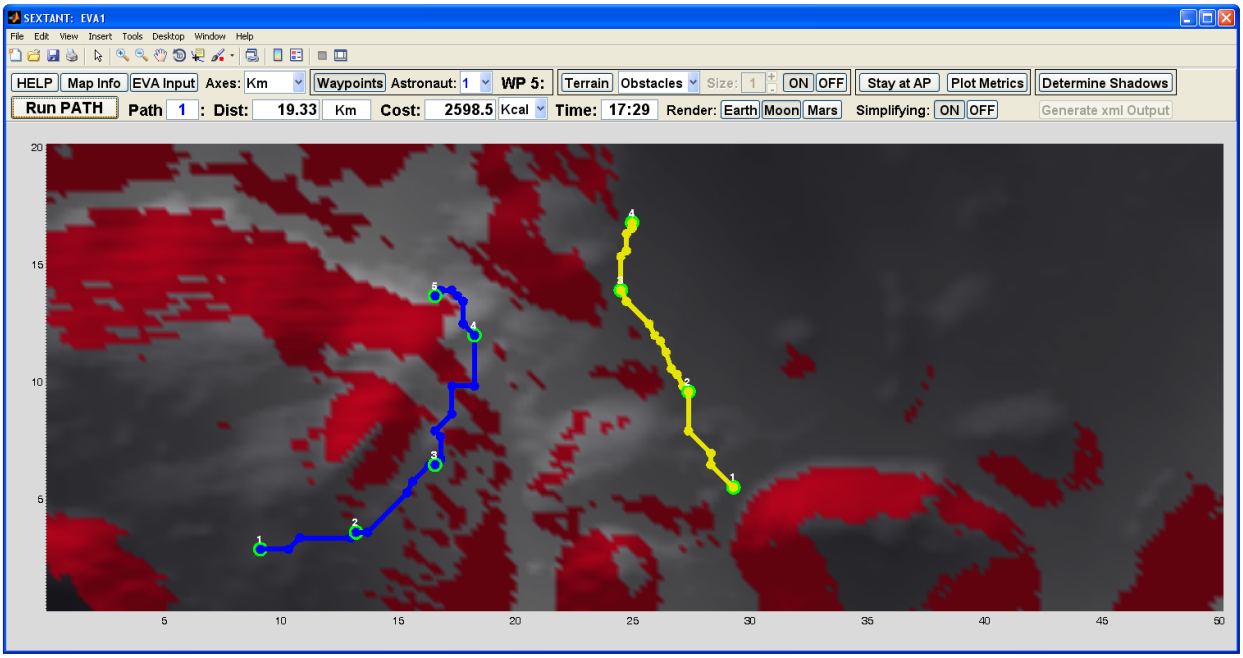


Figure 3.8. Traverse paths for two astronauts on 3D mapping interface

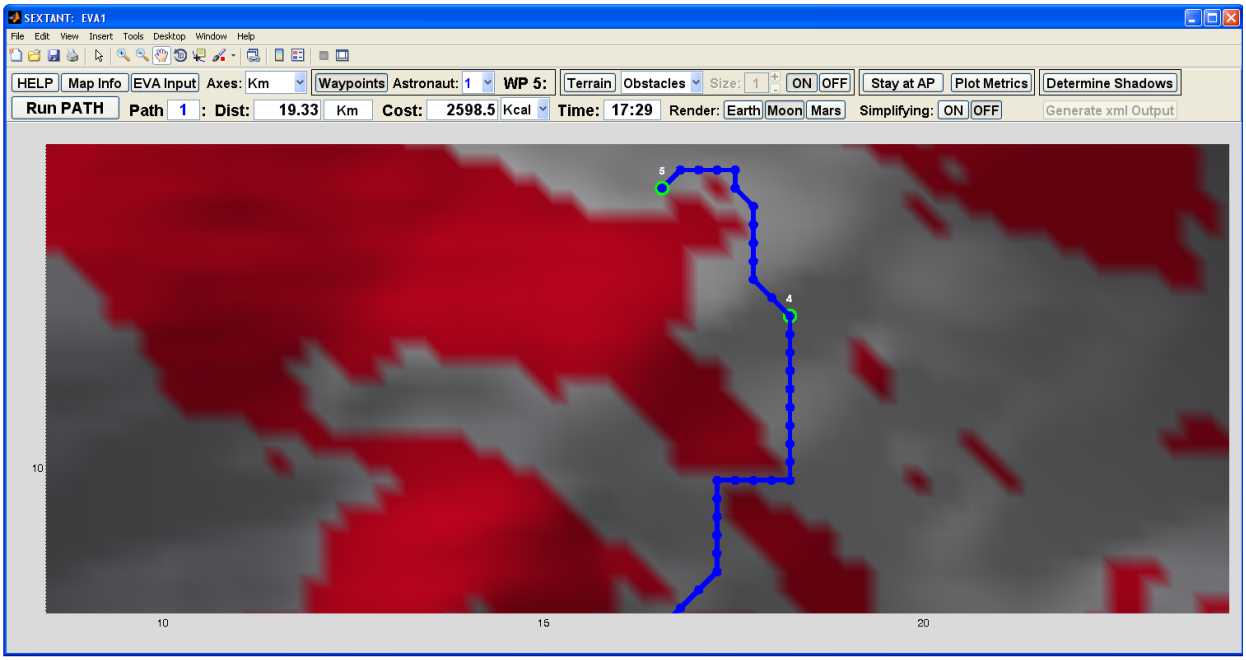


Figure 3.9. Traverse path without simplification

3. Basics of SEXTANT

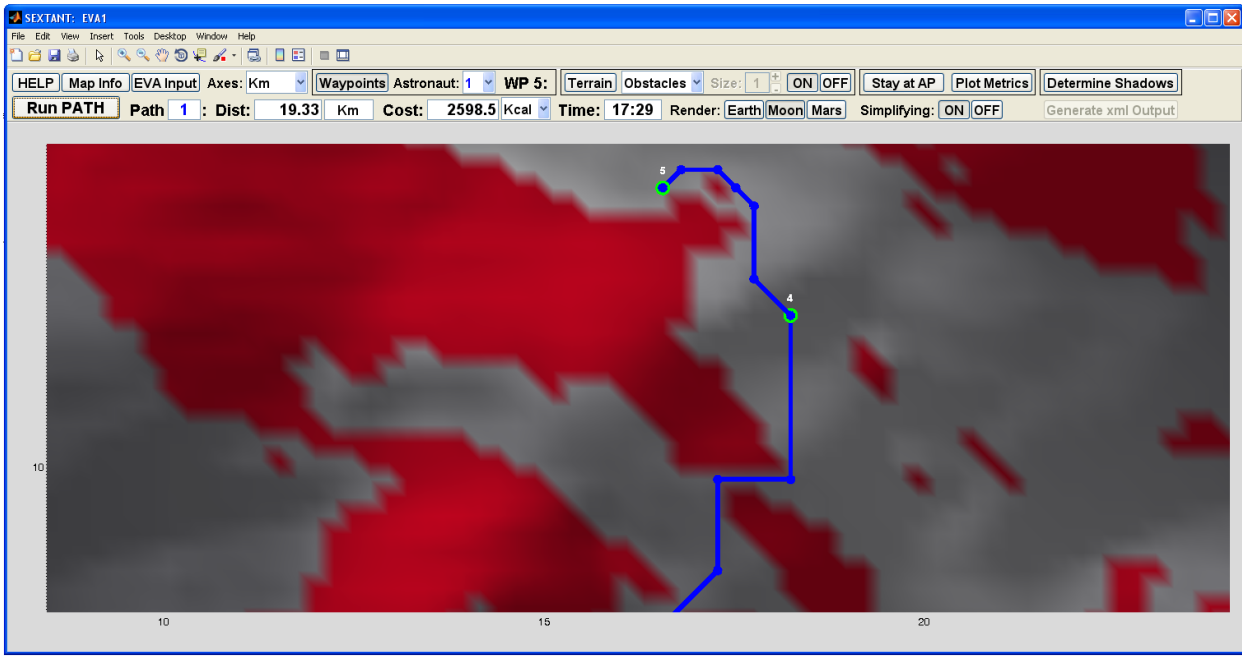


Figure 3.10. Traverse path with simplification

Beyond the determination of the most efficient path, SEXTANT calculates the value of the three possible cost functions at each Path Point, regardless of the explorer type. Specifically, these metrics are the:

1. Cumulative path length from the habitat (the first Activity Point) to the Path Point
2. Time to travel from the habitat to the Path Point along the path
3. Energy cost to travel from the habitat to the Path Point along the path

All three metrics are computed for each Path Point, and the total values for the entire traverse are displayed at the top of the 3D mapping interface. The user has the ability to change the units for both the path length (meters, kilometers, feet, or miles) and the energy cost (kilocalories, BTUs, and kilojoules). The cumulative energy cost is also plotted against the traverse time and distance, to give an overall picture of how this changes throughout the traverse (Figure 3.11 and Figure 3.12).

The sections of the graph with a steeper slope represent more difficult portions of the traverse – where the energy cost is increasing at a faster rate. Areas of the traverse with a lower slope on the graph are easier to travel in comparison. The graphs also display user-inputted time, distance, and energy constraints, indicated by dashed red lines. These show whether or not the traverse is valid, and if it is not, what needs to be changed. In Figure 3.11 and Figure 3.12, it can clearly be seen that the EVA of Astronaut 1 (Figure 3.11) is barely within the distance and

metabolic cost constraints, but greatly exceeds the time constraint. The traverse must be re-planned. On the other hand, Astronaut 2's EVA (Figure 3.12) is well within all of the constraints, and is feasible.

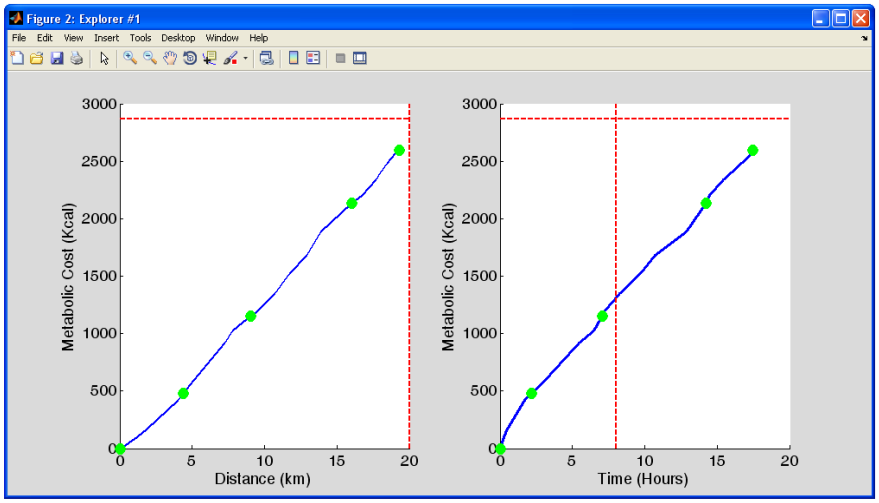


Figure 3.11. Display showing metabolic cost of Astronaut 1's EVA with respect to distance and time

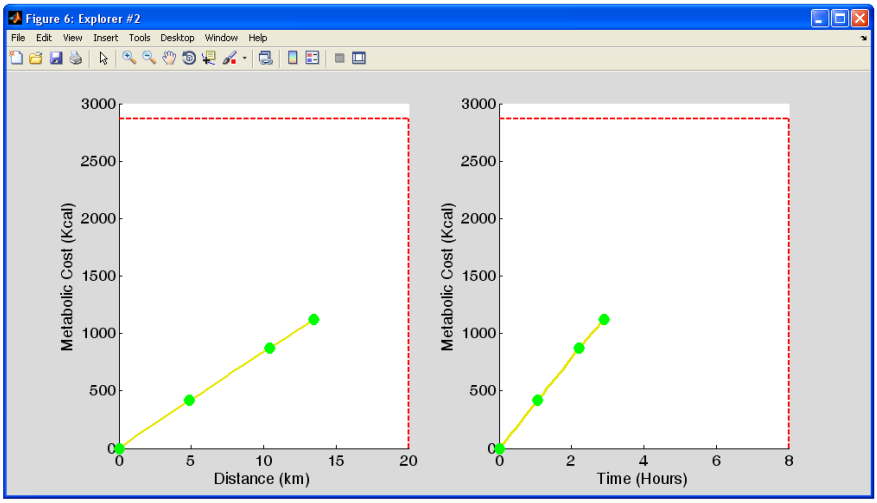


Figure 3.12. Display showing metabolic cost of Astronaut 2's EVA with respect to distance and time

SEXTANT also has the ability to compute shadowing-related metrics for the explorers. These are different for astronauts and transportation rovers, and are the outputs of either the astronaut thermal model or the rover power model. For astronauts, the following additional metrics are computed for each Path Point:

4. Cumulative mass of water needed to replenish ice sublimated away for cooling during the traverse from home to the Path Point

5. Cumulative heater energy required for heating during the traverse from home to the Path Point

Instead of these, transportation rovers have only one additional metric, once again computed for each Path Point:

4. Battery energy level at the current Path Point

To calculate these metrics, the user clicks the *Determine Shadows* button at the top of the mapping interface. The user is then prompted to enter how often the sun position is calculated. A smaller value will give a more accurate result, but increase the computational time. If the value of the time step is more than twice the travel time for the traverse, then the sun will remain in the same position for the entire traverse. After this information is calculated, it is displayed graphically with respect to the traverse distance or time. Figure 3.13 shows the mass of sublimated water and heater energy required for Astronaut 1 to traverse the EVA shown in Figure 3.11.

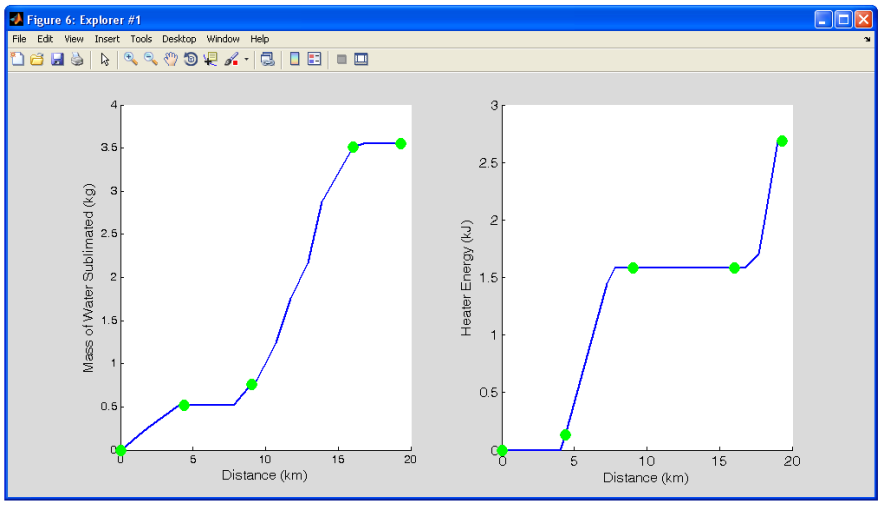


Figure 3.13. Display showing mass of sublimated water (left) and heater power (right) of Astronaut 1's EVA with respect to distance

A graphic of the traverse is also displayed in Figure 3.14, showing which areas are in sunlight, and which are in shadow. In this figure, the white areas are where the solar disk is completely visible (full sunlight), the black areas are where the solar disk is completely hidden (full shadow), and the grey areas are where the solar disk is only partially visible.

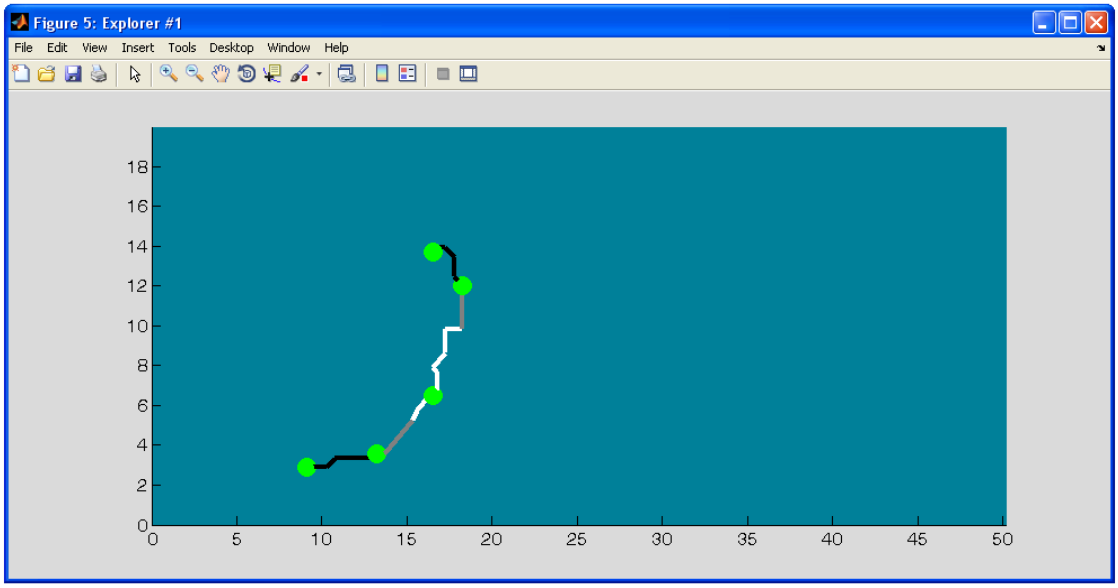


Figure 3.14. Display showing shadowing along Astronaut 1's EVA

Another important function of SEXTANT is the calculation of *return-home paths*, which represent emergency "walk-back" situations. In the mapping interface, the user right-clicks on any point along a path (not necessarily a Path Point) to determine the most efficient path back to home from this point. This path is then displayed with a dashed line, and the corresponding path length, time of travel, metabolic cost, shadowing, and thermal or power metrics are displayed. Figure 3.15 shows a return-home path for Astronaut 2.

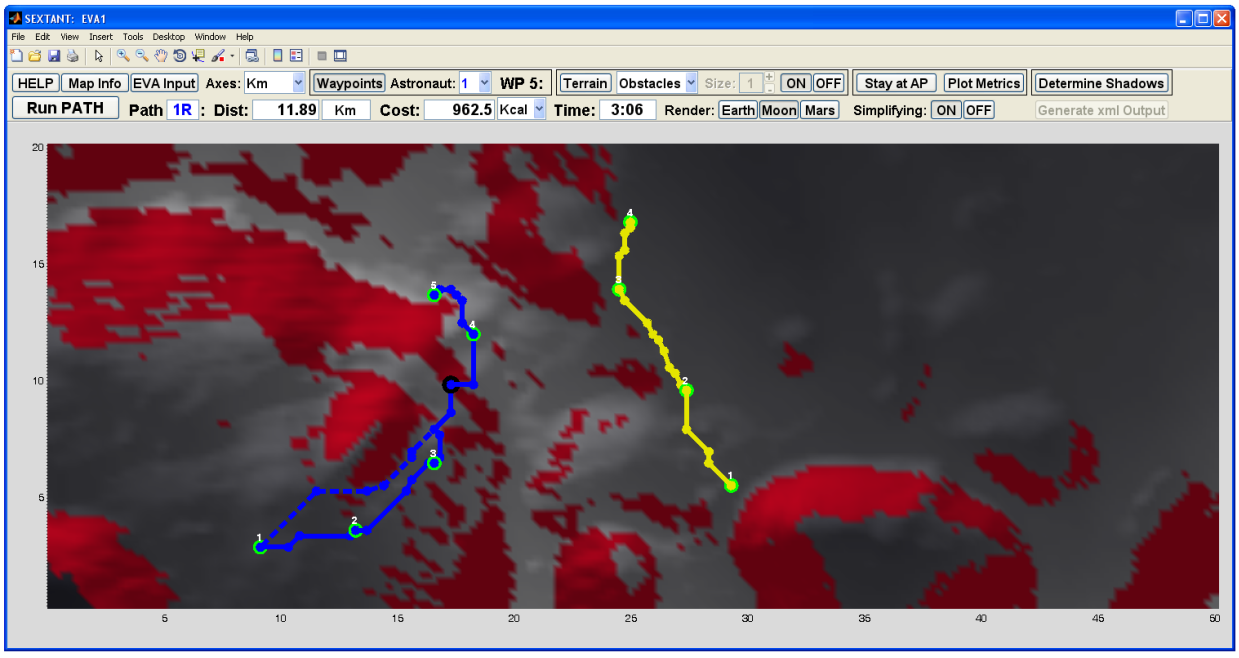


Figure 3.15. Return-home path for Astronaut 1

This function is very important during traverse emergency situations and contingency planning. Explorers only have a limited supply of consumables and energy, which imposes a strict limit on the total EVA length, time, and difficulty. Throughout the traverse, SEXTANT gives the user the ability to calculate the return-home paths for future points along the traverse in real time. Knowing the length, time, energy cost, and shadowing-related metrics of these return-home paths will allow the user to determine the "point of no return" where that astronaut has just enough consumables left to make it back home (with margin). This maximizes the productivity of the EVA while ensuring the astronaut's safety.

3.4 Real-time Navigation

The framework for an implementable, user-centric SEXTANT interface has been laid by integrating SEXTANT with the Individual Mobile Agents System (iMAS), developed at NASA Ames Research Center (Rupert and Boston 2006, Garry 2006, RIACS News 2008, Bleacher et al. 2009). This combined system gives the astronaut user an auditory interface through which he can input, store, and retrieve information about the EVA. In the future, this auditory interface could be combined with a visual interface to serve as a tool that astronauts could use for real-time planning and re-planning while on the surface of the Moon or Mars.

3.4.1 Individual Mobile Agents System Background

The Individual Mobile Agents System (iMAS) is an offshoot of the Mobile Agents Architecture (MAA) developed by NASA Ames in 2001 (Clancey et al. 2003, 2004, 2005, 2006, 2007). MAA, and subsequently iMAS, were designed to integrate the numerous components required during future planetary surface exploration. Multiple suited astronauts, robotic explorers, and transportation rovers will all be used in concert, traveling about and gathering information aided by a number of tools. MMA was created with the goal of bridging the gap between the human explorers and the other exploration systems and equipment on the surface. Each separate system was represented by an "agent" written in the Brahms multi-agent modeling environment. Brahms was originally developed in 1992 by Sierhuis at NASA Ames to simulate work practice, which is the way people perform daily activities in the workplace (Sierhuis et al. 2007). More recently, Brahms has been used to simulate how people and systems interact within an environment. This is the context in which Brahms has been used for both MMA and iMAS. Components of the exploration system are represented by Brahms "agents", which are "software programs that

enable people to receive data from and send commands to automation systems in the world" (Clancey et al. 2006, p. 2). These agents are all modeled with a similar structure. Agents store data about the world as individual *Beliefs*, and react to a model of the environment (*Facts* about *Objects*) that is shared by all agents. Each agent has conditional actions called *Workframes*, inference rules called *Thoughtframes*, and chronological behaviors called *Actions* that help plan its tasks. Finally, agents can communicate with each other by *Asking* about and *Telling* beliefs, and with external hardware via special *Communication Agents*.

iMAS, developed in 2005, has the same function as MAA – helping the astronaut to keep track of the exploration systems and equipment during a traverse. However, it is a more-focused tool that is concerned only with an individual astronaut and his particular traverse. Whereas MAA was a networked system of multiple computers, iMAS runs on one computer carried by the astronaut. The astronaut interacts with iMAS through a speech dialog interface. There is no push-to-talk button; rather, iMAS is constantly listening to the user and responds to recognized commands. There are a wide variety of commands that iMAS can accept. A full list of examples can be seen in Appendix B (Dowding 2009). In all cases, iMAS provides a rising tone to signal that a command has been recognized and accepted.

The capabilities of iMAS fall under four main categories:

1. Activity Tracking
2. Navigation
3. Science Data Logging
4. Biomedical Monitoring

3.4.1.1 Activity Tracking

iMAS has the ability to keep track of an ordered list of activities planned before the traverse. Each activity is modeled by a location and duration, and can be mobile (i.e., traveling from one location to another) or stationary (i.e., taking a regolith sample). The astronaut can start any activity by giving iMAS a verbal command such as "Start my first / second / next activity" or "Start activity: Egress." iMAS keeps track of the time spent on an activity, and issues an auditory alert once the astronaut approaches the end of the scheduled activity duration. If desired, the astronaut can verbally extend the duration of the activity by any amount.

3.4.1.2 Navigation

iMAS can keep track of multiple named locations, which may or may not be associated with activities. The locations can be pre-loaded into iMAS, or created by the astronaut during the traverse by saying, “Call this location Work Site 2”, for example. This location is then created based on the user’s current position on the planetary surface. As the astronaut moves across the terrain, iMAS tracks his position at a specified interval. At any point, the astronaut can ask iMAS, “Where am I?” to receive the distance and heading to the starting point of the traverse – the habitat. iMAS can also give the distance and heading to any named location with respect to the explorer’s current position by answering a command like, “Where is Work Site 2” with a phrase such as, “Work Site 2 is 20 meters, 45° to the left.”

3.4.1.3 Science Data Logging

iMAS logs and stores science data in many different forms. First of all, the astronaut can record voice notes at any time to capture information that is to be reviewed later. The astronaut simply says, “Record a voice note,” and then immediately begins talking. When he is done speaking, the voice note automatically ends. Recording a voice note is faster and easier than writing a physical note, especially with space suit gloves on. The astronaut can also create “sample bags”, which are virtual collections of all the samples gathered at a site. Multiple associations can be made between voice notes, sample bags, the astronaut’s current location, and photographs taken during the traverse. The user simply says, “Associate image one with sample bag one,” for example This is beneficial, as the astronaut can collect samples at a site, take photographs of them, record a voice note describing the overall environment from which they came, and associate them all together along with the location where they were found.

3.4.1.4 Biomedical Monitoring

iMAS monitors the astronaut’s biomedical parameters by connections to the Life Support, Exploration Guidance Algorithm and Consumable Interrogator (LEGACI) developed by Kuznetz at Johnson Space Center (2008). LEGACI uses a collection of biosensors located within the space suit to collect information about the EVA status, consumables, and performance data. iMAS allows the astronaut to access this information in real time. The astronaut can query iMAS about the amount of remaining consumables, which are the available oxygen, feed water, power, and carbon dioxide scrubber capability. iMAS can also report many metabolic parameters to the

astronaut when requested, including the astronaut's metabolic rate, heart rate, number of calories burned, and sweat rate. Example commands are, "Consumable usage." or "What is my met rate?" The space suit status is also available, in terms of the oxygen tank pressure, suit pressure, and an estimate of the heat leak from the space suit to the environment. Lastly, iMAS can perform a "walkback check", where it determines the velocity the astronaut must travel to make it back to the habitat with the remaining amount of consumables. It tells the astronaut this, along with his recent average speed. In the future, the metabolic rate measured by LEGACI could be used to update SEXTANT's equations for the astronaut's energy consumption in real-time. This is discussed in detail in Section 6.3.4.

3.4.2 Demonstrating the Capabilities of iMAS in the Field

iMAS has been used in many different field tests since its inception. It was first tested in January 2006 by Crew 41 at the Mars Society's Mars Desert Research Station (MDRS) in Utah. Many difficulties were encountered, including difficulty connecting iMAS with the GPS receiver and camera, and problems with the speech recognition software (Rupert and Boston 2006). iMAS was tested once again at MDRS in May 2006 by Crew 49, whose main mission was to install MAA in the MDRS habitat to monitor the power systems (Garry 2006). iMAS functioned much better during these simulated EVAs, where it was used to successfully to collect images and voice notes from a user in the field. This information was then downloaded once the user returned to the habitat at the end of the EVA. These field tests demonstrated the usefulness of iMAS in an operational setting. Most recently, in August 2008, iMAS was used in a series of simulated EVAs performed on the Kilauea volcanic shield in Hawai'i (RIACS News 2008, Bleacher et al. 2009). The simulated EVAs were undertaken by geologists from the Smithsonian Institution and Goddard Space Flight Center, and attempted to replicate the second EVA of the Apollo 12 mission in terms of its layout and observations. The performance of iMAS on the EVA was then compared to the use of a traditional geologist's notebook. It was found that iMAS offered advantages in gathering information and associating it with its original location. Many recommendations were also made to help improve iMAS in the future.

3.4.3 Integrated SEXTANT-iMAS System

Within the integrated system, each component has a set of tasks to perform:

SEXTANT	iMAS
• Spatial planning of path Activity Points • Calculation of the most-efficient traverse path	• Tracking the explorer's progress • Giving the heading to a requested Activity or Path Point

3.4.3.1 Tasks of SEXTANT in the Integrated System

The SEXTANT interface is used to specify Activity Point locations spatially on a terrain map of the planetary surface. This is a vast improvement over the method in which traverse objectives are specified in iMAS. iMAS uses a database program called Compendium to organize and store all of the traverse information. Compendium's interface can be seen in Figure 3.16. Prior to the integration with SEXTANT, an iMAS user had to plan the traverse in Compendium by creating *nodes* for each objective and specifying their type, name, location (in terms of latitude and longitude) and other information. The locations must then be linked together with *activities*, which can either be a *movement activity* from one location to another, or a *stationary activity* at a single location. This method of traverse planning is quite non-intuitive, because it does not map well to the actual terrain. Coordinates of latitude and longitude are only numbers, unless linked to a specific point on a map. Furthermore, having to modify the traverse through Compendium directly makes it difficult for users to quickly plan and re-plan paths.

Users can spatially plan the traverse as described in Section 3.3.3, taking full advantage of SEXTANT's capabilities. Activity Points can be placed and modified on the 3D mapping interface, and the path can be optimized for distance, time, or energy cost. Once the traverse is deemed satisfactory, it can be imported into iMAS as an .xml file that is read by Compendium. This file creates a location node for each Path Point and movement activities between each set of consecutive Path Points. These movement activities also specify the duration of each segment, as computed by SEXTANT. This process also creates a file called *gps.dat* with a list of latitude and longitude coordinates along the traverse at a user-specified time interval. If iMAS has no connection to a real GPS receiver, it can read the astronaut's position in time from this file. This allows for the user to simulate an astronaut following the path planned in SEXTANT – a "virtual field test."

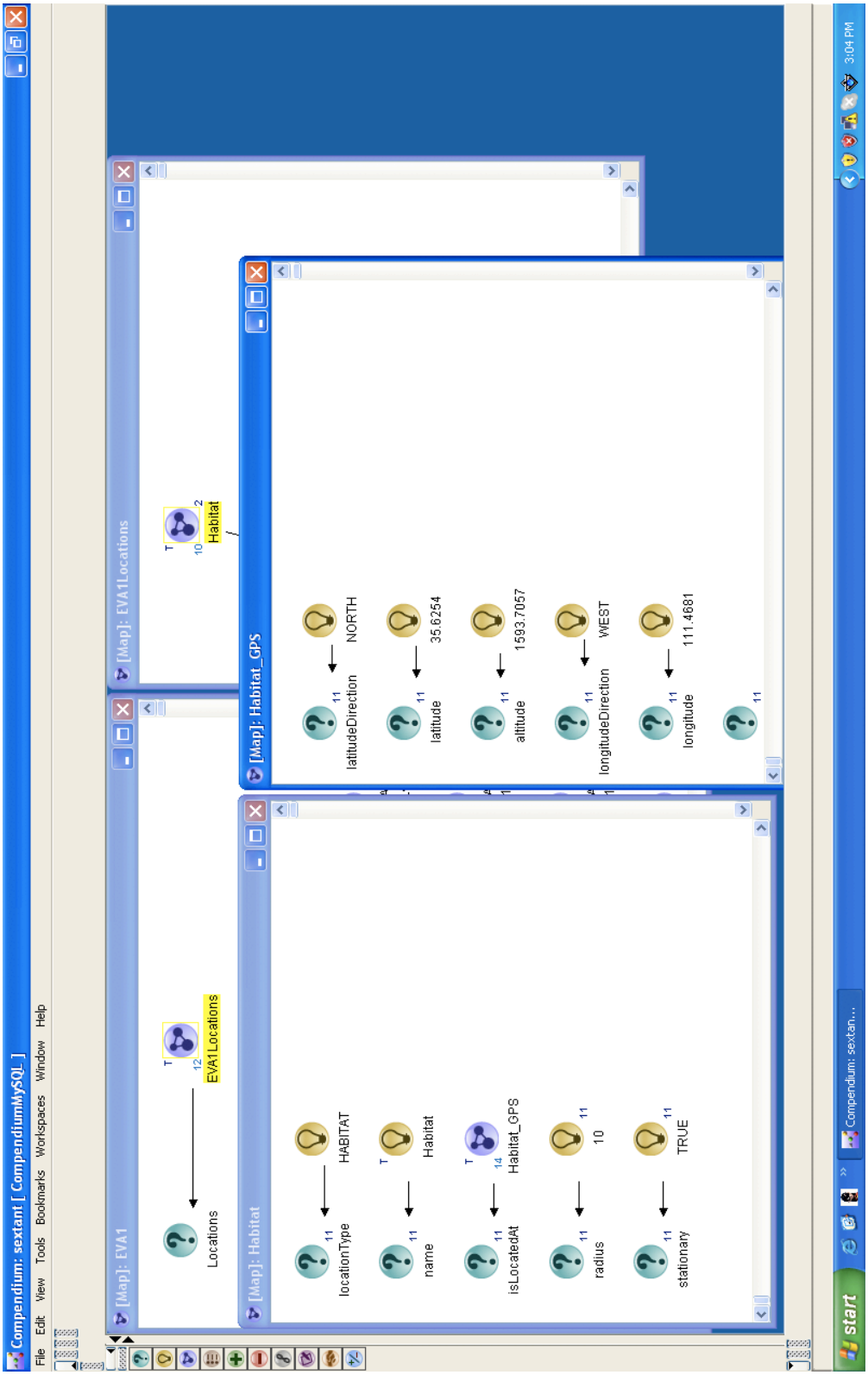


Figure 3.16. Interface of the Compendium database used by iMAS

3.4.3.2 Tasks of iMAS in the Integrated System

Once the traverse has been imported into iMAS, the explorer can be tracked as he follows the planned path. This is accomplished through a connection to a GPS receiver. In lieu of an actual GPS signal, iMAS can read the astronaut's position off the *gps.dat* file produced by SEXTANT. At any point along the traverse, the astronaut can ask iMAS for the heading and direction to a stored Path Points by saying, "Where is Path Point 2", for example. The heading to the requested Path Point is then given relative to the astronaut's current heading. This is an improvement upon earlier versions of iMAS. Previously, the heading to any requested location was given absolutely, meaning with respect to due North (i.e., iMAS would say, "Path Point 2 is 30 meters south-south west."). This was found to be confusing and non-intuitive in the field, as it required the simulated astronauts to constantly know where North was. Now, in tracking the explorer, iMAS stores an ordered set of the most-recent locations. When the location of a Path Point is requested, iMAS determines the explorer's heading with respect to North by drawing a vector between the current position and the previous position. Figure 3.17 shows this heading, along with the others.

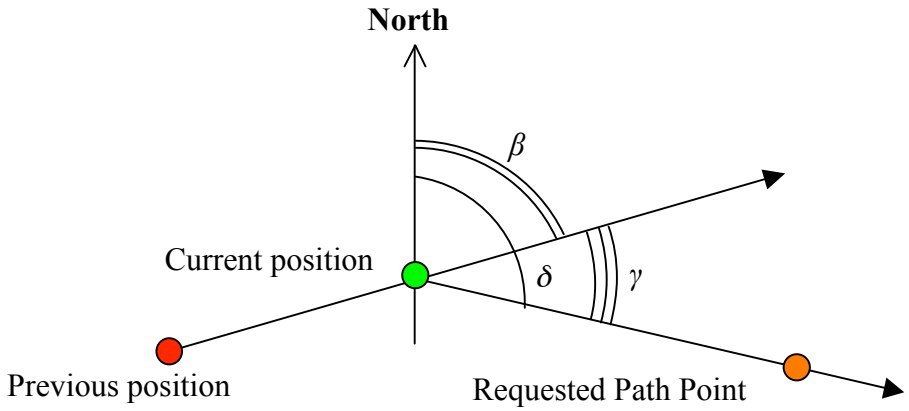


Figure 3.17. Determination of relative heading to requested path point

The explorer's absolute heading (β) is subtracted from the absolute heading to the Path Point (δ), to give the relative heading to the Path Point with respect to the astronaut's current direction (γ). So, iMAS now describes the location of a Path Point by saying, "Path Point 2 is 30 meters, 30 degrees to the right."

There are some limitations inherent to this method of computing the astronaut's heading. It assumes that the explorer travels in a straight line between consecutive locations. It also assumes that the explorer doesn't ever turn around in place. Both of these approximations

simplify the problem, but give a less accurate answer for the heading to the requested path point. Ideally, iMAS would contain a compass that could accurately determine the direction an astronaut was facing. For lunar exploration, this compass cannot be a magnetic one, due to the Moon's weak magnetic field. Star- or sun-trackers, in conjunction with IMUs, may have the ability to accurate determine an astronaut's heading on the lunar surface.

3.4.3.3 Flow of Information in the Integrated System

Figure 3.18 shows how information in the integrated system travels between the user, SEXTANT, a GPS receiver (if present), and iMAS.

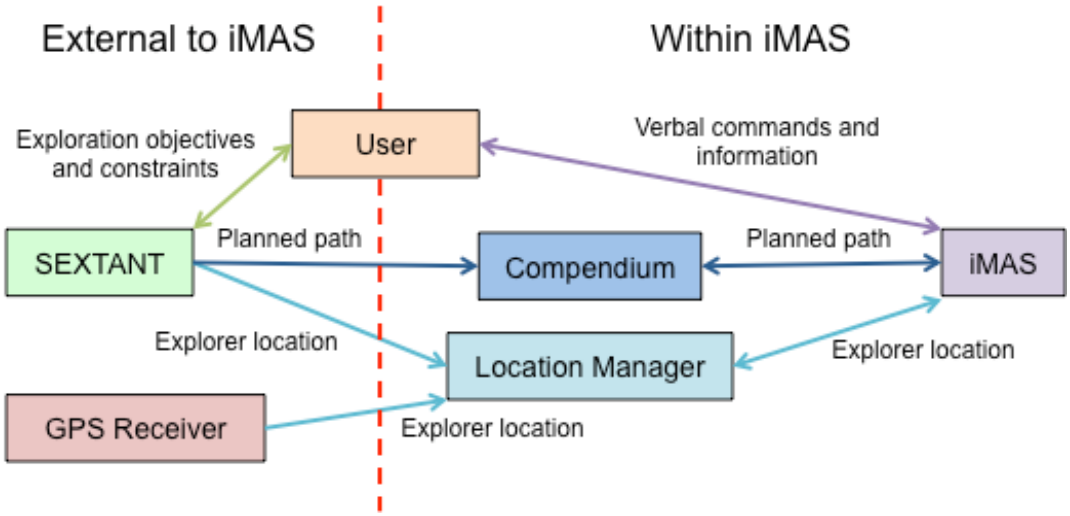


Figure 3.18. Flow of information within integrated SEXTANT-iMAS system

The left half of Figure 3.18 shows the two components that are external to iMAS – SEXTANT and a GPS receiver. The right half of the figure shows the components within iMAS. The user is between the two, signifying that he interacts with both SEXTANT and iMAS. Figure 3.18 shows that while iMAS communicates with Compendium and the Location Manager during a traverse, the interaction between SEXTANT and iMAS is one-way. Information cannot be transferred from iMAS (Compendium or the Location Manager) back to SEXTANT. If the user wishes to plan a new traverse, he must do so in SEXTANT and re-import the plan into iMAS. The traverse in SEXTANT cannot change in real-time based on the information gathered from the user through iMAS. This is something to be improved upon in the future, as discussed in Sections 6.3.1 and 6.3.4.

3.5 Conclusion

This chapter discusses how SEXTANT comprehensively represents a lunar or planetary traverse with three components: the Exploration Objectives, the Environmental Model, and the Explorer Model. This thesis has modified the Environment Model in SEXTANT to include lunar terrain and the location of sunlit and shadowed areas along a traverse. These are very important, as they allow SEXTANT to simulate actual traverses that will occur during future explorations of the Moon. This thesis has also expanded the Explorer Model by adding a transportation rover, which will certainly be used by future astronauts. This further increases the fidelity of the SEXTANT traverse representation. This chapter also presents the SEXTANT user interface, where the user can specify details of the traverse representation, place Activity Points, and view the distance, time, energy cost, thermal, and power metrics of the planned path. Beyond this visual interface, SEXTANT has been integrated with iMAS from NASA Ames to provide a speech dialog interface through which the user can receive real-time guidance and navigation along a traverse. This is a contribution of this thesis, and a first step in creating a SEXTANT interface that it optimized for the astronaut user.

This page intentionally left blank.

Exploring the lunar south pole,
Were astronauts out for a stroll.
Before it had begun,
They'd checked up on the sun
So they'd have enough thermal control.

4 Determination of Shadowing on the Lunar Surface and Computation of Astronaut Thermal Load and Transportation Rover Power Consumption and Generation

As was stated in Section 1.2.4, the major contribution of this thesis is the determination of shadowed areas across a planned traverse, which allows SEXTANT to compute the thermal loading on astronauts and the power consumption and generation of transportation rovers. These metrics directly relate to the usage of consumables during a traverse, which is the main limiting factor for its duration. The determination of sunlit and shadowed areas has been accomplished in collaboration with the MIT Department of Earth, Atmospheric, and Planetary Sciences (EAPS). The lunar elevation map within SEXTANT is constructed from data from the Lunar Orbiter Laser Altimeter (LOLA) instrument, for which EAPS Department Chair Maria Zuber is the Deputy Principle Investigator. Furthermore, the calculation of shadowing relies on techniques developed by Erwan Mazarico, a NASA post-doc working at Goddard Space Flight Center.

4.1 The Lunar Reconnaissance Orbiter

The Lunar Reconnaissance Orbiter (LRO), shown as an artist's rendition in Figure 4.1, was launched on June 18, 2009, the culmination of five years of development ("Lunar Reconnaissance Orbiter" 2010, "LPRP Overview & History" 2010).

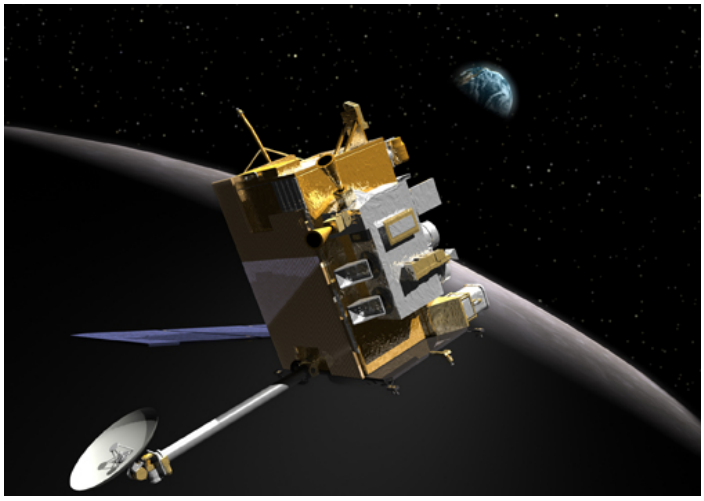


Figure 4.1. Artist's rendition of LRO in orbit around the moon ("Lunar Reconnaissance Orbiter" 2010)

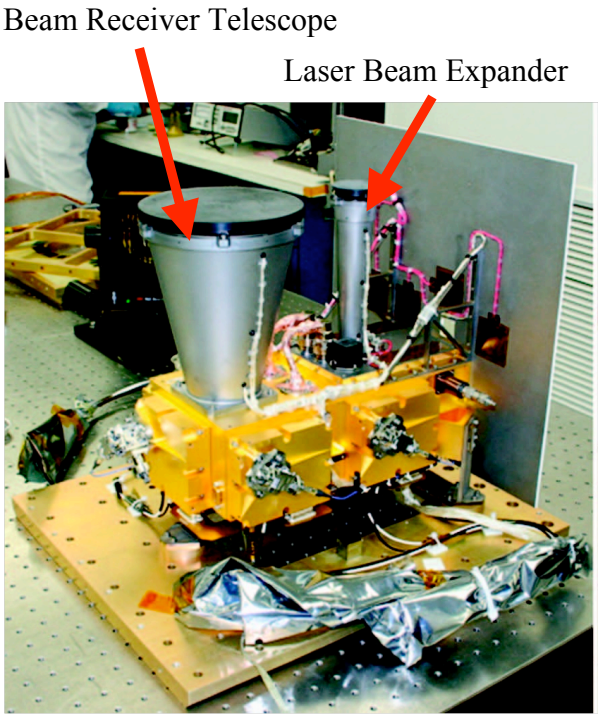
Managed by the NASA Goddard Space Flight Center, the goal of LRO's one-year primary mission is to support future human exploration of the Moon. Specifically, LRO's exploratory objectives are to find safe landing sites, locate potential resources, characterize the radiation environment, and demonstrate new technology ("Lunar Reconnaissance Orbiter" 2010). LRO is accomplishing these tasks with a suite of six instruments:

- Cosmic Ray Telescope for the Effects of Radiation (CRaTER)
- Diviner Lunar Radiometer Experiment (DLRE)
- Lyman Alpha Mapping Project (LAMP)
- Lunar Exploration Neutron Detector (LEND)
- Lunar Reconnaissance Orbiter Camera (LROC)
- Lunar Orbiter Laser Altimeter (LOLA)

While all of these instruments support the explorative goals of LRO, they also have great scientific capabilities. These will be leveraged during the extended mission of LRO, which is to start in the summer of 2010. Goals for this phase will be to investigate the variability of the lunar atmosphere (LAMP) and characterize cold traps on the lunar surface (DLRE), among others (Gladstone et al. 2010, Paige et al. 2010).

4.1.1 The Lunar Orbiter Laser Altimeter

The instrument of most importance to SEXTANT is the Lunar Orbiter Laser Altimeter (LOLA) (Smith et al. 2006, Ramos-Izquierdo et al. 2008, "LOLA Fact Sheet" 2009, Smith et al. 2010a, Smith et al. 2010b). LOLA has two main objectives, both designed to support LRO's overall goals of assisting future human exploration (Smith et al. 2010a). Firstly, LOLA is measuring the location, direction, and magnitude of surface slopes, as well as the elevation variation (also called the surface roughness). Secondly, LOLA is quantifying the reflectance of the lunar surface (the albedo) to look for the presence of water ice crystals. LOLA is accomplishing these goals by using a 1064 nm-wavelength Nd:YAG laser and detector to measure the distance from LRO's 50-km polar orbit to the lunar surface. Figure 4.2 shows a flight model of the LOLA instrument, with the laser beam expander and the beam receiver telescope labeled. The laser beam is split into 5 separate channels that form an "X" pattern on the lunar surface. This is shown in Figure 4.3.



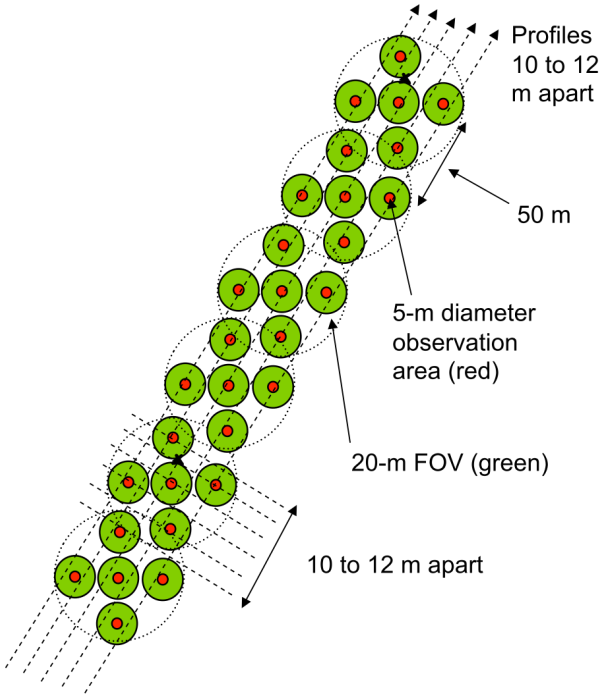


Figure 4.2 (left). LOLA flight model (Ramos-Izquierdo et al. 2008)

Figure 4.3 (right). Ground track of LOLA (Smith et al. 2010a)

Each channel provides a measurement of altitude, surface roughness, and surface brightness. The width of each LOLA ground track is between 50 and 60 m. After the primary mission is complete, LOLA will have produced a complete map of the lunar surface with a resolution of 25 m along the Moon's latitude, and a resolution along the Moon's longitude anywhere from 25 m at the poles to 1200 m (1.2 km) at the equator. As with the entire LRO spacecraft, LOLA offers great opportunities for scientific investigations during and after the completion of the primary mission. These include helping to characterize impact craters, volatiles in permanently-shadowed craters at the lunar poles, volcanic areas, and the lunar regolith (Smith et al. 2010a).

4.2 Determining Shadowing on the LOLA-Derived Elevation Map

SEXTANT gives users the ability to plan paths over any inputted lunar elevation map. Currently, SEXTANT employs a LOLA-produced elevation map of the lunar south pole. Elevation data for the lunar north pole is also available from LOLA, but has not yet been used in SEXTANT. The south pole map is a 500 km by 500 km gnomonic projection of the area south of a lunar latitude

of 80° S. A gnomonic projection is one where all great circles on the globe are represented as straight lines on the map ("Gnomonic" 2006). Therefore, a line between two points on the map corresponds to the actual shortest distance between these two points. The lunar elevation map used in SEXTANT features a regularly-spaced horizontal grid of points with a resolution of 240 m. This resolution is not as good as is truly desired, as some large and important objects like boulders will undoubtedly be missed. However, this map is only a preliminary elevation map from LOLA, and future data products will have much higher resolution. The flexibility of SEXTANT allows for any matrix of elevations to be inputted, no matter what the resolution. Figure 4.4 shows this entire elevation map as seen in SEXTANT, and Figure 4.5 gives the names of prominent craters.

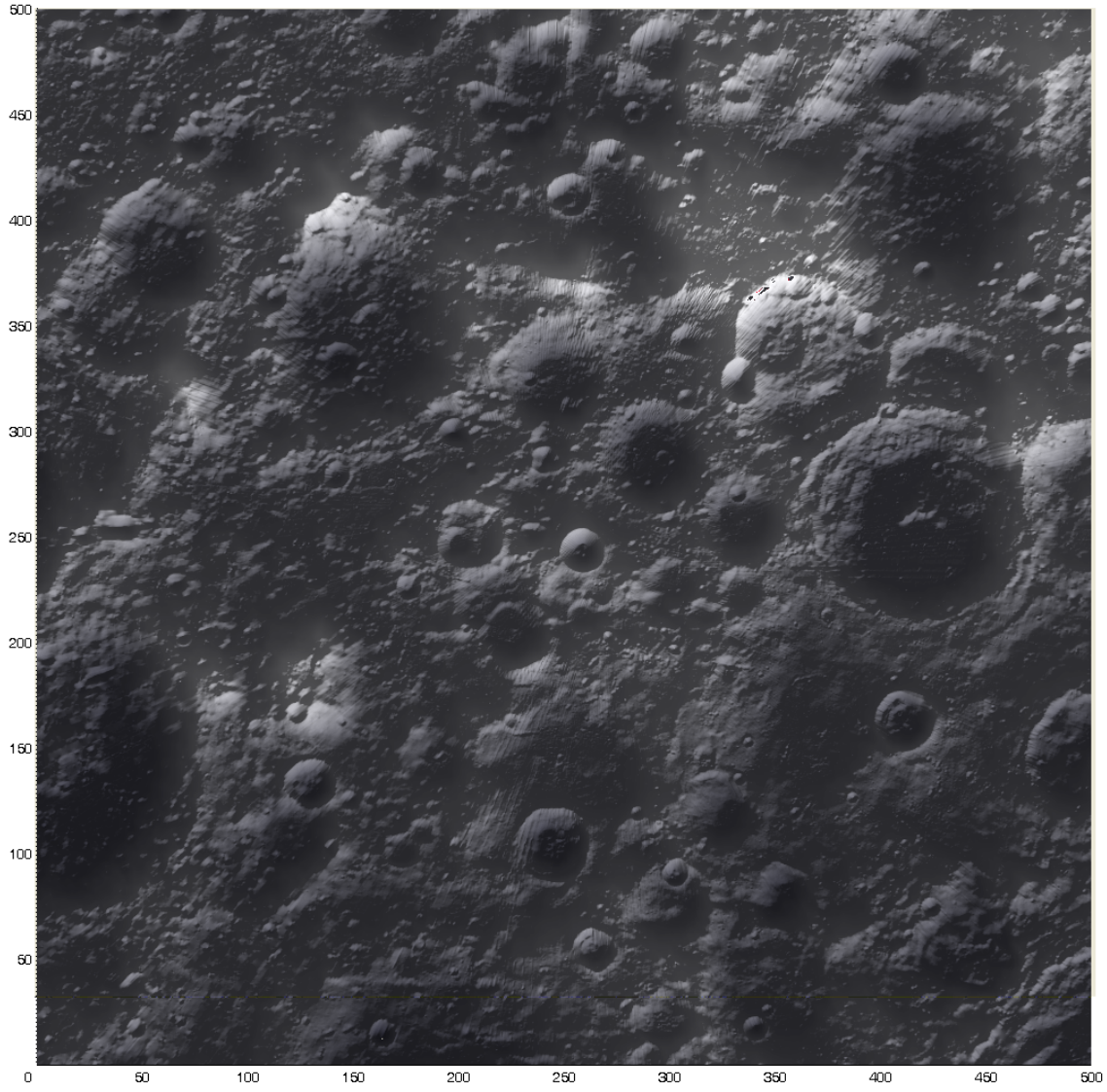


Figure 4.4. Elevation map of lunar south polar region

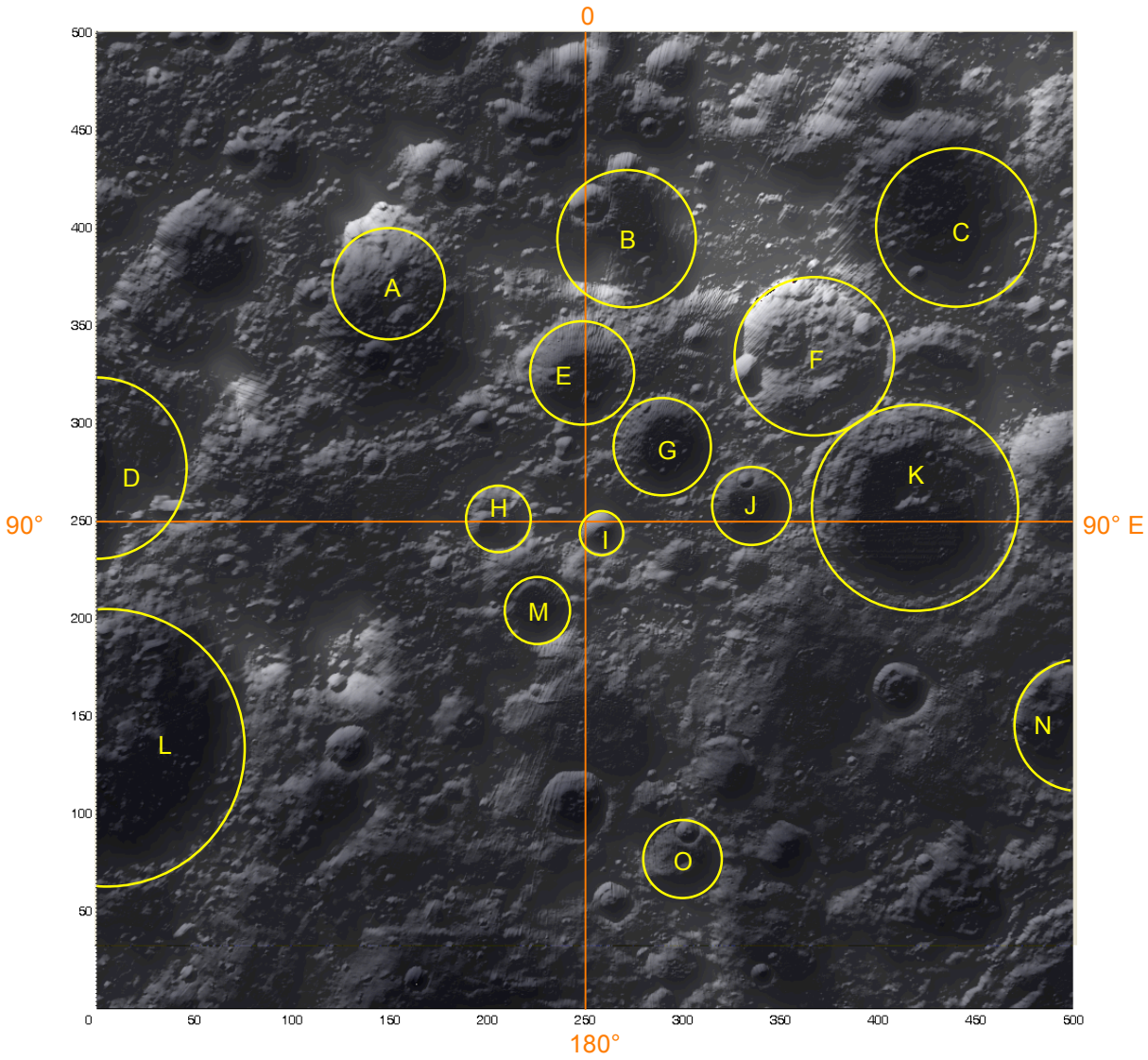


Figure 4.5. Names of prominent craters in lunar south polar region (Bussey and Spudis 2004, Byrne 2005)

A. Cabeus
B. Malapert
C. Scott
D. Drygalski
E. Haworth
F. Nobile
G. Shoemaker
H. De Gerlache
I. Shackleton
J. Faustini
K. Amundsen
L. Ashbrook
M. Sverdrup
N. Idelson
O. Wiechert

The elevation map is stored in a MATLAB file (extension .mat), which is loaded every time SEXTANT is initialized. SEXTANT gives the user the ability to select a portion of the terrain by specifying the limits of this new submap in kilometers from the center. This is explained in Section 3.3.1. Selecting a portion of the overall map is beneficial for shorter traverses, as it involves less computational time. Figure 4.6 shows the entire lunar south polar region, with a submap outlined in yellow. Figure 4.7 shows this submap as selected in SEXTANT.

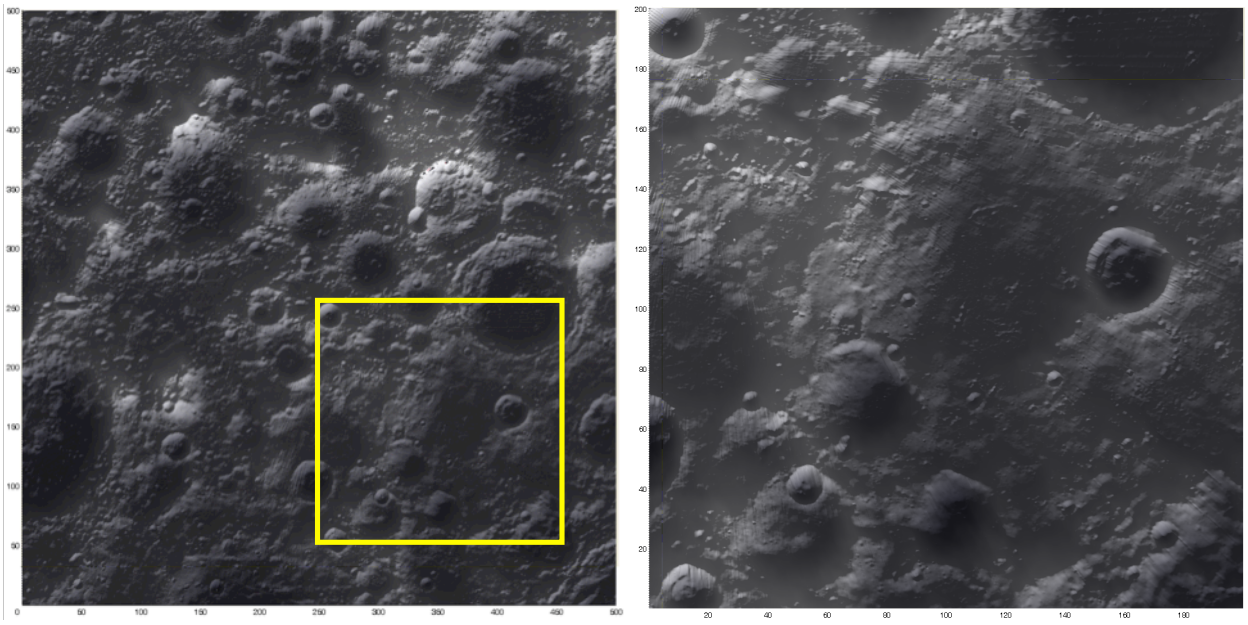


Figure 4.6 (left). Lunar elevation map showing location of selected submap

Figure 4.7 (right). Selected submap of the lunar surface, as seen in SEXTANT

Once a traverse has been planned across the lunar surface, SEXTANT parses the traverse into a series of stages between consecutive Path Points and computes the astronaut thermal load or rover power generation and consumption over each individual stage. In order to do this, the shadowing must be determined at each Path Point. The user begins this process by clicking the *Determine Shadows* button at the top of the SEXTANT 3D mapping interface. As stated in Section 3.3.3, the user is asked to provide the frequency with which the sun position is calculated. The time at which the explorer arrives at each Path Point is then rounded to the closest multiple of this time step. This "binning" gives a series of unique sun times at which the sun position must be calculated. Depending on the magnitude of the time step, it is possible that the sun is in the same position for some consecutive Path Points.

The position of the sun is calculated at each of these times with respect to the Moon's center of mass through the use of the NASA Jet Propulsion Lab SPICE toolkit for MATLAB (called MICE) (Acton 2009). SPICE is a program that gives ephemeris data for any target body in the solar system. Loosely, ephemeris data is defined as the position of a target in the sky as seen from another body, as a function of time. MICE is a version of SPICE that is fully contained in MATLAB; no outside connections are required. Because SPICE returns the sun position with respect to the Moon's center of mass, SEXTANT performs coordinate transformations to get the elevation and direction of the sun from each Path Point at the correct time.

Once the sun position is known for all Path Points, SEXTANT must determine whether or not it is visible at each particular time. This is accomplished using the *horizon method*, used for the Moon by Garrick-Bethell et al. with ground-based radar data (2005) and more recently by Mazarico et al. with the LOLA data (2009, 2010). Mazarico et al. distinguish two distinct steps in this method. First, a database is created of 720 separate horizon maps that each shows the elevation of the horizon in a certain direction for all points on the terrain map. Each of these horizon maps is in a direction that varies 0.5° from the surrounding maps. This step is very time consuming – on the order of a week using fifteen MacPro computers – but must only be performed once. The second step in the horizon method requires referencing these horizon elevation maps to determine whether or not the sun is visible in a certain direction for a Path Point. If the sun direction is not an exact interval of 0.5°, then linear interpolation is used between the adjacent horizon elevation maps. Because the lunar surface elevation map is in gnomonic projection, the sun is in the same direction for each point on the map at one particular time. So, only one horizon elevation map must be referenced for the entire set of Path Points at the same unique sun position time. Once the horizon elevation is calculated for each Path Point, this is compared to the elevation of the sun. As seen from the surface of the Moon, the solar disk has an apparent diameter of 32 arcminutes (0.52°). Because the sun is not a point source, there are three possible conditions for the shadowing at each Path Point:

1. The sun is fully visible (complete sunlight)
2. The sun is partially visible (partial shadow)
3. The sun is not visible at all (complete shadow)

To account for all conditions, SEXTANT computes the percentage of the solar disk visible as a value from 0 to 1 and stores this for each Path Point.

The shadowing for each path stage is determined as the average of the sun visibility percentages for the two Path Points at either end. This information can be used to compute the thermal load on astronauts and the power consumption and generation of transportation rovers for each path stage, and for the entire traverse together.

4.3 Astronaut Thermal Model

Knowing the thermal load on an astronaut throughout an EVA is important because it directly relates to the amount of consumables required for the traverse. For astronauts, the two main thermal consumables of concern are the mass of water required to produce ice for cooling by sublimation and the amount of energy required to run the heater. This heater energy requirement physically manifests itself as the capability of an astronaut's power system. The mass of water for sublimation and power system that can be carried by an astronaut are limited by the volume and mass constraints of the space suit's life support system. As such, it is important to be able to accurately estimate the amount of these consumables that the astronaut will require for a traverse. This ensures that the astronaut will have enough thermal control to complete the EVA and safely return to the habitat.

Having a good thermal model for future EVAs is especially important because of the shadowing conditions at the lunar poles. At the lunar poles, the sun reaches a maximum altitude of 1.5°, meaning that there are always large and long shadows during the day. Furthermore, the stay on the surface will last through both lunar days and nights. For these reasons, it will be difficult to plan EVAs that only traverse sunlit areas. The space suit will require both heating and cooling capabilities. This contrasts EVAs during the Apollo program, where the astronauts were only on the surface during the lunar day. The sun was also higher in the sky, meaning that shadows were shorter and they could be more easily avoided. For these reasons, the Apollo space suits only required a cooling system. Heating was later incorporated into the gloves of the space shuttle extravehicular mobility unit (EMU) to prevent astronauts' fingers from freezing during EVA. Thermal control for a future space suit will be more similar to this than the Apollo space suit, but will require heating for the astronaut's entire body.

The astronaut thermal model in SEXTANT has two components: the first details the heat flux into or out of the space suit, and the second simulates a thermal control system that removes heat from or adds heat to the space suit.

4.3.1 Heat Flux Into and Out of the Space Suit

In the SEXTANT astronaut thermal model, there are five sources of heat transfer to and from the space suit. Table 4.1 details the five sources and their characteristics.

Table 4.1. Sources of heat transfer with the astronaut's space suit

Source of Heat	Internal or External to the Suit	Direction with Respect to the Suit
• Heat from inefficiency of astronaut work	Internal	Into
• Space suit electronics waste heat	Internal	Into
• Direct sun radiation	External	Into
• Radiation between the space suit and the lunar surface	External	Into & Out of
• Radiation between the space suit and deep space	External	Into

The three external heat sources reach the astronaut by conduction through the space suit. The amount of conduction is dictated by the temperature of the outer surface of the space suit, which is a function of the environment. The internal heat sources are added to this external heat to get the total heat flux into or out of the space suit. The sign convention used for this heat flux is positive for heat entering the space suit, and negative for heat leaving the space suit.

4.3.1.1 Heat Transfer from Internal Sources

The metabolic cost of a traverse, as detailed by the equations in Section 3.1.3.1, is the amount of energy consumed by the astronaut when carrying out an EVA. This energy is used in a number of different processes by the astronaut's body. A portion is the basal metabolism, which is used for basic physiological processes like breathing, pumping blood, and digestion (Carr and Newman 2007b). An additional portion is used to do useful work in moving the astronaut across the terrain. The remainder of the energy consumed is transformed into heat, which is both stored in the body and released to the environment.

Equation (4.1) details these components that make up the total energy consumption ($\dot{Q}_m$) (Carr and Newman 2007b).

$$\dot{Q}_m = \dot{W}_w + \dot{W}_{wc} + \dot{W}_{wr} + \dot{W}_{ws} + \dot{Q}_n + \dot{Q}_s \tag{4.1}$$

In Equation (4.1), $\dot{W}_w$ is the external mechanical work, $\dot{W}_{wc}$ is the work done by the ground counterforce, $\dot{W}_{wr}$ is the work done by moving the body's limbs, $\dot{W}_{ws}$ is the work required to move the space suit, $\dot{Q}_n$ is the heat loss to the space suit, and $\dot{Q}_s$ is the heat storage by the body. All of these components have units of Watts. Basal metabolism is absent, because these equations deal with work, and not energy.

A few assumptions are made to simplify this model within SEXTANT. First of all, the work done to move the astronaut's body over a flat surface ($\dot{W}_{wr} + \dot{W}_{wc}$) is assumed to be 0 W. In reality, this locomotion work increases as the surface traction decreases. However, SEXTANT does not currently have the ability to model these surface conditions on the lunar surface, and so a constant value is used. Furthermore, in SEXTANT, it is assumed that all of the heat generated by the inefficiency of work is transferred to the space suit. There is no body heat storage, and $\dot{Q}_s = 0$. In reality, there is heat flow from the muscles to the surrounding tissue and blood, and only some of the heat generated by the inefficiency of work is released to the space suit atmosphere by respiration, conduction, and radiation (Fiala et al. 1999). As a result, the amount of heat from the astronaut's metabolic workload is somewhat overestimated in SEXTANT. Taking these assumptions into account in Equation (4.1) and solving for the heat loss gives:

$$\dot{Q}_n = \dot{Q}_m - \dot{W}_w - \dot{W}_{ws} \tag{4.2}$$

The external mechanical work, $\dot{W}_w$, is the work required to change the potential energy of the astronaut by moving up- or downhill. The amount of energy used for external work at a constant speed can be quantified in a term called the mechanical efficiency (*E*) (Margaria 1976). This is equal to the mechanical work performed by the astronaut divided by his total energy consumption. This gives an equation for $\dot{W}_w$ as:

$$E = \frac{\dot{W}_w}{\dot{Q}_m} \qquad \dot{W}_w = E \cdot \dot{Q}_m \tag{4.3}$$

Margaria experimentally determined the mechanical efficiency as a function of slope, which can be seen in Figure 4.8 (1976).

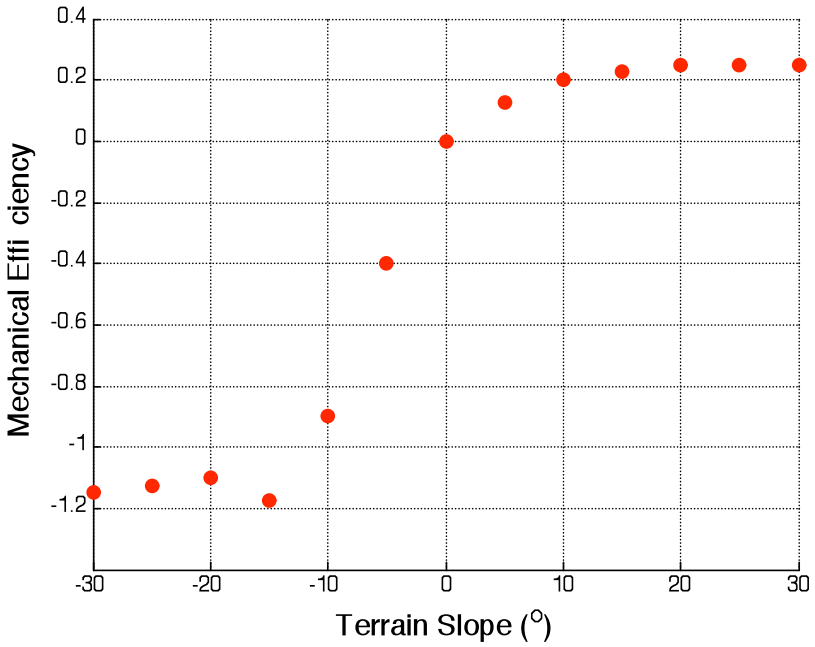


Figure 4.8. Mechanical efficiency as a function of the terrain slope (Adapted from Margaria 1976)

When the terrain slope is 0º, the efficiency is 0. This is because the astronaut's potential energy is not changing. When the astronaut is walking uphill, the mechanical efficiency is greater than 0 because external work is being performed. As the slope increases so does the efficiency, until it reaches a plateau of 0.25 at a slope of 20º. As Margaria states, "this is the characteristic mechanical efficiency of muscle performing positive work, as is also found in isolated muscle preparation" (1976, p. 75). Whenever the astronaut is walking downhill, the efficiency is negative. This occurs because the mechanical work being performed is also negative – the astronaut is traveling in the same direction as the gravitational force. As a result, additional heat from the negative work is being absorbed by the astronaut's muscles. The mechanical efficiency decreases as the terrain slope becomes more negative, until it reaches a value of -1.2 at a slope of -12º. It is important to note that energy is still being consumed because the leg muscles are still active. Margaria's data is approximated in SEXTANT by the piecewise function shown in Table 4.2. Figure 4.9 then plots both Margaria's data and this function.

Table 4.2. Curve fit for mechanical efficiency energetics data

Slope, α	Efficiency, η
α< -12°	-1.2
-12° ≤ α< -10°	0.15 • α + 0.6
-10° ≤ α< -5°	0.1 • α + 0.1
-5° ≤ α< 0°	0.08 • α
0° ≤ α< 5°	0.025 • α
5° ≤ α< 10°	0.015 • α + 0.05
10° ≤ α< 20°	0.005 • α + 0.15
20° ≤ α	0.25

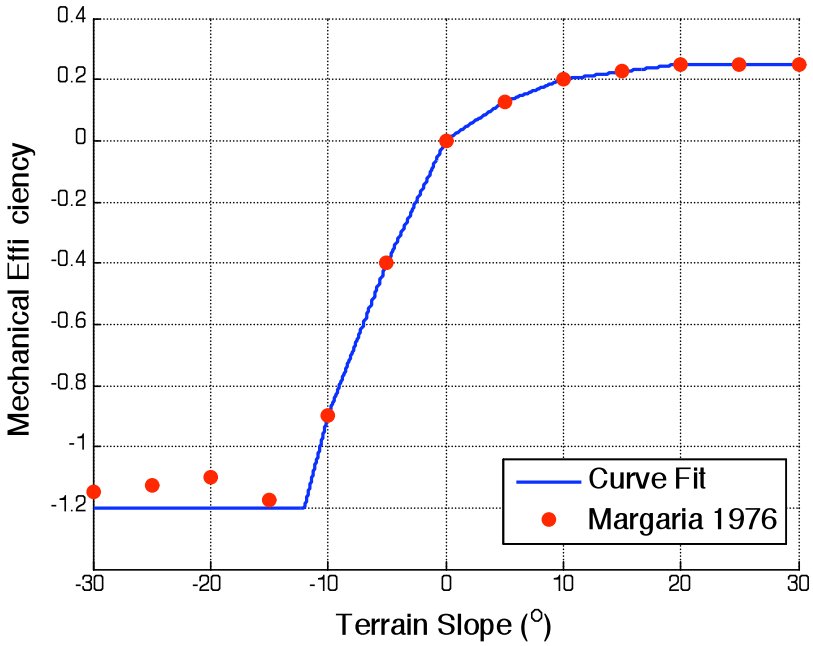


Figure 4.9. Mechanical efficiency as a function of the terrain slope, with curve fit (Adapted from Margaria 1976)

Carr and Newman suggest that the muscles are optimized for the Earth's gravitational field, and that their efficiency decreases as the planetary gravity decreases (2007b). Therefore, the efficiency found from Margaria's data must be multiplied by a scaling factor called E_{planet}, which tells how much the muscle's efficiency decreases relative to Earth. Carr and Newman found that E_{planet} for Mars is 0.78 and E_{planet} for the Moon is 0.48 (2007b). E_{planet} for Earth is, of course, 1. This gives the final equation for the external mechanical work:

$$\dot{W}_w = E_{planet} \cdot E \cdot \dot{Q}_m \tag{4.4}$$

The work used to move the space suit, $\dot{W}_{ws}$, is a significant component of the work done by the astronaut. This work is required to compress the gas within the space suit and deform the space suit material. Carr and Newman noted that the *energy* required to move the space suit is 80% of the total non-basal metabolic rate – the amount of energy consumed for useful work (2007b). This equation is:

$$\dot{Q}_{ws} = 0.8\left(\dot{Q}_m - \dot{Q}_b\right) \tag{4.5}$$

$\dot{Q}_b$ is the basal metabolic rate, which gives $\dot{Q}_m - \dot{Q}_b$ as the non-basal metabolic rate. The basal metabolic rate can be determined from the metabolic rate equations in Table 3.2, with a velocity of 0. This gives:

$$\dot{Q}_b = 0.3772 \cdot m + 8.1765 \tag{4.6}$$

Equation (4.5) gives the energy required to move the space suit, while Equation (4.2) requires the *work* to move the space suit. This work can be found by multiplying the energy required for this task by an efficiency term, E_{ws}. This is:

$$\dot{W}_{ws} = E_{ws}\dot{Q}_{ws} \tag{4.7}$$

This efficiency is assumed to be the characteristic mechanical efficiency for muscles, which is given by Margaria as 0.25 (1976), multiplied by E_{planet}. This gives the final equation for $\dot{W}_{ws}$ as:

$$\dot{W}_{ws} = 0.2E_{planet}\left(\dot{Q}_m - \dot{Q}_b\right) \tag{4.8}$$

Combining Equations (4.2), (4.4), and (4.8), the total heat loss from the astronaut to the space suit is:

$$\begin{aligned} \dot{Q}_n &= \dot{Q}_m - E \cdot E_{planet} \cdot \dot{Q}_m - 0.2 \cdot E_{planet}\left(\dot{Q}_m - \dot{Q}_b\right) \\ \dot{Q}_n &= \dot{Q}_m\left(1 - E \cdot E_{planet} - 0.2 \cdot E_{planet}\right) + 0.2 \cdot E_{planet}\dot{Q}_b \end{aligned} \tag{4.9}$$

Figure 4.10 shows the metabolic expenditure per meter for an astronaut and the associated heat loss per meter. At slopes greater than -2.2° (more positive), the heat loss is less than the metabolic expenditure per meter. Conversely, at slopes less than -2.2° (more negative), the heat loss is actually greater than the metabolic expenditure per meter. This is because the astronaut is performing negative work, and heat is being absorbed by his muscles and released to the space suit environment.

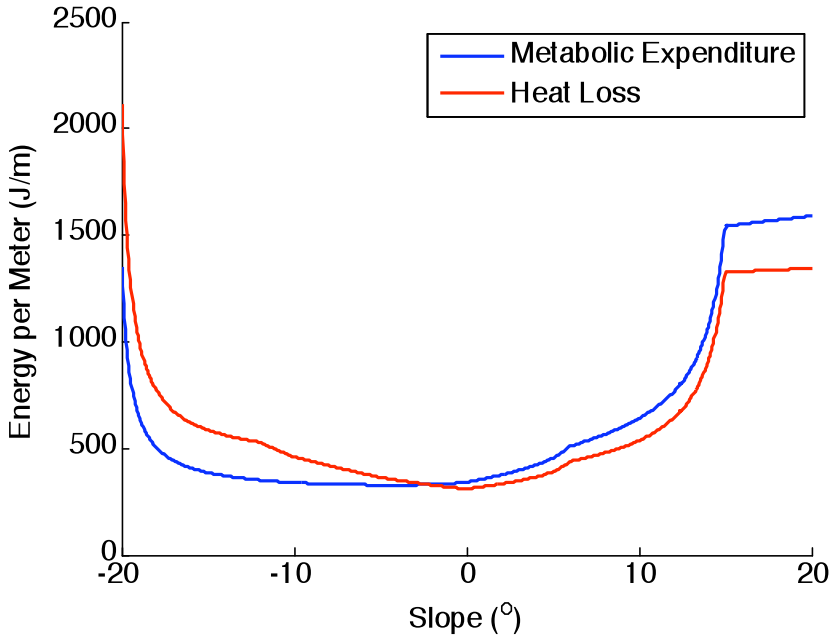


Figure 4.10. Astronaut metabolic expenditure and heat loss per meter

The space suit electronics waste heat is the heat that is produced by the inefficiencies of electrical components in the space suit. Ideally, the space suit electronics waste heat should vary depending on the astronaut's activities at a certain time. As more battery power is required for life support, thermal control, communications, and other components, the electronics waste heat will increase. However, SEXTANT does not have the ability to model the astronaut's changing activity level for a traverse. The electronics waste heat is therefore considered as a user-specified constant value throughout the entire EVA. It is denoted by the variable $\dot{Q}_{waste}$.

4.3.1.3. Heat Transfer to or from External Sources

The direct sun radiation is the largest source of heat into the space suit, when present. This heat flux is determined by the equation:

$$\dot{Q}_{sun} = V_{sun}\alpha_{suit}A_{\perp}I \tag{4.10}$$

V_{sun} is the percentage of the solar disk visible from that particular Path Point, as described in Section 4.2. α_{suit} is the absorptivity of the space suit, which describes the percentage of radiation energy impinging on the space suit that is actually absorbed. The space suit absorptivity used by SEXTANT is 0.18 (Larson and Pranke 2000, p. 714). $A_{\perp}$ is the space suit area perpendicular to the sun. This is important because solar radiation is only contacting this area, and not the entire

surface of the space suit. In SEXTANT, $A_{\perp}$ is assumed to be half of the total space suit surface area (*SA*). *SA* is approximated with the Mosteller formula for body surface area (1987).

$$SA = \sqrt{\frac{m \cdot h}{36}} \tag{4.11}$$

In this equation *m* is the mass of the space-suited astronaut in kg, and *h* is the height of the astronaut in m. Finally, *I* in Equation (4.10) is the flux of solar radiation arriving at the lunar surface. It is known that the solar flux at Earth is 1367 W/m^2, on average. Because the Earth – Moon distance is much, much smaller than the Earth – Sun distance, the average distance between the Sun and the Moon is approximately the same as that between the Sun and Earth. So, the value for the solar flux on the surface of the Moon is also equal to 1367 W/m^2 on average.

Radiation between the space suit and the environment is governed by the equation:

$$\dot{Q}_{suit-env} = \sigma \varepsilon_{suit} \varepsilon_{env} A_{suit} F_{suit-env} \left(T_{env}^4 - T_{suit}^4 \right) \tag{4.12}$$

In SEXTANT, the space suit thermally interacts with two different parts of the environment (besides the sun, which has already been accounted for): the lunar surface and deep space. As a result, Equation (4.12) is applied twice with slightly different parameters. If $\dot{Q}_{suit-env}$ is negative, heat is leaving the space suit to the environmental component, and if $\dot{Q}_{suit-env}$ is positive then heat is entering the space suit from the particular environmental component.

In Equation (4.12), σ is the Stefan-Boltzmann constant, which is equal to $5.6704 \cdot 10^{-8}$ W/m^2-K^4. ε is the emissivity of an object, which is another thermal parameter related to the absorptivity. The emissivity details the amount of radiation energy that is emitted from the object relative to a perfect radiator, also known as a black body, at the same temperature (Larson and Pranke 2000, p. 512). For the space suit, the value of ε_{suit} is 0.837; for the lunar surface, ε_{Moon} is 0.95; and for deep space, ε_{space} is 1 (Balinskas and Tepper 1994, Mendell 2010, Griffin and French 2004). A_{suit} is the total surface area of the space suit (*SA*), and not just half as was seen in Equation (4.10). This is because the $A_{\perp}$ in Equation (4.10) had to account for the area of the suit that was involved in the heat transfer. In Equation (4.12), this is accounted for by $F_{suit\text{-}env}$, the view factor between the suit and the environmental component, instead of by the area. The view factor takes on a value from 0 to 1, and shows the percentage of radiation leaving the space suit that is striking the environmental component. It is a general property of view factors that all of the view factors from any object sum to 1. Within SEXTANT, the only two view factors are between the space suit and the lunar surface and between the space suit and deep space. For

simplification, each of these is assumed to be 0.5. Finally, Equation (4.12) involves the temperature of the space suit and the specific component of the environment. The temperature of deep space is assumed to be 3 K (-270° C). The temperature of the lunar surface is a function of the sun visibility, and can be described by the equation:

$$T_{Moon} = 250 \cdot V_{sun} + 123 \tag{4.13}$$

The temperature reaches a maximum of 373 K (100° C) in complete sunlight and 123 K (-150° C) in complete shadow (Heiken 1991, pp. 35-36). This equation is only an approximation, as it does not directly take into account the amount of time the lunar surface has been in shadow or sunlight. This is important, because the lunar surface does not heat up and cool down instantly as it transitions between shadow and sunlight. Rather, there is a heating or cooling process that takes some time. However, this transient effect has been neglected in the current version of SEXTANT to simplify the model.

The external temperature of the space suit, T_{suit} in Equation (4.12), constantly changes with time. However, in SEXTANT, it is assumed to remain constant for each individual stage of the traverse and only change at Path Points between stages. This allows for the radiation heat transfer in Equation (4.12) to be easily calculated for a stage. The temperature of the space suit at the end of a stage (and consequently the beginning of the next stage) is in turn dictated by the total environmental heat flux, $\dot{Q}_{env} = \dot{Q}_{sun} + \dot{Q}_{suit-Moon} + \dot{Q}_{suit-space}$, during that stage by the equation:

$$T_{suit,2} = \frac{\dot{Q}_{env} t}{m c_{p,suit}} + T_{suit,1} \tag{4.14}$$

t is the stage time, m is the mass of the space suit (50 kg), and $c_{p,suit}$ is the specific heat of the space suit (which is assumed to be 1.09 kJ/kg-K, the specific heat of Kapton, one of the main materials from which the space suit is constructed) ("DuPont Kapton" 2010). $T_{suit,1}$ is the external temperature of the space suit at the beginning of the traverse stage, while $T_{suit,2}$ is the temperature at the end of the stage. The temperature of the space suit at the beginning of first stage of the traverse is assumed to be 300 K (27° C), the same temperature of the habitat from which the astronaut emerges.

The external temperature of the space suit determines the amount of heat conduction through the space suit wall during a stage of the traverse. This is the amount of external heat that contributes to the thermal load on the astronaut. It is described by the equation:

$$\dot{Q}_{ext} = CA_{suit}\left(T_{suit} - T_{atm}\right) \tag{4.15}$$

C is the conductivity of the space suit, which is taken to be 1.19 W/m^2-K (Campbell et al. 2000). A_{suit} is the surface area of the space suit and T_{atm} is the inside atmospheric temperature. It is assumed that the thermal control system works to keep the atmospheric temperature of the space suit at a constant 300 K (27° C). T_{suit} is once again taken to be the external space suit temperature during the stage. This heat flux will be either positive or negative (into or out of the space suit, respectively), depending upon the relative temperature gradient across the space suit.

4.3.1.2 Computation of Total Space Suit Heat Flux

The heat flux that must be balanced out by the thermal control system is the sum of the external heat flux conducted through the space suit and the two internal heat fluxes:

$$\dot{Q}_{total} = \dot{Q}_n + \dot{Q}_{waste} + \dot{Q}_{ext} \tag{4.16}$$

The internal heat fluxes are always positive, as they can only transfer heat into the space suit environment. The sign of the external flux will change depending upon the environmental conditions. The total heat flux can be multiplied by the stage time, which gives the total heat into or out of the space suit during the stage.

4.3.2 Space Suit Thermal Control System

In order to keep the space suit at a constant temperature of 300 K (27° C), the heat gain or loss $\dot{Q}_{total}$ in Equation (4.16) must be balanced by the heat rejection or addition of the thermal control system. If heat is being transferred out of the space suit ($\dot{Q}_{total} < 0$), then an equal amount of heat must be added to the system. If heat is being transferred into the space suit ($\dot{Q}_{total} > 0$), then an equal amount must be removed from the system. This is accomplished through the space suit thermal control system, depicted in Figure 4.11.

The thermal control system consists of three main components that can either reject heat from or add heat to the system. First of all, a radiator covered in Z-93 white paint (ε_{rad} = 0.92) is used to reject an initial amount of heat to deep space. Current space suits do not contain a radiator, but SEXTANT leaves the possibility open for future space suits. To remove the radiator from the model, the user can simply set the emissivity to 0. Secondly, a sublimator is used to release additional heat by converting ice into water vapor. If heat needs to be added to the system, this is done by the third component: a heater. Heat is moved between these three components and the space suit by water-filled tubes. These represent the liquid cooling and ventilation garment (LCVG) currently worn by all astronauts performing EVA. Water from the

LCVG is also used to replace the ice lost through sublimation. The astronaut can change the performance of the thermal control system by setting the LCVG temperature at the inlet to the space suit. On the Apollo space suit, this could take three values to give varying levels of cooling: 294 K, 288 K, and 280 K (21° C, 15° C, and 7° C) (Waligora and Horrigan 1975).

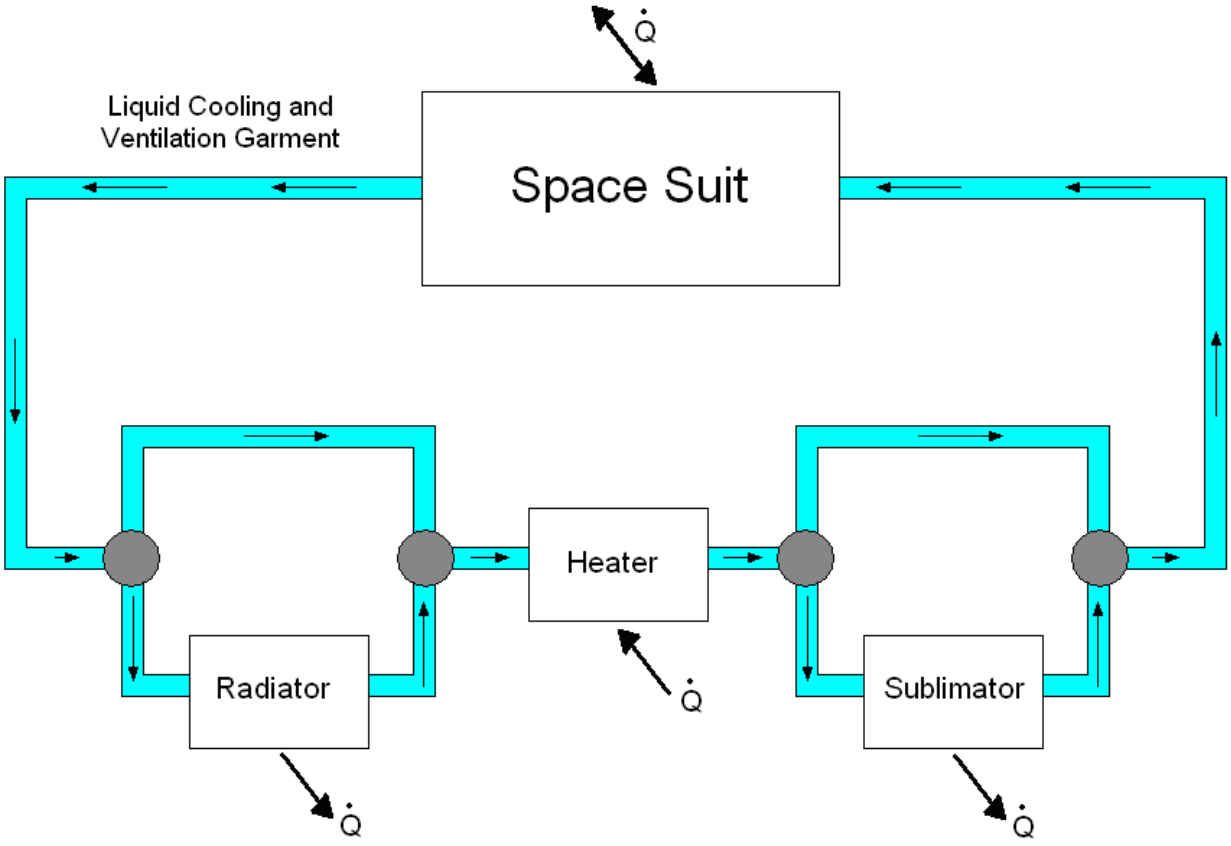


Figure 4.11. Diagram of space suit thermal control system

The temperature of the LCVG water at the space suit outlet is dependent upon the transfer of heat to or from the space suit, as per the equation:

$$T_{LCVG,out} = \frac{\dot{Q}_{total}}{\dot{m}_{water} \cdot c_{p,water}} + T_{LCVG,in} \tag{4.17}$$

$\dot{Q}_{total}$ is the total space suit heat flux found in Equation (4.16), $c_{p,water}$ is the specific heat of water and $\dot{m}_{water}$ is the mass flow rate of water through the LCVG. In SEXTANT, $c_{p,water}$ is taken as 4186 J/kg-K and $\dot{m}_{water}$ is assumed to be 0.03 kg/sec (Balinskas and Tepper 1994).

There are three conditions under which the thermal control system can operate, based on the total space suit heat flux:

1. Heat is being transferred *from* the space suit to the environment
2. Heat is being transferred *to* the space suit, and the radiator is sufficient to remove all of this heat
3. Heat is being transferred *to* the space suit, but the radiator is not sufficient to remove this heat

In the first case, heat must be added to the space suit. The LCVG water bypasses the radiator and flows through the heater. As stated above, the heater then adds an amount of heat to the water that is equal to the amount removed from the space suit. This brings the temperature of the LCVG water back to the inlet value specified by the astronaut. The flow then bypasses the sublimator and returns to the space suit inlet.

In the second case, the heat being added to the space suit is rejected to deep space by the radiator. This is governed by the same general equation as Equation (4.12):

$$\dot{Q}_{rad-space} = \sigma \varepsilon_{rad} A_{rad} \left(T_{space}^4 - T_{rad}^4 \right) \tag{4.18}$$

As was stated earlier, this equation does not contain ε_{space} because this value is 1. The amount of heat that the radiator can reject is a function of its surface properties (ε_{rad}), area (A_{rad}), and temperature (T_{rad}). The radiator is separated from the LCVG water tubes by a gas gap filled with a low pressure gas (Campbell et al. 2000, Trevino et al. 2004). This gas facilitates heat transfer between the cooling water and the radiator. However, the gap can be evacuated to reduce the heat transfer to the radiator and consequently the temperature of the radiator. This allows the radiator to reject the exact amount of heat that is added to the system, cooling the LCVG water to the correct inlet temperature. The flow then bypasses the sublimator before returning to the space suit.

In the final case, the radiator cannot reject all the heat transferred to the space suit. The sublimator must be used to release all of the remaining heat from the LCVG water by transforming solid water ice into water vapor. This then decreases the water temperature to the space suit inlet value. This method of thermal control has been used in American space suits since the Apollo program. In sublimation, the mass of ice required is governed by the equation:

$$\dot{m}_{sub} = \frac{\dot{Q}}{h} = \frac{\dot{m}_{water} c_{p,water} \left(T_{sub,out} - T_{sub,in} \right)}{h} \tag{4.19}$$

$T_{sub,in}$ is the temperature of the LCVG water entering the sublimator, and $T_{sub,out}$ is the temperature leaving the sublimator – the space suit inlet temperature. h is the heat of sublimation

for ice, which is 2594 kJ/kg. This brings the temperature of the LCVG water exactly to the correct inlet temperature. The mass of ice sublimated away is then replenished by the same amount of LCVG water.

4.4 Rover Power Model

Power for transportation rovers comes from two different sources: a solar array and batteries. The first can always provide power when in sunlight, but the second is of finite capacity. The limited amount of battery energy can be a concern, as the rover relies on this source of power when there is insufficient sunlight. As a result, the astronauts must ensure that their rover has enough battery power to complete a planned traverse. Unlike the space suit, which cannot replenish its water supply mid-traverse, the rover can recharge its batteries when in the sun through the excess power generated by the solar array.

The power produced by a solar array (P_{SA}) is described by the following equation:

$$P_{SA} = V_{sun} I \eta_{SA} A_{SA} \tag{4.20}$$

As in the astronaut thermal model, V_{sun} is the percentage of the solar disk visible, and I is the solar flux on the surface of the Moon (1367 W/m^2). η_{SA} is the efficiency of the solar array, which describes how much of the incoming solar flux the solar array can convert to usable power. The efficiency of the solar array depends on its materials and construction. Table 4.3 lists a few of the common solar cell types and their efficiencies:

Table 4.3. Solar cell efficiencies (Hong 2007)

Solar Cell Type	Efficiency, η_{SA}
Silicon	0.17
InGap/GaAs/Ge dual-junction	0.235
InGAp/GaAs/Ge triple-junction	0.26

These are ideal efficiencies, and in themselves do not account for losses due to design inefficiencies, shadowing from the rover itself, or temperature variations. To capture these power losses, SEXTANT multiplies the efficiencies in Table 4.3 by 0.77 (Wertz and Larson 1999). Finally, the last term of Equation (4.20) is A_{SA}, the area of the solar array. Equation (4.20) assumes that the solar array is vertical and has the ability to rotate to remain perpendicular to the incoming solar rays. This allows for the maximum possible power production. Another assumption is that the solar array is new, and its performance has not degraded with time on the

lunar surface. In reality, radiation, thermal outgassing of the material, micrometeroids, and thermal cycling can all decrease the efficiency of a solar array over time (Wertz and Larson 1999). The user has the ability to change the solar array efficiency in SEXTANT, and can manually decrease the value used in order to account for an older solar array.

The total energy capacity of the batteries is described in part by their specific energy (W-hr/kg). This is a property of their construction and materials. The specific energy can be multiplied by the total mass of the batteries to give their energy in W-hr. Currently in SEXTANT, the transportation rovers are assumed to have Yardney Technical Products Lithium-Ion (Li-Ion) batteries with a specific energy of 145 W-hr/kg (Hong 2007). These were chosen because they have been successfully used for over six years on the Mars Exploration Rovers. The mass of the batteries is assumed to be 269 kg, giving a total energy capacity of 39 kW-hr (Hong 2007). While these numbers are currently used within SEXTANT, they are easily modifiable and can be changed by the user to the specifications of any transportation rover.

4.4.1 Rover Energy Consumption During a Traverse

As with the astronaut thermal model, there are three possible operating conditions:

1. The power from the solar array is enough to completely power the rover
2. The power from the solar array is not enough to completely power the rover, and the batteries must provide some power
3. There is no power from the solar array and the batteries must completely power the rover

In the first case, the batteries are not required at all; the solar array is sufficient. If the solar array is providing more energy than necessary, the excess power can be used to recharge the batteries. If the batteries are already at full capacity, the excess power is shunted out of the rover as heat. This operating condition generally occurs whenever there is complete or nearly-complete sunlight. The second case occurs when there is some sunlight, but not enough to completely power the rover. As a result, the batteries must be called into service to provide the additional power required. Finally, in the third case, the rover is completely in shadow and the solar panels do not provide any energy. The batteries are completely responsible for powering the rover.

4.5 Conclusion

The major contribution of this thesis has been the determination of shadowing on the lunar surface and the calculation of the astronaut thermal load and the power consumption and generation for transportation rover. Once a traverse has been planned on the LOLA-derived elevation map of the lunar south pole, the sunlit and shadowed areas are determined by comparing the position of the sun to a pre-built horizon database. This horizon database specifies the altitude of the horizon from all points on the elevation map in 720 different directions spaced at 0.5° increments. From this information, SEXTANT can determine the percentage of the solar disk visible from all points along the path, which gives the shadowing. While interesting in itself, the shadowing is primarily used to determine the usage of important consumables for both astronauts and rovers. The heat load on an astronaut space suit is dependent upon both internal and external heat sources and sinks. The external heat flux is completely dependent upon the shadowing conditions, while the internal heat flux varies with the astronaut's metabolic rate and the terrain slope. The astronaut thermal model tracks the amount of water required to replenish ice used in sublimation cooling and the heater power used for heating throughout the entire traverse. The rover power model details when there is enough sunlight for the rover to rely completely on solar power, and when the batteries must be discharged for power. SEXTANT determines the battery energy level, which is the main rover consumable, at all points along the traverse. These capabilities are all demonstrated in Chapter 5, which shows a number of example traverses.

This page intentionally left blank.

The crew had more paths to prepare,
So SEXTANT helped them to compare.
But they couldn't walk over,
So they drove in the rover.
And they had enough power to spare.

5 Example Astronaut and Rover Traverses

The capabilities of SEXTANT can be used to realistically simulate traverses on the lunar surface years before humans return to the Moon. This is beneficial to designers of lunar architecture, as SEXTANT's ability to model the shadowing-related consumables of sample traverses can help to properly size the thermal control system of space suits and the electrical power systems of transportation rovers required for future explorations. This chapter will explore a number of example traverses that demonstrate these abilities of SEXTANT, and will discuss the resultant dynamics of the astronaut thermal control system and the rover power system.

All example traverses undertaken by suited astronauts or transportation rovers in this chapter occur around the lunar south pole, which is the region covered by the LOLA elevation map used in SEXTANT. A map of the north pole is also available, but not yet used in SEXTANT. This entire map is 500 km by 500 km – an extremely large area for traverses. So, a 120 km by 120 km submap of this area has been selected for the example traverses (Figure 5.1).

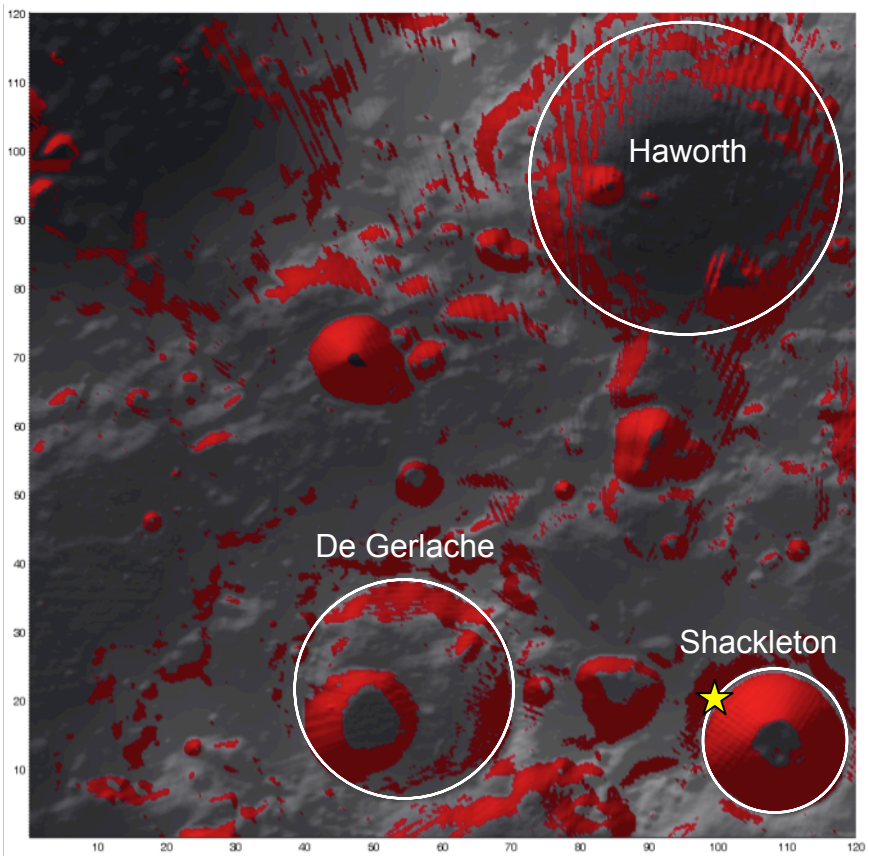


Figure 5.1. 120 km by 120 km submap for example traverses

The south pole is noted by the star, and three prominent craters are labeled. This submap extends 100 km to the left and above the lunar south pole, and 20 km to the right and below. Red areas on the 3D mapping interface indicate obstacles as defined by a maximum slope of 15°.

The starting time for each traverse is set to June 4, 2010 at 10:00 am EDT[2]. At this time, the illumination over the entire submap is as presented by Figure 5.2 and Figure 5.3.

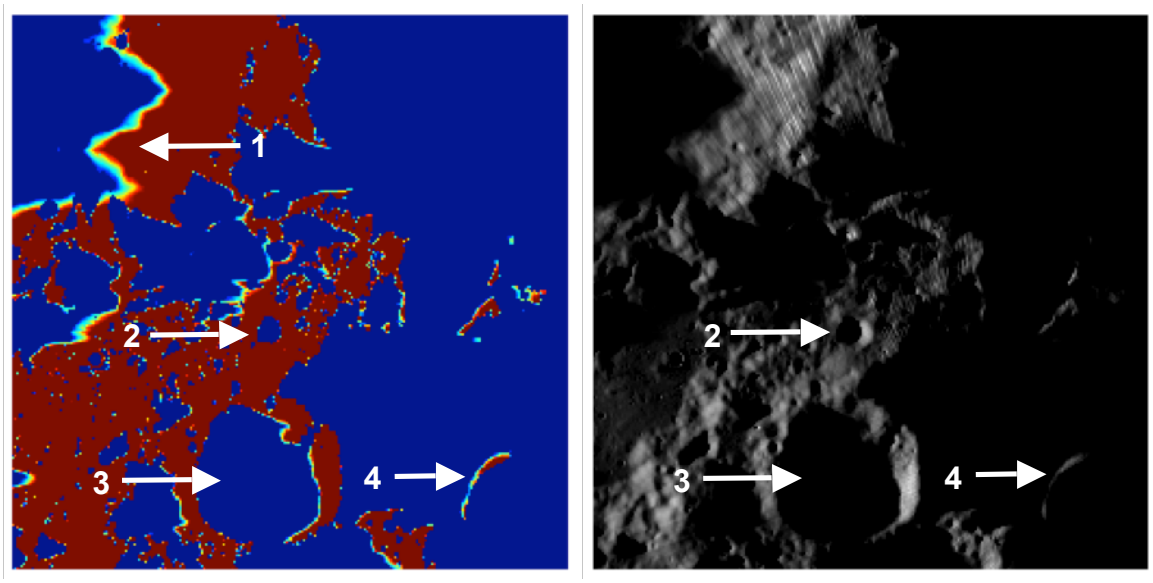


Figure 5.2 (left). Submap sun illumination on June 4, 2010, at 10:00 am EDT

Figure 5.3 (right). Photometric rendering of submap sun illumination

In Figure 5.2, the blue areas are those in complete shadow ($V_{sun} = 0$), and the red areas are in full sunlight ($V_{sun} = 1$). The gradient from blue to red (notation 1 on Figure 5.2), represents regions of partial shadow ($0 < V_{sun} < 1$). Figure 5.2 shows that the left half of the elevation submap is generally in sunlight, while the right half is generally in shadow. It can also be seen that due to the low sun angle, the bottoms of many craters (notations 2 and 3 on Figure 5.2 and Figure 5.3) are in shadow. Similarly, the rim of many craters, like Shackleton (notation 4), rise high enough that they are in sunlight even when the surrounding area is in shadow.

[2] An important date and time in the life of this author and thesis.

5.1 Suited Astronaut EVAs

Three example astronaut EVAs have been planned, all satisfying the same three constraints. These specify the maximum feasible distance, time, and metabolic cost of an EVA. Table 5.1 describes these constraints for suited astronaut EVAs.

Table 5.1. Constraints for suited astronaut EVAs

Constraint	Value
Distance	20 km
Time	8 hours
Metabolic Cost	12000 kJ

In all traverses, the astronaut is assumed to have a mass of 120 kg, which includes his body mass and space suit. The emissivity of the space suit radiator has been set to 0, which removes the radiator from the thermal control system. This mimics the Apollo space suits, which did not have a radiator and only used sublimation for cooling.

5.1.1 EVA 1 – Explanatory Traverse in Shadow and Sunlight

The first example traverse, called EVA 1, takes places in the lower center of the terrain submap, shown in Figure 5.4. Figure 5.5 shows a closer view of EVA 1, outlined as the yellow square in Figure 5.4.

In EVA 1, the astronaut travels from the habitat located on the rim of de Gerlache Crater (Activity Point 1) down to the crater's bottom. The traverse does not return to the habitat, and it can be thought that the astronaut rendezvous with a transportation rover at the end. The astronaut may be sent on this one-way traverse because the route down the crater side is too treacherous for a rover to travel, or requires exploration by a suited astronaut for another reason.

From the habitat, the astronaut travels along the rim of de Gerlache to Activity Points 2 and 3. It appears as though the traverse passes through an obstructed area (notation 1 on Figure 5.5), but in reality there is a small corridor of traversable ground through with the path travels. After reaching Activity Point 3, the astronaut turns and begins traveling down into the crater. Activity Point 4 is partially down the crater wall, while Activity Point 5 is at the bottom of the crater.

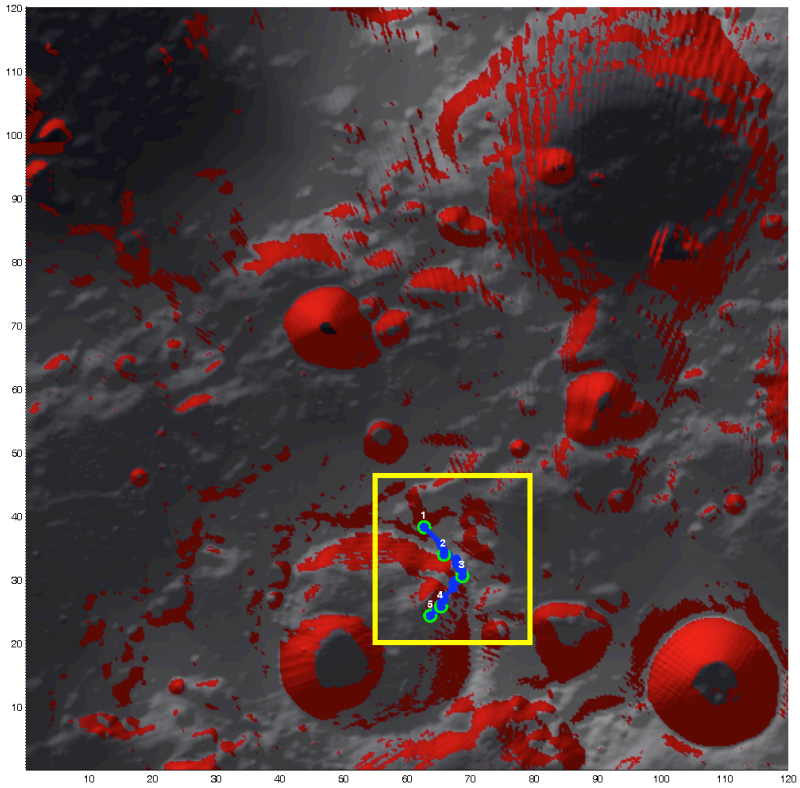


Figure 5.4. Location of EVA 1 on 120 km by 120 km submap

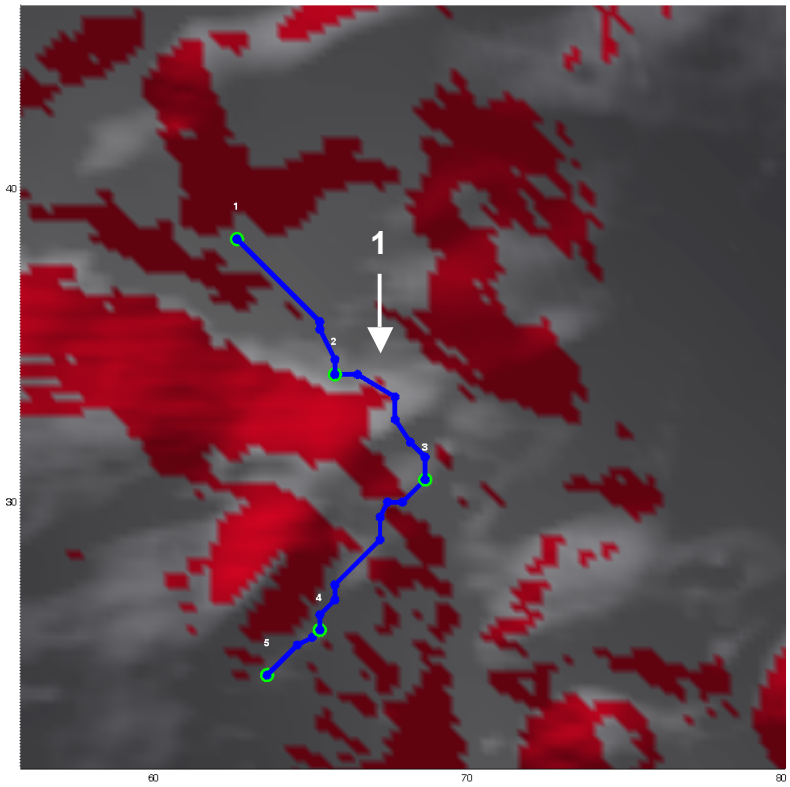


Figure 5.5. Close-up view of EVA 1 on 25 km by 25 km map

5.1.1.1 Traverse Distance, Time, and Metabolic Cost for EVA 1

Table 5.2 details the cumulative distance, time, and metabolic cost from the habitat to each Activity Point in EVA 1.

Table 5.2. Cumulative traverse metrics at EVA 1 Activity Points

Activity Point	Distance from Habitat (km)	Time from Habitat (hr:min)	Metabolic Cost from Habitat (kJ)
2	5.61	2:59	2516.2
3	10.87	4:36	4394.5
4	17.34	6:25	6603.1
5	19.62	7:06	7382.1

Throughout the traverse the astronaut loses 1966 m of elevation. Figure 5.6 shows the elevation profile of the traverse, with elevation normalized to the lowest point reached, which is at Activity Point 5.

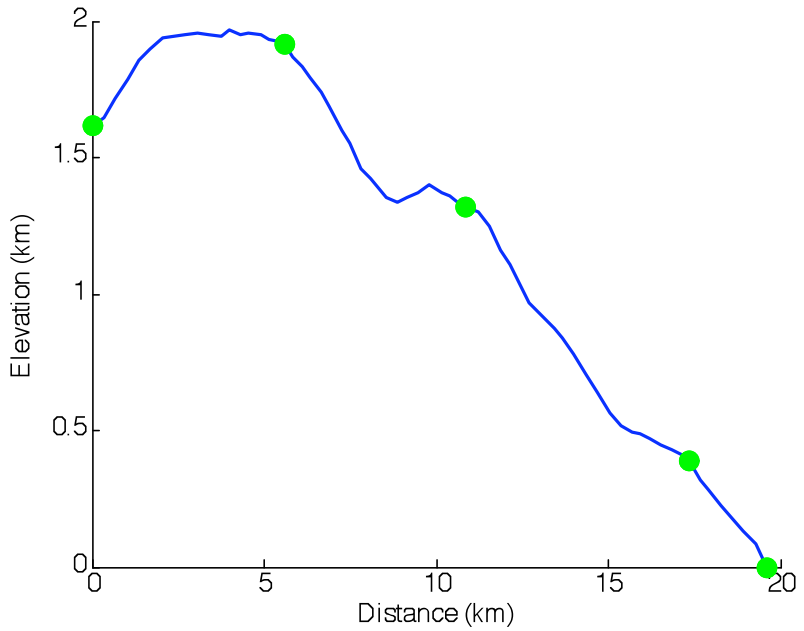


Figure 5.6. Elevation profile of EVA 1

As stated in Section 3.3.3, the depiction of the traverse can be simplified by removing extraneous Path Points and only keeping those where the astronaut changes direction. While the path looks the same on the 3D mapping interface, some information is lost. There is a longer distance between consecutive Path Points, meaning that the traverse resolution is poorer. This leads to less-accurate computations of the distance, time, metabolic cost, and thermal metrics.

Therefore, the path simplification has been turned off for the computation of all figures for the example traverses. Consecutive path points are either 240 m apart (if travel between them is lateral) or 339.4 m apart (if travel between them is diagonal).

Figure 5.7 and Figure 5.8 present the cumulative metabolic cost of EVA 1 with respect to distance and time along the traverse.

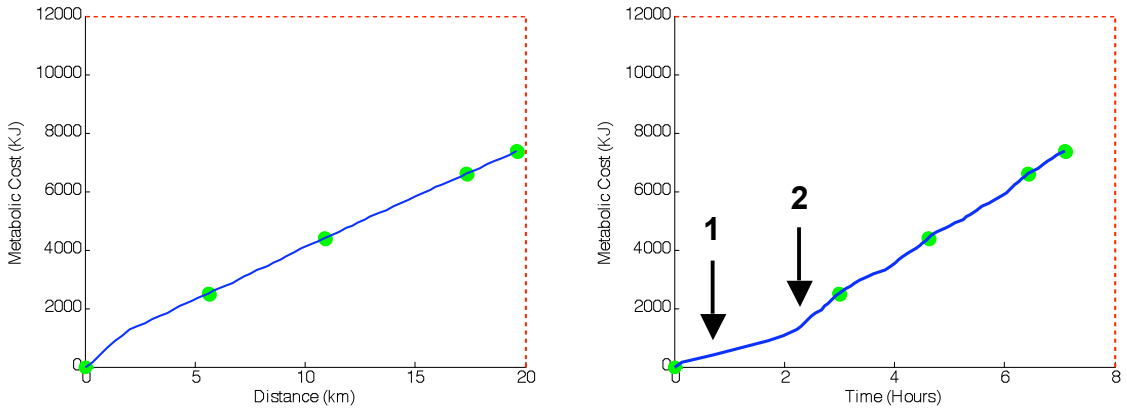


Figure 5.7 (left). Cumulative metabolic cost of EVA 1 with respect to distance

Figure 5.8 (right). Cumulative metabolic cost of EVA 1 with respect to time

The constraints specified in Table 5.2 are plotted on Figure 5.7 and Figure 5.8 as dashed red lines. These allow the user to quickly see how closely the traverse encroaches on the constraints. EVA 1 is well within the time and metabolic cost constraints, but closer to the distance constraint of 20 km.

In Figure 5.7, the slope of the blue line is relatively constant, except near the beginning. This implies that the metabolic cost per unit distance does not vary greatly over most of the EVA. Contrarily, the slope of the line in Figure 5.8 is not constant for the entire traverse. The portions with a shallower slope (like near the beginning, notation 1) indicate that the metabolic cost per unit time is smaller than the portions of the graph with a steeper slope (like just before Activity Point 2, notation 2).

5.1.1.2 Traverse Shadowing and Thermal Metrics for EVA 1

The sun illumination has been determined with a time step of 120 minutes. Over this period, the sun moves approximately 1°. So, the Path Points that make up the 426-minute EVA 1 are rounded to the nearest integer of 120. This gives five bins: 0, 120, 240, and 360, and 480 minutes since the start of the traverse. As is explained in Section 4.2, SEXTANT computes the

shadowing for each Path Point along the traverse. The shadowing of each stage between consecutive Path Points is then determined as the average of the sun visibility for the two ends. The thermal flux into or out of the space suit is computed for each stage, along with the amount of water required for sublimation cooling and the heater power. These two metrics can be analyzed for individual stages, or for the EVA as a whole

Figure 5.9 shows the path of EVA 1 with color-coded segments that denote the shadowing. The gradient ranges from black (complete shadow, $V_{sun} = 0$) through grey (partial shadow, $0 < V_{sun} < 1$) to white (complete sunlight, $V_{sun} = 1$).

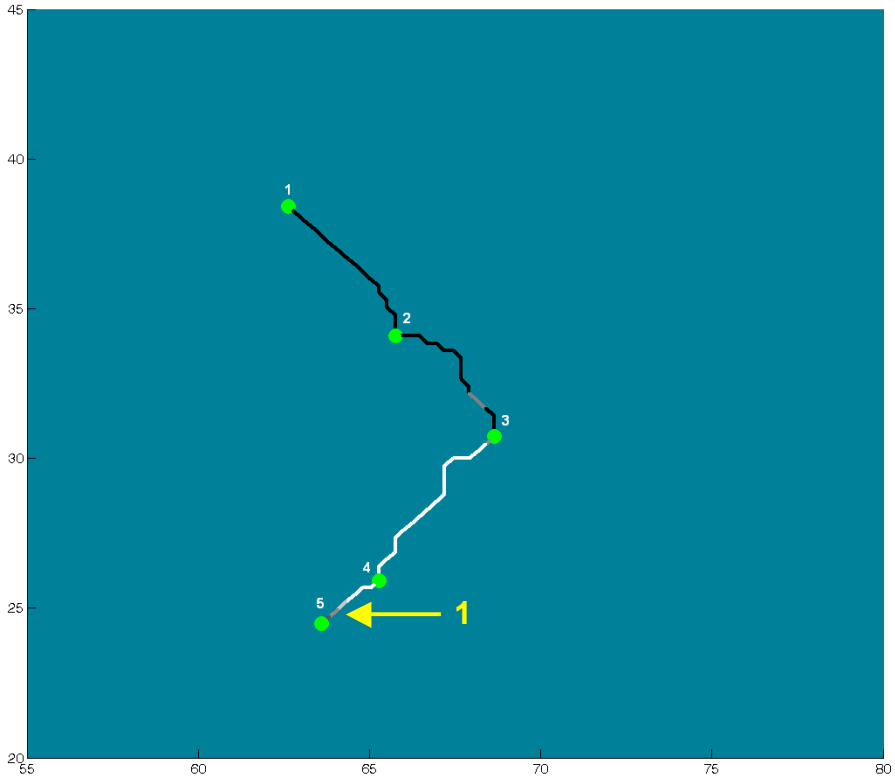
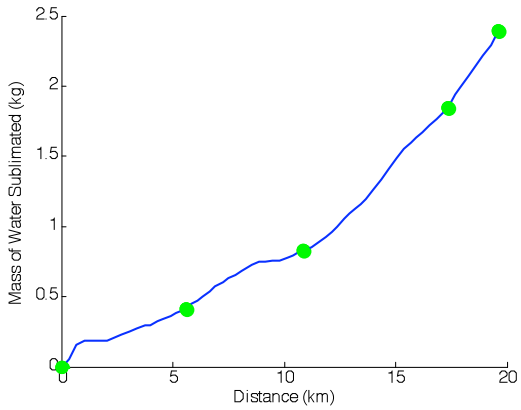


Figure 5.9. Shadowing along EVA 1

While traveling between the habitat (Activity Point 1) and Activity Point 3, the astronaut will be in shadow. Once he crests rim of de Gerlache just after Activity Point 3, he abruptly transitions into sunlight. The astronaut remains in sunlight until he arrives at the crater floor. Here, there is a more gradual transition into shadow, as seen by the noticeable grey path right before Activity Point 5 (notation 1 on Figure 5.9). The astronaut finishes EVA 1 in shadow at Activity Point 5.

5. Example Traverses

The amount of water required to replenish the ice sublimated away for cooling, and the heater power required to keep the astronaut warm are shown in Figure 5.11 through Figure 5.17.



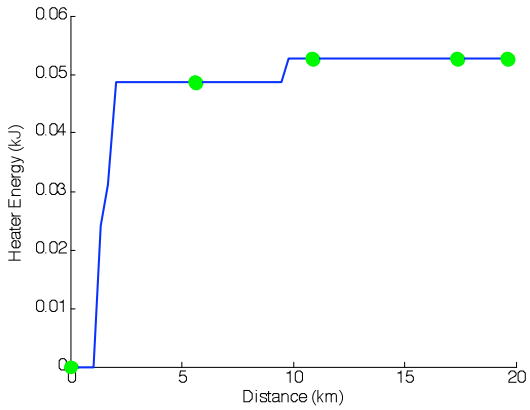
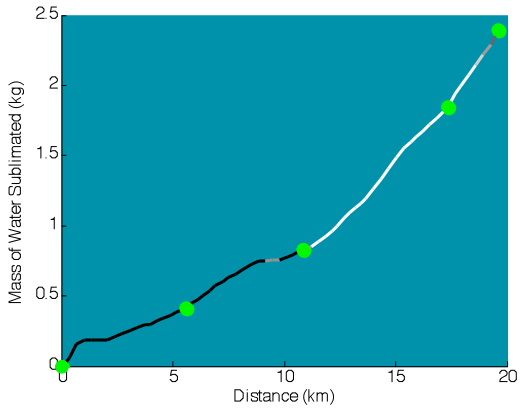


Figure 5.10 (left). Mass of sublimated water during EVA 1 with respect to distance

Figure 5.11 (right). Heater energy during EVA 1 with respect to distance



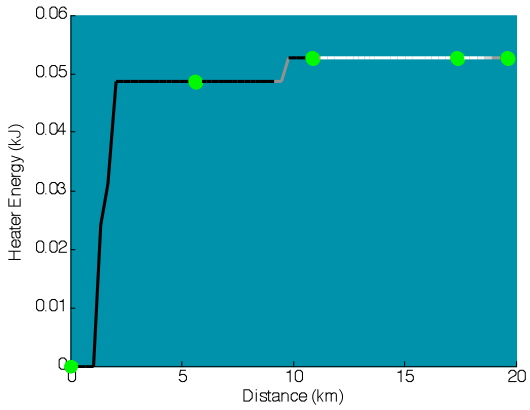
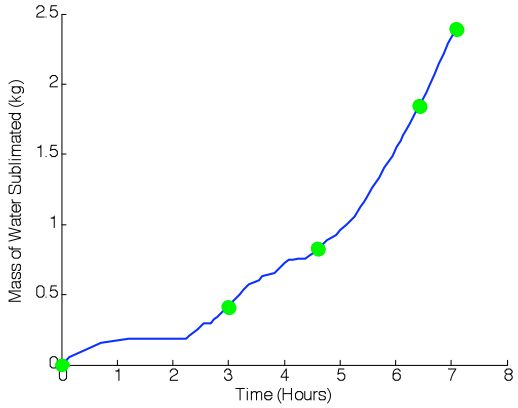


Figure 5.12 (left). Mass of sublimated water during EVA 1 with respect to distance, with shadowing

Figure 5.13 (right). Heater energy during EVA 1 with respect to distance, with shadowing



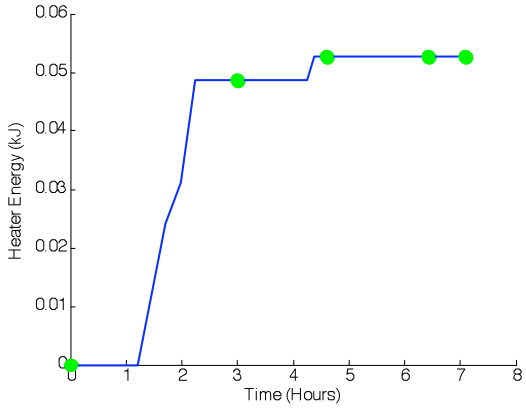
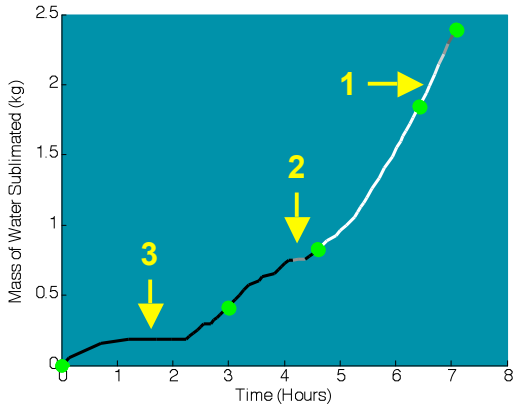


Figure 5.14 (left). Mass of sublimated water during EVA 1 with respect to time

Figure 5.15 (right). Heater energy during EVA 1 with respect to time



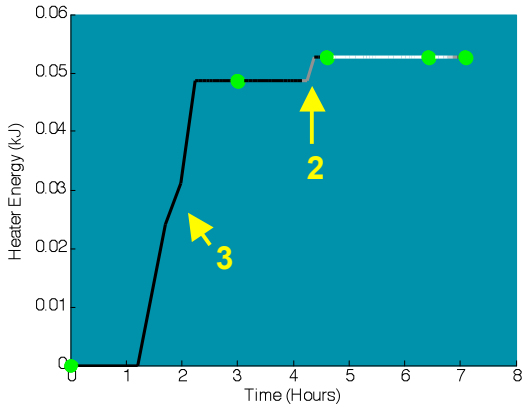


Figure 5.16 (left). Mass of sublimated water during EVA 1 with respect to time, with shadowing

Figure 5.17 (right). Heater energy during EVA 1 with respect to time, with shadowing

The left column of figures (Figure 5.10, Figure 5.12, Figure 5.14, and Figure 5.16) shows that the water needed to replenish the sublimated ice is increasing throughout most of the traverse. However, the rate at which the water is required – represented by the slope of the line in the left column of figures – varies throughout the traverse. The rate is greatest when the astronaut is in full sunlight near the end of the traverse (notation 1 on Figure 5.16). It is slower elsewhere, and even decreases to 0 kg/hr at two periods (notations 2 and 3 on Figure 5.16). During these periods, no cooling is being used by the space suit. Rather, from looking at the right column of figures (Figure 5.11, Figure 5.13, Figure 5.15, and Figure 5.17), it can be seen that heating is required during these periods (notations 2 and 3 on Figure 5.17). This is a function of the design of the thermal control system – either cooling or heating are required, never both at the same time. The right column of figures also shows that only 52.7 J of heating are required during the

traverse. This is a trivial amount. The driving forces behind these thermal dynamics can be seen in looking at the components of the heat flux to and from the space suit during the traverse.

As was discussed in Section 4.3.1, there are three main components of this heat flux: the electronics waste heat from inside the space suit, metabolic heat released from the astronaut's body due to inefficiencies of work, and external heat from the environment. The first is specified to be a constant value of 50 W, while the latter two vary throughout the traverse. Figure 5.18 shows the three components of heat flux with respect to time for each traverse stage, as well as the total heat flux. Because the y-axis of Figure 5.18 is in Watts and the x-axis is in seconds, taking the area under each curve will give the amount of heat for each stage and source/sink. The stage times are not constant throughout the traverse, as the astronaut's velocity for a particular stage is dependent upon the terrain slope. So, it is beneficial to plot the heat for each stage of the traverse, as in Figure 5.19.

Heater power is required whenever the total heat flux to the space suit is negative – more heat is leaving the space suit then is entering it. Figure 5.18 and Figure 5.19 show that there are only four stages in which this occurs (notations 1-4). This confirms the observations from the right column of figures above. Because there is no radiator, cooling by sublimation is required whenever the total heat flux is positive. In these cases, more heat is entering the space suit than is leaving it. The rate at which water is required for cooling is dependent upon the magnitude of the heat flux into the space suit. The greater the heat flux, the faster water is consumed for sublimation.

The total heat flux into or out of the space suit is determined by the interplay between the external heat sources and the internal heat sources. The internal heat sources are always positive, meaning heat is always transferred to the space suit. The metabolic heat flux is always positive, but varies between 106.8 W and 453.6 W. As Equation (4.9) states, the value of this heat flux is dependent upon the metabolic cost and mechanical efficiency of astronaut motion over a traverse stage. At their cores, both of these parameters are functions of the stage terrain slope.

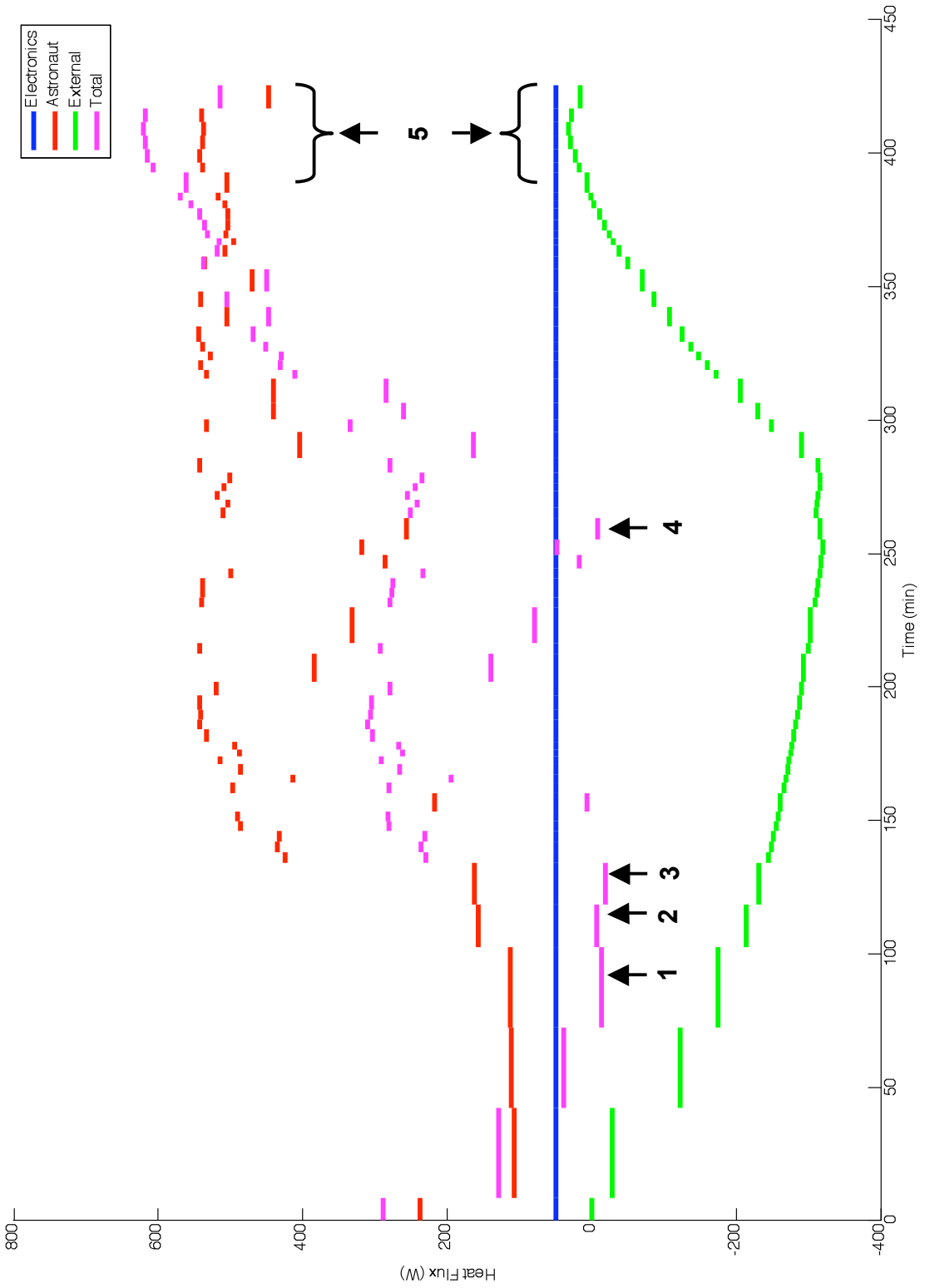


Figure 5.18. Heat flux with respect to time for EVA 1

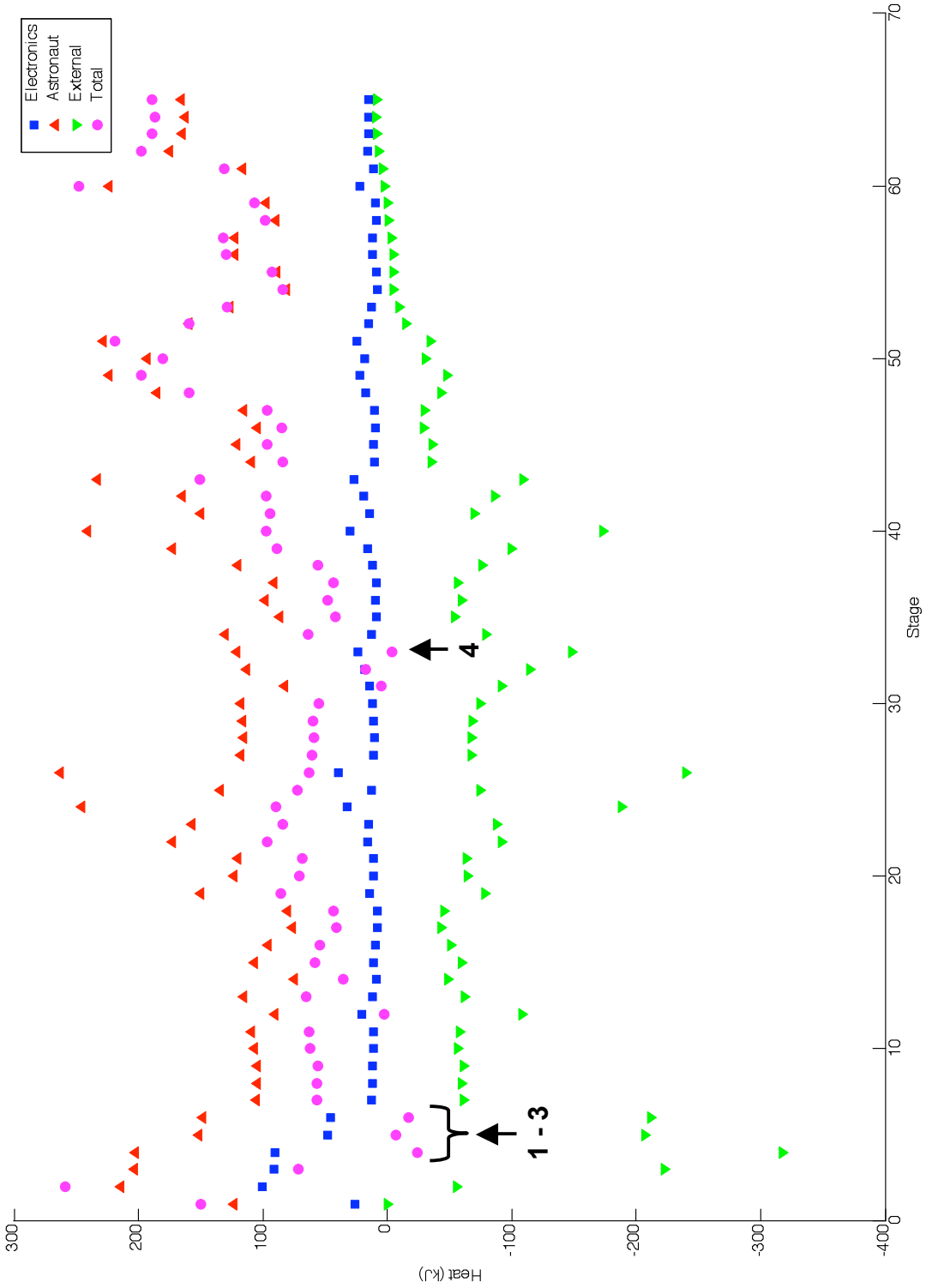
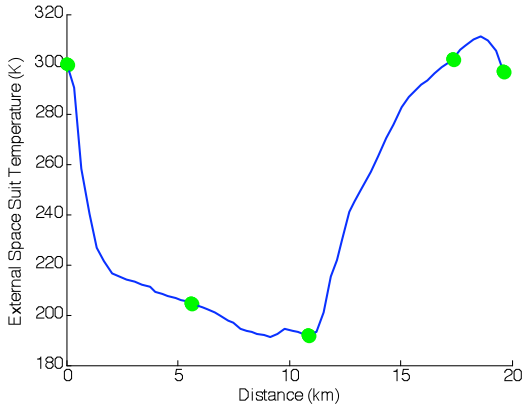


Figure 5.19. Heat with respect to traverse stages for EVA 1

Unlike the internal heat sources, the external heat flux can either be into or out of the space suit, depending on the environmental conditions. In EVA 1, the external heat flux begins at 0 W and then becomes increasingly negative. After reaching a minimum of -320 W, it subsequently increases to a maximum 32.32 W, after which it decreases slightly again. Because heat from the environment reaches the astronaut by conduction through the space suit wall, the dynamics of the external heat flux are a function of the external space suit temperature. This parameter is shown in Figure 5.21 through Figure 5.23.



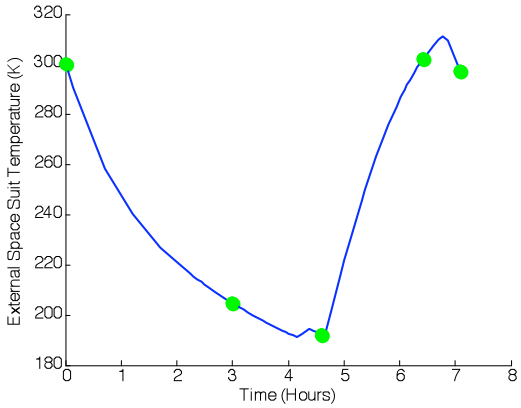
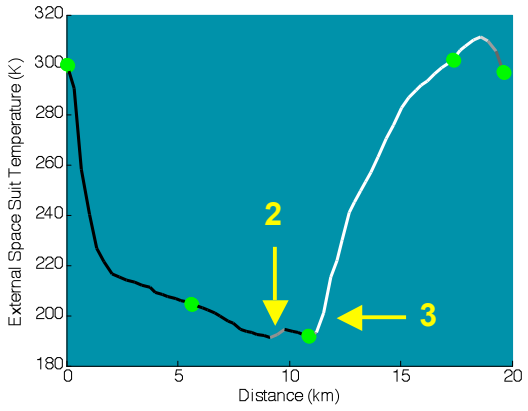


Figure 5.20 (left). External space suit temperature during EVA 1 with respect to distance

Figure 5.21 (right). External space suit temperature during EVA 1 with respect to time



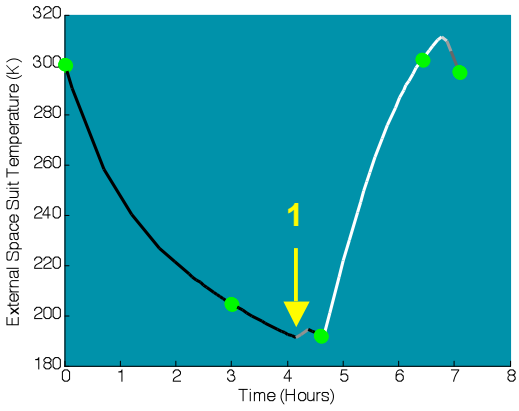


Figure 5.22 (left). External space suit temperature during EVA 1 with respect to distance, with shadowing

Figure 5.23 (right). External space suit temperature during EVA 1 with respect to time, with shadowing

5. Example Traverses

The external space suit temperature is dictated by the temperature of the environment and the incoming solar radiation, as explained by Equations (4.10) and (4.12). For each unique shadowing condition (V_{sun}), there exists a thermal equilibrium temperature (T_{equil}) where the environmental heat into the space suit's external surface equals the heat radiated from this surface to the environment. This temperature is governed by Equation (5.1)

$$T_{equil} = \left(\frac{V_{sun}\alpha_{sun}A_{\perp}I + \sigma\varepsilon_{suit}\varepsilon_{Moon}A_{suit}F_{suit-Moon}T_{Moon}^4 + \sigma\varepsilon_{suit}A_{suit}F_{suit-space}T_{space}^4}{\sigma\varepsilon_{suit}\varepsilon_{Moon}A_{suit}F_{suit-Moon} + \sigma\varepsilon_{suit}A_{suit}F_{suit-space}} \right)^{1/4} \tag{5.1}$$

Figure 5.24 plots the thermal equilibrium temperature as a function of the percentage of the solar disk visible.

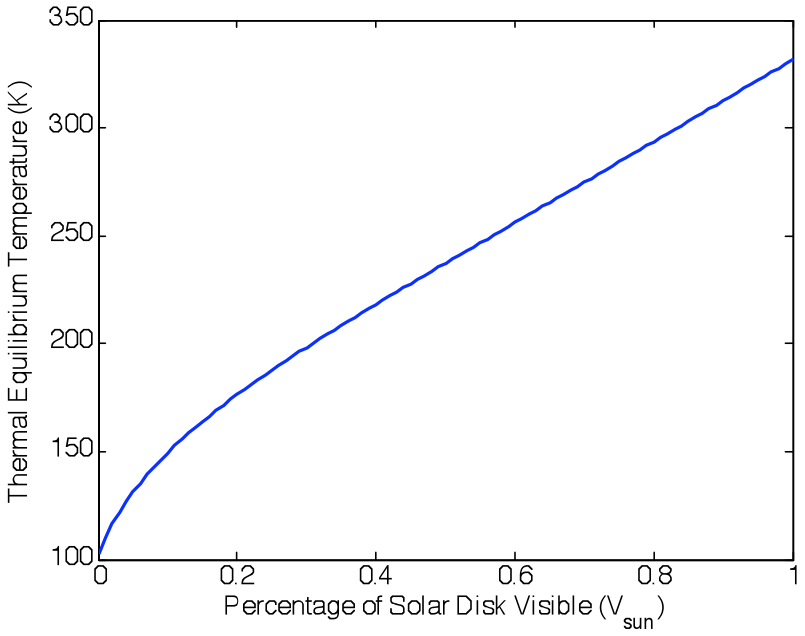


Figure 5.24. Thermal equilibrium for varying shadowing conditions

Whenever the astronaut is in complete shadow, the thermal equilibrium temperature is 102.8 K, and whenever the astronaut is in complete sunlight, the thermal equilibrium temperature is 331.6 K. The external temperature of the space suit always moves towards the current thermal equilibrium temperature over each traverse stage. The rate of temperature change is proportional to the difference between the thermal equilibrium and external space suit temperatures, each to the fourth power $\left(T_{equil}^4 - T_{suit}^4\right)$.

In EVA 1, the astronaut initially steps out of the habitat into shadow. His external space suit temperature is 300 K, while the thermal equilibrium temperature is 102.8 K. As a result, the

external space suit temperature decreases. The conductive heat transfer through the space suit wall is a function of the temperature difference between the outside and inside of the space suit, as explained in Equation (4.15). So, as the external space suit temperature decreases more heat is lost from the space suit. The external space suit temperature reaches a minimum of 191.1 K at 4 hours and 9.5 minutes into the traverse (notation 1 on Figure 5.23). The resultant external heat flux is -320 W, meaning that heat is leaving the space suit to the environment.

The external space suit temperature increases briefly within the partially-shadowed region (V_{sun} = 0.5) of the traverse mid-way between the Activity Points 2 and 3 (notation 2 on Figure 5.22). It then decreases before increasing once again at a much higher rate when the astronaut enters full sunlight (notation 3 on Figure 5.22). In both cases, the increase occurs because the thermal equilibrium temperature is above the external space suit temperature. However, the rate is different because of the magnitude of this difference. At the beginning of the partially-shadowed area, the difference between the thermal equilibrium temperature (237.3 K) to the fourth power and the external space suit temperature (191.1) to the fourth power is $1.04 \cdot 10^9$. The same difference at the beginning of the region of complete sunlight is an order of magnitude greater ($331.6^4 - 193.4^4 = 1.07 \cdot 10^{10}$).

The space suit's temperature reaches a maximum value of 311 K, which occurs 6 hours and 47 minutes into the traverse. Whenever the external space suit temperature is greater than 300 K – the constant internal space suit temperature – the external heat is conduced through the wall of the space suit. These stages are marked on Figure 5.18 (notation 5).

5.1.2 EVA 2 – Traverse Entirely in Sunlight

The second example traverse, called EVA 2, represents an extreme case of the astronaut thermal loading – when the traverse is entirely in sunlight. EVA 2 occurs at the very left of the 120 km by 120 km submap, as shown in Figure 5.25. Figure 5.26 shows a closer view of EVA 2, which is surrounded by the yellow box in Figure 5.25.

5. Example Traverses

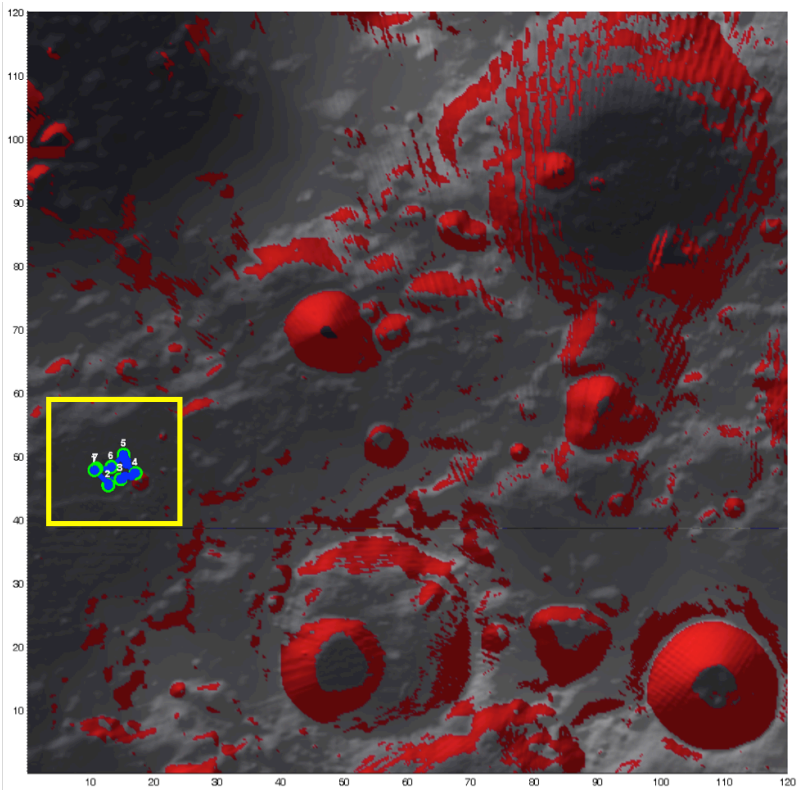


Figure 5.25. Location of EVA 2 on 120 km by 120 km submap

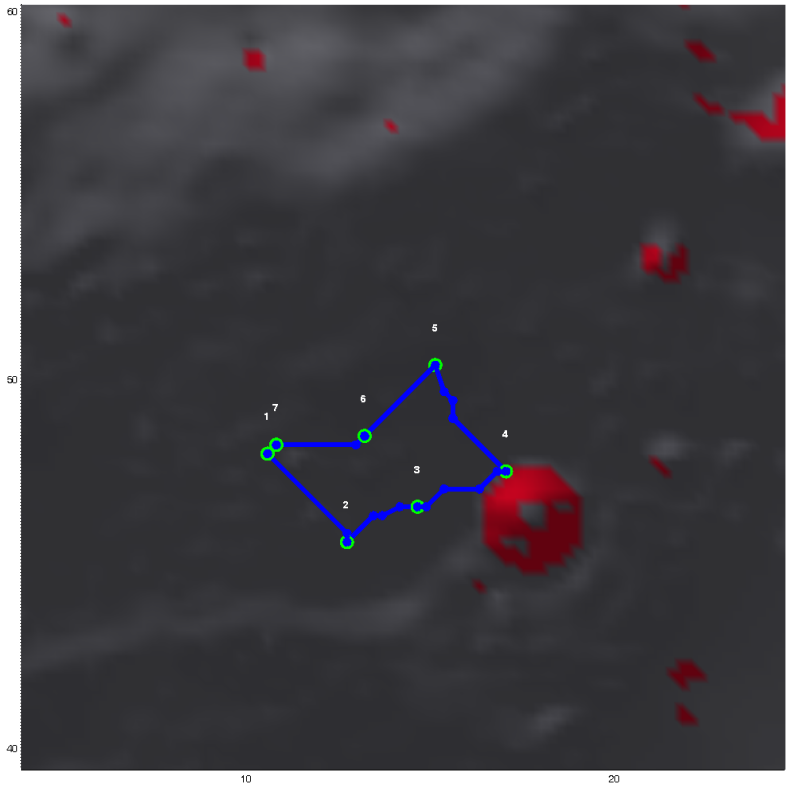


Figure 5.26. Close-up view of EVA 2 on 20 km by 20 km map

This traverse begins at the habitat (Activity Point 1), and then travels to a small crater located approximately 6.5 km away. After reaching the crater, the astronaut returns back to the habitat by a different route. Along the way, the astronaut visits a number of additional Activity Points where other scientific observations are desired. This traverse is relatively flat, as the difference in elevation between the lowest and highest points is only 99.4 meters.

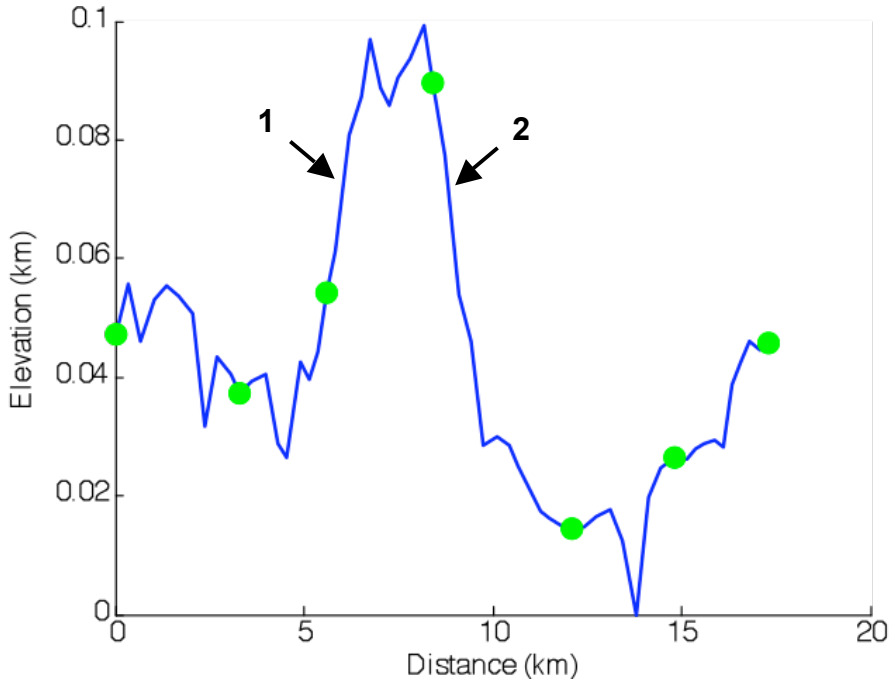


Figure 5.27. Elevation profile of EVA 2

It is important to note that while the elevation profile in Figure 5.27 looks quite steep and jagged, the horizontal scale is two orders of magnitude greater than the vertical scale. In reality, the maximum positive slope encountered is 3.3° (notation 1 on Figure 5.27) the maximum negative slope encountered is -3.9° (notation 2), and the average slope is only 0.03°.

5.1.2.1 Traverse Distance, Time, and Metabolic Cost for EVA 2

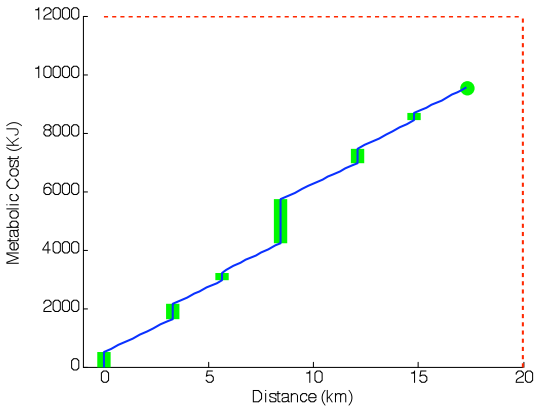
A major difference between EVA 2 and EVA 1 is that the astronaut stops at some of the Activity Points and performs scientific activities. The amount of time spent at each Activity Point can easily be specified through the 3D mapping interface, as discussed in Section 3.3.3. Table 5.3 shows the time spent at each Activity Point in EVA 2.

Table 5.3. Time spent at Activity Points in EVA 2

Activity Point	Time (min)
1	30
2	30
3	15
4	90
5	30
6	15
7	0

When determining the shadowing and related thermal parameters, SEXTANT breaks the time spent at each Activity Point into 1-minute long stages**.** When an astronaut remains stationary at any Activity Point, he is assumed to have a metabolic rate of 280 W (Waligora and Horrigan 1975). This was the average metabolic rate of the Apollo astronauts performing scientific activities on the lunar surface.

Figure 5.28 and Figure 5.29 show the metabolic cost of EVA 2 with respect to the traverse distance and time.



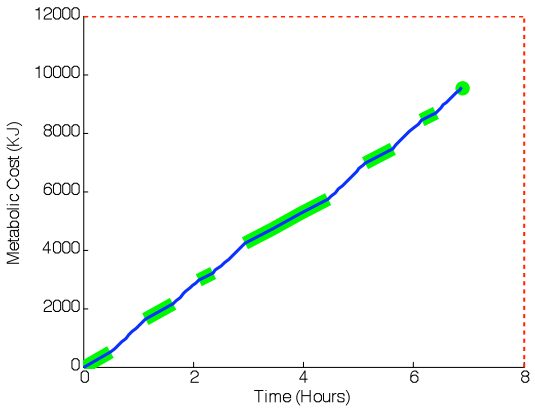


Figure 5.28 (left). Cumulative metabolic cost of EVA 2 with respect to distance

Figure 5.29 (right). Cumulative metabolic cost of EVA 2 with respect to time

The green regions in both Figure 5.28 and Figure 5.29 represent when the astronaut is working at the Activity Points. As was stated above, the metabolic cost of a traverse increases as the astronaut works at an Activity Point. However, the traverse distance does not change, as the astronaut is considered to be "stationary" for that time period (although he is certainly working and moving around locally). This can be seen in Figure 5.28, as the blue line is completely vertical at the Activity Points. On the other hand, the traverse time does increase while the

astronaut is working at an Activity Point. Because the metabolic rate is a constant value for all Activity Points, the slope of the blue line in Figure 5.29 is constant at all Activity Points.

Table 5.4 summarizes the cumulative distance, time, and metabolic cost of the astronaut when arriving and leaving each Activity Point.

Table 5.4. Cumulative traverse metrics at EVA 2 Activity Points

Activity Point	Cumulative Distance (km)		Cumulative Time (hr:min)		Cumulative Metabolic Cost (kJ)	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
1	0	0	0	0:30	0	504
2	3.29	3.29	1:08	1:38	1644.3	2148.3
3	5.61	5.61	2:06	2:21	2966.9	3218.9
4	8.41	8.41	2:57	4:27	4223.2	5735.2
5	12.09	12.09	5:07	5:37	6968.8	7472.8
6	14.80	14.80	6:09	6:24	8423.5	8675.5
7	17.30	----	6:52	----	9547.5	----

5.1.2.2 Traverse Shadowing and Thermal Metrics for EVA 2

As with EVA 1, the sun illumination is calculated for EVA 2 with a time step of 120 minutes. The traverse shadowing can be seen in Figure 5.30.

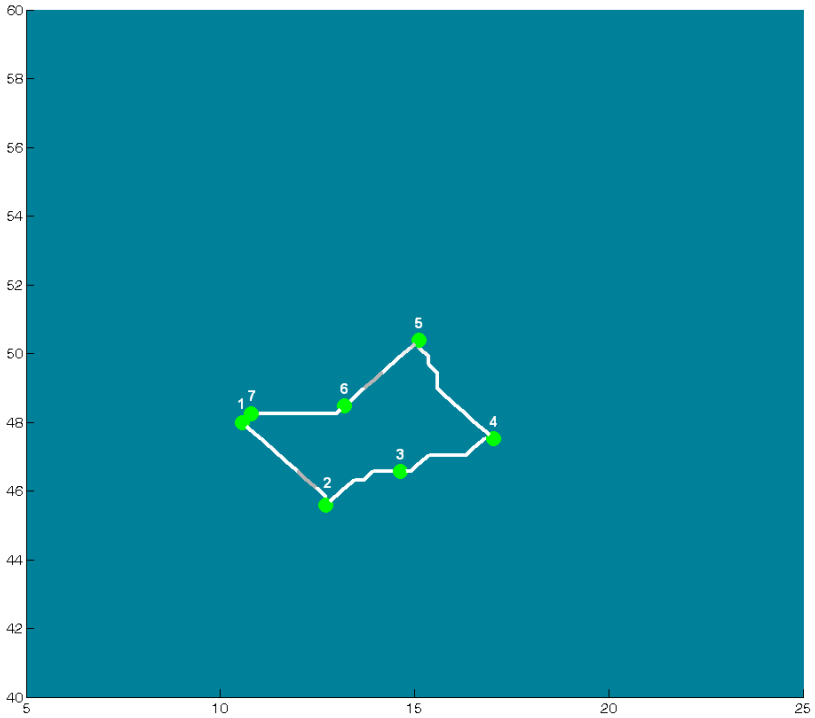
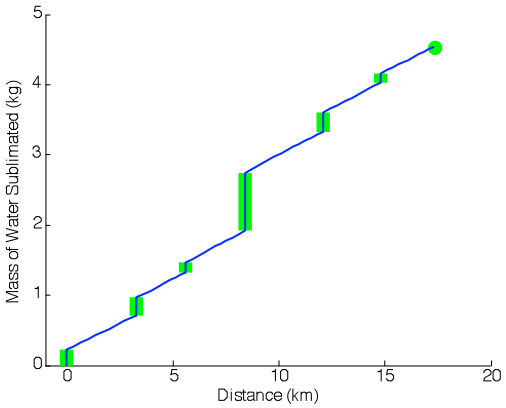


Figure 5.30. Shadowing along EVA 2

Figure 5.30 shows that most of the traverse is in complete sunlight, as desired. While there are some points of the traverse with partial shadowing, the minimum V_{sun} anywhere along the entire traverse is 0.70. With these conditions, it is expected that no heating will be required, and ice will have to be sublimated away for cooling. Figure 5.31 through Figure 5.38 show that this is indeed the case.



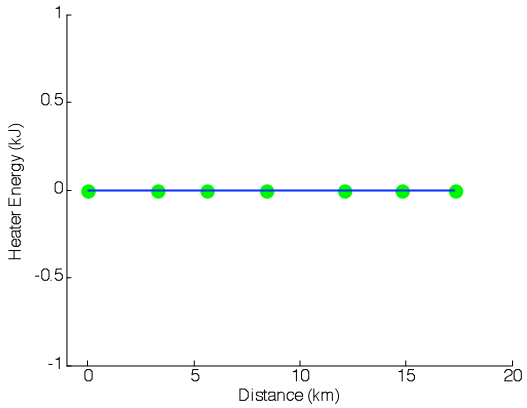
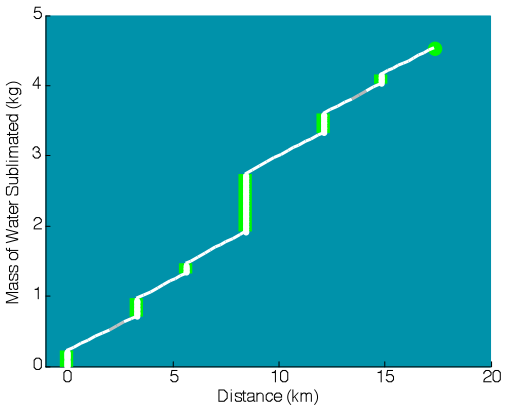


Figure 5.31 (left). Mass of sublimated water during EVA 2 with respect to distance

Figure 5.32 (right). Heater energy during EVA 2 with respect to distance



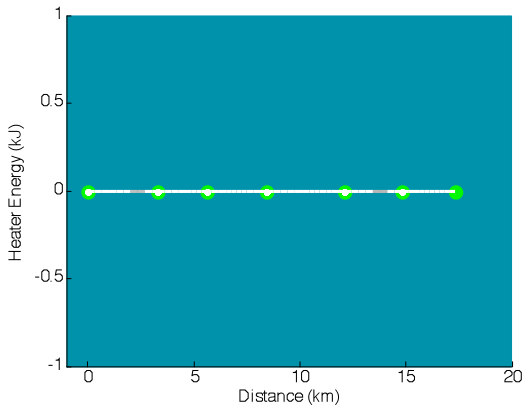


Figure 5.33 (left). Mass of sublimated water during EVA 2 with respect to distance, with shadowing

Figure 5.34 (right). Heater energy during EVA 2 with respect to distance, with shadowing

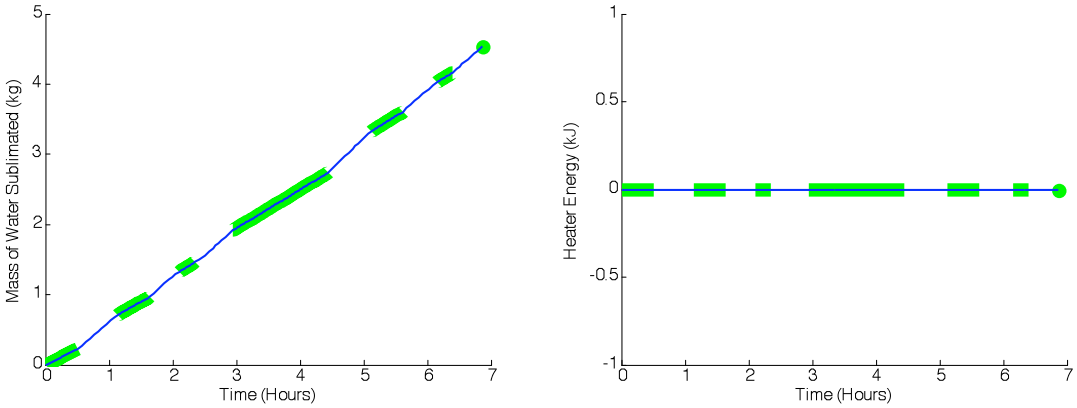


Figure 5.35 (left). Mass of sublimated water during EVA 2 with respect to time

Figure 5.36 (right). Heater energy during EVA 2 with respect to time

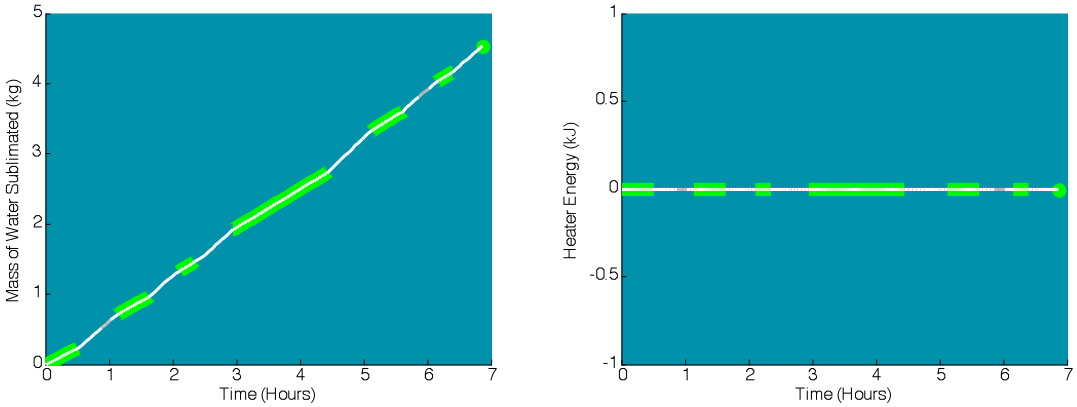


Figure 5.37 (left). Mass of sublimated water during EVA 2 with respect to time, with shadowing

Figure 5.38 (right). Heater energy during EVA 2 with respect to time, with shadowing

Figure 5.39 shows the components of heat flux as a function of time, and Figure 5.40 shows the heat from each source or sink for each stage. Once again, the electronics waste heat flux is a constant 50 W throughout the entire traverse. Because EVA 2 occurs on relatively flat ground, the metabolic rate and mechanical efficiency of astronaut motion are approximately the same for the entire traverse. As a result, the metabolic heat flux is relatively constant throughout the traverse when the astronaut is moving. The metabolic heat flux is also constant during the times when the astronaut is working at an Activity Point, at a value of 255.9 W.

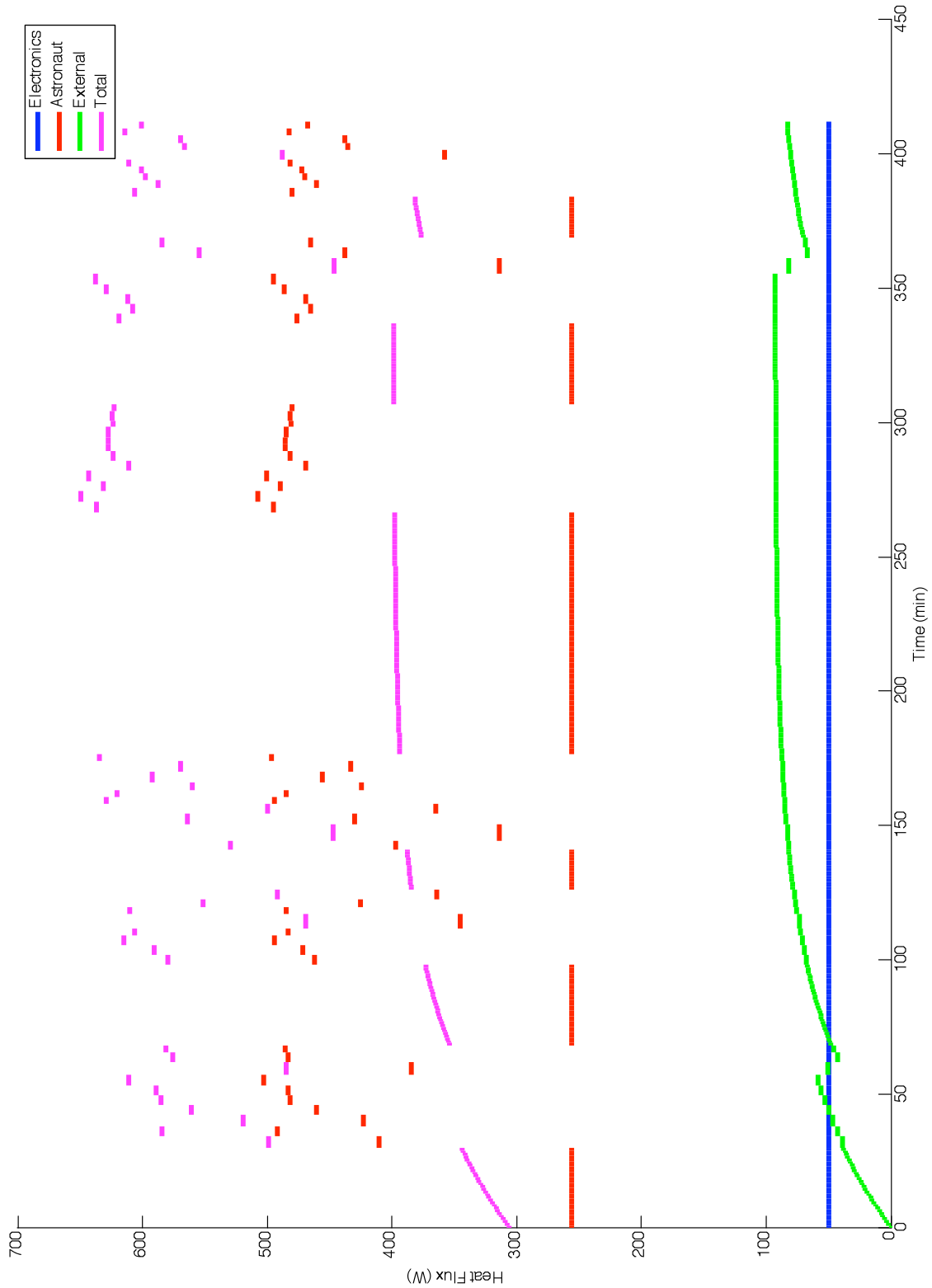


Figure 5.39. Heat flux with respect to time for EVA 2

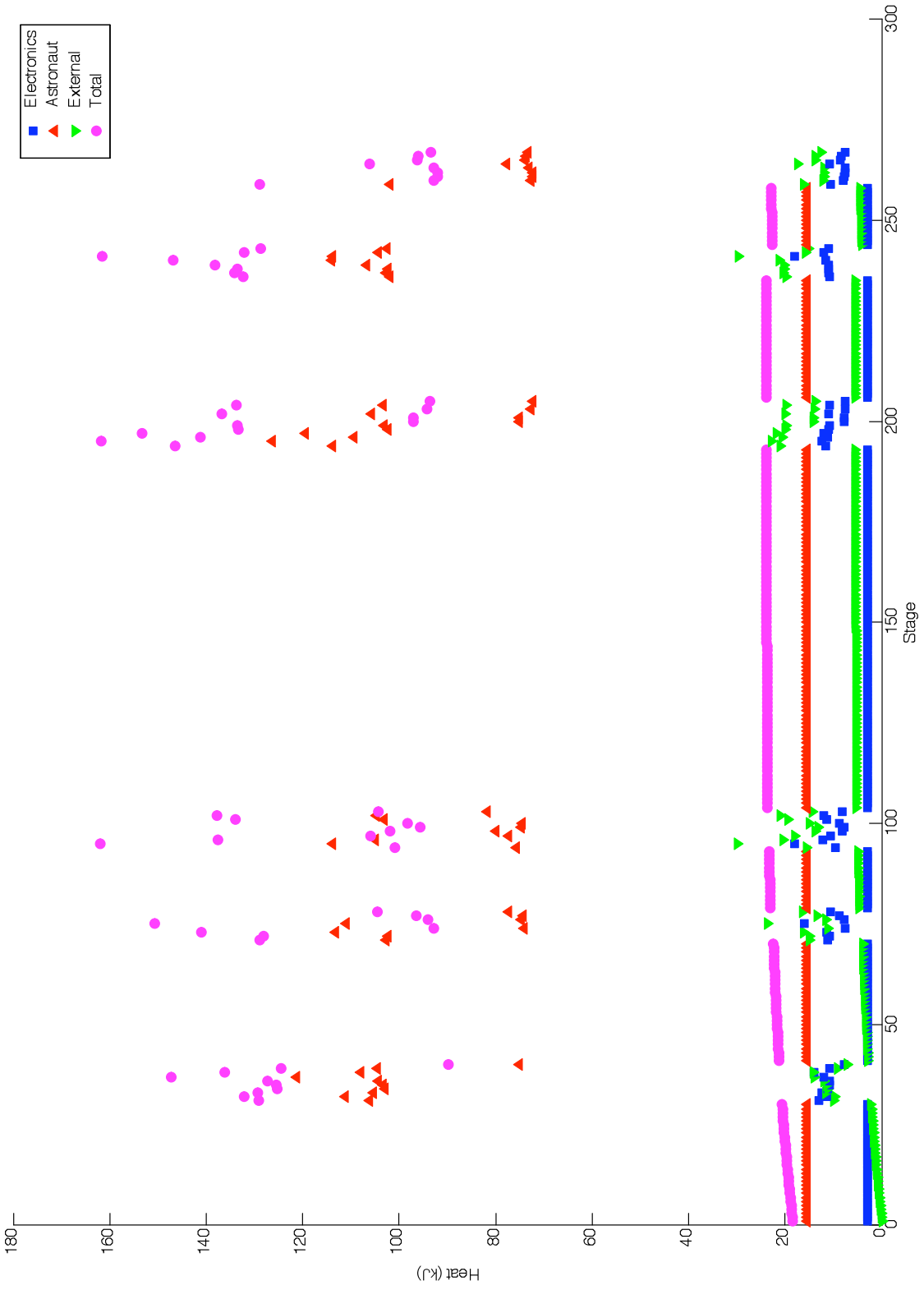
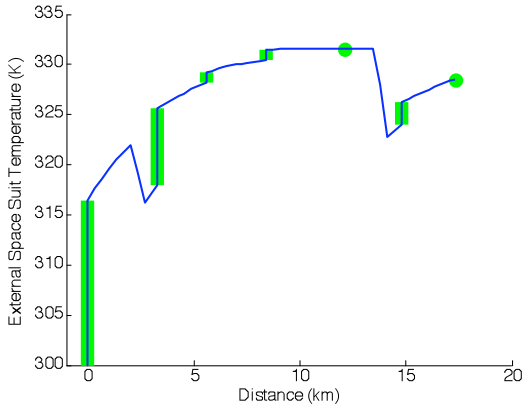


Figure 5.40. Heat flux with respect to traverse stages for EVA 2

5. Example Traverses

As with EVA 1, the external heat flux is determined by the external space suit temperature, which can be seen in Figure 5.42 through Figure 5.44.



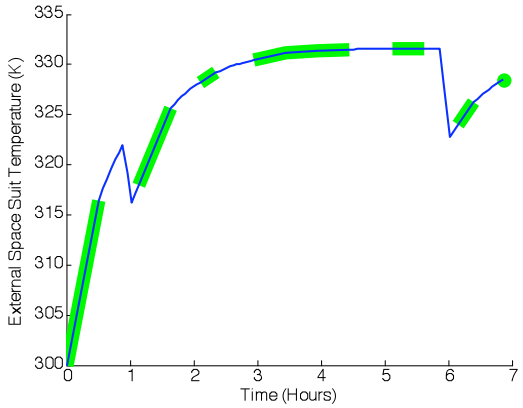
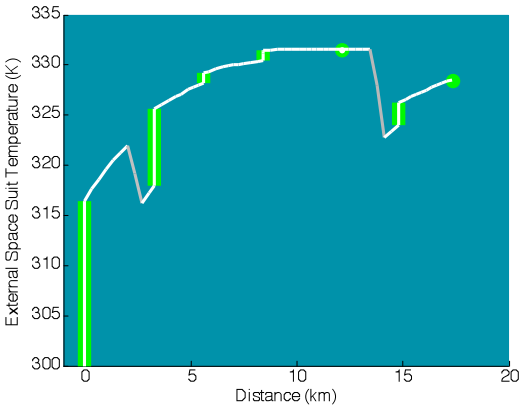


Figure 5.41 (left). External space suit temperature during EVA 2 with respect to distance

Figure 5.42 (right). External space suit temperature during EVA 2 with respect to time



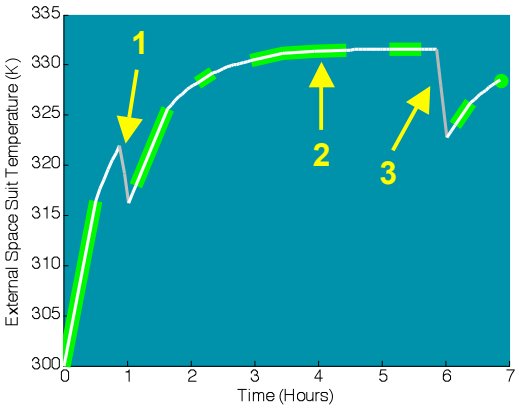


Figure 5.43 (left). External space suit temperature during EVA 2 with respect to distance, with shadowing

Figure 5.44 (right). External space suit temperature during EVA 2 with respect to time, with shadowing

The traverse begins in complete sunlight, where the thermal equilibrium temperature is 331.6 K. The space suit temperature at this point is 300 K, and so it begins to increase. After 53 minutes, the astronaut enters a region of partial shadow (notation 1 on Figure 5.44). While the V_{sun} is only 0.70, the thermal equilibrium temperatures is 274.8 K. The space suit exterior is warmer than this (319.8 K), and so it begins to cool down. This partially-shadowed stage only lasts for 8.8 minutes, after which the astronaut returns to complete sunlight. The space suit temperature then begins to increase once again. The astronaut remains in complete sunlight for

long enough that the external space suit temperature reaches the thermal equilibrium temperature of 331.6 K. This occurs approximately 3 hours and 55 minutes after the traverse begins, when the astronaut is at Activity Point 4 (notation 2 on Figure 5.44). At this point, the external space suit temperature stays constant, as does the conductive heat transfer through the space suit wall. At 5 hours and 50 minutes into the traverse, the astronaut enters another region of partial shadow (V_{sun} = 0.80), and the thermal equilibrium temperature decreases to 293.6 K. The external space suit temperature follows suit (notation 3 on Figure 5.44). Finally, to end the traverse, the astronaut re-enters complete sunlight, and the space suit temperature begins to increase once more.

EVA 2, an extreme thermal case with mostly complete sunlight, requires 4.5 kg of water to replenish the ice sublimated away for cooling. The Apollo space suits had 5 kg of water for this purpose (NASA 1972), meaning that the space suit thermal control system modeled in SEXTANT is well-designed for this constraint. However, it is plausible that water will be a scarcer commodity for future lunar explorations. These missions will last months, and water will either have to be brought from Earth or produced on the lunar surface. Both of these are costly procedures. Therefore, the water requirement of 4.5 kg for EVA 2 may be too high for future explorations. The traverse could be re-planned to be shorter in distance or time, which would both decrease the amount of cooling required. Or, if a shadowed area was found to be nearby, the astronaut could stop here and allow the space suit exterior to cool off before continuing the traverse. This would lower the external heat flux into the space suit, and therefore the necessary cooling.

5.1.3 EVA 3 – Traverse Entirely in Shadow

The third and final example astronaut traverse, EVA 3, represents another extreme case of the astronaut thermal loading – when the entire traverse is in complete shadow. Figure 5.45 shows the traverse on the large 120 km by 120 km submap, and Figure 5.46 shows a close-up of the path.

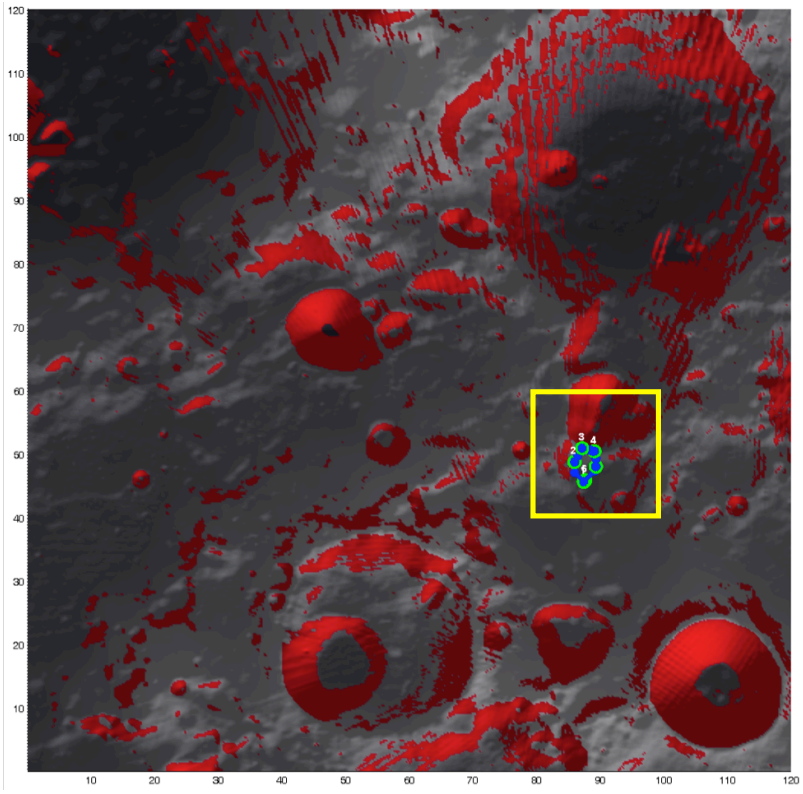


Figure 5.45. Location of EVA 3 on 120 km by 120 km submap

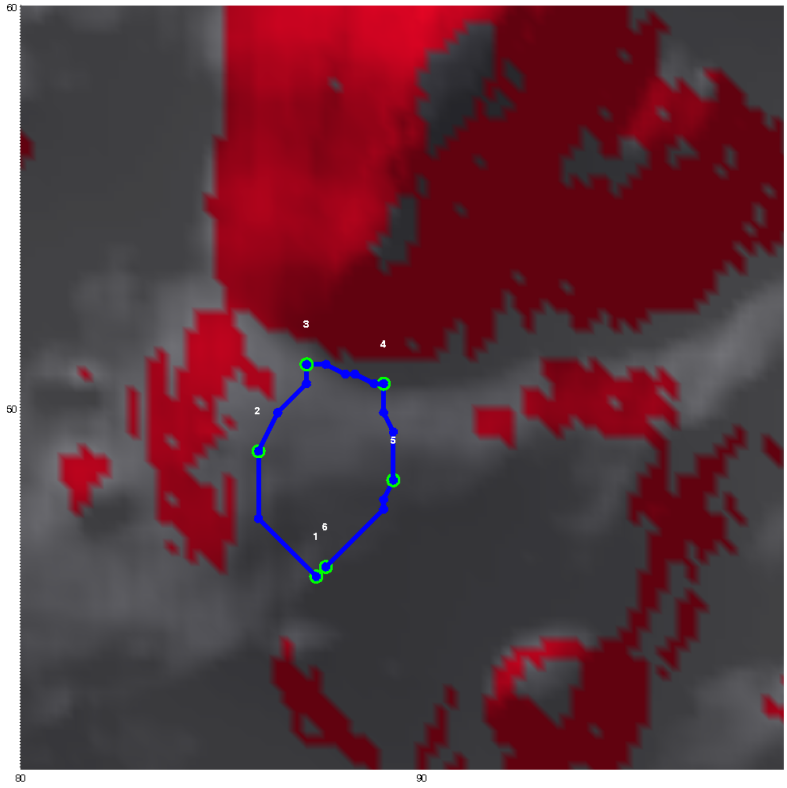


Figure 5.46. Close-up view of EVA 3 on 20 km by 20 km map

The change in elevation for EVA 3 is greater than that for EVA 2, but not as much as in EVA 1. As Figure 5.47 shows, the maximum difference in elevation is 578 m:

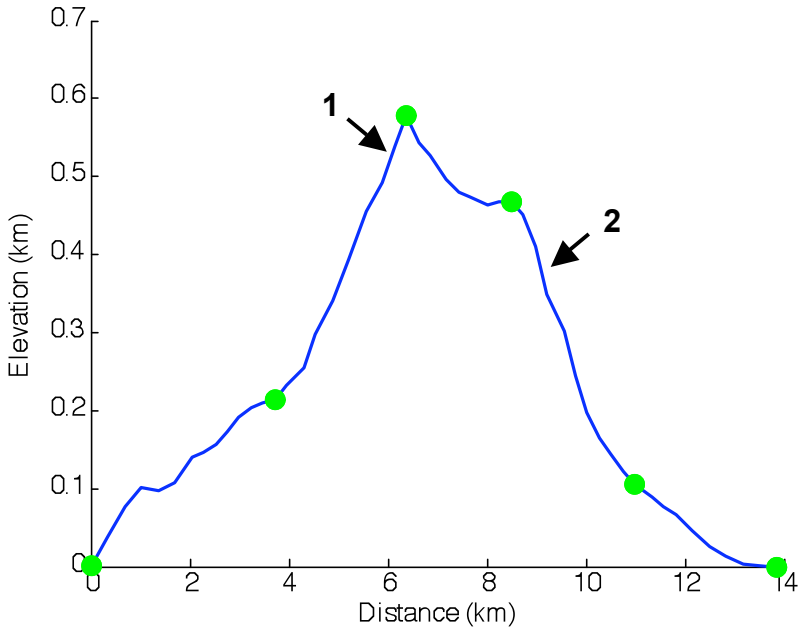


Figure 5.47. Elevation profile of EVA 3

The elevation profile of the traverse is roughly symmetric about Activity Point 3, giving an average slope of -0.27°. The maximum positive slope is 11.4° (notation 1 on Figure 5.47), and the maximum negative slope is -14.3° (notation 2).

5.1.3.1 Traverse Distance, Time, and Metabolic Cost for EVA 3

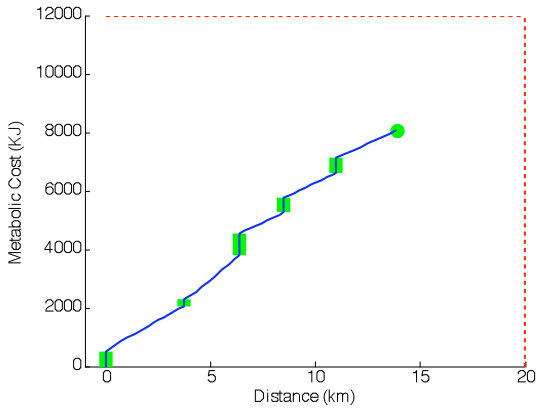
As in EVA 2, the astronaut remains at each of the Activity Points to perform scientific activities. Table 5.5 details the amount of time spent at each of the Activity Points:

Table 5.5. Time spent at Activity Points in EVA 3

Activity Point	Time (min)
1	30
2	15
3	45
4	30
5	30
6	0

5. Example Traverses

Figure 5.48 and Figure 5.49 show the metabolic cost of EVA 3 with respect to distance and time, and Table 5.6 shows the cumulative traverse metrics for the Activity Points along the path.



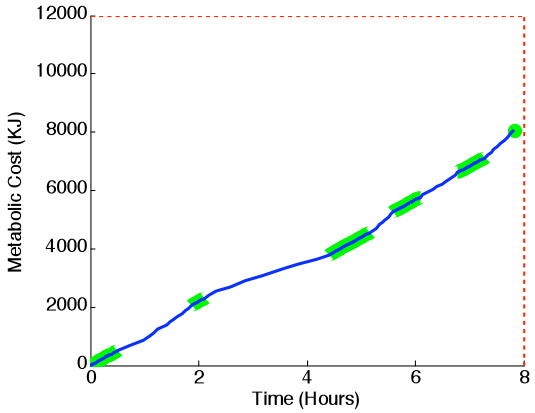


Figure 5.48 (left). Cumulative metabolic cost of EVA 3 with respect to distance

Figure 5.49 (right). Cumulative metabolic cost of EVA 3 with respect to time

Table 5.6. Cumulative traverse metrics at EVA 3 Activity Points

Activity Point	Cumulative Distance (km)		Cumulative Time (H:MM)		Cumulative Metabolic Cost (kJ)	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
1	0	0	0:00	0:30	0	504
2	3.72	3.72	1:52	2:07	2050.6	2302.6
3	6.37	6.37	4:25	5:10	3800.2	4556.2
4	8.49	8.49	5:35	6:05	5265.2	5769.2
5	10.99	10.99	6:47	7:17	6619.0	7123.0
6	13.85	----	7:49	----	8071.0	----

The traverse has been designed to be within the distance, time, and metabolic cost constraints given in Table 5.1, and this is indeed true. The closest to any constraint is the traverse time, which only has a margin of 11 minutes. Even with the extra metabolic cost of working at the Activity Points, the metabolic cost of the traverse is only 8071 kJ, well within limits.

5.1.3.2 Traverse Shadowing and Thermal Metrics for EVA 3

Once the initial path has been plotted, the shadowing can be determined. This is displayed in Figure 5.50.

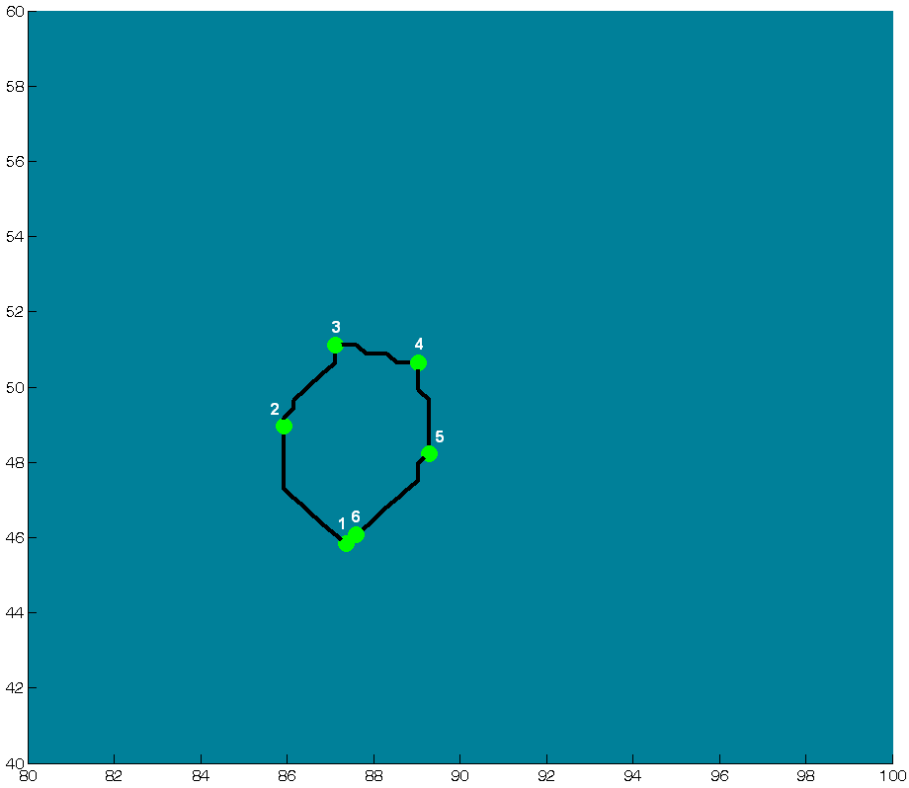


Figure 5.50. Shadowing along EVA 3

As was stated earlier, the astronaut is in complete shadowing during the entire traverse. Figure 5.51 through Figure 5.58 show the amount of water required for sublimation and heater power used during EVA 3.

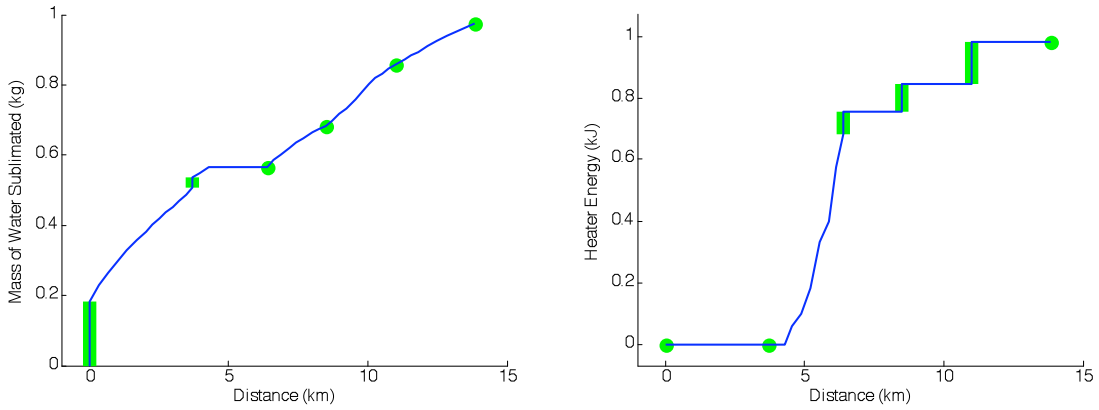
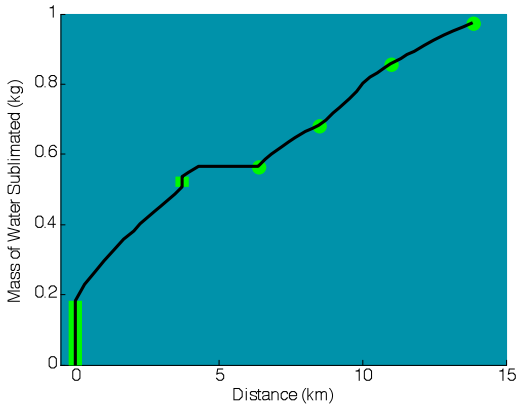


Figure 5.51 (left). Mass of sublimated water during EVA 3 with respect to distance

Figure 5.52 (right). Heater energy during EVA 3 with respect to distance



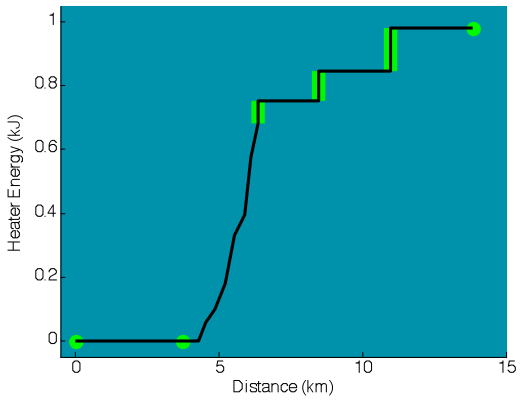
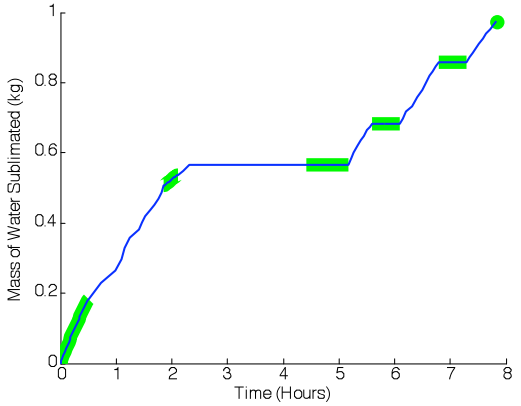


Figure 5.53 (left). Mass of sublimated water during EVA 3 with respect to distance, with shadowing

Figure 5.54 (right). Heater energy during EVA 3 with respect to distance, with shadowing



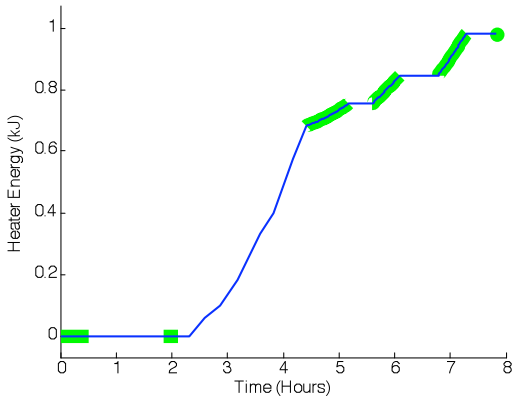
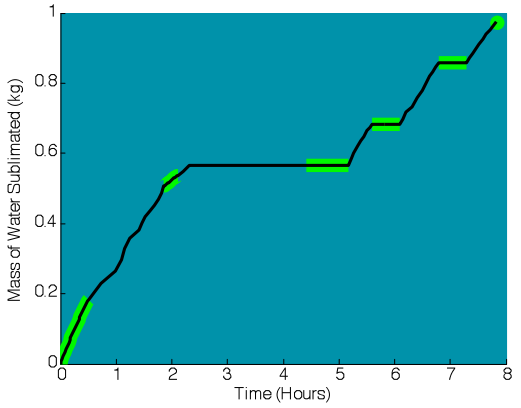


Figure 5.55 (left). Mass of sublimated water during EVA 3 with respect to time

Figure 5.56 (right). Heater energy during EVA 3 with respect to time



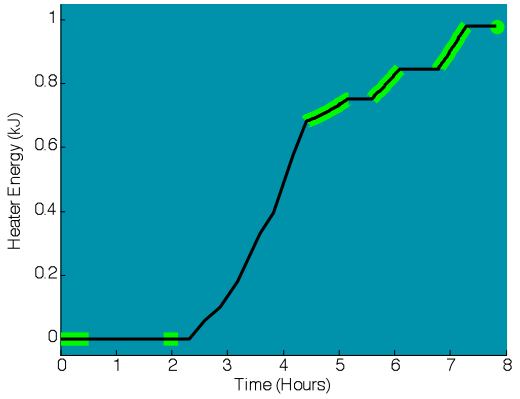


Figure 5.57 (left). Mass of sublimated water during EVA 3 with respect to time, with shadowing

Figure 5.58 (right). Heater energy during EVA 3 with respect to time, with shadowing

Contrary to what one might think, EVA 3 – and any traverse in complete shadow – does not require only heating. Sublimation cooling is still necessary during some stages of the traverse. The reason for this can be explained through Figure 5.59 and Figure 5.60, which show the stage heat fluxes with respect to time and the stage heat into or out of the space suit. In the beginning of the traverse, the electronics waste heat flux and astronaut metabolic heat flux are enough to overwhelm the minimal heat loss to the environment. Therefore, the net heat transfer is still into the space suit, and cooling is required. Eventually, at 2 hours and 19 minutes after the start of EVA 3, the total heat flux becomes negative, and heat is lost from the space suit to the environment. This occurs because the astronaut is traveling over a steep portion of the traverse, between Activity Points 2 and 3. This is the area right before the "peak" on the plot of elevation in Figure 5.47. As detailed in Table 3.1, the astronaut's velocity decreases as the terrain slope increases. As a result, the lengths of these stages are much longer than any others in the traverse. Integrated over their entire length, these stages release generally the same amount of heat as the other movement stages (Figure 5.60). However, because of their lengths they have a lower heat flux. Therefore, the heat loss to the environment overwhelms the internal sources of heat during the stage, and the total heat transfer is out of the space suit. Heating is therefore required.

During subsequent movements by the astronaut, the metabolic heat flux and the electronics waste heat combined are greater than the external heat flux, meaning that the net heat transfer is to the space suit. Sublimation cooling is used to remove this excess heat, which can be seen in the left column of figures above (Figure 5.51, Figure 5.53, Figure 5.55, and Figure 5.57). Whenever the astronaut is working at an Activity Point (notations 1, 2, and 3 on Figure 5.59), he is not using as much energy as while walking. The space suit exterior is also very cold by this point, and the conduction heat transfer through the space suit to the environment dominates the electronics waste heat flux and the metabolic heat flux. Once again, heating is required to account for this heat loss.

At the beginning of the traverse, when the astronaut leaves the habitat, the external space suit temperature is 300 K. This is the same as the internal space suit temperature, and there is no heat conduction through the wall of the space suit. The external space suit temperature begins to decrease, which continues throughout the entire traverse. This can be seen in Figure 5.61 through Figure 5.64.

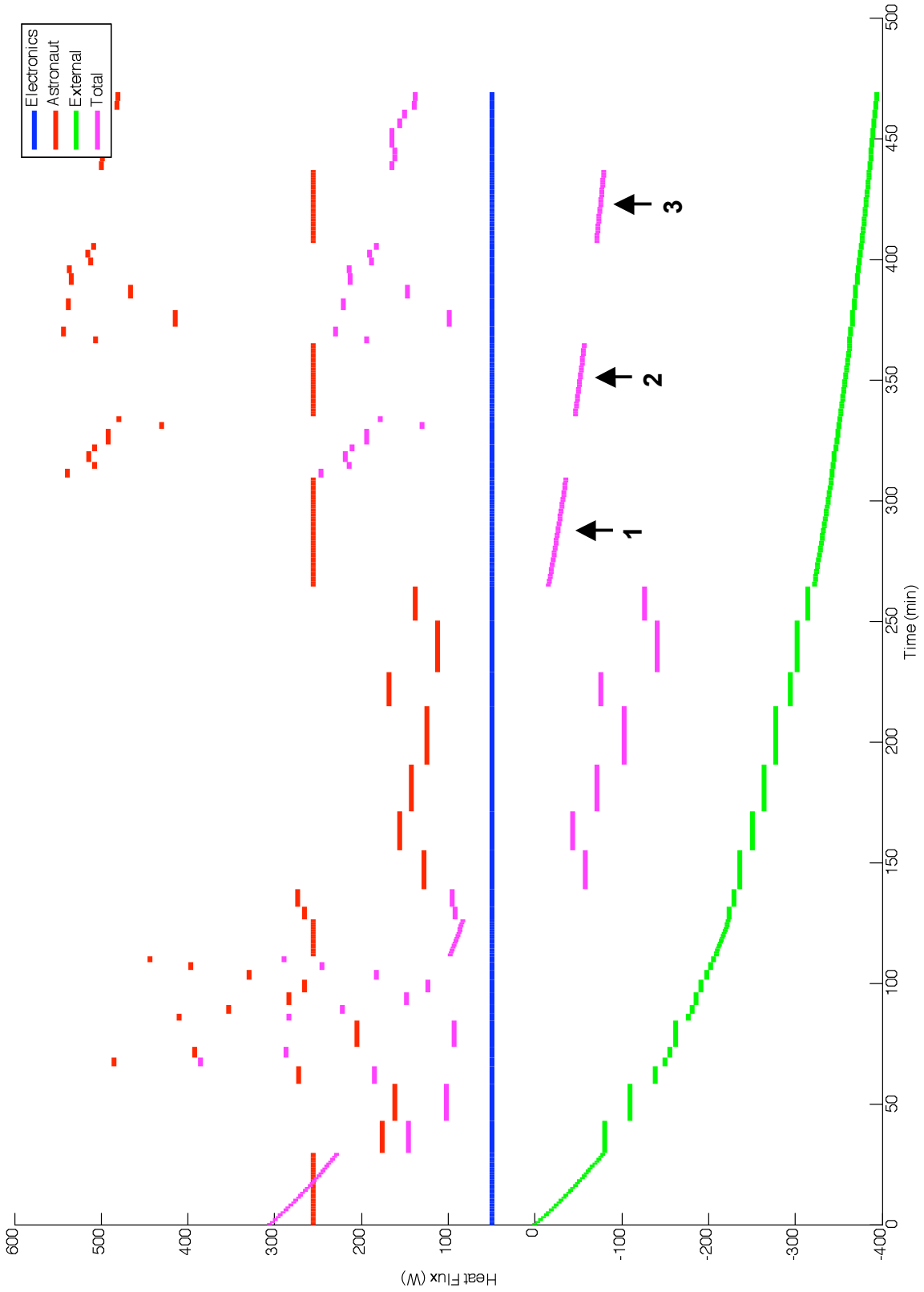


Figure 5.59. Heat flux with respect to time for EVA 3

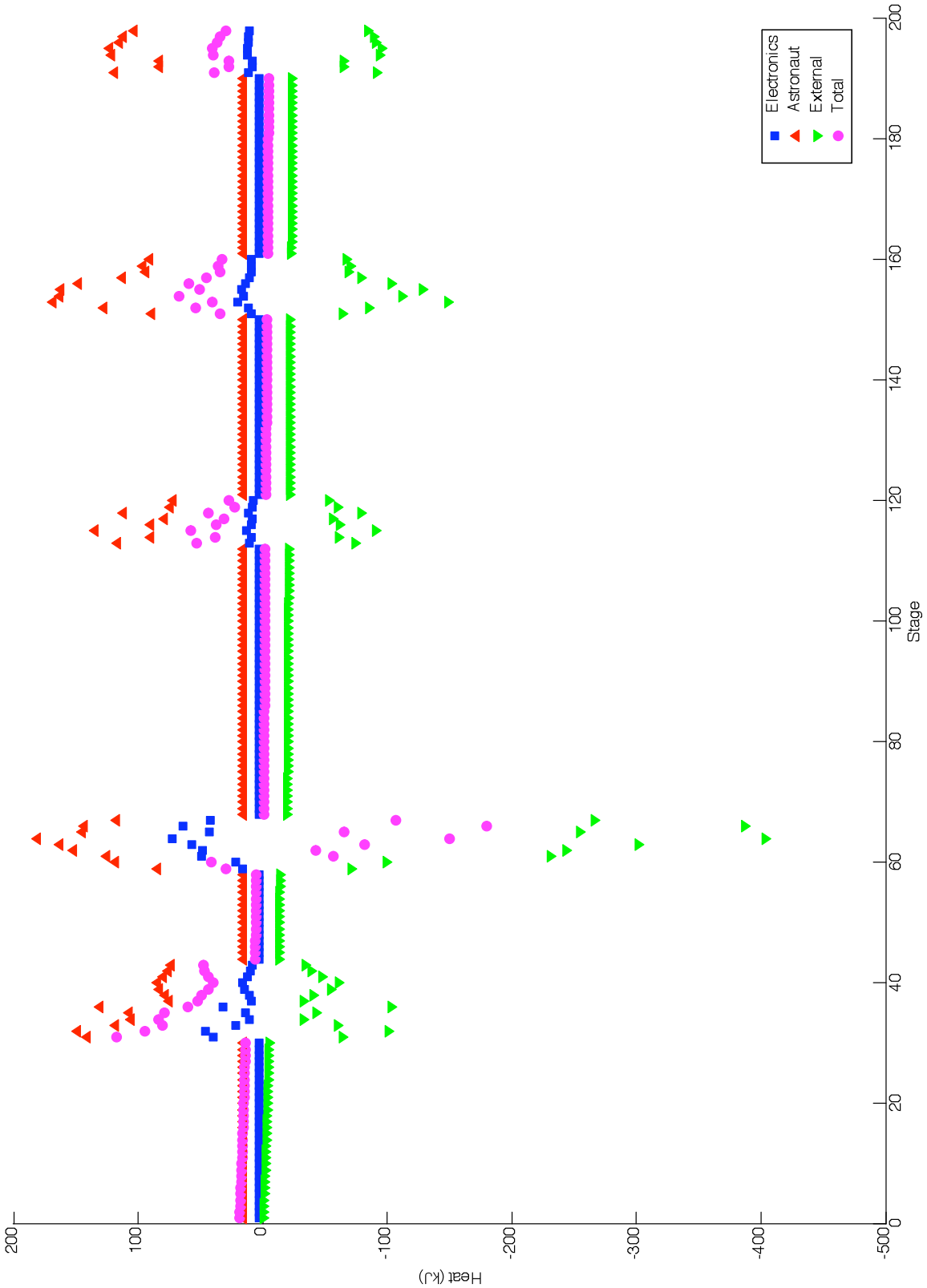
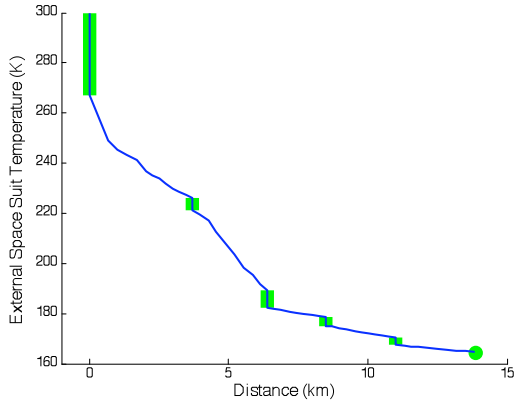


Figure 5.60. Heat flux with respect to traverse stages for EVA 3



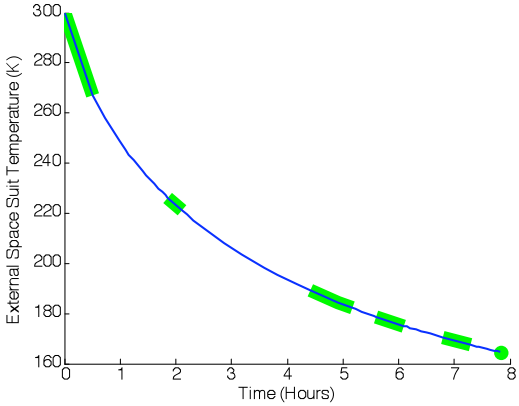
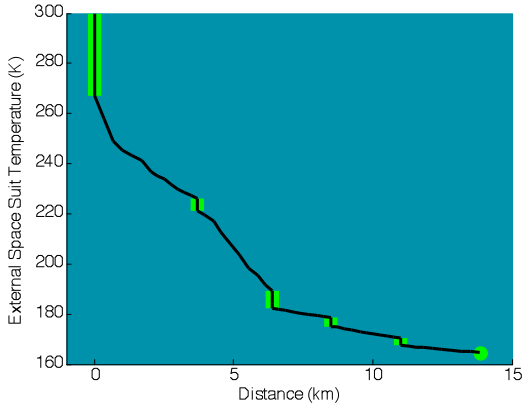


Figure 5.61 (left). External space suit temperature during EVA 3 with respect to distance

Figure 5.62 (right). External space suit temperature during EVA 3 with respect to time



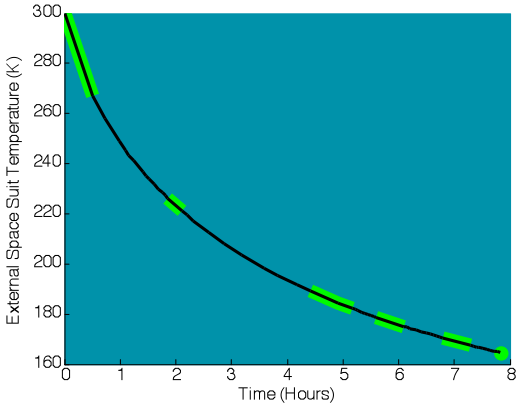


Figure 5.63 (left). External space suit temperature during EVA 3 with respect to distance, with shadowing

Figure 5.64 (right). External space suit temperature during EVA 3 with respect to time, with shadowing

Because the environmental conditions (V_{sun} and the lunar surface temperature) do not change throughout the entire traverse, the space suit temperature constantly decreases towards the thermal equilibrium temperature of 102.8 K. This is clear in Figure 5.62 and Figure 5.64, and the temperature curve with respect to time is a smooth curve. As expected, the rate of the temperature change is greater at the beginning of the traverse, when the difference between the external space suit temperature and the thermal equilibrium temperature is the greatest. At the end of the traverse the temperature is 165.9 K, which is 68% of the possible temperature drop.

5.2 Example Rover Traverse

5.2.1 Planning the Desired Traverse

For traverses with a transportation rover or robotic explorer, SEXTANT focuses on how the shadowing affects the rover's power system. The main metric calculated is the energy level of the rover's batteries. Because explorations of the lunar south pole will encounter large shadowed areas, it is important for astronauts to ensure that the rover's batteries have enough energy to complete a given traverse and return home. As stated in Section 4.4.1, the batteries can be recharged by the solar array if it is producing excess power. One example traverse is presented here that shows how a rover traverse can be planned and modified to fit within all power constraints. This traverse covers a large portion of the 120 km by 120 km submap in Figure 5.1.

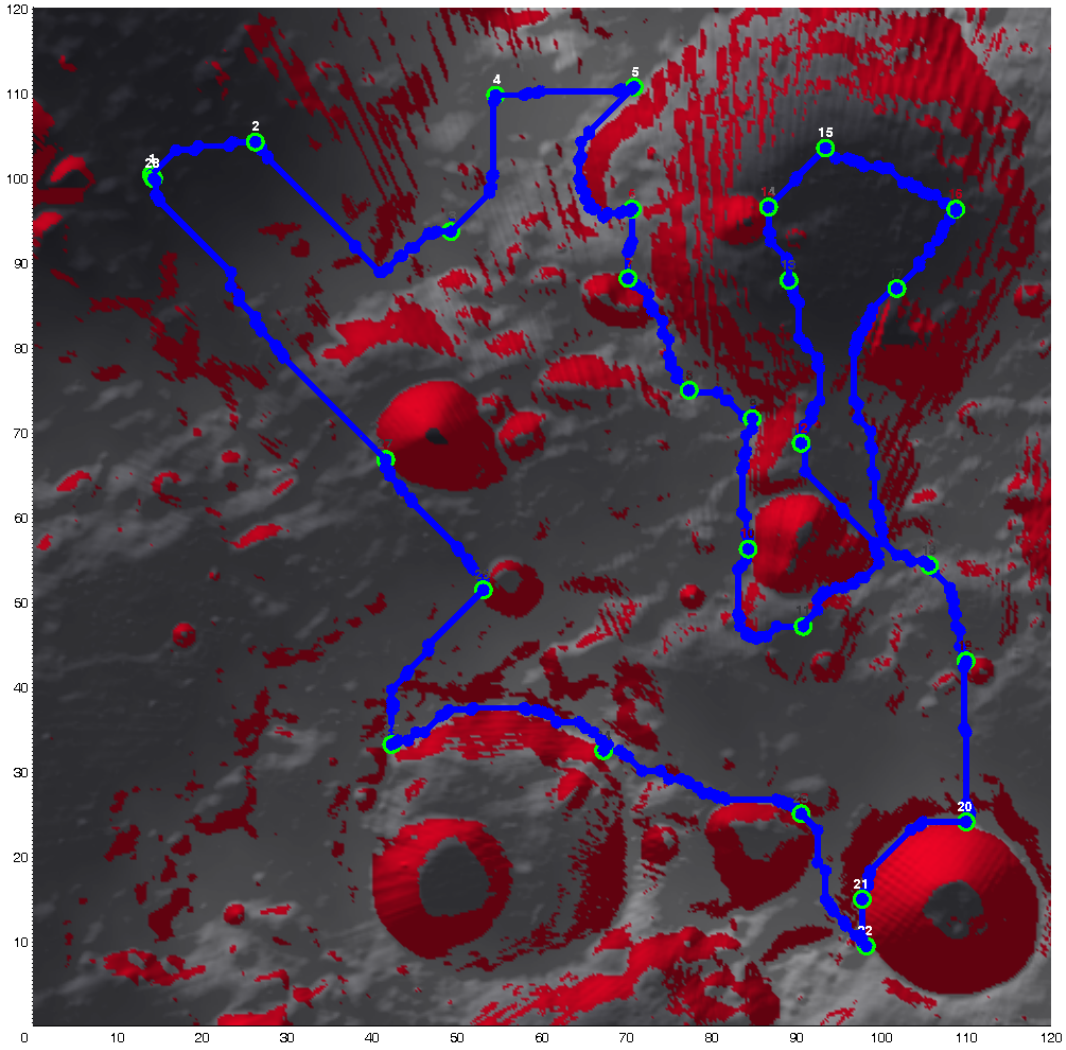


Figure 5.65. Location of rover traverse on 120 km by 120 km submap

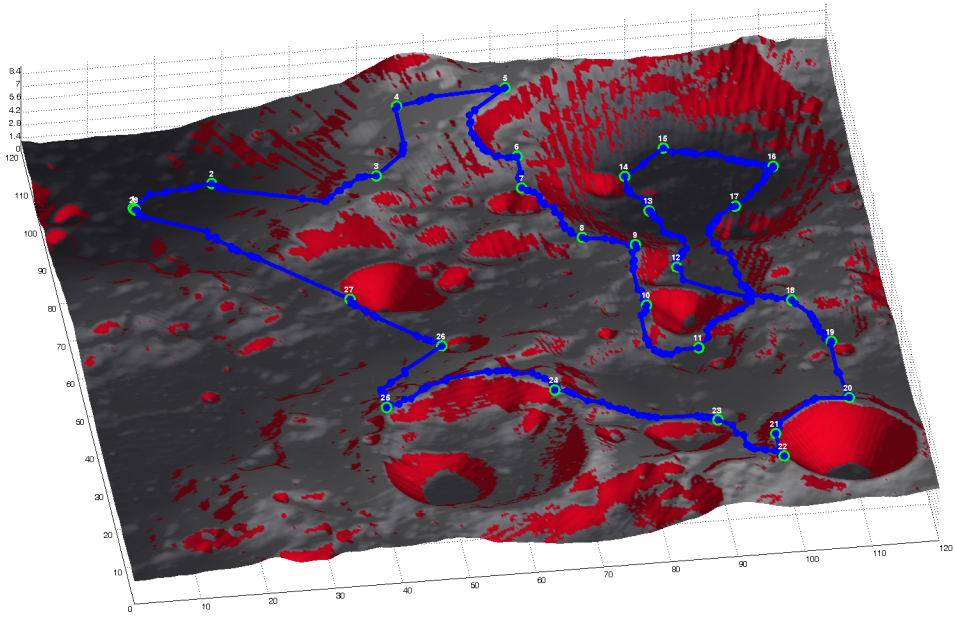


Figure 5.66. 3D view of rover traverse

The traverse originates from the habitat (Activity Point 1), in a low point in the upper-left corner of the map. The rover begins by climbing a hill, gaining 5.8 km of altitude between the habitat and Activity Point 5, as Figure 5.67 shows. (Note that this elevation profile is exaggerated - the the horizontal scale is two orders of magnitude greater than the vertical scale).

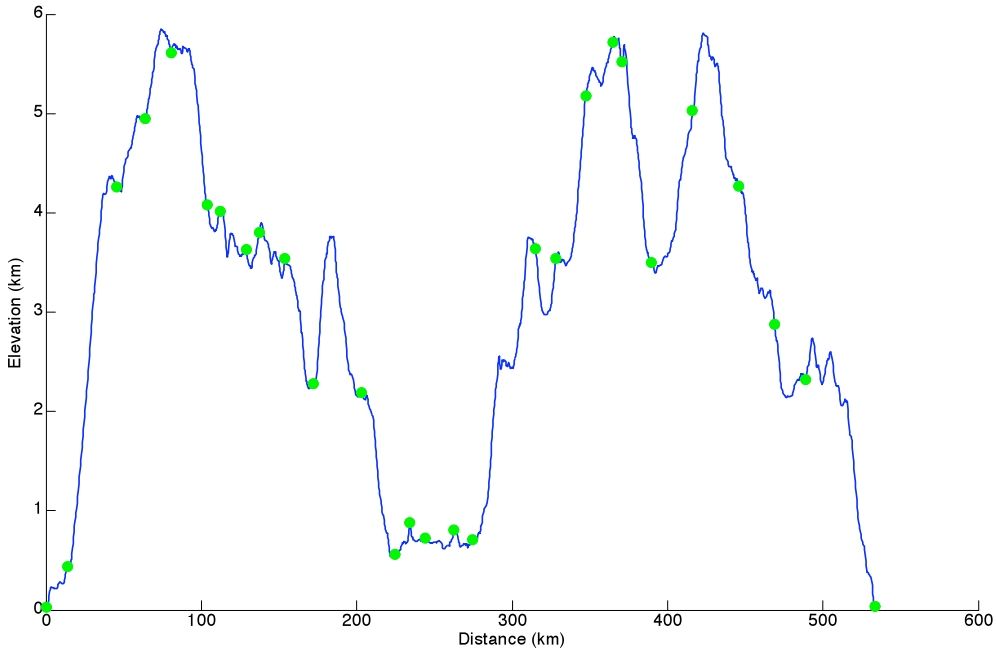


Figure 5.67. Elevation profile of rover traverse

Once the rover crests the hill, it travels downhill around the rim of Haworth Crater. The rover stops at Activity Points 6 and 7 for 30 minutes and 60 minutes, respectively, for scientific observations. The rover then continues around the crater rim, stopping at Activity Point 10 to examine the small, deep crater just below Haworth Crater on the map. The rover then travels down into Haworth Crater to explore its floor. Besides the habitat, this is the lowest point reached on the traverse (notation 1 on Figure 5.67). The rover stops for a total of 6 hours in this region, distributed amongst Activity Points 13 – 17. Exploring the bottom of craters is an important task, as these areas expose lower portions of the Moon's crust that are very scientifically interesting. The bottom of Haworth Crater is also a permanently-shadowed region, where there is the greatest chance of finding water ice or other volatiles (Mazarico et al. 2010) Once the exploration of the Haworth Crater floor is complete, the rover travels out of the crater and to the rim of Shackleton Crater. The rover spends 90 minutes at both Activity Points 20 and 22, and then travels back towards the habitat. Along the way, the rover stops at Activity Points 24 – 27 for varying amounts of time, in order to explore additional craters. Table 5.7 summarizes the time spent at each Activity Point.

Table 5.7. Time spent at Activity Points in rover traverse

Activity Point	Time (min)	Activity Point	Time (min)	Activity Point	Time (min)
1	60	11	0	21	0
2	0	12	0	22	90
3	0	13	30	23	0
4	0	14	45	24	60
5	0	15	60	25	30
6	30	16	180	26	15
7	60	17	30	27	15
8	0	18	30	28	0
9	0	19	0		
10	60	20	90		

All in all, this rover traverse covers 533.50 km, lasts 50 hours and 17 minutes (with stops), and uses 331.8 GJ of energy. As the traverse lasts over 50 hours, long-duration stops of 7 hours must be added to the traverse at reasonable times to allow the astronauts to sleep. Figure 5.68 and Figure 5.69 show the energy usage as a function of distance and time.

5. Example Traverses

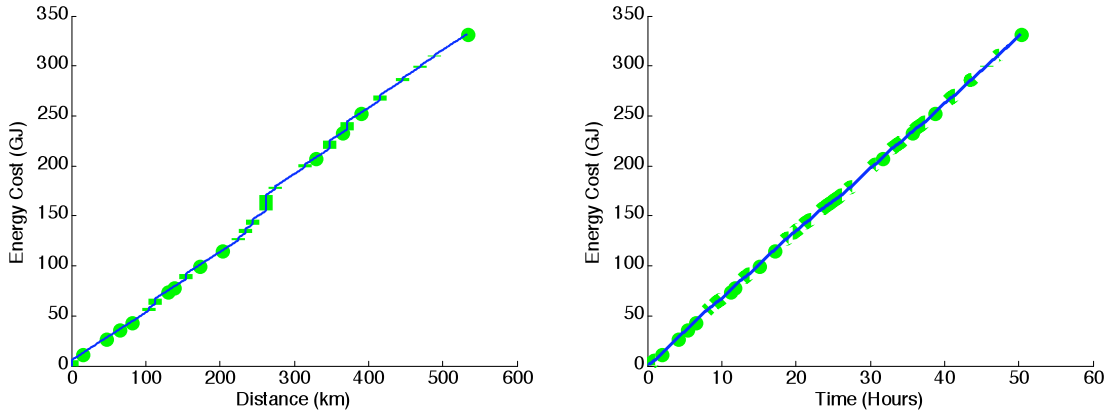


Figure 5.68 (left). Cumulative energy cost of rover traverse with respect to distance

Figure 5.69 (right). Cumulative energy cost of rover traverse with respect to time

The specific numbers for the cumulative distance, time, and metabolic cost when arriving and departing all of the Activity Points can be found in Appendix C.

Like the suited astronauts performing EVA, the rover also has constraints on how far and for how long it can travel during any traverse. These limits are 1000 km and 14 Earth days (Culbert 2009). There is no specific energy cost constraint, as the battery energy level constraint fully captures the energy usage. The distance and time of this example traverse, while long, are well within the stated constraints. As such, red dashed lines noting the constraints have been left off of Figure 5.68 and Figure 5.69. In order to check whether or not the energy requirements of the traverse are valid, the shadowing conditions must be analyzed.

Figure 5.70 shows the shadowing along the rover traverse. The traverse begins in shadow, but reaches an area of sunlight between Activity Points 2 and 5. After Activity Point 5, the rover spends most of its time in shadow as it circles Haworth Crater and then descends to the crater floor. There are brief periods of sunlight around Activity Points 9 and 10. More sunlight is encountered while on the rim of Shackleton Crater, around Activity Point 21. The portion of the traverse from Shackleton Crater back to the habitat has some sunlit portions, mainly between Activity Points 24 and 26. All in all, the rover spends 78.3% of its time in complete shadow, 8.2% in partial shadow, and 13.5% in complete sunlight. Because the rover spends so much time in complete shadow, this traverse will require a great deal of battery power. It will be essential to make sure that the rover battery is large enough so that it never runs out of energy.

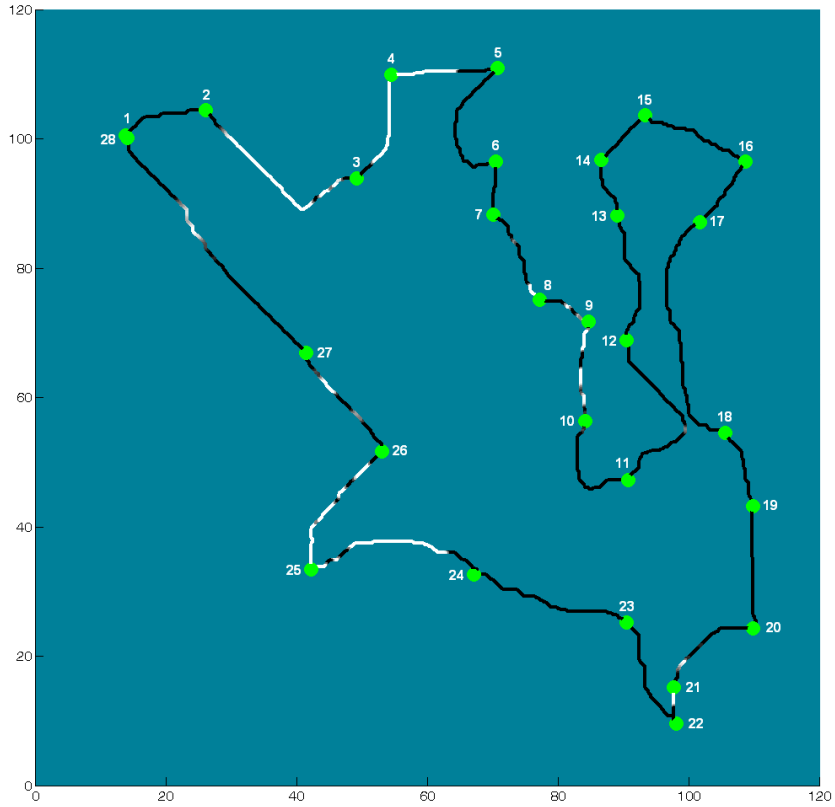


Figure 5.70. Shadowing along rover traverse

The rover's battery energy level is plotted as a function of both time and distance along the traverse in Figure 5.71 through Figure 5.74.

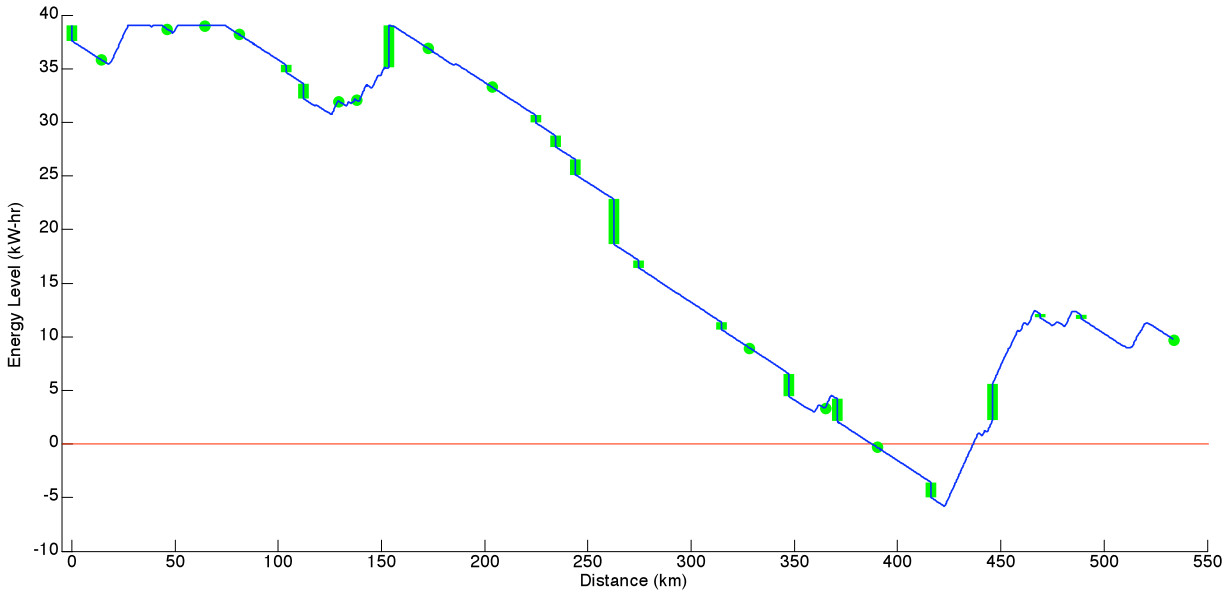


Figure 5.71. Battery energy level during rover traverse with respect to distance

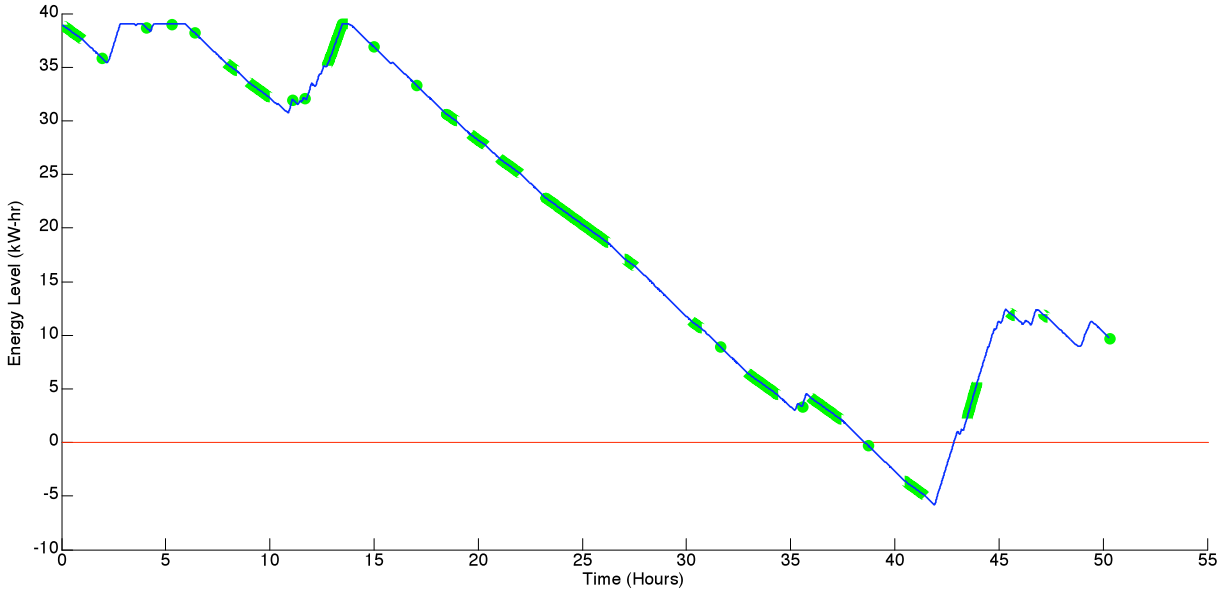


Figure 5.72. Battery energy level during rover traverse with respect to time

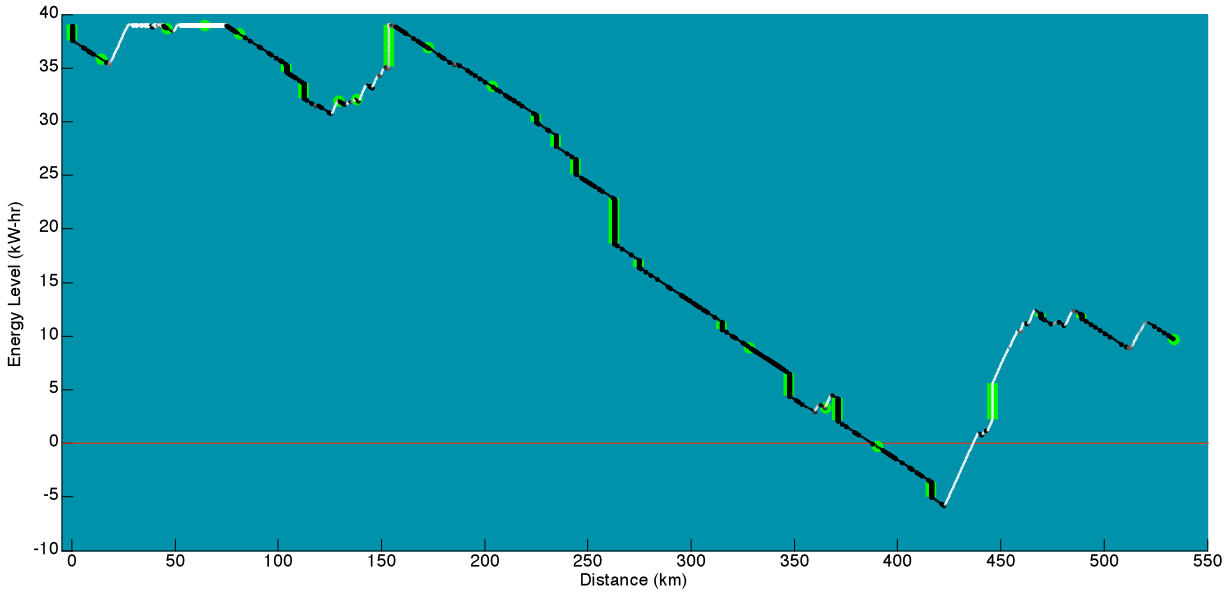


Figure 5.73. Battery energy level during rover traverse with respect to distance, with shadowing

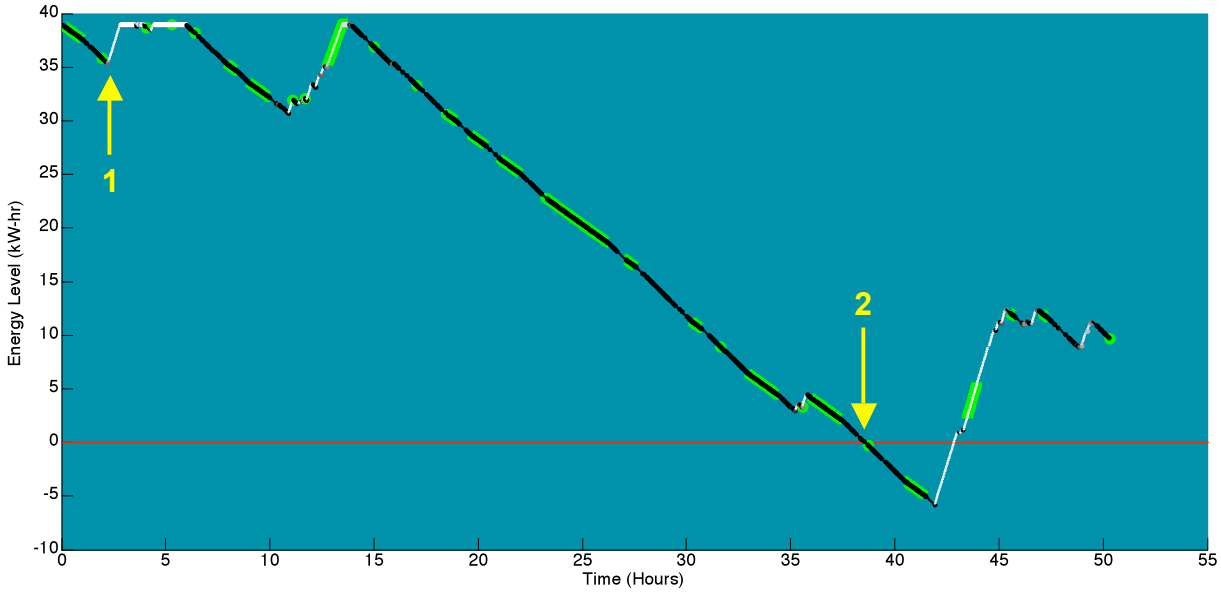


Figure 5.74. Battery energy level during rover traverse with respect to time, with shadowing

The traverse begins in darkness, and so the solar array cannot produce any power. The rover must rely on batteries. The batteries discharge from their initial level of 39 kW-hr to 35.4 kW-hr in 2 hours and 11 minutes. At this point, the rover enters the first sunlit region (notation 1 on Figure 5.74). The solar array is sized such that it produces more than enough power to run the rover when in complete sunlight. The excess power is used to recharge the battery. The battery regains its maximum energy level after only 38 minutes. Once this is reached, the excess power produced by the solar arrays is not needed, and is shunted off as heat. This discharge-recharge cycle continues, as the rover uses battery energy between Activity Points 5 and 9, and recharges once again to the full level. However, after Activity Point 10, the rover spends a long period of time in complete shadow, powering itself solely by the batteries. There is no opportunity for recharging. This leads to a major issue, as the rover's energy level reaches 0 W right before Activity Point 23, 38 hours and 34 minutes after the traverse begins (notation 2 on Figure 5.74). This would be a catastrophic failure, as the rover is over 100 km away from the habitat, an unrealistic distance for astronauts to travel on foot. So, modifications must be made to the traverse to spend more time in sunlight, either while moving or stationary.

These modifications can be accomplished in conjunction with another task – to add long-duration stops along the traverse where the astronauts can sleep. It is important to ensure that they occur in the sunlight so that the solar arrays can completely power the rover. This will also

allow the batteries to recharge, maybe enough even to fix the current energy constraint violations. Looking at Figure 5.70, it can be seen that Activity Point 10 is in sunlight and 12 hours and 44 minutes along the traverse. This is a good length of time for the astronaut's first day of exploration. Instead of stopping here for 60 minutes, as was originally desired (Table 5.7), the traverse is modified to stop at Activity Point 10 for 7 hours (420 minutes).

The modified traverse still covers 533.50 km, but now lasts 56 hours and 17 minutes and requires 363.9 GJ of energy. The rover battery energy for this new traverse is shown in Figure 5.75 through Figure 5.78.

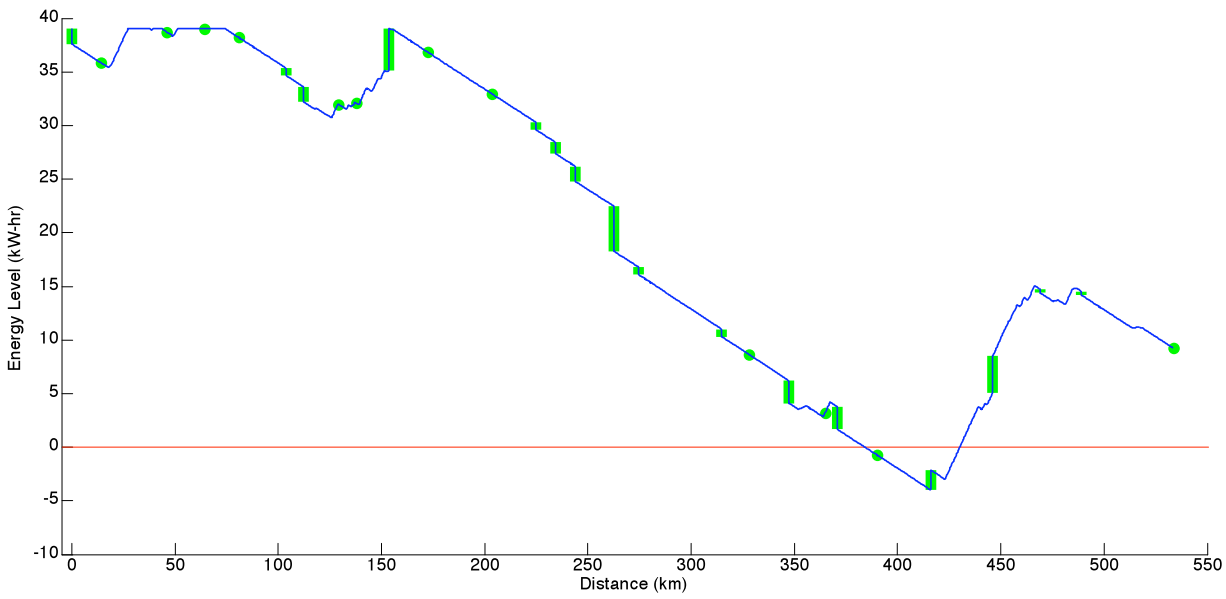


Figure 5.75. Battery energy level during first modified rover traverse with respect to distance

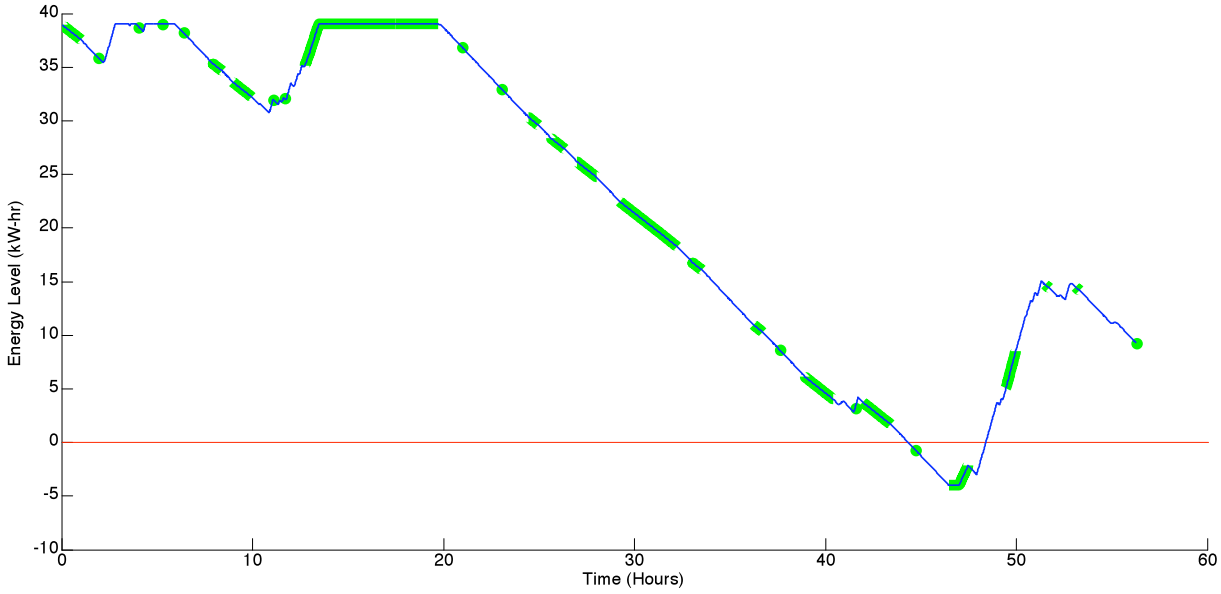


Figure 5.76. Battery energy level during first modified rover traverse with respect to time

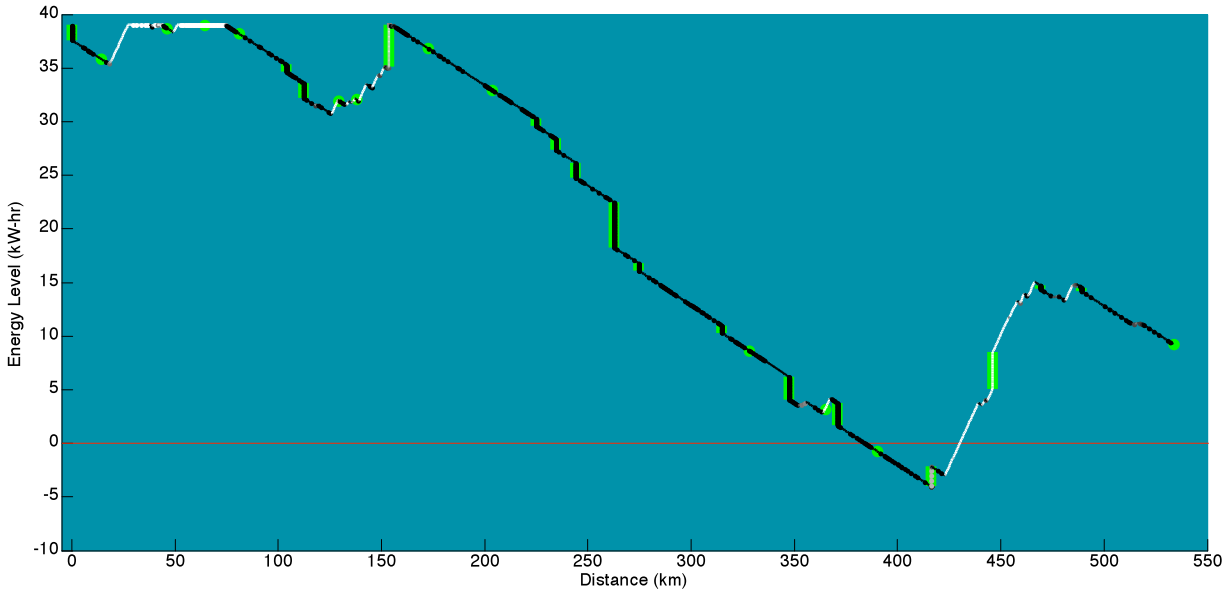


Figure 5.77. Battery energy level during first modified rover traverse with respect to distance, with shadowing

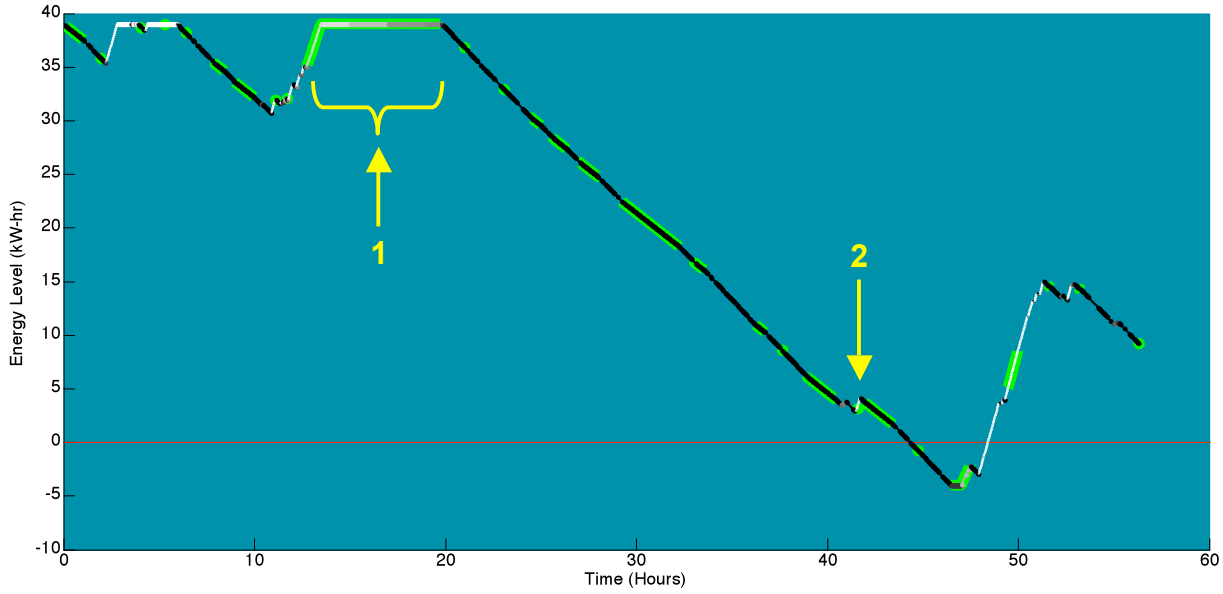


Figure 5.78. Battery energy level during first modified rover traverse with respect to time, with shadowing

The solar array powers the rover during the entire stop at Activity Point 10 (notation 1 on Figure 5.78). Consequently, the battery is able to fully recharge. However, the rover still runs out of battery energy right before Activity Point 23. Designating one long-duration sleep stop has not fixed the energy constraint violation. Furthermore, after the rover leaves Activity Point 10, there are still 37 hours and 33 minutes left in the traverse. So, another long-duration stop in the sun can be specified to give the astronauts a chance to sleep and recharge the batteries. Looking at the sun illumination map of the region (Figure 5.2), it can be seen that there is a region of sunlight on the rim of Shackleton Crater (notation 3). The rover already travels in this region, through Activity Points 20, 21, and 22. Stops are currently made at Activity Points 20 and 22, where the rover is in shadow. However, looking at Figure 5.70, one can see that Activity Point 21 is in sunlight. The battery recharges here somewhat, but the amount of recharging is minimal (only 1.329 kW-hr) because the rover only passes through this region, (notation 2 on Figure 5.78). By stopping at Activity Point 21 instead of Activity Point 22, the solar array can recharge the battery while the rover is stationary and prevent the rover from running out of energy. Both Activity Points are also on the rim of Shackleton Crater, and so minimal scientific losses should occur from this reassignment. The rover also arrives at Activity Point 21 at a time of 21 hours and 50 minutes after leaving Activity Point 10. This would make for a long day, but would be the best

location for a second stop for sleeping. So, the traverse is modified once again so that the rover stays at Activity Point 21 for 7 hours. Figure 5.79 through Figure 5.82 show the battery energy level with respect to distance and time for this second modified traverse.

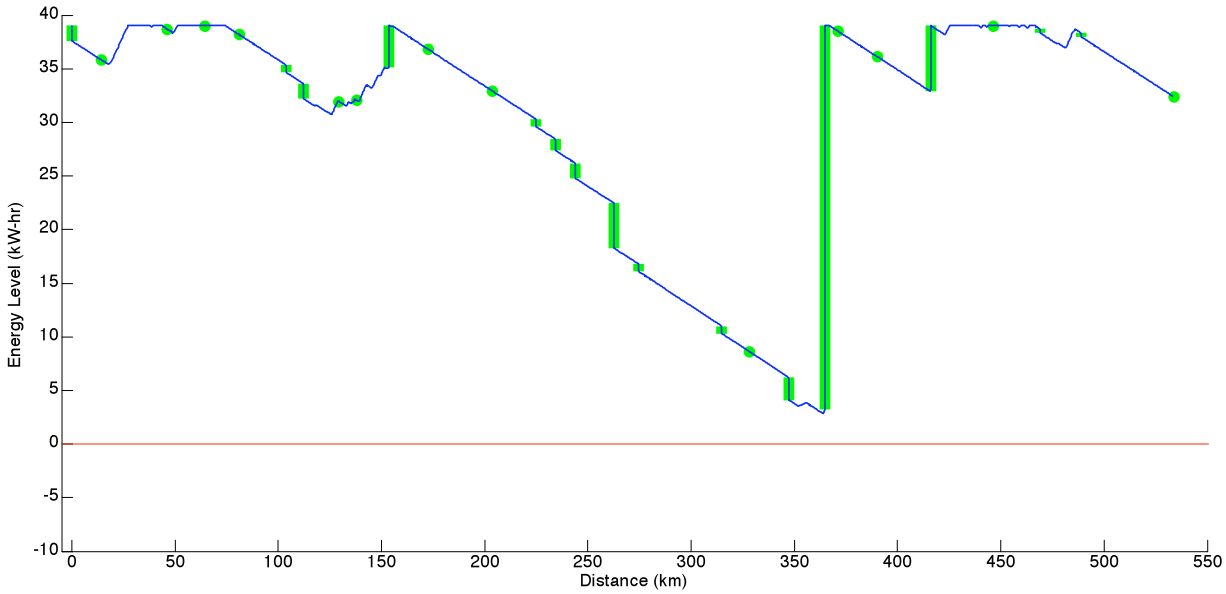


Figure 5.79. Battery energy level during second modified rover traverse with respect to distance

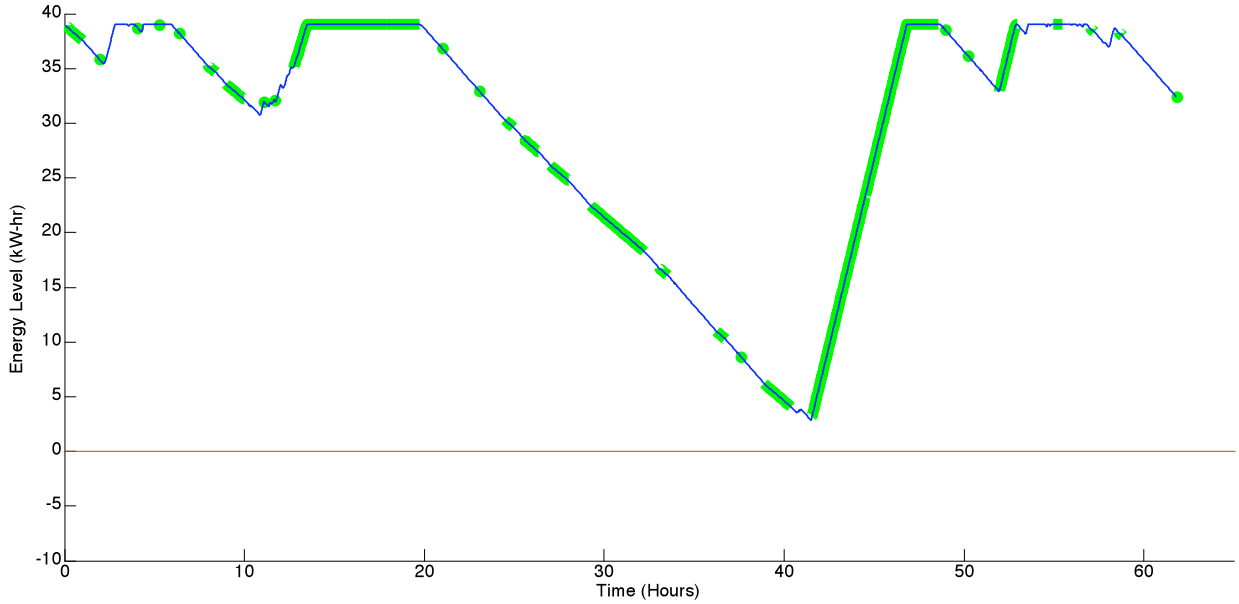


Figure 5.80. Battery energy level during second modified rover traverse with respect to time

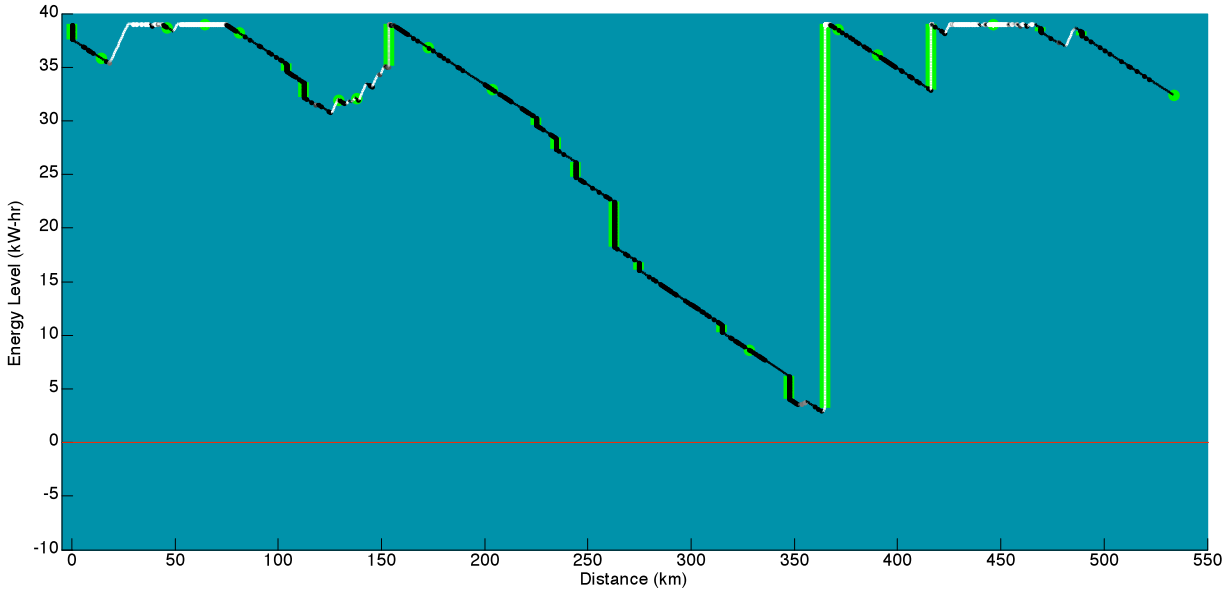


Figure 5.81. Battery energy level during second modified rover traverse with respect to distance, with shadowing

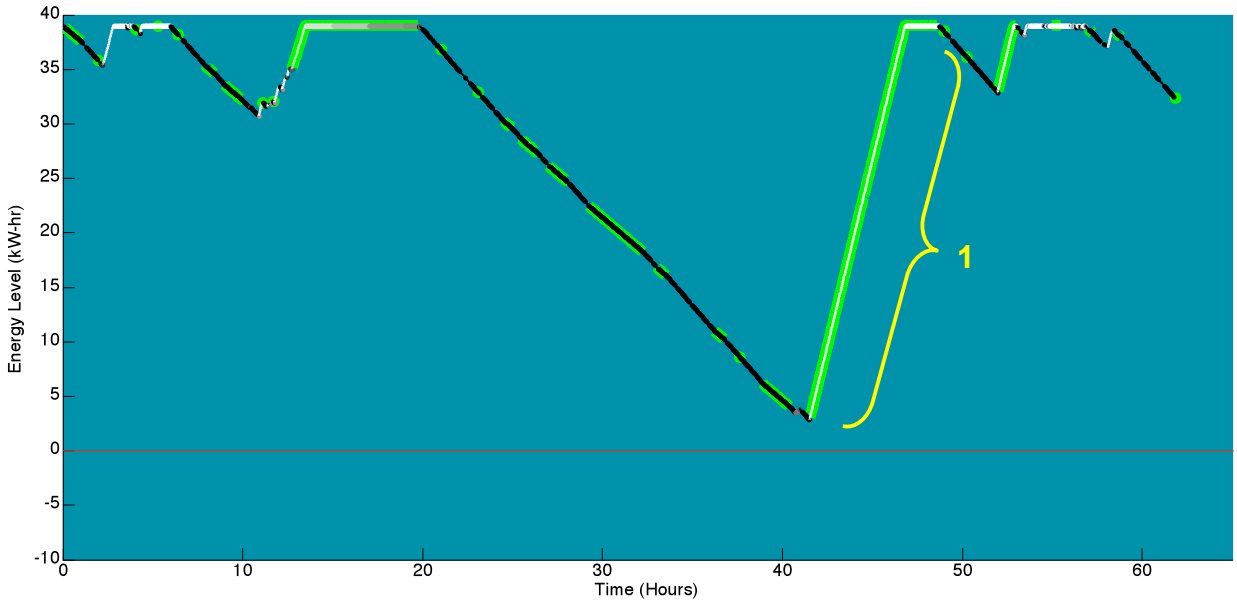


Figure 5.82. Battery energy level during second modified rover traverse with respect to time, with shadowing

When stopped at Activity Point 21, the batteries have a chance to recharge completely (notation 1 on Figure 5.82). This prevents the rover from running out of power along the traverse. Rather, the minimum energy level is now 2.87 kW-hr, a margin of 7.3%. After the rover leaves Activity Point 21, the remainder of the traverse takes 13 hours and 13 minutes to complete. This is small enough to complete without another sleep period. So, the traverse is deemed to be feasible now that a sufficient number of sleep periods have been added and the energy constraints have been satisfied.

5.2.2 Generating Return-Home Paths

One final important consideration is the amount of energy that it takes to return directly to the habitat from points along the traverse. It may be the case that the most energy-efficient path computed by SEXTANT from a certain point back to the habitat takes the rover through shadowed areas where the battery energy is completely exhausted. If this is the case, there are two options: the designated return-home path from this point can be one that is not the most energy-efficient route, but takes the rover through more sunlight; or the traverse itself can be modified so that the rover always has enough energy to return to the habitat. SEXTANT allows the user to specify return-home paths from any point along the traverse and to determine the shadowing and battery energy level along this route. This allows him to test whether or not the energy constraints would be violated on a return-home path. The rover energy level at the beginning of the return-home path is the value that it had at that particular point in the parent traverse.

The remainder of the chapter displays two return-home paths from different points along the finalized traverse. The first return-home path begins on the floor of Haworth Crater between Activity Points 14 and 15, as seen in Figure 5.83. Notation 1 is the starting point of the return-home path, and notation 2 points out the actual path. Figure 5.84 shows that this return-home traverse begins in shadow, and then finishes in sunlight closer to the habitat. It is 101.91 km long, takes 6 hours and 47 minutes to complete, and requires 48.52 GJ of energy. Figure 5.85 through Figure 5.88 show that the rover battery energy is never completely exhausted. It starts off at 26.79 kW-hr, and reaches a minimum of only 25.74 kW-hr. This is certainly within the energy constraints, and the rover would easily be able to return from the floor of Haworth Crater to the habitat.

5. Example Traverses

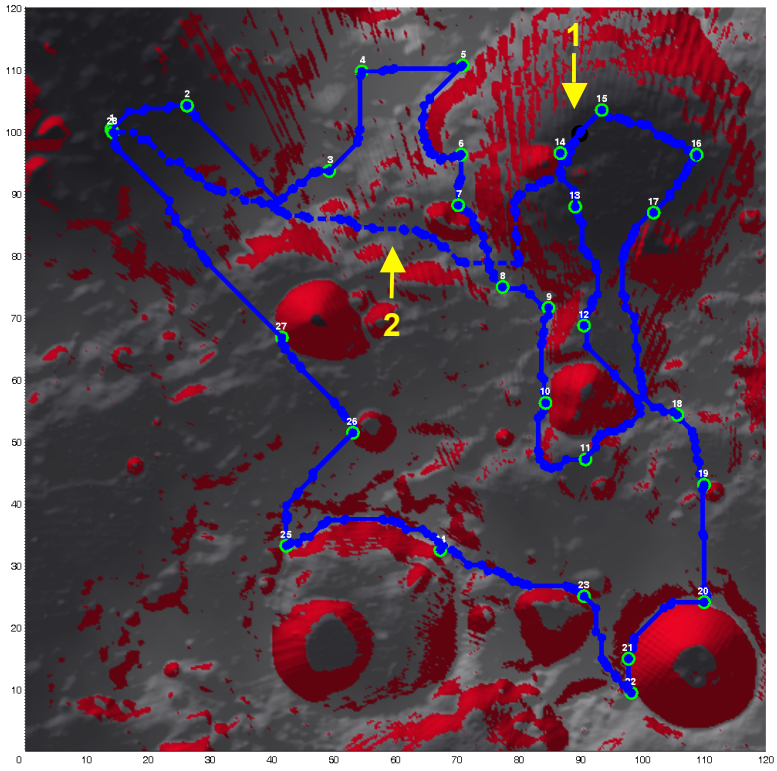


Figure 5.83. Rover return-home path 1 from Haworth Crater to habitat

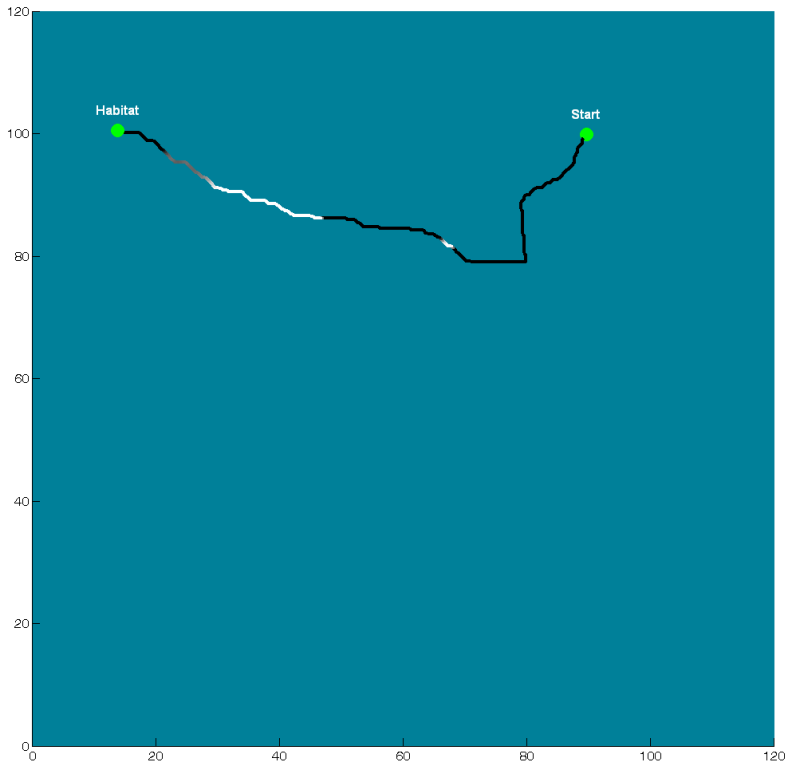


Figure 5.84. Shadowing along rover return-home path 1

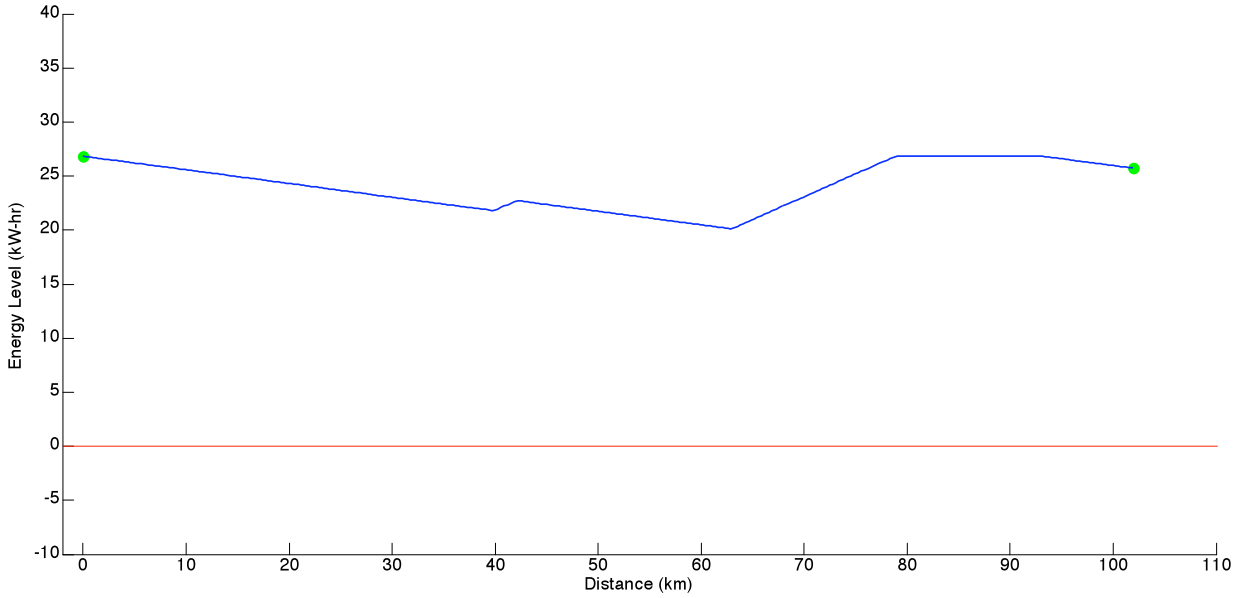


Figure 5.85. Battery energy level during rover return-home path 1 with respect to distance

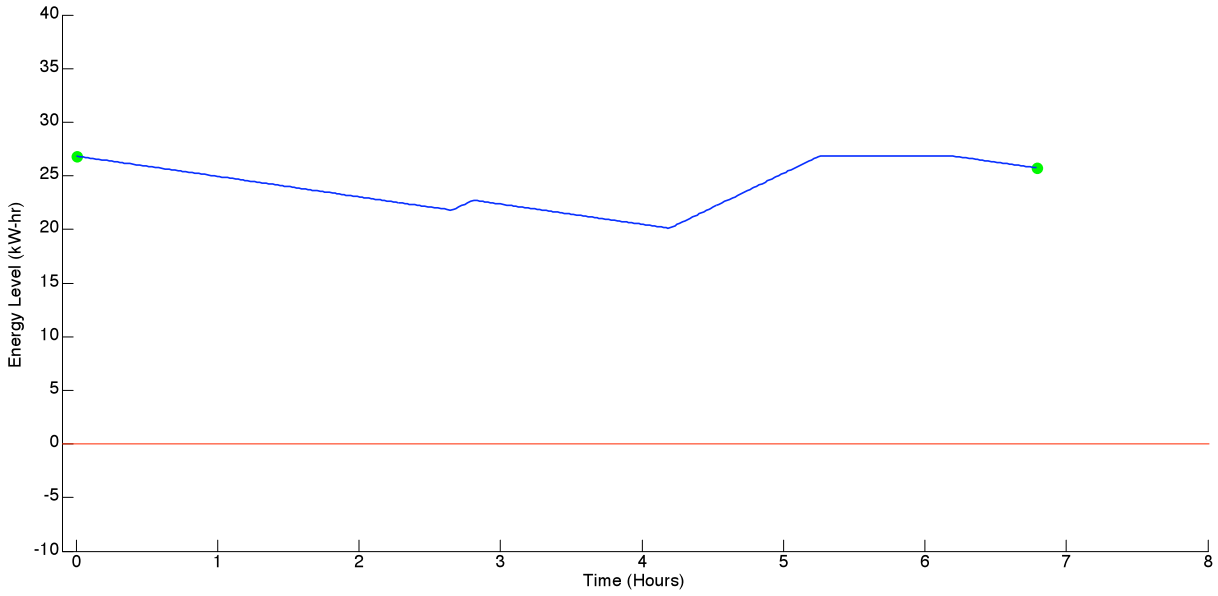


Figure 5.86. Battery energy level during rover return-home path 1 with respect to time

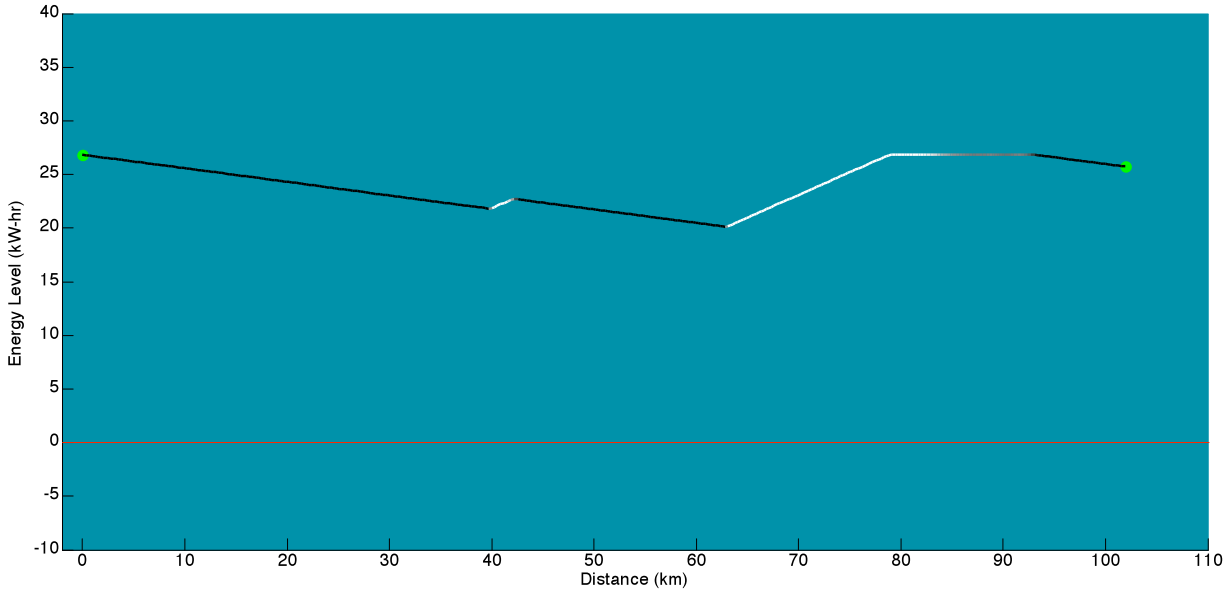


Figure 5.87. Battery energy level during rover return-home path 1 with respect to distance, with shadowing

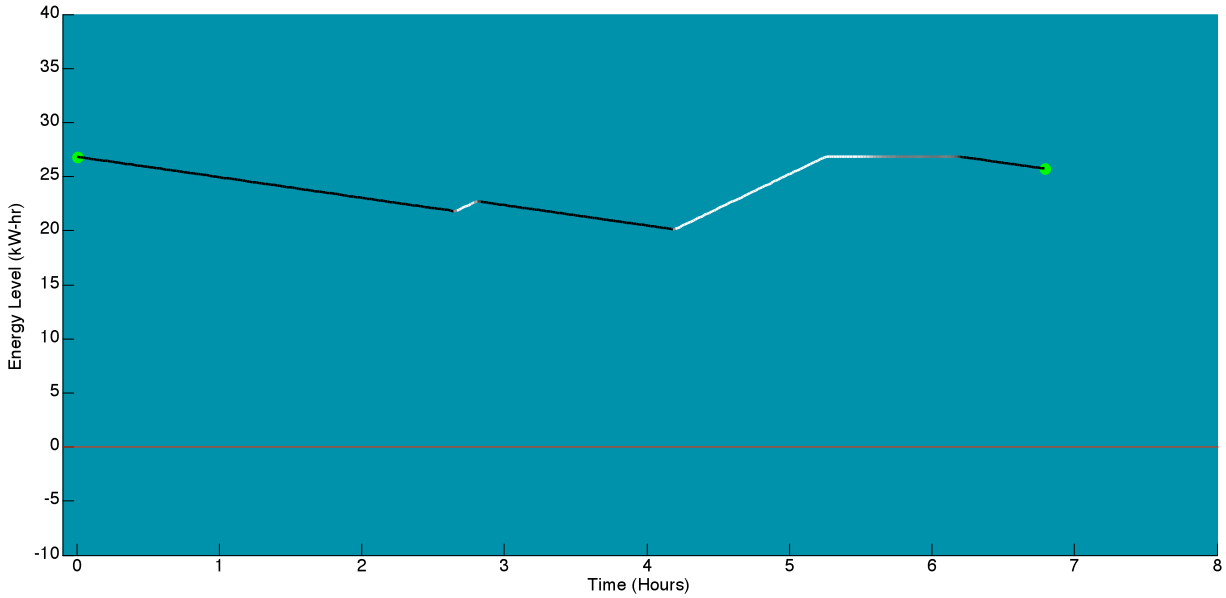


Figure 5.88. Battery energy level during rover return-home path 1 with respect to time, with shadowing

The second return-home path begins between Activity Points 20 and 21, on the rim of Shackleton Crater (notation 1 on Figure 5.89). It is 135.44 km back to the habitat, which takes 9

hours and 1 minute and 64.3 GJ of energy. Like the first return-home path, this route begins in shadow and then eventually enters portions of sunlight (Figure 5.90).

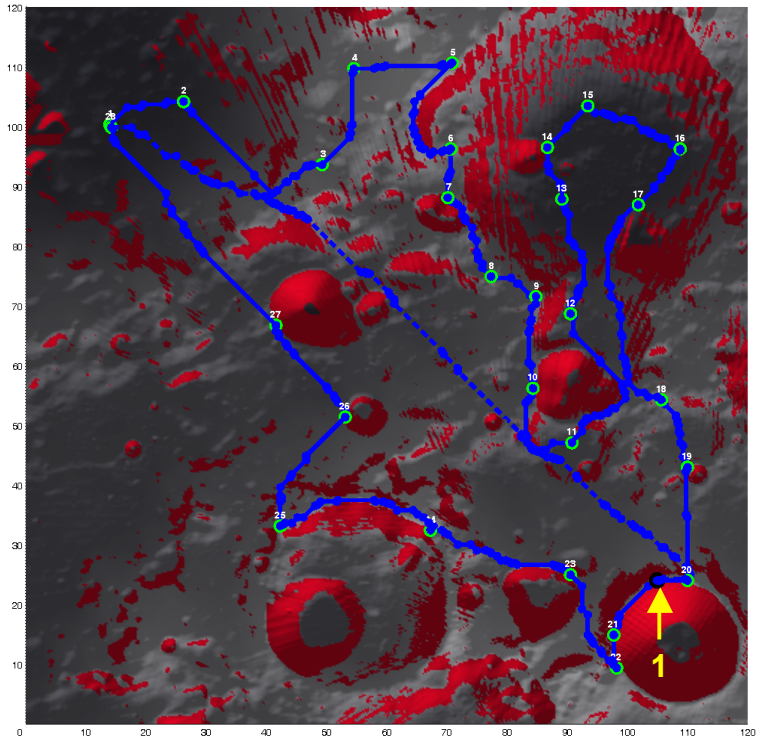


Figure 5.89. Rover return-home path 2 from Shackleton Crater to habitat

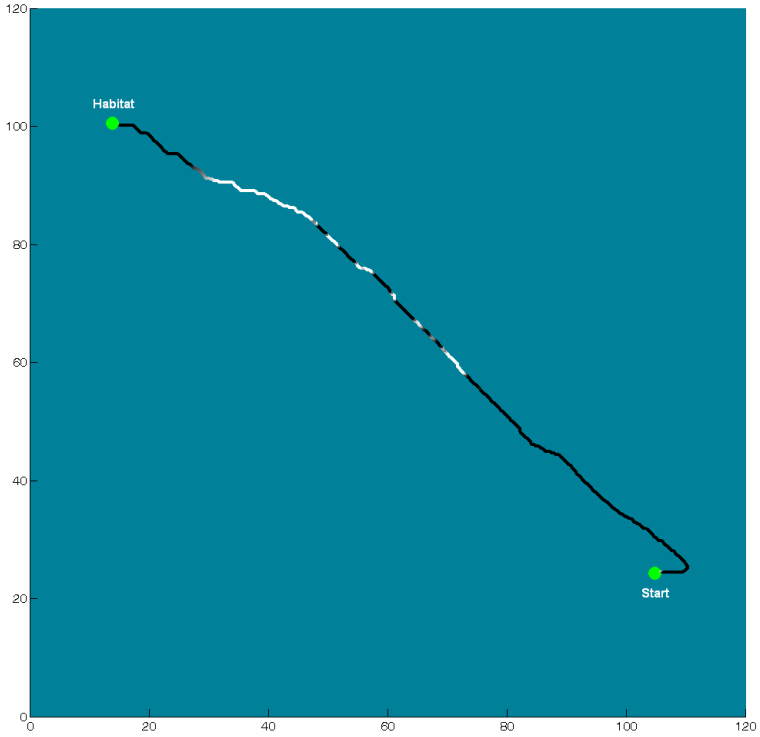


Figure 5.90. Shadowing along rover return-home path 1

5. Example Traverses

Figure 5.91 through Figure 5.94 show that during this return-home path, the rover battery is completely discharged after only 2 hours. This is because the initial energy level is low – only 3.574 kW-hr. In an emergency, the rover would not be able to complete this path back home. If the original traverse was not modified, in an emergency the rover would have to travel to Activity Point 21, remain stationary while the solar arrays recharged the batteries enough, and then continue on directly to the habitat. This would take additional time, and so it may not be the best solution. SEXTANT gives the user the ability to modify this traverse and see how the rover energy level changes. Eventually, the user can come to a solution that satisfies the energy constraints for both the original traverse and the return-home paths from points along the traverse. In the future, SEXTANT can be made to automatically check the return-home paths from points along the traverse, to see if the rover has enough energy to return home. If so, the user can be alerted and can make a decision on how best to modify the traverse.

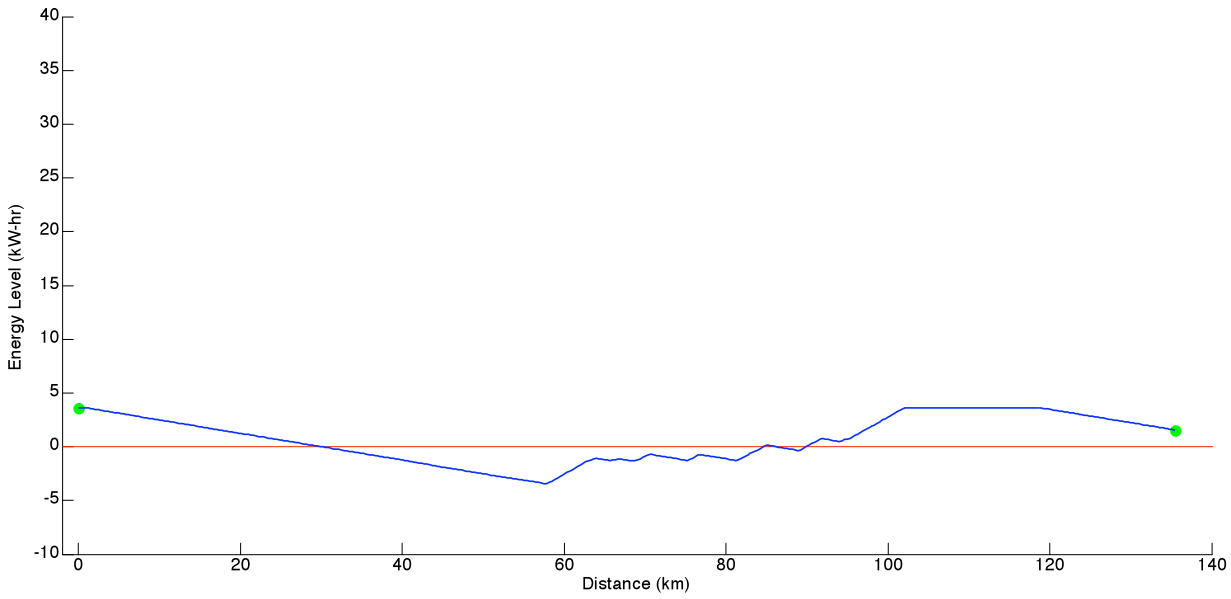


Figure 5.91. Battery energy level during rover return-home path 2 with respect to distance

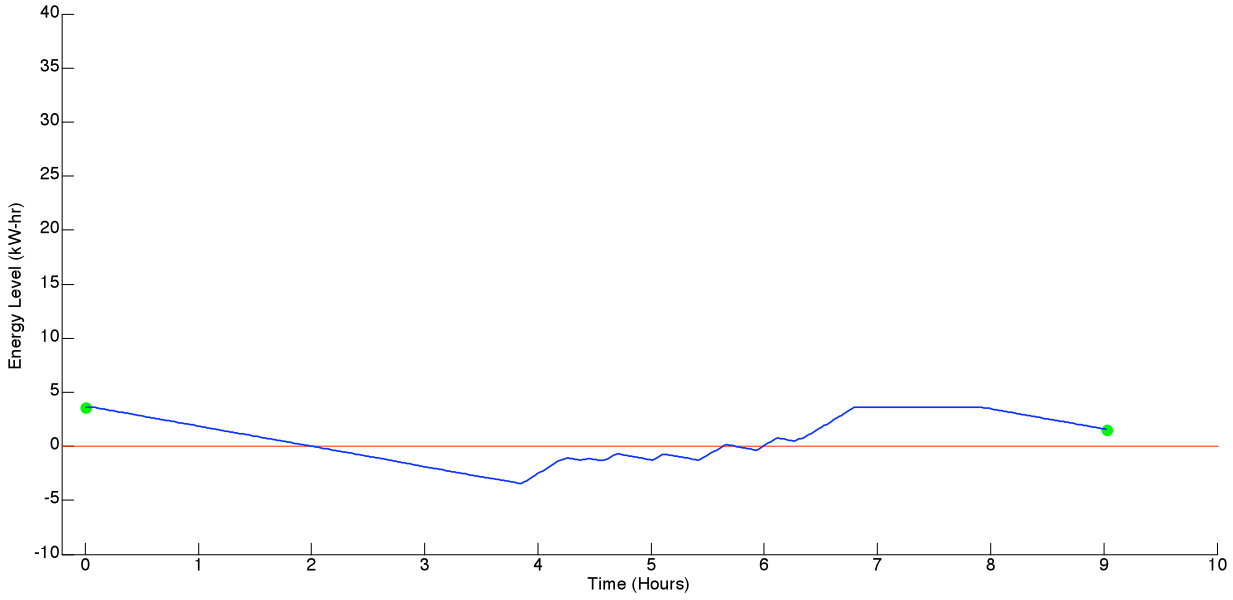


Figure 5.92. Battery energy level during rover return-home path 2 with respect to time

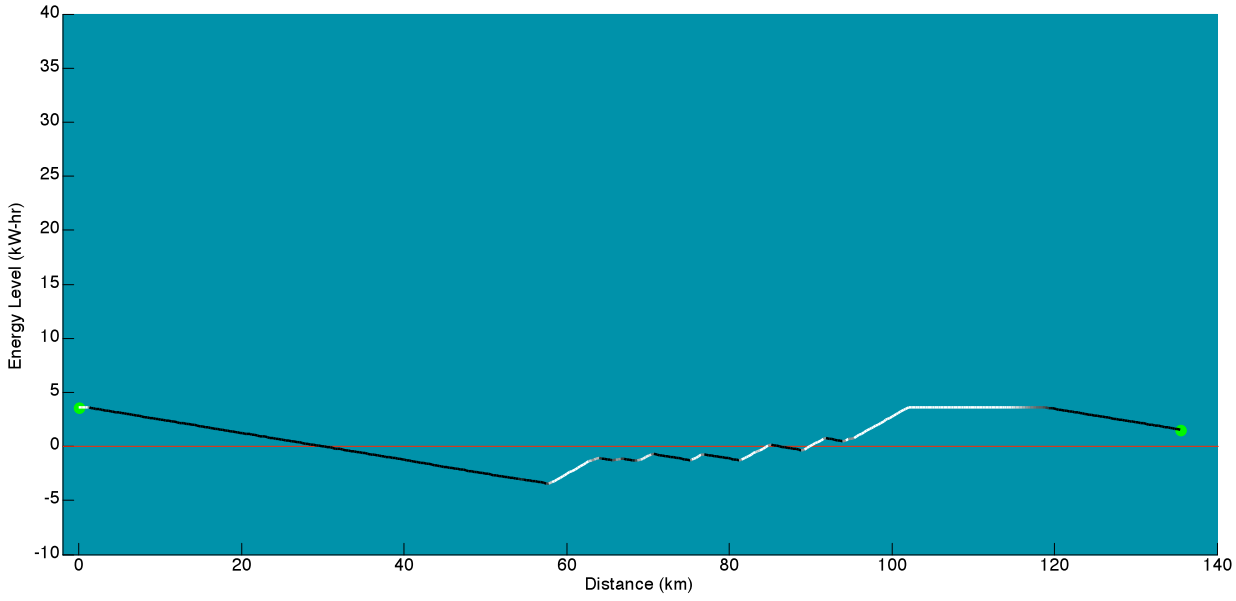


Figure 5.93. Battery energy level during rover return-home path 2 with respect to distance, with shadowing

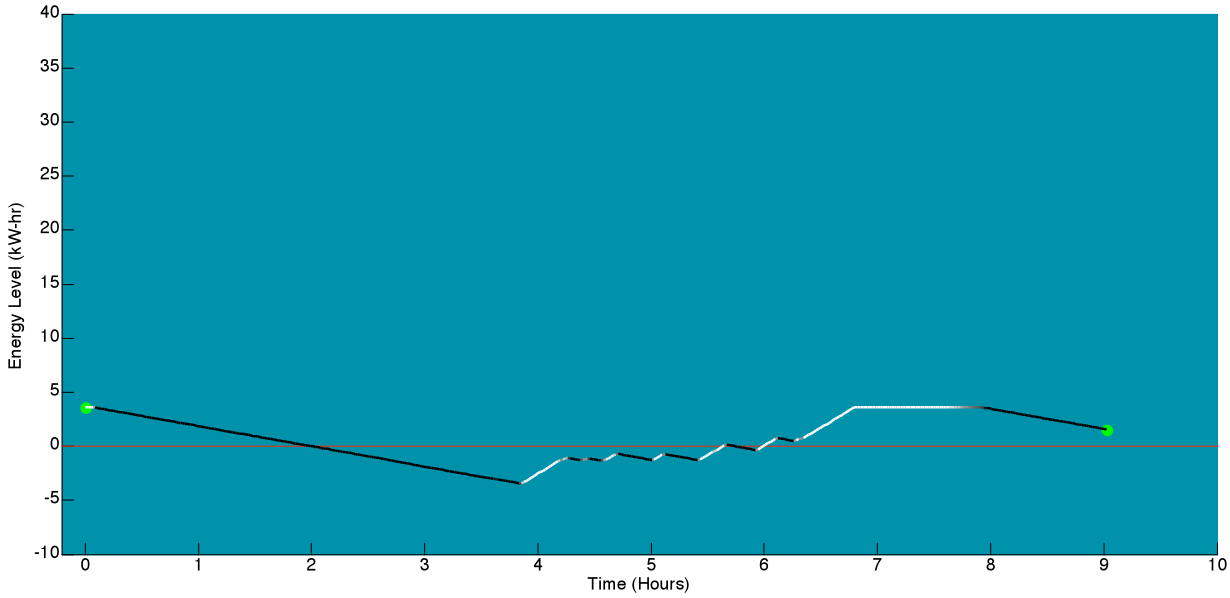


Figure 5.94. Battery energy level during rover return-home path 2with respect to time, with shadowing

5.3 Conclusion

This chapter has shown how SEXTANT determines shadowing along a lunar traverse and uses this information to compute the thermal loading on an astronaut and the power usage and generation of a transportation rover. This is crucial for determining whether or not a traverse that seems to meet topological and metabolic constraints is actually feasible. For an astronaut, it is important to ensure that the mass of water required for sublimation cooling and the amount of heater power required both do not violate given constraints. If constraints are violated, the thermal control system would not have enough capability to protect the astronaut through the entire traverse. For a transportation rover, it is critical to know that the rover will not run out of battery energy in a shadowed area along the traverse. If this were to happen, the rover could be stuck too far from the habitat for the astronauts to walk there. SEXTANT gives the user the ability to check all of these circumstances for a planned traverse. If the user discovers that some constraint is being violated, he can continuously modify the path and check the shadowing-related metrics to create a valid traverse before the explorer leaves the habitat.

This page intentionally left blank.

This thesis is almost complete;
Results and discussions concrete.
But one's told by professors
To leave work for successors;
There is, and they're in for a treat.

6 Conclusions and Future Work

6.1 Motivation for SEXTANT

When humans return to the Moon in the future, likely in preparation for further explorations to Mars, their activities will be much more aggressive and demanding than any previous lunar explorations. Astronauts will explore the lunar surface both on foot during Extravehicular Activities (EVAs) and using transportation rovers similar to NASA's Lunar Electric Rover (LER) prototype. These explorations will last anywhere from 8 hours to 14 days, and cover 20 km up to 1000 km. The upper bound of this capability represents a 4400% increase in traverse duration and 4900% increase in traverse distance over the longest Apollo EVA. The frequency of traverses will also greatly increase, as astronauts will be working outside of the habitat or exploring the lunar surface nearly every day.

The difficulty and complexity of this challenging exploration plan will be complicated by inherent issues with navigation on the lunar surface. Astronauts have had trouble judging distances on the Moon due to the lack of aerial perspective and the absence of familiar objects of known size. As a result, the Apollo astronauts often had difficulty in evaluating how long it would take them to travel to waypoints along their EVA. They also had trouble identifying the navigational landmarks, like craters, which allowed them to locate their position. These challenges can all be clearly seen in the second EVA of the Apollo 14 mission, where the astronauts had to abandon their destination of Cone Crater because the EVA schedule had slipped so much due to confusion and a loss of situational awareness (Carr et al. 2003). With the long duration and high frequency of traverses occurring in future explorations, it will be extremely critical to overcome these challenges.

6.2 Thesis Contributions

The Surface Exploration Traverse Analysis and Navigation Tool (SEXTANT) has been designed to help mitigate the difficulties and increase the ability of astronauts to carry out explorations on the lunar surface. This thesis has described a number of critical improvements and additions to SEXTANT to comprehensively model future lunar explorations for both human astronauts and transportation rovers.

6. Conclusions and Future Work

The major contributions of this thesis are:

1. Incorporation of a lunar elevation map for traverse planning over the lunar surface
2. Development of an explorer model for transportation rovers
3. Integration of SEXTANT with the NASA Ames Individual Mobile Agents System (iMAS) for real-time guidance along a traverse
4. The prediction of shadowed and sunlit areas along a traverse, and the resultant thermal load for suited astronauts and power generation and usage for transportation rovers

6.2.1 Incorporation of a Lunar Elevation Map

Previously, SEXTANT only had the ability to plan paths over terrestrial maps. While beneficial for field tests and algorithm development, this did not allow for realistic simulations of future lunar explorations. The work in this thesis has expanded SEXTANT's capabilities to allow the user to plan lunar traverses. Elevation maps of both lunar poles have been generated by the Lunar Orbiter Laser Altimeter (LOLA) spacecraft currently orbiting the Moon. SEXTANT currently imports a 500 km by 500 km elevation map of the lunar south pole, a focused area of exploration because of the potential for water ice in permanently shadowed craters. This map is a regularly spaced grid with 240-meter horizontal resolution and 0.1-meter vertical resolution. The user can select a submap of any size from this original map, and plot traverses with the SEXTANT 3D mapping interface. The equations that describe the energy cost of a traverse for astronauts or transportation rovers are also modified to take into account the lesser gravity on the Moon. Being able to plan traverses across the lunar surface is a large step in accurately modeling future traverses.

6.2.2 Development of a Transportation Rover Explorer Model

This thesis describes the transportation rover explorer model within SEXTANT, which was developed as a part of this work. Earlier versions of SEXTANT only contained an explorer model for suited astronauts. Because the space suit life support system is only being designed for 8 hours, an astronaut carrying out an EVA on foot will not be able to explore a large area (Culbert 2009). He will be confined to a 10-15 km radius around the point of departure, whether a permanent habitat or rover. It is almost certain that future explorations will rely heavily upon transportation rovers to support the astronauts in longer and lengthier traverses. While there will

still be suited EVAs originating from both the habitat and transportation rover, the rover will be the main method of transportation. Furthermore, NASA's Lunar Electric Rover (LER) concept includes large windows at the bottom of the rover's cockpit, so that the astronauts can explore the lunar surface during intravehicular activity (IVA) without having to perform an EVA.

Figure 6.1. Geologic explorations of the Arizona desert from the Lunar Electric Rover prototype (Gambino 2010)

Because of the certain prevalence of transportation rovers in future lunar explorations, it is extremely beneficial that SEXTANT is able to plan paths for them. This will help to make sure that any traverse undertaken will not be in any danger of running out of consumables. Furthermore, SEXTANT gives the user the ability to plot paths for multiple explorers at the same time. So, traverses with multiple transportation rovers, or rovers and suited astronauts, can be developed. These give the user greater flexibility to specify whatever traverse architecture will exist during future lunar missions.

6.2.3 Integration of SEXTANT and iMAS Astronaut Assistant System

The current SEXTANT interface is not one that has been optimized for the astronaut user. There are a number of improvements to be made before SEXTANT would be carried by astronauts on the Moon or another planetary surface (Section 6.3.3). Nevertheless, this thesis marks an initial step towards this implementation through the integration of iMAS and SEXTANT. Using SEXTANT, the astronaut can plan a traverse and optimize it fully through the 3D mapping interface. This traverse must be an Earth-based example, because iMAS currently only has the capability to track the astronaut through GPS. Once the traverse has been generated as desired, the astronaut can import the plan into iMAS. This preserves the locations of Path Points along

the traverse, and the time that the user should arrive at each. Then, using the iMAS auditory interface, the astronaut can physically travel along the path and ask for directions to the Path Points. The directions to a Path Point are returned as the distance and heading relative to the astronaut's direction of travel. It can be envisioned that a form of this auditory interface would be used in conjunction with a visual interface for future explorations. The visual interface could be used for spatially planning and re-planning traverses and global situational awareness, while the auditory interface could be used for navigation, guidance, and quick re-planning tasks like, "Take me home." The integration of iMAS and SEXTANT is a step that demonstrates some of these future capabilities.

6.2.4 Determination of Shadowing, Astronaut Thermal Load, and Rover Power Generation and Usage

The final, and largest, contribution of this thesis is the determination of shadowing along lunar traverses. More importantly, knowledge of the shadowed and sunlit portions of a traverse allows SEXTANT to track the usage of thermal consumables for an astronaut on EVA and the power generation and usage of transportation rovers. This is a critical step in ensuring that planned traverses are feasible. In computing the shadowing along a traverse, SEXTANT accounts for the movement of the sun with time. This is important, as the sun can move a full 180° across the sky during a 14-day traverse. Accounting for sun illumination and shadowing ensures that the traverse model is as realistic as possible. The metric of shadowing is the percentage of the solar disk visible, which can range from 0 (complete shadow) to 1 (complete sunlight).

The astronaut thermal model in SEXTANT computes the heat flux to or from the space suit with respect to three components: the external environment, waste heat from the space suit electronics, and heat released by the astronaut's body due to inefficiencies in the usage of metabolic energy. The heat flux to or from the environment is a function of the shadowing and the external space suit temperature. The metabolic heat flux is a function of the astronaut's metabolic rate and the terrain slope, and the electronics waste heat is a user-specified constant. The thermal control system balances out the heat input or output by either removing heat through the sublimation of ice or adding heat with a heater. The amount of water needed to replenish the sublimated ice and the heater power are both cumulatively tracked through the traverse, so that the user can see what portions of the traverse place the most stress on the thermal control system.

With SEXTANT, the user can easily re-plan the traverse and view how the thermal metrics change. The user can continue to modify the traverse until it satisfies the thermal constraints.

Instead of looking at thermal characteristics of a rover, SEXTANT tracks its power consumption and generation. A rover can use the solar arrays when in sunlight, but must rely on batteries when in shadows. The solar array also gives the ability to recharge the battery when there is excess power. However, if the rover is in a long period of shadow, the battery power may dip to undesirable levels. The rover could run out of power mid-traverse, or reach a point where there is not enough battery power to return home. Similar to the astronaut thermal model, the user can plan a desired rover traverse with SEXTANT and easily see if it is feasible. If the constraints are violated, he can modify the path a number of times, seeing how the power consumption and generation changes. As an example, when the battery energy is running low the user can identify an area where there is sunlight, and specify that the rover remain stationary here for a period of time. This recharges the battery, allowing the traverse to continue in the shadows.

6.2.5 Overall Contributions

To the best of the author's knowledge, this thesis has developed the most comprehensive and accurate representation of suited astronaut EVAs and transportation rover traverses on the Moon. SEXTANT is the only traverse planning tool that allows users to optimize lunar traverses on the basis of metabolic cost, and the only tool that investigates the shadowing-related thermal or power characteristics of specific paths. These capabilities provide benefits to two different groups in two different time frames. Firstly, SEXTANT can assist engineers and lunar architects during the design phase of future hardware and missions. The designers can plan candidate traverses over the lunar surface to see how long they will take to accomplish, how much energy will be required, what the thermal requirements for astronaut EVAs will be, and how much power will be necessary for rover traverses. This mission planning will help to size the hardware so that the most ambitious traverse desired can be satisfied, with margin. As the hardware parameters evolve, these can be updated in SEXTANT so that the traverse planning capabilities are also current. The second utility of SEXTANT is further off in the future, but also extremely important. SEXTANT could be used by astronauts in real-time during future lunar missions to plan and re-plan paths in real-time. An astronaut preparing for a traverse could first plan it with SEXTANT, ensuring that there is enough margin between the traverse metrics and the related constraints. More importantly, an astronaut out on a traverse could use SEXTANT to re-plan his

path in response to some changing condition, like an equipment failure or a new region of scientific interest. The astronaut could quickly modify the Activity Points, re-run the optimization routine, and view the details of the new traverse. If it is valid he could continue on this new path, but if it is not then he could re-plan once again. The astronaut could also use the visual or auditory interface to follow the path that SEXTANT has created. SEXTANT will help to give the astronauts more autonomy in traverse planning. If conditions change, a new path can be planned with accuracy and thoroughness from the Moon, without much consultation with the ground. Giving astronauts more autonomy while keeping them safe will be a large step in making future explorations more efficient and effective. This will be absolutely critical for Mars exploration, where real-time interaction with Mission Control is impossible.

6.3 Future Work

Even though SEXTANT's representation of traverses is quite comprehensive, there is still a great deal of future work to be performed. This section will discuss a few of these areas and why they are important.

6.3.1 Integration with a Lunar Navigation System

One current assumption of SEXTANT is that there will be some type of lunar navigation system in place when astronauts return to the Moon. Such a system would allow SEXTANT to track the astronaut through an actual lunar traverse and give real-time guidance along the planned path. This is a necessary functionality for SEXTANT to become an implementable mission planning tool. Many different proposals for such a navigation system have been put forth, using inertial measuring units (IMUs), radio beacons, and star trackers, for example. Global positioning systems are not a viable candidate as they would be expensive to develop, and magnetometers would not be effective due to the moon's weak magnetosphere (Powell 2006)

IMUs are an well-studied solution to this problem that have already been used in many space missions. These are instrument packages that use accelerometers and gyroscopes to tell how an object has moved with respect to its initial position. IMUs were used for navigation by the Lunar Roving Vehicle (LRV) on the Apollo 15, 16, and 17 missions, and successfully helped to keep astronauts on-track. On problem with IMUs is that they can "drift" over time, as small errors get compounded. To compensate for this problem, IMUs must be augmented with a secondary sensor. Li proposes that this be accomplished with small vision sensors placed on the

astronaut's torso or helmet (Li et al. 2009). The vision sensors identify when the user is stopped, and then the IMU data can be "reset". This technique, called "zero velocity update", nullifies the drift and increases the accuracy of the IMU.

SEXTANT can be combined with an IMU system for real-time guidance over the lunar surface. This would be similar to the integrated SEXTANT-iMAS system, and could even use the same architecture. A traverse can be planned in SEXTANT and then followed by an astronaut. Position updates from the IMU would be sent to SEXTANT with some set frequency, and SEXTANT would track the explorer. When requested, SEXTANT would give guidance along the planned path. The astronaut's heading would be known absolutely from the IMU data, and so the heading assumption used in iMAS (Section 3.4.3.2) would not have to be made. SEXTANT could also give the astronaut an alert if he traveled too far off-course.

6.3.2 Include Sun Position in Cost Functions

Besides being used to determine shadowing, the sun position can be incorporated into the cost function that determines the most efficient path. Astronauts navigating across a planetary surface, whether on foot or driving a rover, have a difficult time heading directly into or away from the sun (Carr et al. 2003). This is due to the low contrast caused by the back-scattering properties of the lunar regolith. Walking directly into the sun, astronauts also have problems with glare that obscures terrain features. Consequently, the astronauts must slow their velocity to prevent an accident.

The sun illumination can be added into the SEXTANT cost functions through the metric of a *sun score*, developed by Carr, et al. and refined by Márquez (Carr et al. 2003, Márquez 2008). This variable measures how favorable the sun position is for a certain path direction in terms of the ease of navigation. The sun score (*SS*) is a function of the relative azimuth (θ) and elevation (φ) angles between the astronaut and sun position:

$$SS = \left(\cos(2\theta) + 1\right) \cdot \left(\cos(2\varphi) + 2\right) \tag{6.1}$$

These angles can be determined from the difference between the absolute angles of the sun and astronaut. The orientation of the astronaut is known, and the orientation of the sun can be determined by knowing the geographic position of the astronaut and the date and time of the traverse. Figure 6.2 shows a depiction of these angles.

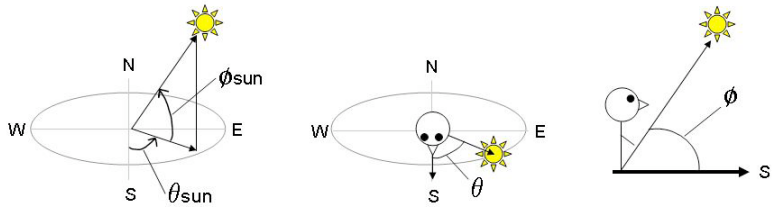


Figure 6.2. Graphical representation of absolute and relative sun azimuth and elevation angles (Márquez 2008)

A high sun score indicates that the lighting conditions are not good for the traverse, while a low sun score signifies good lighting conditions. The optimal sun score for navigation is obtained when the sun is high in the sky (φ is approaching 90°) and perpendicular to the direction of travel (θ = 90° or 270°). These conditions produce the most contrast on the surface. Once the sun score is developed, it can be weighed against the traverse energy cost, distance, and time to find an overall cost function. The traverse can the be optimized on this combined metric.

6.3.3 Develop a User-Centric Display

The present SEXTANT interface is an engineering interface, designed to demonstrate the algorithms and functionality of SEXTANT. It is not optimized for an astronaut to use on a lunar or planetary surface. There are many improvements to make before SEXTANT has a viable user interface for astronauts – enough for a separate master's thesis. For example, there is currently too much information presented. An astronaut does not care how much energy he will be using during a traverse. Rather, he cares about how much margin he has for all his constraints – oxygen, water, cooling, and heating. These constraint values and margins should be clearly displayed to the astronaut, with the more-detailed information available upon request. Furthermore, the arrangement of the interface buttons should be grouped according to their function and frequency of use. In the 3D mapping interface, some provision for denoting the terrain slope should also be made. Currently, the user can view the terrain slope at any point in SEXTANT by right-clicking on it. In the future, contour lines could be placed on the 3D mapping interface to show the astronauts where steep and shallow sections are. This will facilitate planning and increase an astronaut's situational awareness. Finally, the physical interface between a suited astronaut and SEXTANT must be investigated. An astronaut wearing a space suit has large, bulky gloves, which makes it difficult to use a computer. An auditory system like iMAS circumvents this issue for some commands, but a visual / tactile interface will still be necessary in a SEXTANT user interface. One must be created that is large enough to

display all the required information, small enough to be carried or worn by the astronaut, and simple enough to interact with while on EVA.

6.3.4 Measure Metabolic Costs in Real-Time

In the current version of SEXTANT, there is one base set of cost functions used to calculate the most efficient traverse paths for all astronauts. These are detailed in Section 3.1.3. Beyond accounting for explorer mass, these equations do not consider the varied metabolic expenditures for different astronauts over the same traverse. Furthermore, the terrain slope is currently the only surface characteristic that factors into the metabolic cost. To expand SEXTANT beyond these baseline metrics, real-time metabolic costs can be measured during traverses and linked to specific soil characteristics.

One system that performs real-time metabolic costs for individual astronauts is the Life Support, Exploration Guidance Algorithm and Consumable Interrogator (LEGACI) developed by Kuznetz and previously discussed in Section 3.4.1.4 (Kuznetz 2008). LEGACI uses a number of biosensors located within the space suit to collect information about the EVA status, consumables, and performance data. One of these metrics is the astronaut's real-time metabolic rate, calculated from proprietary algorithms using multiple different biosensor and accelerometer outputs. This metabolic rate is time-averaged over each minute and correlated with the current terrain characteristics. With this information, LEGACI builds a library of the astronaut's metabolic rate for different terrain characteristics such as slope and soil composition.

SEXTANT can then create new, real-time cost functions from the LEGACI library tailored to each individual astronaut. These cost functions can be updated and improved continuously after each EVA as more and more data is gathered. Instead of using the baseline cost functions to preplan traverses, the new astronaut-specific costs can be used to give a more accurate calculation of the most efficient path across the planetary surface. Furthermore, SEXTANT can use the LEGACI information during EVAs to check and compare the astronaut's performance to the expected metabolic cost. If the astronaut is under- or over-performing, the traverse can be re-planned immediately to keep the traverse within the operational constraints. LEGACI has also been integrated with iMAS, and so the astronaut would be able to get real-time updates on the level of his consumables and metabolic expenditure.

6.3.5 Optimize Scientific Return for a Traverse

When the user inputs Activity Points into SEXTANT, he must enter them in the order they are to be visited. Additionally, SEXTANT plots a traverse between all specified Activity Points, regardless of whether or not the path violates the operational constraints of distance, time, energy, thermal, or power. It is the user's job to ensure the validity of the traverse. In future versions of SEXTANT, the search algorithm can be modified to visit Activity Points in the order that maximizes the scientific return, while remaining within the operational constraints. To do this, each Activity Point can be assigned an importance, based on the perceived amount of science or exploration value that can be gained from visiting the site. Then, when SEXTANT computes the most efficient path, it would visit as many sites as possible, traveling to the sites with higher importance first. The order in which the waypoints are placed on the 3D mapping interface would have no influence on the eventual path. In doing this, SEXTANT must know the consumable limits of the astronaut or rover and how they are influenced by the energy expenditure. Thus, the planning algorithm would be able to ensure that the astronaut visits all sites along the planned path and returns home before exhausting his or her resources.

6.4 Final Thoughts

Even though the specific goals of lunar and planetary exploration are unknown, NASA will undoubtedly reach out to the Moon and Mars with both manned and robotic missions. While the timeline for these missions is not yet firm, it is still crucial to begin developing the next generation of technology that will assist these explorers, like SEXTANT. As this thesis has shown, SEXTANT contains the most comprehensive and accurate representation of lunar traverses of any tool. It can be used to plan safe, efficient, and effective traverses across the lunar surface for suited astronauts and transportation rovers. It also gives these explorers the autonomy to plan their own paths, without constantly relying on Earth. This will be essential in missions to the Moon and Mars, whatever the architecture and goals. Development on SEXTANT should continue, as with improvements it will be an invaluable mission planning tool for future explorers.

7 References

Acton, C., *SPICE Toolkit*, The Navigation and Ancillary Information Facility, 17 April 2009, <http://naif.jpl.nasa.gov/naïf/index.html>, 15 Dec. 2009.

"Apollo Lunar Mission Table", *The Moon: A Guide*, The Society for Popular Astronomy, <http://www.popastro.com/moonwatch/moon_guide/apollotable.php>, 2008.

Balinskas, R., and Tepper, E., *Extravehicular mobility unit requirements evolution*, Contract No. NAS 9-17873, Windsor Locks, CT: Hamilton Sundstrand, 1994.

Bleacher, J., Garry, B., Zimbelman, J., Clancey, W., Sierhuis, M., "The use of individual Mobile Agents Systems (iMAS) in the field: A geologist's perspective on value for future planetary missions", Poster presented at the 2009 Lunar Science Forum, NASA Ames Research Center, Moffet Field, CA, 21-23 July 2009.

Bussey, B., and Spudis, P.D., *The Clementine Atlas of the Moon*, Cambridge, UK: Cambridge University Press, 2004.

Byrne, C.J., *Lunar Orbiter Photographic Atlas of the Near Side of the Moon*, New York: Springer, 2005.

Campbell, A.B., French, J.D., Nair, S.S., Miles, J.B., Lin, C.H., "Thermal Analysis and Design of an Advanced Space Suit", *Journal of Thermophysics and Heat Transfer*, 14(2), 2000, pp. 151-160.

Carr, C. E., *Distributed Architectures for Mars Surface Exploration*, M.S. thesis, Massachusetts Institute of Technology, 2001.

Carr, C.E., and Newman, D.J., "Space Suit Bioenergetics: Cost of Transport During Walking and Running", *Aviation, Space, and Environmental Medicine*, 78(12), 2007, pp. 1093-1102.

Carr, C.E., and Newman, D.J., "Space Suit Bioenergetics: Framework and Analysis of Unsuited and Suited Activity", *Aviation, Space, and Environmental Medicine*, 78(11), 2007, pp. 1013-1022.

Carr, C. E., Newman, D. J., and Hodges, K., "Geologic Traverse Planning for Planetary EVA", SAE 2003-01-2416, 33rd International Conference on Environmental Systems, Vancouver, BC, Canada, 7-10 July 2003.

Clancey, W. J., Sierhuis, M., Alena, R. L., Berrios, D., Dowding, J., Graham, J. S., Hirsh, R. L., Garry, W. B., Semple, A., Rupert, S. M., and van Hoof, R. J. J., "Planning for Exploration: The Mobile Agents 2005 Field Test at MDRS", 8th International Mars Society Conference, Boulder, CO, 11-14 Aug. 2005.

7. References

Clancey, W. J., Sierhuis, M., Alena, R.L., Berrios, D., Dowding, J., Graham, J.S., Tyree, K.S., Hirsh, R.L., Garry, W.B., Semple, A., Buckingham Shum, S.J., Shadbolt, N. and Rupert, S., *Automating CapCom using Mobile Agents and Robotic Assistants*, NASA 2007-214554, Washington, DC: NASA, 2007.

Clancey, W. J, Sierhuis, M., Alena, R.L., Crawford, S., Dowding, J., Graham, J., Kaskiris, C., Tyree, K. S., and van Hoof, R. J. J., "The Mobile Agents integrated field test: The Mars Desert Research Station April 2003", Proceedings of The 17th International FLAIRS Conference, Miami, FL, 17-19 May 2004.

Clancey, W.J., Sierhuis, M., Alena, R.L., Dowding, J., Scott, M., van Hoof, R., "Power System Agents: The Mobile Agents 2006 Field Test at MDRS", 9th International Mars Society Conference, Washington, DC, 3-6 Aug. 2006.

Clancey, W. J., Sierhuis, M., Kaskiris, C. and van Hoof, R. J. J., "Advantages of Brahms for specifying and implementing a multiagent human-robotic exploration system", Proceedings of The 16th International FLAIRS Conference, St. Augustine, FL, 11-15 May 2003, pp. 7-11.

Costes, N. C., Farmer, J. E., George, E. B., *Mobility Performance of the Lunar Roving Vehicle: Terrestrial Studies – Apollo 15 Results*, NASA Technical Report R-401, Huntsville, AL: Marshall Space Flight Center, 1972.

Culbert, C., "Lunar Surface Systems Project Overview", Presentation at the U.S. Chamber of Commerce Programmatic Workshop on NASA Lunar Surface Systems Concepts, Washington, DC, 25-27 Feb. 2009.

Cummings, M.L., Brzezinski, A.S., and Lee, J.D., "The Impact of Intelligent Aiding for Multiple Unmanned Aerial Vehicle Schedule Management", *IEEE Intelligent Systems: Special Issue on Interacting with Autonomy*, 22(2), 2007, pp. 52-59.

Dowding, J., Personal communication, 22 Jan 2009.

"DuPont Kapton NH Polyimide Film Technical Data Sheet", *DuPont*, <http://www2.dupont.com/Kapton/en_US/assets/downloads/pdf/HN_datasheet.pdf>, 2010.

Essenburg, J. R., *Mission Planning and Navigation Support for Lunar and Planetary Exploration*, M.S. thesis, Massachusetts Institute of Technology, 2008.

Fanger, P. O., *Thermal Comfort*, New York: McGraw-Hill, 1970.

Fiala, D., Lomas K. J., Stohrer M., "A computer model of human thermoregulation for a wide range of environmental conditions: the passive system", *Journal of Appiled Physiology*, 87(5), 1999, pp. 1957-1972.

Gambino, M., "NASA's New Lunar Rover", *Smithsonian.com*, Smithsonian Institution, <*http://www.smithsonianmag.com/multimedia/photos/?c=y&articleID=78436247&page=3*> Jan. 2010.

Garrick-Bethell I., Byrne, S., Hoffman, J.A., and Zuber, M.T., "Areas of Favorable Illumination at the Lunar Poles Calculated from Topography", Abstract, 36th Lunar and Planetary Science Conference, Houston, TX, 14-18 March 2005.

Garry, B., "EVA Reports", *Mars Desert Research Station Crew 46 Log Book*, The Mars Society, <http://desert.marssociety.org/MDRS/fs05/0503/eva.asp>, 2006.

Gernhardt, M., "Extravehicular Activities (EVA) and Pressurized Rovers", in D. Cooke, G. Yoder, L. Leshin, and M. Gernhardt, *Exploration Systems Mission Directorate Lunar Architecture Update*, Presentation at the AIAA Space 2007 Conference and Exposition, Long Beach, CA, 18-20 Sept. 2007, pp. 53-57.

Gladstone, G.R., Stern, S.A., Retherford, K.D., Black, R.K., Slater, D.C., Davis, M.W., Versteeg, M.H., Persson, K.B., Parker, J.W., Kaufmann, D.E., Egan, A.F., Greathouse, T.K., Feldman, P.D., Hurley, D., Pryor, W.R., Hendrix, A.R., "LAMP: The Lyman Alpha Mapping Project on NASA's Lunar Reconnaissance Orbiter Mission", *Space Science Review, Space Science Review,* 150(1-4), 2010, pp. 161-181.

"Gnomonic", *Map Projections*, U.S. Geological Survey, <http://egsc.usgs.gov/isb/pubs/MapProjections/projections.html#gnomonic>, 2006.

Griffin, M.D., and French, J.R., *Space Vehicle Design*, 2nd ed., Available online through *Knovel*, Reston, VA: American Institute of Aeronautics and Astronautics, 2004.

Hart, P. E., Nilsson, N. J., and Raphael, B, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems, Science, and Cybernetics SSC4(2),* 1968, pp. 100-107.

Heiken, G. H., Vaniman, D. T., and French, B. M. (Eds.), *Lunar Sourcebook*, New York: Cambridge University Press, 1991.

Hong, S., *Design of Power Systems for Extensible Surface Mobility Systems on the Moon and Mars*, M.S. thesis, Massachusetts Institute of Technology, 2007.

Johnson, A. W., Newman, D. J., Waldie, J. M., Hoffman, J. A., "An EVA Mission Planning Tool based on Metabolic Cost Optimization", SAE 2009-01-2562, 39th International Conference on Environmental Systems, Savannah, GA, 12-16 July 2009.

Johnson, B. J., "Optimization of a Planning Support System for Planetary Exploration Extravehicular Activities", Paper presented to the Massachusetts Institute of Technology Summer Research Program, 2008.

7. References

Johnson, K., Ren, L., Kuchar, J., Oman, C., "Interaction of Automation and Time Pressure in a Route Replanning Task", Proceedings of the 2002 International Conference on Human-Computer Interaction in Aeronautics, Cambridge, MA, 23-25 Oct. 2002, pp. 132-137.

Johnston, R. S., and Hull, W. E., *Section 1, Chapter 2: Apollo Missions*, in R. S. Johnston, L. F. Dietlein, and C. A. Berry (Eds.), *Biomedical Results of Apollo,* NASA SP-368, Washington, DC: NASA, 1975, pp. 9-40.

Jones, E. M., "Climbing Cone Ridge – Where Are We?", *Apollo 14*, Apollo Lunar Surface Journal, <http://www.hq.nasa.gov/alsj/a14/a14.tocone.html>, 2010.

Jordan, N. C., *Multidisciplinary Spacesuit Modeling and Optimization: Requirement Changes and Recommendations for the Next-Generation Spacesuit Design*, M.S. thesis, Massachusetts Institute of Technology, 2006.

Kranz, J. H., "Aerial Perspective", *Sensation and Perception Tutorials*, Hanover College Psychology Department, <http://psych.hanover.edu/krantz/art/aerial.html>, 2005.

Kuznetz, L. H., "Designing a Smart Suit for the Moon and Mars", SAE 2008-01-1952, 38th International Conference on Environmental Systems, San Francisco, CA, 29 June - 3 July 2008.

Larson, W. J., and Pranke, L. K. (Eds.), *Human Spaceflight Mission Analysis and Design*, New York: McGraw-Hill, 2000.

Layton, C., Smith, P. J., McCoy, C. E., "Design of a Cooperative Problem-Solving System for En-Route Flight Planning: An Empirical Evaluation", *Human Factors*, 36(1), 1994.

Li, R., Yilmaz, A., Banks, M., Oman, C., Bhasin, K., "Prototype Development for a Lunar Astronaut Spatial Orientation and Information System", Presentation at the 2009 Lunar Science Forum, NASA Ames Research Center, Moffet Field, CA, 21-23 July 2009.

Lindqvist, L. V. J., *Multidisciplinary Extravehicular Activity Mission Optimization for Lunar Exploration*, M.S. thesis, Technishe Universität München, 2008.

"LOLA Fact Sheet", NASA Goddard Space Flight Center, <http://lunar.gsfc.nasa.gov/lola/images/LOLA_Fact_Sheet.pdf>, 2009.

"LPRP Overview & History", *Lunar Precursor Robotic Program*, NASA Marshall Space Flight Center, <http://moon.msfc.nasa.gov/background.html>, 2010.

"Lunar Mission Timeline", *Lunar Science and Exploration*, Lunar and Planetary Institute, <http://www.lpi.usra.edu/lunar/missions/>, 2010.

"Lunar Reconnaissance Orbiter", NASA Goddard Space Flight Center, <http://lunar.gsfc.nasa.gov/>, 2010.

Margaria, R., *Biomechanics and Energetics of Muscular Exercise*, Oxford, United Kingdom: Oxford University Press, 1976.

Márquez, J. J., *Human-Automation Collaboration: Decision Support for Lunar and Planetary Exploration*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

Márquez, J. J., and Cummings, M.L., "Design and Evaluation of Path Planning Decision Support for Planetary Surface Exploration", *Journal of Aerospace Computing, Information, and Communication*, 5(5), 2008, pp. 57-71.

Márquez, J.J., Cummings, M.L., Roy, N., Kunda, M., and Newman, D.J., "Collaborative Human-Computer Decision Support for Planetary Surface Traversal", AIAA 2005-6993, Infotech@Aerospace Conference, Arlington, VA, 26-29 Sept. 2005.

Márquez, J. J., and Newman, D. J., "Planning and Re-planning for Planetary Extravehicular Activities: Analysis of Excursions in a Mars-Analog Environment and Apollo Program", SAE 2006-01-2297, 36th International Conference on Environmental Systems, Norfolk, VA, 17-20 July 2006.

Márquez, J. J., and Newman, D. J., "Recommendations for Real-Time Decision Support Systems for Lunar and Planetary EVAs", SAE 2007-01-3089, 37th International Conference on Environmental Systems, Chicago, IL, 9-12 July 2007.

Masetti, M., "The Mars Orbiter Laser Altimeter", NASA Goddard Space Flight Center, <http://mola.gsfc.nasa.gov/>, 2007.

Mazarico, E., Neumann, G.A., Smith, D.E., Zuber, M.T., "Illumination Conditions in the Lunar Polar regions from Lunar Orbiter Laser Altimeter (LOLA) Data", Poster presented at the 2009 AGU Fall Meeting, San Francisco, CA, 14-18 Dec. 2009.

Mazarico, E., Neumann, G.A., Smith, D.E., Zuber, M.t., Torrence, M.H., "Illumination Conditions of the Lunar Polar Regions Using LOLA Topography", Submitted to *Icarus*, 2010.

McCurdy, M., "Planning Tools for Mars Surface Operations: Human-Computer Interaction Lessons Learned", 2009 IEEE Aerospace Conference, Big Sky, MT, 7-14 March 2009.

Mendell, W.W., "How cold are the floors of lunar polar shadowed craters?", Abstract, 41st Lunar and Planetary Science Conference, The Woodlands, TX, 1-5 March 2010.

"Mercury Laser Altimeter (MLA)", MESSENGER Education and Public Outreach, <http://www.messenger-education.org/instruments/mla.htm>, 2002.

Mosteller, R.D., "Simplified Calculation of Body Surface Area", Letter, *New England Journal of Medicine*, 317(17), 1987.

7. References

Muehlberger, W.R., "Apollo 16 Traverse Planning and Field Procedures", in G.E. Ulrich, C.A. Hodges, and W.R. Muehlberger (Eds.) *Geological Survey Professional Paper 1048: Geology of the Apollo 16 area, Central Lunar Highlands*, Washington, DC: U.S. Government Printing Office, 1981, pp. 10-20.

NASA, *Apollo 14 Mission Report* (*MSC-04112)*. Houston, TX: Manned Spacecraft Center, 1971.

NASA, "MOLA, MLA, and LOLA: Three Generations of Laser Altimeters", *Lunar Librarian Newsletter*, 1(3), <http://lunar.gsfc.nasa.gov/newsletter/LRO-Sept06.pdf>, 2006.

NASA, "Traverse Planning Parameters", *Apollo 17 Lunar Surface Procedures,* Houston, TX: NASA Manned Spacecraft Center, <http://history.nasa.gov/alsj/a17/a17lspSect4.4.pdf>, 1972.

Noda, H., Araki, H., Goossens, S., Ishihara, Y., Matsumoto, K., Tazawa, S., Kawano, N., Sasaki, S., "Illumination conditions at the lunar polar regions by KAGUYA (SELENE) laser altimeter", *Geophysical Research Letters*, 35, 2008.

Norris, J. S., Powell, M. W., Vona, M. V., Backes, P. G., Wick, J. V., "Mars Exploration Rover Operations with the Science Activity Planner", Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18-22 April 2005, pp. 4629-4634.

Oravetz C., *Human Estimation of Slope, Distance, and Height of Terrain in Simulated Lunar Conditions*, M.S. thesis, Massachusetts Institute of Technology, 2009.

Paige, D.A., Foote, M.C., Greenhagen, B.T., Schofield, J.T., Calcutt, S., Vasavada ,A.R., Preston, D.J., Taylor, F.W., Allen, C.C., Snook, K.J., Jakosky, B.M., Murray, B.C., Soderblom, L.A., Jau, B., Loring, S., Bulharowski, J., Bowles, N.E., Thomas, I.R., Sullivan, M.T., Avis, C., De Jong, E.M., Hartford, W., McCleese, D.J., "The Lunar Reconnaissance Orbiter Diviner Lunar Radiometer Experiment", *Space Science Review, Space Science Review,* 150(1-4), 2010, pp. 125-160.

Parasuraman, R., Riley, V., "Humans and Automation: Use, Misuse, Disuse, Abuse", *Human Factors*, 39(2), 1997, pp. 230-253.

Parasuraman, R., Sheridan, T.B., and Wickens, C.D., "A Model for Types and Levels of Human Interaction with Automation", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 30(3), 2000, pp. 286-297.

Patel, A., "Heuristics", *Game Programming*, Stanford University CS Theory Group, <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>, 2010.

Paulig, X., *Thermal Model of Space Suits*, M.S. thesis, Technishe Universität München, 2007.

Powell, D., "Moon's Magnetic Umbrella Seen as Safe Haven for Explorers", *Space.com*, TechMediaNetwork, <http://www.space.com/scienceastronomy/061114_reiner_gamma.html>, 14 Nov. 2006.

Ramos-Izquierdo, L., Scott III, V.S., Connelly, J., Schmidt, S., Mamakos, W., Guzek, J., Peters, C., Liiva, P., Rodriguez, M., Cavanaugh, J., Riris, H., "Optical System Design and Integration of the Lunar Orbiter Laser Altimeter", *Applied Optics*, 48(16), 2008, pp. 3035-3049.

Review of U.S. Human Spaceflight Plans Committee, *Seeking a Human Spaceflight Program Worth of a Great Nation*, Washington, DC: NASA, <http://www.nasa.gov/pdf/396093main_HSF_Cmte_FinalReport.pdf>, 2009.

RIACS News, "Mobile Agents used by Geologists in Volcanic Setting to Develop Operational Procedures for Lunar Missions", Research Institute for Advanced Computer Science, <http://www.riacs.edu/outreach/news/091108.jsp>, 11 Sept 2008.

Rupert S., and Boston, P., "Science Reports", *Mars Desert Research Station Crew 41 Log Book*, The Mars Society, <http://desert.marssociety.org/MDRS/fs05/0109/sci.asp>, <http://desert.marssociety.org/MDRS/fs05/0111/sci.asp>, 2006.

Santee, W. R., Allison, W. F., Blanchard, L. A., and Small, M. G., "A proposed model for load carriage on sloped terrain", *Aviation, Space, and Environmental Medicine,* 72(6), 2001.

Sheridan, T.B., and Verplank, W., *Human and Computer Control of Undersea Teleoperators*, Massachusetts Institute of Technology, Man-Machine Systems Laboratory, Department of Mechanical Engineering, 1978.

Sierhuis, M., Clancey, W. J., and van Hoof, R. J. J., "Brahms: a multi-agent modeling environment for simulating work processes and practices", *International Journal of Simulation and Process Modelling*, 3(3), 2007, pp. 134-152.

Smith, D.E., "Mars Orbiter Laser Altimeter (MOLA) Science Investigation", NASA Goddard Space Flight Center, Solar System Exploration Division, <http://ssed.gsfc.nasa.gov/tharsis/mola.html>, 2003.

Smith, D. E., Zuber, M. T., Jackson, G. B., Cavanaugh, J. F., Neumann, G. A., Riras, H., Sun, X., Zellar, R. S., Coltharp, C., Connelly, J., Katz, R. B., Kleyner, I., Liiva, P., Matuszeski, A., Mazarico, E. W., McGarry, J. F., Novo-Gradac, A., Ott, M. N., Peters, C., Ramos-Izquierdo, L. A., Ramsey, L., Rowlands, D. D., Schmidt, S., Scott III, V. S., Shaw, G. B., Smith, J. C., Swinski, J., Torrence, M. H., Unger, G., Yu, A. W., Zagwodzki, T. W., "The Lunar Orbiter Laser Altimeter Investigation on the Lunar Reconnaissance Orbiter Mission", *Space Science Review,* 150(1-4), 2010, pp. 209-241.

Smith, D.E., Zuber, M.T., Neumann, G.A., Lemoine, F.G., Mazarico, E., Torrence, M.H., McGarry, J.F., Rowlands, D.D., Head, J.W., Duxbury, T.H., Aharonson, O., Lucey, P.G., Robinson, M.S., Barnouin, O.S., Cavanaugh, J.F., Sun, X., Liiva, P., Mao, D., Smith, J.C., Bartels, A.E., "Initial Observations from the Lunar Orbiter Laser Altimeter (LOLA)", Submitted to *Geophysical Research Letters*, 2010.

7. References

Smith, D. E., Zuber, M. T., Torrence, M. H., McGarry, J. F., Pearlman, M., "Laser Ranging to the Lunar Reconnaissance Orbiter (LRO)", Presentation at the 15th International Laser Ranging Workshop, Canberra, Australia, 16-20 Oct. 2006.

Smith, P. J., McCoy, C. E., Layton, C., "Brittleness in the Design of Cooperative Problem-Solving Systems: The Effects on User Performance", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 27(3), 1997.

Spudis, P. D., "Moon", *World Book Online Reference Center*, World Book, Inc., <http://www.nasa.gov/worldbook/moon_worldbook.html>, 2004.

"Surface Operations", *Apollo 14 Mission*, Lunar and Planetary Institute, <http://www.lpi.usra.edu/lunar/missions/apollo/apollo_14/surface_opp/>, 2010.

Trevino, L., Copeland, R.J., Elliott, J.E., Weislogel, M., "Freeze Tolerant Radiator for Advanced EMU", SAE 2004-01-2263, 34th International Conference on Environmental Systems, Colorado Springs, CO, 19-22 July 2004.

Waligora, J. M., and Horrigan, D. J., *Section 2, Chapter 4: Metabolism and Heat Dissipation during Apollo EVA Periods*, In R. S. Johnston, L. F. Dietlein, and C. A. Berry (Eds.), *Biomedical Results of Apollo,* NASA SP-368, Washington, DC: NASA, 1975, pp. 115-128.

Wilkinson, N., "Implementation of a Traverse Generation Assistant for Long-Distance Pressurized Rover Expeditions", in C. S. Cockell (Ed.), *Martian Expedition Planning*, Science and Technology Series, Vol. 107, San Diego, CA: American Astronautical Society, 2004.

Wertz, J. R., and Larson, W. J. (Eds.), *Space Mission Analysis and Design*, 3rd ed., Hawthore, CA: Microcosm Press, and New York: Springer, 1999.

Wood, B. M., and Wood, Z. J., "Energetically Optimal Travel across Terrain: Visualizations and a New Metric of Geographic Distance with Archaeological Applications", SPIE 6060-15, 18th Annual Symposium on Electronic Imaging, San Jose, CA, 16-19 Jan. 2006.

Appendix A Pictorial Description of A* Graph Search Algorithm

Figure A.1 represents the grid of nodes over which the A* search takes place. Each node is represented by a circle and has a unique name *A* through *H,* or *S*. Node *S* is the starting node and node *G* is the goal node. The number in the lower half of each node is the value of the heuristic function at that particular node. The nodes are connected by straight lines, which are the edges. Each edge has a cost associated with traveling over it between consecutive nodes, which is represented by the number alongside the edge. This graph does not allow for diagonal movements, like the true SEXTANT graph does, for simplification's sake.

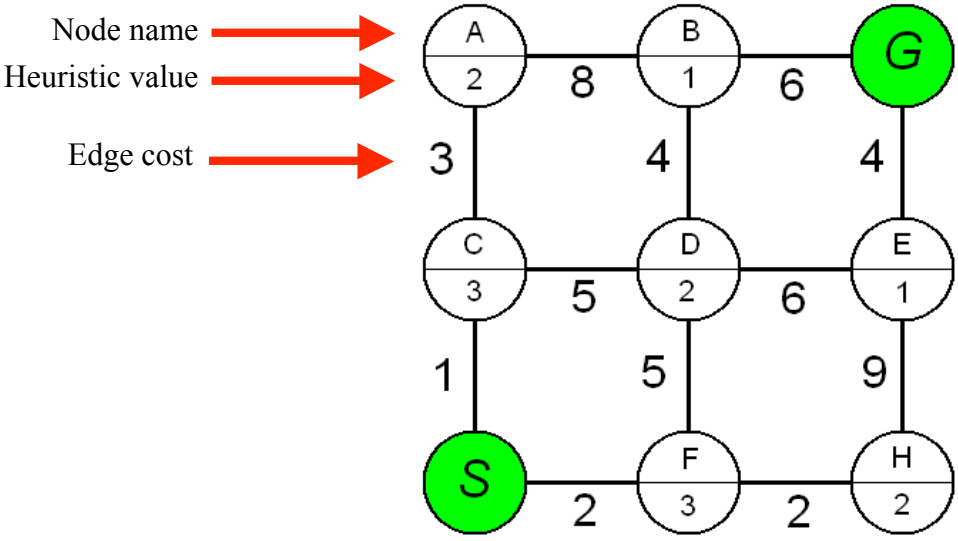


Figure A.1. Grid of nodes over which A* search occurs

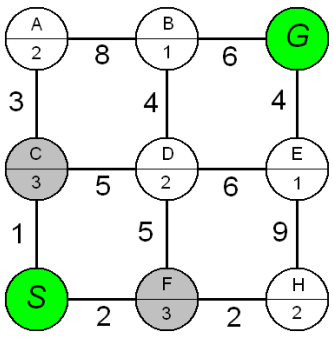


Figure A.2. A* Step one

The A* search begins by expanding the nodes adjacent from *S*, which are nodes *C* and *F*. The expanded nodes are marked in grey on Figure A.2. These nodes are put onto queue, along with the cost to move to them. As explained in Section 3.2, this cost to move to a general node *N* is a combination of two factors: the cumulative cost to travel from the starting node *S* to node *N* and the value of the heuristic at node *N*. After nodes *C* and *F* are expanded, the queue looks like:

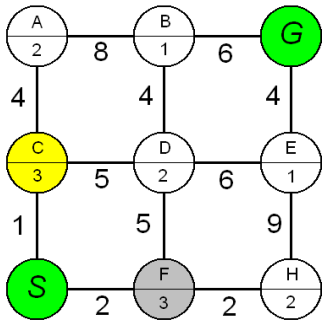


Figure A.3. A* step two

Node	Path	Travel Cost	Heuristic	Total Cost
C	S -> C	1	3	4
F	S -> F	2	3	5

The lowest-cost path, traveling from *S* to *C*, is removed from the queue and the A* search moves to node *C*. This is noted in yellow on Figure A.3.

Now, the A* search expands the nodes adjacent to *C*, which are nodes *A* and *D* (Figure A.4). The cumulative cost to move from node *S* to nodes *A* or *D* only takes into account the cost to move from *S* to *C*, and then from *C* to either *A* or *D*. The heuristic function for node *C* is no longer important. As nodes *A* and *D* are expanded, they are added to the queue still containing the path from *S* to *F*.

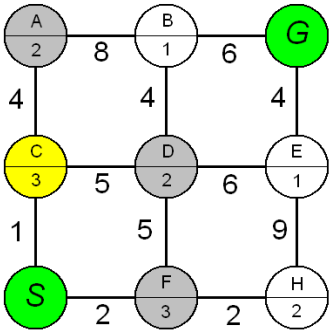


Figure A.4. A* step three

Node	Path	Travel Cost	Heuristic	Total Cost
F	S -> F	2	3	5
A	S -> C -> A	1 + 4 = 5	2	8
D	S -> C -> D	1 + 5 = 6	2	9

Even though nodes *A* and *B* were just expanded, the lowest-cost path on the queue is from *S* to *F*. So, this path is taken off the queue, and the A* search moves to node *F* (Figure A.5).

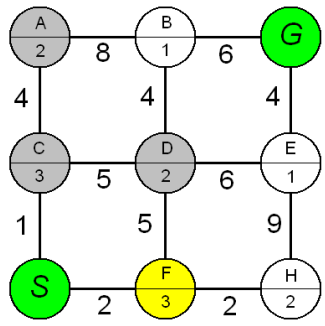


Figure A.5. A* step four

Now, the A* search expands the nodes adjacent to *F*, which are nodes *D* and *H* (Figure A.6). Node *H* has not been reached by any path, and so it is simply added to the queue. However, node *D* has already been reached by a different path – from node *C*. So, the cost of moving along the path *S* -> *F* -> *D* is compared to the cost of moving along the path *S* -> *C* -> *D*, and only the lowest-cost path is kept on the queue.

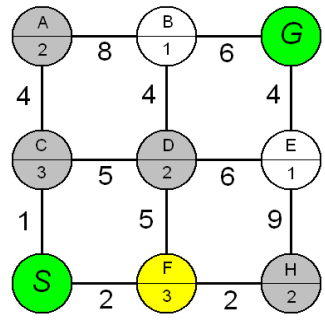


Figure A.6. A* step five

Node	Path	Travel Cost	Heuristic	Total Cost
A	S -> C -> A	1 + 4 = 5	2	7
D	S -> C -> D	1 + 5 = 6	2	8
D	S -> F -> D	2 + 5 = 7	2	9
H	S -> F -> H	2 + 2 = 4	2	6

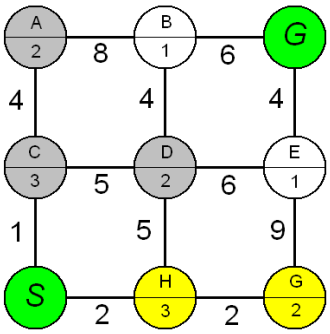


Figure A.6. A* step six

Node	Path	Travel Cost	Heuristic	Total Cost
A	S -> C -> A	1 + 4 = 5	2	7
D	S -> C -> D	1 + 5 = 6	2	8
H	S -> F -> H	2 + 2 = 4	2	6

So, node *D* will never be reached through node *F*. The lowest-cost path on the queue is from node *F* to node *H*, and so the A* search moves to *H* (Figure 7).

The A* search continues until the goal node *G* is reached. It is important to note that this process does not stop when the goal node is *expanded* – put on the queue. Rather, the search terminates when a path from *S* to *G* is removed from the queue because it is the lowest-cost path.

Appendix B iMAS Language Examples

Author: John Dowding, NASA Ames Research Center and UC Santa Cruz

User commands in plain text, *iMAS responses in italics*

B.1 General

Hello
    Yes, <Name>. I am listening
What time is it?
    The current time is 2 47 pm.
Increase Volume
    Increasing output volume to 60%
Decrease Volume
    Decreasing output volume to 40%
Shut up
    <falling tone>
Say that again

Go to sleep
    I am going to sleep now
Wake up
    I am waking up!

B.2 Activities

List activities
    You have 3 activities, Ingress, Work at Work Site 1, and Egress
Start my first activity
    Do you want to start your first activity?
    Yes
    <rising tone>
    Started monitoring activity Egress.
Start my next activity
    Do you want to start the next activity?
    Yes
    <rising tone>
    Started monitoring activity work at work site one
Start ingress activity
    Are you ready to start ingress activity?
    Yes
    <rising tone>
    Started monitoring activity ingress.
What is my current activity?

Your current activity is Ingress.
What is my next activity?
Your next activity is Egress.
How much time is left?
There are 15 minutes remaining in your current activity.
Extend duration by twenty minutes.
Do you want to extend current activity duration by twenty minutes?
Yes
<rising tone>
The duration for ingress activity is extended by twenty minutes.

B.3 Navigation and Automatic Associations

List locations
You have created 4 locations: EV1, Inside Hab, Habitat, and Work Site 1.
Where am I?
You are 99 meters west from habitat.
Where is work site one?
Work site 1 is 120 meters east from you.
Turn on automatic associations
Turning on automatic associations
Automatically associate with work site one.
Now automatically associating with work site one
Call this location work site two.
Do you want to declare the area within a radius of 10 meters work site two?
Yes
<rising tone>
The site is defined as work site two.
Now automatically associating with work site two.
Name this location work station one.
Do you want to name this location work station one?
Yes
<rising tone>
The station is defined as work station one.
Location work station one is associated with work site two.
List locations near me.
The following locations are within 20 meters of you: Work Site 2, Work Station 1.

B.4 Science Data

Record a voice note
Do you want to record a voice note?
Yes
Please begin recording voice note now:
This is a sample voice note taken at work site 2.
<rising tone>

Finished recording and labeled it voice note 1.
Voice note 1 is associated with location work site 1.

Record a voice log
Entering voice log now:
This is the 1st part of a voice log.
This is the 2nd part of a voice log
…
stop recording.
Leaving voice logging mode.
Finished recording and labeled it voice note 1.

Play voice note 1
Playing voice note: voice note 1.

Continue voice note 1.
Continue recording voice note 1.
This is the 2nd part of voice of that sample voice note.

Continue voice log 1.
Continue recording voice note now:
More of the voice log.
Stop recording.
Leaving voice logging mode.
Finished recording.

Create a sample bag?
Do you want to create sample bag 1?
Yes
<rising tone>
Sample bag was created and labeled sample bag 1.
Location work site 1 is associated with sample bag 1.

Download one image from my camera.
Do you want to download one image?
Yes
The most recent image was successfully downloaded from your camera and labeled image 1.
Location work site one is associated with image one.

Download all images from my camera.
Do you want to download all images?
Yes
5 images were downloaded from your camera and label image collection 1.
Location work site one is associated with image collection 1.

Associate image one with sample bag one.
Attempting to associate image 1 with sample bag 1.
Image 1 is associated with sample bag 1.

Label image collection one pictures of work site two.
Image collection 1 has been relabeled as image of work site two.

List voice notes
You have recorded one voice note: voice note 1.

List sample bags.

You have created one sample bag: sample bag 1.

List images

You have downloaded one image: image 1.

List image collections.

You have downloaded one image collection: image of work site two.

B.5 Metabolic Parameters

Consumable usage

O2 has 366 minutes remaining. Feed water has 401 minutes remaining.
Power has 400 minutes remaining. Scrubber has 420 minutes remaining.
Limiting consumable has 366 minutes remaining.

Cooling status

Your metabolic rate is 812 BTUs per hour. Your body heat storage is 4 BTUs.
Your LCVG inlet temp is 76 degrees farenheit, and the optimum is 64.

What is my EVA time?

You are 7 minutes into the e v a.

Feed water status

You have 431 minutes of feed water remaining.

Walk back check

The hab is 5.3 miles away. You have 295 minutes of life support left.
You must average 1.3 miles per hour to make it back safely.
Your average for the last 12 minutes has been 1.9 miles per hour.

What is my heart rate?

Your heart rate is 56 beats per minute.

Heat leak.

Your heat leak is 50 BTUs per hour.

What is my heat storage?

Your heat storage is -16 BTUs. You are thermally neutral.

Calories

You have burned 138 KCals.
Food bar consumption at 500 Kcal intervals is recommended.

What is my limiting consumable?

Your limiting consumable is feed water. It has 285 minutes remaining.

What is my met rate?

Your metabolic rate is 1041 BTUs per hour.

Oxygen status

You have 289 minutes of oxygen remaining.

O2 pressure

Your oxygen pressure is 756 PSI.

How much power do I have left?

You have 381 minutes of power remaining.

What is my PSIA?

Your suit pressure is 19.0 PSIA.

How much scrubber is left?

You have 360 minutes of scrubber remaining.

Suit leak check

You have a suit leak of -33.3 CC per minute. This is 113 percent better than spec.

Sweat

You have lost 0.0 pounds of sweat.

Drinking regularly prior to each half pound of sweat lost is strongly recommended.

Appendix C Distance, Time, and Energy Cost Data for Rover Traverse

C.1 Original Rover Traverse

Activity Point	Cumulative Distance (km)		Cumulative Time (HH:MM)		Cumulative Energy Cost (GJ)	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
1	0.00	0.00	0:00	1:00	0.00	5.32
2	13.83	----	1:55	----	11.86	----
3	45.64	----	4:02	----	27.28	----
4	63.91	----	5:15	----	35.94	----
5	80.63	----	6:22	----	43.90	----
6	103.90	103.90	7:55	8:25	54.90	57.55
7	112.26	112.26	8:59	9:59	61.51	66.83
8	128.73	----	11:04	----	74.61	----
9	137.56	----	11:40	----	78.80	----
10	153.71	153.71	12:44	13:44	86.44	91.76
11	172.20	----	14:58	----	100.48	----
12	203.09	----	17:02	----	115.21	----
13	224.67	224.67	18:28	18:58	125.39	128.05
14	234.31	234.31	19:36	20:21	132.61	136.60
15	244.05	244.05	21:00	22:00	141.18	146.49
16	262.39	262.39	23:14	26:14	155.12	171.07
17	274.64	274.64	27:03	27:33	176.83	179.49
18	314.87	314.87	30:13	30:43	198.79	201.45
19	327.94	----	31:36	----	207.67	----
20	347.49	347.49	32:54	34:24	217.07	225.05
21	364.92	----	35:34	----	233.31	----
22	370.64	370.64	35:56	37:26	236.01	243.99
23	389.70	----	38:43	----	253.03	----
24	416.06	416.06	40:28	41:28	265.59	270.91
25	446.00	446.00	43:28	43:58	285.11	287.76
26	468.91	468.91	45:29	45:44	298.58	299.91
27	489.04	489.04	47:05	47:20	309.40	310.73
28	533.50	----	50:17	----	331.78	----

C.2 First Modified Rover Traverse

Shaded values are those that have changed from the original rover traverse.

Activity Point	Cumulative Distance (km)		Cumulative Time (HH:MM)		Cumulative Energy Cost (GJ)	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
1	0.00	0.00	0:00	1:00	0.00	5.32
2	13.83	----	1:55	----	11.86	----
3	45.64	----	4:02	----	27.28	----
4	63.91	----	5:15	----	35.94	----
5	80.63	----	6:22	----	43.90	----
6	103.90	103.90	7:55	8:25	54.90	57.55
7	112.26	112.26	8:59	9:59	61.51	66.83
8	128.73	----	11:04	----	74.61	----
9	137.56	----	11:40	----	78.80	----
10	153.71	153.71	12:44	19:44	86.44	123.67
11	172.20	172.20	20:58	20:58	132.38	132.38
12	203.09	----	23:02	----	147.11	----
13	224.67	----	24:28	----	157.30	----
14	234.31	234.31	25:36	26:21	164.51	168.50
15	244.05	244.05	27:00	28:00	173.08	178.40
16	262.39	262.39	29:14	32:14	187.03	202.98
17	274.64	274.64	33:03	33:33	208.74	211.40
18	314.87	314.87	36:13	36:43	230.70	233.35
19	327.94	----	37:36	----	239.57	----
20	347.49	347.49	38:54	40:24	248.98	256.95
21	364.92	----	41:34	----	265.21	----
22	370.64	370.64	41:56	43:26	267.92	275.89
23	389.70	----	44:43	----	284.93	----
24	416.06	416.06	46:28	47:28	297.49	302.81
25	446.00	446.00	49:28	49:58	317.01	319.67
26	468.91	468.91	54:29	51:44	330.48	331.81
27	489.04	489.04	53:05	53:20	341.31	342.64
28	533.50	----	56:17	----	363.68	----

C.3 Second Modified Rover Traverse

Shaded values are those that have changed from the first modified rover traverse.

Activity Point	Cumulative Distance (km)		Cumulative Time (HH:MM)		Cumulative Energy Cost (GJ)	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
1	0.00	0.00	0:00	1:00	0.00	5.32
2	13.83	----	1:55	----	11.86	----
3	45.64	----	4:02	----	27.28	----
4	63.91	----	5:15	----	35.94	----
5	80.63	----	6:22	----	43.90	----
6	103.90	103.90	7:55	8:25	54.90	57.55
7	112.26	112.26	8:59	9:59	61.51	66.83
8	128.73	----	11:04	----	74.61	----
9	137.56	----	11:40	----	78.80	----
10	153.71	153.71	12:44	19:44	86.44	123.67
11	172.20	172.20	20:58	20:58	132.38	132.38
12	203.09	----	23:02	----	147.11	----
13	224.67	----	24:28	----	157.30	----
14	234.31	234.31	25:36	26:21	164.51	168.50
15	244.05	244.05	27:00	28:00	173.08	178.40
16	262.39	262.39	29:14	32:14	187.03	202.98
17	274.64	274.64	33:03	33:33	208.74	211.40
18	314.87	314.87	36:13	36:43	230.70	233.35
19	327.94	----	37:36	----	239.57	----
20	347.49	347.49	38:54	40:24	248.98	256.95
21	364.92	364.92	41:34	48:34	265.21	302.44
22	370.64	370.64	48:56	48:56	305.14	305.14
23	389.70	----	50:13	----	314.18	----
24	416.06	416.06	51:58	52:58	326.74	332.06
25	446.00	446.00	54:58	55:28	346.26	348.92
26	468.91	468.91	56:59	57:14	359.73	361.06
27	489.04	489.04	58:35	58:50	370.55	371.88
28	533.50	----	61:47	----	392.93	----

Appendix D SEXTANT MATLAB Code

As of May 20, 2010, this is the most recent version of SEXTANT. It is attached to this thesis for completeness. The most current version of SEXTANT can be obtained from Aaron Johnson by e-mailing him at aaronwj@mit.edu.

```
% SEXTANT   Version 4.0
% Aaron Johnson - Last edited May 20, 2010
% Mission planning interface employing a PATH-based optimization
%
% SEXTANT alone initializes a prompt for the user to load elevation data
%  from file. Elevation text files or Matlab data files may be used
%
% SEXTANT(ELEVMAP) loads the matrix ELEVMAP as the elevation data.
%
% SEXTANT(...,'lite') calls the 'lite' option, which uses simpler
%  surface rendering to speed plotting time and prevent problems on someF
%  machines.
%
% SEXTANT(...,'moon') calls the 'moon' option, where the user can load the
%  elevation map of the lunar south pole
%
%  Map information and all other mission data are entered on the following
%  menus. SEXTANT will then open a GUI for point-and-click waypoint
%  editing, terrain editing, traverse path optimization, and displaying
%  those paths along with all cost data.
%  Data files are written for an independent render engine.
%
%  Several mission scenarios may be loaded into multiple instances of
%  SEXTANT simultaneously.


% SEXTANT is a single function that iteratively calls itself with three
% parameters: PROGRESS, SELECT, and DATA. The parameter PROGRESS determines
% which section of code is to be executed, and the parameter SELECT
% determines which sub-section or option to execute when applicable. This
% is accomplished through SWITCH constructs, with the main progress switch
% beginning on Line 178. DATA is a structure holding all necessary
% application and mission data, which is passed and updated in each
% iterative call to SEXTANT.
% Any open SEXTANT GUI contains its current DATA structure in the
% 'UserData' property, and all GUIs are shielded from the command line.
% This allows multiple instances of SEXTANT to be run simultaneously.

function sextant(Progress,Select,Data)
%% ************ CHECK FUNCTION CALL & LOAD ELEVATION MAP ******************
if nargin <= 2  % This section only runs on the initial call to SEXTANT
switch nargin
   case 0
      calldata = 'LoadFromFile'; Data.Lite = '';  moon = '';
   case 1
      if ~ischar(Progress)
         calldata = 'Progress'; Data.Lite = '';  moon = '';
      elseif strcmp(Progress,'lite')
         calldata = 'LoadFromFile'; Data.Lite = '"lite"';  Data.moon = '';
      elseif strcmp(Progress,'moon')
         calldata = 'LunarMap'; Data.Lite = '';  moon = 'moon';
      else
         calldata = 'error';
      end
   case 2
      params = {'Progress','Select','error'}; Data.Lite = '"lite"';  moon = '';
      calldata = params{find(strcmp('lite',{Select,Progress,'lite'}),1)};
end
mfilepath = mfilename('fullpath');
cd(mfilepath(1:end-7))  % Change directory to that containing SEXTANT
[Data.Obst,Data.Soil,Data.SciR,Data.Othe,Data.hasWP] = deal(false);%Terrain cost map
                                          %boolean values
switch calldata
   case 'LunarMap'
%        Data.DIRbins='../Sun Illumination Downloads/Database'; % location of 'database'
%        Data.FURNSHfile='./spice/simple.furnsh'; % location of SPICE meta-kernel
```

```
%       Data.PATH_MICE='./mice';
%       pwddir=pwd;
%       addpath([pwddir,'/common']);
%
%       % start MICE ( for Sun positions, etc. )
%       Data.PATH_MICE='./mice';
%       addpath([Data.PATH_MICE,'/src/mice/']);
%       addpath([Data.PATH_MICE,'/lib/']);
%       Data.clktol='10:000';
%       cspice_kclear;
%       cspice_furnsh(Data.FURNSHfile);

%       % initialization

        %GRDtopo='./data/LOLA.data264km.gnomonic.bin';
        %fid=fopen(GRDtopo,'rb');
        %xyz=fread(fid,[3 inf],'double','ieee-le');
        %fclose(fid);

        %n=sqrt(size(xyz,2));
        %x=reshape(xyz(1,:),n,n);
        %y=reshape(xyz(2,:),n,n);
        %z=reshape(xyz(3,:),n,n);
        %clear xyz

        load('./database_SP_topo_gnomonic.mat','x','y','z')
        % Transpose z matrix so it's in the right orientation
        z = z';
        %surf(za(400:1900,400:1900))
        %shading interp

        x1 = input('Left-most x-coordinate (in km): ');
        x2 = input('Right-most x-coordinate (in km): ');
        if x2<=x1
           disp('Invalid entry')
           x2=input('Right-most x-coordinate (in km): ');
        end
        y1 = input('Top-most y-coordinate (in km): ');
        y2 = input('Bottom-most y-coordinate (in km): ');
        if y2>=y1
           disp('Invalid entry')
           y2=input('Bottom-most y-coordinate (in km): ');
        end

        % Left-, right-, top-, and bottom-most indices in the smaller map
        % Inclusive
        xIndices = 1100+floor([x1 x2]./.24);
        yIndices = 1100-floor([y1 y2]./.24);

        % Remember:  The y-direction is the rows and the x-direction is the
        % columns
        % Also, remember that you have transposed z
        Elevmap = z(yIndices(1):yIndices(2),xIndices(1):xIndices(2)).*1000;    % Elevation above 1737.4 km sphere (mean radius)
        % Left-, right-, top-, and bottom-most distances in the smaller map
        % Measured with respect to 0 in the center of the big map
        xLimits = -263.88+(xIndices-1)*.24;
        yLimits = 263.88-(yIndices-1)*.24;

        clear z
        %surf(Elevmap)
        %shading interp
        res = 240;
        utm = 0;
        ndt = -9999;
        xll = 0;    % Note:  This is the letter "l" and not the number "one"
        yll = 0;

    case 'LoadFromFile'  % Load elevation map from file
        [datafile,filepath] = uigetfile({'*.mat;*.txt',...
                        'Matlab data files or Elevation text files'},...
                        'Load Elevation Data...');
        if ~datafile, return, end  % Exit SEXTANT

        if strcmp(datafile(end-3:end),'.txt') % Load from Elevations text file
           Mapid = fopen([filepath,datafile],'r');
           if ~strcmp(fscanf(Mapid,'%s',1),'ncols') % Check if proper file
              message = ['Incorrect data or data type:\n\n',...
                    'Text file must be a PATH Elevations file.\n\n',...
                    'Example:  "EVA name"_Elevations.txt'];
              waitfor(warndlg(sprintf(message),'Load Elevation Data'))
```

```
            eval(['sextant(',Data.Lite,')'])  % Restart
            return
         end
         frewind(Mapid)
         Scandata1 = textscan(Mapid,'%*[^ ]%n',4);  % Read 4 data values
         hasUTM = strcmp(fscanf(Mapid,'%s',1),'UTMzone');  % UTM check
         fseek(Mapid,-8+hasUTM,'cof');
         Scandata2 = textscan(Mapid,'%*[^ ]%n',2+hasUTM); % Rest of data
         Mapdata = [Scandata1{1};Scandata2{1}];
         xll = Mapdata(3);
         yll = Mapdata(4);
         utm = hasUTM*round(Mapdata(5));  % Nonzero if hasUTM
         res = Mapdata(end-1);
         ndt = Mapdata(end);
         Elevmap = dlmread([filepath,datafile],'',...  % Read elevations
                   [6+hasUTM 0 Mapdata(2)+5+hasUTM Mapdata(1)-1]);
         fclose(Mapid);          % Check for matching terrain cost maps
         for tmap = {'Obstacles' 'SoilMech' 'SciReturn' 'Other'}
            try TCM.(tmap{1}) = dlmread([filepath,datafile(1:max(end-14,1)),...
                tmap{1},'.txt'],'',[6+hasUTM 0 Mapdata(2)+5+hasUTM Mapdata(1)-1]);
               Data.(tmap{1}(1:4)) = all(size(TCM.(tmap{1}))==size(Elevmap));
            catch  %#ok<CTCH>
            end
         end

      elseif strcmp(datafile(end-3:end),'.mat') % Load from Matlab data file
         Mapdata = load([filepath,datafile]);
         Vars = sort(fieldnames(Mapdata));
         if length(Vars) == 1
            Elevmap = Mapdata.(Vars{1});
         elseif length(Vars) >= 2
            elevi = listdlg('Name','Load Elevation Data',...
                     'ListString',Vars,...
                     'SelectionMode','single',...
                     'ListSize',[182 67],...
                     'PromptString',...
                     {'Please select the field',...
                      'containing Elevation data:'});
            if isempty(elevi)
               eval(['sextant(',Data.Lite,')'])  % Restart
               return
            end
            Elevmap = Mapdata.(Vars{elevi});
         end
         Check = whos('Elevmap');
         if isempty(Vars) || ~strcmp(Check.class,'double') || min(Check.size)<2
            message = ['Incorrect data or data type:\n\n',...
                     'Input must be an elevation map matrix.'];
            waitfor(warndlg(sprintf(message),'Load Elevation Data'))
            eval(['sextant(',Data.Lite,')'])  % Restart
            return
         end
         i=1;  mapinfo = {1,0,0,0,-9999,''};  % Default map values
         for mv = {'Resolution','xllcorner','yllcorner','UTMzone','NoData','moon',...
               'xIndices','yIndices','xLimits','yLimits'}
            if any(strcmp(mv{1},Vars))  % Check for existing map values
               mapval = Mapdata.(mv{1});
               %Check = whos('mapval');
               %if strcmp(Check.class,'double') && all(size(mapval)==1)
                  mapinfo{i} = mapval;
               %end
            end
            i = i+1;
         end
         [res,xll,yll,utm,ndt,moon,xIndices,yIndices,xLimits,yLimits] = mapinfo{:};  % Set map values
%           [res,xll,yll,utm,ndt,moon] = mapinfo{:};
         for tmap = {'Obstacles' 'SoilMech' 'SciReturn' 'Other'}
            if any(strcmp(tmap{1},Vars)) %Check for matching terrain cost maps
               [Termap,TCM.(tmap{1})] = deal(Mapdata.(tmap{1}));
               Check = whos('Termap');
               Data.(tmap{1}(1:4)) = any(strcmp(Check.class,{'double','logical'})) &&...
                                     all(size(Termap)==size(Elevmap));
            end
         end
         if any(strcmp('Waypoints',Vars)) % Check for existing waypoints
            [Wpts,Data.LoadWaypoints] = deal(Mapdata.Waypoints);
            Check = whos('Wpts');
            Data.hasWP = strcmp(Check.class,'cell') && Check.size(2)==2;
         end
```

```
      else                                % Incorrect filetype
         message = ['Incorrect file type:\n\n',...
                  'Elevation data file must be a\n',...
                  'Matlab (.mat) or Text (.txt) file.'];
         waitfor(warndlg(sprintf(message),'Load Elevation Data'))
         eval(['sextant(',Data.Lite,')'])  % Restart
         return
      end

   case {'Progress' 'Select'}  % Elevation map entered as an argument
      res = 1;  xll = 0;  yll = 0;  utm = 0;  ndt = -9999;  % Defaults
      Elevmap = eval(calldata);
      Check = whos('Elevmap');
      if ~strcmp(Check.class,'double') || min(Check.size)<2 || ndims(Elevmap)~=2
         message = ['Incorrect input argument:\n\n',...
                  'Argument must be an elevation map matrix.'];
         warndlg(sprintf(message),'Open Mission Planner')
         return  % Exit SEXTANT
      end

   otherwise  % Unrecognized parameter
      disp('Error: SEXTANT only accepts "lite" as a parameter option')
      return  % Exit SEXTANT
end
Data.Elev = true;
Progress = 'Initialize';    % Elevation map etc. loaded, go to 'Initialize'
end

switch Progress % This switch determines which code to execute in each call
%% ************ DEFAULT DIRECTORIES & DATA INITIALIZATION *****************
case 'Initialize'
Data.DIRbins='../Sun Illumination Downloads/Database'; % location of 'database'
Data.FURNSHfile='./spice/simple.furnsh'; % location of SPICE meta-kernel
Data.PATH_MICE='./mice';
pwddir=pwd;
addpath([pwddir,'/common']);

% start MICE ( for Sun positions, etc. )
Data.PATH_MICE='./mice';
addpath([Data.PATH_MICE,'/src/mice/']);
addpath([Data.PATH_MICE,'/lib/']);
Data.clktol='10:000';
cspice_kclear;
cspice_furnsh(Data.FURNSHfile);

Data.Work_dir = pwd;            % Working directory containing sextant.m
macpc = {'/','C:\'};
Data.Render_dir = [macpc{1+ispc},'Content'];  % Root for render engine
% The three lines below give trouble sometimes
% If they do, just comment them out (provided you have a
% "Traverse_Coordinates" folder in your Content folder)
for dir = {Data.Render_dir, [Data.Work_dir,macpc{1+ispc}(end),'Traverse_Coordinates']}
   if ~exist(dir{1},'file'), mkdir(dir{1}), end
end
Elevmap(Elevmap==ndt) = NaN;   % Recognize "no data" values
Data.Elevations = Elevmap;     % Elevation data, loaded in the section above
[Data.Rows,Data.Cols] = size(Elevmap);
Data.Resolution = res;         % Map values assigned in section above
Data.xllcorner = xll;
Data.yllcorner = yll;
Data.UTMzone = utm;
Data.NoData = -9999;           % Value entered in saved maps for no data
Data.EVAname = 'EVA1';         % Default mission name
Data.Explorers{1} = 'Astronaut';%Explorer type: Astronaut or Rover
Data.NumExp = 1;               % Number of explorers
Data.MaxSlope = 15;            % Maximum traversable slope
Data.moon = moon;
Data.electronicsPower = 1400;    % W
if ~isempty(Data.moon)
   Data.Planet = 2;
   Data.xIndices = xIndices;
   Data.yIndices = yIndices;
   Data.xLimits = xLimits;
   Data.yLimits = yLimits;
else
   Data.Planet = 1;              % 1: 'earth', 2: 'moon', 3: 'mars'
end
Data.Simplifying = 'SimpON';
Data.Weight(1) = 120;           % Suited explorer mass in kg
%Data.CurrentTime = now;
```

Appendices

```
Data.CurrentTime = 7.342934166666666e+005;
Datenow = datevec(Data.CurrentTime);        % Default time is right now
Data.Month = Datenow(2);
Data.Day = Datenow(3);
Data.Year = Datenow(1);
Data.Hour = Datenow(4);
Data.Minute = Datenow(5);
Data.TimeZone = 8;  % Use 8 for EST, 6 for CST, 14 for MST, 16 for PST
Data.Scrsize = get(0,'ScreenSize');
% Added by Aaron 1/18
Data.Scale = 1;
Data.CostScale = 1;
for tmap = {'Obstacles' 'SoilMech' 'SciReturn' 'Other'}
   if Data.(tmap{1}(1:4)) %Existing terrain cost maps found in section above
      TCM.(tmap{1}) = round(max(TCM.(tmap{1}),0)); %Clean up existing maps
      Data.(tmap{1}) = min(TCM.(tmap{1}),1+3*(~strcmp(tmap{1},'Obstacles')));
   end
end
Data.Callback = false;  % Indicates if call is made from SEXTANT GUI
Data.Path = false;      % Indicates if command to run PATH was madef
sextant('Map',0,Data)                              % Call to 'Map'

%% ************ MAP INFO MENU ***********************************************
case 'Map'
switch Select
   case 0  % Initialize Map Information menu
      MapIn = figure('Name','SEXTANT',...
            'Position',[round(Data.Scrsize(3)/4) round(Data.Scrsize(4)/3) 340 300],...
            'Color',[.92549 .913725 .847059],...
            'Resize','off',...
            'IntegerHandle','off',...
            'DockControls','off',...
            'MenuBar','none',...
            'NumberTitle','off',...
            'CloseRequestFcn','sextant("Map","Close",[])');

      utm = {'default',[1 1 1],'n/a',abs(Data.UTMzone)};
     % UIcontrols: {Handle,Style,Position,String,Value,...
     %              HorizontalAlignment,FontSize,BackgroundColor,Callback}
      Mui = {'na','text',[15 260 310 30],'Elevation Map Information',1,...
               'center',16,[.58824 .96078 .86275],'';...
            'na','text',[17 220 306 25],sprintf('The entered map is %d x %d',...
               Data.Rows,Data.Cols),1,'center',16,[1 1 1],'';...
            'na','text',[15 175 155 25],'Map Resolution:',1,...
               'left',16,'default','';...
            'ResolutionH','edit',[171 174 87 27],sprintf('%.3f',Data.Resolution),...
               1,'center',18,[1 1 1],'sextant("Map",1,[])';...
            'na','text',[260 175 65 25],'meters',1,...
               'left',16,'default','';...
            'na','text',[40 130 150 20],'UTM Zone (1-60):',1,...
               'left',14,'default','';...
            'UTMzoneH','edit',[195 126 40 25],utm{3+(Data.UTMzone~=0)},...
               1,'center',14,[1 1 1],'sextant("Map",4,[])';...
            'UTMnsH','popup',[238 150 57 1],{'North','South'},...
               1+(Data.UTMzone<0),'left',10,utm{1+(Data.UTMzone~=0)},'';...
            'na','text',[15 100 135 15],'Lower-left X-coordinate:',1,...
               'left',10,'default','';...
            'xllcornerH','edit',[15 80 145 20],sprintf('%.5f',Data.xllcorner),1,...
               'center',12,utm{1+(Data.UTMzone~=0)},'sextant("Map",2,[])';...
            'na','text',[180 100 135 15],'Lower-left Y-coordinate:',1,...
               'left',10,'default','';...
            'yllcornerH','edit',[180 80 145 20],sprintf('%.5f',Data.yllcorner),1,...
               'center',12,utm{1+(Data.UTMzone~=0)},'sextant("Map",3,[])';...
            'na','push',[15 16 144 50],'Continue',0,...
               'center',18,'default','sextant("Map","Continue",[])';...
            'NewmapH','push',[161 16 106 50],'New map...',0,...
               'center',14,'default','sextant("Map","NewMap",[])';...
            'na','push',[269 16 56 50],'Quit',0,...
               'center',14,'default','sextant("Map","Close",[])'};
      for n = 1:size(Mui,1)         % Create the GUI using the info above
         Data.(Mui{n,1}) = uicontrol('Style',Mui{n,2},...
                        'Position',Mui{n,3},...
                        'String',Mui{n,4},...
                        'Value',Mui{n,5},...
                        'HorizontalAlignment',Mui{n,6},...
                        'FontSize',Mui{n,7},...
                        'BackgroundColor',Mui{n,8},...
                        'CallBack',Mui{n,9});
      end
      Data.UTMzone = abs(Data.UTMzone); % < 0 UTM interpreted as South Hem
```

```
      Data.Do = {};  % Tasks to perform when called from Mission GUI
      if Data.Callback
         set(Data.NewmapH,'Enable','off')
         if Data.Planet ~= 1
            set([Data.UTMzoneH,Data.xllcornerH,Data.yllcornerH,Data.UTMnsH],...
               'Enable','off')
         end
      end
      set(MapIn,'UserData',Data,...              % Set GUI to store Data
            'HandleVisibility','callback')
                                                 % GUI controls
   case {1 2 3 4}  % Edit data entries
      % Select is the name of the variable that we're switching
      % 1 = Resolution
      % 2 = x11corner
      % 3 = y11corner
      % 4 = UTMzone
      MapIn = gcf;
      Data = get(MapIn,'UserData');
      Vars = {'Resolution', .001, 199.999, '%.3f';...  %{Variable,min,max,format}
            'xllcorner', 0, 9999999.99999, '%.5f';...
            'yllcorner', 0, 9999999.99999, '%.5f';...
            'UTMzone', 0, 61, '%.0f'};
      Newvalue = sscanf(get(Data.([Vars{Select,1},'H']),'String'),'%f');
      % sscanf = read string under format control
      if ~isempty(Newvalue)  % If value too small or big, set to min/max
         Data.(Vars{Select,1}) = min(max(Newvalue,Vars{Select,2}),Vars{Select,3});
         Data.Do = {Data.Do{:},'NewMaps','ClrPaths'};
         if Select==1  % Resolution edit ==> Don't use existing Obs(tacles)
            Data.Obst = false;
            Data.Do = {Data.Do{:},'Scale','Obs'};
         end
      end
      set(Data.([Vars{Select,1},'H']),'String',...
         sprintf(Vars{Select,4},Data.(Vars{Select,1})))
      if Select==4  % UTM zone edit
         if Data.UTMzone >= 1 && Data.UTMzone <= 60
            Data.UTMzone = round(Data.UTMzone);
            set([Data.xllcornerH,Data.yllcornerH,Data.UTMnsH],...
               'BackgroundColor',[1 1 1])
         else
            Data.UTMzone = 0;
            set(Data.UTMzoneH,'String','n/a')
            set([Data.xllcornerH,Data.yllcornerH,Data.UTMnsH],...
               'BackgroundColor','default')
         end
      end
      set(MapIn,'UserData',Data)

   case {'Continue' 'NewMap' 'Close'}  % Buttons
      MapIn = gcf;
      Data = get(MapIn,'UserData');
      Data.UTMzone = Data.UTMzone*(3-2*get(Data.UTMnsH,'Value')); % +N,-S
      delete(MapIn)
      if Data.Callback
         task = {Data.Do,{}};                    % Call to 'Update'
         sextant('Update',task{1+strcmp(Select,'Close')},Data)
      elseif strcmp(Select,'Continue')
         sextant('Input',0,Data)                 % Call to 'Input'
      elseif strcmp(Select,'NewMap')
         eval(['sextant(',Data.Lite,')'])  % Restart
      end
end

%% ************ EVA INPUT MENU ********************************************
case 'Input'
switch Select
   case 0  % Initialize EVA Input menu
      chkbx = 25*(Data.Obst+(Data.Soil+Data.SciR+Data.Othe+Data.hasWP)*~Data.Callback);
      PathIn = figure('Name','SEXTANT',...
            'Position',[round(Data.Scrsize(3)/4) round(Data.Scrsize(4)/8) 400 525+chkbx],...
            'Color',[.92549 .913725 .847059],...
            'Resize','off',...
            'IntegerHandle','off',...
            'DockControls','off',...
            'MenuBar','none',...
            'NumberTitle','off',...
            'CloseRequestFcn','sextant("Input","Close",[])');

      macpc = {'/','\'};    % Append / or \
```

Appendices

```
  Data.Work_dir = regexprep(Data.Work_dir,'\w$',['$0',macpc{1+ispc}]);
  Data.Render_dir = regexprep(Data.Render_dir,'\w$',['$0',macpc{1+ispc}]);
% UIcontrols: {Handle,Style,Position,String,Value,...
%              HorizontalAlignment,FontSize,BackgroundColor,Callback}
  Gui = {'na','text',[15 485+chkbx 370 30],'SEXTANT Traverse Input',...
         1,'center',16,[.58824 .96078 .86275],'';...
       'na','text',[15 440+chkbx 135 25],'Name of EVA:',1,...
         'left',16,'default','';...
       'EVAnameH','edit',[155 440+chkbx 230 25],Data.EVAname,1,...
         'left',16,[1 1 1],'sextant("Input","Name",[])';...
       'MaxSlopeH1','text',[15 395 105 25],'Max Slope:',1,...
         'left',16,'default','';...
       'MaxSlopeH','edit',[127 395 48 25],Data.MaxSlope,1,...
         'center',16,[1 1 1],'sextant("Input",1,[])';...
       'MaxSlopeH2','push',[175 407 15 15],'+',0,...
         'center',12,'default','sextant("Input","MxSlp",[])';...
       'MaxSlopeH3','push',[175 393 15 15],'-',0,...
         'center',12,'default','sextant("Input","MxSlm",[])';...
       'na','text',[15 323 75 25],'Planet:',1,...
         'left',16,'default','';...
       'na','text',[105 345 50 25],'Earth',1,...
         'left',14,'default','';...
       'PlanetH1','radio',[165 345 25 25],'',Data.Planet==1,...
         'left',14,'default','sextant("Input",11,[])';...
       'na','text',[105 320 50 25],'Moon',1,...
         'left',14,'default','';...
       'PlanetH2','radio',[165 320 25 25],'',Data.Planet==2,...
         'left',14,'default','sextant("Input",12,[])';...
       'na','text',[105 295 50 25],'Mars',1,...
         'left',14,'default','';...
       'PlanetH3','radio',[165 295 25 25],'',Data.Planet==3,...
         'left',14,'default','sextant("Input",13,[])';...
       'na','text',[15 245 115 25],'Mass (kg):',1,...
         'left',16,'default','';...
       'WeightH','edit',[127 245 53 25],Data.Weight(end),1,...
         'center',16,[1 1 1],'sextant("Input",2,[])';...
       'na','text',[210 395 53 25],'Date:',1,...
         'left',16,'default','';...
       'MonthH','edit',[263 395 27 25],Data.Month,1,...
         'center',16,[1 1 1],'sextant("Input",3,[])';...
       'na','text',[290 395 8 25],'/',1,...
         'center',16,'default','';...
       'DayH','edit',[298 395 27 25],Data.Day,1,...
         'center',16,[1 1 1],'sextant("Input",4,[])';...
       'na','text',[325 395 8 25],'/',1,...
         'center',16,'default','';...
       'YearH','edit',[333 395 52 25],Data.Year,1,...
         'center',16,[1 1 1],'sextant("Input",5,[])';...
       'na','text',[210 345 55 25],'Time:',1,...
         'left',16,'default','';...
       'HourH','edit',[275 345 35 25],sprintf('%d%d',zeros(Data.Hour==0),Data.Hour),...
         1,'center',16,[1 1 1],'sextant("Input",6,[])';...
       'na','text',[310 345 8 25],':',1,...
         'center',16,'default','';...
       'MinuteH','edit',[320 345 35 25],sprintf('%d%d',zeros(Data.Minute<10),Data.Minute),...
         1,'center',16,[1 1 1],'sextant("Input",7,[])';...
       'na','text',[210 295 110 25],'Time Zone:',1,...
         'left',16,'default','';...
       'TimeZoneH','listbox',[210 239 175 56],{' Alaska Daylight',...
         ' Alaska Standard',' Atlantic Daylight',...
         ' Atlantic Standard',' Central Daylight',...
         ' Central Standard',' Eastern Daylight',...
         ' Eastern Standard',' HawaiiAleutian Daylt.',...
         ' HawaiiAleutian Std.',' Newfoundland Std.',...
         ' Newfoundland Daylt.',' Mountain Daylight',...
         ' Mountain Standard',' Pacific Daylight',...
         ' Pacific Standard'},Data.TimeZone,...
         'left',11,[1 1 1],'sextant("Input","Tzone",[])';...
       'ExplorerH1','toggle',[20 194 120 25],'Astronaut',strcmp(Data.Explorers{end},...
         'Astronaut'),'center',14,'default','sextant("Input","Astronaut",[])';...
       'ExplorerH2','toggle',[140 194 120 25],'On Rover',strcmp(Data.Explorers{end},...
         'Rover'),'center',14,'default','sextant("Input","Rover",[])';...
       'ExplorerH3','toggle',[260 194 120 25],'Robot',strcmp(Data.Explorers{end},...
         'Robot'),'center',14,'default','sextant("Input","Robot",[])';...
       'na','text',[90 149 165 25],'Explorer Number:',1,...
         'left',16,'default','';...
       'ExpNumH','popup',[260 175 45 1],1:Data.NumExp,Data.NumExp,...
         'center',14,[1 1 1],'sextant("Input","SelExp",[])';...
       'na','push',[255 80 125 45],'Add Explorer',0,...
         'center',14,'default','sextant("Input","AddExp",[])';...
```

```
        'StartH','push',[20 80 235 45],'START',0,...
          'center',18,'default','sextant("Input","Start",[])';...
        'na','text',[15 40 120 18],'Render Directory:',1,...
          'left',11,'default','';...
        'Render_dirH','edit',[15 20 318 20],Data.Render_dir,1,...
          'left',10,[1 1 1],'sextant("Input",21,[])';...
        'na','push',[334 18 51 25],'Browse',0,...
          'center',10,'default','sextant("Input",22,[])'};...
  for tmap = {'Obstacles' 'SoilMech' 'SciReturn' 'Other'}
    if Data.(tmap{1}(1:4)) %Choice to use existing terrain cost maps (from previous runs)
      Gui = vertcat(Gui,{'na','text',[50 409+chkbx 300 25],...
        ['Use existing ',tmap{1},' map:'],1,'left',16,'default','';...
      [tmap{1}(1:4),'H'],'check',[330 407+chkbx 25 25],'',Data.Callback,'left',...
        16,'default',['sextant("Input","',tmap{1}(1:4),'",[])']});
      Data.(tmap{1}(1:4)) = Data.Callback; %Default not use existing maps
      if Data.Callback, break, end  % On callback only give Obst box
      chkbx = chkbx-25;
    end
  end
  if Data.hasWP && ~Data.Callback
    % Data.hasWP == true if there are already waypoints
    Gui = vertcat(Gui,{'na','text',[50 409+chkbx 300 25],...
      'Load existing Waypoints:',1,'left',16,'default','';...
    'hasWPH','check',[330 407+chkbx 25 25],'',0,...
      'left',16,'default','sextant("Input","hasWP",[])'});
    Data.hasWP = false;  % Default is to not use existing waypoints
  end
  for n = 1:size(Gui,1)        % Create the GUI using the info above
    Data.(Gui{n,1}) = uicontrol('Style',Gui{n,2},...
                'Position',Gui{n,3},...
                'String',Gui{n,4},...
                'Value',Gui{n,5},...
                'HorizontalAlignment',Gui{n,6},...
                'FontSize',Gui{n,7},...
                'BackgroundColor',Gui{n,8},...
                'CallBack',Gui{n,9});
  end
  Data.Do = {};  % Tasks to perform when called from Mission GUI
  if Data.Callback
    set([Data.MaxSlopeH,Data.MaxSlopeH1,Data.MaxSlopeH2,...
         Data.MaxSlopeH3],'Enable','off')
    set(Data.StartH,'String','Continue')
  end
  set(PathIn,'UserData',Data,...              % Set GUI to store Data
        'HandleVisibility','callback')
                                              % GUI controls
case {1 2 3 4 5 6 7}  % Edit data entries
  PathIn = gcf;
  Data = get(PathIn,'UserData');
  Vars = {'MaxSlope', 0, 90;...   % {Variable name, min, max}
        'Weight', 1, 9999;...
        'Month', 1, 12;...
        'Day', 1, 31;...
        'Year', 2008, 2030;...
        'Hour', 0, 23;...
        'Minute', 0, 59};   % i = Explorer# for Weight, 1 otherwise
  i = (Select==2)*(get(Data.ExpNumH,'Value')-1) + 1;
  Newvalue = sscanf(get(Data.([Vars{Select,1},'H']),'String'),'%f');
  if ~isempty(Newvalue)  % If value too small or big, set to min/max
    Newvalue = min(max(Newvalue,Vars{Select,2}),Vars{Select,3});
    Data.(Vars{Select,1})(i) = round(Newvalue);
    if Select == 1
      Data.Do = {Data.Do{:},'ClrPaths','Obs','NewMaps'};
    elseif Select == 2
      Data.Do = {Data.Do{:},i};
    else
      Data.Do = {Data.Do{:},'ClrPaths','Sun'};
    end
  end
  z = zeros(((Select==6 && Data.Hour==0) ||...   % Prepends a zero to
        (Select==7 && Data.Minute<10)),1);  % Hr/Min when needed
  set(Data.([Vars{Select,1},'H']),'String',...
        sprintf('%d%d',z,Data.(Vars{Select,1})(i)))
  set(PathIn,'UserData',Data)

case 'Name'  % Edit EVA name
  PathIn = gcf;
  Data = get(PathIn,'UserData');
  Newname = get(Data.EVAnameH,'String');
  Newname = strrep(Newname,' ','');  % Eliminate spaces
```

```
    if ~isempty(Newname) && all(isstrprop(Newname,'alphanum')+(Newname=='_'))
       Data.EVAname = Newname;  % Must be alphanumeric or underscore
       Data.Do = {Data.Do{:},'NewFiles'};
    end
    set(Data.EVAnameH,'String',Data.EVAname)
    set(PathIn,'UserData',Data)

 case {'Astronaut' 'Rover' 'Robot'}  % Select explorer type
    PathIn = gcf;
    Data = get(PathIn,'UserData');
    en = get(Data.ExpNumH,'Value');
    Data.Explorers{en} = Select;
    set(Data.ExplorerH1,'Value',strcmp(Select,'Astronaut'))
    set(Data.ExplorerH2,'Value',strcmp(Select,'Rover'))
    set(Data.ExplorerH3,'Value',strcmp(Select,'Robot'))
    if strcmp(Select,'Rover')
       Data.Weight=500;
       set(Data.WeightH,'String',Data.Weight)
    else
       Data.Weight=120;
       set(Data.WeightH,'String',Data.Weight)
    end
    Data.Do = {Data.Do{:},en};
    set(PathIn,'UserData',Data)

 case {'MxSlp' 'MxSlm'}  % Slope increment & decrement buttons
    PathIn = gcf;
    Data = get(PathIn,'UserData');
    if strcmp(Select,'MxSlp')
       Data.MaxSlope = min(Data.MaxSlope+1,90);
    elseif strcmp(Select,'MxSlm')
       Data.MaxSlope = max(Data.MaxSlope-1,0);
    end
    set(Data.MaxSlopeH,'String',Data.MaxSlope)
    Data.Do = {Data.Do{:},'ClrPaths','Obs','NewMaps'};
    set(PathIn,'UserData',Data)

 case {11 12 13}  % Planet radio buttons
    PathIn = gcf;
    Data = get(PathIn,'UserData');
    Data.Planet = Select-10;
    set(Data.PlanetH1,'Value',Select==11)
    set(Data.PlanetH2,'Value',Select==12)
    set(Data.PlanetH3,'Value',Select==13)
    Data.Do = {Data.Do{:},'ClrPaths','Planet'};
    set(PathIn,'UserData',Data)

 case 'Tzone'  % Select time zone
    PathIn = gcf;
    Data = get(PathIn,'UserData');
    Data.TimeZone = get(Data.TimeZoneH,'Value');
    Data.Do = {Data.Do{:},'ClrPaths'};
    set(PathIn,'UserData',Data)

 case {21 22}  % Edit Render directory
    PathIn = gcf;
    Data = get(PathIn,'UserData');
    if Select==21                          % Manual edit
       Newdir = get(Data.Render_dirH,'String');
    else                                   % Browse button
       Newdir = uigetdir(Data.Render_dir,['Select directory for',...
                                  'external renderer data:']);
    end
    if ~isempty(Newdir) && ischar(Newdir)
       if ~exist(Newdir,'file')
          warndlg([Newdir,' is not a valid directory.'],...
                'Edit Render Directory')
       else
          macpc = {'/','\'};      % Append / or \
          Data.Render_dir = regexprep(Newdir,'\w$',['$0',macpc{1+ispc}]);
          Data.Do = {Data.Do{:},'NewFiles'};
       end
    end
    try set(Data.Render_dirH,'String',Data.Render_dir)
       set(PathIn,'UserData',Data), catch  %#ok<CTCH>
    end

 case {'SelExp' 'AddExp'}  % Select explorer & Add Explorer button
    PathIn = gcf;
    Data = get(PathIn,'UserData');
```

```
      if strcmp(Select,'AddExp')  % Add explorer
         Data.NumExp = Data.NumExp+1;
         set(Data.ExpNumH,'String',1:Data.NumExp,'Value',Data.NumExp)
         Data.Explorers{Data.NumExp} = Data.Explorers{Data.NumExp-1};
         Data.Weight(Data.NumExp) = Data.Weight(Data.NumExp-1);
         Data.Do = {Data.Do{:},'AddExp'};
      end
      en = get(Data.ExpNumH,'Value');
      set(Data.ExplorerH1,'Value',strcmp(Data.Explorers{en},'Astronaut'))
      set(Data.ExplorerH2,'Value',strcmp(Data.Explorers{en},'Rover'))
      set(Data.ExplorerH3,'Value',strcmp(Data.Explorers{en},'Robot'))
      set(Data.WeightH,'String',Data.Weight(en))
      set(PathIn,'UserData',Data)

   case {'Obst' 'Soil' 'SciR' 'Othe' 'hasWP'}  % Use existing maps toggles
      PathIn = gcf;
      Data = get(PathIn,'UserData');
      Data.(Select) = get(Data.([Select,'H']),'Value');
      if strcmp(Select,'Obst')
         state = {'on','off'};
         set([Data.MaxSlopeH,Data.MaxSlopeH1,Data.MaxSlopeH2,...
            Data.MaxSlopeH3],'Enable',state{1+Data.Obst})
      end
      set(PathIn,'UserData',Data)

   case 'Start'  % START button
      PathIn = gcf;
      Data = get(PathIn,'UserData');
      if exist([Data.Render_dir,Data.EVAname,'_Elevations.txt'],'file') &&...
        ~Data.Callback   % Change Data.Callback to false
         Choice = questdlg(sprintf('%s already exists.\n\nOK to overwrite?',...
               Data.EVAname),'Overwrite Mission...','Yes','No','No');
         if strcmp(Choice,'No'), return, end
      end
      delete(PathIn)
      Data.UTMzone = Data.UTMzone*(Data.Planet==1); % UTMzone 0 off earth
      if ~Data.Callback
         message = ['Calculating obstacles,\n',...
                  ' writing map files,\n',...
                  ' preparing surface...'];
         Data.calcmsg = helpdlg(sprintf(message),'SEXTANT');
         sextant('CostMaps',[],Data)            % Call to 'CostMaps'
      else
         sextant('Update',Data.Do,Data)          % Call to 'Update'
      end

   case 'Close'  % Close GUI, exit SEXTANT
      PathIn = gcf;
      Data = get(PathIn,'UserData');
      if ~Data.Callback
         Choice = questdlg(['Close ',Data.EVAname,' without saving?'],...
                     'Exit SEXTANT...','Yes','No','No');
         if strcmp(Choice,'Yes')
            delete(PathIn)
         end
      else
         delete(PathIn)
         sextant('Update',{},Data)               % Call to 'Update'
      end
end

%% ************ SET SLOPES, OBSTACLES, SOIL MECH, SCI RETURN, OTHER *******
case 'CostMaps'
[gx,gy] = gradient(Data.Elevations,Data.Resolution); % Calculates the gradient of the Data.Elevations matrix where Data.Resolution is the spacing between points
Data.Slopes = atan(sqrt(gx.^2+gy.^2))*(180/pi);  % Slopes in degrees
Data.Slop = true;
if ~Data.Obst       % Find obstacles (not using an existing obstacles map)
   Data.Obstacles = Data.Slopes > Data.MaxSlope; % 1 if obstacle, else 0
end
if all(all(Data.Obstacles))  % Check if entire map is obstacle
   try delete(Data.calcmsg), catch end  %#ok<CTCH>
   message = ['The map has no travereable terrain\n',...
            'and is entirely obstacles.\n\n',...
            'Increase the maximum slope.'];
   waitfor(warndlg(sprintf(message),'SEXTANT'))
   sextant('Input',0,Data)  % Restart back at 'Input'
   return
end
Data.Obst = true;
for tmap = {'SoilMech' 'SciReturn' 'Other'}
```

```
   if ~Data.(tmap{1}(1:4))       % If no map loaded, default map all zero
      Data.(tmap{1}) = zeros(Data.Rows,Data.Cols);
   end
end
sextant('SaveMaps','',Data)                    % Call to 'SaveMaps'

%% ************ SAVE MAP FILES *****************************************
case 'SaveMaps'
form = [ones(Data.Cols,1)*'%.4f ';'   \n'].';            % Write map files
for tmap = {'Elevations' 'Obstacles' 'Slopes' 'SoilMech' 'SciReturn' 'Other'} % Writes a separate text file for Elevations, Obstacles, Slopes, SoilMech, SciReturn, and
Other
   if (Data.(tmap{1}(1:4)) && ~Data.Path) || (Data.Path && Data.([tmap{1}(1:4),'Ed']))
      try save([Data.EVAname,'_Data'],'-struct','Data',tmap{1},Select), catch end %#ok<CTCH>
      cd(Data.Render_dir)
      for Outfile = {Data.EVAname,'Current'}
         mpf = fopen([Outfile{1},'_',tmap{1},'.txt'],'wt');
         fprintf(mpf,['ncols        %d\n',...
                  'nrows        %d\n',...
                  'xllcorner    %.8f\n',...
                  'yllcorner    %.8f\n'],...
                  Data.Cols,Data.Rows,Data.xllcorner,Data.yllcorner);
         if Data.UTMzone ~= 0
            fprintf(mpf,'UTMzone      %d\n',Data.UTMzone);
         end
         fprintf(mpf,['cellsize     %.3f\n',...
                  'NODATA_value  %d\n'],Data.Resolution,Data.NoData);
         fprintf(mpf,form,max(Data.(tmap{1}).',Data.NoData)); %Matrix, NoData=-9999
         fclose(mpf);
      end
      cd(Data.Work_dir)
   end
   form = [ones(Data.Cols,1)*'%d ';' \n'].';
   Select = '-append';
end
if ~Data.Path
   try save([Data.EVAname,'_Data'],'-struct','Data','Resolution','xllcorner',...
                  'yllcorner','UTMzone','NoData','Explorers',...
                  'moon','xIndices','yIndices','xLimits','yLimits','-append'), catch %#ok<CTCH>
   end
end
if ~Data.Callback
   sextant('Mission',0,Data)                   % Call to 'Mission'
end

%% ************ MISSION PLANNER GUI ****************************************
case 'Mission'
switch Select
   case 0  % Define waypoint, traverse path, and cost cell arrays
      [Data.Waypoints{1:Data.NumExp,1:2}] = deal([]);
      [Data.WayHandles{1:Data.NumExp,1:2}] = deal([]);
      [Data.Pathpoints{1:Data.NumExp,1:2}] = deal([]);
      [Data.PathHandles{1:Data.NumExp,1:2}] = deal([]);
      [Data.Distance{1:Data.NumExp,1:2}] = deal([]);
      [Data.MetCost{1:Data.NumExp,1:2}] = deal([]);
      [Data.Time{1:Data.NumExp,1:2}] = deal([]);
      [Data.Shadows{1:Data.NumExp,1:2}] = deal([]);
      [Data.TSuit{1:Data.NumExp,1:2}] = deal([]);
      [Data.StageSuitHeat{1:Data.NumExp,1:2}] = deal([]);
      [Data.stationaryTime{1:Data.NumExp,1:2}] = deal([]);
      [Data.ElevEd,Data.SlopEd,Data.ObstEd,...  % Info edited booleans
       Data.SoilEd,Data.SciREd,Data.OtheEd,Data.WayPEd] = deal(false);
      Data.R = '';          % Set to 'R' for "return home" path
      Data.TEsize = 1;      % Terrain edit size default
      Data.datadisp = 1;    % Determines which waypoint data to display
      Data.prevwp = [-1 -1]; % The previously selected waypoint coords
      Data.Callback = true;  % Calls to previous sections now "Callback"
                             % Develop terrain surface for GUI
      Data.Elmin = min(min(Data.Elevations)); % Minimum elevation
      Data.Eldiff = max(max(Data.Elevations))-Data.Elmin + 10^-3; % Maximum elevation
      Data.ColorLim = [Data.Elmin, Data.Elmin+Data.Eldiff*64/63;...
            Data.Elmin, Data.Elmin+Data.Eldiff*64/63; 0 2; 0 2; 0 2];
      Data.ColorObsRed = min(Data.Elevations+Data.Obstacles*10^6,...
                        Data.Elmin+Data.Eldiff*64/63);
      try delete(Data.calcmsg), catch end  %#ok<CTCH>
                             % Initialize SEXTANT GUI
      Data.MPfig = figure('Name',['SEXTANT:  ',Data.EVAname,],...
         'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/32),...
                  round(Data.Scrsize(3)*15/16) round(Data.Scrsize(4)*7/8)],...
         'Color',[.92549 .913725 .847059],...
         'IntegerHandle','off',...
```

```
    'DockControls','off',...
    'NumberTitle','off',...
    'Renderer','OpenGL',...
    'ResizeFcn','sextant("Mission","Resize",[])',...
    'CloseRequestFcn','sextant("Mission","Close",[])');
if isempty(strfind(system_dependent('getos'),'Vista'))
    set(Data.MPfig,'Pointer','fullcrosshair')
end

Data.MPaxes = axes('Units','pixels');          % Axes for surf plot
try                             % Surf plot with obstacles in red
    Data.MPsurf = surf(Data.MPaxes,0:Data.Cols-1,0:Data.Rows-1,...
        Data.Elevations,'CData',Data.ColorObsRed,...
        'FaceColor','interp',...
        'EdgeColor','none',...
        'ButtonDownFcn','sextant("Mission","Click",[])');
catch  %#ok<CTCH>
    delete(Data.MPfig)
    message = ['\n-----------------------------------------------',...
        '\nMatlab has encountered an error while trying\n',...
        'to create a surface rendering.\n\n',...
        'This is a bug caused by use of the HELP command.\n\n',...
        'Please exit and restart Matlab.\n',...
        '-----------------------------------------------\n'];
    error('Help:Figure_or_Axes',message)
end                             % Axes scaling (initially meters)
[xsize,ysize] = deal(Data.Resolution*Data.Cols,Data.Resolution*Data.Rows);
mapsize = max(xsize,ysize);
mag = floor(log10(mapsize));
scale = round(mapsize/10^mag)*10^(mag-1);
zaspect = Data.Resolution*min(1,10*Data.Eldiff/mapsize);
zmag = floor(log10(Data.Eldiff));
zscale = round(Data.Eldiff/10^zmag)*10^(zmag-1)*2;
set(Data.MPaxes,'YDir','reverse',...     % Set plot axes properties
        'View',[0 90],...
        'DataAspectRatio',[1 1 zaspect],...
        'CLim',Data.ColorLim(1,:),...
        'XLim',[0 Data.Cols-1],...
        'YLim',[0 Data.Rows-1],...
        'ZLim',[Data.Elmin-.01*Data.Eldiff, Data.Elmin+1.25*Data.Eldiff],...
        'XTick',(scale:scale:xsize)/Data.Resolution,...
        'YTick',(mod(ysize-.001,scale)+.001:scale:ysize)/Data.Resolution,...
        'ZTick',Data.Elmin:zscale:Data.Elmin+1.26*Data.Eldiff,...
        'XTickLabel',scale:scale:xsize,...
        'YTickLabel',ysize-(mod(ysize-.001,scale)+.001):-scale:0,...
        'ZTickLabel',0:zscale:1.25*Data.Eldiff,...
        'TickLength',[0 0],...
        'Color',[.92549 .913725 .847059])
if isempty(Data.Lite)   % This code does not execute in 'lite' mode
    % This code places the illumination on the surface
    % Needs to be updated
    set(Data.MPsurf,'FaceLighting','gouraud','BackFaceLighting','lit')
    material([.4 .8 0])        % Set surface reflectance properties
    Data.Sun = light('Position',[sin((Data.Hour+Data.Minute/60)*pi/12),...
        -cos((Data.Hour+Data.Minute/60)*pi/12),...
        .014+.006*sin((Data.Hour+Data.Minute/60)*pi/24)],...
        'Style','infinite');  % Illumination, varies by time
end
if Data.UTMzone~=0  % Display compass when lat/long is active
   y = Data.Rows/20;  x = Data.Cols-2-y/10;
   z = max(max(Data.Elevations(1:ceil(y*5/4),floor(end-y/2):end)))+.05*Data.Eldiff;
   line([x-y*9/32 x-y*9/32 x x],...
        [1+y*17/16 1+y*5/8 1+y*17/16 1+y*5/8],[z z z z],...
        'Color','k','LineWidth',2,'HitTest','off')
   line([x-y*9/32 x-y*9/64 x-y*9/64 x-y*9/64 x],...
        [1+y/4 1+y/16 1+y*9/16 1+y/16 1+y/4],[z z z z z],...
        'Color','k','LineWidth',2,'HitTest','off')
end
                                % Render mode colormaps
Mcolors = [[0 .3 .15; .9 .7 .4],[.2 .2 .25; .99 .99 1],[.25 .15 .1; 1 .8 .5]];
colors1 = zeros(63,9);                  % Terrain cost colormaps
Tcolors = [[.85 .85 .85; .5 .05 .1],[.85 .85 .85; .25 .1 .6],[.85 .85 .85; 0 0 .8]];
colors2 = zeros(64,9);
for i = 1:9
    colors1(:,i) = linspace(Mcolors(1,i),Mcolors(2,i),63);
    colors2(:,i) = linspace(Tcolors(1,i),Tcolors(2,i),64);
end
Data.Colors = [[colors1; .95 .02 .15 .95 .02 .15 .95 .02 .15], colors2];
colormap(Data.MPaxes,Data.Colors(:,3*Data.Planet-2:3*Data.Planet))
Data.Ecolors = {[0 0 1],[.9 .9 0],[.5 0 .5],[1 .5 0],[0 1 1],... % Explorer
```

Appendices

```
                [1 0 1],[.4 .2 0],[.8 .35 .35],[1 1 1],[0 0 0]}; % colors

  Data.MPmenu = uipanel('Units','pixels');    % Panel for UI controls
% UIcontrols: {Handle,Style,Position,String,Value,HorizontalAlignment,
%                FontSize,BackgroundColor,Enable,Callback}
  state = {'off','on'};
  apbutton = {'Specify AP','Stay at AP'};
  MPui = {'MPhelpH','push',[5 34 60 25],'HELP',0,'center',12,...
          'default','on','sextant("Mission","Help",[])';...
          %'sextant("Mission","Help",[])' is a call to
          % switch with the name "Help"
        'MPmapiH','push',[65 34 85 25],'Map Info',0,'center',12,...
          'default','on','sextant("Mission","Map",[])';...
        'MPevaiH','push',[150 34 85 25],'EVA Input',0,'center',12,...
          'default','on','sextant("Mission","Input",[])';...
        'na','text',[240 34 50 22],'Axes: ',1,'right',12,...
          'default','on','';...
        'MPscaleH','popup',[290 60 75 1],{'Meters','Km','Feet','Miles'},...
          1,'left',12,[1 1 1],'on','sextant("Mission","Scale",[])';...
        'na','frame',[370 31 297 32],'',1,'left',12,...
          'default','on','';...
        'MPwpmodeH','toggle',[374 34 90 25],'Waypoints',1,'center',...
          12,'default','on','sextant("Mission","WP",[])';...
        'MPexptxtH','text',[468 34 85 22],[Data.Explorers{1} ': '],1,'right',...
          12,'default','on','';...
        'MPexpnumH','popup',[553 60 40 1],1:Data.NumExp,1,...
          'center',12,[1 1 1],'on','sextant("Mission","SelExp",[])';...
        'MPwaytitleH','text',[598 34 64 23],'Start:',1,...
          'center',14,'default','on','';...
        'na','frame',[672 31 358 32],'',1,'left',12,...
          'default','on','';...
        'MPtermodeH','toggle',[676 34 65 25],'Terrain',0,'center',...
          12,'default','on','sextant("Mission","TER",[])';...
        'MPtertypeH','popup',[746 60 100 1],{'Obstacles','Soil mech','Sci return','Other'},...
          1,'left',12,[1 1 1],'on','sextant("Mission","TerON",[])';...
        'MPtersizetxtH','text',[851 34 45 22],'Size: ',1,'right',...
          12,'default','off','';...
        'MPtersizeH','edit',[896 34 30 25],1,1,'center',12,...
          [1 1 1],'off','';...
        'MPtersizepH','push',[926 47 15 15],'+',0,'right',12,...
          'default','off','sextant("Mission","TSp",[])';...
        'MPtersizemH','push',[926 32 15 15],'-',0,'right',12,...
          'default','off','sextant("Mission","TSm",[])';...
        'MPterOnH','toggle',[946 34 40 25],'ON',1,'center',12,...
          'default','on','sextant("Mission","TerON",[])';...
        'MPterOffH','toggle',[986 34 40 25],'OFF',0,'center',12,...
          'default','on','sextant("Mission","TerOFF",[])';...
        'na','push',[5 1 133 30],'Run PATH',0,'center',14,...
          'default','on','sextant("Mission","PATH",[])';...
        'na','text',[148 1 95 23],'Path         :',1,'left',14,...
          'default','on','';...
        'MPpathenH','edit',[195 1 40 25],'',1,'center',14,...
          [1 1 1],'inactive','';...
        'na','text',[250 1 50 23],'Dist: ',1,'right',14,...
          'default','on','';...
        'MPdistNH','edit',[300 1 82 25],'',1,'right',14,...
           [1 1 1],'inactive','';...
        'MPdistUH','edit',[385 1 60 25],'Meters',1,'center',12,...
           [1 1 1],'inactive','';...
        'na','text',[455 1 55 23],'Cost: ',1,'right',14,...
          'default','on','';...
        'MPcostNH','edit',[510 1 87 25],'',1,'right',14,...
          [1 1 1],'inactive','';...
        'MPcostUH','popup',[600 28 60 1],{'Kcal','BTU','KJ'},1,...
          'left',12,[1 1 1],'on','sextant("Mission","CostU",[])';...
        'na','text',[670 1 55 23],'Time: ',1,'right',14,...
          'default','on','';...
        'MPtimeH','edit',[725 1 70 25],'',1,'center',14,...
          [1 1 1],'inactive','';...
        'na','text',[805 1 65 22],'Render: ',1,'right',12,...
          'default','on','';...
        'MPrendmH1','toggle',[870 1 50 25],'Earth',Data.Planet==1,...
          'center',12,'default','on','sextant("Mission",1,[])';...
        'MPrendmH2','toggle',[920 1 50 25],'Moon',Data.Planet==2,...
          'center',12,'default','on','sextant("Mission",2,[])';...
        'MPrendmH3','toggle',[970 1 50 25],'Mars',Data.Planet==3,...
          'center',12,'default','on','sextant("Mission",3,[])';...
        % The three below have been added by me
        'na','frame',[1035 31 214 32],'',1,'left',12,...
          'default','on','';...
```

```
        'MPspecifyH','push',[1039 34 100 25],apbutton{strcmp(Data.moon,'moon')+1},0,'center',12,...
           'default','on','sextant("Mission","Specify",[])';...
        'na','text',[1035 1 95 22],'Simplifying: ',1,'right',12,...
           'default','on','';...
        'MPsimplifyingH1','toggle',[1130 1 40 25],'ON',1,'center',12,...
           'default','on','sextant("Mission","SimpON",[])';...
        'MPsimplifyingH2','toggle',[1170 1 40 25]','OFF',0,'center',12,...
           'default','on','sextant("Mission","SimpOFF",[])';...
        'MPpathpointsH','push',[1258 1 165 25],'Generate xml Output',0,'center',12,...
           'default',state{-strcmp(Data.moon,'moon')+2},'sextant("Mission","Points",[])';...
        'MPplotsH','push',[1146 34 100 25],'Plot Metrics',0,'center',12,...
           'default','on','sextant("Mission","Plots",[])';...
        'na','frame',[1254 31 173 32],'',1,'left',12,...
           'default','on','';...
        'MPshadowsH','push',[1258 34 165 25],'Determine Shadows',0,'center',12,...
           'default',state{strcmp(Data.moon,'moon')+1},'sextant("Mission","Shadows",[])'};
        %state{-strcmp(Data.moon,'moon')+1}
   for n = 1:size(MPui,1)         % Create the GUI using the info above
      Data.(MPui{n,1}) = uicontrol(Data.MPmenu,'Style',MPui{n,2},...
                  'Position',MPui{n,3},...
                  'String',MPui{n,4},...
                  'Value',MPui{n,5},...
                  'HorizontalAlignment',MPui{n,6},...
                  'FontSize',MPui{n,7},...
                  'FontWeight','bold',...
                  'BackgroundColor',MPui{n,8},...
                  'Enable',MPui{n,9},...
                  'CallBack',MPui{n,10});
   end
   set(Data.MPfig,'UserData',Data,...          % Set GUI to store Data
          'HandleVisibility','callback')

   if Data.hasWP                          % Load existing waypoints
      Data.WayPEd = true;
      hold on
      for en = 1:min(Data.NumExp,size(Data.LoadWaypoints,1))
         if ~isempty(Data.LoadWaypoints{en})
         Data.Waypoints{en} = [min(max(Data.LoadWaypoints{en}(:,1),0),Data.Cols-1),...
                        min(max(Data.LoadWaypoints{en}(:,2),0),Data.Rows-1)];
         for i = 1:size(Data.Waypoints{en},1)
            Data.Waypoints{en}(i,3) = Data.Elevations(Data.Waypoints{en}(i,2)+1,...
                                       Data.Waypoints{en}(i,1)+1);
         end
         Data.WayHandles{en}(1) = scatter3(Data.MPaxes,Data.Waypoints{en}(:,1),...
            Data.Waypoints{en}(:,2),Data.Waypoints{en}(:,3)+.05*Data.Eldiff,...
            120,Data.Ecolors{mod(en-1,10)+1},'filled',...
            'ButtonDownFcn',sprintf('sextant("Mission","Click",%d)',en));
         txt = {'H',{'H';num2str((1:i-1).')}};
         WPtxt = text(Data.Waypoints{en}(:,1),...
            Data.Waypoints{en}(:,2)-.007*Data.Rows,...
            Data.Waypoints{en}(:,3)+.07*Data.Eldiff,txt{1+(i>1)},...
            'HorizontalAlignment','center',...
            'VerticalAlignment','bottom',...
            'Color',[1 1 1],'FontWeight','bold','HitTest','off');
         Data.WayHandles{en} = [Data.WayHandles{en}(1);WPtxt];
         end
      end
      hold off
      set(Data.MPfig,'UserData',Data)   % Store the Waypoint data
      set(Data.MPexpnumH,'Value',en)
      sextant('Mission','SelExp',[]) % Select last explorer loaded
   end



                                       % GUI Controls
case 'Shadows'
   % Knowing the shadows
   % Is important in staying
   % Thermally comfy

   MPfig = gcf;
   Data = get(MPfig,'UserData');

   Data.DIRbins='../Sun Illumination Downloads/Database'; % location of 'database'
   Data.FURNSHfile='./spice/simple.furnsh'; % location of SPICE meta-kernel
   Data.PATH_MICE='./mice';

   isReturn = Data.R;
```

```
en = get(Data.MPexpnumH,'Value');
enR = en + ~isempty(isReturn)*Data.NumExp;

% Basic variables
Rmoon = 1737.4;
Rsun = 690000;
iobserver = '301';
itarget = '10';
refframe = 'MOON_PA';
latproj = -90;
lonproj = 0;

if ~isempty(Data.moon) && ~isempty(Data.Pathpoints{enR})
   disp(' ')
   disp('What is the binning for shadows? (in minutes)')
   shadowScaling = input('(Enter "0" to skip shadowing calculation): ');
   if shadowScaling == 0
      sextant('Mission','Plots','');
      return
   end
   shadowScaling = shadowScaling*60;              % seconds

   % Need to load normals, cartesian coordinates, and other
   % variables
   load('./database_SP_topo_normals.mat','nx','ny','nz');
   load('./database_SP_topo_cartesian.mat','xvec','yvec','zvec');
   load('./database_SP_general.mat','theta_vec2','dth','nth');
   % Transpose all of the matries so the indexes are correct
   nx = nx';
   ny = ny';
   nz = nz';
   xvec = xvec';
   yvec = yvec';
   zvec = zvec';

   % Get the indices of each Path Point in the large map
   % Used to pick values for the Path Points off the large
   % matrices

   ppIndices = [Data.yIndices(1)+Data.Pathpoints{enR}(:,2), Data.xIndices(1)+Data.Pathpoints{enR}(:,1)];

   for n=1:size(Data.Pathpoints{enR},1)
      % Get the nx, ny, and nz values for each Path Point
      normals(n,1) = nx(ppIndices(n,1), ppIndices(n,2));
      normals(n,2) = ny(ppIndices(n,1), ppIndices(n,2));
      normals(n,3) = nz(ppIndices(n,1), ppIndices(n,2));

      % Get the xvec, yvec, and zvec values for each Path Point
      cartesians(n,1) = xvec(ppIndices(n,1), ppIndices(n,2));
      cartesians(n,2) = yvec(ppIndices(n,1), ppIndices(n,2));
      cartesians(n,3) = zvec(ppIndices(n,1), ppIndices(n,2));
   end
   clear nx ny nz xvec yvec zvec

   % x position in transposed z; y position in transposed z; elevation;
   % time at arrival

   % Note: there is no -1 with the Index, because the location on the
   % map is measured starting from 0 (and not 1, like the indicies
   % of a matrix)
   ppSun(:,1:4) = [Data.xLimits(1)+(Data.Pathpoints{enR}(:,1))*.24...
      Data.yLimits(1)-(Data.Pathpoints{enR}(:,2))*.24...
      Data.Pathpoints{enR}(:,3)...
      Data.Time{enR}];                            % seconds

   % Find the unique sun times so we don't have to do a lot of
   % calls to MICE
   % The non-unique times
   sunTimes=round(ppSun(:,4)/shadowScaling);      % seconds
   [bee,mmmL,nnnL] = unique(sunTimes);
   [b2,mmmF,nnnF] = unique(sunTimes,'first');
   % bee and b2 are the multiples of the sunTimes (they should be
   % a list of integers from 0 to a number, and both be the same)
   % mmL and mmF are explained below
   % nnnL and nnnF are the bins of each Path Point.  They're not
   % used

   mmm = [mmmF, mmmL];
   % Each row of mmm is a different 'bin' for the sun times
   % The first column, mmmF, is the first Path Point in that bin
```

```
% The second column, mmmL, is the last Path Point in that bin

% The unique times
[sunTimesToGet] = bee*shadowScaling;

% Create vector of (unique) ephemeris times
et = zeros(1,length(sunTimesToGet));
et(1) = cspice_str2et(datestr(Data.CurrentTime));
for n=2:length(sunTimesToGet)
   et(n) = et(1)+sunTimesToGet(n);
end

% Unique times
pos_=cspice_spkpos(itarget,et,refframe,'LT+S',iobserver);
% Put the big matrix back together
% For each Path Point
%for n=1:length(sunTimes)
%    pos_(:,n) = posUnique(:,nnnL(n));
%end
xsun_=pos_(1,:);
ysun_=pos_(2,:);
zsun_=pos_(3,:);

[lonsun_,latsun_,rsun_]=cart2sph(xsun_,ysun_,zsun_);
lonsun_=mod(lonsun_*180/pi,360);
latsun_=latsun_*180/pi;

[xpsun_,ypsun_]=project_gnomonic(lonsun_,latsun_,lonproj,latproj,Rmoon);
rpsun_=sqrt(xpsun_.^2+ypsun_.^2);

%anglesunget_=mod(atan2(ysun_,xsun_)*180/pi+90,360); % which angle do we need to get the ELEV map?
anglesunget_=mod(-lonsun_+90,360);

% For all path points
%x = ppSun(:,1);
%y = ppSun(:,2);
%z = ppSun(:,3);
%nx = normals(:,1);
%ny = normals(:,2);
%nz = normals(:,3);
%       xvec = cartesians(:,1);
%       yvec = cartesians(:,2);
%       zvec = cartesians(:,3);

% Observer position for all Path Points
obspos=[0 0 -3000]; % ~1300 km above the South Pole
obsr=sqrt((obspos(1)-ppSun(:,1)).^2+(obspos(2)-ppSun(:,2)).^2+(obspos(3)-ppSun(:,3)).^2);
emission_=acosd((normals(:,1).*(obspos(1)-ppSun(:,1))+normals(:,2).*(obspos(2)-ppSun(:,2))+normals(:,3).*(obspos(3)-ppSun(:,3)))./obsr);

visibFULL = 0;

% For each unique time
for itt=1:size(pos_,2)
   angradsun=atand(Rsun/rsun_(itt)); % current angular radius
   anglesunget=anglesunget_(itt); % tells us which maps to fetch

   % Cartesian coordinates for the Path Points associated with this
   % sun position only
   x = ppSun(mmm(itt,1):mmm(itt,2),1);
   y = ppSun(mmm(itt,1):mmm(itt,2),2);
   z = ppSun(mmm(itt,1):mmm(itt,2),3);
   xsun_(itt);
   ysun_(itt);
   zsun_(itt);
   xvec = cartesians(mmm(itt,1):mmm(itt,2),1);
   yvec = cartesians(mmm(itt,1):mmm(itt,2),2);
   zvec = cartesians(mmm(itt,1):mmm(itt,2),3);
   matrixIndices = ppIndices(mmm(itt,1):mmm(itt,2),:);

   elsun=xvec.*(-xvec+xsun_(itt))+yvec.*(-yvec+ysun_(itt))+zvec.*(-zvec+zsun_(itt));   % dot product
   elsun=elsun./(Rmoon+ppSun(mmm(itt,1):mmm(itt,2),3)/1000);
   elsun=elsun./sqrt((-xvec+xsun_(itt)).^2+(-yvec+ysun_(itt)).^2+(-zvec+zsun_(itt)).^2);
   % For each Path Point associated with this sun position
   elsun=(90-acosd(elsun));

   % get_horizon_elevation (pasted in from function)
   %clear isunall isunloaded isunorder

   % decide which files need to be opened
   if (anglesunget<theta_vec2(1))
```

```
    isun=0;
  else
    isun=find(and(anglesunget>=theta_vec2,anglesunget<theta_vec2+dth));
  end

  isunall=isun+[0 1];
  isunall(isunall<1)=isunall(isunall<1)+nth;
  isunall(isunall>nth)=isunall(isunall>nth)-nth+1;
  isun(isun<1)=isun(isun<1)+nth;

  isunloaded=isunall;
  isunorder=1:length(isunloaded);
  elev_=NaN*ones(size(x,1),size(x,2),length(isunall));
  for i=1:length(isunall)
    filetmp=sprintf('%s/elev%03d',Data.DIRbins,isunall(i));
    if (exist(filetmp,'file'))
%       fprintf('reading elev%03d \n',isunall(i));
      fid=fopen(filetmp,'rb');
      elevtmp=fread(fid,inf,'single','ieee-le');
      n=sqrt(numel(elevtmp));
      elevtmp=reshape(elevtmp,n,n);
      %Next line is resizing by Aaron
%       elevtmp=elevtmp(600:1600,600:1600);
      for n=1:length(x)
        elev_(n,1,isunorder(i))=elevtmp(matrixIndices(n,1),matrixIndices(n,2));
        %elev_(:,:,isunorder(i))=elevtmp';
      end
      clear elevtmp
      fclose(fid);
    else
      elev_(:,:,isunorder(i))=NaN;
    end
  end

  % 0 and 1 are the two angles
  x0=theta_vec2(isunall(1));
  y0=elev_(:,:,isunorder(1));
  x1=theta_vec2(isunall(2));
  if (x1<x0)&&(anglesunget<=x0)
    x0=x0-360;
  elseif (x1<x0)&&(anglesunget>x0)
    x1=x1+360;
  end
  y1=elev_(:,:,isunorder(2));

  xc=anglesunget;
  yc=elsun;
  rc=angradsun;

  % interpolated elevation
  yi=y0+(y1-y0).*(xc-x0)/(x1-x0);

  %
  % calculating how much of the sum is seen
  % takes into account a tilted slope
  visib=max(min((yc-yi+rc)/(2*rc),1),0);

  k=find(and(visib~=0,visib~=1));

  a=(y1(k)-y0(k))/(x1-x0);
  b=y0(k)-yc(k)-x0*a;

  A=1+a.^2;
  B=2*(a.*b-xc);
  C=xc^2+b.^2-rc^2;

  xi1=(-B-sqrt(B.^2-4*A.*C))./(2*A);
  xi2=(-B+sqrt(B.^2-4*A.*C))./(2*A);

  yi1=y0(k)+(y1(k)-y0(k)).*(xi1-x0)/(x1-x0);
  yi2=y0(k)+(y1(k)-y0(k)).*(xi2-x0)/(x1-x0);

  v1x=xi1-xc;
  v1y=yi1-yc(k);
  v1z=zeros(size(xi1));
  v1r=sqrt(v1x.^2+v1y.^2+v1z.^2);

  v2x=xi2-xc;
  v2y=yi2-yc(k);
  v2z=zeros(size(xi2));
```

```
    v2r=sqrt(v2x.^2+v2y.^2+v2z.^2);

    area1=abs(v1x.*v2y-v1y.*v2x)/2;
    alpha=acos((v1x.*v2x+v1y.*v2y+v1z.*v2z)./v1r./v2r);

    alpha(yc(k)>yi(k))=2*pi-alpha(yc(k)>yi(k));
    area2=alpha/2*rc^2;

    area=area2;
    area(alpha>=pi)=area(alpha>=pi)+area1(alpha>=pi);

    k2=B.^2-4*A.*C>=0;
    visib(k(k2))=area(k2)/(pi*rc^2);
    visibFULL = vertcat(visibFULL,visib);

    %
    clear v1* v2* A B C area* xi* yi* xc yc rc x0 y0 x1 y1 yi
    %get_visibility_ratio;   % calculates interpolated maps; calculates sun disk visibility
end

Data.Shadows{enR} = visibFULL(2:end);

% Data.MetCost in BTU
% Data.Time in seconds

emissivityMoon = 0.95;
absorptivitySuit = 0.18;    % From Balinskas
emissivitySuit = 0.837;     % From Balinskas
solarConstant = 1367;   % W/m^2
sigma = 5.6704e-8;      % W/m^2*K^4
VFSuitMoon = 0.5;
VFSuitSpace = 0.5;
A_Du = 2.469;           % m^2
TSuit_in = 300;
TSpace = 3;         % K
mSuit = 50;
cp_Suit = 1000;         % J/kg-K

suitBatteryHeat = 50;   % W
A_rad = 0.09290304;     % m^2 (1 ft^2)
%emissivityRad = 0.92;   % Z-93 white paint
emissivityRad = 0;

m_dot = 240*.45359237/3600; % kg/sec
cp_water = 4186;        % J/(kg-K)
h_subWater = 2594000;   % J/kg
C = 1.19;               % Suit conductivity - W/m^2-K

T_in = 288;

efficiencyPlanets = [1 0.78 0.48];
Eplanet = efficiencyPlanets(Data.Planet);
Qbasal = 0.9772*Data.Weight(en) + 8.1765;  % W

% Rover Constants
% efficiency_SA = 0.17;     % Si
% efficiency_SA = 0.235;    % InGap/GaAs-on-Ge dual-junction
efficiency_SA = 0.26;       % InGap/GaAs-on-Ge triple-junction
efficiency_SA = efficiency_SA*0.77; % To account for losses
A_SA = 30;   % m^2
batterySpecificEnergy = 145;    % W-hr/kg; Seung-Bum
mBattery = 269;             % kg; Seung-Bum

start = '';

 % If you're doing a return home path, check and see if your
 % starting point is a Path Point along the parent traverse
if ~isempty(isReturn)
    start = find((Data.Pathpoints{en}(:,1)==Data.Pathpoints{enR}(1,1)) & ...
                 (Data.Pathpoints{en}(:,2)==Data.Pathpoints{enR}(1,2)),1);
end
% If you have a return home path, if you have information for
% your starting point along the parent traverse, and if you
% have shadowing data for the parent traverse
if ~isempty(start) && ~isempty(Data.Shadows{en})
    if strcmp(Data.Explorers{enR},'Rover')
        Data.BatteryCapacity{enR}(1) = Data.BatteryCapacity{en}(start);
    else
        Data.SublimatedWater{enR}(1) = Data.SublimatedWater{en}(start);
        Data.HeaterPower{enR}(1) = Data.HeaterPower{en}(start);
```

```
        Data.TSuit{enR}(1) = Data.TSuit{en}(start);
    end
% If you DON'T have a return home path, if you DON'T have
% information for your starting point along the parent
% traverse, or if you DON'T have shadowing data for the parent
% traverse
else
    if strcmp(Data.Explorers{enR},'Rover')
        Data.BatteryCapacity{enR}(1) = batterySpecificEnergy*mBattery;     % W-hr
    else
        Data.SublimatedWater{enR}(1) = 0;
        Data.HeaterPower{enR}(1) = 0;
        Data.TSuit{enR}(1) = 300;
    end
end

if strcmp(Data.Explorers{enR},'Rover')
    for n=2:size(Data.Pathpoints{enR},1)
        stageMetCost = (Data.MetCost{enR}(n)-Data.MetCost{enR}(n-1));   % Joules
        stageTime = Data.Time{enR}(n) - Data.Time{enR}(n-1);            % seconds
        stageShadowing = (Data.Shadows{enR}(n) + Data.Shadows{enR}(n-1))/2;

        P_SA = stageShadowing*solarConstant*efficiency_SA*A_SA;    % W

        if P_SA >= (stageMetCost/stageTime)    % We can run everything off the solar panels
            powerBattCharge = P_SA - stageMetCost/stageTime;  % W
            if Data.BatteryCapacity{enR}(n-1) < Data.BatteryCapacity{enR}(1)   % Battery is less than full capacity
                Data.BatteryCapacity{enR}(n) = Data.BatteryCapacity{enR}(n-1) + powerBattCharge*stageTime/3600; % W-hr
                if Data.BatteryCapacity{enR}(n) > Data.BatteryCapacity{enR}(1)  % If the batteries have been over-charged
                    Data.BatteryCapacity{enR}(n) = Data.BatteryCapacity{enR}(1);
                end
            else    % Battery is already at full capacity
                Data.BatteryCapacity{enR}(n) = Data.BatteryCapacity{enR}(n-1);
            end
        else
            powerBattDischarge = stageMetCost/stageTime - P_SA;    % W
            Data.BatteryCapacity{enR}(n) = Data.BatteryCapacity{enR}(n-1) - powerBattDischarge*stageTime/3600;  % W-hr
        end
    end

else
    for n=2:size(Data.Pathpoints{enR},1)

        stageMetCost = (Data.MetCost{enR}(n)-Data.MetCost{enR}(n-1));   % BTUs
        stageDistance = (Data.Distance{enR}(n)-Data.Distance{enR}(n-1));   % m
        stageSlope = 180/pi*atan((Data.Pathpoints{enR}(n,3)-Data.Pathpoints{enR}(n-1,3))/stageDistance);
        if Data.Pathpoints{enR}(n,3) > Data.Pathpoints{enR}(n-1,3)
            stageSlope = abs(stageSlope);
        else
            stageSlope = -1*abs(stageSlope);
        end
        stageShadowing = (Data.Shadows{enR}(n) + Data.Shadows{enR}(n-1))/2;

        if stageSlope < -12
            efficiency = -1.2;
        elseif (stageSlope >= -12 && stageSlope < -10)
            efficiency = 0.15*stageSlope+0.6;
        elseif (stageSlope >= -10 && stageSlope < -5)
            efficiency = 0.1*stageSlope+0.1;
        elseif (stageSlope >= -5 && stageSlope < 0)
            efficiency = 0.08*stageSlope;
        elseif (stageSlope >= 0 && stageSlope < 5)
            efficiency = 0.025*stageSlope;
        elseif (stageSlope >= 5 && stageSlope < 10)
            efficiency = 0.015*stageSlope+0.05;
        elseif (stageSlope >= 10 && stageSlope < 20)
            efficiency = 0.005*stageSlope+0.15;
        elseif (stageSlope >= 20)
            efficiency = 0.25;
        else    % If we're standing still
            efficiency = 0;
        end

        stageTime = Data.Time{enR}(n) - Data.Time{enR}(n-1);

        Qn = (stageMetCost*(1-efficiency*Eplanet-0.2*Eplanet))*1055.05585/stageTime + 0.2*Eplanet*Qbasal;  % W (J/s)
        % stageMetCost is in BTUs (1055.05585 J/BTU)
        % Qbasal is already in W

        % Moon's temperature is 100 deg-C in sunlight and -150
```

```
            % deg-C in shadows
            TMoon = (stageShadowing*250-150)+273;     % K

            directSunRadiation = stageShadowing*absorptivitySuit*(A_Du/2)*solarConstant;  % W
            % Assume suit is kept at a constant temperature
            heatSuitMoon = sigma*emissivityMoon*emissivitySuit*A_Du*VFSuitMoon*(TMoon^4-Data.TSuit{enR}(n-1)^4); % W
            heatSuitSpace = sigma*emissivitySuit*A_Du*VFSuitSpace*(TSpace^4-Data.TSuit{enR}(n-1)^4); % W

            % Compute suit outside temperature at the end of the stage
            % Based on the stage external heat transfer
            Data.TSuit{enR}(n) = (directSunRadiation+heatSuitMoon+heatSuitSpace)*stageTime/(mSuit*cp_Suit) + Data.TSuit{enR}(n-1);  % K

            % TSuit_out is the temperature of the suit at the
            % beginning of the stage
            heatSuitConduction = C*A_Du*(Data.TSuit{enR}(n-1)-TSuit_in);   %

            stageSuitHeat = suitBatteryHeat + Qn + heatSuitConduction;   % W
            Data.StageSuitHeat{enR}(n-1,1:4) = [suitBatteryHeat Qn heatSuitConduction stageSuitHeat];

            % LCG outlet temperature
            T_out = stageSuitHeat/(m_dot*cp_water)+T_in;    % K
            if T_out < 273
               disp('Frozen!')
            end

            % Maximum amount of heat that can be rejected by radiator
            % Assumes that the radiator is the same temperature as the
            % water
            Q_radMax = sigma*A_rad*emissivityRad*(T_out^4 - TSpace^4);   % W

            if stageSuitHeat < Q_radMax
               % Reject all heat through radiator
               massSublimatedWater = 0;

               if stageSuitHeat > 0
                  heaterPower = 0;     % kJ
               else
                  heaterPower = -stageSuitHeat*stageTime/1000;     % kJ
               end
            else
               % Temperature after radiator
               T_rad = T_out - Q_radMax/(m_dot*cp_water);  % K

               sublimationHeat = m_dot*cp_water*(T_rad - T_in);    % W
               massSublimatedWater = sublimationHeat/h_subWater*stageTime;   % kg

               heaterPower = 0;
            end

            Data.SublimatedWater{enR}(n) = massSublimatedWater+Data.SublimatedWater{enR}(n-1);
            Data.HeaterPower{enR}(n) = heaterPower+Data.HeaterPower{enR}(n-1);

         end
      end

      set(MPfig,'UserData',Data)

      sextant('Mission','Plots','');

   elseif isempty(Data.moon)
      message = 'Shadows can only be determined for lunar traverses.';
      Data.minfo = helpdlg(sprintf(message),'SEXTANT');
      set(MPfig,'UserData',Data)
   else
      message = 'There is no traverse path to find.';
      Data.minfo = helpdlg(sprintf(message),'SEXTANT');
      set(MPfig,'UserData',Data)
   end



% Writes xml file that can be read by the Compendium database
% Allows SEXTANT plan to be imported into imAS
% Aaron Johnson - 8/26/09
case 'Points'
   isReturn = Data;
   MPfig = gcf;
   Data = get(MPfig,'UserData');
   en = get(Data.MPexpnumH,'Value');
```

Appendices

```
enR = en + ~isempty(isReturn)*Data.NumExp;

if ~isempty(Data.Pathpoints{enR})

   % Covert from UTM to Latitude/Longitude
   if Data.UTMzone ~= 0
      xPathPoints = ((Data.Pathpoints{enR}(:,1)+.5).*Data.Resolution) + Data.xllcorner - 500000;
      yPathPoints = ((Data.Rows - 1 - (Data.Pathpoints{enR}(:,2)+.5)).*Data.Resolution) + Data.yllcorner - 10000000*(Data.UTMzone<0);

      for n = 1:length(xPathPoints)
         x = xPathPoints(n);
         y = yPathPoints(n);
         e  = (1-6356752.314^2/6378137^2)^(1/2);
         mu = y/(.9996*6378137*(1-e^2/4-3/64*e^4-5/256*e^6));
         e1 = (1-(1-e^2)^(1/2))/(1+(1-e^2)^(1/2));
         J  = [3/2*e1-27/32*e1^3, 21/16*e1^2-55/32*e1^4,...
               151/96*e1^3,       1097/512*e1^4];
         fp = mu+J(1)*sin(2*mu)+J(2)*sin(4*mu)+J(3)*sin(6*mu)+J(4)*sin(8*mu);
         e2 = e^2/(1-e^2);
         C  = e2*cos(fp)^2;
         T  = tan(fp)^2;
         R  = 6378137*(1-e^2)/(1-e^2*sin(fp)^2)^(3/2);
         N  = 6378137/(1-e^2*sin(fp)^2)^(1/2);
         D  = x/(.9996*N);
         Qa = [N*tan(fp)/R, D^2/2, (5+3*T+10*C-4*C^2-9*e2)*D^4/24,...
               (61+90*T+298*C+45*T^2-3*C^2-252*e2)*D^6/720];
         Qo = [D, (1+2*T+C)*D^3/6, (5-2*C+28*T-3*C^2+8*e2+24*T^2)*D^5/120];
         lat  = (fp-Qa(1)*(Qa(2)-Qa(3)+Qa(4)))*180/pi;
         long = abs(Data.UTMzone)*6-183+((Qo(1)-Qo(2)+Qo(3))/cos(fp))*180/pi;

         % Create matrix of coordinates for each Path Point
         Data.LatLong{enR}(n,:) = [lat long];
      end
   end

   % Round Data.Time{en}
   % Otherwise, timing won't work correct in iMAS
   timeRounded = round(Data.Time{en});

   % fileName is .txt so you can copy and paste it into seed .xml
   % file
   % This can be changed
   fileName = ['COMPENDIUM_',Data.EVAname,'_Exp',num2str(en),isReturn,'.txt'];
   %fileName = 'forSEXTANT.xml';
   fid = fopen(fileName, 'w');
   view = '<?xml version="1.0" encoding="UTF-16"?>\n<!DOCTYPE model SYSTEM "Compendium.dtd">\n<model rootview="1">\n    <views>\n';
   link = '    <links>\n';

   dateString=strcat(num2str(Data.Hour),num2str(Data.Minute),num2str(Data.Month),num2str(Data.Day),num2str(Data.Year));
   % Create begining of node string
   % Home window node
   node = strcat('         <node id="1" type="2" extendedtype="" originalid="" author="SEXTANT" created="',dateString,...
      '" lastModified="',dateString,'" label="Home Window" state="2" lastModificationAuthor="SEXTANT">\n             <details>\n                 <page
nodeid="1" author="SEXTANT" created="',dateString,...
      '" lastModified="',dateString,'" pageno="1">Home Window</page>\n             </details>\n             <source></source>\n             <image width="0"
height="0"></image>\n             <background></background>\n             <coderefs></coderefs>\n             <shortcutrefs></shortcutrefs>\n         </node>\n');

   % Initial codeID
   codeInfo = 4000000001;
   % All required code labels            % CodeID:
   codeLabels ={'EVA';                   % 4000000001
            'EvaAstronaut';          % 4000000002
            'Locations';             % 4000000003
            'MapLocation';           % 4000000004
            'GPSData';               % 4000000005
            'location';              % 4000000006
            'Plan';                  % 4000000007
            'ScheduledActivities';   % 4000000008
            'MovementActivity';      % 4000000009
            'ActivityAttribute';     % 4000000010
            'StationaryActivity'};   % 4000000011

   codeString = '    <codes>\n';
   % Create the entire code string that describes all the codes
   % required by iMAS
   for b = 1:length(codeLabels)
      codeString = strcat(codeString,'         <code id="',num2str(codeInfo),'" author="SEXTANT" created="',dateString,...
         '" lastModified="',dateString,'" name="',codeLabels{b},'" description="No Description" behavior="No Behavior" />\n');
      codeInfo = codeInfo+1;
   end
```

```
% Format is: [NodeID ParentID Type LinkID ToID]
% Create nodes that are required for all .xml files
% (Separate from anything dealing with individual Path Points)
%                                   NodeID:                 ParentID:       ToID:
compendiumInfo = [2 1 2 0 0 4000000001;          % EVA#                  Home
            3 2 3 0 0 0;                % hasPlan                EVA#
            4 2 2 1000000001 3 4000000007;   % EVA#Plan               EVA#          hasPlan
            5 2 3 0 0 0;                % Locations              EVA#
            6 2 2 1000000002 5 4000000003;   % EVA#Locations          EVA#          Locations
            7 6 3 0 0 0;                % Locations              EVA#Locations
            8 2 3 0 0 0;                % hasPerformer           EVA#
            9 2 4 1000000003 8 4000000002;   % HabCom                 EVA#          hasPerformer
            10 2 4 1000000004 8 4000000002;  % AstroOne               EVA#          hasPerformer
            11 2 4 1000000005 8 4000000002;  % AstroTwo               EVA#          hasPerformer
            12 2 3 0 0 0;                % Collected Science Data   EVA#
            13 4 3 0 0 0;                % ScheduledActivities      EVA#Plan
            14 4 2 1000000006 13 4000000008;  % AstroActivities         EVA#Plan       ScheduledActivities
            15 4 3 0 0 0;                % Duration               EVA#Plan
            16 4 4 1000000007 15 0;          % Total EVA Time         EVA#Plan       Duration
            17 4 3 0 0 0;                % hasPerformer           EVA#Plan
            10 4 0 1000000008 17 0;          % AstroOne               EVA#Plan       hasPerformer          (No node created)
            11 4 0 1000000009 17 0;          % AstroOne               EVA#Plan       hasPerformer          (No node created)
            18 4 3 0 0 0;                % Collected Science Data   EVA#Plan
            ...
            ...                     % MOVEMENT ACTIVITY:
            19 0 3 0 0 4000000010;           % name                   MoveToPathPoint(#+1)                (No views or links created)
            20 0 3 0 0 4000000010;           % activityType           MoveToPathPoint(#+1)                (No views or links created)
            21 0 4 0 0 0;                % movementActivity       MoveToPathPoint(#+1)                (No views or links created)
            22 0 3 0 0 4000000010;           % startLocation          MoveToPathPoint(#+1)                (No views or links created)
            23 0 3 0 0 4000000010;           % endLocation            MoveToPathPoint(#+1)                (No views or links created)
            24 0 3 0 0 4000000010;           % duration               MoveToPathPoint(#+1)                (No views or links created)
            25 0 3 0 0 4000000010;           % durationThreshold      MoveToPathPoint(#+1)                (No views or links created)
            26 0 4 0 0 0;                % 1                      MoveToPathPoint(#+1)                (No views or links created)
            27 0 3 0 0 4000000010;           % deviationThreshold     MoveToPathPoint(#+1)                (No views or links created)
            28 0 4 0 0 0;                % 1000000                MoveToPathPoint(#+1)                (No views or links created)
            29 0 3 0 0 4000000010;           % hasPerformer           MoveToPathPoint(#+1)                (No views or links created)
            ...
            ...                     % LOCATION
            30 0 3 0 0 0;                % name                   PathPoint#                          (No views or links created)
            31 0 3 0 0 0;                % isLocatedAt            PathPoint#                          (No views or links created)
            32 0 3 0 0 0;                % stationary             PathPoint#                          (No views or links created)
            33 0 4 0 0 0;                % TRUE                   PathPoint#                          (No views or links created)
            34 0 3 0 0 0;                % radius                 PathPoint#                          (No views or links created)
            35 0 4 0 0 0;                % 10                     PathPoint#                          (No views or links created)
            36 0 3 0 0 0;                % locationType           PathPoint#                          (No views or links created)
            37 0 4 0 0 0;                % SITE                   PathPoint#                          (No views or links created)
            38 0 4 0 0 0;                % HABITAT                PathPoint#                          (No views or links created)
            ...
            ...                     % LOCATION GPS
            39 0 3 0 0 0;                % latitude               PathPoint#_GPS                      (No views or links created)
            40 0 3 0 0 0;                % latitudeDirection      PathPoint#_GPS                      (No views or links created)
            41 0 3 0 0 0;                % longitude              PathPoint#_GPS                      (No views or links created)
            42 0 3 0 0 0;                % longitudeDirection     PathPoint#_GPS                      (No views or links created)
            43 0 3 0 0 0;                % northing               PathPoint#_GPS                      (No views or links created)
            44 0 3 0 0 0;                % easting                PathPoint#_GPS                      (No views or links created)
            45 0 3 0 0 0;                % zoneLetter             PathPoint#_GPS                      (No views or links created)
            46 0 3 0 0 0;                % zoneNumber             PathPoint#_GPS                      (No views or links created)
            47 0 3 0 0 0];               % altitude               PathPoint#_GPS                      (No views or links created)

compendiumLabels = {[Data.EVAname];              % 2
             'hasPlan';                  % 3
             [Data.EVAname,'Plan'];      % 4
             'Locations';                % 5
             [Data.EVAname,'Locations'];  % 6
             'Locations';                % 7
             'hasPerformer';             % 8
             'HabCom';                   % 9
             'AstroOne';                 % 10
             'AstroTwo';                 % 11
             'Collected Science Data';   % 12
             'ScheduledActivities';      % 13
             'AstroActivites';           % 14
             'Duration';                 % 15
             num2str(timeRounded(end));  % 16
             'hasPerformer';             % 17
             'AstroOne';                 %      10
             'AstroTwo';                 %      11
             'Collected Science Data';   % 18
```

```
                'name';                     % 19
                'activityType';             % 20
                'movementActivity';
                'startLocation';
                'endLocation';
                'duration';
                'durationThreshold';        % 25
                '1';
                'deviationThreshold';
                '1000000';
                'hasPerformer';
                'name';                     % 30
                'isLocatedAt';
                'stationary';
                'TRUE';
                'radius';
                '10';                       % 35
                'locationType';
                'SITE';
                'HABITAT';
                'latitude';
                'latitudeDirection';        % 40
                'longitude';
                'longitudeDirection';
                'northing';
                'easting';
                'zoneLetter';               % 45
                'zoneNumber';
                'altitude'};                % 47

for k=1:length(compendiumInfo)
   % No view creation is specified by compendiumInfo(k,2) == 0
   % function [retVS] = viewString(view, nodeID, parentID)
   if compendiumInfo(k,2) ~= 0
      view = viewString(view, compendiumInfo(k,1), compendiumInfo(k,2));
   end

   % No node creation is specified by compendiumInfo(k,3) == 0
   % function [retNS] = nodeString(node, nodeID, type, label,
   % codeID)
   if compendiumInfo(k,3) ~= 0
      node = nodeString(node, compendiumInfo(k,1), compendiumInfo(k,3), compendiumLabels{k}, compendiumInfo(k,6));
   end

   % No link creation is specified by compendiumInfo(k,4) == 0
   % function [retLS] = linkString(link, linkID, fromID, toID,
   % linkviewID)
   if compendiumInfo(k,4) ~= 0
      link = linkString(link, compendiumInfo(k,4), compendiumInfo(k,1), compendiumInfo(k,5), compendiumInfo(k,2));
   end
end

% For indicies sake, we want to create the "Habitat"
% (PathPoint1) as the next index:
currentPP = 48;
% Next available index after that:
index = 49;
% Initial linkIndex
linkIndex = 1000000010;

% Number of Path Points:
tempSize = size(Data.Pathpoints{enR});

% For each PathPoint:
for m = 1:tempSize(1)
   clear PathPointInfo;
   clear pathPointLabels;

   el = 1;    % Index for labels

   % Need to create a link from each Movement Activity to the
   % one before, except for the first Movement Activity
   if m ~= 1
      needLink = [-linkIndex previousMA];
      linkIndex = linkIndex + 1;
   else
      needLink = [0 0];
   end

      % MOVEMENT ACTIVITY:                    NodeID:          ParentID:          ToID:          CodeID:
```

```
        pathPointInfoA = [index 14 2 needLink 4000000009;          % MoveToPathPoint(#+1)   AstroActivities      MoveToPathPoint(#)
MovementActivity
                19 index 0 0 0 0;                  % name              MoveToPathPoint(#+1)                    (No node created)
                index+1 index 4 linkIndex 19 0;          % MoveToPathPoint(#+1)   MoveToPathPoint(#+1)   name
                20 index 0 0 0 0;                  % activityType      MoveToPathPoint(#+1)                    (No node created)
                21 index 0 linkIndex+1 20 0;          % movementActivity      MoveToPathPoint(#+1)   activityType          (No node
created)
                22 index 0 0 0 0];                 % startLocation     MoveToPathPoint(#+1)                    (No node created)

        % This code links the current Path Point to startLocation
        % In pathPointInfoC, the next Path Point is created
        % When we come back here, the current Path Point has
        %     already been created in the previous Movement
        %     Activity
        % So, we only have to create a node here if we are at
        %     the first Path Point (m==1)
        if m==1
          pathPointInfoB = [currentPP index 2 linkIndex+2 22 4000000004];   % PathPoint(#)      MoveToPathPoint(#+1)   startLocation      MapLocation
        elseif m~=1
          pathPointInfoB = [currentPP index 0 linkIndex+2 22 4000000004];   % PathPoint(#)      MoveToPathPoint(#+1)   startLocation      MapLocation
(No node created)
        end

        pathPointInfoC= [23 index 0 0 0 0;                 % endLocation       MoveToPathPoint(#+1)                    (No node created)
                index+2 index 2 linkIndex+3 23 4000000004;       % PathPoint(#+1)      MoveToPathPoint(#+1)   endLocation      MapLocation
                24 index 0 0 0 0;                  % duration          MoveToPathPoint(#+1)                    (No node created)
                index+3 index 4 linkIndex+4 24 0;          % Travel Time       MoveToPathPoint(#+1)   duration
                25 index 0 0 0 0;                  % durationThreshold     MoveToPathPoint(#+1)                    (No node created)
                26 index 0 linkIndex+5 25 0;          % 1                 MoveToPathPoint(#+1)   durationThreshold          (No node created)
                27 index 0 0 0 0;                  % deviationThreshold    MoveToPathPoint(#+1)                    (No node created)
                28 index 0 linkIndex+6 27 0;          % 1000000           MoveToPathPoint(#+1)   deviationThreshold          (No node
created)
                29 index 0 0 0 0;                  % hasPerformer      MoveToPathPoint(#+1)                    (No node created)
                10 index 0 linkIndex+7 29 0;          % AstroOne          MoveToPathPoint(#+1)   hasPerformer          (No node
created)
                11 index 0 linkIndex+8 29 0;          % AstroTwo          MoveToPathPoint(#+1)   hasPerformer          (No node
created)
                18 index 0 0 0 0];                 % Collected Science Data    MoveToPathPoint(#+1)                    (No node created)


        % LOCATION:
        pathPointInfoD = [currentPP 6 0 linkIndex+9 7 0;          % PathPoint(#)      EVA#Locations      Locations          (No node
created)
                30 currentPP 0 0 0 0;                  % name              PathPoint(#)                    (No node created)
                index+4 currentPP 4 linkIndex+10 30 4000000006;       % PathPoint(#) (type 4)    PathPoint(#)      name      location
                31 currentPP 0 0 0 0;                  % isLocatedAt       PathPoint(#)                    (No node created)
                index+5 currentPP 2 linkIndex+11 31 4000000005;       % PathPoint(#)_GPS      PathPoint(#)      isLocatedAt      GPSData
                32 currentPP 0 0 0 0;                  % stationary        PathPoint(#)                    (No node created)
                33 currentPP 0 linkIndex+12 32 0;          % TRUE              PathPoint(#)      stationary          (No node created)
                34 currentPP 0 0 0 0;                  % radius            PathPoint(#)                    (No node created)
                35 currentPP 0 linkIndex+13 34 0;          % 10                PathPoint(#)      radius          (No node created)
                36 currentPP 0 0 0 0];                 % locationType      PathPoint(#)                    (No node created)

        % The first PathPoint must be of locationType "HABITAT"
        % All others are "SITE"
        if m == 1
          pathPointInfoE = [38 currentPP 0 linkIndex+14 36 0];          % HABITAT           PathPoint(#)      locationType          (No node
created)
        elseif m ~= 1
          pathPointInfoE = [37 currentPP 0 linkIndex+14 36 0];          % SITE              PathPoint(#)      locationType          (No node
created)
        end

        pathPointInfoF = [39 index+5 0 0 0 0;                 % latitude          PathPoint(#)_GPS                    (No node created)
                index+6 index+5 4 linkIndex+15 39 0;          % PathPoint(#) Latitude     PathPoint(#)_GPS      latitude          (No node
created)
                40 index+5 0 0 0 0;                % latitudeDirection     PathPoint(#)_GPS                    (No node created)
                index+7 index+5 4 linkIndex+16 40 0;          % NORTH or SOUTH        PathPoint(#)_GPS      latitudeDirection
                41 index+5 0 0 0 0;                % longitude         PathPoint(#)_GPS                    (No node created)
                index+8 index+5 4 linkIndex+17 41 0;          % PathPoint(#) Longitude    PathPoint(#)_GPS      longitude
                42 index+5 0 0 0 0;                % longitudeDirection    PathPoint(#)_GPS                    (No node created)
                index+9 index+5 4 linkIndex+18 42 0;          % EAST or WEST          PathPoint(#)_GPS      longitudeDirection          (No node
created)
                43 index+5 0 0 0 0;                % northing          PathPoint(#)_GPS                    (No node created)
                44 index+5 0 0 0 0;                % easting           PathPoint(#)_GPS                    (No node created)
                45 index+5 0 0 0 0;                % zoneLetter        PathPoint(#)_GPS                    (No node created)
                46 index+5 0 0 0 0;                % zoneNumber        PathPoint(#)_GPS                    (No node created)
                47 index+5 0 0 0 0;                % altitude          PathPoint(#)_GPS                    (No node created)
                index+10 index+5 4 linkIndex+19 47 0];          % PathPoint(#) Altitude     PathPoint(#)_GPS      altitude          (No node
created)
```

Appendices

```
% For all PathPoints but the last, need to create a Movement Activity
% Movement Activity is
% [pathPointInfoA;pathPointInfoB;pathPointInfoC]
if m ~= tempSize(1)
   pathPointInfo = [pathPointInfoA;pathPointInfoB;pathPointInfoC;pathPointInfoD;pathPointInfoE;pathPointInfoF];
elseif m == tempSize(1)
   pathPointInfo = [pathPointInfoD;pathPointInfoE;pathPointInfoF];
else
   disp('Error in calculation of pathPointInfo');
end

% Remember the index of the MoveToPathPoint(#+1)
% Movement Activity
previousMA = index;
% Rember the index of PathPoint#
currentPP = index+2;

%Advance indicies to next available index
index = index+11;
linkIndex = linkIndex+20;

% Convert latitude from +- 90 to North/South
if Data.LatLong{enR}(m,1) > 0
   Data.NSdir{enR}{m} = 'NORTH';
else
   Data.NSdir{enR}{m} = 'SOUTH';
   Data.LatLong{enR}(m,1) = -Data.LatLong{enR}(m,1);
end

% Convert longitude from +-180 to East/West
if Data.LatLong{enR}(m,2) > 0
   Data.EWdir{enR}{m} = 'EAST';
else
   Data.EWdir{enR}{m} = 'WEST';
   Data.LatLong{enR}(m,2) = -Data.LatLong{enR}(m,2);
end

% Call the first PathPoint "Habitat"
if m == 1
   pathPointLabels = {['MoveToPathPoint',num2str(m+1)];                    % index
                      ['MoveToPathPoint',num2str(m+1)];                    % index+1
                      'Habitat';                                           % currentPP
                      ['PathPoint',num2str(m+1)];                          % index+2
                      num2str(timeRounded(m+1)-timeRounded(m));            % index+3
                      'Habitat';                                           % index+4
                      'Habitat_GPS';                                       % index+5
                      strcat(num2str(floor(Data.LatLong{enR}(m,1))),num2str((Data.LatLong{enR}(m,1)-floor(Data.LatLong{enR}(m,1)))*60));      %
Latitude     % index+6
                      Data.NSdir{enR}{m};                                  % index+7
                      strcat(num2str(floor(Data.LatLong{enR}(m,2))),num2str((Data.LatLong{enR}(m,2)-floor(Data.LatLong{enR}(m,2)))*60));      %
Longitude    % index+8
                      Data.EWdir{enR}{m};                                  % index+9
                      num2str(Data.Pathpoints{enR}(m,3))};   % Altitude     % index+10
% For all PathPoints but the last, need to create a
% Movement Activity
elseif m ~= 1 && m ~= tempSize(1)
   pathPointLabels = {['MoveToPathPoint',num2str(m+1)];                    % index
                      ['MoveToPathPoint',num2str(m+1)];                    % index+1
                      ['PathPoint',num2str(m+1)];                          % index+2
                      num2str(timeRounded(m+1)-timeRounded(m));            % index+3
                      ['PathPoint',num2str(m)];                            % index+4
                      ['PathPoint',num2str(m),'_GPS'];                     % index+5
                      strcat(num2str(floor(Data.LatLong{enR}(m,1))),num2str((Data.LatLong{enR}(m,1)-floor(Data.LatLong{enR}(m,1)))*60));      %
Latitude     % index+6
                      Data.NSdir{enR}{m};                                  % index+7
                      strcat(num2str(floor(Data.LatLong{enR}(m,2))),num2str((Data.LatLong{enR}(m,2)-floor(Data.LatLong{enR}(m,2)))*60));      %
Longitude    % index+8
                      Data.EWdir{enR}{m};                                  % index+9
                      num2str(Data.Pathpoints{enR}(m,3))};   % Altitude     % index+10
elseif m == tempSize(1)
   pathPointLabels = {['PathPoint',num2str(m)];                            % index+4
                      ['PathPoint',num2str(m),'_GPS'];                     % index+5
                      strcat(num2str(floor(Data.LatLong{enR}(m,1))),num2str((Data.LatLong{enR}(m,1)-floor(Data.LatLong{enR}(m,1)))*60));      %
Latitude     % index+6
                      Data.NSdir{enR}{m};                                  % index+7
                      strcat(num2str(floor(Data.LatLong{enR}(m,2))),num2str((Data.LatLong{enR}(m,2)-floor(Data.LatLong{enR}(m,2)))*60));      %
Longitude    % index+8
                      Data.EWdir{enR}{m};                                  % index+9
                      num2str(Data.Pathpoints{enR}(m,3))};   % Altitude     % index+10
```

```
        else
            disp('Error with calculation of pathPointLabels');
        end

    for k=1:length(pathPointInfo)
        % No view creation is specified by compendiumInfo(k,2) == 0
        % function [retVS] = viewString(view, nodeID, parentID)
        if pathPointInfo(k,2) ~= 0
            view = viewString(view, pathPointInfo(k,1), pathPointInfo(k,2));
        end
        % No node creation is specified by compendiumInfo(k,3) == 0
        % function [retNS] = nodeString(node, nodeID, type, label,
        % codeID)
        %
        % Note:  This is different than the previous section
        % independent of Path Points.  Here, the labels index
        % ("el") only advances when a node is created.
        if pathPointInfo(k,3) ~= 0
            node = nodeString(node, pathPointInfo(k,1), pathPointInfo(k,3), pathPointLabels{el}, pathPointInfo(k,6));
            el = el+1;
        end
        % No link creation is specified by compendiumInfo(k,4) == 0
        % function [retLS] = linkString(link, linkID, fromID, toID,
        % linkviewID)
        if pathPointInfo(k,4) ~= 0
            link = linkString(link, pathPointInfo(k,4), pathPointInfo(k,1), pathPointInfo(k,5), pathPointInfo(k,2));
        end
    end

end    % End of loop for each PathPoint


% Combine strings and print to file
fprintf(fid,[view,'    </views>\n    <nodes>\n',node,'    </nodes>\n',link,'    </links>\n',codeString,'    </codes>\n</model>']);
fclose(fid);




% GPS.dat file
% Here, also, generate gps.dat file for simulated GPS
%     isReturn = Data;
%     MPfig = gcf;
%     Data = get(MPfig,'UserData');
%       en = get(Data.MPexpnumH,'Value');
%
%     enR = en + ~isempty(isReturn)*Data.NumExp;

fileName = 'gps.dat';
fid = fopen(fileName, 'wt');
gpsDateString=strcat(num2str(Data.Month),num2str(Data.Day),num2str(Data.Year));

if strcmp(Data.EWdir{enR}{1},'EAST')
    EWdir = 'E';
else
    EWdir = 'W';
end

if strcmp(Data.NSdir{enR}{1},'NORTH')
    NSdir = 'N';
else
    NSdir = 'S';
end

latString = strcat(num2str(floor(Data.LatLong{enR}(1,1))),num2str((Data.LatLong{enR}(1,1)-floor(Data.LatLong{enR}(1,1)))*60));
longString = strcat(num2str(floor(Data.LatLong{enR}(1,2))),num2str((Data.LatLong{enR}(1,2)-floor(Data.LatLong{enR}(1,2)))*60));

gpsTraceString = strcat('$GPS,',latString,',',NSdir,',',longString,...
        ',',EWdir,',',gpsDateString,',0000001,099,0,,1*\n');

gpsScaling = input('How many seconds between each GPS point?: '); % How many seconds between each point
% UIcontrols: {Handle,Style,Position,String,Value,...
%         HorizontalAlignment,FontSize,BackgroundColor,Callback}



if ~isnumeric(gpsScaling)
    gpsScaling = 30;
else
    if gpsScaling < 0
```

Appendices

```
            gpsScaling = 30;
        end
    end
    index = 2;

    % Compute list of path point coordinates, in meters
    % (0,0) at lower left corner
%       xPathPoints = Data.Pathpoints{enR}(:,1)*Data.Resolution;                % now in m
%       yPathPoints = (Data.Rows - Data.Pathpoints{enR}(:,2))*Data.Resolution;      % now in m

%       % Scaled travel time
%       travelTimeMin = timeRounded(2) - timeRounded(1);
%       for m = 3:tempSize(1)
%           tempTime = timeRounded(m) - timeRounded(m-1);
%           if tempTime < travelTimeMin
%               travelTimeMin = tempTime;
%           end
%       end
%
%       if gpsScaling > travelTimeMin
%           gpsScaling = travelTimeMin/2;
%       end


    for m=2:tempSize(1) % For each set of Path Points
        %disp('-----------------------------------------------------------------------------------------------')
        %disp(['From ',num2str(m),' to ',num2str(m-1)])
        % This is going to be a repeat for the first one
        %latString = strcat(num2str(floor(Data.LatLong{enR}(m,1))),num2str((Data.LatLong{enR}(m,1)-floor(Data.LatLong{enR}(m,1)))*60));
        %longString = strcat(num2str(floor(Data.LatLong{enR}(m,2))),num2str((Data.LatLong{enR}(m,2)-floor(Data.LatLong{enR}(m,2)))*60));

        %indexString = strcat('000000',num2str(index));

        %gpsTrace = strcat('$GPS,',latString,',',NSdir,',',longString,...
        %    ',',EWdir,',',gpsDateString,',',indexString,',07,0,,1*\n');
        %gpsTraceString = strcat([gpsTraceString, gpsTrace]);
        %index = index+1;

        travelTime = timeRounded(m) - timeRounded(m-1);
        adjustedTravelTime = travelTime/gpsScaling;

        % FOR LENGTH OF LAT/LONG METHOD
%           deltaX = xPathPoints(m) - xPathPoints(m-1);
%           deltaY = yPathPoints(m) - yPathPoints(m-1);
%           xVelocity = deltaX / adjustedTravelTime;
%           yVelocity = deltaY / adjustedTravelTime;
        deltaLat = Data.LatLong{enR}(m,1) - Data.LatLong{enR}(m-1,1);
        deltaLong = Data.LatLong{enR}(m,2) - Data.LatLong{enR}(m-1,2);
        latVelocity = deltaLat / adjustedTravelTime;
        longVelocity = deltaLong / adjustedTravelTime;


        % FOR LENGTH OF LAT/LONG METHOD
%           a = 6378137;    % Equatorial radius of Earth (m)
%           b = 6356752.3;  % Polar radius of Earth (m)
%
%           lengthLat = pi/180*(a*b)^2/((a*cosd(Data.LatLong{enR}(m,1)))^2+(b*sind(Data.LatLong{enR}(m,1)))^2)^(3/2);   % meters
%           lengthLong = pi/180*cosd(Data.LatLong{enR}(m,1))*sqrt((a^4*cosd(Data.LatLong{enR}(m,1))^2 +
b^4*sind(Data.LatLong{enR}(m,1))^2)/((a*cosd(Data.LatLong{enR}(m,1)))^2+(b*sind(Data.LatLong{enR}(m,1)))^2));   % meters

        % Convert latitude from +- 90 to North/South
        if strcmp(Data.EWdir{enR}{m},'EAST')
            EWdir = 'E';
        else
            EWdir = 'W';
        end
        % Convert longitude from +-180 to East/West
        if strcmp(Data.NSdir{enR}{m},'NORTH')
            NSdir = 'N';
        else
            NSdir = 'S';
        end

        % (n*yVelocity + xPathPoints(m))/lengthLat is latitude of
        % specific point
        % (n*xVelocity)... is longitude of specific point

        % NOTE:  Latitude is the y coordinate and Longitude is the
        % x coordinate
        for n=1:ceil(adjustedTravelTime-1)
```

```
            % FOR LENGTH OF LAT/LONG METHOD
%               newLat = (n*yVelocity)/lengthLat+Data.LatLong{enR}(m-1,1);
%               newLong = (n*xVelocity)/lengthLong+Data.LatLong{enR}(m-1,2);

            newLat = (n*latVelocity)+Data.LatLong{enR}(m-1,1);
            newLong = (n*longVelocity)+Data.LatLong{enR}(m-1,2);

            % Check if we've crossed into a different hemisphere
            if newLat < 0
               if strcmp(NSdir,'N')
                  NSdir = 'S';
               else
                  NSdir = 'N';
               end
               newLat = -newLat;
            end

            if newLong < 0
               if strcmp(EWdir,'E')
                  EWdir = 'W';
               else
                  EWdir = 'E';
               end
               newLong = -newLong;
            elseif newLong > 180
               if strcmp(EWdir,'E')
                  EWdir = 'W';
               else
                  EWdir = 'E';
               end
               newLong = 360-newLong;
            end

            latString = strcat(num2str(floor(newLat)),num2str((newLat-floor(newLat))*60));
            longString = strcat(num2str(floor(newLong)),num2str((newLong-floor(newLong))*60));

            indexString = strcat('000000',num2str(index));

            gpsTrace = strcat('$GPS,',latString,',',NSdir,',',longString,...
               ',',EWdir,',',gpsDateString,',',indexString,',07,0,,1*\n');
            gpsTraceString = strcat([gpsTraceString, gpsTrace]);
            index = index+1;
         end

         %index

         % Add a Path Point
         latString = strcat(num2str(floor(Data.LatLong{enR}(m,1))),num2str((Data.LatLong{enR}(m,1)-floor(Data.LatLong{enR}(m,1)))*60));
         longString = strcat(num2str(floor(Data.LatLong{enR}(m,2))),num2str((Data.LatLong{enR}(m,2)-floor(Data.LatLong{enR}(m,2)))*60));

         indexString = strcat('000000',num2str(index));

         gpsTrace = strcat('$GPS,',latString,',',NSdir,',',longString,...
            ',',EWdir,',',gpsDateString,',',indexString,',099,0,,1*\n');
         gpsTraceString = strcat([gpsTraceString, gpsTrace]);
         index = index+1;
      end

      fprintf(fid,gpsTraceString);
      fclose(fid);

      % Display message so the user knows that the .xml file has been
      % created
      message = 'Finished with creating .xml file for Compendium';
      helpdlg(sprintf(message),'SEXTANT');
   end
% End of case 'Points'

case 'Specify'
   MPfig = gcf;
   Data = get(MPfig,'UserData');

   en = get(Data.MPexpnumH,'Value');

   % If you're on the Moon, pressing this button will let you specify
   % a length of time spent at an Activity Point
   if strcmp(Data.moon,'moon')
      timeScaling = 1;

      if ~isempty(Data.Pathpoints{en})
```

```
    if strcmp(Data.Explorers{en},'Rover')
       Qbasal = Data.electronicsPower;
    else
       Qbasal = 280*0.00094781712; %BTU/sec;
    end

    for n=1:size(Data.Waypoints{en},1)
       stationaryTime = input(['How long spent at Activity Point ',num2str(n),'? (min): ']);
       Data.stationaryTime{en}(n) = Data.stationaryTime{en}(n)+stationaryTime;

       % Get the coordinates of each waypoint
       clx = Data.Waypoints{en}(n,1);
       cly = Data.Waypoints{en}(n,2);
       % Find the pathpoint number that correspond to the waypoint
       pp = find((Data.Pathpoints{en}(:,1)==clx) & ... %pp=Point along
              (Data.Pathpoints{en}(:,2)==cly),1);   %  traverse path

       numEntries = ceil(stationaryTime/timeScaling);

       tempPP = zeros(size(Data.Pathpoints{en},1)+numEntries,3);
       tempDist = zeros(size(Data.Pathpoints{en},1)+numEntries,1);
       tempMetCost = zeros(size(Data.Pathpoints{en},1)+numEntries,1);
       tempTime = zeros(size(Data.Pathpoints{en},1)+numEntries,1);

       % Copy the entries in PP before the new entries
       tempPP(1:pp,:) = Data.Pathpoints{en}(1:pp,:);
       tempDist(1:pp,1) = Data.Distance{en}(1:pp,1);
       tempMetCost(1:pp,1) = Data.MetCost{en}(1:pp,1);
       tempTime(1:pp,1) = Data.Time{en}(1:pp,1);

       if numEntries > 1
          % Put in the new entries
          for m=1:1:numEntries-1
             index = pp+m;
             tempPP(index,:) = tempPP(pp,:);
             tempDist(index,1) = tempDist(pp,1);
             tempTime(index,1) = tempTime(index-1,1) + timeScaling*60;
             tempMetCost(index,:) = tempMetCost(index-1,1) + Qbasal*timeScaling*60;
          end
       end
       % Put in the last new entries
       if mod(stationaryTime,timeScaling) == 0
          lastOne = timeScaling;
       else
          lastOne = mod(stationaryTime,timeScaling);
       end
       tempPP(pp+numEntries,:) = tempPP(pp,:);
       tempDist(pp+numEntries,1) = tempDist(pp,1);
       tempTime(pp+numEntries,1) = tempTime(pp+numEntries-1,1) + lastOne*60;
       tempMetCost(pp+numEntries,1) = tempMetCost(pp+numEntries-1,1) + Qbasal*lastOne*60;

       % Put back the rest of the old entries
       tempPP(pp+numEntries+1:end,:) = Data.Pathpoints{en}(pp+1:end,:);
       tempDist(pp+numEntries+1:end,1) = Data.Distance{en}(pp+1:end,1);
       tempTime(pp+numEntries+1:end,1) = Data.Time{en}(pp+1:end,1)+(tempTime(pp+numEntries,1)-tempTime(pp,1));
       tempMetCost(pp+numEntries+1:end,1) = Data.MetCost{en}(pp+1:end,1)+(tempMetCost(pp+numEntries,1)-tempMetCost(pp,1));

       Data.Pathpoints{en} = tempPP;
       Data.Distance{en} = tempDist;
       Data.Time{en} = tempTime;
       Data.MetCost{en} = tempMetCost;
    end

%      en = enR + ~isempty(Data.R)*Data.NumExp;
%      set(Data.MPpathenH,'String',sprintf('%d%s',enR,Data.R),...
%                  'ForegroundColor',ecolors{mod(enR-1,10)+1})
    distU = get(Data.MPscaleH,'Value');
    distR = {1,'%.0f'; .001,'%.2f'; 3.28084,'%.0f'; .0006213712,'%.2f'};
    set(Data.MPdistNH,'String',sprintf(distR{distU,2},...
                  Data.Distance{en}(end)*distR{distU,1}))
    costR = [.2521644 1 1.055056];
    set(Data.MPcostNH,'String',sprintf('%.1f',...
       Data.MetCost{en}(end)*costR(get(Data.MPcostUH,'Value'))))
    hourmin = [floor(Data.Time{en}(end)/3600),...
          round(rem(Data.Time{en}(end),3600)/60)];
    set(Data.MPtimeH,'String',sprintf('%d:%d%d',...
                  hourmin(1),zeros(hourmin(2)<10),hourmin(2)))

  else
    disp('You must plan a path first')
```

```
            return
        end
    % If you're on Earth, pressing this button will let you specify the
    % location of an Activity Point in UTM coordinates
    else    % Specify AP
        if ~isempty(Data.Pathpoints{en})   % Check if path exists
                MPfigH = uisuspend(MPfig);
                Choice = questdlg('Editing waypoints will clear the traverse path.',...
                        sprintf('Edit Explorer %d',en),'OK','Cancel','OK');
                uirestore(MPfigH)
                if strcmp(Choice,'Cancel'), return, end
                delete(Data.PathHandles{[en,en+Data.NumExp]})
                set(Data.WayHandles{en}(1),'SizeData',120,...
                    'CData',Data.Ecolors{mod(en-1,10)+1})
                Data.Waypoints{en+Data.NumExp} = [];
                for vars = {'Pathpoints' 'PathHandles' 'Distance' 'MetCost' 'Time' 'Shadows' 'SublimatedWater' 'HeaterPower' 'BatteryCapacity' 'TSuit'
'StageSuitHeat'}
                    [Data.(vars{1}){[en,en+Data.NumExp]}] = deal([]);
                end
                set([Data.MPpathenH,Data.MPdistNH,Data.MPcostNH,...
                     Data.MPtimeH],'String','')
                set(MPfig,'UserData',Data)
        end

        xCoordTemp = input(['X coordinate of waypoint in UTM coordinates, Zone ',num2str(Data.UTMzone),': ']);
        yCoordTemp = input(['Y coordinate of waypoint in UTM coordinates, Zone ',num2str(Data.UTMzone),': ']);
        disp(' ');

        maxX = Data.xllcorner + (Data.Cols-1)*Data.Resolution;
        maxY = Data.yllcorner + (Data.Rows-1)*Data.Resolution;

        if (xCoordTemp < maxX) && (xCoordTemp > Data.xllcorner) && ...
           (yCoordTemp < maxY) && (yCoordTemp > Data.yllcorner)

           xdistance = xCoordTemp - Data.xllcorner;
           ydistance = yCoordTemp - Data.yllcorner;

           checkX = round(xdistance/Data.Resolution);
           checkY = Data.Rows - round(ydistance/Data.Resolution);

        b = size(Data.Obstacles);

           if (checkY >= b(1)) || (checkX >= b(2))
              disp('Point is too close to edge of map.')
           elseif (Data.Obstacles(checkY+1, checkX+1) == 0)
              Data.Waypoints{en} = [Data.Waypoints{en}; checkX checkY,...
                    Data.Elevations(checkY+1,checkX+1)];
           else
              disp('WARNING:  Waypoint is on an obstacle.')
           end

        else
           disp('WARNING:  Waypoint is outside the map.')
        end

        delete(Data.WayHandles{en})      % Clear prev waypoints
        if ~isempty(Data.Waypoints{en})  % Plot waypoints in GUI
                hold on
                Data.WayHandles{en}(1) = scatter3(Data.MPaxes,...
                   Data.Waypoints{en}(:,1),Data.Waypoints{en}(:,2),...
                   Data.Waypoints{en}(:,3)+.05*Data.Eldiff,...
                   120,Data.Ecolors{mod(en-1,10)+1},'filled',...
                   'ButtonDownFcn',sprintf('sextant("Mission","Click",%d)',en));
                %txt = {'H',{'H';num2str((1:size(Data.Waypoints{en},1)-1).')}};
                txt = {'1',{num2str((1:size(Data.Waypoints{en},1)).')}};
                WPtxt = text(Data.Waypoints{en}(:,1),...
                   Data.Waypoints{en}(:,2)-.007*Data.Rows,...
                   Data.Waypoints{en}(:,3)+.07*Data.Eldiff,...
                   txt{1+(size(Data.Waypoints{en},1)>1)},...
                   'HorizontalAlignment','center',...
                   'VerticalAlignment','bottom',...
                   'Color',[1 1 1],'FontWeight','bold','HitTest','off');
                Data.WayHandles{en} = [Data.WayHandles{en}(1);WPtxt];
                hold off
                set(Data.MPwaytitleH,'String',sprintf('WP %d:',...
                            size(Data.Waypoints{en},1)))
        else
                set(Data.MPwaytitleH,'String','Start:')
                Data.WayHandles{en} = [];
        end
```

```
    end
    set(MPfig,'UserData',Data)

case 'Help'
    macpc = {'CTRL+','RIGHT-'};
    helpmsg = ['MISSION PLANNER GUI\n','_____\n\n',...
            'LEFT-CLICK:     Add waypoints or terrain characteristics\n\n',...
            'SHIFT+CLICK:  Clear waypoints or terrain characteristics\n\n',...
            'SHIFT+CLICK ON PATH: Find return home path\n\n',...
            'DOUBLE-CLICK:  Heighten terrain characteristics\n\n',...
            macpc{1+ispc},'CLICK:  Display terrain or waypoint data, select paths\n',...
            '_____\n\n',...
            'Map Info & EVA Input buttons:  Reopen menus to change mission data\n\n',...
            'Scale menu:  Update axes scaling with selected units\n\n',...
            'Waypoints & Terrain buttons:  Select edit mode\n\n',...
            'Explorer menu:  Select the current explorer\n\n',...
            'Terrain menu:   Select the terrain characteristic to display\n\n',...
            'Size Control:   Adjust the size of the terrain edit rectangle\n\n',...
            'Terrain ON & OFF:  Turn terrain characteristic display on & off\n\n',...
            'Run PATH button:  Run the PATH-based optimization to find traverse paths\n\n',...
            'Render buttons:  Select the terrain render mode'];
    questdlg(sprintf(helpmsg),'SEXTANT Help','OK','OK');

case {'Map' 'Input'}
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set(MPfig,'Visible','off')
    try delete(Data.minfo), catch end  %#ok<CTCH> % Clear map data text if exists
    sextant(Select,0,Data)  % Call to 'Map' OR 'Input'

case 'Scale'  % Change axes scale
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    distU = get(Data.MPscaleH,'Value');
    distR = {1,'%.0f','Meters'; .001,'%.2f','Km';...
          3.28084,'%.0f','Feet'; .0006213712,'%.2f','Miles'};
    [xsize,ysize] = deal(Data.Resolution*Data.Cols,Data.Resolution*Data.Rows);
    mapsize = max(xsize,ysize);
    mag = floor(log10(mapsize*distR{distU,1}));
    scale = round(mapsize*distR{distU,1}/10^mag)*10^(mag-1);
    tscale = scale/distR{distU,1}/Data.Resolution;
    zmag = floor(log10(Data.Eldiff*distR{distU,1}));
    zscale = round(Data.Eldiff*distR{distU,1}/10^zmag)*10^(zmag-1)*2;
    set(Data.MPaxes,'XTick',tscale:tscale:Data.Cols,...
                'YTick',mod(Data.Rows-.001/Data.Resolution,tscale)+.001/Data.Resolution:tscale:Data.Rows,...
                'ZTick',Data.Elmin:zscale/distR{distU,1}:Data.Elmin+1.25*Data.Eldiff,...
                'XTickLabel',scale:scale:xsize*distR{distU,1},...
                'YTickLabel',ysize*distR{distU,1}-(mod((ysize-.001)*distR{distU,1},...
                        scale)+.001*distR{distU,1}):-scale:0,...
                'ZTickLabel',0:zscale:1.25*Data.Eldiff*distR{distU,1})
    %'YTick',mod(Data.Rows-1,tscale)+1:tscale:Data.Rows,...

    % So I don't really know what's going on with YTick
    get(Data.MPaxes,'YTick');
    get(Data.MPaxes,'YTickLabel');

    set(Data.MPdistUH,'String',distR{distU,3})
    if ~isempty(get(Data.MPdistNH,'String'))
       path = get(Data.MPpathenH,'String');
       en = get(Data.MPexpnumH,'Value') + Data.NumExp*(path(end)=='R');
       set(Data.MPdistNH,'String',sprintf(distR{distU,2},...
                        Data.Distance{en}(end)*distR{distU,1}))
    end
    if ~isempty(Data.Pathpoints{1})
       % So, we want it to replot the figures for all of the main
       % explorers (no return home paths)
       Data.Scale = distU;
       Data.Newpaths = 1:1:Data.NumExp;
       Data.R = '';
       set(MPfig,'UserData',Data)
       sextant('Mission','Plots','');
    end

case {'WP' 'TER'}  % Select edit mode
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set(Data.MPwpmodeH,'Value',strcmp(Select,'WP'))
    set(Data.MPtermodeH,'Value',strcmp(Select,'TER'))
    state = {'off','on','inactive'};
    set([Data.MPexptxtH,Data.MPexpnumH,Data.MPwaytitleH,Data.MPterOnH,...
```

```
        Data.MPterOffH],'Enable',state{1+strcmp(Select,'WP')})
    set([Data.MPtersizetxtH,Data.MPtersizepH,Data.MPtersizemH],...
        'Enable',state{1+strcmp(Select,'TER')})
    set(Data.MPtersizeH,'Enable',state{1+2*strcmp(Select,'TER')})
    if strcmp(Select,'TER')
        sextant('Mission','TerON',[])
    end

case 'SelExp'  % Select explorer for waypoint edits & data display
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    enR = get(Data.MPexpnumH,'Value');
    ecolors = {Data.Ecolors{1:8},'k','k'};
    set(Data.MPexpnumH,'ForegroundColor',ecolors{mod(enR-1,10)+1})
    txt = {'Start:',sprintf('WP %d:',size(Data.Waypoints{enR},1))};
    title = txt{1 + ~isempty(Data.Waypoints{enR})};
    set(Data.MPwaytitleH,'String',title)
    set(Data.MPexptxtH,'String',[Data.Explorers{enR},': '])
    if ~isempty(Data.Pathpoints{enR})  % Set the cost displays
        en = enR + ~isempty(Data.R)*Data.NumExp;
        set(Data.MPpathenH,'String',sprintf('%d%s',enR,Data.R),...
                    'ForegroundColor',ecolors{mod(enR-1,10)+1})
        distU = get(Data.MPscaleH,'Value');
        distR = {1,'%.0f'; .001,'%.2f'; 3.28084,'%.0f'; .0006213712,'%.2f'};
        set(Data.MPdistNH,'String',sprintf(distR{distU,2},...
                            Data.Distance{en}(end)*distR{distU,1}))
        costR = [.2521644 1 1.055056];
        set(Data.MPcostNH,'String',sprintf('%.1f',...
            Data.MetCost{en}(end)*costR(get(Data.MPcostUH,'Value'))))
        hourmin = [floor(Data.Time{en}(end)/3600),...
                round(rem(Data.Time{en}(end),3600)/60)];
        set(Data.MPtimeH,'String',sprintf('%d:%d%d',...
                            hourmin(1),zeros(hourmin(2)<10),hourmin(2)))
    else
        set([Data.MPpathenH,Data.MPdistNH,Data.MPcostNH,Data.MPtimeH],'String','')
    end

case {'TerON' 'TerOFF'}  % Select terrain map to display
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set(Data.MPterOnH,'Value',strcmp(Select,'TerON'))
    set(Data.MPterOffH,'Value',strcmp(Select,'TerOFF'))
    state = {'off' 'on'};
    set(Data.MPtertypeH,'Enable',state{1+strcmp(Select,'TerON')})
    ter = get(Data.MPtertypeH,'Value');
    tcm = {'Elevations','ColorObsRed','SoilMech','SciReturn','Other'};
    set(Data.MPaxes,'CLim',Data.ColorLim(1+ter*strcmp(Select,'TerON'),:))
    set(Data.MPsurf,'CData',Data.(tcm{1+ter*strcmp(Select,'TerON')}))
    if strcmp(Select,'TerOFF') || ter==1
        rendm = find([get(Data.MPrendmH1,'Value'),...
            get(Data.MPrendmH2,'Value'),get(Data.MPrendmH3,'Value')]);
    else
        rendm = ter+2;
    end
    colormap(Data.MPaxes,Data.Colors(:,3*rendm-2:3*rendm))

case {'SimpON' 'SimpOFF'}
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set(Data.MPsimplifyingH1,'Value',strcmp(Select,'SimpON'))
    set(Data.MPsimplifyingH2,'Value',strcmp(Select,'SimpOFF'))
    Data.Simplifying = Select;
    set(MPfig,'UserData',Data)

case {'TSp' 'TSm'}  % Change terrain map edit rectangle size
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    sizes = [.1 .2 .3 .4 .5 .6 .7 .8 .9 1 1.5 2 2.5 3 4 5 6 7 8 9 10];
    ces = find(sizes==Data.TEsize);
    if strcmp(Select,'TSp') && ces < 21
        Data.TEsize = sizes(ces+1);
    elseif strcmp(Select,'TSm') && ces > 1
        Data.TEsize = sizes(ces-1);
    end
    set(Data.MPtersizeH,'String',Data.TEsize)
    set(MPfig,'UserData',Data)

case 'CostU'    % Change cost units
    MPfig = gcf;
    Data = get(MPfig,'UserData');
```

```
    if ~isempty(get(Data.MPcostNH,'String'))
       costR = [.2521644 1 1.055056];  % Ratios: Kcal, BTU, KJ
       set(Data.MPcostNH,'String',sprintf('%.1f',...
          Data.MetCost{get(Data.MPexpnumH,'Value')}(end) * ...
          costR(get(Data.MPcostUH,'Value'))));
    end
    if ~isempty(Data.Pathpoints{1})
       % So, we want it to replot the figures for all of the main
       % explorers (no return home paths)
       Data.CostScale = get(Data.MPcostUH,'Value');
       Data.Newpaths = 1:1:Data.NumExp;
       Data.R = '';
       set(MPfig,'UserData',Data)
       sextant('Mission','Plots','');
    end

 case {1 2 3}  % Change render mode
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set(Data.MPrendmH1,'Value',Select==1)
    set(Data.MPrendmH2,'Value',Select==2)
    set(Data.MPrendmH3,'Value',Select==3)
    if get(Data.MPterOffH,'Value') || get(Data.MPtertypeH,'Value')==1
       colormap(Data.MPaxes,Data.Colors(:,3*Select-2:3*Select))
    end

 case 'PATH'  % Run PATH button or return home path
    paths2do = Data;  % For return home path, this is the explorer #
    MPfig = gcf;
    Data = get(MPfig,'UserData');
    set([Data.MPhelpH,Data.MPmapiH,Data.MPevaiH],'enable','off')
    try delete(Data.minfo), catch end  %#ok<CTCH>
    if isempty(paths2do)  % Nominal case (non return home)
       % There are no paths to calculate
       Data.MPfigH = uisuspend(MPfig);
       [numWP,hasPath] = deal(zeros(Data.NumExp,1));
       for i = 1:Data.NumExp        % Find which explorers need paths
          numWP(i) = size(Data.Waypoints{i},1);
          hasPath(i) = ~isempty(Data.Pathpoints{i});
       end
       paths2do = find(numWP>=2 & ~hasPath).'; % Array of explorer #'s
    end
    if ~isempty(paths2do)    % Save all edits & calculate the new paths
       Data.Path = true;
       try save([Data.EVAname,'_Data'],'-struct','Data','Waypoints','-append'), catch end %#ok<CTCH>
       sextant('SaveMaps','-append',Data)      % Call to 'SaveMaps'
       [Data.ObstEd,Data.SoilEd,Data.SciREd,...  % Reset edit booleans
        Data.OtheEd,Data.WayPEd] = deal(false);
       sextant('PATH',paths2do,Data)              % Call to 'PATH'
       Data = get(MPfig,'UserData');  % Path data saved
       if ~isempty(Data.Newpaths)
          hold on
          for en = Data.Newpaths(end:-1:1)       % Plot the new paths
             if isempty(Data.R)  % Make waypoints big & green
                set(Data.WayHandles{en}(1),'SizeData',200,'CData',[0 1 0])
             end
             C = Data.Ecolors{mod(mod(en-1,Data.NumExp),10)+1};
             axes(Data.MPaxes)  % Make this GUI's axes current
             Data.PathHandles{en}(1) = line(Data.Pathpoints{en}(:,1),...
                Data.Pathpoints{en}(:,2),...
                Data.Pathpoints{en}(:,3)+.05*Data.Eldiff,...
                'Color',C,'LineWidth',4,'Marker','o',...
                'MarkerEdgeColor',C,'MarkerFaceColor',C,'MarkerSize',5,...
                'ButtonDownFcn',sprintf('sextant("Mission","Click",%d)',en));
          end
          hold off
          if ~isempty(Data.R)  % Set to dotted line for return home path
             set(Data.PathHandles{en}(1),'LineStyle','--')
          end
          set(Data.MPexpnumH,'Value',mod(Data.Newpaths(1)-1,Data.NumExp)+1)
       end
       set(MPfig,'UserData',Data)        % Store data
       sextant('Mission','SelExp',[])  % Set cost displays
       sextant('Mission','Shadows',[])  % Shadowing
    else
       message = 'There are no new traverse paths to find.';
       if any(numWP==1)
          message = [message,'\n\nA path requires a start and at least 1 waypoint.'];
       end
       Data.minfo = helpdlg(sprintf(message),'SEXTANT');
```

```
        set(MPfig,'UserData',Data)
    end
    set([Data.MPhelpH,Data.MPmapiH,Data.MPevaiH],'enable','on')
    uirestore(Data.MPfigH)
    if ~isempty(Data.errpath)  % If error on any path
        message = ['An error occured on path%s: ',...
                num2str(Data.errpath,' %d,')];
        message = [message(1:end-1),'%s\n\nMake sure waypoints are not\n',...
                'enclosed by obstacles.'];
        s = {'','s'};
        errordlg(sprintf(message,s{1+(length(Data.errpath)>1)},Data.R),'Path Error');
    end

case 'Plots'
    % Specify the constraints:
    timeConstraint = [8 8];
    metCostConstraint = [2868 2868];
    distConstraint = [20 20];

    MPfig = gcf;
    Data = get(MPfig,'UserData');
    % Data.Newpaths will give you all the explorers for which you are
    % currently calculating a path
    if ~isempty(Data.Newpaths)
        en = Data.Newpaths;
    else
        en = get(Data.MPexpnumH,'Value');
    end

    % Gives a "1" if this is a return home path.
    % You can't calculate regular and return home paths at the same
    % time
    isReturn = Data.R;

    % This should convert the scale
    distScalingFactor = [1, .001, 3.28084, .0006213712];
    distScalingLabels = {'m','km','ft','mi'};

    metCostScalingFactor = [.2521644 1 1.055056];
    metCostScalingLabels = {'Kcal','BTU','KJ'};

    % For each explorer
    for i=1:length(en)
        enR = en(i);

        % Scale the distance, elevation, and met cost
        scaledDistance = Data.Distance{enR}*distScalingFactor(Data.Scale);
        scaledElevation = Data.Pathpoints{enR}(:,3)*distScalingFactor(Data.Scale);
        scaledMetCost = Data.MetCost{enR}*metCostScalingFactor(Data.CostScale);

        % Determine the number of waypoints in this traverse
        numWaypoints = size(Data.Waypoints{enR},1);

        % Determine the indices of each Activity Point
        pp = zeros(numWaypoints,1);
        for n=1:numWaypoints
            % Get the coordinates of each Activity Point
            clx = Data.Waypoints{enR}(n,1);
            cly = Data.Waypoints{enR}(n,2);
            % Find the pathpoint numbers that correspond to each
            % Activity Point
            pp(n,1) = find((Data.Pathpoints{enR}(:,1)==clx) & ...   %pp=Point along
                    (Data.Pathpoints{enR}(:,2)==cly),1,'first');  %  traverse path
            pp(n,2) = find((Data.Pathpoints{enR}(:,1)==clx) & ...   %pp=Point along
                    (Data.Pathpoints{enR}(:,2)==cly),1,'last');   %  traverse path
        end

        % Create vectors with the waypoint information so you can plot
        % this as green circles
        clear waypointDist waypointTime waypointMetCost waypointSublimatedWater waypointHeaterPower ...
            waypointBatteryCapacity waypointElevation waypointTSuit
        waypointDist = zeros(numWaypoints,2);
        waypointTime = waypointDist;
        waypointMetCost = waypointTime;
        waypointSublimatedWater = waypointMetCost;
        waypointHeaterPower = waypointSublimatedWater;
        waypointBatteryCapacity = waypointHeaterPower;
        waypointElevation = waypointBatteryCapacity;
        waypointTSuit = waypointElevation;
```

Appendices

```
        for m=1:numWaypoints
            waypointDist(m,1:2) = scaledDistance(pp(m,1:2));
            waypointTime(m,1:2) = Data.Time{enR}(pp(m,1:2))/3600;
            waypointMetCost(m,1:2) = scaledMetCost(pp(m,1:2));
            waypointElevation(m,1:2) = scaledElevation(pp(m,1:2));
            if ~isempty(Data.Shadows{enR})
                if strcmp(Data.Explorers{enR},'Rover')
                    waypointBatteryCapacity(m,1:2) = Data.BatteryCapacity{enR}(pp(m,1:2));
                else
                    waypointSublimatedWater(m,1:2) = Data.SublimatedWater{enR}(pp(m,1:2));
                    waypointHeaterPower(m,1:2) = Data.HeaterPower{enR}(pp(m,1:2));
                    waypointTSuit(m,1:2) = Data.TSuit{enR}(pp(m,1:2));
                end
            end
        end

        % Plot the elevation
        whitebg([1 1 1])
        elevationNorm = min(scaledElevation);
        figure
        hold on
        plot(scaledDistance,scaledElevation-elevationNorm,'linewidth',2)
        plot(waypointDist,waypointElevation-elevationNorm,'go','MarkerFaceColor','g','MarkerSize',10)
        hold off
        xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',16)
        ylabel(['Elevation (',distScalingLabels{Data.Scale},')'],'fontsize',16)
        set(gca,'FontSize',16)

        % Get the color for the current explorer
        C = Data.Ecolors{mod(mod(enR-1,Data.NumExp),10)+1};
        Col = C;

        % Plot the energy cost with respect to distance and time
        a = figure;
        subplot(1,2,1)
        hold on
        % Plot the green bars for the locations at Activity Points
        for n=1:size(pp,1)
            plot(scaledDistance(pp(n,1):pp(n,2)),scaledMetCost(pp(n,1):pp(n,2)),'g','LineWidth',10)
        end
        plot(scaledDistance,scaledMetCost,'Color',C,'LineWidth',2)
        plot([0 distConstraint(1)],metCostConstraint,'r--','LineWidth',2)
        plot(distConstraint,[0 metCostConstraint(1)],'r--','LineWidth',2)
%           plot(waypointDist,waypointMetCost,'go','MarkerFaceColor','g','MarkerSize',10)
        hold off
        xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',14)
        if strcmp(Data.Explorers{enR},'Rover')
            ylabel(['Energy Cost (',metCostScalingLabels{Data.CostScale},')'],'fontsize',14)
        else
            ylabel(['Metabolic Cost (',metCostScalingLabels{Data.CostScale},')'],'fontsize',14)
        end
        set(gca,'fontsize',14)

        subplot(1,2,2)
        hold on
        for n=1:size(pp,1)
            plot(Data.Time{enR}(pp(n,1):pp(n,2))/3600,scaledMetCost(pp(n,1):pp(n,2)),'g','LineWidth',10)
        end
        plot(Data.Time{enR}/3600,scaledMetCost,'Color',C,'LineWidth',2.5)
        plot([0 timeConstraint(1)],metCostConstraint,'r--','LineWidth',2)
        plot(timeConstraint,[0 metCostConstraint(1)],'r--','LineWidth',2)
        plot(waypointTime,waypointMetCost,'go','MarkerFaceColor','g','MarkerSize',10)
        hold off
        xlabel('Time (Hours)','fontsize',14)
        if strcmp(Data.Explorers{enR},'Rover')
            ylabel(['Energy Cost (',metCostScalingLabels{Data.CostScale},')'],'fontsize',14)
        else
            ylabel(['Metabolic Cost (',metCostScalingLabels{Data.CostScale},')'],'fontsize',14)
        end
        set(gca,'fontsize',14)

        set(a,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
                round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

    % If we have thermal information...
    if ~isempty(Data.Shadows{enR})
        plotWaypoints(:,1) = Data.Waypoints{enR}(:,1)*Data.Resolution;
        plotWaypoints(:,2) = (Data.Rows - Data.Waypoints{enR}(:,2))*Data.Resolution;
        plotPathpoints(:,1) = Data.Pathpoints{enR}(:,1)*Data.Resolution;
        plotPathpoints(:,2) = (Data.Rows - Data.Pathpoints{enR}(:,2))*Data.Resolution;
```

```
plotWaypoints = plotWaypoints/1000;
plotPathpoints = plotPathpoints/1000;

% Plot the traverse with sun illumination
sunTraverse = figure;
axis([0 abs(Data.xLimits(2)-Data.xLimits(1)) 0 abs(Data.yLimits(2)-Data.yLimits(1))])
hold on
% Plot the first Activity Point:
plot(plotWaypoints(1,1),plotWaypoints(1,2),'go','MarkerFaceColor','g','MarkerSize',12)
iiiiiiiiiiiiii = 2;

for n=2:size(plotPathpoints,1)
   color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
   plot([plotPathpoints(n-1,1) plotPathpoints(n,1)],...
      [plotPathpoints(n-1,2) plotPathpoints(n,2)],'Color',[color color color],'LineWidth',3);
   if ~isempty(intersect(pp(:,1),n))
      plot(plotWaypoints(iiiiiiiiiiiiiii,1),plotWaypoints(iiiiiiiiiiiiii,2),'go','MarkerFaceColor','g','MarkerSize',12)
      iiiiiiiiiiiiii = iiiiiiiiiiiiii+1;
   end
   %pause(0.3)
   % Uncomment the pause above if you want to see the traverse
   % plot piece-by-piece.  Note than when this is happening
   % you can't click on another Figure window or things will
   % get crazy.
end
plot(plotWaypoints(:,1),plotWaypoints(:,2),'go','MarkerFaceColor','g','MarkerSize',10)
hold off
set(sunTraverse,'Position',[10 10 800 800],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])
set(gca,'Fontsize',12,'Color',[0 .5 .6])

% If the explorer is a rover
if strcmp(Data.Explorers{enR},'Rover')
   % Plot the battery capacity with respect to distance
   b=figure;
   whitebg([1 1 1])
   hold on
   for n=1:size(pp,1)
      plot(scaledDistance(pp(n,1):pp(n,2)),Data.BatteryCapacity{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
   end
   plot(scaledDistance,Data.BatteryCapacity{enR}/1000,'Color',C,'LineWidth',2)
   %plot(waypointDist,waypointBatteryCapacity,'go','MarkerFaceColor','g','MarkerSize',10)
   hold off
   xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',16)
   ylabel('Energy Level (kW-hr)','fontsize',16)
   set(b,'Position',[100 100 1800 800],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])
   set(gca,'Fontsize',16)

   % Plot the battery capacity with respect to distance, with
   % shadowing
   c=figure;
   hold on
   for n=1:size(pp,1)
      plot(scaledDistance(pp(n,1):pp(n,2)),Data.BatteryCapacity{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
   end
   for n=2:size(Data.Pathpoints{enR},1)
      color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
      plot([scaledDistance(n-1) scaledDistance(n)],[Data.BatteryCapacity{enR}(n-1) Data.BatteryCapacity{enR}(n)]/1000,'Color',[color color
color],'LineWidth',2.5)
   end
   %plot(waypointDist,waypointBatteryCapacity,'go','MarkerFaceColor','g','MarkerSize',10)
   hold off
   xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'FontSize',16)
   ylabel('Energy Level (kW-hr)','FontSize',16)
   set(gca,'Fontsize',16,'Color',[0 .5 .6])

%     set(c,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
%           round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])
   set(c,'Position',[100 100 1800 800],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

   % Plot the battery capacity with respect to time
   bb=figure;
   whitebg([1 1 1])
   hold on
   for n=1:size(pp,1)
      plot(Data.Time{enR}(pp(n,1):pp(n,2))/3600,Data.BatteryCapacity{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
   end
   plot(Data.Time{enR}/3600,Data.BatteryCapacity{enR}/1000,'Color',C,'LineWidth',2)
   %plot(waypointDist,waypointBatteryCapacity,'go','MarkerFaceColor','g','MarkerSize',10))
   hold off
   xlabel('Time (Hours)','FontSize',16)
```

```
            ylabel('Energy Level (kW-hr)','fontsize',16)
            set(bb,'Position',[100 100 1800 800],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])
            set(gca,'Fontsize',16)

            % Plot the battery capacity with respect to time, with
            % shadowing
            cc=figure;
            hold on
            for n=1:size(pp,1)
               plot(Data.Time{enR}(pp(n,1):pp(n,2))/3600,Data.BatteryCapacity{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
            end
            for n=2:size(Data.Pathpoints{enR},1)
               color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
               plot([Data.Time{enR}(n-1) Data.Time{enR}(n)]/3600,[Data.BatteryCapacity{enR}(n-1) Data.BatteryCapacity{enR}(n)]/1000,'Color',[color color
color],'LineWidth',2.5)
            end
            %plot(waypointDist,waypointBatteryCapacity,'go','MarkerFaceColor','g','MarkerSize',10)
            hold off
            xlabel('Time (Hours)','FontSize',16)
            ylabel('Energy Level (kW-hr)','FontSize',16)
            set(gca,'Fontsize',16,'Color',[0 .5 .6])
            set(cc,'Position',[100 100 1800 800],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

         % If the explorer is an astronaut...
         else
            % Plot sublimated water and heater power with respect to
            % distance and time together
            b=figure;
            whitebg([1 1 1])
            subplot(1,2,1)
            hold on
            for n=1:size(pp,1)
               plot(scaledDistance(pp(n,1):pp(n,2)),Data.SublimatedWater{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
            end
            plot(scaledDistance,Data.SublimatedWater{enR},'Color',C,'LineWidth',2)
            plot(waypointDist,waypointSublimatedWater,'go','MarkerFaceColor','g','MarkerSize',10)
            %plot([0 distConstraint(1)],metCostConstraint,'r--','LineWidth',2)
            %plot(distConstraint,[0 metCostConstraint(1)],'r--','LineWidth',2)
            hold off
            xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
            ylabel('Mass of Water Sublimated (kg)','fontsize',12)
            %text(waypointDist(3),waypointMetCost(3),'3')

            subplot(1,2,2)
            hold on
            for n=1:size(pp,1)
               plot(scaledDistance(pp(n,1):pp(n,2)),Data.HeaterPower{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
            end
            plot(scaledDistance,Data.HeaterPower{enR}/1000,'Color',C,'LineWidth',2)
            plot(waypointDist,waypointHeaterPower/1000,'go','MarkerFaceColor','g','MarkerSize',10)
            %plot([0 timeConstraint(1)],metCostConstraint,'r--','LineWidth',2)
            %plot(timeConstraint,[0 metCostConstraint(1)],'r--','LineWidth',2)
            hold off
            xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
            ylabel('Heater Energy (kJ)','fontsize',12)
            set(gca,'Fontsize',12)

             set(b,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
                    round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

            % Plot sublimated water and heater power with respect to
            % time and distance, with shadowing
            c=figure;
            subplot(1,2,1)
            hold on
            for n=1:size(pp,1)
               plot(scaledDistance(pp(n,1):pp(n,2)),Data.SublimatedWater{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
            end
            for n=2:size(Data.Pathpoints{enR},1)
               color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
               plot([scaledDistance(n-1) scaledDistance(n)],[Data.SublimatedWater{enR}(n-1) Data.SublimatedWater{enR}(n)],'Color',[color color
color],'LineWidth',2.5)
            end
            %plot(waypointDist,waypointSublimatedWater,'go','MarkerFaceColor','g','MarkerSize',10)
            hold off
            xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
            ylabel('Mass of Water Sublimated (kg)','fontsize',12)
            %text(waypointDist(3),waypointMetCost(3),'3')
            set(gca,'Fontsize',12,'Color',[0 .5 .6])
```

```
subplot(1,2,2)
hold on
for n=1:size(pp,1)
   plot(scaledDistance(pp(n,1):pp(n,2)),Data.HeaterPower{enR}(pp(n,1):pp(n,2))/1000,'g','LineWidth',10)
end
for n=2:size(Data.Pathpoints{enR},1)
   color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
   plot([scaledDistance(n-1) scaledDistance(n)],[Data.HeaterPower{enR}(n-1) Data.HeaterPower{enR}(n)]/1000,'Color',[color color color],'LineWidth',2.5)
end
%plot(waypointDist,waypointHeaterPower/1000,'go','MarkerFaceColor','g','MarkerSize',10)
hold off
xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
ylabel('Heater Energy (kJ)','fontsize',12)
set(gca,'fontsize',12,'Color',[0 .5 .6])

set(c,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
          round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

% Plot space suit temperature with repsect to distance and
% time
d = figure;
whitebg([1 1 1])
subplot(1,2,1)
hold on
for n=1:size(pp,1)
   plot(scaledDistance(pp(n,1):pp(n,2)),Data.TSuit{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
end
plot(scaledDistance,Data.TSuit{enR},'linewidth',2)
%plot(waypointDist,waypointTSuit,'go','MarkerFaceColor','g','MarkerSize',10)
hold off
xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
ylabel('External Space Suit Temperature (K)','fontsize',12)

subplot(1,2,2)
whitebg([1 1 1])
hold on
for n=1:size(pp,1)
   plot(Data.Time{enR}(pp(n,1):pp(n,2))/3600,Data.TSuit{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
end
plot(Data.Time{enR}/3600,Data.TSuit{enR},'linewidth',2)
%plot(waypointTime,waypointTSuit,'go','MarkerFaceColor','g','MarkerSize',10)
hold off
xlabel('Time (Hours)','fontsize',12)
ylabel('External Space Suit Temperature (K)','fontsize',12)
set(gca,'fontsize',12)
set(d,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
          round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

% Plot space suit temperature with respect to distance and time,
% with shadows
e = figure;
subplot(1,2,1)
hold on
for n=1:size(pp,1)
   plot(scaledDistance(pp(n,1):pp(n,2)),Data.TSuit{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
end
for n=2:size(Data.Pathpoints{enR},1)
   color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
   plot([scaledDistance(n-1) scaledDistance(n)],[Data.TSuit{enR}(n-1) Data.TSuit{enR}(n)],'Color',[color color color],'LineWidth',2.5)
end
%plot(waypointDist,waypointTSuit,'go','MarkerFaceColor','g'
%,'MarkerSize',10)
hold off
xlabel(['Distance (',distScalingLabels{Data.Scale},')'],'fontsize',12)
ylabel('External Space Suit Temperature (K)','fontsize',12)
set(gca,'FontSize',12,'Color',[0 .5 .6])

% Plot space suit temperature with respect to time, with
% shadows
subplot(1,2,2)
hold on
for n=1:size(pp,1)
   plot(Data.Time{enR}(pp(n,1):pp(n,2))/3600,Data.TSuit{enR}(pp(n,1):pp(n,2)),'g','LineWidth',10)
end
for n=2:size(Data.Pathpoints{enR},1)
   color = (Data.Shadows{enR}(n-1)+Data.Shadows{enR}(n))/2;
   plot([Data.Time{enR}(n-1) Data.Time{enR}(n)]/3600,[Data.TSuit{enR}(n-1) Data.TSuit{enR}(n)],'Color',[color color color],'LineWidth',2.5)
end
%plot(waypointTime,waypointTSuit,'go','MarkerFaceColor','g','MarkerSize',10)
hold off
```

```
            xlabel('Time (Hours)','fontsize',12)
            ylabel('External Space Suit Temperature (K)','fontsize',12)
            set(gca,'FontSize',12,'Color',[0 .5 .6])
            set(e,'Position',[round(Data.Scrsize(3)/32) round(Data.Scrsize(4)/2),...
                    round(Data.Scrsize(3)*10/16) round(Data.Scrsize(4)*13/32)],'Name',['Explorer #',num2str(enR- ~isempty(isReturn)*Data.NumExp),isReturn])

            % Plot all heat fluxes with respect to time
            figure
            whitebg([1 1 1])
            hold on
            for n=1:length(Data.Time{enR})-1
               plot([Data.Time{enR}(n) Data.Time{enR}(n+1)]/60,[Data.StageSuitHeat{enR}(n,1) Data.StageSuitHeat{enR}(n,1)],'b','linewidth',4)
               plot([Data.Time{enR}(n) Data.Time{enR}(n+1)]/60,[Data.StageSuitHeat{enR}(n,2) Data.StageSuitHeat{enR}(n,2)],'r','linewidth',4)
               plot([Data.Time{enR}(n) Data.Time{enR}(n+1)]/60,[Data.StageSuitHeat{enR}(n,3) Data.StageSuitHeat{enR}(n,3)],'g','linewidth',4)
               plot([Data.Time{enR}(n) Data.Time{enR}(n+1)]/60,[Data.StageSuitHeat{enR}(n,4) Data.StageSuitHeat{enR}(n,4)],'m','linewidth',4)
            end
            xlabel('Time (min)','fontsize',12)
            ylabel('Heat Flux (W)','fontsize',12)
            legend('Electronics','Astronaut','External','Total')
            set(gca,'FontSize',12)
            hold off

            % Plot all heats with respect to stage
            figure
            whitebg([1 1 1])
            hold on
            for n=1:length(Data.StageSuitHeat{enR})-1
               plot(n,Data.StageSuitHeat{enR}(n,1)*(Data.Time{enR}(n+1)-Data.Time{enR}(n))/1000,'bs','MarkerFaceColor','b','MarkerSize',8)
               plot(n,Data.StageSuitHeat{enR}(n,2)*(Data.Time{enR}(n+1)-Data.Time{enR}(n))/1000,'r^','MarkerFaceColor','r','MarkerSize',8)
               plot(n,Data.StageSuitHeat{enR}(n,3)*(Data.Time{enR}(n+1)-Data.Time{enR}(n))/1000,'gv','MarkerFaceColor','g','MarkerSize',8)
               plot(n,Data.StageSuitHeat{enR}(n,4)*(Data.Time{enR}(n+1)-Data.Time{enR}(n))/1000,'mo','MarkerFaceColor','m','MarkerSize',8)
            end
            xlabel('Stage','fontsize',12)
            ylabel('Heat (kJ)','fontsize',12)
            legend('Electronics','Astronaut','External','Total')
            set(gca,'FontSize',12)
            hold off
         end
      end

      Data.Newpaths = '';
      set(MPfig,'UserData',Data)
      end

   case 'Click'  % Mouse click: Waypoints, terrain edits, paths, data display
      ClOnPath = Data;  % If a path clicked on, this is the explorer #
      MPfig = gcf;
      Data = get(MPfig,'UserData');
      Task = get(MPfig,'SelectionType');       % Click type
      clpt = get(Data.MPaxes,'CurrentPoint');  % Click location
      clx = max(min(round((clpt(1,1)+clpt(2,1))/2),Data.Cols-1),0);
      cly = max(min(round((clpt(1,2)+clpt(2,2))/2),Data.Rows-1),0);
      try delete(Data.minfo), catch end  %#ok<CTCH> % Clear map text
      switch Task
         case {'normal' 'extend' 'open'} % Left-Click,Shift+Click,Double
            if strcmp(Task,'extend') && ~isempty(ClOnPath) && numel(ClOnPath)==1 && ...
                  ClOnPath<=Data.NumExp && ~isempty(Data.Pathpoints{ClOnPath}) && ...
                  ~Data.Obstacles(cly+1,clx+1) % Shift+Click path: return home
               en = ClOnPath+Data.NumExp;
               hold on
               Data.WayHandles{en} = scatter3(Data.MPaxes,clx,cly,...
                  Data.Elevations(cly+1,clx+1)+.05*Data.Eldiff,300,[0 0 0],'filled');
               hold off
               Data.MPfigH = uisuspend(MPfig);
               Choice = questdlg(sprintf('Calculate return home path for Explorer %d?',...
                           ClOnPath),'Return Home','Yes','No','Yes');
               if strcmp(Choice,'No')
                  delete(Data.WayHandles{en})
                  Data.WayHandles{en} = [];
                  uirestore(Data.MPfigH)
               else
                  Data.Explorers{en} = Data.Explorers{ClOnPath};
                  delete(Data.PathHandles{en}) %Clear prev return path
                  for vars = {'SublimatedWater' 'HeaterPower' 'BatteryCapacity' 'TSuit' 'StageSuitHeat'}
                     [Data.(vars{1}){en}] = deal([]);
                  end
                  Data.Waypoints{en} = [clx cly Data.Elevations(cly+1,clx+1);
                              Data.Waypoints{ClOnPath}(1,:)];
                  Data.PathHandles{en}(2) = Data.WayHandles{en};
                  Data.WayHandles{en} = [];
```

```
        Data.R = 'R';  % Indicates "return home" path
        set(MPfig,'UserData',Data)
        sextant('Mission','PATH',en)  % Call to 'PATH' option
        %sextant('Mission','Points','R')
        Data = get(MPfig,'UserData');
        Data.R = '';
    end
                                % Waypoint edit mode
elseif get(Data.MPwpmodeH,'Value') && ~strcmp(Task,'open')
    en = get(Data.MPexpnumH,'Value');  % Explorer #
    if ~isempty(Data.Pathpoints{en})   % Check if path exists
        MPfigH = uisuspend(MPfig);
        Choice = questdlg('Editing waypoints will clear the traverse path.',...
                 sprintf('Edit Explorer %d',en),'OK','Cancel','OK');
        uirestore(MPfigH)
        if strcmp(Choice,'Cancel'), return, end
        delete(Data.PathHandles{[en,en+Data.NumExp]})
        set(Data.WayHandles{en}(1),'SizeData',120,...
            'CData',Data.Ecolors{mod(en-1,10)+1})
        Data.Waypoints{en+Data.NumExp} = [];
        for vars = {'Pathpoints' 'PathHandles' 'Distance' 'MetCost' 'Time' 'Shadows' 'SublimatedWater' 'HeaterPower' 'BatteryCapacity' 'TSuit' 'StageSuitHeat'}
            [Data.(vars{1}){[en,en+Data.NumExp]}] = deal([]);
        end
        set([Data.MPpathenH,Data.MPdistNH,Data.MPcostNH,...
             Data.MPtimeH],'String','')
        set(MPfig,'UserData',Data)
    end
    Data.WayPEd = true;
    if strcmp(Task,'normal')  % Left-Click: add waypoint
        if Data.Obstacles(cly+1,clx+1) ||...   %Click on obs
          (~isempty(Data.Waypoints{en}) &&... %or prev waypt
           all(Data.Waypoints{en}(end,1:2)==[clx cly]))
           return
        end                  % Append new waypoint
        Data.Waypoints{en} = [Data.Waypoints{en}; clx cly,...
                              Data.Elevations(cly+1,clx+1)];
    else                     % Shift+Click: erase waypoint
        if isempty(Data.Waypoints{en}), return, end
        Data.Waypoints{en} = Data.Waypoints{en}(1:end-1,:);
    end
    delete(Data.WayHandles{en})      % Clear prev waypoints
    if ~isempty(Data.Waypoints{en})  % Plot waypoints in GUI
        hold on
        Data.WayHandles{en}(1) = scatter3(Data.MPaxes,...
            Data.Waypoints{en}(:,1),Data.Waypoints{en}(:,2),...
            Data.Waypoints{en}(:,3)+.05*Data.Eldiff,...
            120,Data.Ecolors{mod(en-1,10)+1},'filled',...
            'ButtonDownFcn',sprintf('sextant("Mission","Click",%d)',en));
        %txt = {'H',{'H';num2str((1:size(Data.Waypoints{en},1)-1).')}};
        txt = {'1',{num2str((1:size(Data.Waypoints{en},1)).')}};
        WPtxt = text(Data.Waypoints{en}(:,1),...
            Data.Waypoints{en}(:,2)-.007*Data.Rows,...
            Data.Waypoints{en}(:,3)+.07*Data.Eldiff,...
            txt{1+(size(Data.Waypoints{en},1)>1)},...
            'HorizontalAlignment','center',...
            'VerticalAlignment','bottom',...
            'Color',[1 1 1],'FontWeight','bold','HitTest','off');
        Data.WayHandles{en} = [Data.WayHandles{en}(1);WPtxt];
        hold off
        set(Data.MPwaytitleH,'String',sprintf('WP %d:',...
                             size(Data.Waypoints{en},1)))
    else
        set(Data.MPwaytitleH,'String','Start:')
        Data.WayHandles{en} = [];
    end
                                % Terrain edit mode
elseif get(Data.MPtermodeH,'Value')
    for en = 1:Data.NumExp
        haspaths = ~isempty(Data.Pathpoints{en});
        if haspaths
            MPfigH = uisuspend(MPfig);
            Choice = questdlg(['Editing the terrain will',...
                               'clear all traverse paths.'],...
                              'Edit Terrain','OK','Cancel','OK');
            uirestore(MPfigH)
            if strcmp(Choice,'Cancel'), return, end
            delete(Data.PathHandles{:})
            [Data.Pathpoints{:},Data.PathHandles{:},...
             Data.Distance{:},Data.MetCost{:},Data.Time{:},...
             Data.Waypoints{Data.NumExp+1:end}] = deal([]);
```

```
                set([Data.MPpathenH,Data.MPdistNH,...
                    Data.MPcostNH,Data.MPtimeH],'String','')
                for i = 1:Data.NumExp
                   if ~isempty(Data.Waypoints{i})
                      set(Data.WayHandles{i}(1),'SizeData',120,...
                         'CData',Data.Ecolors{mod(i-1,10)+1})
                   end
                end
                break
             end
          end
          tmap = {'Obstacles' 'SoilMech' 'SciReturn' 'Other'};    % Create tmap
          Terrain = tmap{get(Data.MPtertypeH,'Value')};
          Data.(Terrain(1:4)) = true;         % Cost map exists
          Data.([Terrain(1:4),'Ed']) = true;  % Map edited, to be saved
          er = round(Data.Rows*Data.TEsize/200);
          ec = round(Data.Cols*Data.TEsize/200); % Edit rectangle
          [lr,ur,lc,uc] = deal(max(cly+1-er,1),min(cly+1+er,Data.Rows),...
                       max(clx+1-ec,1),min(clx+1+ec,Data.Cols));
          % Left-Click sets all values in the edit rectangle to 1
          % Double Click sets all values to 2 (besides Obstacles)
          % Shift+Click sets all values to zero
          Data.(Terrain)(lr:ur,lc:uc) = (strcmp(Task,'normal') + ...
             (1+~strcmp(Terrain,'Obstacles'))*strcmp(Task,'open'));
          if strcmp(Terrain,'Obstacles')
             Data.ColorObsRed = min(Data.Elevations+Data.Obstacles*10^6,...
                           Data.Elmin+Data.Eldiff*64/63);
             Terrain = 'ColorObsRed';
          end
          set(Data.MPsurf,'CData',Data.(Terrain))
       end

    case 'alt'                          % Right-Click: data display
       dtext = '';  info = [];  en = [];  wp = [];
       distU = get(Data.MPscaleH,'Value');
       distR = {1,'%.0f','m'; .001,'%.2f','km';...
             3.28084,'%.0f','ft'; .0006213712,'%.2f','mi'};
       costU = get(Data.MPcostUH,'Value');
       costR = {.2521644,'Kcal'; 1,'BTU'; 1.055056,'KJ'};
       if ~isempty(ClOnPath)    % If a path was clicked on
          en = ClOnPath(1);    % explorer# or #+Data.NumExp for return
          enR = mod(en-1,Data.NumExp)+1; % explorer#
          if en~=enR  % Return home path clicked on
             Data.R = 'R';
             set(MPfig,'UserData',Data)
          end
          set(Data.MPexpnumH,'Value',enR)     % Set overhead path
          sextant('Mission','SelExp',[])   % cost displays
          Data.R = '';
          wp = ClOnPath(end);  % waypoint # if passed
          if numel(ClOnPath)==1  % If no waypt passed, find nearest
             [D,wp] = min((Data.Waypoints{en}(:,1)-clx).^2+... %wp=Nearest
                      (Data.Waypoints{en}(:,2)-cly).^2);   %  waypoint
          end
          clx = Data.Waypoints{en}(wp,1);  % Move clx,cly to the
          cly = Data.Waypoints{en}(wp,2);  % waypoint coordinates
          if all([clx,cly]==Data.prevwp)
             Data.datadisp = mod(Data.datadisp,5)+1;
          end
          Data.prevwp = [clx,cly];
          numwp = size(Data.Waypoints{en},1);
          if ~isempty(Data.Pathpoints{en})
             pp = find((Data.Pathpoints{en}(:,1)==clx) & ... %pp=Point along
                   (Data.Pathpoints{en}(:,2)==cly),1);   %  traverse path
             if isempty(pp), pp=1; end
          else
             Data.datadisp = 5;
          end
          txt = {'Start Point:',sprintf('Waypoint %d:',wp-1)};
          dtext = sprintf([txt{1+(wp>1)},'\n']);
          while Data.datadisp<=4  % Display cost data
             hasdata = true;
             if Data.datadisp==1 && wp>1      % Cost from start
                header = 'Cost from start';
                dist = Data.Distance{en}(pp);
                mcost = Data.MetCost{en}(pp);
                time = Data.Time{en}(pp);
             elseif Data.datadisp==2 && wp>1 && numwp>2
                header = 'Cost from prev WP'; % Cost from prev waypt
                prevwp = Data.Waypoints{en}(wp-1,1:2);
```

```
                prevpp = find((Data.Pathpoints{en}(:,1)==prevwp(1))&...
                        (Data.Pathpoints{en}(:,2)==prevwp(2)),1);
                if isempty(prevpp), prevpp=1; end
                dist = Data.Distance{en}(pp)-Data.Distance{en}(prevpp);
                mcost = Data.MetCost{en}(pp)-Data.MetCost{en}(prevpp);
                time = Data.Time{en}(pp)-Data.Time{en}(prevpp);
             elseif Data.datadisp==3 && wp<numwp && numwp>2
                header = 'Cost to next WP';  % Cost to next waypoint
                nextwp = Data.Waypoints{en}(wp+1,1:2);
                nextpp = find((Data.Pathpoints{en}(:,1)==nextwp(1))&...
                        (Data.Pathpoints{en}(:,2)==nextwp(2)),1);
                dist = Data.Distance{en}(nextpp)-Data.Distance{en}(pp);
                mcost = Data.MetCost{en}(nextpp)-Data.MetCost{en}(pp);
                time = Data.Time{en}(nextpp)-Data.Time{en}(pp);
             elseif Data.datadisp==4 && wp<size(Data.Waypoints{en},1)
                header = 'Cost to end';      %Cost to end
                dist = Data.Distance{en}(end)-Data.Distance{en}(pp);
                mcost = Data.MetCost{en}(end)-Data.MetCost{en}(pp);
                time = Data.Time{en}(end)-Data.Time{en}(pp);
             else
                Data.datadisp = Data.datadisp+1;
                hasdata = false;
             end
             if hasdata
                dtext = [dtext,header,...
                      '\nDist:  ',distR{distU,2},' ',distR{distU,3},...
                      '\nCost: %.1f',' ',costR{costU,2},...
                      '\nTime: %d:%d%d']; %#ok<AGROW>
                hrmin = [floor(time/3600) round(rem(time,3600)/60)];
                info = [dist*distR{distU,1}, mcost*costR{costU,1},...
                     hrmin(1), zeros(hrmin(2)<10), hrmin(2)];
                break
             end
          end
       end
       if isempty(ClOnPath) || Data.datadisp == 5  % Display general info
          elu = {'m', 'ft', 1, 3.28084};
          dtext = [dtext,'Elev:   %.2f',elu{round(distU/2)},...
                    '\nSlope: %.2f°']; %#ok<AGROW>
          info = [Data.Elevations(cly+1,clx+1)*elu{2+round(distU/2)},...
                Data.Slopes(cly+1,clx+1)];
          % Changed to display UTM coordinates instead of Lat/Long for FMARS
          if Data.UTMzone~=0
             xLoc = (clx.*Data.Resolution) + Data.xllcorner;
             yLoc = ((Data.Rows - cly).*Data.Resolution) + Data.yllcorner;
%            % Lat/Long: uwgb.edu/dutchs/UsefulData/UTMFormulas.htm
%            x  = Data.xllcorner+(clx+.5)*Data.Resolution-500000;
%            y  = Data.yllcorner+(Data.Rows-1-(cly+.5))*Data.Resolution-...
%                 10000000*(Data.UTMzone<0);
%            e  = (1-6356752.314^2/6378137^2)^(1/2);
%            mu = y/(.9996*6378137*(1-e^2/4-3/64*e^4-5/256*e^6));
%            e1 = (1-(1-e^2)^(1/2))/(1+(1-e^2)^(1/2));
%            J  = [3/2*e1-27/32*e1^3, 21/16*e1^2-55/32*e1^4,...
%                 151/96*e1^3,       1097/512*e1^4];
%            fp = mu+J(1)*sin(2*mu)+J(2)*sin(4*mu)+J(3)*sin(6*mu)+J(4)*sin(8*mu);
%            e2 = e^2/(1-e^2);
%            C  = e2*cos(fp)^2;
%            T  = tan(fp)^2;
%            R  = 6378137*(1-e^2)/(1-e^2*sin(fp)^2)^(3/2);
%            N  = 6378137/(1-e^2*sin(fp)^2)^(1/2);
%            D  = x/(.9996*N);
%            Qa = [N*tan(fp)/R, D^2/2, (5+3*T+10*C-4*C^2-9*e2)*D^4/24,...
%                 (61+90*T+298*C+45*T^2-3*C^2-252*e2)*D^6/720];
%            Qo = [D, (1+2*T+C)*D^3/6, (5-2*C+28*T-3*C^2+8*e2+24*T^2)*D^5/120];
%            lat  = (fp-Qa(1)*(Qa(2)-Qa(3)+Qa(4)))*180/pi;
%            long = abs(Data.UTMzone)*6-183+((Qo(1)-Qo(2)+Qo(3))/cos(fp))*180/pi;
%            LAT  = [fix(lat)  fix(rem(lat,1)*60)  rem(rem(lat,1)*60,1)*60];
%            LONG = [fix(long) fix(rem(long,1)*60) rem(rem(long,1)*60,1)*60];
          % ******************************************************
          dtext = [dtext,'\nX Coord: %.4f',...
                    '\nY Coord: %.4f'];
          info = [info,xLoc,yLoc];
          dtext = [dtext,'\nUTM Zone:',num2str(Data.UTMzone)];
          end
          for tmap = {'SoilMech' 'SciReturn' 'Other'}
             if Data.(tmap{1}(1:4))
                dtext = [dtext,'\n',tmap{1}(1:5),':  %d']; %#ok<AGROW>
                info = [info,Data.(tmap{1})(cly+1,clx+1)]; %#ok<AGROW>
             end
          end
```

```
            end
            aln = {'left','right','bottom','top'};   % Text alignments
            hold on
            Data.minfo(1) = scatter3(Data.MPaxes,clx,cly,...
                 Data.Elevations(cly+1,clx+1)+.05*Data.Eldiff,60,[0 1 0],'filled');
            Data.minfo(2) = text(clx,cly,...
               Data.Elevations(cly+1,clx+1)+.2*Data.Eldiff,...
               sprintf(dtext,info),...
               'BackgroundColor',[.92 .865 .7],...
               'HorizontalAlignment',aln{1+(clx>.8*Data.Cols)},...
               'VerticalAlignment',aln{3+(cly<.1*Data.Rows)},...
               'ButtonDownFcn',sprintf('sextant("Mission","Click",[%d %d])',en,wp));
            hold off
      end
      set(MPfig,'UserData',Data)

   case 'Resize'  % Resize GUI window
      MPfig = gcf;
      Data = get(MPfig,'UserData');
      figsize = get(MPfig,'Position');
      set(Data.MPmenu,'Position',[0 figsize(4)-64 figsize(3)+2 66]);
      set(Data.MPaxes,'Position',[50 24 figsize(3)-65 figsize(4)-90]);

   case 'Close'  % Close GUI, save edits, exit SEXTANT
      MPfig = gcf;
      Data = get(MPfig,'UserData');
      if any([Data.ObstEd,Data.SoilEd,Data.SciREd,Data.OtheEd,Data.WayPEd])
         Choice = questdlg(sprintf([Data.EVAname,' has been edited\n\n',...
                  'Exit without running PATH?']),'Exit SEXTANT...',...
                  'Save edits','Don''t save','Cancel','Cancel');
      else
         Choice = questdlg(['Finished with ',Data.EVAname,'?'],...
                     'Exit SEXTANT...','Yes','Cancel','Cancel');
      end
      if ~strcmp(Choice,'Cancel')
         delete(MPfig)
         if strcmp(Choice,'Save edits')
            try save([Data.EVAname,'_Data'],'-struct','Data','Waypoints','-append'), catch end %#ok<CTCH>
            Data.Path = true;
            sextant('SaveMaps','-append',Data)      % Call to 'SaveMaps'
         end
      end
end

%% ************ UPDATE DATA & FILES & MISSION GUI *************************
case 'Update'       % Runs after callback to Map Info or EVA Data GUI
if ~isempty(Select)
message = '';
header = ['Changing these data values may clear\n',...
         'or re-write the following data:\n\n'];
for task = {{'ClrPaths','All traverse paths'} {'Obs','Obstacles'} ...
          {'NewMaps','Map data files'}}
   if any(strcmp(task{1}{1},Select))
      message = [header,message,'- ',task{1}{2},'\n']; %#ok<AGROW>
      header = '';
   end
end
AddEx = sum(strcmp('AddExp',Select));  % Number of added explorers
NumExp = Data.NumExp-AddEx;           % Previous number of explorers
if ~any(strcmp('ClrPaths',Select))
   ClrPath = [];
   for task = Select
      if ~ischar(task{1}) && task{1}<=NumExp && ...
        ~isempty(Data.Pathpoints{task{1}}) && ~any(ClrPath==task{1})
         ClrPath = [ClrPath, task{1}]; %#ok<AGROW>
      end
   end
   if ~isempty(ClrPath)
      message = [header,message,'- Traverse path(s): ',...
            num2str(sort(ClrPath),' %d'),'\n'];
   end
else
   ClrPath = 1:NumExp;
end
if any(strcmp('NewFiles',Select))
   message = [message,'\nOnly paths created after this point will be\n',...
                'written with the new name and/or directories.'];
end
if ~isempty(message)          % Asks if it's OK to make applicable changes
   Choice = questdlg(sprintf(message),'Change Data Values...','OK','Cancel','OK');
```

```
    if strcmp(Choice,'Cancel')  % Cancel without saving changes
       set(Data.MPfig,'Visible','on')
       return
    end
end
if any(strcmp('Scale',Select))            % Rescale map
    [gx,gy] = gradient(Data.Elevations,Data.Resolution);
    Data.Slopes = atan(sqrt(gx.^2+gy.^2))*(180/pi);  % Slopes in degrees
    mapsize = Data.Resolution*max(Data.Rows,Data.Cols);
    zaspect = Data.Resolution*min(1,10*Data.Eldiff/mapsize);
    set(Data.MPaxes,'DataAspectRatio',[1 1 zaspect])
    set(Data.MPfig,'UserData',Data)
    sextant('Mission','Scale',[])  % Call scaling routine
end
if ~Data.Obst && any(strcmp('Obs',Select)) %Recalculate obstacles
    Data.Obstacles = Data.Slopes > Data.MaxSlope; % 1 if obstacle, else 0
    Data.Obst = true;
    Data.ColorObsRed = min(Data.Elevations+Data.Obstacles*10^6,...
                       Data.Elmin+Data.Eldiff*64/63);
    if get(Data.MPtertypeH,'Value')==1
       set(Data.MPsurf,'CData',Data.ColorObsRed)
    end
end
if any(strcmp('NewMaps',Select)) || any(strcmp('NewFiles',Select)) %New map files
    sextant('SaveMaps','',Data)         % Call to 'SaveMaps'
    set(Data.MPfig,'Name',['SEXTANT:  ',Data.EVAname,' - SEXTANT'])
end
for en = ClrPath                          % Clear paths
    delete(Data.PathHandles{[en,en+NumExp]})
    [Data.Pathpoints{[en,en+NumExp]},Data.PathHandles{[en,en+NumExp]},...
     Data.Distance{[en,en+NumExp]},Data.MetCost{[en,en+NumExp]},...
     Data.Time{[en,en+NumExp]},Data.Waypoints{en+NumExp}] = deal([]);
    if ~isempty(Data.Waypoints{en})
       set(Data.WayHandles{en}(1),'SizeData',120,'CData',Data.Ecolors{mod(en-1,10)+1})
    end
end
if ~isempty(ClrPath)
    try save([Data.EVAname,'_Data'],'-struct','Data','Pathpoints','Distance',...
                                  'MetCost','Time','-append'), catch %#ok<CTCH>
    end
end
if any(ClrPath==get(Data.MPexpnumH,'Value'))  % Clear cost displays
    set([Data.MPpathenH,Data.MPdistNH,Data.MPcostNH,Data.MPtimeH],'String','')
end
if AddEx                                  % Add explorer
    [newexp{1:AddEx,1:2}] = deal([]);
    for vars = {'Waypoints' 'WayHandles' 'Pathpoints' 'PathHandles',...
              'Distance' 'MetCost' 'Time'}
       Data.(vars{1}) = vertcat(Data.(vars{1}),newexp);
    end
    set(Data.MPexpnumH,'String',1:Data.NumExp)
    try save([Data.EVAname,'_Data'],'-struct','Data','Explorers','-append'), catch end %#ok<CTCH>
end
if any(strcmp('Planet',Select))           % Change planet
    sextant('Mission',Data.Planet,[])
end
if isempty(Data.Lite) && any(strcmp('Sun',Select))  % Move sun
    set(Data.Sun,'Position',[sin((Data.Hour+Data.Minute/60)*pi/12),...
              -cos((Data.Hour+Data.Minute/60)*pi/12),...
              .014+.006*sin((Data.Hour+Data.Minute/60)*pi/24)])
end
set(Data.MPfig,'UserData',Data)           % Save Data changes
end
set(Data.MPfig,'Visible','on')            % Re-open Mission GUI

%% ************ RUN THE PATH OPTIMIZATION *********************************
case 'PATH'
    tic
enR = mod(Select-1,Data.NumExp)+1;  % enR is explorer# even for return home
message = ['Running traverse optimization...\n',...
          '\nTraverse path%s: ',num2str(enR,' %d,')];
s = {'','s'};
pathmsg = helpdlg([sprintf(message(1:end-1),s{1+(length(Select)>1)}),Data.R],'SEXTANT');
                             % Make sparse copies of terrain cost maps
[obstacles,soilmech,scireturn,other] = deal(sparse(Data.Obstacles),...
    sparse(Data.SoilMech),sparse(Data.SciReturn),sparse(Data.Other)); %#ok<NASGU>
elevation = Data.Elevations;
resolution = Data.Resolution;
obstacles(isnan(obstacles)) = 1;  % Clean up any NaNs in the maps
[elevation(isnan(elevation)),soilmech(isnan(soilmech)),...
```

Appendices

```
   scireturn(isnan(scireturn)),other(isnan(other))] = deal(0); %#ok<NASGU>
grav = [1,1/6,1/3];
gravity = 9.8*grav(Data.Planet);  % Set the planet gravity
[dimr,dimc] = deal(Data.Rows,Data.Cols);
E = sparse([5,ones(1,dimc-2),6;4*ones(dimr-2,1),zeros(dimr-2,dimc-2),2*ones(dimr-2,1);8,3*ones(1,dimc-2),7]);
path_err = false;  % Signals an error in finding traverse paths
Data.errpath = [];
for en = Select
   enR = mod(en-1,Data.NumExp)+1;
   Data.CostWeighting{enR} = 'MetCost';
   mass = Data.Weight(enR);
   waypoints_x_y = Data.Waypoints{en}(:,1:2)+1; %(x,y) coords to matrix index
   waypoints = Data.Waypoints{en}(:,1)*dimr+Data.Waypoints{en}(:,2)+1;
   % /////// Optimization Routine: Brandon Johnson, August 5, 2008 ///////
   [Distance,Cost,Time] = deal(0);
   smoothed_route = waypoints(1);
   [skipped,skip_waypoint,straight_line] = deal(false);
%    altwaypoints = ones(1,length(waypoints));
%    altwaypoints(2:2:end) = 0;  % Used to alternate which matrices are for start or finish
   % Waypoint Loop
   for k = 1:(length(waypoints)-1)
      start = waypoints(k);
      finish = waypoints(k+1);
      I = start;  J = finish;
      previousa = zeros(dimr,dimc);
      testcosta = sparse(dimr,dimc);
      Fcosta = sparse(dimr,dimc);
      costa = zeros(dimr,dimc);
      MetCosta = zeros(dimr,dimc);    % AWJ
      timea = zeros(dimr,dimc);
      distancea = zeros(dimr,dimc);
      costa(start) = 0.00001;
      % Main Loop
      while I ~= J
      % Determines neighboring nodes of I
         x = ceil(I/dimr);
         y = mod(I-1,dimr)+1;  % Find (x,y) coorinates of I
         if E(I) == 0  % Middle
            II = [I-1 I+dimr I+1 I-dimr I+dimr-1 I+dimr+1 I-dimr+1 I-dimr-1]; %all
            IIs = [x y-1; x+1 y; x y+1; x-1 y; x+1 y-1; x+1 y+1; x-1 y+1; x-1 y-1];
         elseif E(I) == 1  % Top edge
            II = [I+dimr I+1 I-dimr I+dimr+1 I-dimr+1]; %right down left bottomright bottomleft
            IIs = [x+1 y; x y+1; x-1 y; x+1 y+1; x-1 y+1];
         elseif E(I) == 2  % Right edge
            II = [I-1 I+1 I-dimr I-dimr+1 I-dimr-1]; %up down left bottomleft topleft
            IIs = [x y-1; x y+1; x-1 y; x-1 y+1; x-1 y-1];
         elseif E(I) == 3  % Bottom edge
            II = [I-1 I+dimr I-dimr I+dimr-1 I-dimr-1]; %up right left topright topleft
            IIs = [x y-1; x+1 y; x-1 y; x+1 y-1; x-1 y-1];
         elseif E(I) == 4  % Left edge
            II = [I-1 I+dimr I+1 I+dimr-1 I+dimr+1]; %up right down topright bottomright
            IIs = [x y-1; x+1 y; x y+1; x+1 y-1; x+1 y+1];
         elseif E(I) == 5  % Top left corner
            II = [I+dimr I+1 I+dimr+1]; %right down bottomright
            IIs = [x+1 y; x y+1; x+1 y+1];
         elseif E(I) == 6  % Top right corner
            II = [I+1 I-dimr I-dimr+1]; %down left bottomleft
            IIs = [x y+1; x-1 y; x-1 y+1];
         elseif E(I) == 7  % Bottom right corner
            II = [I-1 I-dimr I-dimr-1]; %up left topleft
            IIs = [x y-1; x-1 y; x-1 y-1];
         else  % Bottom left corner
            II = [I-1 I+dimr I+dimr-1]; %up right topright
            IIs = [x y-1; x+1 y; x+1 y-1];
         end
         remov_val = ~(~costa(II) & ~obstacles(II));
         II(remov_val) = [];
         IIs(remov_val,:) = [];
      % ******************** COST FUNCTION: METABOLIC ********************
         % Calculate the local costs for I using the Metabolic Cost function
         % Distance from I to II
         diag = (II==I-dimr-1 | II==I+dimr-1 | II==I+dimr+1 | II==I-dimr+1); %diagonals
         cart = (II==I-1 | II==I+dimr | II==I+1 | II==I-dimr); %up, right, down, left
         dist = zeros(1,length(II));
         dist(diag) = resolution*sqrt(2); % Diagonal dist is sqrt(2) greater
         dist(cart) = resolution;
         % Slope from I to II
         slope = 180/pi*atan((elevation(II)-elevation(I))./dist);
         % Velocity as a function of slope
         if strcmp(Data.Explorers,'Rover')   % For transportation rovers
```

```
    V = 4.17*ones(1,length(II));
    % Power
    power = zeros(1,length(slope));
    f = (slope<0);
       power(f) = 0.060*3.6*V(f)*mass + -0.3.*0.0073.*3.6.*mass.*slope(f).*(grav(Data.Planet)).*V(f);
    g = (slope>0);
       power(g) = 0.060*3.6*V(g)*mass + 0.0073.*3.6.*mass.*slope(g).*(grav(Data.Planet)).*V(g);
    h = (slope==0);
       power(h) = 0.060*3.6*V(h)*mass;
    power = power + Data.electronicsPower;
    currentMetCost = power.*dist./V; % Joules

    C = currentMetCost;
 else                              % For astronauts
    V = zeros(1,length(II));
    a = (slope<=-20 | slope>=15);
      V(a) = 0.05;
    b = (slope>-20 & slope<=-10);
      V(b) = 0.095*slope(b)+1.95;
    c = (slope>-10 & slope<0);
      V(c) = 0.06*slope(c)+1.6;
    d = (slope>=0 & slope<6);
      V(d) = -0.2*slope(d)+1.6;
    e = (slope>=6 & slope<15);
      V(e) = -0.039*slope(e)+0.634;
    % Power
    Rfactor = 0.661*V.*cos(pi*slope/180)+0.115;
    power = zeros(1,length(Rfactor));
    f = (slope<0);
      power(f) = (2.4*mass*gravity*V(f).*sin(pi*slope(f)/180).*(0.3.^(abs(slope(f))/7.65)))+((3.28*mass+71.1)*Rfactor(f));
    g = (slope>0);
      power(g) = (3.5*mass*gravity*V(g).*sin(pi*slope(g)/180))+((3.28*mass+71.1)*Rfactor(g));
    h = (slope==0);
      power(h) = (3.28*mass+71.1)*Rfactor(h);
    % Metabolic Cost
    currentMetCost = power.*dist./V;
    currentMetCost = currentMetCost*0.00094781712;  % Metcost converted from J to BTUs

    if strcmp(Data.CostWeighting{enR},'MetCost')
       C = currentMetCost;
    elseif strcmp(Data.CostWeighting{enR},'Dist')
       C = dist;
    elseif strcmp(Data.CostWeighting{enR},'Time')
       C = dist./V;
    else
       disp('Oops')
    end
 end

 % Heuristic Function for A* Algorithm
 % P. Amit, see http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
 H_straight = zeros(1,length(C));
 H_diagonal = zeros(1,length(C));
 for i = 1:size(IIs,1)
    xx = abs(IIs(i,1)-waypoints_x_y(k+1,1));
    yy = abs(IIs(i,2)-waypoints_x_y(k+1,2));
    H_straight(i) = xx+yy;
    if xx > yy
       H_diagonal(i) = yy;
    else
       H_diagonal(i) = xx;
    end
 end

 if strcmp(Data.Explorers,'Astronaut')
    if strcmp(Data.CostWeighting{enR},'MetCost')
       if Data.Planet == 1     % Earth
          hCost = resolution*(1.504*mass+53.298)*0.00094781712;   % in BTUs
       else                    % Moon
          hCost = resolution*(2.295*mass+52.936)*0.00094781712;   % in BTUs
       end
    elseif strcmp(Data.CostWeighting{enR},'Dist')
       hCost = resolution;
    elseif strcmp(Data.CostWeighting{enR},'Time')
       hCost = resolution/1.6;
    else
       disp('Oops')
    end
 else
    if strcmp(Data.CostWeighting{enR},'MetCost')
```

```
            hCost = resolution*(.216*mass+Data.electronicsPower/4.167)*0.00094781712;    % in BTUs
         elseif strcmp(Data.CostWeighting{enR},'Dist')
            hCost = resolution;
         elseif strcmp(Data.CostWeighting{enR},'Time')
            hCost = resolution/4.167;
         else
            disp('Oops')
         end
      end
      H = hCost*(H_straight-2*H_diagonal)...
         +(sqrt(2)*hCost)*H_diagonal;
      D = (~testcosta(II) | costa(I)+C<testcosta(II));

      % Store the costs
      testcosta(II(D)) = costa(I)+C(D);  % Total metabolic cost
      Fcosta(II(D)) = costa(I)+C(D)+H(D);  % Metabolic + Heuristic estimate
      MetCosta(II(D)) = MetCosta(I)+currentMetCost(D);     % AWJ
      timea(II(D)) = timea(I)+dist(D)./V(D);  % Total time
      distancea(II(D)) = distancea(I)+dist(D);  % Total distance
      previousa(II(D)) = I;  % Used to back-track for finding the route
   % ********************** END OF COST FUNCTION **********************
      testcosta(I) = 0;
      Fcosta(I) = 0;
   % Find minimum value in both Fcosts
      K = find(Fcosta);
      [v,N] = min(Fcosta(K));
      I = K(N);   % Move to the point of minimum cost
   % Update costs & check if paths intersect
      costa(I) = testcosta(I);
   end  % End of main loop
   j = I;
   % Traverse path is solved!  Now find the route
   if ~exist('j','var'), path_err = true; break, end  % Signals error, break
   route = [j,zeros(1,dimr+dimc)];
   count = 1;
   i = previousa(j);
   while i ~= 0
      count = count+1;
      route(count) = i;
      i = previousa(i);
   end
   route1 = route(find(route,1,'last'):-1:1);
   route = [j,zeros(1,dimr+dimc)];
   route2 = route(1:find(route,1,'last'));
   testline = Midpoint(route1(1),route2(end)); %Tests if route is straight to simplify smoothing
   if length(testline)==length([route1(2:end-1) route2]) && all(testline == [route1(2:end-1) route2])
      new_route = route2(end);
      straight_line = 1;
   else
      new_route = Smooth([route1(1:end-1) route2]); % Call smoothing function
      new_route = new_route(2:end);
   end
   smoothed_route = [smoothed_route,new_route]; %#ok<AGROW>
   Update_lists  % Update stored costs
   clear j
end
% ////////////////// END OF OPTIMIZATION ROUTINE ////////////////////
if path_err  % Handle any path errors
   Data.errpath = [Data.errpath,enR];  %#ok<AGROW>
   Select = Select(Select~=en);  % Purge from new path list
   path_err = false;
   continue
end
Data.Pathpoints{en} = [floor((smoothed_route-.5)/dimr); mod(smoothed_route-1,dimr)].';
[Data.Distance{en},Data.MetCost{en},Data.Time{en}] = deal(Distance.',Cost.',Time.');
Data.stationaryTime{en} = zeros(size(Data.Waypoints{en},1),1);
end
try delete(pathmsg), catch end  %#ok<CTCH>
toc


%% ************ STORE PATHS, WRITE RENDER & COORD FILES ********************
for en = Select
   cd(Data.Render_dir)
   enR = mod(en-1,Data.NumExp)+1;
   for Outfile = {Data.EVAname,'Current'}    % Write Waypoint render files
      wrf = fopen(sprintf([Outfile{1},'_Waypoints%d%s.txt'],enR,Data.R),'wt');
      fprintf(wrf,'way%d %d %d\n',[(1:size(Data.Waypoints{en},1)).',...
         Data.Waypoints{en}(:,1),Data.Rows-1-Data.Waypoints{en}(:,2)].');
      fclose(wrf);                        % Write Traverse render files
```

```
    trf = fopen(sprintf([Outfile{1},'_Traverse%d%s.txt'],enR,Data.R),'wt');
    fprintf(trf,'path%d %d %d\n',[(1:length(Data.Distance{en})).',...
       Data.Pathpoints{en}(:,1),Data.Rows-1-Data.Pathpoints{en}(:,2)].');
    fclose(trf);                        % Write Cost render files
    crf = fopen(sprintf([Outfile{1},'_Costs%d%s.txt'],enR,Data.R),'wt');
    fprintf(crf,'cost%d %.2f %.2f %.2f\n',[(1:length(Data.Distance{en})).',...
       Data.Distance{en},Data.Time{en},Data.MetCost{en}].');
    fclose(crf);
  end
  for i = 1:length(Data.Distance{en}); % Append elevations at path coords
    Data.Pathpoints{en}(i,3) = Data.Elevations(Data.Pathpoints{en}(i,2)+1,...
                                  Data.Pathpoints{en}(i,1)+1);
  end

  % The following is an example of exporting traverse Lat/Long data in text files
  % This may be deleted if not desired
  if Data.UTMzone ~= 0
    [LAT,LONG] = deal(zeros(i,3));
    for k = 1:i  % Lat/Long at each Pathpoint
      % Lat/Long: uwgb.edu/dutchs/UsefulData/UTMFormulas.htm
      x  = Data.xllcorner+(Data.Pathpoints{en}(k,1)+.5)*Data.Resolution-500000;
      y  = Data.yllcorner+(Data.Rows-1-(Data.Pathpoints{en}(k,2)+.5))*...
                  Data.Resolution-10000000*(Data.UTMzone<0);
      e  = (1-6356752.314^2/6378137^2)^(1/2);
      mu = y/(.9996*6378137*(1-e^2/4-3/64*e^4-5/256*e^6));
      e1 = (1-(1-e^2)^(1/2))/(1+(1-e^2)^(1/2));
      J  = [3/2*e1-27/32*e1^3, 21/16*e1^2-55/32*e1^4,...
           151/96*e1^3,       1097/512*e1^4];
      fp = mu+J(1)*sin(2*mu)+J(2)*sin(4*mu)+J(3)*sin(6*mu)+J(4)*sin(8*mu);
      e2 = e^2/(1-e^2);
      C  = e2*cos(fp)^2;
      T  = tan(fp)^2;
      R  = 6378137*(1-e^2)/(1-e^2*sin(fp)^2)^(3/2);
      N  = 6378137/(1-e^2*sin(fp)^2)^(1/2);
      D  = x/(.9996*N);
      Qa = [N*tan(fp)/R, D^2/2, (5+3*T+10*C-4*C^2-9*e2)*D^4/24,...
           (61+90*T+298*C+45*T^2-3*C^2-252*e2)*D^6/720];
      Qo = [D, (1+2*T+C)*D^3/6, (5-2*C+28*T-3*C^2+8*e2+24*T^2)*D^5/120];
      lat  = (fp-Qa(1)*(Qa(2)-Qa(3)+Qa(4)))*180/pi;
      long = abs(Data.UTMzone)*6-183+((Qo(1)-Qo(2)+Qo(3))/cos(fp))*180/pi;
      LAT(k,:)  = [fix(lat)  fix(rem(lat,1)*60)  rem(rem(lat,1)*60,1)*60];
      LONG(k,:) = [fix(long) fix(rem(long,1)*60) rem(rem(long,1)*60,1)*60];
    end
    cd([Data.Work_dir,'Traverse_Coordinates'])
    pcf = fopen(sprintf([Data.EVAname,'_Coords%d%s.txt'],enR,Data.R),'wt');
    fprintf(pcf,'Explorer %d%s  Lat , Long:\n',enR,Data.R);
    fprintf(pcf,'point%d  %.0f %.0f %.2f , %.0f %.0f %.2f\n',...
      [(1:length(Data.Distance{en})).',LAT(:,1),abs(LAT(:,2:3)),LONG(:,1),abs(LONG(:,2:3))].');
    fclose(pcf);
  end

end
cd(Data.Work_dir)  % Append paths & costs to Matlab data file
try save([Data.EVAname,'_Data'],'-struct','Data','Pathpoints','Distance',...
   'MetCost','Time','-append'), catch  %#ok<CTCH>
end
Data.Newpaths = Select;          % Note all successful paths
set(Data.MPfig,'UserData',Data)   % Store all path and cost data

%% ************ INCORRECT 3 ARGUMENT CALL TO PATHMASTER ********************
otherwise
disp('Error: Incorrect call to pathmaster')

%% End of Progress switch
end

%% ************ PATH OPTIMIZATION SUBFUNCTIONS ****************************
% Update Cost, Distance, & Time lists
function Update_lists
  prev_cost = Cost(end);
  prev_distance = Distance(end);
  prev_time = Time(end);
  if ~straight_line
       for p = new_route
            Cost(end+1) = prev_cost+MetCosta(p); %#ok<AGROW>
            Distance(end+1) = prev_distance+distancea(p); %#ok<AGROW>
            Time(end+1) = prev_time+timea(p); %#ok<AGROW>
       end
  else
    Cost(end+1) = prev_cost+MetCosta(j);%+MetCostb(j);
```

```
        Distance(end+1) = prev_distance+distancea(j);%+distanceb(j);
        Time(end+1) = prev_time+timea(j);%+timeb(j);
    end
end

% Path Smoothing
function [smooth_route] = Smooth(route)
    smooth_route = route;
    if strcmp(Data.Simplifying,'SimpON')
        remove_val = zeros(1,length(smooth_route));
        remove_val(1) = 1; %find(remove_val,1,'first') wont give an empty matrix
        p = 1;
        while p < length(route)
            u = route(p);
            for ip = (2+p):length(route)
                v = route(ip);
                line1 = Midpoint(u,v);  % Calls Midpoint function
                if length(line1)==length(route(p:ip)) && all(line1==route(p:ip)) %If route(p:ip) is straight, remove points between
                    remove_val(find(remove_val,1,'last')+1) = ip-1;
                elseif v==u-dimr-1 || v==u+dimr-1 || v==u+dimr+1 || v==u-dimr+1
                    remove_val(find(remove_val,1,'last')+1) = ip-1;
                else
                    break;
                end
            end
            p = ip-1;
        end
        remove_val(1) = [];
        remove_val = remove_val(1:find(remove_val,1,'last'));
        smooth_route(remove_val) = [];
    end
end

% Midpoint Algorithm
% Modified from N. Chattrapiban's version of Bresenham's Algorithm
% Used to find the straightest path between two points
function [myline] = Midpoint(a,b)
    x = ceil([a b]/dimr);
    y = mod([a b]-1,dimr)+1;
    XX = x(1);
    YY = y(1);
    steep = (abs(y(2)-y(1)) > abs(x(2)-x(1)));
    if steep
        t = x; x = y; y = t;
    end
    if x(1) > x(2)
        t = x(1); x(1) = x(2); x(2) = t;
        t = y(1); y(1) = y(2); y(2) = t;
    end
    delx = x(2)-x(1);
    dely = abs(y(2)-y(1));
    err = 0;
    x_n = x(1);
    y_n = y(1);
    if y(1) < y(2), ystep = 1; else ystep = -1; end
    myline = zeros(1,delx+1);
    for nn = 1:delx+1
        if steep
            myline(nn) = 1+dimr*(y_n-1)+(x_n-1);
        else
            myline(nn) = 1+dimr*(x_n-1)+(y_n-1);
        end
        x_n = x_n + 1;
        err = err + dely;
        if bitshift(err,1) >= delx, % same as -> if 2*err >= delx,
            y_n = y_n + ystep;
            err = err - delx;
        end
    end
    if myline(1) ~= (XX-1)*dimr+YY
        myline = myline(end:-1:1);
    end
end

% Functions for creating pathPoint xml file
% Aaron Johnson 8/25

% viewString adds to the string that composes the "views" portion of the
% Compendium xml file
function [retVS] = viewString(view, nodeID, parentID)
```

```
   dateString=strcat(num2str(Data.Hour),num2str(Data.Minute),num2str(Data.Month),num2str(Data.Day),num2str(Data.Year));
   retVS = strcat('          <view viewref="',num2str(parentID),'" noderef="',num2str(nodeID),'" XPosition="0" YPosition="0" created="0" lastModified="0"
showTags="true" showText="true" showTrans="true" showWeight="true" smallIcon="false" hideIcon="false" labelWrapWidth="25" fontsize="12"
fontface="Dialog" fontstyle="0" foreground="-16777216" background="-1">\n          </view>\n');
   retVS = strcat(view,retVS);
end

% nodeString adds to the string that composes the "nodes" portion of the
% Compendium xml file
function [retNS] = nodeString(node, nodeID, type, label, codeID)
   dateString=strcat(num2str(Data.Hour),num2str(Data.Minute),num2str(Data.Month),num2str(Data.Day),num2str(Data.Year));

   % If codeID == 0, then there is no code for this particular node
   if codeID~=0
      code = strcat('                  <coderef coderef="',num2str(codeID),'" />\n');
   else
      code = '';
   end

   retNS = strcat('          <node id="',num2str(nodeID),'" type="',num2str(type),'" extendedtype="" originalid="" author="SEXTANT" created="',...
      dateString,'" lastModified="',dateString,'" label="',label,'" state="2" lastModificationAuthor="SEXTANT">\n              <details>\n                  <page nodeid="',...
      num2str(nodeID),'" author="SEXTANT" created="',dateString,'" lastModified="',dateString,...
      '" pageno="1"></page>\n              </details>\n              <source></source>\n              <image width="0" height="0"></image>\n
<background></background>\n              <coderefs>\n',code,...
      '              </coderefs>\n              <shortcutrefs></shortcutrefs>\n          </node>\n');
   retNS = strcat(node, retNS);
end

% linkString adds to the string that composes the "links" portion of the
% Compendium xml file
function [retLS] = linkString(link, linkID, fromID, toID, linkviewID)
   % If the linkID is < 0, this means that the order of inputs to this
   % function is: (link, linkID, toID, fromID, linkviewID)
   if linkID < 0;
      linkID = -linkID;
      temp = fromID;
      fromID = toID;
      toID = temp;
   end

   dateString=strcat(num2str(Data.Hour),num2str(Data.Minute),num2str(Data.Month),num2str(Data.Day),num2str(Data.Year));
   retLS = strcat('          <link id="',num2str(linkID),'" created="',dateString,'" lastModified="',dateString,'" author="SEXTANT" type="45" originalid="" from="',...
      num2str(fromID),'" to="',num2str(toID),'" label="" arrow="1">\n              <linkviews>\n                  <linkview id="',...
      num2str(linkviewID),'"/>\n              </linkviews>\n          </link>\n\n');
   retLS = strcat(link, retLS);
end

% End of SEXTANT
end
```