A Dynamic Model of the Extravehicular Mobility Unit (EMU): Human Performance Issues During EVA

by

David B. Rahn

S.B. Aeronautics and Astronautics
Massachusetts Institute of Technology, 1995

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 1997

Signature of Author ....
Department of Aeronautics and Astronautics
May 23, 1997

Certified by ....
Professor Dava J. Newman
Thesis Supervisor

Accepted by ....
Professor Jaime Peraire
Chairman, Department Graduate Committee

A Dynamic Model of the Extravehicular Mobility Unit (EMU): Human Performance Issues During EVA

by
David B. Rahn

Submitted to the Department of Aeronautics and Astronautics
on May 23, 1997 in partial fulfillment of the
requirements for the Degree of Master of Science in
Aeronautics and Astronautics

Abstract

A dynamic model of the extravehicular mobility unit (EMU) space suit was developed and applied to the simulation of several extravehicular activity (EVA) tasks. The space suit model was created from mass, inertia, and torque data to augment the unsuited 12-segment human model used in previous studies. A modified Preisach model was introduced to describe the hysteretic torque characteristics of joints in a pressurized space suit, and implemented numerically based on observed suit parameters. Four computational simulations were then performed to observe the effect of suit constraints on simulated astronaut performance, involving manipulation of the Spartan astrophysics payload on STS-63. It was found that the shoulder joint work required for a suited EVA crewmember to move the Spartan payload while in an inefficient posture was a full order of magnitude greater than the unsuited condition. Moving to a posture more accommodating to the suit's neutral position, the simulated astronaut completed the task with an actual decrease in shoulder work from the unsuited condition; to compensate for reduced upper-body mobility, however, the ankle joint was forced to use its long lever arm to manipulate the payload, resulting in ankle work 207% greater than in the unsuited condition. In this situation, muscle fatigue would set in more quickly than expected, a problem for the long-duration EVA missions expected on the International Space Station. The Spartan results agree with anecdotal evidence of post-EVA ankle fatigue, and demonstrate the effectiveness of both the space suit model and the simulation technique.

Thesis Supervisor: Dava J. Newman
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

Funding and resources for this work were provided by NASA Grant NAGW-4336 and the MIT Man-Vehicle Lab.

I cannot begin this page without thanking Professor Dava Newman from the bottom of my heart for wonderful advice and friendship; I don't think I would have survived without your guidance and support over the last three years. You've seen my transition from a bright-eyed UROP to a more ... refined persona, watching in amazement, I'm sure. Our conversations have done more to shape my future than you might think! It has been a pleasure working with you, and I'm sure our paths will cross in the not-so-distant future.

This experience would have been something entirely less appealing without the crew of the Man-Vehicle Lab; Dawn, Chris, Jen, Keoki, Mikes T. And M., Patricia, Prashant, Matt, Adam, Scott, Albert, Anna, there are too many already and I have to be forgetting somebody. You helped make it a home worth coming to (although windows would have been nice....). And to Dr. Chuck Oman and Professor Larry Young for heading us all in the right direction with journal club and lab events, thank you for allowing me to join your group.

Many outside sources contributed to this thesis: from Grant, with his incomparable simulation code, to Bernadette Luna and a host of people at Ames for the data, all at JSC for their work (especially to IGOAL for your enigmatic software), with great things still to come. Cal Seaman, Cab Callaway, Bruce Webbon, Jeff Hoffman, Carlos Roithmayr, and everybody at NASA who lent a suggestion or two in our direction, thank you. The hectic pace of your schedules didn't stop you from taking the time to offer help and support. I wish you all the best.

Thanks of a different sort must go to my housemates, Minh, Eugene, and Jun, who managed not to evict me during my less-than-organized residence, and brought some social outings my way to avoid computer seclusion. Long-distance thanks to Charlie, Francesca, and Sue, who helped maintain my sanity through the process. I suppose at some point I should be giving them support in return.

Finally, to my family for years of support and love; Mom and Dad, you're the greatest. And Mary, wow, what can I say. You're just too cool. They always believed that I would return to California someday. They were mostly right.

Table of Contents

ABSTRACT 2

ACKNOWLEDGMENTS 3

TABLE OF CONTENTS 4

LIST OF FIGURES 6

LIST OF ACRONYMS 7

1. INTRODUCTION 8

1.1 Motivation 8

1.2 Objective and Contribution 9

1.3 Summary of Thesis 10

2. BACKGROUND 11

2.1 Extravehicular Activity (EVA) 12
2.1.1 Historical Perspective 12
2.1.2 Extravehicular Activity Simulation 14

2.2 The Space Suit 17
2.2.1 Environmental and Physiological Concerns 17
2.2.2 A History of the Extravehicular Mobility Unit (EMU) 20
2.2.2.1 Early Suit Designs 20
2.2.2.2 Mercury 22
2.2.2.3 Gemini 23
2.2.2.4 Apollo 25
2.2.2.5 Shuttle 26
2.2.3 The Constant Volume Challenge 27

2.3 Flexible Fabric Modeling 29
2.3.1 Finite Element Methods 30
2.3.2 Empirical Methods 33

2.4 Musculoskeletal Modeling 35

3. METHODS 39

3.1 EMU Space Suit Model 40
3.1.1 Inertial Properties 40
3.1.2 Suit-imposed Torques 41
3.1.3 Hysteresis Model 43
3.1.3.1 Introduction to Hysteresis 43
3.1.3.2 The Preisach Model 46

3.1.3.3 A Geometric Interpretation 47
3.1.3.4 The Preisach Function 52
3.1.3.5 Numerical Implementation of the Hysteresis Model 56

3.2 Human model **61**
3.2.1 Skeletal Model 61
3.2.2 Muscle Models 62
3.2.2.1 Torsional Equivalent 63
3.2.2.2 Lumped Muscle Models 64

3.3 Simulation Method **65**
3.3.1 SD/FAST Framework 66
3.3.2 Payload Translation Task 66
3.3.2.1 Rigid Lower Body 68
3.3.2.2 Compliant Lower Body 69

4. RESULTS **69**

4.1 Hysteresis Model Test **69**

4.2 Payload Translation Task **72**
4.2.1 Rigid Lower Body 72
4.2.2 Compliant Lower Body 78

5. DISCUSSION **85**

5.1 Hysteresis Model Test **85**

5.2 Payload Translation Task -- Rigid Lower Body **86**

5.3 Payload Translation Task -- Compliant Lower Body **87**

6. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK **91**

7. REFERENCES **96**

APPENDIX A: TRANSITION CURVE AND MESH CONSTRUCTION CODE **100**

APPENDIX B: TEST FUNCTION FOR F(A,B) MESH **107**

APPENDIX C: VISUALIZATION PROCESSING FOR MESH TEST **111**

APPENDIX D: ELBOW JOINT TEST **113**

APPENDIX E: SIMULATION CODE **118**

APPENDIX F: PROCESSING FOR SIMULATION RESULTS **149**

List of Figures

Figure 2.1 Historical and Future NASA EVA Time [NASA, 1996]. 16
Figure 2.2 Early Pressure Suits. 21
Figure 2.3 Four-layer construction of the Mercury space suit. 22
Figure 2.4 Gemini (left) and Apollo (right) space suits. 24
Figure 2.5 Space Shuttle EMU multi-layer soft shell construction. 27
Figure 2.6 Volume change during joint flexion. 28
Figure 2.7 Nearly constant-volume space suit joint. 28
Figure 2.8 Finite element representation of a swatch of cloth. 30
Figure 2.9 An empirically determined model of system $f(x,u,t)$. 34
Figure 2.10 The human elbow joint is a third class lever. 35
Figure 2.11 Three-segment model of a human arm. 36
Figure 2.12 Hill-type muscle model [Winters, 1995]. 37
Figure 3.1 Hysteresis nonlinearity observed at the elbow joint of the EMU. 42
Figure 3.2 Hysteresis transducer. 43
Figure 3.3 Input-output time history for a hysteresis transducer. 44
Figure 3.4 Hysteresis transducer with local memory. 45
Figure 3.5 Hysteresis transducer with nonlocal memory. 45
Figure 3.6 A simple hysteresis operator $\hat{\gamma}_{\alpha\beta}$. 46
Figure 3.7 Hysteresis decomposition into simplest hysteresis operators $\hat{\gamma}_{\alpha\beta}$. 47
Figure 3.8. The $\alpha \geq \beta$ half-plane. 48
Figure 3.9 Triangle T at time t_1. 48
Figure 3.10 Region $S^+(t_2)$, bounded by the lines $\alpha = u_1$ and $\beta = u_2$. 49
Figure 3.11 The staircase interface $I(t)$. 50
Figure 3.12 Angle versus torque curve for a space suit joint, with saturation limits. 51
Figure 3.13 EMU-modified half-plane $\alpha \leq \beta$. 52
Figure 3.14 First-order transition curves. 53
Figure 3.15 The f-u plane (left) and α–β plane (right) during a first-order reversal. 53
Figure 3.16 Subdivision of $S^+(t)$ into multiple trapezoids. 55
Figure 3.17 The discretized region T. 57
Figure 3.18 The angle $\theta(\alpha',\beta')$. 58
Figure 3.19 The complete set of first-order transition curves for the elbow joint. 59
Figure 3.20 Cells bounded by either three or four vertices. 60
Figure 3.21 Human skeletal model. 61
Figure 3.22 Torsional equivalent muscle models. 63
Figure 3.23 Agonist-antagonist lumped-muscle model. 64
Figure 3.24 Schematic representation of a Hill muscle model. 65
Figure 3.25 Simulation flow diagram. 66
Figure 3.26 Spartan payload translation task. 67
Figure 3.27 Bias position q_{bias} for the elbow joint. 68
Figure 4.1 Elbow joint trajectory and suit-imposed torque during model test. 70
Figure 4.2 The surface $f(\alpha,\beta)$ at $t = 60$ seconds (elbow joint test). 71
Figure 4.3 Sequential astronaut position during payload translation task, rigid lower body. 72
Figure 4.4 Joint positions during payload translation task, rigid lower body. 73
Figure 4.5 Joint torques for payload translation task, rigid lower body. 76
Figure 4.6 Cumulative joint work for payload translation task, rigid lower body. 77
Figure 4.7 Astronaut position during payload translation task, compliant lower body. 79
Figure 4.8 Joint positions for payload translation task, compliant lower body. 80
Figure 4.9 Joint torques for payload translation task, compliant lower body. 83
Figure 4.10 Cumulative joint work for payload translation task, compliant lower body. 84

List of Acronyms

d.o.f.	degree(s) of freedom
EMU	Extravehicular Mobility Unit
EVA	Extravehicular Activity
ILC	International Latex Corporation
JSC	Johnson Space Center
LCG	Liquid Cooling Garment
LCVG	Liquid Cooling and Ventilation Garment
LEO	Low Earth Orbit
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
PGA	Pressure Garment Assembly
PLSS	Portable (Primary) Life Support System
RMS	Remote Manipulator System
STS	Space Transportation System
TMG	Thermal and Micrometeoroid Protection Garment

1. Introduction

Chapter One addresses the specific motivation for developing a space suit dynamic model and the more general motivation behind the current research effort in extravehicular activity (EVA) computer simulation. It discusses the place of a space suit model in the overall simulation framework, and then details the specific objective and contribution of this thesis.

1.1 Motivation

The extravehicular mobility unit (EMU) space suit presents a mechanically complex, highly constrained working environment that limits astronaut motion considerably. As a result, numerous studies have attempted to characterize the particular workspace to which a suited astronaut is restricted [Reinhardt, 1989]; dimensions aid extravehicular activity planners in describing feasible EVA tasks, assuring that the reach envelope required for a given task will not exceed the reach envelope available to the astronaut. Computer-aided design facilities have already incorporated similar data into EVA planning software [Price, 1994; Pardue, 1996]. Workspace definitions, however, do not take into account the appreciable dynamic effects that a 146 kg space suit with significant joint torques will have on astronaut strength and mobility during an EVA task. Data is only beginning to appear with regard to available suited strength [West, 1989; Dionne, 1991; Morgan, 1996], and thus make its way into the EVA design process. Even if armed with this data, EVA designers will only be able to make rough estimates of strength needed vs. strength available for a given task. These static measurements of performance do not allow for the direct comparison of tasks that have been choreographed in a particular way, or of the astronaut effort required to execute each task in terms of total work performed, distribution of work among the various muscle groups, and issues involving the gradual onset of fatigue.

Computer simulation of EVA tasks provides one solution to these questions, allowing designers to observe how space suit effects may influence overall EVA performance. A dynamic space suit model can record astronaut movement through a simulated EVA task and apply appropriate inertial and suit-generated loads to crewmember limbs during the simulation.

The resulting time histories of position, velocity, and acceleration for various body segments then allow for quantitative analysis of the work being performed. Problem areas can be identified, and choreography refined to account for predicted complications.

This method also provides a six-degree-of-freedom arena in which the simulated EVA crewmember can work, allowing for unrestrained multibody interactions in a microgravity environment. Although physical simulators such as the Johnson Space Center's neutral buoyancy facility are excellent for both astronaut training and developing EVA choreography, these and other ground-based devices have inherent limitations; it has been shown that quantitative pre-flight analysis of EVA tasks in six degrees of freedom can help to reveal the sometimes quite subtle differences that accompany work in an EVA setting [Schaffner, 1997].

1.2 Objective and Contribution

The objective of this research effort is to accurately model and simulate astronauts during the execution of an EVA task, providing a six-degree-of-freedom environment in which to explore new EVA tools and techniques. This thesis specifically addresses the EMU space suit, investigating possible models for the suit, developing computer code for model implementation, and presenting results based on application of the space suit model to EVA task simulation.

Accurate modeling of the EMU space suit is essential to the realistic simulation of any EVA task. The astronaut is unavoidably forced to work within the confines of the suit, and it affects every movement he or she makes for hours on end. The EMU's heavy "backpack," the primary life support system (PLSS), generates unfamiliar inertial loads, the structure of the suit places hard limits on the available workspace, and, most importantly, the suit imposes restrictive torques on astronaut motion. These effects combine with the unfamiliarity of working in a microgravity environment to pose significant hazards to an EVA crewmember. The objectives of this thesis are therefore to create a model of the space suit, and to explore its applicability to EVA simulation.

To meet these goals, an original dynamic model of the EMU space suit is developed, based on data obtained from NASA and Hamilton Standard (Windsor Locks, CT). The Preisach model is introduced as a mathematical technique for modeling suit joint characteristics, and implemented in computer code on an SGI Indigo^2 workstation. Finally, the model is tested during three simulations of astronaut motion during EVA.

1.3 Summary of Thesis

This introductory chapter presents the motivation and objectives of this thesis, followed by chapters discussing the background, methods, results, discussion, and final conclusions of the research. Chapter Two introduces extravehicular activity as a key enabling technology for spaceflight, providing a brief history of past EVA missions and considering various options for the simulation and training of EVA operations. It then touches on the history of the EMU space suit, focusing on construction issues and critical technologies. A short summary of methods for flexible fabric modeling is included, followed by arguments for empirically based, data-driven models. Finally, several biomechanical models for describing the underlying human operator are presented, with an emphasis on properly choosing the level of detail appropriate for this research.

Chapter Three discusses the methods used to create a dynamic model of the EMU, representing the main contribution of this thesis. Basic inertial properties are examined, and a modified Preisach model for hysteresis is introduced to allow accurate representation of imposed space suit joint torques. The implementation of the hysteresis model is discussed in detail, and described in terms of existing space suit data. Musculoskeletal models for the astronaut are then addressed, considering several different lumped-parameter approaches. Finally, the multibody dynamic simulation methodology is described along with several simulations created to test the EMU model.

Chapter Four presents the results of a test simulation for the hysteresis model at the EMU elbow joint, and two payload translation simulations involving the Spartan astrophysics payload. Astronaut position during each simulation is displayed in terms of individual joint angles, and sequential diagrams of crewmember position are included to facilitate interpretation of the kinematics data. Torques at each joint are calculated, and total cumulative work is computed from these torques and their corresponding joint velocities.

Chapter Five discusses the simulation results, highlighting interesting aspects of the data and comparing the results to previous research. The elbow joint test simulation is discussed first, followed by the rigid body payload translation and the compliant body payload translation tasks.

The EMU model is observed to have a dramatic effect on astronaut posture during simulated EVA tasks, and to increase the workload of a suited crewmember in unexpected ways. Finally, the work of this thesis is compared to existing research and similar investigations, highlighting possible opportunities for collaboration and technology transfer.

Chapter Six reviews the main contributions of this research, and discusses potential applications for the EMU model. It includes recommendations for experimental work that could be performed to help refine the model, and discusses opportunities for future research in the areas of space suit modeling and EVA simulation. A summary of this thesis is provided, emphasizing the gap in existing simulation technology filled by this research. Future work in EVA computer simulation based on this suit model will improve crewmember training and comfort, and will work with other simulation methods to provide an integrated environment for EVA design and analysis.

2. BACKGROUND

This chapter presents background information relevant to modeling the extravehicular mobility unit space suit. It begins by familiarizing the reader with extravehicular activity in general, providing a brief history of humans in space and then outlining ways in which computer simulation can improve EVA preparation and effectiveness. The text first highlights several milestones in EVA work throughout the space program, with emphasis on the importance of EVA as a key enabling technology for space travel. Various methods of EVA preparation and training are then discussed, indicating how computer simulation complements existing technologies and provides added insight for EVA missions.

The second section traces space suit development over seven decades, illustrating how early mobility problems were overcome by ingenuity in suit design and increased interest from the space program. It chronicles the various improvements made on successive EVA suits, which ultimately led to the design of the EMU in service today on the Space Shuttle; the current EMU is then described in greater detail.

As a survey of possible models for the soft fabric construction of the EMU space suit, the third section presents a brief history of fabric modeling. It explores connections between the textile industry and computer graphics research, and documents ways in which these communities have attempted to model woven material. Several finite-element approaches to textile modeling are discussed, and shown to be impractical for the scale of the EMU model and EVA simulation. Arguments are then presented for the empirically-based model chosen to represent the space suit.

The final section touches on biomechanical models for the human operator inside the space suit. The human model is an essential feature of any simulation analyzing or attempting to predict human motion, and many levels of model complexity are available. Several models for describing the musculoskeletal system are discussed, with an emphasis on properly choosing the level of detail appropriate for this research.

2.1 Extravehicular Activity (EVA)

Extravehicular activity refers to work performed in space, outside the confines of a larger vehicle. Astronauts wear protective space suits during EVA missions, and are generally attached to the larger spacecraft by means of foot restraints, handholds, and tethers. Crewmembers perform EVA tasks in a variety of situations, including planned repair or service missions to orbiting spacecraft, contingency problem-solving missions under unusual or emergency conditions, and EVA research tasks in preparation for greater challenges. Extravehicular activities are often mission-critical, and frequently involve multiple crewmembers performing delicate and complex tasks. Though daunting, these tasks are based on some prior experience in the space environment -- astronauts have been walking in space for over thirty years.

2.1.1 Historical Perspective

The first spacewalk took place in 1965, when Soviet cosmonaut Alexei Leonov emerged from his Voskhod space capsule to float free in space for over ten minutes, tethered to his spacecraft by a 5 meter umbilical. This successful EVA demonstrated that a human being could exist somewhat comfortably in space, protected only by a pressurized suit from the cold vacuum outside. Leonov's pioneering excursion introduced a decade of rapid EVA progress for both Soviet and American space programs, improving technology and technique with every mission [Newman, 1997].

America's first experience in EVA was the Gemini program, which experimented with various operational aspects of working in space, including the positioning of foot restraints, handholds, and controlling crewmember exertion and fatigue. It was not until the Apollo program, however, that an American space program utilized the full operational capability of the EVA astronaut.

Apollo crewmembers accumulated 160 total hours of extravehicular activity time exploring the lunar surface, and their ability to conduct experiments while suited illustrated the great scientific value of EVA capability. Soil and rock samples were collected on a regular basis, and short trips from the lander were executed flawlessly. The Apollo program led to swift advances in suit design, and it demonstrated the feasibility of pressure suit use on a planetary surface.

EVA continued to prove its value during the Skylab program, as early as the arrival of the first crew. A damaged hull and missing solar panel created problems from the start, but an impromptu EVA allowed the Skylab II crew to deploy a second solar panel and to bring the station back to operational status. Problems during the space program showed that emergency and other unplanned situations were well suited for EVA solutions, allowing a skilled astronaut to bring human problem-solving abilities and operational flexibility directly to the worksite.

During the late Apollo and Skylab programs, the Russians experienced some difficulty with EVA reliability, but they resumed operations in 1977 during the Salyut 6 program. Several maintenance and repair EVA missions were successfully completed, paving the way for regular EVA excursions during the Salyut 7 space station program. Russian cosmonauts carried out multiple scientific experiments during EVA, and continued to maintain the capability to perform contingency EVAs in emergency situations.

The United States began performing EVA missions under the new Space Transportation System (STS), also known as the Space Shuttle, in 1983. EVAs were performed routinely through 1985, giving American astronauts valuable experience working in space. Astronauts performed both planned and contingency EVA missions, exploring the use of special EVA tools and work techniques. Making history in February 1984, Bruce McCandless successfully demonstrated the Manned Maneuvering Unit (MMU) as the first free-flying crew propulsion device to be used in space.

More recently, two service missions for the Hubble Space Telescope (HST) dramatically demonstrated the necessity for and effectiveness of the EVA crewmember. Following the disastrous news that the multibillion-dollar space telescope was flawed, four EVA astronauts on Hubble Space Telescope Service Mission 1 (HST SM-01) spent a record-setting five consecutive days of EVA upgrading the damaged optics. Extensive pre-flight training helped the mission to its spectacular success, which generated months of positive media exposure for NASA and the entire space program. A key benefit of EVA programs, the human drama of astronauts doing their work in space is a captivating scene with which to encourage interest in the space sciences throughout the world.

The success of critical EVA missions like these depends greatly on NASA's commitment to extensive pre-mission EVA preparation and training. Without the practice afforded by endless hours of training, EVA missions would be far more hazardous and unpredictable than the forgone successes that modern spacewalks have seemingly become. The space environment is unforgiving, however, and ground-based training does not always provide a complete picture of what is to come; new methods such as computational simulation are currently being developed to complement and improve upon existing physical simulation methods.

2.1.2 Extravehicular Activity Simulation

Extravehicular activity takes place in an environment utterly unfamiliar to human beings. Minute forces can move massive objects with the greatest of ease, moving around is more an issue of pointing in the right direction and controlling angular momentum than moving any one part of your body, and all activities are performed inside the giant shell of a space suit. NASA recognized some of the problems that might be encountered if astronauts were expected to adapt instantaneously to the microgravity environment, and initiated an extensive EVA training and mission preparation program to help acclimate crewmembers to differences they might encounter.

NASA began with physical simulators, hoping to give astronauts work experience in a setting that approximates microgravity. Large water tanks were constructed as neutral buoyancy training facilities, e.g., the large facility at Johnson Space Center in Houston, and since then every planned EVA mission has been rehearsed in the water. Immersion in water can provide a zero-gravity "feel" by canceling the force of gravity with buoyancy, but it also adds the phantom forces of hydrostatic drag and water displacement inertia to counteract astronaut motion. EVA rehearsals taking place in the water can actually train astronauts to expect these false forces, conditioning their neuromuscular system to anticipate water drag and pressure nonexistent in space, and thereby hinder EVA performance. Additionally, astronauts have noted that the work environment is not exactly the same: letting a tool drop in the water tank sends it to the bottom within seconds, while the same tool in space would remain motionless in front of the astronaut.

Eliminating water's unwanted viscosity, air-bearing floors have also been used extensively to train astronauts in two-dimensional translational motion. Small jets of air hold astronauts and the objects they might interact with, such as satellites, away from the ground on a frictionless cushion of air; this simulator has proven very useful for linear translation and mass-handling tasks, helping astronauts to better judge the responses of massive yet weightless objects. Unfortunately, the three degrees of freedom (including rotation) that this simulator provides only reproduce 50% of the true environment, and attempting to include more degrees of freedom via some mechanical apparatus only serves to introduce unwanted friction. Additionally, the friction forces present in the floor itself and the small air drag encountered with large objects can affect the realism of a simulated EVA condition.

Many of these problems are solved by using a specially modified aircraft to fly parabolic, zero-gravity trajectories; NASA's KC-135 training aircraft was designed for just that purpose. Passengers on this aircraft are provided with approximately 25 seconds of free-fall time, during which they are free to move about the weightless environment in six degrees of freedom. The unlimited range of motion creates a very realistic simulation, but the limited available space inside the fuselage of the aircraft usually constrains astronauts to intravehicular activity (moving about the interior of a spacecraft) simulations. Additionally, the limited duration of weightlessness does not provide a very good approximation of a six or seven hour EVA. To make up for this deficiency, pilots fly 40-50 repeated sequences of zero-g parabolas and 2-g pullouts, but this usually serves only to induce motion sickness in the cabin. Vomiting inside a space suit would be extremely dangerous, creating a real possibility of clogged air lines and choking; therefore, EVA tasks are usually not simulated on KC-135 aircraft.

Each of these simulation methods has a specific type of task for which it is well-suited, and the various simulators have been used in combination to provide adequate venues for research and training certain aspects of EVA missions. They are all limited, however, by the realities of a one-g field on earth. Combinations of various physical simulators will never exactly reconstruct the conditions experienced on orbit, and may introduce difficulties during spaceflight if astronauts learn to expect forces or torques that are merely artifacts of an imperfect simulator. Additionally, physical simulators are extraordinarily expensive to operate for the amount of time that complicated EVA missions require. The Hubble servicing mission, for example, spent millions of dollars on operations training in the JSC neutral buoyancy facility alone. With the severe demands of International Space Station EVA missions looming on the horizon (Figure 2.1), there is a clear need for more economical means of simulating EVA tasks that might help to fill in the gaps left by physical simulators.

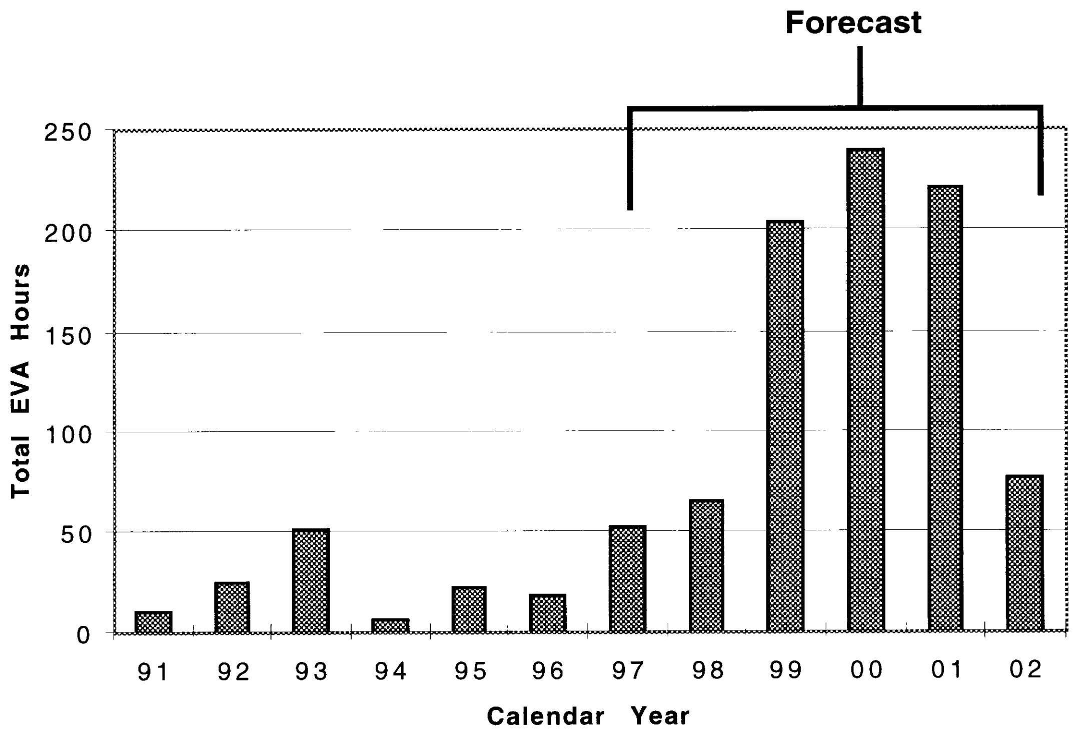


Figure 2.1 Historical and Future NASA EVA Time [NASA, 1996].

Computational dynamic simulation embraces this role well, skirting the limitations of physical simulators by moving into the virtual computer world, while keeping time and expense to a minimum. Quantitative analysis of EVA operations is possible without large-scale experimental devices, and produces results important to the EVA community [Schaffner, 1995]. Problem areas can be identified, and EVA choreography refined to account for predicted complications. This method also provides a full six-degree-of-freedom arena in which the simulated EVA crewmember can work, allowing for unrestrained multibody interactions in a microgravity environment. Computer simulation can tease out the subtle differences between physical simulation and the actual operation, complementing existing simulation technologies and ensuring that EVA will continue to enjoy the successes for which it is famous.

These simulations would be lacking an important characteristic, however, if they failed to account for the dynamic effects of the EVA crewmember's space suit. The pressure suit has been an integral part of the aerospace world since World War II, and deserves a special place in space history.

2.2 The Space Suit

Space is, quite simply, the most hostile environment known to human life. None of Earth's myriad life support mechanisms exist in the spaceflight environment, and new and threatening elements such as space debris only compound the problem. Astronauts must therefore replicate Earth's atmosphere in tightly sealed and protected spacecraft, bringing the necessities of life with them into space.

From time to time it is necessary for astronauts to leave the confines of their vehicle and address problems directly at the worksite. It is impossible to leave the spacecraft without some form of environmental protection, and for this reason, a miniature spacecraft able to perform all requisite life support functions was developed: the space suit.

2.2.1 Environmental and Physiological Concerns

As early as the nineteenth century, pioneering aviators had already discovered many of the dangers presented by the unforgiving aerospace environment. High-altitude balloonists knew that any flight above three or four thousand meters required warm clothing to protect against cold temperatures and a special oxygen supply to keep the pilots from passing out. As these early adventurers traveled to higher and higher altitudes, they quickly discovered that oxygen alone was insufficient to keep them alive, and that a positive-pressure delivery system was necessary to force oxygen into their lungs.

These balloon pilots had directly experienced the atmospheric changes that take place during the transition from sea-level to spaceflight altitudes. At sea level, the atmosphere is composed primarily of four gases at a total pressure of 101.3 kPa, but this relatively thick layer disperses rapidly with altitude (Table 2.1).

Table 2.1 Atmospheric Composition at Sea Level and 5,500 m.

Gas	Concentration	Partial Pressure at Sea Level (kPa)	Partial Pressure at 5,500 m (kPa)
Nitrogen	78.08%	79.1	38.4
Oxygen	20.95%	21.2	10.3
Argon	0.93%	0.9	0.4
Carbon Dioxide	0.03%	0.03	0.01

The partial pressure of oxygen (ppO_2), which is the net contribution of oxygen molecules to the overall pressure, has decreased to 10.3 kPa by 5,500 m; humans cannot function to make intelligent decisions at partial pressures of oxygen lower than 13 kPa. Therefore, all excursions to high altitude require a special apparatus to provide the pilot with a minimum 13 kPa ppO_2. This can be achieved by either increasing the concentration of oxygen relative to other atmospheric gases, or by providing a higher total pressure of all gases and thus increasing the ppO_2 as well. In space, there is no atmospheric pressure -- a space suit must provide a minimum 13 kPa 100% O_2 environment for astronauts to function normally.

In addition to the problem of atmospheric composition, there are a number of unique aspects to the space environment that affect the ability of humans to do work. These include thermal, radiation, and micrometeoroid hazards that must be met with some layer of protection between the astronaut and the outside environment. At a typical Space Shuttle orbiting altitude of 200 km, the EVA crewmember must face a temperature of 582° C, a total pressure of less than 9×10^{-5} N/m^2, and multiple forms of dangerous radiation. Although the temperature is not a direct measure of heat at this altitude, helpful effects such as convective cooling on Earth are not present, and heat transfer must be addressed in special ways.

Electromagnetic radiation comes to low earth orbit (LEO) in 2 forms: solar cosmic radiation (SCR) and galactic cosmic radiation (GCR). SCR comes from two types of ionizing radiation emitted by the sun, both in the form of high-energy particles (protons, electrons, and photons). First, the solar wind produces a stream of ionized particles (protons and electrons) at a density of about 5 cm^{-3} that blows toward Earth, frequently becoming trapped in the Van Allen belts at 300-1200 km altitude. In addition to this continual solar wind, solar flares produced by storms in the solar magnetosphere generate high radiation doses in very short periods of time.

Galactic cosmic radiation, on the other hand, consists of high-energy particles and heavy nuclei originating beyond the solar system. Low altitude orbits are largely shielded from GCR radiation by the Earth's magnetic field; interplanetary or deep space travel, however, will be open to the effects of GCR and man-made shielding must be provided.

Micrometeoroids and other forms of space debris are yet another hazard in the space environment. The average flux of micrometeoroids through low earth orbit is poorly known, but most particles have masses below 10^{-6}g. Still, the relative velocities of these objects can cause serious damage to spacecraft and space suits. Potentially an even greater danger are the more than 40,000 bits of space debris larger than 1 cm currently floating in earth orbit, left by previous operational or damaged spacecraft. Space suit design must take into account the possibility of puncture by such an object; literally billions of smaller particles are assumed to exist in the same orbital environment.

It is clear that designing a space suit to perform flawlessly under these harsh conditions is no easy task. Still, the spirit and dreams of mankind have led us to step boldly into this new frontier, and counter the environmental hazards as best we can. Beginning with the first pressure suit for high altitude flight, space suit design work has progressed in both scientific understanding and engineering acumen, and the current Space Shuttle EMU pays homage to those advances with its effective design.

2.2.2 A History of the Extravehicular Mobility Unit (EMU)

The space suit performs three basic functions: maintenance and constraint of the pressurized atmosphere necessary for life support, protection from environmental hazards, and provision for enough mobility to perform useful work. The life support system must control suit pressure, regulate the atmospheric mixture of gases, monitor and adjust temperature and humidity, and supply all power, computer, and communications facilities for the astronaut; this integrated system is enclosed in the portable life support system (PLSS) mounted on the back of the EMU. The actual suit garment provides environmental protection, using a multi-layer construction to absorb micrometeoroid strikes, reflect harmful radiation, circulate air and cooling water, and contain the pressurized atmosphere. Finally, unique features are incorporated into suit construction that allow for unusually high mobility in a garment that resembles a large balloon. The complexities of such a design have intrigued engineers for nearly 70 years, and have yet to be perfected.

2.2.2.1 Early Suit Designs

The first full pressure suits were manufactured by the B.F. Goodrich company in the late 1930s as an attempt to overcome the hazards of high-altitude aviation. Company engineer Russell Colley designed a series of pressure suits for the daredevil and adventurer Wiley Post, who was determined to achieve high-speed stratospheric flight. These primitive suits were molded in the shape of a seated pilot using liquid latex, and they held about 17 kPa when fully pressurized (Figure 2.2, left). An outer layer of cotton was then added for thermal protection, and metal rings were fitted about the knees to bunch the fabric, allowing for slight movement.

In the early 1940s, U.S. military aviation began to take an interest in pressure suit research and began to fund various efforts across the country. This new generation of suits maintained a pressure of 24 kPa, and they incorporated a new “tomato worm” design with segmented rubber bellows at the joints, allowing the wearer to move without changing the joint volume appreciably (Figure 2.2, center). Unfortunately, this was only true in the unpressurized case, and the suits invariably ballooned up to become rigid and uncomfortable when pressurized. Military aviators had no way of performing their assigned duties, e.g., sighting through and manipulating a bomb sight, while wearing such a suit, so it remained a research project with no operational status.

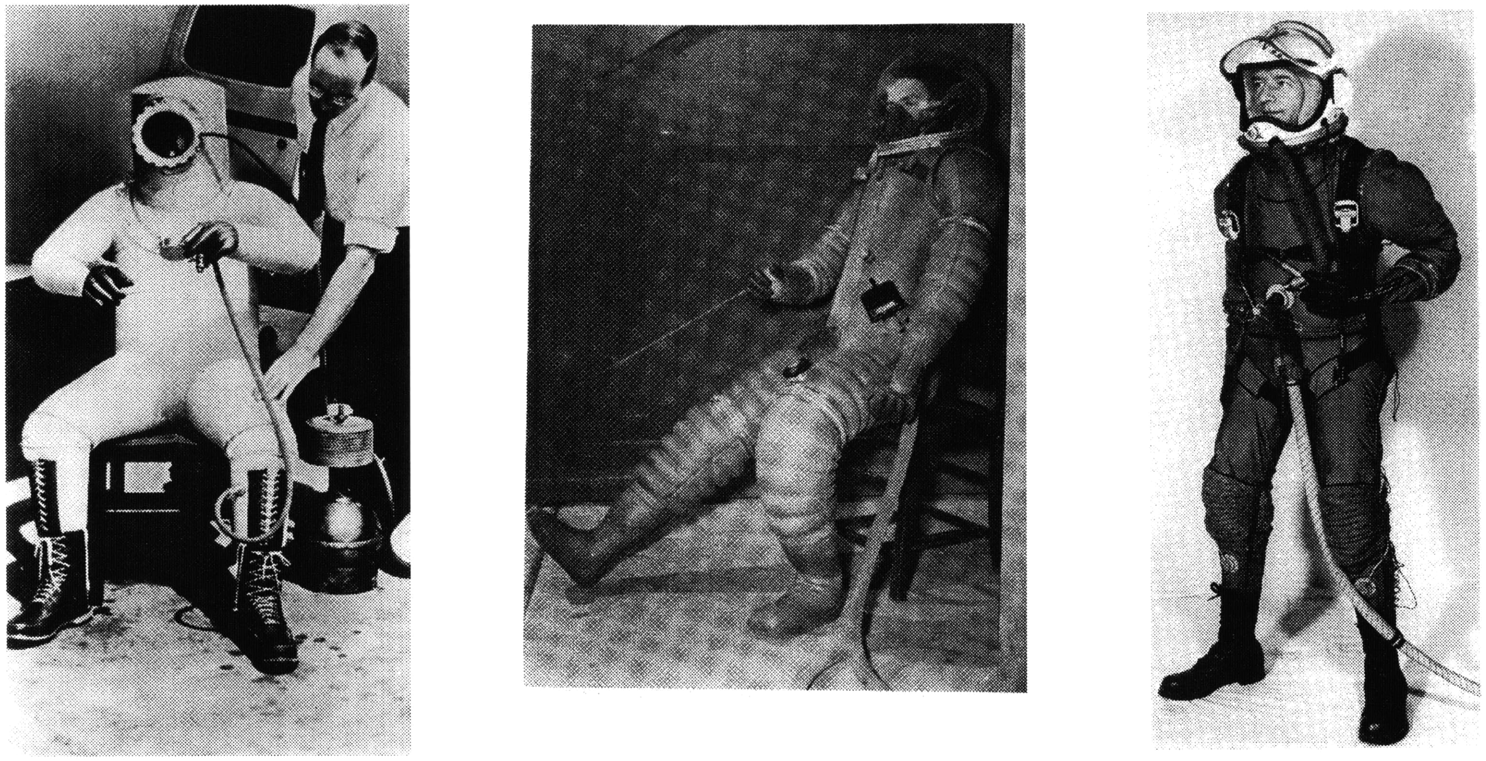


Figure 2.2 Early Pressure Suits.
The first full pressure suit (left), the tomato-worm design (center), and the David Clark A/P22S-3 (right). (Kozloski, 1994)

Lack of mobility remained the number one problem in pressure suit design until 1956, when the David Clark Company created the A/P22S-3 suit for the Air Force (figure 2.2, right). The suit incorporated a soft nylon netting on the outside of the pressure bladder, which constrained the pressurized suit to a predetermined shape and effectively prevented it from ballooning out of control. The industry had been waiting for this critical breakthrough, and it represented the turning point in pressurized suit design. The resulting suit was fairly mobile, comfortable, and provided adequate thermal protection in its outer layers. B.F. Goodrich then took a similar approach, and improved upon David Clark's Air Force design for the Navy, with modifications including an automatic pressure control system, rotating joint bearings, and a robust ventilation system.

2.2.2.2 Mercury

In 1959, a newly created NASA selected the B.F. Goodrich Company over ILC and David Clark to produce the Mercury space suit. Their suit evolved during the course of the various Mercury missions, but retained a general 4-layer construction throughout (Figure 2.3). An outer layer of aluminized nylon provided flame and abrasion protection for the astronaut. Just under that, neoprene-coated nylon enclosed the actual pressure vessel to provide insurance for leakage protection. The pressure bladder was made of vulcanized, double-walled nylon, ventilated along its inner wall for air circulation and cooling. Finally, the astronaut wore long-john underwear with interwoven patches that circulated air for comfort and temperature control.

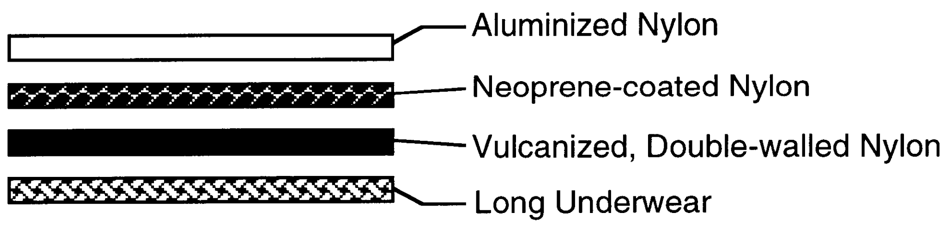


Figure 2.3 Four-layer construction of the Mercury space suit.

Mobility was somewhat improved in the Mercury suits; Colley and the Goodrich engineers addressed the constant-volume problem in a slightly different manner than the David Clark Company. Instead of a whole-body "linknet," they sewed inelastic restraining cords along the elbow and knee joints to prevent the pressurized suit from expanding (see section 2.2.3). Combined with their rotating joint bearings, many of the initial mobility problems associated with full-pressure suits had been solved.

2.2.2.3 Gemini

The Gemini program provided the first real test of American space suits: outer space. For the first time, a space suited astronaut would attempt to work outside the protective environment of the spacecraft. Additionally, the suit would have to be comfortable enough for multi-day missions; the crew would wear different intravehicular activity (IVA) suits while inside the capsule and don special extravehicular suits for EVA. Although many variants were used over the course of the 2-year Gemini program, the standard Gemini space suit was the David Clark Company's G4C (Figure 2.4).

The most significant change from the Mercury suit was the implementation of David Clark's "linknet" restraint system, instead of the Goodrich restraining cords that were sewn on the outer edges of knee and elbow joints. The linknet layer was similar in design to David Clark's A/P22S-3 Air Force pressure suit: a loosely interconnected network of light but strong fibers was woven throughout the suit, forming a fishnet-like layer to restrain suit volume. The Gemini IVA suit, worn inside the spacecraft, consisted of five layers; the EVA suit added an additional 17 layers for environmental protection during spacewalks.

Like the Mercury suit, astronauts wore long underwear as a base layer, connected to biomedical sensors attached to the body. The next two "comfort" layers of the basic Gemini suit were oxford nylon layers of different weights that helped to distribute ventilation gas over the entire body and to separate the astronaut from the subsequent pressure bladder. The bladder itself was fabricated from neoprene-coated ripstop nylon, and the restraint layer directly above was the innovative linknet. IVA suits then added a covering layer of Nomex, called Dupont HT-1 (High Temperature), which protected the suit against snags from spacecraft instrumentation and sharp objects.

The EVA suit retained the same pressure bladder and restraint layer configuration, but replaced the IVA suit's single Nomex layer with a thick, protective covering fabricated from multiple layers. Two inner HT-1 neoprene-coated Nomex layers served as micrometeoroid protection, and they were covered by 14 layers of alternating aluminized Mylar and superinsulating unwoven Dacron for thermal and radiation protection. Finally, the suit was covered with a third Nomex layer to give abrasion and further micrometeoroid protection.

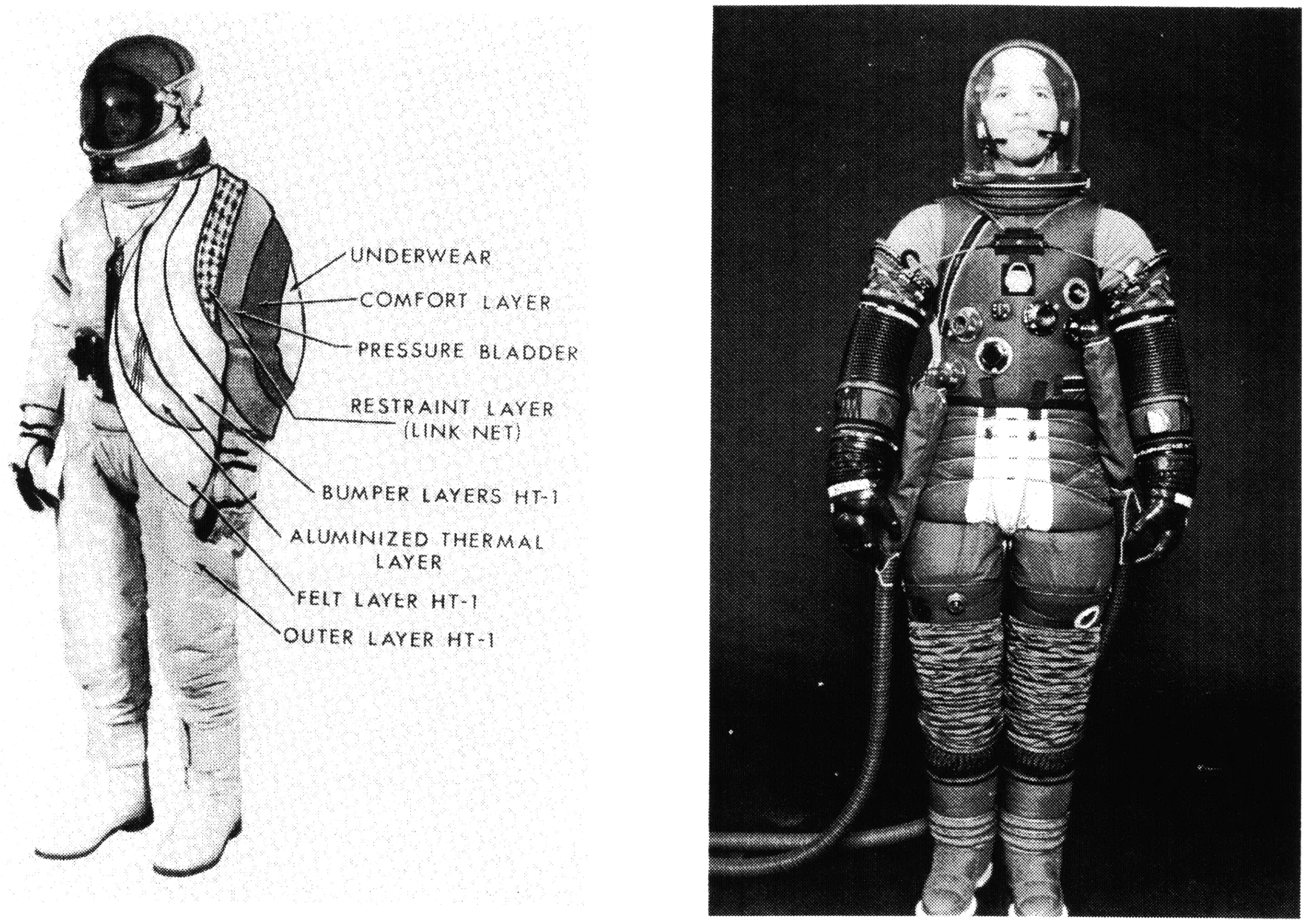


Figure 2.4 Gemini (left) and Apollo (right) space suits. (Kozloski, 1994)

Gemini astronauts performed both stand-up (in which the astronaut merely stands in the open hatch area and never leaves the vehicle) and complete EVA missions, proving the feasibility of doing useful work in space. Complications developed during several EVAs, but provided good experience for future missions. For example, *Gemini 9-A* astronaut Gene Cernan experienced problems keeping his body within his EVA work area, increasing his workload while attempting to complete several tasks. Simply maintaining body position required so much energy that he overheated and suffered early fatigue, cutting the mission short. Important lessons were learned about EVA restraints and astronaut placement, contributing to future Apollo improvements.

2.2.2.4 Apollo

The Apollo program saw several key changes in space suit design. The primary suit was manufactured by the International Latex Corporation (Dover, Delaware) under a contract from Hamilton Standard, the overall life support integrator. It included advances in joint design, using a flexible convolute design rather than the block-and-tackle mechanism used to physically size the suit, and waist and neck flexion joints were added to provide sitting capability and greater comfort during long-duration EVAs (Figure 2.4). All together, the Apollo Lunar EVA suit was composed of 25 separate layers. The suit also replaced umbilical life support lines with a portable life support system (PLSS) designed for remote operation on the lunar surface. The complete suit system, including the PLSS, was designated the extravehicular mobility unit (EMU).

Problems with overheating on the Gemini missions were taken seriously during Apollo suit design. Anticipating much greater workloads during lunar EVAs than zero-gravity Gemini missions, improvements had to be made on Gemini's air-cooled suit. The long underwear worn by Gemini astronauts was replaced in the Apollo suit by a three-layer liquid cooling garment (LCG), comprised of a lower nylon layer, a vinyl tubing layer which circulated cooling liquid, and an upper nylon spandex layer to restrain the tubing. The circulating water could transfer heat away from the astronaut by direct conduction, and virtually eliminate perspiration.

The pressure garment assembly (PGA) was similar to the Gemini design, with some modifications. ILC engineers retained the general linknet design to restrain each joint, adding a cable/block-and-tackle assembly to further restrain the shoulders and knees. The outer EVA covering was also similar to Gemini, incorporating the alternating Mylar and Dacron layers, but it added a Kapton layer for fire and thermal radiation protection as well as Teflon outer coverings for abrasion and flame resistance.

Apollo astronauts tested the EMU thoroughly during operations on the lunar surface. It was found that some joints, especially lower-body regions, were extremely stiff during extended periods of walking on the surface; similarly, problems in the thigh area of the suit created balance instabilities, and the lunar gravity caused the astronauts to lean forward while upright. *Apollo 17* astronaut Eugene Cernan reported great difficulty operating a rock drill while encumbered by a pressurized suit, and noted a rise in his heart rate. Still, surface EVAs were extraordinarily successful and produced groundbreaking scientific experiments, proving yet again the necessity for EVA capability.

2.2.2.5 Shuttle

The current operational space suit for the Space Shuttle program is the EMU manufactured by Hamilton Standard (PLSS) and ILC (fabric suit). The EMU is a direct descendant of the Apollo suit, designed with many of the same features as the 1970 version. The space suit is constructed from two pieces: a hard upper torso (HUT) and a soft-shell pressure garment surrounding the limbs. The HUT is the main structural member for the suit, to which all other pieces are mounted. It is connected via bearings to the soft fabric arms and the lower torso assembly (LTA), also made of fabric. The PLSS is mounted to the rear of the HUT; this is a key dynamic aspect of the suit, for it moves the suited astronaut's center of mass to the rear of its usual location.

The astronaut still wears a liquid cooling and ventilation garment (LCVG), made from nylon and spandex with a tricot lining. Instead of Apollo's separate cooling tube layer, ethylene-vinyl-acetate tubes are woven throughout the spandex layer. The soft-shell pressure garment is fabricated from urethane coated nylon, covered by a woven Dacron restraint layer. As with the Apollo EVA suits, outer environmental protection is provided by a thermal and micrometeoroid protection garment (TMG). The TMG liner is neoprene-coated ripstop nylon, shielded by five layers of aluminized Mylar with a Dacron backing for thermal and radiation protection . Finally, a

Teflon-coated blend of Kevlar and Nomex synthetic fibers covers the suit for puncture, abrasion, and tear protection (Figure 2.5).

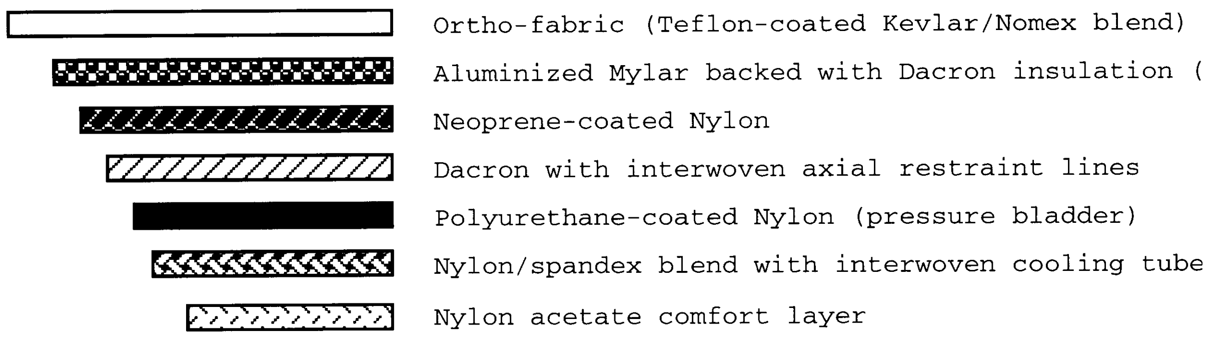

Ortho-fabric (Teflon-coated Kevlar/Nomex blend)
Aluminized Mylar backed with Dacron insulation (
Neoprene-coated Nylon
Dacron with interwoven axial restraint lines
Polyurethane-coated Nylon (pressure bladder)
Nylon/spandex blend with interwoven cooling tube
Nylon acetate comfort layer

Figure 2.5 Space Shuttle EMU multi-layer soft shell construction.

The multi-layer construction is somewhat bulky, and restricts motion largely to the central workspace in front of the astronaut. The space suit and life support system weigh a combined 146 kg when fully charged with consumables for EVA operation; the added mass makes it critical to move with slower, determined motions to prevent needless energy expenditure while moving the mass of the suit.

The enclosed suit is also pressurized to 29.6 kPa, substantially lower than Shuttle cabin pressure but high enough to prevent decompression sickness during the transition from cabin to suit pressure, following a brief 1-3 hour prebreathe. Higher pressure suits would allow for a shorter or eliminated prebreathe stage before EVA operations, but soft-shell high pressure suits would encounter more substantial forces due to changing volume within the suit. Pressure and volume issues contribute substantially to the suit forces imposed on an astronaut, and are a continual challenge to EVA designers.

2.2.3 The Constant Volume Challenge

Changing the volume of an enclosed space, like the Shuttle EMU, requires work. If the EMU elbow joint is represented by a simple cylinder, then the enclosed volume is $\frac{\pi}{4}D^2L$, where D is the cylinder diameter and L is the length [Newman, 1997]. If the cylinder is now bent, the inside edge collapses while the outside edge remains extended, and the enclosed volume is reduced (Figure 2.6).

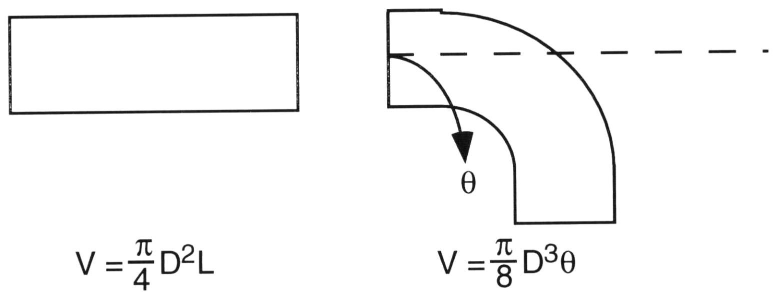


Figure 2.6 Volume change during joint flexion.

Assuming a pressure-regulated environment like the EMU, the reduction in volume under constant pressure requires work:

$$W = \int PdV = p(V_{INITIAL} - V_{FINAL}). \qquad (2.1)$$

Under these conditions, the elbow joint would require over 230 Newtons of force to move. This is a clear impossibility for astronauts who would like to do useful work in space over long periods of time. Fortunately, ingenious solutions such as the David Clark Company's "linknet" restraint system have addressed this problem with great success, and advanced development hard-shell suits have come close to eliminating the difficulty altogether. The current Space Shuttle EMU uses axial restraint lines to hold back the excess fabric in joints, providing room for the joint to expand on one side and contract on the other, thus maintaining a nearly constant volume (Figure 2.7).

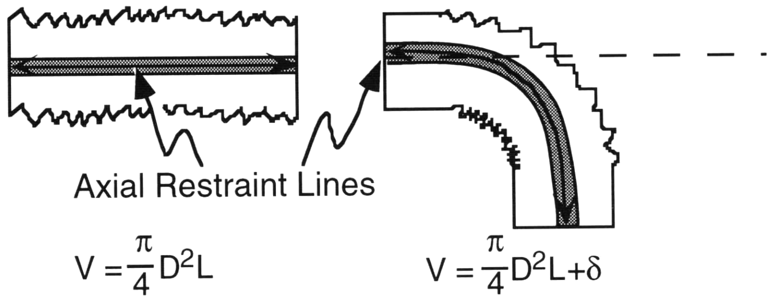


Figure 2.7 Nearly constant-volume space suit joint.

In practice, however, the suit joint does not bend so that its centerline is exactly circular; unbalanced forces cause it to shift to the outside. Also, axial restraint lines are offset slightly from the centerline to ensure balance and stability at the joint. During motion, the axial restraints will provide close to constant volume, but over- and under-shoot by small amounts, especially near the edges of the range of motion at extreme angles of flexion and extension. Therefore the astronaut must ultimately do small amounts of work moving the joint from position to position. Additionally, bunching and expanding the actual fabric construction of the joint requires some degree of work.

Finding ways to maintain constant volume in a pressurized suit while providing adequate mobility for the suited individual is a problem that has challenged engineers for decades. Still, the current soft-shell suits must ultimately give way to changing volume at certain areas of a joint's workspace, and demand that mathematical models of suit dynamics take this property into account. Modeling methods for flexible fabric shells must also be explored, and used to design a model that closely approximates EMU parameters.

2.3 Flexible Fabric Modeling

Computer modeling of woven fabrics has gained great popularity in the last ten years, generating interest in both the textile industry and the computer graphics community [Breen, 1996]. Textile engineers are working to understand the mechanical properties of woven fabrics, including stress-strain relationships, thread interaction, and bending moments in order to design new fabrics and improve clothing performance. Most work has focused on experimental testing of various fabrics [Kawabata, 1980], but scientists have recently become more interested in apparel modeling as an aid to computer-aided manufacturing. Automated manufacturing in the clothing industry has traditionally been limited by the inability of robots to manipulate flimsy, flexible garments in deterministic ways; with an accurate model of fabric dynamics, however, machines on assembly lines could be programmed to pick clothes up, analyze them for defects, and pack them for shipping.

Computer graphics research, on the other hand, has continually been questing to produce more realistic "virtual actors" for computer animation and entertainment. The search has created great interest in ways to "clothe" actors in realistic garments, and to simulate objects like drapery and tablecloths. What started out as a set of simple geometric tools for drawing realistic fabric has rapidly expanded into a plethora of detailed physical models describing fabric behavior, and computer science is now looking to the textile industry for detailed descriptions of material properties. Most of the realistic models presented in the literature are finite element types, and they are explored first as a possibility for EMU fabric modeling.

2.3.1 Finite Element Methods

Finite element techniques represent cloth models as discrete triangular or rectangular grids with small masses at each intersection node (Figure 2.8).

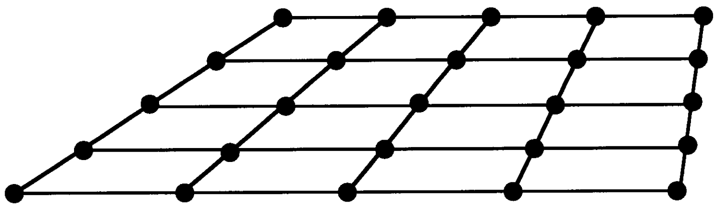


Figure 2.8 Finite element representation of a swatch of cloth.

Forces between nodes are calculated as simple or complex differential equations based on Newton's Law, for example:

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{1}{m}\left(\mathbf{F}_{external} + \mathbf{F}_{spring} + \mathbf{F}_{visco.} + \mathbf{F}_{plastic}\right), \qquad (2.1)$$

where **r** is the position of a node in 3-space and the forces are external, position-dependent, velocity-dependent, and plastic, respectively [Sakaguchi, 1991]. To obtain solutions for position and velocity of the nodes, equation 2.1 is discretized by a finite-difference method, resulting in a set of simultaneous ordinary differential equations. These equations are then numerically integrated over time to produce trajectories for each node.

This type of general model allows scientists to simulate practically any situation by modifying the underlying forces **F** to model arbitrarily complex dynamic systems. For example, a simulation of fabric falling through a medium such as air or water might incorporate a full numerical implementation of the Navier-Stokes equations to develop aerodynamic forces for inclusion in $\mathbf{F}_{\text{external}}$ above. Similarly, mechanical properties of the fabric can be lumped together in position-dependent and velocity-dependent categories to create visco-elastic forces within the material. These forces can be described at an even lower level by attempting to model the behavior of individual threads in a woven textile environment, though the cost in processing time for such an effort becomes prohibitive [Keefe, 1994].

One important property that can be included in $\mathbf{F}_{\text{ext}}$ is fabric interaction with itself and with other objects. Collision detection for rigid bodies is a well-explored subject [Moore, 1988], but it becomes more complex when dealing with a flexible cloth. Several investigators have explored algorithms for performing these computations [Lafleur, 1991; Yang, 1993], and, while successful, they suffer from excessive overhead computing node-to-node distances between the fabric nodes and underlying nodes representing the human shape. Most techniques therefore utilize a form of "windowing," which only computes collision algorithms for nodes known to be in proximity to one another. This solution would not be sufficient for the EMU case, though, if the model were attempting to simulate the various layers of the EMU garment. Multiple layers interacting at a large number of sites would pose a serious computational challenge to such an implementation.

The central drawback to physically-based modeling approaches is the sheer number of computations that must be performed to adequately simulate large sections of fabric. For example, [Eberhardt, 1996] presents an excellent particle-based model for cloth deformation that shows great promise for application to EMU suit modeling efforts. His model is based on empirical cloth data from tension, shearing, and bending experiments, and accounts for important features such as anisotropy and hysteresis in the fabric data. Such an approach seems perfect for mimicking EMU suit performance, until the computation overhead is analyzed (Table 2.2). Even though this model is considerably faster than most finite-element methods, its processing times remain in the realm of frame-by-frame rendering, unsuitable for on-the-fly, real time simulation.

Table 2.2 Computation Time for Particle-Based Draping Simulation.
Computation times in seconds for a single timestep (40 ms) during simulation of a cloth draping over the indicated objects, on a Silicon Graphics R8000 workstation with additional R8010 fpu **[Eberhardt, 1996]**.

	30 x 30 Grid 900 Particles			**52 x 52 Grid 2,704 Particles**		
Example	**min. (s)**	**max. (s)**	**mean (s)**	**min. (s)**	**max. (s)**	**mean (s)**
Table	0.3	7	5	7	67	**21**
Round Table	3	8	6	10	94	**24**
Sphere	1	9	6	4	50	**27**

A relatively small grid (52 x 52) of 2,704 particles requires almost 30 seconds to compute a single frame of animation for the relatively simple task of draping cloth over a sphere, which is a mathematically well-described object. The human, on the other hand, is much more geometrically complex, and likely to be in contact with several sections of the EMU inner lining simultaneously. It is no wonder that many physically-based models are restricted to running on SGI Power or Cray supercomputers.

Finite element methods have been extremely successful in replicating the observed behavior of flexible objects in a limited number of situations. Unfortunately, the current benchmark for such simulations is the "draping problem," in which a computer simulates a large section of fabric falling through a medium (air, water, oil) and coming to rest draped over an object such as a rod, sphere, or human. Though clearly beneficial to both the apparel industry and computer animation, these simulations have been optimized for specific conditions that would hinder performance in the more general EVA environment. Even if finite element calculations were practicable, an EVA suit model would still need to perform volume calculations on the enclosed inner surfaces of the suit to recover the pressure forces generated as the suit volume changes while the astronaut moves about.

Combining these volume calculations with the already astronomical processor time required to simulate soft fabric on the scale of a full EMU space suit by finite element means, it is therefore impractical to use such an approach to model and simulate the suit during EVA simulation. Considering the complexity and processing time required (up to 6 hours) to perform basic unsuited multibody dynamic simulations [Schaffner, 1997], discrete node-based methods are simply too costly in terms of simulation efficiency.

The rapid growth of textile simulation and constant improvement in available processing power, however, show that finite element techniques may be an excellent way to model the EMU soft-shell construction in the future. Preliminary studies assessing the applicability of node-based approaches to the simulations of a multi-layer shell, such as the space suit, are encouraged. These studies may show that finite element methods should be strongly considered for the next generation of suit models.

2.3.2 Empirical Methods

Although finite element and other physically-based methods choose parameters (stress, strain, damping, shearing, etc.) empirically, they must ultimately break their system down into thousands of individual points or complex systems of equations, wasting valuable processor time. Fully empirical methods, on the other hand, simply find an input-output relationship between two parameters and apply the result directly. Though the resulting model is not derived from first principles and not nearly as elegant, the end result is identical as long as the data on which the model is based covers all possible areas of the operational state space.

For the EMU, this requires data for each joint throughout its entire range of motion. All observed suit effects on the crewmember should be taken into account, including static loading, dynamic or velocity-related resistance, and increased angular inertia. The model then becomes a recording of observed effects in every possible condition, and simulation is just a matter of recalling the appropriate data and applying it at the proper time, as suggested in Figure 2.9.

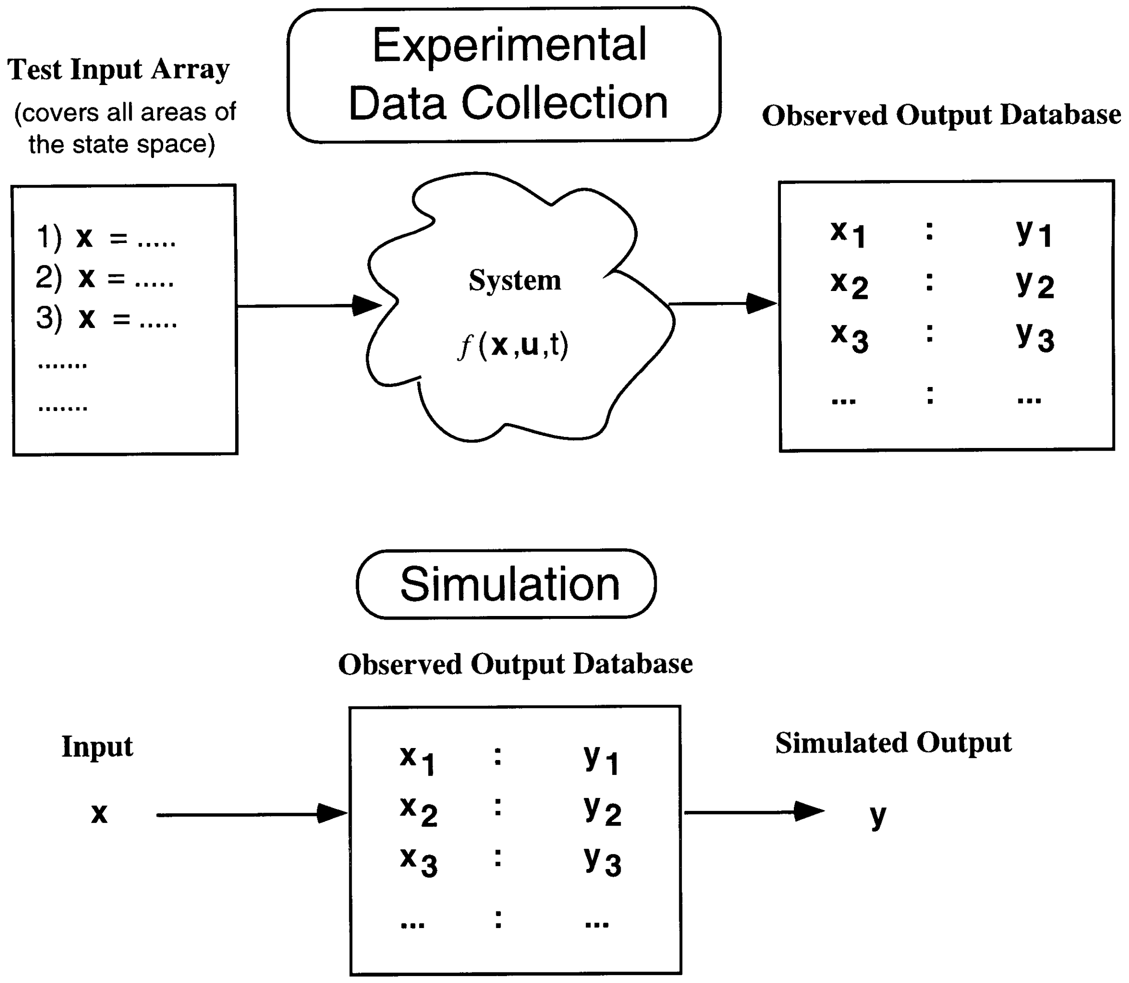


Figure 2.9 An empirically determined model of system *f(x,u,t)*.

This data-driven model is extremely fast (on the order of milliseconds), provided that the lookup database of observed outputs is organized in an efficient manner. If the output database is, in fact, complete, then the model will also be an exact model of the system $f(\mathbf{x}, \mathbf{u}, t)$ and the system will be fully identified. In practice, however, it is usually impossible to obtain a full set of input-output data. This is not a problem with linear systems, where matching a few key parameters to the data achieves complete system identification and easily extends the model throughout state space. For nonlinear systems, however, the form of the system may be entirely unknown. Therefore the model must be constructed from a few well-chosen test inputs, and interpolation used to extend the model into a continuous set of values. This paradigm is used in the next chapter to create a nonlinear, data-driven model of the extravehicular mobility unit space suit.

2.4 Musculoskeletal Modeling

Before creating a complete simulation of astronauts at work, however, one additional element of the system must be specified: a model of the human being inside the space suit. The underlying goal of this research effort is to examine the human performance detriments and benefits that arise from various and dynamic environmental parameters during EVA, including effects from the space suit. In order to better understand how these issues influence the human body, it is necessary to include a model of the human operator in the simulation framework. This model facilitates the representation of simulation data in familiar terms, such as the angular positions of the various human joints and the muscular work required to execute a given task.

A musculoskeletal model is a simple representation of the human body's biomechanical structure as a system of mathematically related elements. It includes a model for the skeletal structure of the human (passive) as well as a model for the muscular system (active) that serves to control the position of the skeleton. All skeletal muscles act as levers, converting linear muscle contraction to joint torque, rotation, and movement of the skeleton; the most common type is the third class lever shown in Figure 2.10.

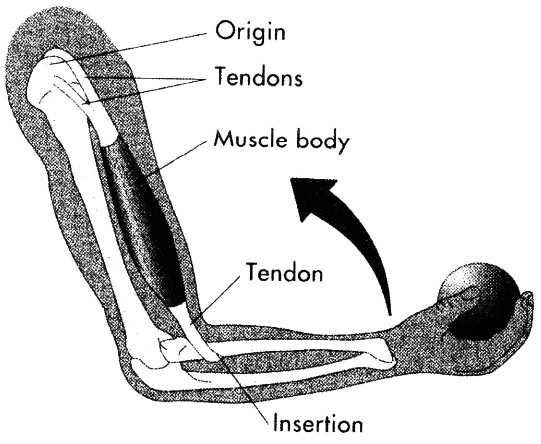


Figure 2.10 The human elbow joint is a third class lever. [Thibodeau, 1996]

Skeletal models can be geometrically complex, but they are generally well described by experimental observation. Most models use a multi-segment approach that constructs a human figure from limb segments, such as the upper arm, lower arm, and hand (Figure 2.11).

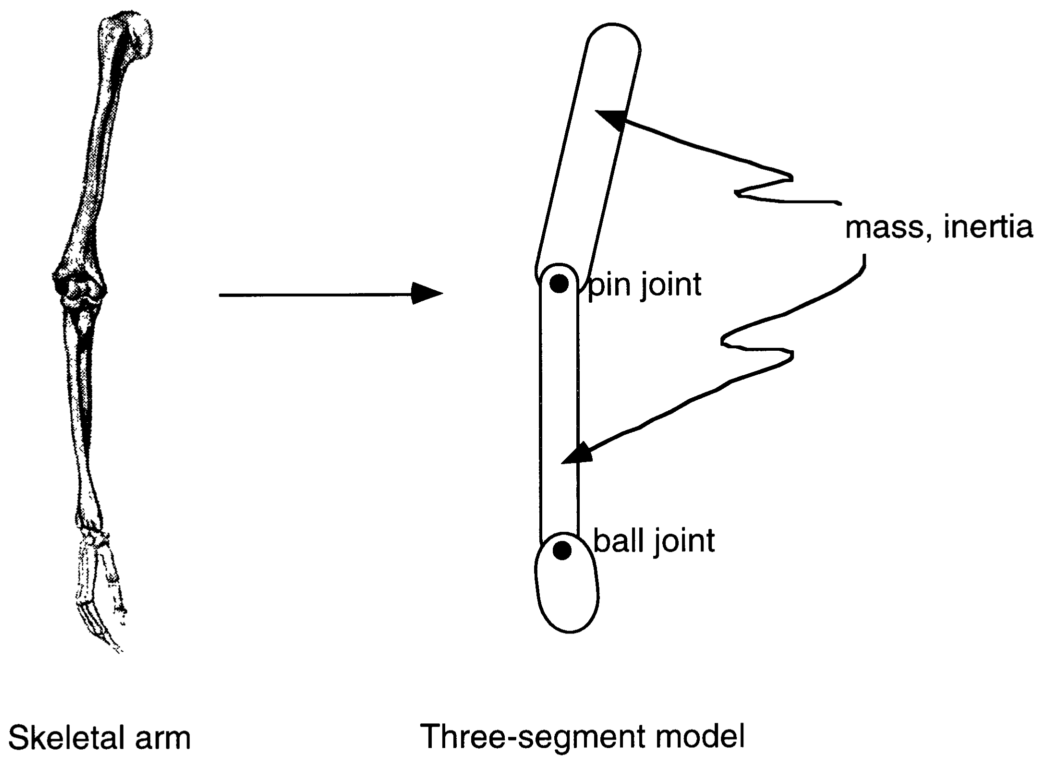


Figure 2.11 Three-segment model of a human arm.

Levels of detail are selected specifically for the task at hand; for example, a musculoskeletal model created to investigate a new EMU glove design would necessarily incorporate individual segments for each bone of the hand. Larger-scale simulations, however, might treat the hand as a rigid body articulated only at the wrist for computational simplicity. If unusually high or low forces are observed at the hand using this latter model, a more detailed model might later be implemented to investigate the region more thoroughly.

Muscle tissue is strikingly similar in design across all known animal kingdoms [McMahon, 1984]. Muscles are universally composed of thick bundles containing muscle "fibers;" these fibers are bioelectrically stimulated by the nervous system to contract in specific patterns and sequences, generating movement at skeletal joints. Muscles in the human body usually act in opposing agonist-antagonist pairs, such as the bicep and tricep muscles, to actuate limb motion in either direction. Additionally, each muscle is frequently composed of several synergistic heads (e.g., the 3 heads of the tricep), which act in a coordinated manner to stabilize and fine-tune the specific direction of motion. In addition to different parts of the same muscle generating similar forces, whole muscles can act in the same direction to assist in a single movement. The countless possibilities for musculoskeletal structure and function demand efficient models that can be implemented without undue complexity.

Many ground-breaking research efforts involving muscle models can be traced directly to the work of A.V. Hill [Hill, 1938], who first proposed a simplified, lumped-parameter model for muscle based on laboratory observations. This model introduces a contractile element (CE) in parallel with a lightly-damped elastic tissue element (PE) and in series with a second elastic element (SE) as shown in Figure 2.12.

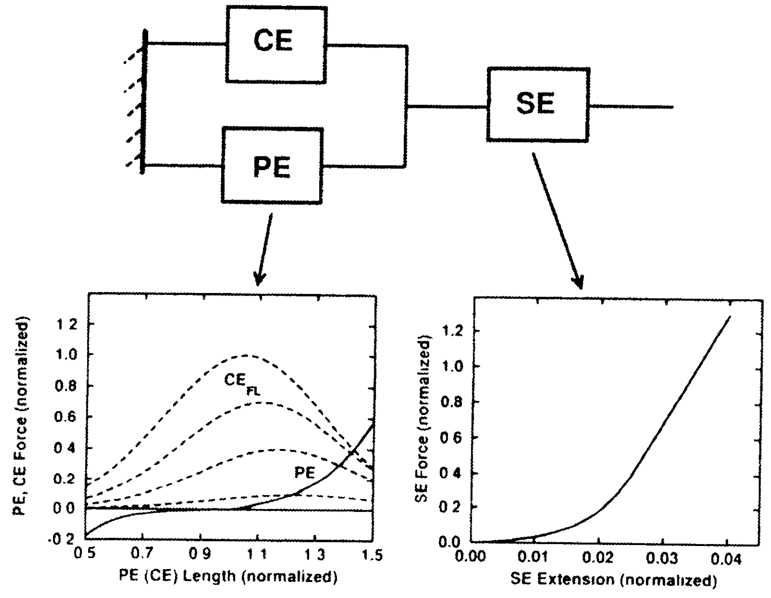


Figure 2.12 Hill-type muscle model [Winters, 1995].
Nonlinear force-length relationships for the CE, PE, and SE elements. The CE is shown at several levels of activation (FL).

The SE and PE elements can be represented as nonlinear springs, and the CE force-generating element contracts with a force that is both velocity and length dependent. Many implementations of this model assume maximally fast movements, or maximal contraction of muscle. Movements can then be represented as sequential maximal contractions of agonist and antagonist muscles. Everyday movements, however, usually involve lower speeds and submaximal activation; maximal exertion models may not be as accurate or applicable to such tasks. With these caveats in mind, models have been developed that better approximate the wide range of muscle performance, including provisions for both intrafusal and extrafusal muscle [Winters, 1995].

It is possible to implement the Hill model without specifying either PE dynamics or a force-length relationship for the CE [Happee, 1992], greatly simplifying the analysis, but such models are only applicable for small changes in muscle length and for joint angles close to the center of the normal range of motion. Nevertheless, such models have generated useful data within their limited regions of applicability [Happee, 1995], and the results are highly correlated to experimental findings. The central drawback to the Happee implementation (1992, 1994, 1995), however, is the resulting complexity of the overall system. The relatively simple task of shoulder flexion and extension within a limited (10°) range of motion was simulated using no less than 20 muscles divided into 95 individual "muscle elements." An even more detailed model of myodynamic processes can be found in [Hatze, 1981], exploring muscle activation at the cellular level.

The question of how detailed to make a muscle model is well addressed by Winters (1995), and the answer is largely task-specific. For the simulation of human muscle, large limb motions yield high muscular forces acting through moment arms that are often quite small and act in several different planes of motion. The resulting system is often highly sensitive to initial conditions, and must be modeled with great care. The more detailed the model, the higher the probability that results will be highly sensitive to variations in one or more of the model parameters.

Winters suggests that for each task under investigation, only a subset of a highly detailed model's parameters will typically be of significance for the specific movement being analyzed. It is therefore prudent to utilize models that capture the essential features of a specific motion, but do not over- or under-simplify the musculoskeletal system for that specific task. As an example of this philosophy, Greene and McMahon (1979) represent the lower body anti-gravity musculature under normal and hypergravity conditions with a damped mass-spring system.

Their simple lumped-parameter analysis represents the opposite end of the spectrum from Happee's 95-element shoulder model, above. For Greene and McMahon, all musculoskeletal parameters have been lumped into two basic measurements: overall system stiffness and a damping ratio.

Considering the high costs of computation for more complex models [Pandy, 1992], simplified lumped-parameter models are far more efficient for analyzing large movements with limited resources. These larger movements may not require the detailed muscle-by-muscle analysis that a physiologist might need for anatomically accurate movement investigations. Lumped-parameter models help to identify key modes of system operation, and they can estimate larger measures of task performance where per-muscle data is not required. It is important, however, not to haphazardly group muscles together and thereby oversimplify the problem, leading to inaccurate conclusions based on models that neglect key effects. This places more pressure on the accurate characterization and grouping of various musculoskeletal parameters, but lumped-parameter models have been explored to some degree [Winters, 1985].

For the purposes of this research, lumped-parameter models have several desirable characteristics. They maintain computational simplicity (relative to individual muscle models) while including an appropriate level of detail, and produce substantial biomechanical insight without excessive physiological modeling. Specific muscle models that address these issues will be considered in greater detail in Chapter Three.

3. METHODS

A dynamic model for the EMU is developed in section one of this chapter, representing the main contribution of this thesis. Basic inertial properties are examined, and the hysteresis model introduced in section 3.3 is shown to allow accurate representation of imposed space suit joint torques. The human model constructed for trial simulations is discussed in section two, and the final section describes the simulations that test the EMU model.

3.1 EMU Space Suit Model

Three key characteristics of space suit dynamics are used to create the EMU model: mass, inertia, and suit-imposed torques. An empirically-based model is developed and implemented numerically based on observed suit parameters, utilizing a modified Preisach model to describe the hysteretic torque characteristics at space suit joints.

3.1.1 Inertial Properties

The most basic properties of the EMU that can be modeled are its mass and various moments of inertia. Mass properties were obtained from Hamilton Standard (Windsor Locks, CT), and are summarized in Table 3.1.

Table 3.1 EMU Mass Properties

EMU Element	Mass (kg) [includes consumables]
Primary Life Support System	44.73
Secondary Oxygen Pack	10.19
Avionics and Battery	16.97
Hard Upper Torso	12.29
Arm Assemblies	7.87
Lower Torso Assembly	20.72
Helmet and Visor	8.23
Gloves	2.31
LCVG	2.94
Total EMU Mass	**146.40**

The mass characteristics are included in the system description file used by the SD/FAST software package to generate the equations of motion for the EMU model that are used for simulation (Appendix E). Moments of inertia were then calculated for the suit elements based on suit dimensions from the NASA STD-3000 Man-Systems Integration Standards; these moments are also included in the system description file.

3.1.2 Suit-imposed Torques

The most profound effect of the EMU on astronaut performance involves the imposed torques, or "springback forces," in the joints; these forces are generated when the suit fails to maintain a constant volume during movement, and the astronaut does work to change suit volume and position. Also, the multi-layer soft shell construction of the suit behaves much like a giant winter coat: as the limbs move back and forth between joint limits, the suit fabric tends to bunch up, and eventually the sheer quantity of material compressed into a small volume creates a firm limit on the maximum flexibility of a joint.

Space suit design engineers, of course, attempt to minimize these effects in order to maximize suit flexibility and astronaut mobility[Klaus, 1989; Menendez, 1993]. For the most part, suit designers have succeeded; joint restraints are quite effective in maintaining a nearly constant joint volume through the most frequently used areas of an astronaut's workspace, and innovative designs such as flat-patterned mobility joints allow easy flexing of joints without substantial twisting, bunching, or stretching of thermal micrometeoroid garment (TMG) material.

The effects being modeled, then, are not the major suit forces associated with early space suit designs which lacked these improvements [Kozloski, 1994]. Rather, the attempt is to model the EMU's deviation from a "perfect" suit, one which would exert no forces to counteract astronaut movement. In practice, subtle changes in the orientation of joint restraints during a particular motion can lead to minute variations in the restraint forces applied to EMU joints. Additionally, TMG fabric will be thick and bulky regardless of joint design innovation, and it will contribute some small countertorque to astronaut movement. These variations lead to slightly varying joint volumes during particular sequences of arm and leg motion, and increasing fabric bunching toward the joint limits; the result is the characteristic joint angle vs. suit torque curve shown in Figure 3.1, obtained from torque measurements of a pressurized suit elbow over its range of motion [West, 1989].

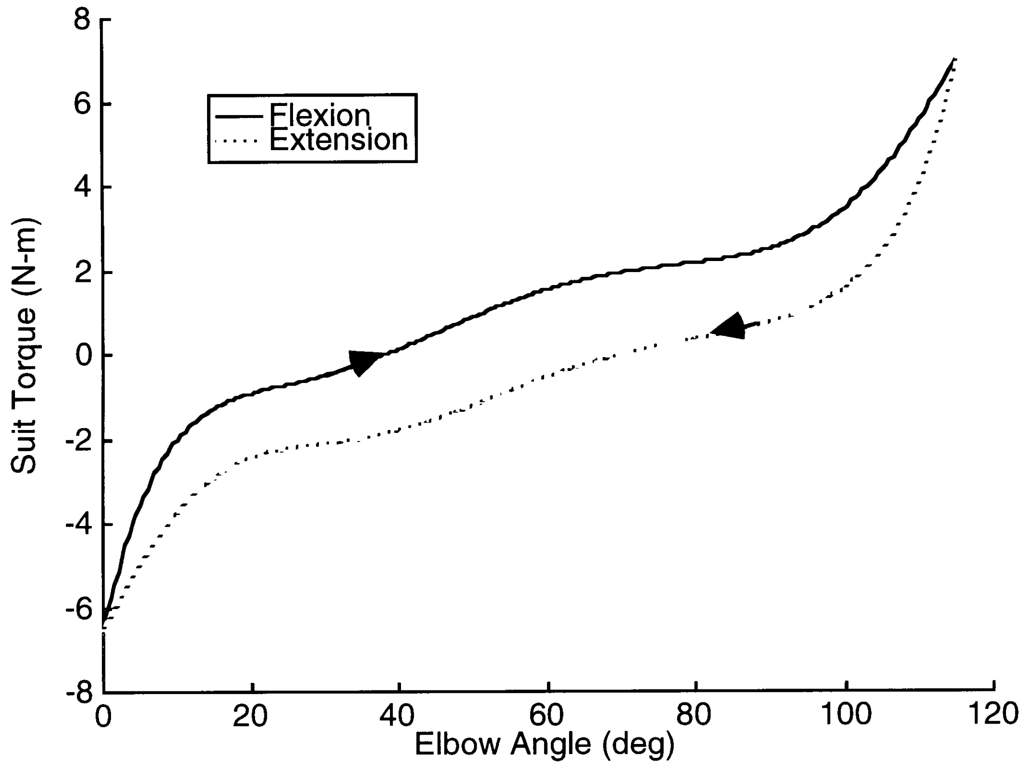


Figure 3.1 Hysteresis nonlinearity observed at the elbow joint of the EMU.

This modeling effort has focused on developing a dynamic model of the EMU from existing suit data, which presumably represents some combination of the effects that various suit imperfections have on suit performance. Although mass and inertial properties complete the model, these springback torques represent the critical element for assessing detriments to human performance in the EMU.

The torque vs. range-of-motion curve for a suit joint reveals a significant degree of hysteresis, and must be modeled as such. Hysteresis represents a form of "memory" in the space suit joint; countertorques generated by the suit are uniquely dependent on a specific previous history of joint motions which have been somehow "stored" in the structure of the suit. In order to effectively represent this property in a simulation framework, it is necessary to create a mathematical model of the phenomenon.

3.1.3 Hysteresis Model

This section develops a mathematical model for hysteresis based on decades of hysteresis modeling, and discusses the specific implementation required for EMU applications. A brief introduction to hysteresis is provided in the first section, followed by a summary of the Preisach model, used across many scientific disciplines to describe hysteretic systems. A conversion process to move observed data into the EMU model is presented, and finally numerical methods for efficient computation are discussed.

3.1.3.1 Introduction to Hysteresis

Systems that involve hysteresis in one form or another are common throughout science, and their abundance has led to a number of quantitative investigations of the phenomenon. The most famous description was developed by F. Preisach in 1935, who took an intuitive approach to modeling the hysteresis observed in magnetic systems. His model was later given mathematical rigor by the Russian mathematician M. Krasnoselskii [1983], and generalized to an abstract form that encouraged widespread application. For a thorough discussion of the Preisach model and its later modifications, Mayergoyz [1991] is an excellent reference, and his geometric interpretation of the Preisach model will be discussed below.

The first step is a definition of hysteresis. As seen in Figure 3.2, the hysteretic system element can be characterized by its input-output relationship.

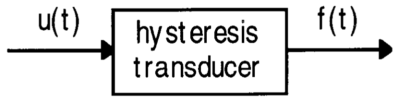


Figure 3.2 Hysteresis transducer.

A *hysteresis transducer* is a system element with input $u(t)$ and output $f(t)$ that creates a multibranch nonlinearity for which branch-to-branch transitions occur after input extrema (Figure 3.3). The nonlinearity shown has three changes of direction; a new branch of the nonlinearity is created after each input extremum.

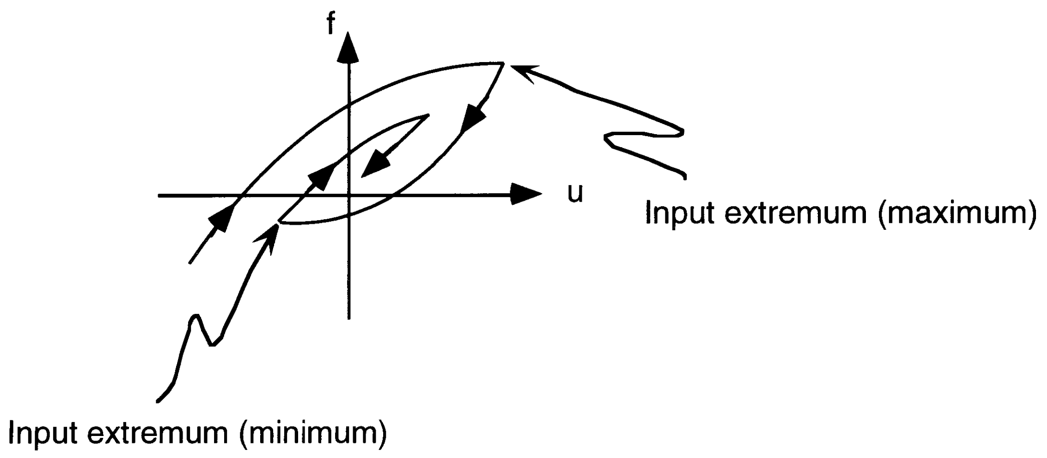


Figure 3.3 Input-output time history for a hysteresis transducer.

The classical Preisach model is static, meaning that future output values depend only on past extrema and not the speed of transition between these extrema. It does not account for the dynamic, or rate-dependent properties that may be present in hysteresis nonlinearities. For slow input variations, however, the static model performs well; it is assumed herein that time effects are negligible and a static model applies. This assumption is made based on empirical evidence from EVA video footage, during which suited crewmembers universally move with small, deliberate motions that are slow and controlled.

Another assumption is that the hysteresis nonlinearities observed with the EMU are of the "nonlocal memory" type (Figure 3.5): a particular sequence of past extremum values of input $u(t)$, $t<t_0$, is required to specify a unique path in the f-u plane. "Local memory" hysteresis transducers, on the other hand, define $f(t)$ for all future $u(t)$, $t \geq t_0$, based entirely on the current state (Figure 3.4) without regard for previous values.

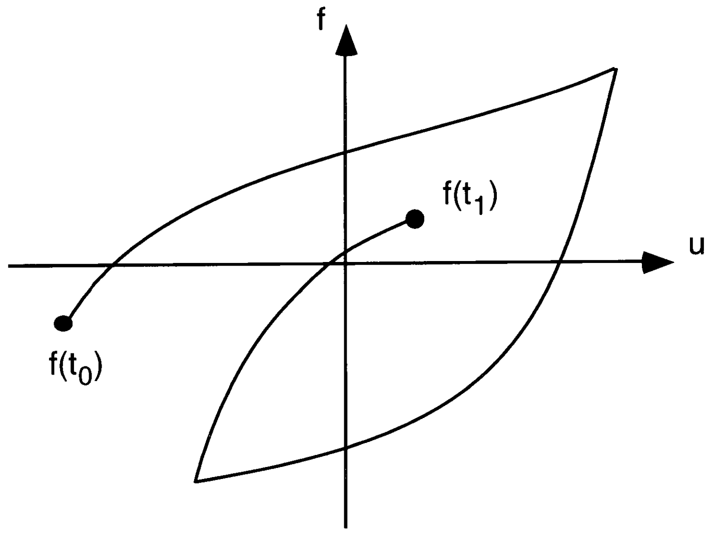


Figure 3.4 Hysteresis transducer with local memory.
Given $f(t_0)$ and $u(t>t_0)$, $f(t>t_0)$ is deterministic and $f(t_1)$ is unique.

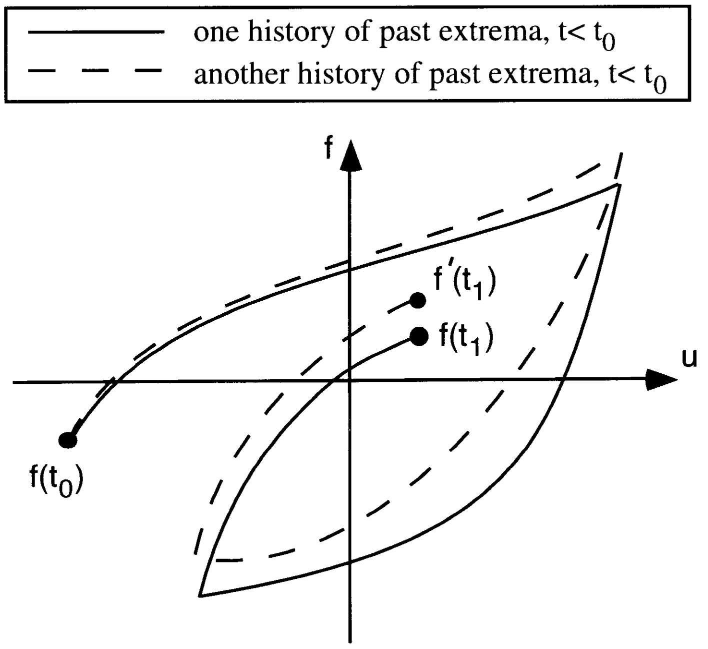

one history of past extrema, t< t0
another history of past extrema, t< t0
f
f'(t1)
f(t1)
u
f(t0)

Figure 3.5 Hysteresis transducer with nonlocal memory.
Starting from $f(t_0)$, identical input histories $u(t>t_0)$ can produce different trajectories in the f-u plane and specify different points $f(t_1)$ if $u(t<t_0)$ and $u'(t<t_0)$ specify different past histories of input extrema.

Hysteresis transducers with local memories are therefore Markovian, insofar as future states of the system will be determined precisely by the current state and future inputs. Here the intuitive argument is made that the instantaneous state of a space suit is uniquely dependent on the particular arrangement of fibers and restraints in the suit at a specific instant in time. This arrangement is decidedly non-Markovian, that is, the fibers in a space suit TMG are highly unlikely to revisit identical positions throughout the suit at the same instant in the future. Space suit states are therefore instantaneously unique and poorly described as a Markov process. Additionally, experimental studies have shown that hysteresis transducers with nonlocal memories are the more likely candidate for most applications [Mayergoyz, 1991]. A complete mathematical description of this nonlocal type is provided by the Preisach model for hysteresis.

3.1.3.2 The Preisach Model

The building blocks for the Preisach model are hysteresis operators $\hat{\gamma}_{\alpha\beta}$, defined by their switching values of input α and β, shown in Figure 3.6.

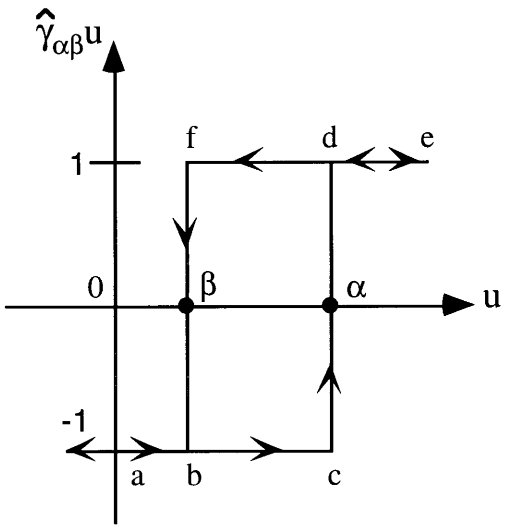
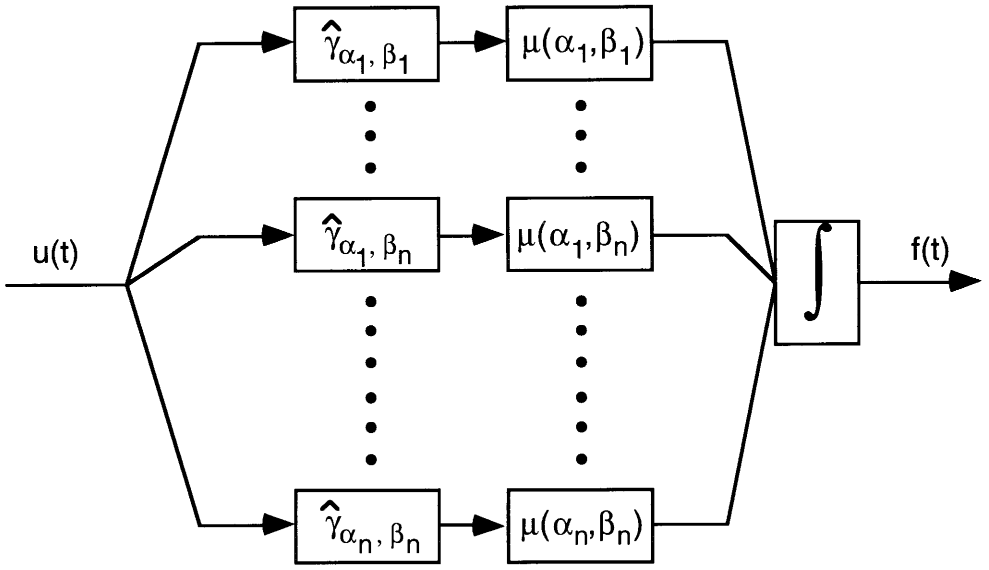


Figure 3.6 A simple hysteresis operator $\hat{\gamma}_{\alpha\beta}$.

For monotonically increasing input starting below β, output follows the ascending branch *abcde* and saturates at 1 (the "up" position). Similarly, the $\hat{\gamma}_{\alpha\beta}$ operator traces the descending curve *edfba* for monotonically decreasing values of input, saturating at -1 (the "down" position). For input values on the open interval (-∞, +∞), the hysteresis operator $\hat{\gamma}_{\alpha\beta}$ thus produces only two values of output, 1 and -1. Now consider a complete set of operators $\hat{\gamma}_{\alpha\beta}$ whose switching values blanket the entire range of input, and also allow for an arbitrary weight function *μ(α,β)* to scale operator outputs individually. Then the Preisach model for hysteresis can be defined as

$$f(t) = \iint_{\alpha \geq \beta} \mu(\alpha, \beta)\hat{\gamma}_{\alpha\beta} u(t) d\alpha d\beta . \tag{3.1}$$

The input *u(t)* is applied to each simple hysteresis operator in succession, appropriately weighted, and then summed over the set of all operators to produce the total output *f(t)* (see Figure 3.7).



$\hat{\gamma}_{\alpha_1, \beta_1}$
$\mu(\alpha_1,\beta_1)$
$\hat{\gamma}_{\alpha_1, \beta_n}$
$\mu(\alpha_1,\beta_n)$
u(t)
f(t)
$\hat{\gamma}_{\alpha_n, \beta_n}$
$\mu(\alpha_n,\beta_n)$

Figure 3.7 Hysteresis decomposition into simplest hysteresis operators $\hat{\gamma}_{\alpha\beta}$.

Operators are triggered to change state as *u(t)* crosses their unique switching values (α,β), and thus affect the total output incrementally through the integral described in equation 3.1. The geometric interpretation of this result presented below will clarify the way in which such a system retains a "memory" of past events.

3.1.3.3 A Geometric Interpretation

Each simple hysteresis operator is associated with a unique set of up and down switching values α and β, and there exists an infinite set of operators on the α - β plane so that they are continuously distributed. These operators are represented on the half-plane α ≥ β, where each point (α, β) corresponds to its respective hysteresis operator (see Figure 3.8). This diagram assumes that the "up" switching threshold α is greater than the "down" threshold β, which will be amended later for application to the EMU. Triangle ***T*** represents the hysteretic workspace, bounded above by the upper limit of system saturation α_0 and to the left by the lower limit of saturation β_0.

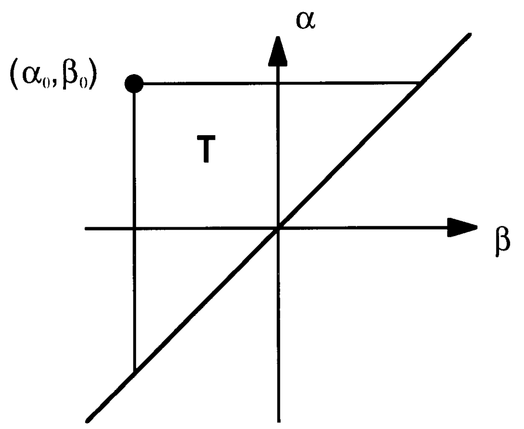


Figure 3.8. The $\alpha \geq \beta$ half-plane.

Assume that input *u(t)* is initially below β_0, and all hysteresis operators are in the "down" position. Then each operator has a value of -1, and the output $f(t_0)$ is equal to the negative saturation value for the system, f^-. Now let the input increase monotonically to a maximum value u_1 at some time t_1 (Figure 3.9).

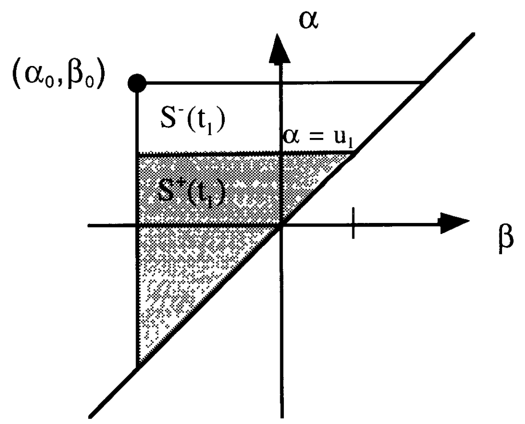


Figure 3.9 Triangle *T* at time t_1.
Input *u(t)* has increased through the workspace to switch on all operators in region $S^+(t_1)$. The interface between the two regions is the line $\alpha = \alpha_1 = u_1$.

As the input is increased, each operator $\hat{\gamma}_{\alpha\beta}$ with an "up" switching value less than *u(t)* is turned on, weighted by its appropriate $\mu(\alpha,\beta)$, and added to equation 3.1 to compose the current *f(t)*. This process subdivides triangle *T* into two regions: $S^+(t)$ where all operators are in the +1 or "up" position, and $S^-(t)$ where all operators remain in the -1 or "down" position. As *u(t)* increases, the two regions are separated by a horizontal link moving slowly up the α axis.

Let input *u(t)*, after increasing to u_1, change direction and monotonically decrease to u_2 at time t_2. As the input decreases, all operators whose "down" switching value is greater than *u(t)* will switch back to the "down" state, as shown in Figure 3.10.

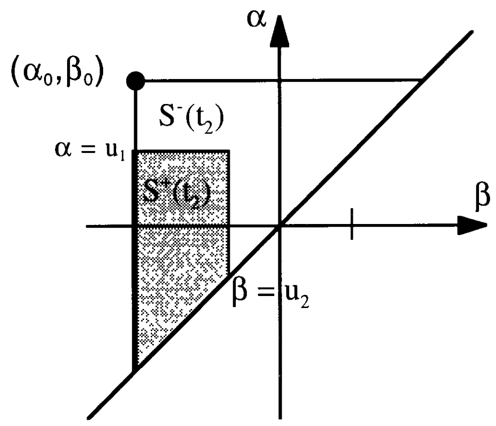


Figure 3.10 Region $S^+(t_2)$, bounded by the lines $\alpha = u_1$ and $\beta = u_2$.

Decreasing input results in the formation of a new vertical link on the diagram, starting at $\alpha = \beta = u_1$ and moving to the left along the β axis.

If this process continues for a while, and input extrema continue to decrease in absolute value, the interface *I(t)* between the regions $S^+(t)$ and $S^-(t)$ will become a complex staircase whose vertices represent a complete history of all past input extrema (Figure 3.11).

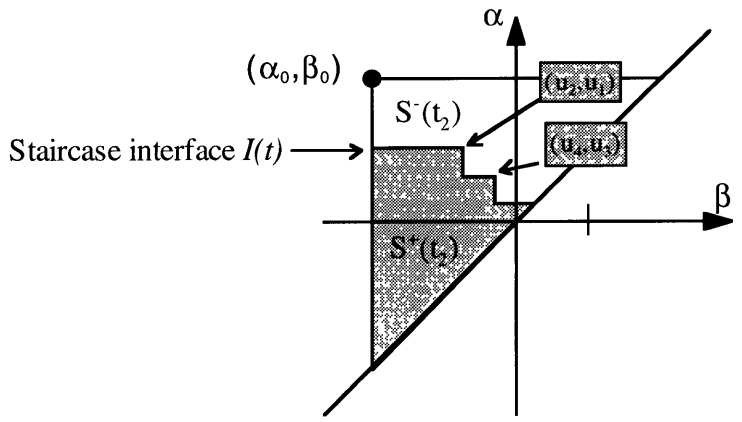


Figure 3.11 The staircase interface *I(t)*.
The interface is formed by vertices representing the past extrema of input *u(t)*.

If at any point *u(t)* increases to a value greater than recent upper extrema, several of the vertices may be eliminated from the interface; for example, if *u(t)* were to increase to a value greater than u_1, all vertices would be eliminated and the interface would once again resemble the straight line of Figure 3.9. This is the "wiping out" property characteristic of most hysteretic systems, allowing large input swings to effectively reset the system.

The above figures illustrate the way in which the Preisach model maintains a record of past events in the system, creating a detailed history of input extrema; additionally, it simplifies the calculation of transducer output *f(t)*. From the geometry of Figure 3.11, it is clear that equation 3.1 can be split into two parts, representing the two regions $S^+(t)$ and $S^-(t)$:

$$f(t) = \iint_{\alpha\geq\beta} \mu(\alpha,\beta)\hat{\gamma}_{\alpha\beta}u(t)d\alpha d\beta = \iint_{S^+(t)} \mu(\alpha,\beta)(+1)d\alpha d\beta + \iint_{S^-(t)} \mu(\alpha,\beta)(-1)d\alpha d\beta \qquad (3.2)$$

Evaluation of these two integrals depends on 1) the unique shape of the regions $S^+(t)$ and $S^-(t)$, which is stored in the extrema coordinates of the interface *I(t)*, and 2) experimental specification of the weighting function $\mu(\alpha,\beta)$, which will determine the shape of our overall hysteresis nonlinearity.

To accurately represent the EMU hysteresis environment, some modifications must be made to the general model. The Preisach model has been developed under the assumption that input *u(t)* moves on the unbounded interval (-∞,+∞) while output is restricted to the bounded interval [f^-,f^+]. Any input values over some upper threshold $u_{max} = \alpha_0$ result in a positively saturated output $f(t) = f^+$, and input values below some lower threshold $u_{min} = \beta_0$ lead to a negatively saturated output $f(t) = f^-$.

The corresponding situation for an EMU joint begins with joint torque input $\tau(t)$ on the unbounded interval $(-\infty,+\infty)$ and produces an output joint angle $\theta(t)$ on the bounded interval $[\theta_{min}, \theta_{max}]$. It is more useful in the simulation framework, however, to calculate an imposed joint torque $\tau(t)$ based on input joint angle $\theta(t)$. This allows the space suit to be represented as a simple dynamic constraint on astronaut motion, rather than a complex constraint on forward-dynamic control algorithms. Although iterative approaches might provide a feasible implementation of the torque-input system, the angle-input simplification is easily achieved by borrowing heavily from the mathematics of the Preisach model, if not the spirit. The hysteresis operator decomposition described in equation 3.1 applies to any set of hysteresis curves, and can be used for calculating $\tau(t)$ or $\theta(t)$ without difficulty.

The positive and negative saturation values of output at each joint are therefore represented simply by the maximum and minimum reported torque values as seen in Figure 3.12.

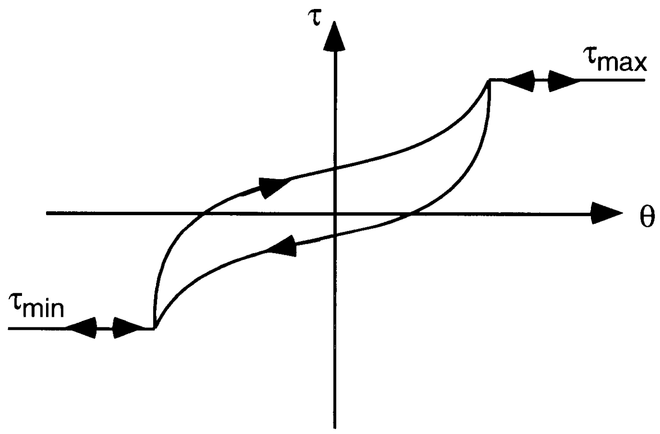


Figure 3.12 Angle versus torque curve for a space suit joint, with saturation limits.

Additionally, the direction of travel (clockwise) is opposite from that of the traditional hysteresis model (recall Figure 3.6). Thus the "up" switching threshold α is less than the "down" switching threshold β for all operators $\hat{\gamma}_{\alpha\beta}$, and the triangle T lies on the half-plane $\alpha \leq \beta$ as shown in Figure 3.13.

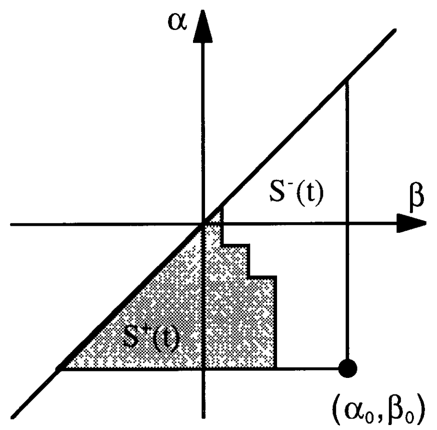


Figure 3.13 EMU-modified half-plane $\alpha \leq \beta$.

Having explored the Preisach model defined by equation 3.1, the only remaining unknown is how to obtain the Preisach weighting function $\mu(\alpha,\beta)$ for numerical implementation.

3.1.3.4 The Preisach Function

The Preisach weighting function $\mu(\alpha,\beta)$ is an essential part of equation 3.2, and must be defined by experimental hysteresis data. It has been shown [Mayergoyz, 1991] that the set of first-order transition curves for a hysteresis nonlinearity, shown in Figure 3.14, is sufficient to define the weighting function; these are the curves formed by the first reversal of input direction after a monotonic increase from below the negative saturation point.

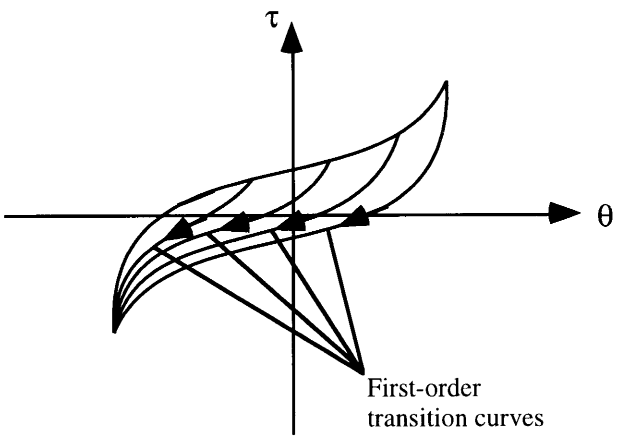


Figure 3.14 First-order transition curves.
First order transition curves are formed as the first branch of a hysteresis nonlinearity following monotonic increase of the input.

These first-order transition curves are converted to weights for $\mu(\alpha,\beta)$ by comparing their incremental changes to the changes taking place on the α–β diagram, as shown in Figure 3.15.

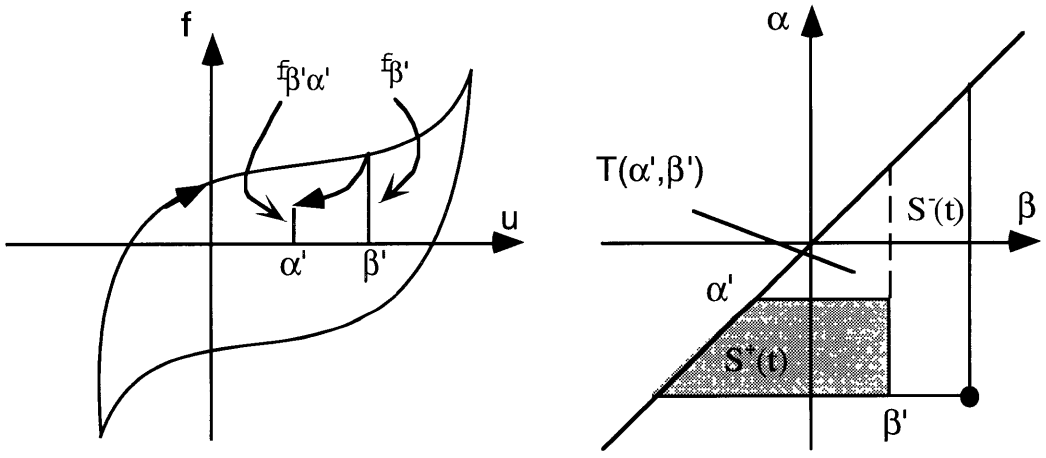


Figure 3.15 The *f-u* plane (left) and α–β plane (right) during a first-order reversal.

As the input increases from the negative saturation limit, a single vertical link moving from left to right is formed on the α–β diagram, creating a region $S^{+}(t)$ in which the operators $\hat{\gamma}_{\alpha\beta}$ have been switched to the "up" position. When the input changes direction at $u = \beta'$ and creates the first-order transition branch shown on the left of Figure 3.15, the vertical link on the α–β diagram stops moving and a new horizontal link is created at $\alpha = \beta = \beta'$, moving down along the α axis. Switching operators to their "down" states as it moves, the horizontal link sweeps through triangle $T(\alpha', \beta')$ until it reaches the current time at $u = \alpha'$. During this last segment, output $f(t)$ has decreased from $f_{\beta'}$ to $f_{\beta'\alpha'}$ while triangle $T(\alpha', \beta')$ was subtracted from $S^+(t)$ and added to $S^-(t)$. Using equation 3.2, the difference in area between the triangles becomes

$$\begin{aligned} f_{\beta'\alpha'} - f_{\beta'} &= \left(\iint\limits_{S^+(t_2)} \mu(\alpha,\beta)d\alpha d\beta - \iint\limits_{S^-(t_2)} \mu(\alpha,\beta)d\alpha d\beta \right) - \left(\iint\limits_{S^+(t_1)} \mu(\alpha,\beta)d\alpha d\beta - \iint\limits_{S^-(t_1)} \mu(\alpha,\beta)d\alpha d\beta \right) \\ &= -2 \iint\limits_{T(\alpha',\beta')} \mu(\alpha,\beta)d\alpha d\beta \end{aligned} \tag{3.3}$$

and using the definition

$$F(\alpha',\beta') = \frac{1}{2}(f_{\beta'} - f_{\beta'\alpha'}), \tag{3.4}$$

then

$$F(\alpha',\beta') = \iint\limits_{T(a',\beta')} \mu(\alpha,\beta)d\alpha d\beta. \tag{3.5}$$

Thus the integral of $\mu(\alpha,\beta)$ is known within the triangle $T(\alpha',\beta')$, corresponding to a simple difference in experimentally measured output $f(t)$ along a first-order transition curve. A finite number of first-order reversal curves will define a finite set of triangles $T(\alpha',\beta')$ whose integral area is known, and interpolation between triangles will then facilitate integration over an arbitrary area $S^+(t)$, shown below.

Region $S^+(t)$ is first subdivided into multiple trapezoids (Figure 3.16) whose integral area is easily obtained by differences of the smaller triangles $T(\alpha',\beta')$.

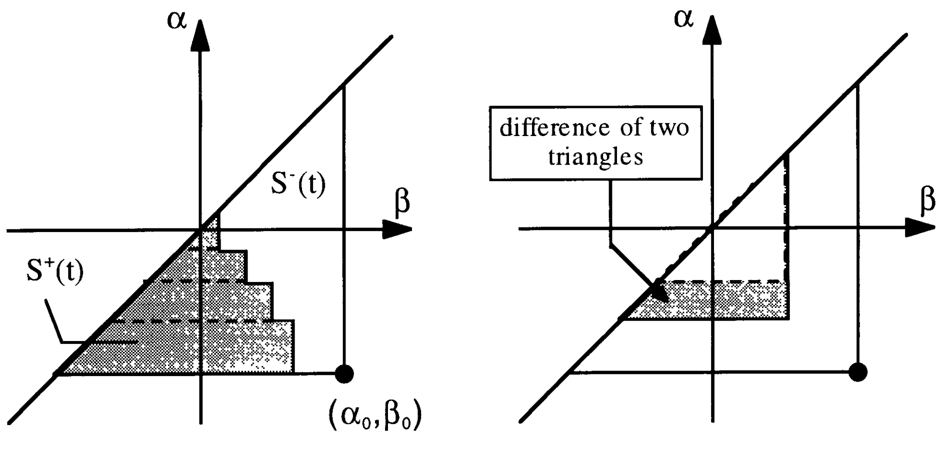


Figure 3.16 Subdivision of $S^+(t)$ into multiple trapezoids.

If there are n trapezoids designated Q_k, then the integral over $S^+(t)$ becomes

$$\iint_{S^+(t)} \mu(\alpha,\beta)d\alpha d\beta = \sum_{k=1}^{n(t)} \iint_{Q_k(t)} \mu(\alpha,\beta)d\alpha d\beta, \tag{3.6}$$

where n and Q_k will change with time as the interface $I(t)$ changes with input variations. Letting M_k and m_k represent the k maxima and minima, respectively, which appear on the α–β diagram, each vertex of the interface $I(t)$ can be assigned coordinates in terms of these input extrema (Figure 3.11). Trapezoid integrals then become

$$\iint_{Q_k(t)} \mu(\alpha,\beta)d\alpha d\beta = \iint_{T(M_k,m_{k-1})} \mu(\alpha,\beta)d\alpha d\beta - \iint_{T(M_k,m_k)} \mu(\alpha,\beta)d\alpha d\beta, \tag{3.7}$$

and from (3.5), each triangle integral is related to a simple difference along a first-order transition curve:

$$\iint_{T(M_k,m_{k-1})} \mu(\alpha,\beta)d\alpha d\beta = F(M_k,m_{k-1}). \tag{3.8}$$

To accumulate $f(t)$, add and subtract the integral $S^+(t)$ from (3.2) to obtain

$$f(t) = -\iint_T \mu(\alpha,\beta)d\alpha d\beta + 2\iint_{S^+(t)} \mu(\alpha,\beta)d\alpha d\beta, \tag{3.9}$$

where the leftmost integral is simply the outer loop hysteresis curve, represented by $F(\alpha_0,\beta_0)$. Then the total output at any time t is calculated by inserting the piecewise integration (3.6) into (3.9):

$$\begin{aligned} f(t) &= -F(\alpha_0,\beta_0) \\ &+2\sum_{k=1}^{n(t)-1}\left[F(M_k,m_{k-1})-F(M_k,m_k)\right]+2\left[F(M_n,m_{n-1})-F(M_n,u(t))\right] \end{aligned} \tag{3.10}$$

where the final term is simply the most recent trapezoid, defined on top by the line $\alpha=u(t)$. Equation (3.10) assumes that the current input is decreasing, and that the final link on the α–β diagram is horizontal. For increasing input, the last link on the α–β diagram will be vertical and the final term should be replaced by $2F(u(t),m_{n-1})$, to account for the final triangle-shaped region in $S^+(t)$.

3.1.3.5 Numerical Implementation of the Hysteresis Model

Exact implementation of (3.10) demands that a value of $F(\alpha,\beta)$ be available for every point in T, so that any combination $F(M_k,m_{k-1})$ can be represented. As Figure 3.13 and equation (3.4) illustrate, however, each small triangle in T and its corresponding $F(\alpha,\beta)$ are derived from a specific difference along a first-order transition curve. For complete coverage, an infinite set of first-order transition curves would have to be obtained experimentally. This is clearly a practical impossibility, so the continuous region T is instead discretized using data from a few experimentally measured first-order transition curves, creating a mesh of nodes between which values of $F(\alpha,\beta)$ can be interpolated (Figure 3.17).

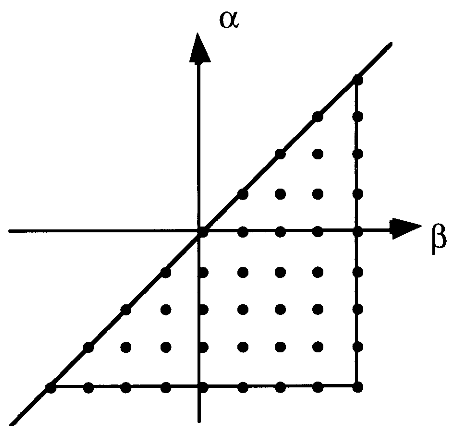


Figure 3.17 The discretized region *T*.

The experimental observations on which the EMU model is based, however, consist of a single major hysteresis loop (refer to Figure 3.1) for each joint. The single descending branch is considered a first-order transition curve, but a complete family of curves is necessary to properly discretize the region and generate a reasonable model.

To estimate the full set of transition curves necessary for the EMU model, the following property of the Preisach model is invoked (cf. Figure 3.18):

$$\mu(\alpha', \beta') = \frac{1}{2} \frac{\partial \tan\theta(\alpha', \beta')}{\partial \alpha'}. \tag{3.11}$$

The angle $\theta(\alpha',\beta')$ is formed between the horizontal and a tangent at $u = \alpha'$ to the first-order reversal curve originating at β' . From equation 3.11, the Preisach function $\mu(\alpha',\beta')$ is positive, and therefore valid, if tan $\theta(\alpha',\beta')$ is a monotonically increasing function of α' for a fixed β'. Looking at a set of first-order transition curves evaluated at the same point $u = \alpha'$, the curves' first-order derivatives should increase with β', their starting location. In general, this means that their slopes should increase from the bottom (major-loop) transition curve upwards to the upper major-loop branch.

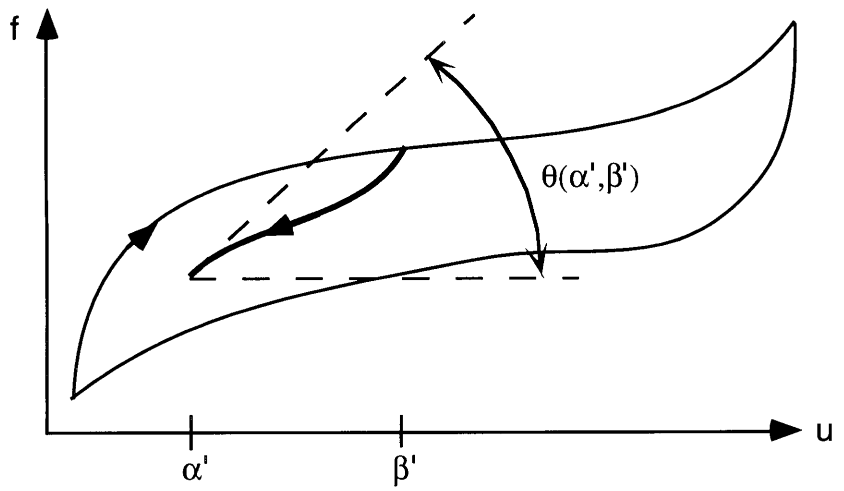


Figure 3.18 The angle ***θ(α',β')*****.**

Using the constraint that (3.11) places on transition curve derivatives, an algorithm was developed to create a set of *k* first-order transition curves based on the available EMU data, i.e., the shape of the outermost descending branch:

$$\frac{dx_k}{d\alpha'} = \frac{dx_{k-1}}{d\alpha'} + g, \qquad (3.12)$$

where curve x_0 (at $k = 0$) is simply the major descending curve, and g is the "grouping" factor, representing the relative distance between curves. Figure 3.19 shows the family of curves created by a MATLAB script (Appendix A) which implements this algorithm. It should be emphasized that these curves are educated guesses for the form of the EMU hysteresis nonlinearity, based solely on common patterns observed throughout a variety of hysteretic systems. A complete model would require a full set of first-order transition curves to be experimentally measured.

Having obtained a full set of first-order transition curves, a surface T is constructed for the evaluation of *f(t)* by (3.10). Each curve is first discretized into nodes separated by 2 degree intervals, and each node is assigned its value of $f_{\alpha\beta}$ ($\tau_{\alpha\beta}$ for the EMU model). The nodes are used to create a mesh of cells over triangle T, where each vertex is associated with a specific value of $f_{\alpha\beta}$. Over the course of the simulation, a history of extrema $\{M_k, m_k\}$ are collected, defining the interface *I(t)*.

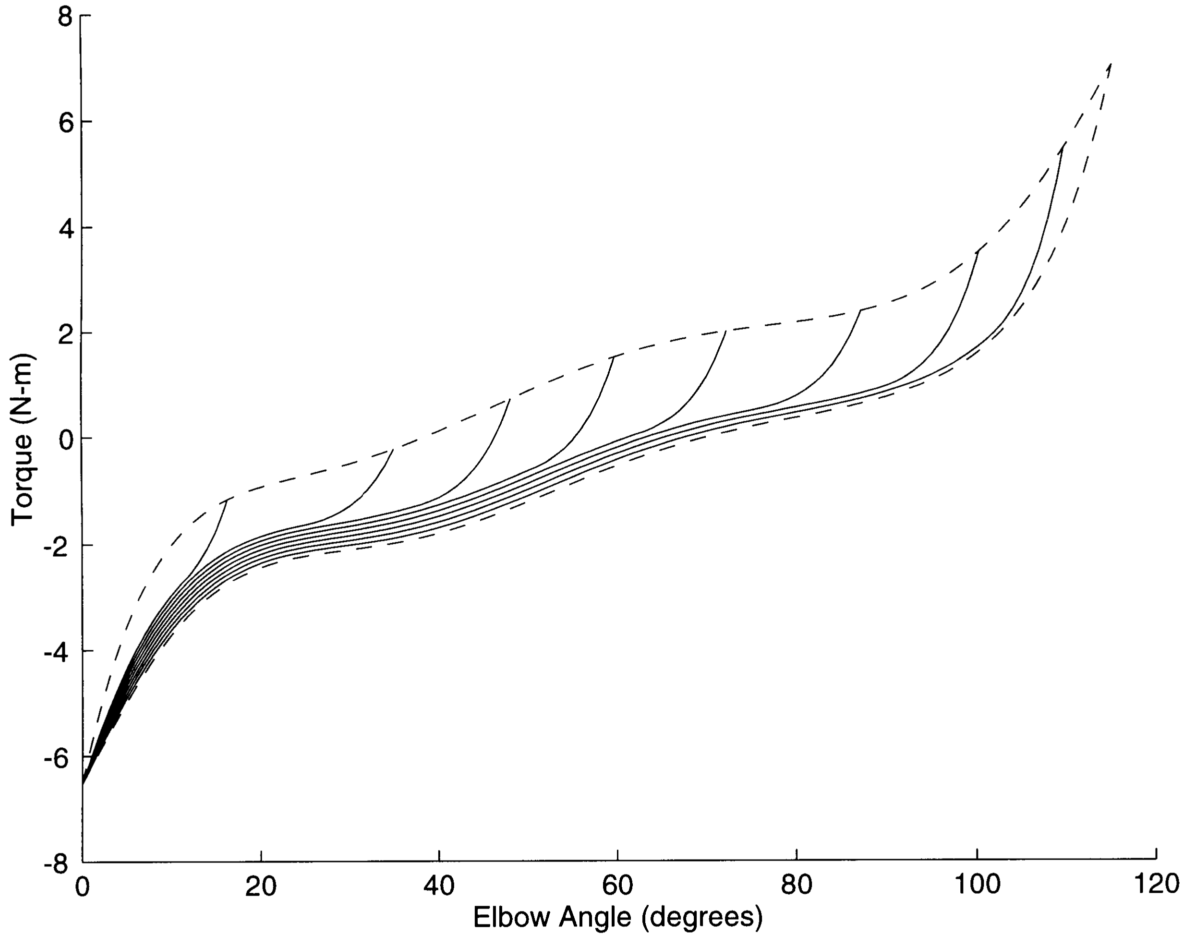


Figure 3.19 The complete set of first-order transition curves for the elbow joint.

To calculate (3.10), each point (M_k,m_k) and (M_k,m_{k-1}) is dropped through a search tree to determine the cell in T to which it belongs. The value of $F(M_k,m_k)$ is then computed by interpolating between the four (or three) vertices of the cell (Figure 3.20).

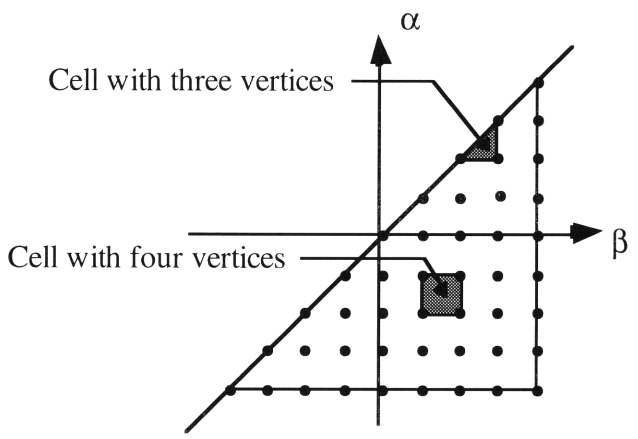


Figure 3.20 Cells bounded by either three or four vertices.

For a four-sided cell, the interpolation used is

$$f_{\alpha\beta} = A + B\alpha + C\beta + D\alpha\beta, \tag{3.13}$$

and a three-sided, triangular cell uses

$$f_{\alpha\beta} = A + B\alpha + C\beta. \tag{3.14}$$

In order to expedite computation of (3.13) and (3.14), coefficients A, B, C and D can be precomputed as the solution to the linear system

$$\mathbf{f}_{\alpha\beta} = \mathbf{X}\mathbf{k}, \tag{3.15}$$

with

$$\mathbf{f}_{\alpha\beta} = \begin{bmatrix} f_{\alpha_1\beta_1} \\ f_{\alpha_2\beta_2} \\ f_{\alpha_3\beta_3} \\ f_{\alpha_4\beta_4} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & \alpha_1 & \beta_1 & \alpha_1\beta_1 \\ 1 & \alpha_2 & \beta_2 & \alpha_2\beta_2 \\ 1 & \alpha_3 & \beta_3 & \alpha_3\beta_3 \\ 1 & \alpha_4 & \beta_4 & \alpha_4\beta_4 \end{bmatrix} \qquad \mathbf{k} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix},$$

because the vertices (α_k, β_k) and their values $f_{\alpha\beta}$ are known. Coefficients A-D are computed for each cell and stored along with the cell location, obviating actual storage of the values $f_{\alpha\beta}$.

3.2 Human model

This section introduces the human model that was used as a base to simulate the astronaut. The skeletal model is discussed first, followed by a brief survey of potential muscle models.

3.2.1 Skeletal Model

The human skeletal model used for these simulations is a 12-segment, 28 degree of freedom (d.o.f) system (Figure 3.21).

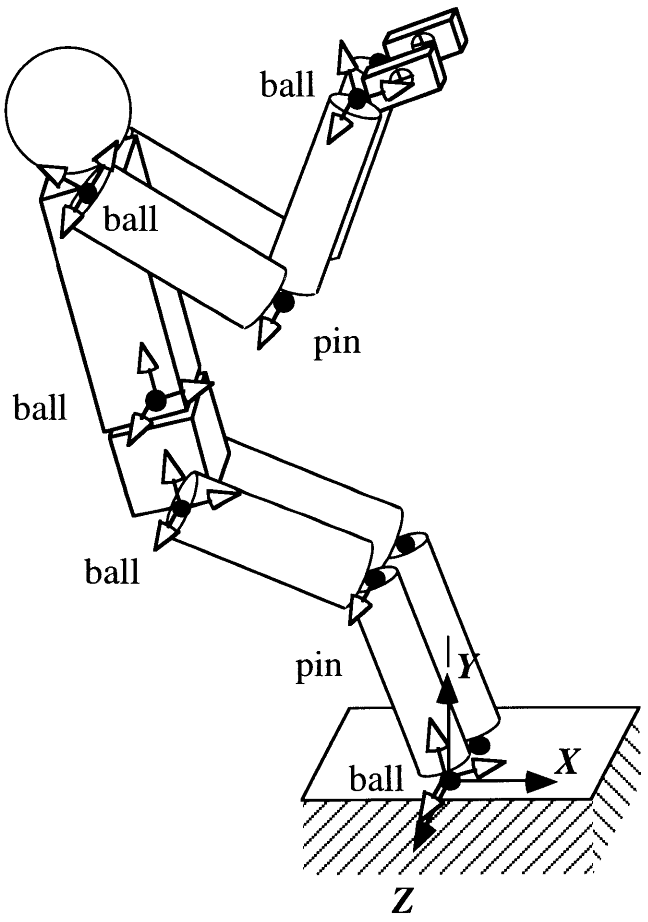


Figure 3.21 Human skeletal model.

The ankle, hip, wrist, and shoulder joints are modeled with 3 d.o.f. ball joints, and the knee and elbow joints are modeled using 1 d.o.f. pin joints. The neck is welded to the upper torso (no relative motion is allowed), and the entire system is assumed to be fixed at the ankle joint to an inertial reference frame (the foot restraint). For these simulations, the EMU joint torque is only applied in a sagittal plane of motion (z-axis torque), and therefore the ball joints will be restricted to a single degree of rotational freedom. Inertial properties for the skeletal segments are from a program written by K. Jackson, based on a 50th percentile man.

3.2.2 Muscle Models

An extremely simple muscle model was chosen to provide human joint dynamics for the astronaut model. Each joint is modeled as a pure torque generator, actuating the limb directly over a fixed range of motion. This model has been used to observe macro-level effects [Benati, 1980; Crowninshield, 1981], but it is not assumed to represent muscle dynamic parameters in a true sense. A basic viscoelastic model is added for passive joint control in Section 3.3, below, which adds joint stiffness and damping to the musculoskeletal model but neglects key muscle parameters such as the length-velocity-tension relationship at the contractile element (cf. Chapter Two). Several muscle models are discussed below, however, for reference and for future implementation.

Selecting a muscle model suitable for the simulation framework depends largely on the required output, and is necessarily task-specific, as discussed in Chapter Two. The system can be simplified to widely varying levels of detail, according to the required physiological accuracy of the investigation. For example, forward dynamic simulation of muscle reflex activity would demand provisions for a complete neural control feedback model and neural excitation dynamics, in addition to the basic Hill muscle parameters. Inverse dynamic simulation of a larger-scale system, on the other hand, might be interested only in the basic workload levels associated with a particular task; in this case, the neural control model might add excess complexity not required for the system under consideration. This thesis is concerned more with the latter condition, in which neural control patterns are interesting but unnecessary for the stated objectives.

The muscle model used in these inverse-dynamic simulations should trace the neuromusculoskeletal system back to the muscle forces or joint torques required to generate movement, but it should not be overly concerned with control aspects on a neural level. Two models that meet this criterion will be discussed briefly, below. Although the simulations presented in this thesis do not include an explicit implementation of one of the following muscle models, future work should address these issues more thoroughly.

3.2.2.1 Torsional Equivalent

The torsional equivalent muscle model is possibly the most basic approach to muscle modeling on a joint-by-joint basis. Although simpler models exist for larger, lumped subsystems (cf. the two-parameter lower-body model in [Jackson, 1997]), EMU data has been obtained on a per-joint basis, encouraging a similar structure for the underlying human model.

The simplest torsional model is the pure torque generator, as implemented in this thesis. For larger-scale systems, the somewhat questionable assumption can be made that inertial properties dominate, and viscoelastic influences are either negligible or linear [Crowninshield, 1981]. Then muscles can be approximated as pure torque generators, operating directly on the limbs of interest. Though simple and easily implemented, the nonlinear filtering properties of muscle are ignored [Winters, 1985], and model parameters must be modified for each specific task. Two lumped-parameter, torsional muscle equivalents that improve slightly upon the pure torque generator are shown in Figure 3.22.

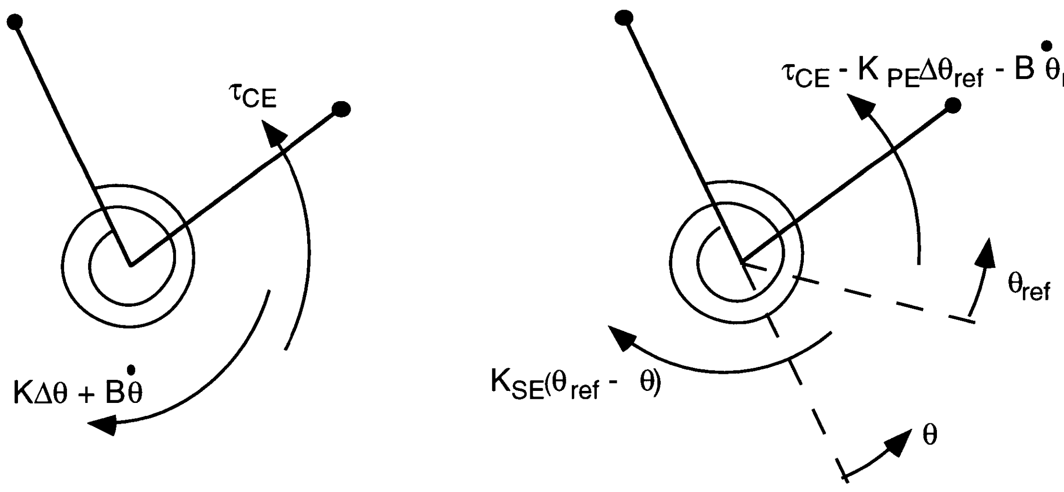

τ_{CE}
$K\Delta\theta + B\dot{\theta}$
$\tau_{CE} - K_{PE}\Delta\theta_{ref} - B\dot{\theta}_{ref}$
θ_{ref}
$K_{SE}(\theta_{ref} - \theta)$
θ

Figure 3.22 Torsional equivalent muscle models.

The model on the left is a simple two-parameter model that provides a torque-generating element in series with a damped elastic element. For convenience, the stiffness and damping constants have been indicated as linear parameters, although it would be equally suitable to have $K(\theta,\dot{\theta})$ or $B(\theta,\dot{\theta})$. The right-hand model differentiates the SE and PE components of the classic Hill model, while maintaining a torque-based approach. These models approximate the stiffness and damping properties of a joint, but fail to include the geometric and dynamic asymmetries characteristic of antagonist-pair muscle models.

3.2.2.2 Lumped Muscle Models

More complex than the torsional model, yet a computationally efficient alternative to representing muscles individually is the lumped-muscle model, in which synergistic muscles are grouped together. This type of model has been used extensively throughout the biomechanical community, and produces interesting results without undue complexity [Winters, 1985].

The simplest form is shown in Figure 3.23, where all flexors have been lumped into a single "equivalent flexor," or agonist, and all extensors have been lumped into a corresponding "equivalent extensor," or antagonist. There are now only two muscles to simulate at this joint, replacing what could be more than 10 synergistic and stabilizing muscles at a true physiological joint.

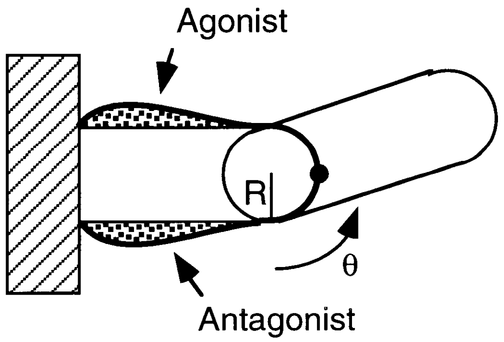


Figure 3.23 Agonist-antagonist lumped-muscle model.

Cocontraction of the extensor and flexor muscles sets a specific joint stiffness, and differential contraction produces a net joint torque. Each of these two muscles can then be represented in the

Hill form (cf. Figure 2.4) as a linear or nonlinear combination of CE, PE, and SE elements (Figure 3.24). The lumped-muscle parameters K_{SE}, K_{PE}, and B may be linear or nonlinear, and they are constructed from experimental observation, effectively combining the characteristics of several synergistic muscles.

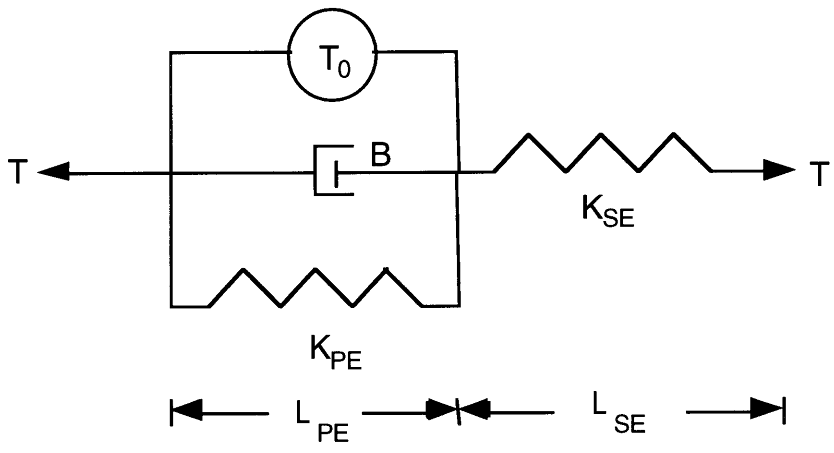


Figure 3.24 Schematic representation of a Hill muscle model.

This model does not account for the precise contribution from each muscle, however, and can suffer if applied to precise movements under close scrutiny. Developing models for the neural excitation of synergistic muscle groups, for example, is not possible with a lumped-muscle model. In this case, a separate muscle model for each synergist [Happee, 1995] would be required to explore interactions between cocontracting muscles.

3.3 Simulation Method

The suit model was demonstrated as a constraint on the existing 12-segment human model, which was created during preliminary simulation work by G. Schaffner (1995). The system allows a given EVA task to be described in terms of hand trajectories, which are translated by the SD/FAST inverse kinematic solver into a history of joint rotations required to perform the task. Inverse dynamics routines then determine the torque required at each joint to produce the necessary joint rotations. The resulting time history of joint torques is used to compute the work done at each joint during the execution of a specific task.

3.3.1 SD/FAST Framework

The equations of motion for a 12-segment linked mass model are highly complex, and are ill-suited for analytical derivation; therefore, the SD/FAST dynamics package from Symbolic Dynamics (CA) was used to derive the system equations of motion using Kane's formulation [Kane, 1985]. The simulation procedure is shown in Figure 3.25.

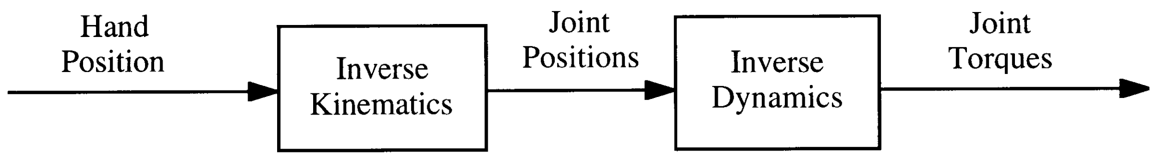


Figure 3.25 Simulation flow diagram.

Hand position, velocity, and acceleration are prescribed in Cartesian coordinates; inverse kinematics routines then solve for joint trajectories that are compatible with the given Cartesian motion. The resulting time history of joint positions (in joint coordinates) is then delivered to an inverse dynamics routine, solving for the joint torques required to produce the observed kinematic data. This framework was used to execute the payload translation task described below, and the simulation code is included in Appendix E.

3.3.2 Payload Translation Task

Two simulations of an astronaut manipulating the 1,200 kg Spartan astronomy payload were performed; each simulation was performed under two separate conditions: unsuited, to determine the baseline work level required to move the payload, and suited, incorporating the EMU suit model.

Both simulations involve an astronaut pulling the Spartan payload directly forward, after grasping it in the far range of his workspace (Figure 3.25) This task is easily extensible to other similar operations, such as extracting a module from the Hubble Space Telescope.

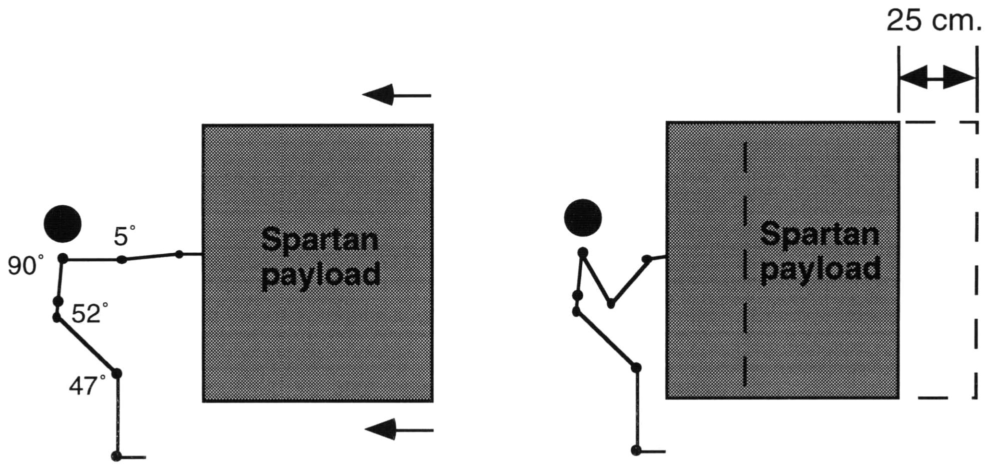


Figure 3.26 Spartan payload translation task.

The center of mass (CM) of each hand is connected to the edge of the payload by a weld joint; this ensures that the payload follows the hand, remaining attached to the hand segment. The wrist is given a much greater range of motion than in previous simulations (±70°); this simulates the effect of extending and flexing different fingers to rotate the astronaut's grip while maintaining a relatively constant wrist angle. A highly flexible wrist joint is essential because the musculoskeletal model does not include fingers. The feet are anchored to an inertially fixed foot restraint; movement of the foot restraint such as might be experienced on the end of the remote manipulator arm, is not included. Vibrational effects from the arm would clearly be important, however, and should be addressed in future, more detailed simulations.

The astronaut was required to move the Spartan payload 25 cm along a linear trajectory at 0° with respect to inertial space over a time interval of 10 seconds. To create interesting acceleration and torque profiles, as well as to generate a smooth motion of the payload, the CM of each hand was prescribed to move along this trajectory with a sinusoidal velocity profile. The equations used to generate the movement were:

$$\begin{aligned} X &= At + \left(\frac{A\mathrm{T}}{2\pi}\right)\sin\left(\frac{2\pi}{\mathrm{T}}t + \pi\right) \\ \dot{X} &= A + A\cos\left(\frac{2\pi}{\mathrm{T}}t + \pi\right) \qquad , \\ \ddot{X} &= \left(-\frac{2\pi A}{\mathrm{T}}\right)\sin\left(\frac{2\pi}{\mathrm{T}}t + \pi\right) \end{aligned} \tag{3.16}$$

where A = 0.025 m/s and T = 10.0 s. The crewmember's initial lower-body posture coincides roughly with human zero-gravity neutral posture [NASA, 1991], and the upper-body position is well within the astronaut's predicted workspace envelope [Reinhardt, 1989].

3.3.2.1 Rigid Lower Body

For the first simulation, the astronaut's lower body and torso were locked in a rigid position, forcing the task to be executed by arm movement alone. Only the shoulder, elbow and wrist joints were allowed complete freedom of movement in three, one, and three degrees of freedom, respectively . Passive dynamic control was introduced at the shoulder, elbow, and wrist joints to simulate damped control and relative muscle strength using the relation

$$\tau_{\text{joint}} = -k_{\text{rot}}(q_{\text{joint}} - q_{\text{bias}}) - b_{\text{damping}}\dot{q}_{\text{joint}}, \qquad (3.16)$$

allowing for joint stiffness k and neutral position q_{bias}. Bias positions for joints are the angle between the axis of the proximal segment and the neutral position at which the distal segment will rest motionless in zero gravity (Figure 3.26), and joint stiffnesses k are based on Winters and Stark (1985).

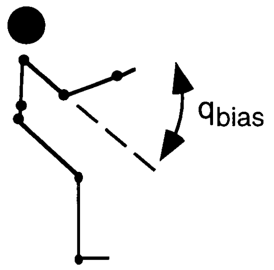


Figure 3.27 Bias position q_{bias} for the elbow joint.

The rigid lower body simulation was performed under two conditions: first, with passive musculoskeletal dynamics alone in an "unsuited" condition, and second, in a "suited" condition to demonstrate the effect of the suit on astronaut workload. The task was identical for both conditions, but the second condition added the dynamic properties developed above for the EMU space suit: mass, inertia, and imposed torques at the shoulder, elbow, and knee joints. The ankle joint was provided with a first-order torque, simply a linear extension of the average torque given in the NASA STD-3000, as ankle joint suit data was unavailable.

3.3.2.2 Compliant Lower Body

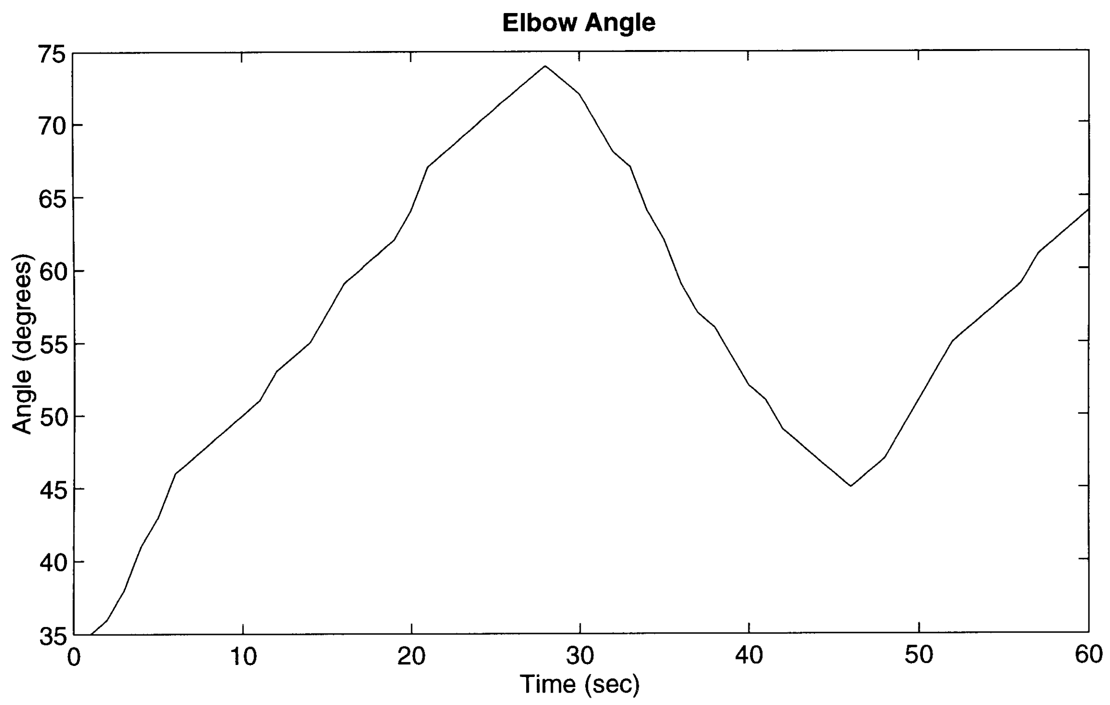
The second simulation then introduced a compliant lower body, allowing full motion in all joints. Passive dynamic control according to (3.16) was included at the hip, knee, and ankle joints, and suit effects at the knee were implemented. The task was identical to the first simulation, prescribing the hand CM to move along the same linear trajectory, but the astronaut was allowed to "choose" the posture from which to perform the task by introducing a compliant lower body. Again, the simulation was performed twice, under "suited" and "unsuited" conditions that differed only by the application of space suit dynamics.

4. Results

This chapter presents the results of the hysteresis model test simulation and two payload translation simulations. Astronaut position during each simulation is displayed in terms of individual joint angles, and sequential diagrams of crewmember position are included to facilitate interpretation of the angle data. Torques at each joint are reported, and total cumulative work is calculated from these torques and their corresponding joint velocities.

4.1 Hysteresis Model Test

Results of a test simulation for the hysteresis model at the suit elbow joint are shown in Figure 4.1. The upper trace plots elbow joint angle as a function of time, starting at 35 degrees away from full extension and flexing through 74 degrees, before returning to greater extension. The lower plot represents suit elbow torque over this period, plotting torque versus angle for the duration of the motion. Note the characteristic pattern of hysteresis observed as the elbow is flexed and then extended. Figure 4.2 illustrates how equation 3.1 is used to determine the space suit elbow torque during this test; the interface $I(t)$ is shown plotted on the surface $f(\alpha, \beta)$, with the shaded area representing the area $S^{+}(t)$ and the lighter area $S^{-}(t)$. The large arrow indicates the motion of the current input, increasing along the α axis. If the input increases past the previous maximum at 74°, the previous vertices of $I(t)$ will be eliminated (the "wiping-out" property) and the interface will be the single line $\alpha = u(t)$.



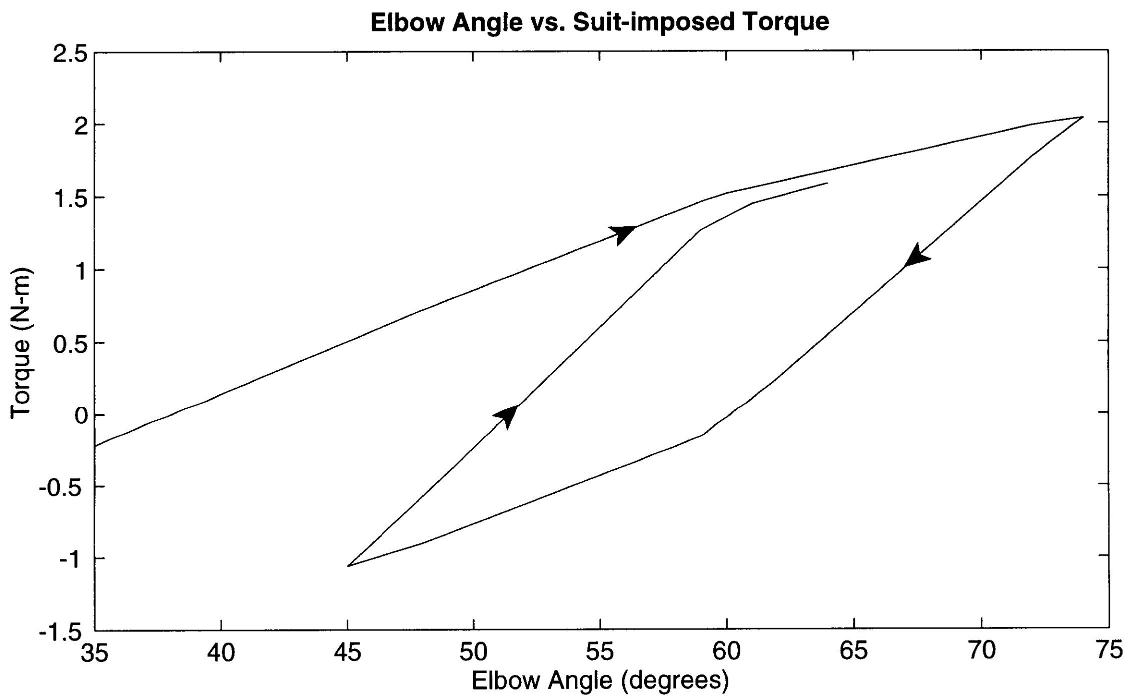


Figure 4.1 Elbow joint trajectory and suit-imposed torque during model test.

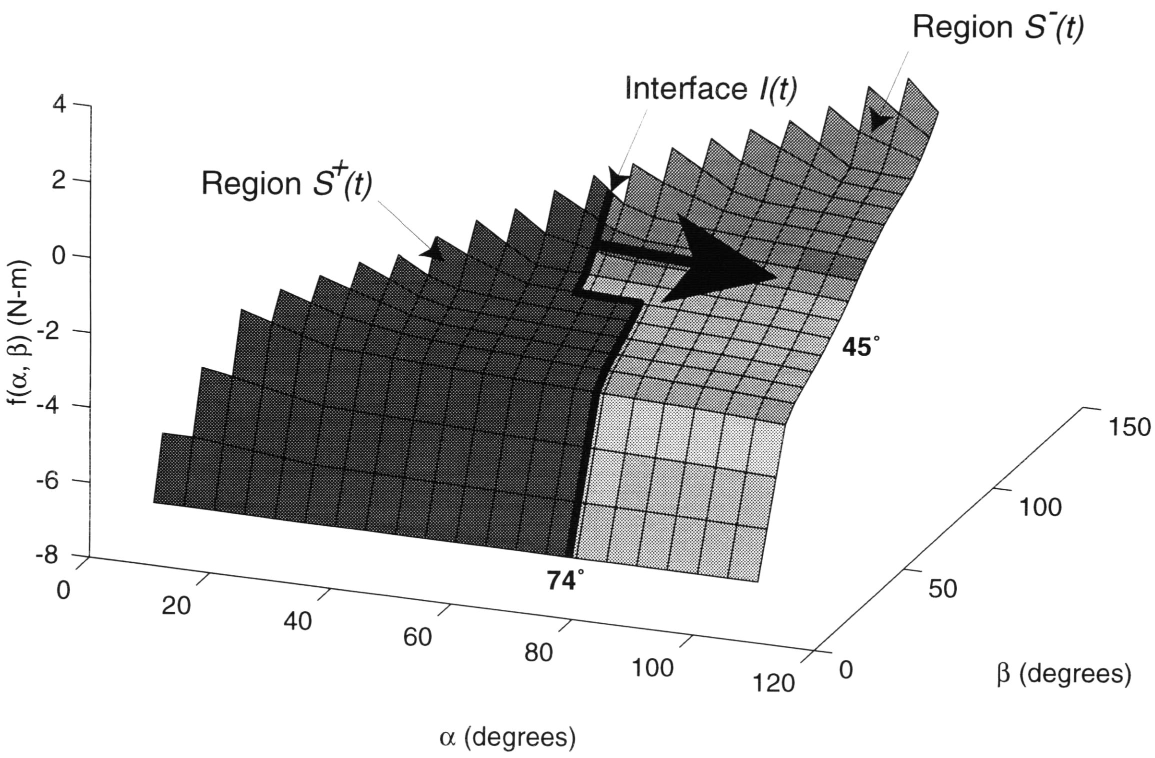


Figure 4.2 **The surface $f(\alpha,\beta)$ at t = 60 sec (elbow joint test).**
A value for *f*, the total torque applied by the EMU elbow joint, is calculated by integrating the shaded area based on equation 3.1. On this diagram, the first maximum was at 74° ($\alpha = 74°$), the first minimum was at 45° ($\beta = 45°$), and the current input is $u = \alpha = 64°$, moving up the α axis (refer to Figure 4.1).

4.2 Payload Translation Task

The following two sections present results for the payload translation task simulations, including both rigid and compliant architectures for the lower body. Each simulation is carried out under two conditions: unsuited, without space suit forces and torques, and suited, adding the EMU dynamic model to the simulation.

4.2.1 Rigid Lower Body

The sequence of position diagrams shown in Figure 4.3 qualitatively illustrates overall astronaut motion during the payload translation task with a rigid lower body, at 2 second intervals; position is identical for both the suited and unsuited conditions. Figure 4.4 then plots astronaut joint positions during the task. As anticipated, the ankle, knee, and hip joints remain motionless during the simulation, because they were prescribed to maintain a constant joint angle. The hands gradually pull the mass inward, causing rotation in the shoulder and elbow joints. The wrist rotates from an initial position in line with the forearm to 55° downward tilt; it is important to keep in mind that this 55° rotation includes both wrist position and the grip position of the fingers.

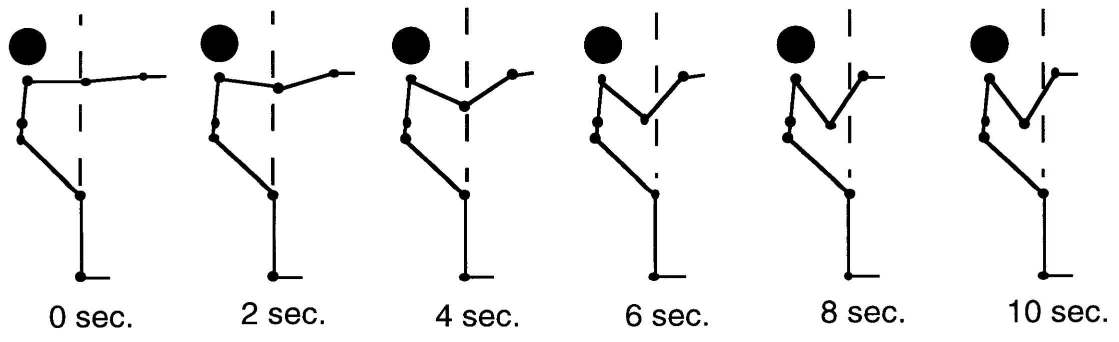


Figure 4.3 Sequential astronaut position during payload translation task, rigid lower body.

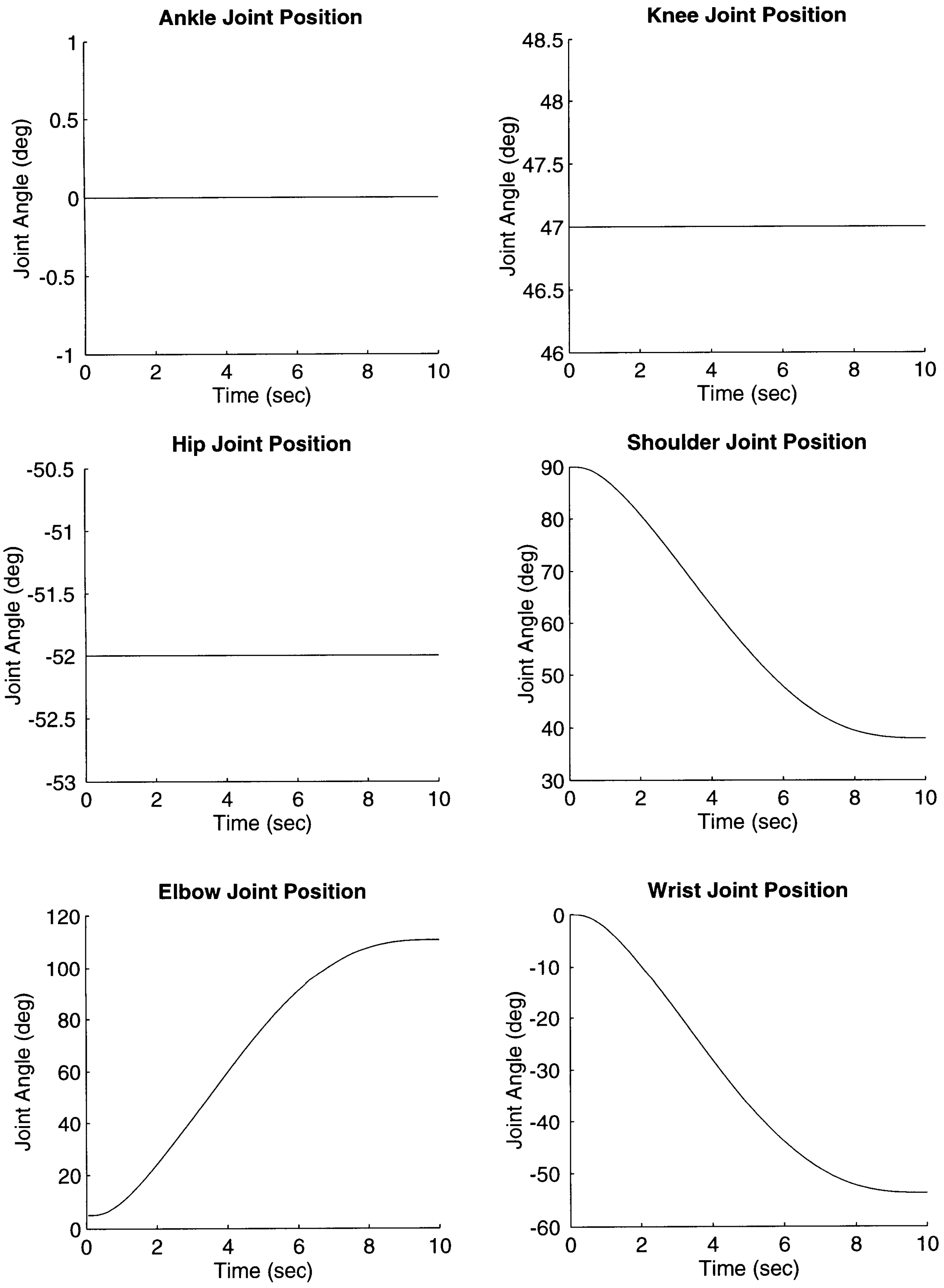


Figure 4.4. Joint positions during payload translation task, rigid lower body. Plots reflect both suited and unsuited trajectories (identical).

Inverse dynamic computation was then applied to the motions illustrated in Figures 4.3-4.4, recovering the joint torques required to produce the observed movement. These joint torques are shown in Figure 4.5, for both suited and unsuited conditions.

The ankle, knee and hip joints exhibit sinusoidal torque behavior throughout the motion. The wrist joint exerts no torque at all, but the shoulder and elbow joints exert small torques during this simulation. The unsuited shoulder joint produces an extremely small torque, varying between -0.25 Nm and 0.25 Nm, while the suited shoulder joint produces a much larger range of torques from -1.5 Nm to 8.7 Nm. Similarly, the unsuited elbow joint gives torque values ranging from -2.1 Nm to 0.9 Nm, while the suited elbow joint varies between -7.1 Nm and 4.9 Nm.

The torque values were then converted to instantaneous power using the relation

$$P = \tau\omega, \tag{4.1}$$

where the instantaneous power P at a joint is equal to the product of its current angular velocity ω and the torque it is producing. Power values were used to calculate the work performed at each joint by integrating, using

$$W = \int P dt. \tag{4.2}$$

Here the step size of integration is equal to the simulation step size, 0.05 seconds. Work at each joint is shown in Figure 4.6 as cumulative work, where the cumulative work at time t is represented by equation 4.2, evaluated over the interval [0.0, t].

The ankle, knee, hip, and wrist joints exhibit no net work performed, while the more active shoulder and elbow joints do appreciable work moving the payload (Table 4.1).

Table 4.1 Shoulder and elbow work, with rigid lower body.

Joint	**Unsuited Total Work (N-m)**	**Suited Total Work (N-m)**	**Difference (N-m)**	**Suited Premium**
Shoulder	0.16	4.40	4.25	2740 %
Elbow	1.37	4.25	2.88	210 %
Total	**1.53**	**8.65**	**7.12**	**467 %**

The suited astronaut with a rigid lower body does a 2740 % premium of work at the shoulder joint and 210 % more work at the elbow joint to produce motion identical to that of the unsuited astronaut. Overall, the crewmember performs almost 4.7 times more work in the suited condition than in the unsuited condition.

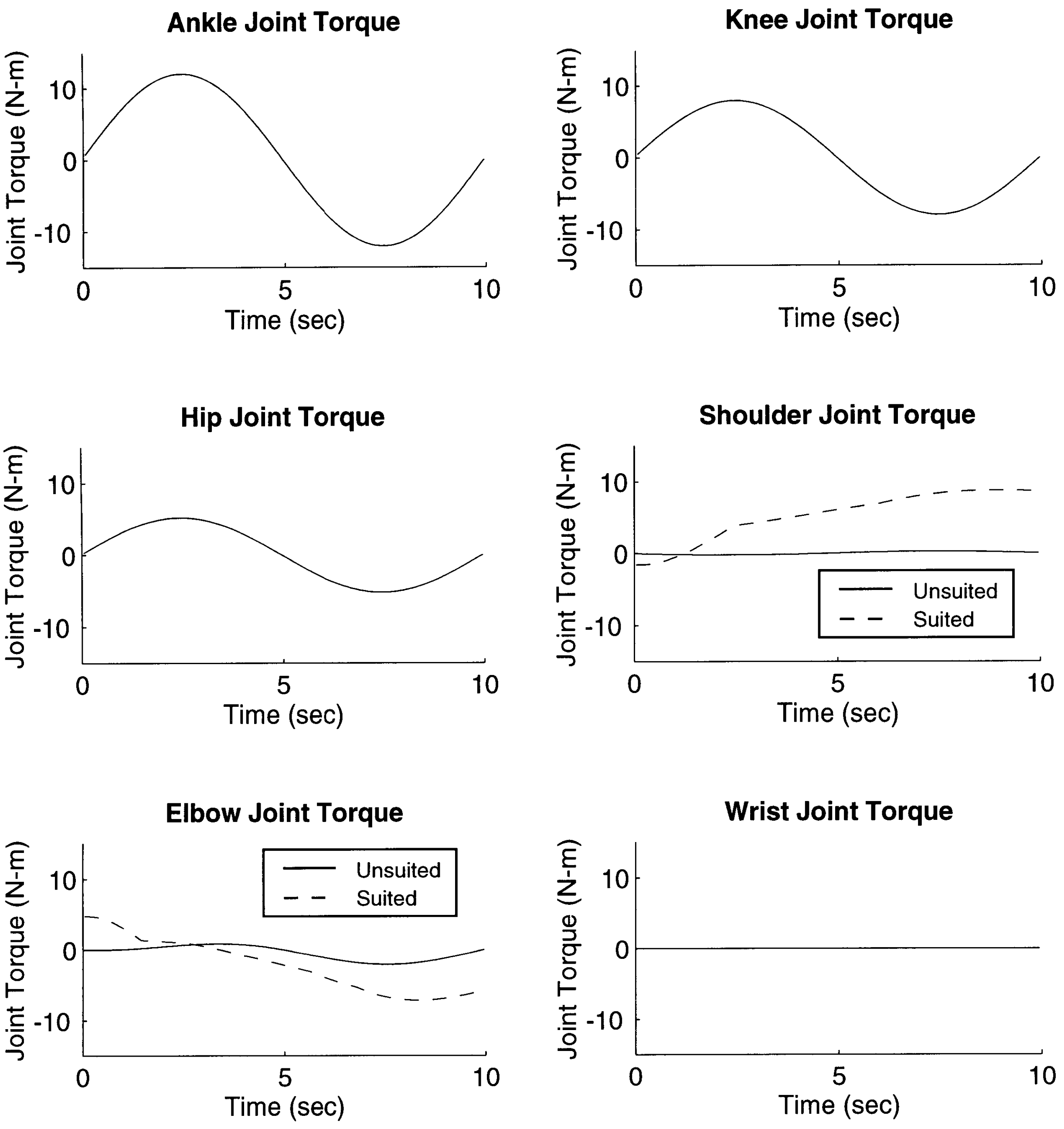


Figure 4.5. Joint torques for payload translation task, rigid lower body.
Single traces reflect torques for both suited and unsuited conditions (identical).

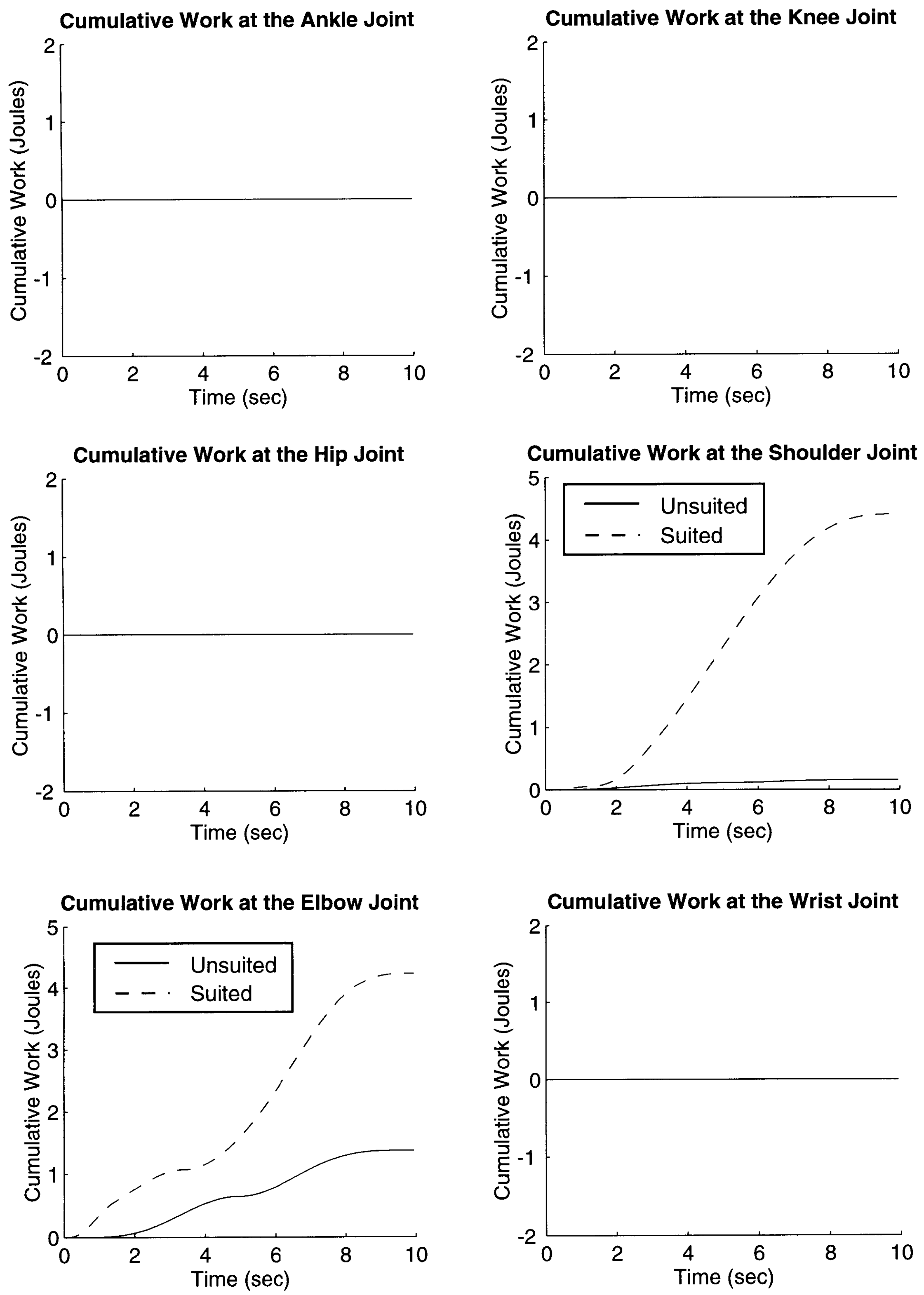


Figure 4.6. Cumulative joint work for payload translation task, rigid lower body. Single traces reflect work for both suited and unsuited conditions (identical).

4.2.2 Compliant Lower Body

In the real world, however, the astronaut would not be required to maintain a rigid lower body. The second simulation was therefore performed with a slight modification from the first: ankle, knee, and hip joints were allowed to move, and passive lower-body control was introduced in the form of joint stiffness and damping. Figure 4.7 shows two sequences of astronaut positions for the payload translation task with a compliant lower body, illustrating the difference in movement between suited and unsuited conditions; individual joint angle time histories are then given in Figure 4.8.

The knee and hip joints maintain relatively constant position throughout the task, the suited and unsuited conditions differing by less than 2°. The suited astronaut's ankle joint, however, finishes the task in plantar flexion, a full 12° greater than the unsuited ankle joint. The suited elbow and shoulder joints, on the other hand, finish in 80 and 40 degrees *less* flexion, respectively. The suited astronaut's wrist completes the task at 15° extension, compared to the unsuited astronaut's -55° extension; again, the wrist rotations allow for finger orientation and grip changes, giving a greater range of motion than the wrist joint alone.

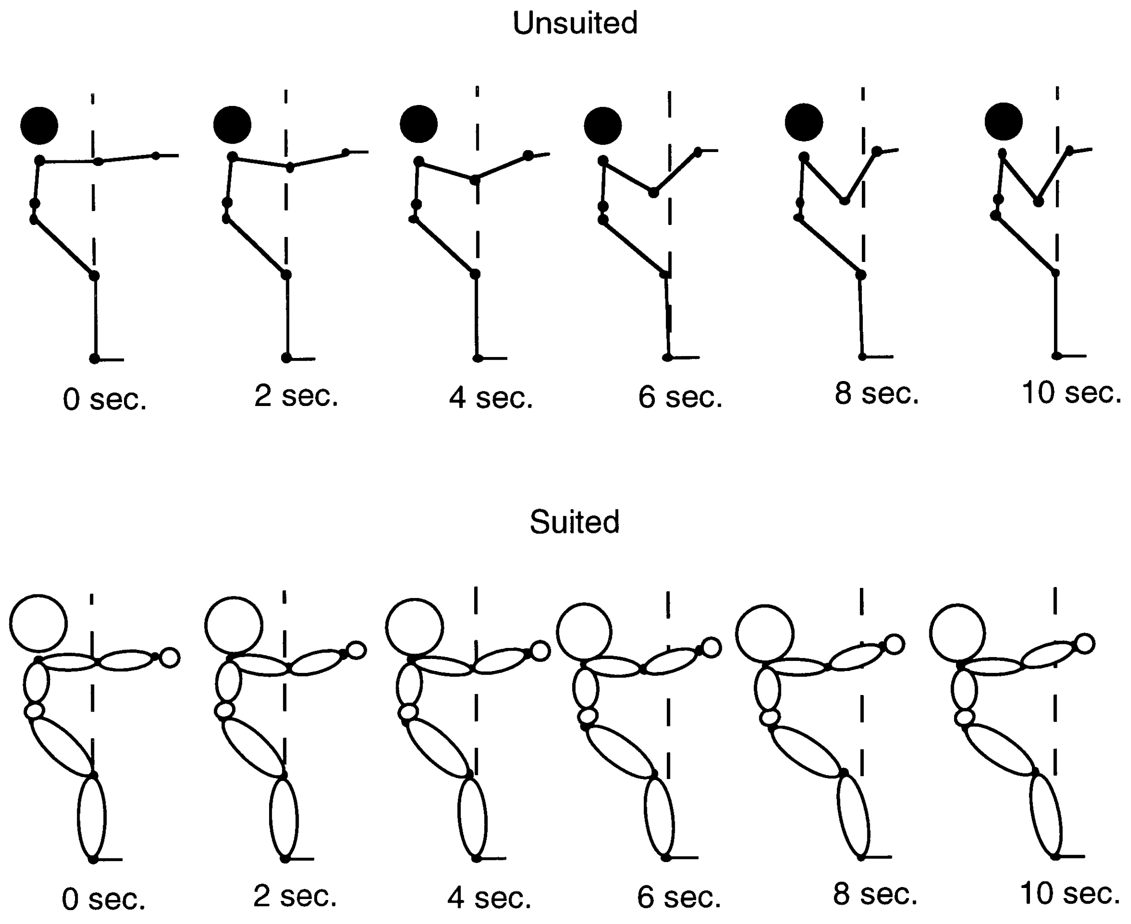


Figure 4.7 Astronaut position during payload translation task, compliant lower body.

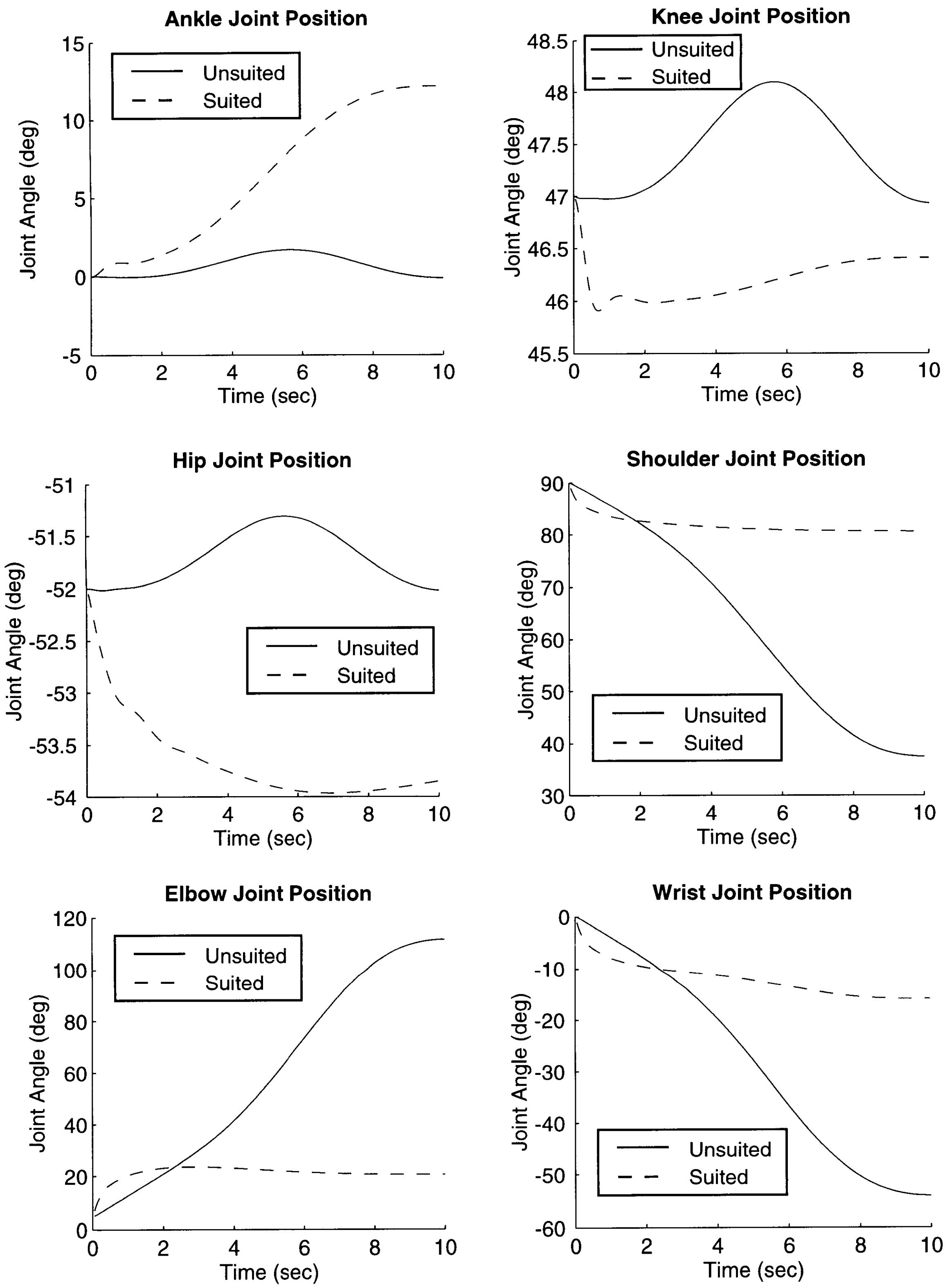


Figure 4.8. Joint positions during payload translation task, compliant lower body.

Inverse dynamic analysis recovers the joint torques required to perform the translation task with a compliant lower body; these torques are shown in Figure 4.9. Ankle, knee, and hip joints all show sinusoidal torque patterns, with the suited values being universally larger than their unsuited counterparts. Shoulder joint torque is slightly larger in the suited condition, and elbow joint torque is the only value larger in the unsuited condition.

Peak torque differential between the unsuited and suited conditions occurs during the payload deceleration phase at t = 8.05 sec; these values are tabulated in Table 4.2.

Table 4.2 Maximum torque differential between suited and unsuited values.

Joint	Larger Torque	Differential
Ankle	Suited	10.0 %
Knee	Suited	10.1 %
Hip	Suited	20.1 %
Shoulder	Suited	308 %
Elbow	Unsuited	189 %
Wrist	Neither	--

These torques were then used to calculate total work performed during the task using equations 4.1 and 4.2. Cumulative work time histories at individual joints are shown in Figure 4.10, and total work is presented in Table 4.3.

Table 4.3 Total work during payload translation task, compliant lower body.

Joint	Unsuited Total Work (N-m)	Suited Total Work (N-m)	Difference (larger condition)	Premium
Ankle	0.56	1.71	1.15 (suited)	207 %
Knee	0.24	0.11	0.13 (unsuited)	110 %
Hip	0.10	0.11	0.01 (suited)	9 %
Shoulder	0.12	0.02	0.10 (unsuited)	490 %
Elbow	1.53	0.06	1.47 (unsuited)	264 %
Wrist	0.0	0.0	--	--
Total	**2.55**	**2.01**	**0.54 (unsuited)**	**27 %**

The two joints with significantly higher work levels, a full order of magnitude larger than the rest of the joints (see Figure 4.10), are the ankle joint in the suited condition (1.7 Nm) and the elbow joint in the unsuited condition (1.5 Nm). The hip joint performs almost identical work, at a low level, for both conditions. The knee, shoulder, and elbow all perform greater work in the unsuited condition, while the ankle stands out as the high workload joint in the suited condition.

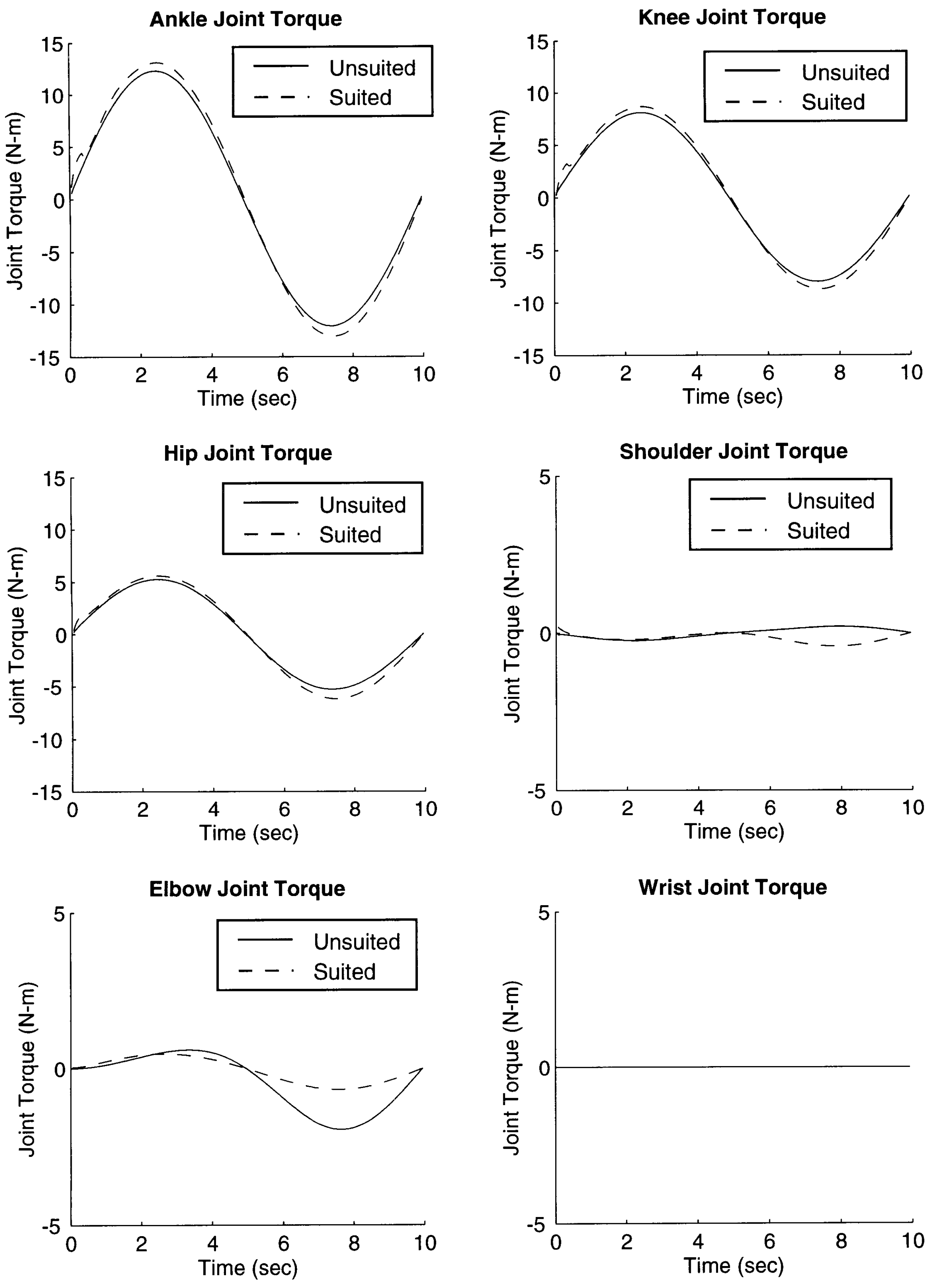


Figure 4.9. Joint torques for payload translation task, compliant lower body.

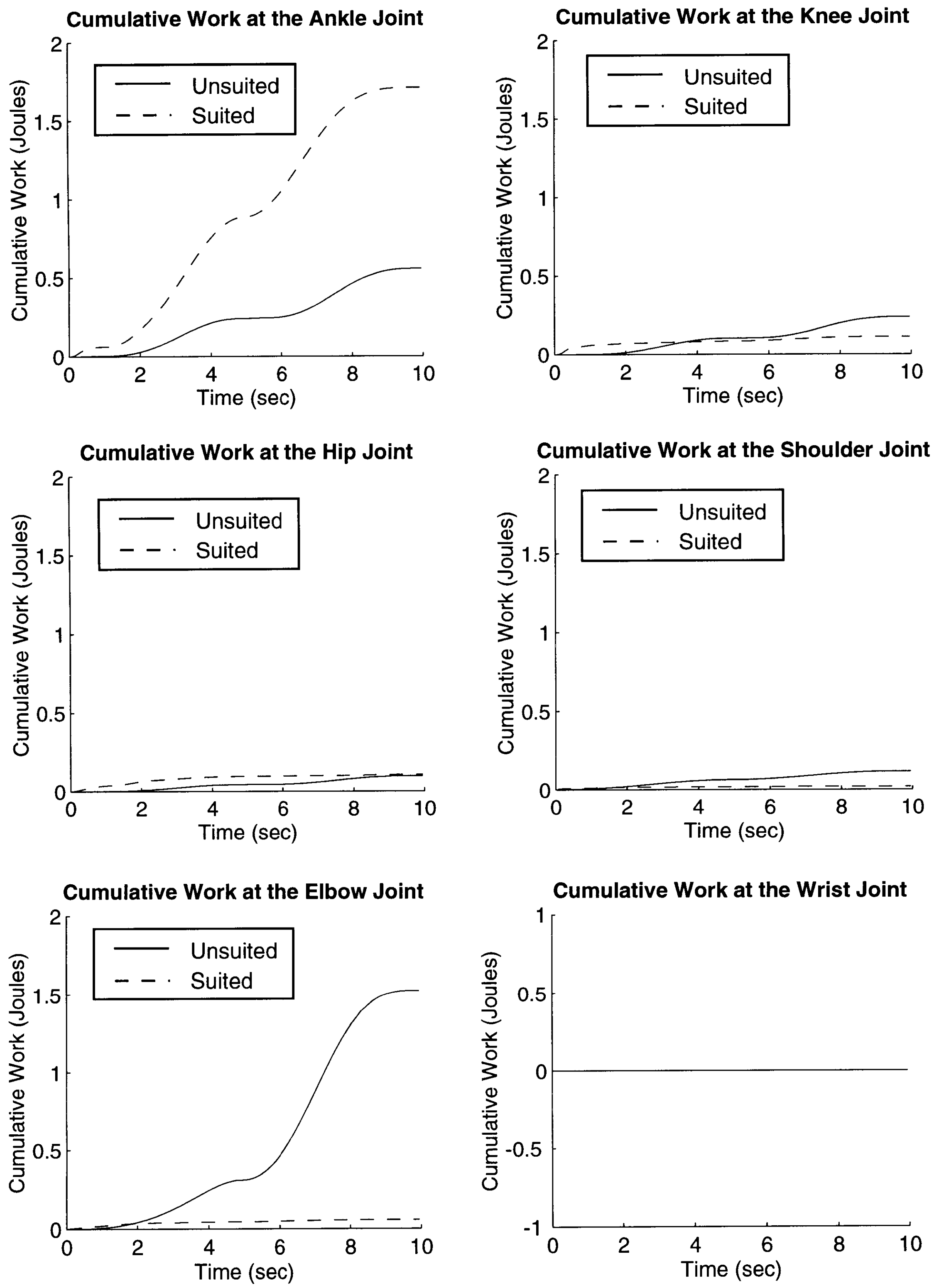


Figure 4.10. Cumulative joint work for payload translation task, compliant lower body.

Although peak torques varied from joint to joint, the velocities in the ankle joint (suited condition) and the elbow joint (unsuited condition) led to increased work at both joints. Figure 4.11 clearly shows that most of the work came from the ankle and elbow joints, with a marked increase at the elbow in the unsuited condition and at the ankle in the unsuited condition. These results will be discussed in Chapter Five, below.

5. Discussion

This chapter interprets the results presented above, and attempts to explain anomalous or suspect data from the simulations. The elbow joint test simulation is discussed first, followed by the rigid body payload translation and the compliant body payload translation tasks.

5.1 Hysteresis Model Test

The elbow joint test demonstrates the functionality of both the hysteresis model and its implementation in computer code (Appendix B). The elbow angle versus imposed torque plot (Figure 4.1) shows nicely the hysteresis nonlinearity produced by the EMU model, creating new branches at the appropriate maximum (74°) and minimum (45°). The discontinuities visible in the first derivative of the elbow angle versus imposed torque plot near 60°, where the imposed torque seems to turn sharply to a new slope, are "edges" in the mesh of Figure 4.2, representing the resolution of the hysteresis grid. These discontinuities appear because of the interpolation performed between data points. If the interpolation used spline fits instead of linear + hyperbolic interpolation, the first derivative would be with continuous slopes.

For the current implementation, the concern is to accurately represent the general character of the hysteresis nonlinearities and not necessarily to be concerned with torque differences on the order of 0.01-0.05 Nm. Additionally, the first-order transition curves used to create the model were crude estimates, probably valid within 10-30% of the true values. The errors in estimating these curves are therefore on the order of 0.1-0.2 Nm, a full order of magnitude higher than the estimated interpolation error; interpolation considerations can be neglected safely. If a precise simulation were required, finer data would have to be used and spline fits would become considerably more attractive.

5.2 Payload Translation Task -- Rigid Lower Body

The translation task performed with a rigid lower body effectively illustrates what would happen if a suited astronaut attempted to perform a task exactly as he would in a shirtsleeve environment. During greater and greater flexion of the elbow and shoulder joints, EMU suit forces begin to increase appreciably and affect the astronaut's workload. The workspace limitations placed on the astronaut by the space suit mean that he must use different motions to perform simple tasks than those he might be familiar with in a shirtsleeve environment, or pay a severe penalty in energy expenditure (2740% at the shoulder, 210% at the elbow) for excursions into unfavorable joint positions.

During this simulation, the ankle, knee, and hip joints exert the reaction torques necessary to maintain a motionless lower body position; therefore, they mimic the sinusoidal velocity and acceleration profiles of the overall motion, visible in Figure 4.5. The suited astronaut's shoulder and elbow joints continue to build greater torques throughout the simulation, as they flex into regions of the workspace with high EMU torque penalties; again, an astronaut would not normally move into these areas of the workspace, and the high torque values observed here illustrate the reasons for his caution. Wrist torque remains nonexistent due to the conditions imposed by the simulation: the astronaut is charged with pulling the Spartan payload directly toward himself, and does not need to exert a direct torque on the payload. The wrist is acting as a frictionless joint, allowing the elbow and shoulder to bear the full load of the work required to move the mass. The range of motion of the wrist has been extended considerably ($\pm 70^\circ$) to allow for joint positions to reflect orientations of the wrist joint and the grip of the fingers; the "lumped" wrist joint never reaches the edge of its workspace, even at -55°, so the torque remains zero.

The cumulative shoulder joint work required to perform this task is shown in Figure 4.6. Relative to the unsuited condition, the suited astronaut does an order of magnitude more work at the shoulder and twice as much work at the elbow to produce the same result. Almost all of the excess shoulder work is produced as a consequence of the simulation constraints. Unable to use his lower body, the astronaut moves his arms into inefficient areas of the suit workspace (near the joint limits) to complete the linear trajectory. At these extreme joint angles, he must exert a great deal of energy to maintain a smaller joint volume through certain portions of the task (see equation 2.1).

5.3 Payload Translation Task -- Compliant Lower Body

The second simulation is more realistic, providing the astronaut with a flexible lower body controlled by simple joint stiffnesses. The posture that the astronaut assumes while performing the translation task with a compliant lower body is strikingly different for the suited and unsuited conditions (see Figures 4.7 and 4.8). Although knee and hip joints maintain a similar orientation for both suited and unsuited conditions, the shoulder and elbow joints flex a great deal more (40° and 80°, respectively) in the unsuited condition while the ankle joint flexes more (12°) in the suited condition. To achieve the mandated translation of the payload in the previous simulation, with a rigid lower body, the suited astronaut was required to pull his upper and lower arms in close to his side. The suited, compliant astronaut in this simulation is able to manipulate the payload from a more suit-neutral posture, his arms floating easily in front of his body.

The torques necessary to complete the task (Figure 4.9) support this idea: the unsuited astronaut exerts greater torque at his elbow joint, while the suited astronaut exerts greater torque at the ankle, knee, and shoulder joints. The unsuited astronaut uses mostly elbow work to complete the task (see Table 4.2), while the suited astronaut uses a combination of elements. Although the shoulder torque has the greatest apparent difference between suited and unsuited conditions, its relative magnitude is not as great as the other torques. The cumulative work in Figure 4.10 reveals that the ankle joint performs most of the work for the suited condition, while the elbow joint performs most of the work in the unsuited condition.

It should be noted at this point that the full reaction torques observed at the ankle, knee, and hip joints for this task are not converted directly into work values. Work is calculated as the time integral of torque x angular velocity, and the hip and knee joints, for example, have only very slight velocities though they support significant torques. This is isometric torque, and it is not included in the traditional definition of work (work requires motion). Work does take place inside the muscle during these periods of isometric exertion, however, and is a critical issue for future studies involving more detailed muscle models. For the purposes of this simulation, though, it is the differential work performed between suited and unsuited conditions that is of interest. The unsuited and suited reaction torques are identical due to the identical motion of the payload mass, and they cancel each other out in any differences; if work values could be computed for the isometric periods, the reaction torques would become a normalizing factor, and would be removed from the suited-unsuited difference regardless. The exclusion of isometric torque production is thus an inherent normalization for the data, and presents no problems with respect to the work calculations.

Allowing for a compliant lower body substantially reduces the work done at the shoulder and elbow joints (compare Figures 4.6 and 4.9). In fact, the work performed in the suited and unsuited conditions with a compliant lower body is almost identical throughout the duration of the task, and total work for the suited condition is actually 26.7 *% lower* than the unsuited condition. Taking advantage of the long lever arm from shoulder to foot restraint, the suited astronaut uses his ankles to move the payload through the same distance as the unsuited subject while doing minimal work. Total work for the suited condition is less than the unsuited condition because the efficiency of using the long shoulder-to-ankle lever arm is higher than elbow and shoulder contraction, given the imposed passive dynamic control at the various joints. The lower body joints were set to have greater stiffness than the shoulder and elbow joints; therefore, the unsuited astronaut chose to perform the motion with his upper body. The suited astronaut, although forced to use his ankles instead of his upper body, performed less net work because the ankle joint was able to achieve the same linear movement at the payload without a large angular displacement at the joint. The control torques were related to displacement and the work was related to angular velocity, so the ankle did not have to perform as much work to move the payload the same linear distance. The particular geometry of this task lent itself well to using small ankle motion, as opposed to larger upper body movements, to achieve the task objective.

The suited results for the compliant lower body simulation also compare well with the rigid body simulation. Working with the suit close to a zero-energy neutral posture, more of the work performed actually translates to payload motion; in the rigid lower body simulation, most of the work done was excess work allocated to changing space suit volume at the shoulder and elbow joints. The total work required to complete the suited task with a compliant lower body is a full 77% lower than the rigid condition.

Although this solution appears optimal from an energy standpoint, it comes with serious drawbacks. The shoulder and elbow joints work to move the payload under the rigid-body and unsuited compliant-body conditions, but they are supplied with muscles conditioned for heavier work on Earth. The small musculature about the ankle joint, however, has been trained on Earth to supply the minor corrections necessary for balance during quiet standing. These muscles are not generally used to move massive payloads about in space; when called upon to do so, they may fatigue quickly under unfamiliar loads. These findings agree with anecdotal evidence provided by astronauts subsequent to certain EVA missions which have generated complaints of sore ankles, especially following the use of a foot restraint.

The astronaut's posture in the portable foot restraint (both feet together) leads to increased reaction torques at the ankle joint (Figure 4.9), which agree with the findings of Schaffner (1997). This study produced similar results, but emphasizes the postural configuration imposed by the EMU space suit. The addition of a space suit aggravates an already poor situation, increasing torque levels at the ankle joint beyond the already high reaction torque values required to maintain an upright posture. These results do not include a full dynamic model for the EMU ankle joint, however; data for the ankle has not yet been obtained, and may alter simulation findings. As noted by Schaffner, experimental testing would help to resolve modeling errors similar to this.

The work presented in this thesis complements the studies of skill in EVA mass handling by McDonald *et al.* (1997), who are investigating aspects of postural control and human-environment interactions as they apply to EVA task definition and worksite design. Their studies characterize the ability of astronauts to successfully manipulate objects in microgravity, from a variety of orientations and postures. One of the factors they identify as affecting an astronaut's choice of motion is the mechanical constraint imposed by the EMU space suit on joint ranges of motion. The EMU model developed herein is a tool that can be applied to models of postural configuration and control as an environmental constraint, and enhance the predictive ability of models to quantify workload and skill in EVA operations. Additionally, results from the experiments performed by McDonald *et al.* would provide interesting data to examine for the effects of suit dynamic properties on astronaut motion. Simulations of the experimental tasks could be performed and compared with their data to assess the accuracy of the EMU space suit model.

This space suit model also contributes to the work performed at NASA facilities by Morgan *et al.* (1996), who examined available suited strength during EVA operations. Their paradigm involves developing a predictive model for the maximum effort available to astronauts while in the space suit, but does not address the representation of the imposed suit torques mathematically. The hysteresis modeling approach taken in Chapter Three provides a good mathematical foundation for construction of a model from the NASA data, and it extends the trends observed in their discrete suit data to a continuous function throughout the astronaut's workspace. The NASA group has proposed detailed dynamic simulations of suited astronauts, and the model created here proves the feasibility of that goal. Additionally, the EMU torque data will undoubtedly be instrumental in developing future implementations of a space suit model with greater precision.

Although computer representations of the EMU have been used for EVA design and analysis [Price, 1994; Pardue, 1996], none of these studies have incorporated suit dynamical effects. Previous work has focused primarily on the logistical aspects of EVA timelining and choreography, using computer graphics techniques to ensure that an astronaut will be able to complete a task without exceeding the boundaries of the constraining reach envelope. Therefore, most of the research in the literature characterizes the workspace of an astronaut in terms of areas that are "possible to reach" and "not possible to reach;" this thesis enables an augmented form of analysis in which the workspace is a continuous volume of joint locations with varying penalties of energy expenditure. The previously geometric analyses will now be able to incorporate a dynamic suit model and optimize tasks for both energy and geometric considerations.

The empirical, data-driven space suit model chosen for this thesis represents a different approach than the model of space suit glove construction developed by Main *et al.* (1995). Focusing on the physical basis for fabric modeling in a bending pressurized cylinder (see sections 2.2.3 and 2.3), their simulation work has centered around glove design and fabric selection. Empirical models are inherently faster, performing relatively few computations to derive suit forces, but lack the theoretical foundation of Main's physically-based models. The scale of the multibody dynamics simulations presented herein requires a relatively simple (computationally speaking) space suit model, especially in light of the goal to develop real-time interactive simulations for training purposes. If the Main model is shown to effectively reproduce space suit joint forces and provide fast enough algorithms, however, it could be a useful and more elegant replacement for the empirical model. The integrated approach presented in this thesis for physiological, space suit, and environmental interaction modeling is a macroscopic framework on which individual models such as the Main pressurized fabric model can be placed in the future.

Tests performed by Pantaleano and Lacey (1992) involving loads imparted to the suit as a result of handling massive objects during EVA show that maximum axial restraint loads are a key design parameter for space suits; these studies would benefit dynamic simulation, possibly incorporating the fabric modeling techniques discussed in Chapter Two or those of Main *et al.*, above. Ideally, fabric properties of the suit could be combined with the EMU model presented in this thesis to create a complete human/suit model that could answer both human factors and space suit construction questions. An integrated model would also allow space suit joint designers to benefit from computer-based rapid prototyping, facilitating quantitative analysis during the conceptual phase of design and helping to mitigate the expensive process of prototype construction and evaluation.

It should be emphasized that the results presented here are highly sensitive to the specific torques imposed by the hysteresis model for suit dynamics. As noted in Chapter Three, the hysteresis model is only as accurate as the data from which it is constructed; currently only data for the major hysteresis curves, and not a complete family of first-order transition curves, has been obtained. Suit torque data from the central ranges of motion, and from actual EVA operations, would be invaluable in constructing a more accurate joint model.

6. Conclusions and Recommendations for Future Work

Although these simulated EVA missions would most likely be successful regardless of the specific way that they were performed, they highlight important human factors issues that should be taken into account during EVA design. For example, the postures suggested by the Spartan payload simulations indicate that ankle support may be a necessary part of future EVA foot restraints; alternative boot designs could be incorporated into the model for rapid-prototype analysis. The Spartan simulation also aids in task definition, restricting the available workspace by identifying regions of increased work and decreased task efficiency. These simulations can be used to develop a more comfortable working environment for the EVA astronaut, and help to identify specific EVA tasks that might lead to unanticipated problems.

The EMU suit model was successfully implemented, and served to enhance the realism and validity of the EVA task simulations. The space suit model contributes new ideas about energy conservation and expenditure in the context of optimal trajectory generation for astronaut motions; future forward-dynamic simulations could include optimal control methods to more accurately describe the ways in which humans might use their instinctive knowledge of "restrictive" or "highly mobile" areas of the suit workspace to choose energy-optimal joint trajectories. A forward dynamics simulation approach would also remove the strict task definitions imposed by the current method of task prescription and subsequent inverse dynamics computation.

Experimental verification of results from the EMU space suit model is a clear priority for any research involving pure simulation, and it is possibly even more important when the model is based on a sparse data set like the EMU space suit data. Unfortunately, direct measurement of joint torques during Shuttle EVA missions is a difficult proposition; there are several good alternatives, however. If the speed of movement is maintained at a suitably low level, data from neutral buoyancy facilities could be used in lieu of actual on-orbit data to confirm simulation results. Torque or joint angle measurements would be a great deal more feasible in water tanks on Earth than in space, and would be an excellent first step towards correlating simulation results with EMU data.

Similarly, the EMU model would benefit greatly from detailed dynamometer sessions with a pressurized space suit. In particular, first order transition curves at the joints (where each joint is flexed from full extension to a certain maximum flexion and then back again, extending the maximum flexion angle with each subsequent trial) throughout the ranges of motion would be extremely helpful in piecing together an accurate hysteresis model. Additionally, dynamic parameters like EMU suit joint damping are currently poorly known; although these values would extend the model's capability, it is unlikely that astronauts would move with great enough velocities to encounter a velocity-dependent torque in the suit.

Future EVA models should also incorporate greater physiological realism, and address issues related to the intelligent control of astronaut movement. Musculoskeletal models allow the transformation of joint torque values into muscle activation energies, which in turn will provide data relating to the work performed by specific muscle groups. Muscle models also help to quantify isometric work performed during the course of a simulation. Additionally, implementation of an astronaut movement control model will allow for forward dynamics solutions, driven by goal-oriented EVA tasks rather than rigidly defined limb motions.

Although these simulations produced an interesting result, the major contribution of this thesis is more than the task-specific conclusions drawn in the Discussion, above. These simulations only represent a sample of the quantitative analysis possible with the specification of a dynamic EMU space suit model. It is a tool that can be used to explore a wide range of issues in EVA preparation, planning, and training; future experimental and analytical work can both improve this model and benefit from its use.

The objectives of this thesis were to develop a dynamic model for the EMU space suit, and to illustrate key properties of the model through several simulations. To achieve this, an original dynamic model of the EMU space suit was developed, based on data obtained from NASA and Hamilton Standard; three key characteristics of space suit dynamics were used to create the EMU model: mass, inertia, and suit-imposed torques. Mass and inertial properties were included through the SD/FAST software package for multibody dynamics, and an empirically-based, data-driven model for the torques imposed by space suit joints was developed and implemented numerically based on observed suit parameters.

The model for imposed suit torque focused on existing suit data, which presumably represents some combination of the effects that various suit imperfections, such as volume variability and fabric stiffness, have on suit performance. An empirical model based on this data was chosen rather than a ground-up, physically based model for reasons of computational efficiency, and for the expectation that experimental data would better represent the true characteristics of space suit joints until fabric dynamics are better understood. EMU joint torque measurements reveal a significant degree of hysteresis, and were therefore modeled using a modified Preisach model for hysteresis.

The Preisach model provides an effective framework for representing hysteretic systems mathematically, and it was shown that the model can represent suit torque as a function of joint angle, even though the theory traditionally produces joint angle as a function of joint torque. To convert the basic space suit data to a more comprehensive database for application throughout the workspace of each joint, families of hysteresis curves were constructed based on Preisach model parameters. The hysteresis model was then implemented numerically, tested at each joint, and specific results were presented for a test simulation at the elbow joint. The simulated elbow joint was flexed and extended in sequence while recording the torque output provided by the model; results reproduce the observed hysteretic properties of the EMU elbow joint.

A human musculoskeletal model was constructed to provide the underlying framework and reference point for work and energy measurements, utilizing basic biomechanical modeling techniques. The skeletal system of the human body was represented as a thirteen-segment model connected by pin and ball joints, and muscles were represented as simple passive elements with stiffness and damping characteristics. Several active-state muscle models were explored for possible future implementation, but eliminated from the current study due to the increase in complexity required by these models. The space suit and human models were subsequently combined, and used in two simulations as a demonstration of the overall EMU model's capability; each simulation was repeated under both suited and unsuited conditions to observe the effect of space suit dynamics on task performance.

The first simulated EVA task described an astronaut pulling the Spartan astrophysics payload forward while maintaining a rigid lower body posture, and illustrated what would happen if a suited astronaut attempted to perform a task exactly as he would in a shirtsleeve environment. During greater and greater flexion of the elbow and shoulder joints, EMU suit forces began to increase appreciably and affect the astronaut's workload. Severe penalties in energy expenditure (2740% at the shoulder, 210% at the elbow) were observed for excursions into unfavorable joint positions; under normal circumstances, an EVA astronaut would never venture into these energy-intensive regions of the workspace, but the simulation is a good example of how strength constraints are placed on the astronaut. The next simulation built on this result, and revealed an interesting effect that space suit constraints have on EVA performance.

The second demonstration task was identical to the first, with the exception that the astronaut's lower body was now allowed to move, and "choose" a posture from which to perform the task using a compliant lower body musculoskeletal model. Basic muscle stiffnesses were included to provide simple postural control in the lower body, and the simulation was again performed in both suited and unsuited conditions. There was a dramatic difference between the posture that the astronaut assumed while performing the task in suited and unsuited conditions: the shoulder and elbow joints flexed a great deal more (40° and 80°, respectively) in the unsuited condition, while the ankle joint flexed more (12°) in the suited condition. The unsuited astronaut primarily used arm movements to complete the task, while the suited astronaut manipulated the payload with extended arms to assure that all EMU joints remained in optimal low-energy positions; this resulted in a significant difference in work distribution between the suited and unsuited conditions.

The suited astronaut, with extended arms, performed most of the work with movement of the ankle joint (85%), while the unsuited astronaut used more upper-body strength (60% at the elbow, 22% at the ankle). Taking advantage of the long lever arm from shoulder to foot restraint, the suited astronaut used his ankles to move the payload through the same distance as the unsuited subject while doing minimal work; the ankle joint was able to achieve the same linear movement at the payload without a large angular displacement at the joint. Total work performed at the shoulder and elbow joints for the suited condition also decreased dramatically (77%) from the suited, rigid lower body simulation that forced the astronaut into unfavorable postures. These results illustrate the important role that the space suit plays in defining astronaut posture during EVA, and the implications that specific postural configurations have in determining work distribution and energy expenditure.

In summary, this thesis has developed and implemented a dynamic model of the Space Shuttle EMU, and applied the model to several key simulations of EVA tasks. The model addresses a key deficit of current computerized space suit models, adding the dynamic effects of suit inertia and torque to geometrically-based models. The suit fills an important gap in the existing multibody dynamics simulation system for EVA task analysis, and helps to define the restricted workspace that confines astronauts in space. Future work in EVA computer simulation based on this suit model will enhance astronaut training, efficiency, and comfort, and complement existing physical simulators to ensure a new millennium of humans working successfully in space.

7. References

Benati, M., Gaglio, S., Marasso, P., Tagliasco, V., Zaccaria, R. (1980). "Anthropomorphic robotics. I. Representing mechanical complexity. II. Analysis of manipulator dynamics and the output motor impedance." Biological Cybernetics **38**: 125-140, 141-150.

Breen, D. E. (1996). "Computer Graphics in Textiles and Apparel Modeling." IEEE Computer Graphics and Applications **16**(5): 100-102.

Crowninshield, R., Brand, R.A. (1981). "A physiologically based criterion of muscle force prediction in locomotion." Journal of Biomechanics **14**: 793-801.

Dionne, S. (1991). AX-5, Mk III, and Shuttle Space Suit Comparison Test Summary. Moffett Field, CA, NASA Ames Research Center.

Eberhardt, B., Weber, A, Strasser, W. (1996). "A Fast, Flexible, Particle-System Model for Cloth Draping." IEEE Computer Graphics and Applications **16**(5): 52-59.

Happee, R. (1992). "Time optimality in the control of human movements." Biological Cybernetics **66**: 357-366.

Happee, R., Van der Helm, F.C.T. (1995). "The control of shoulder muscles during goal directed movements." Journal of Biomechanics **28**(10): 1179-1191.

Hatze, H. (1981). Myocybernetic control models of skeletal muscle: Characteristic and applications. Pretoria, University of South Africa.

Hill, A. V. (1938). "The heat of shortening and the dynamic constants of muscle." Proceedings of the Royal Society **126B**: 136-195.

Jackson, D. K. (1997). Development of Full-Body Models for Human Jump Landing Dynamics and Control. Cambridge, MA, Massachusetts Institute of Technology. Doctoral Dissertation, Department of Aeronautics and Astronautics.

Kane, T. R., Levinson, D.A. (1985). DYNAMICS: Theory and Applications, McGraw-Hill, Inc.

Kawabata, S. (1980). The Standardization and Analysis of Hand Evaluation. Textile Machinery Society of Japan, Osaka, Japan.

Keefe, M. (1994). "Solid Modeling Applied to Fibrous Assemblies Part II: Woven Fabric." J. of the Textile Institute **85**(4): 350-359.

Klaus, D. M., West, P.R. (1989). Performance Evaluation of Advanced Space Suit Concepts for Space Station. 19th Intersociety Conference on Environmental Systems, San Diego, CA, Society of Automotive Engineers, Inc.

Kozloski, L. D. (1994). U.S. Space Gear. Washington, D.C., Smithsonian Institution Press.

Lafleur, B., Thalmann, N.M., Thalmann, D. (1991). Cloth Animating with Self-collision Detection. Modeling in Computer Graphics. T. L. Kunii. Berlin, Springer-Verlag: 179-187.

Mayergoyz, I. D. (1991). Mathematical Models of Hysteresis. New York, Springer-Verlag.

McMahon, T. (1984). Muscles, Reflexes, and Locomotion. Princeton, NJ, Princeton University Press.

Menendez, V., Labourdett, X., Baez, J.M. (1993). Performance of EVA Suit Mobility Joints: Influence of Driving Parameters. 23rd International Conference on Environmental Systems, Colorado Springs, CO, Society of Automotive Engineers.

Moore, M. W., J. (1988). "Collision Detection and Response for Computer Animation." Computer Graphics (Proc. Siggraph) **22**(4): 289-298.

Morgan, D. A., Wilmington, R.P., Pandya, A.K. (1996). Comparison of Extravehicular Mobility Unit (EMU) Suited and Unsuited Isolated Joint Strength Measurements. Houston, TX, NASA: Lyndon B. Johnson Space Center. Technical Paper 3613.

NASA (1991). Man Systems Integration Standards 3000. Volume 3.

NASA (1996). EVA Project Roadmap Plan. Houston, TX, EVA Project Office, Johnson Space Center.

Newman, D. J., Barratt, M. (1997). Life Support and Performance Issues for Extravehicular Activity (EVA). Introduction to Space Life Sciences, Kleger Press, Orion Books.

Pandy, M. G., Anderson, F.C., Hull, D.G. (1992). "A Parameter Optimization Approach for the Optimal Control of Large-Scale Musculoskeletal Systems." Journal of Biomechanical Engineering **114**: 450-460.

Pardue, F., Pandya, A.K., Maida, J. (1996). Creation of the Advanced Extravehicular Mobility Unit (EMU) Computer Graphics Model. Houston, TX, GRAF Facility, Lyndon B. Johnson Space Center.

Price, L. R., Fruhwirth, M.A., Knutson, J.G. (1994). Computer Aided Design and Graphics Techniques for EVA Analysis. 24th International Conference on Environmental Systems and 5th European Symposium on Space Environmental Control Systems, Friedrichshafen, Germany, Society of Automotive Engineers.

Reinhardt, A. (1989). Results and Applications of a Space Suit Range-of-Motion Study. 19th Intersociety Conference on Environmental Systems, San Diego, CA, Society of Automotive Engineers.

Sakaguchi, Y., Minoh, M., Ikeda, K. (1991). "PARTY: Physical Environment of Artificial Reality for Dress Simulation (I) -- A Dynamically Deformable Model of Dress." Trans. Soc. of Electronics, Information and Communications: 25-32.

Schaffner, G. (1995). Dynamic Analysis of Extravehicular Activity (EVA). Cambridge, Massachusetts Institute of Technology. Master's Thesis, Department of Aeronautics and Astronautics.

Schaffner, G., Newman, D.J., Robinson, S.K. (1997). Inverse Dynamic Simulation and Computer Animation of Extravehicular Activity (EVA). AIAA 35th Aerospace Sciences Meeting, Reno, NV, American Institute of Aeronautics and Astronautics, Inc.

Thibodeau, G. A., Patton, K.T. (1996). Anatomy and Physiology. St. Louis, Mosby-Year Book Inc.

West, P. R., Trausch, S., Stelly, C. (1989). Space Suit Component Unmanned Torque and Range Measurement Test Plan, NASA Lyndon B. Johnson Space Center. Report CTSD-SS-329.

Winters, J. M., Stark, L. (1985). "Analysis of Fundamental Human Movement Patterns Through the Use of In-Depth Antagonistic Muscle Models." IEEE Transactions on Biomedical Engineering **32**(18): 826-839.

Winters, J. M. (1995). "How detailed should muscle models be to understand multi-joint movement coordination?" Human Movement Science **14**: 401-442.

Yang, Y., Thalmann, N.M. (1993). An Improved Algorithm for Collision Detection in Cloth Animation with Human Body. Proc. of Pacific Graphics. Singapore, World Scientific Press: 237-251.

Appendix A: Transition Curve and Mesh Construction Code

This appendix contains the MATLAB code used to construct the family of transition curves required for the hysteresis model, and process that data into a mesh suitable for using in the simulations. The file shown processes elbow data, but can be used equivalently at any joint.

```
true = 1;
false = 0;
data = false;

% The order for the polynomial fit to hysteresis data
order = 9;

clg;
hold on;
clear curveset;

% If we want to plot the data first
if data == true
 plot(elbow(:,1),elbow(:,2), 'g');
 plot(elbow(:,1),elbow(:,3), 'r');
end

% Do an nth-order polynomial fit to the major ascending and descending
curves
[elbowfit2,A] = polyfit(elbow(:,1),elbow(:,2),order);
[elbowfit3,A] = polyfit(elbow(:,1),elbow(:,3),order);

% Evaluate these fits at 0.5 degree intervals and plot.
theta = 0:.5:115;
ascending = polyval(elbowfit2,theta);
descending = polyval(elbowfit3,theta);
plot(theta,ascending,'r--');
plot(theta,descending,'g--');
```

```
% Find a family of curves, given the number of curves,
% "bunching" and "risetime."
% Bunching represents how tightly packed the curves are to the lower,
% descending branch.  Risetime dictates how quickly the first order
% transition curves “rise” to hit the major, ascending curve.

curves = 8;
bunching = 2;
risetime = 1.25;

% Set merge point, where all curves meet at the neg. sat. point
for cnum = 0:curves-1
 curveset(1,(2*cnum+1):(2*cnum+2)) = [0 polyval(elbowfit3,0)];
end

start = 0;
finish = 115;

% To terminate the family
breakout = (finish-start)/(curves+1);
breakcurve = curves;
breakpoint = 0;
breaking = 0;

% Construct curves, enforcing the derivative rule (eqn. 3.11)
for i = 1:(finish-start)

 slope_min = polyval(elbowfit3,start+i)-polyval(elbowfit3,start+i-1);
 slope_max = polyval(elbowfit2,start+i)-polyval(elbowfit2,start+i-1);
 slope_inc = (slope_max-slope_min)/(curves+1)/bunching;


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % To break curves off
 if fix(rem(i,breakout)) == 0
  breaking = 1;
 end

 if breaking == 1
    slope(breakcurve) = risetime*slope(breakcurve);
    breakpoint = curveset(i,2*(breakcurve-1)+2)+slope(breakcurve);
    curveset(i+1,(2*(breakcurve-1)+1):(2*(breakcurve-1)+2)) = [start+i
breakpoint];

    % cut it off and find the proper intersection point if outside of major
loop
    if breakpoint > polyval(elbowfit2,start+i)
     mp = polyval(elbowfit2,start+i)-polyval(elbowfit2,start+i-1);
     bp = polyval(elbowfit2,start+i)-mp*(start+i);
     mb = breakpoint - curveset(i,2*(breakcurve-1)+2);
     bb = breakpoint - mb*(start+i);
     newpoint = [-mp 1;-mb 1]\([bp bb]');
     curveset(i+1,(2*(breakcurve-1)+1):(2*(breakcurve-1)+2)) = newpoint' ;
     breaking = 0;
     breakcurve = breakcurve -1 ;
    end
  end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 if breakcurve == 0
  break
 end
 if breaking == 0
  for cnum = 1:breakcurve
   slope(cnum) = slope_min+slope_inc*cnum;
   curveset(i+1,(2*(cnum-1)+1):(2*(cnum-1)+2)) =
      [start+i curveset(i,2*(cnum-1)+2)+slope(cnum)];
  end
 else
  for cnum = 1:(breakcurve-1)
   slope(cnum) = slope_min+slope_inc*cnum;
   curveset(i+1,(2*(cnum-1)+1):(2*(cnum-1)+2)) =
      [start+i curveset(i,2*(cnum-1)+2)+slope(cnum)];
  end
 end

end

% Create an array that holds each curve's last point
for cnum = 1:curves-1
 lastpoint(cnum) = 1;
 while ~(curveset(lastpoint(cnum),2*cnum+2) == 0)
  lastpoint(cnum) = lastpoint(cnum) + 1;
 end
 lastpoint(cnum) = lastpoint(cnum) -1;

plot(curveset(1:lastpoint(cnum),2*cnum+1),curveset(1:lastpoint(cnum),2*cnum+
2),'y');
end

plot(curveset(:,1),curveset(:,2),'y');


%shift the lastpoints over 1 to make room for the first curve's lastpoint
lastpoint(3:curves+1) = lastpoint(1:curves-1);
[lastpoint(2),temp] = size(curveset);


% Add the major loop descending branch as the first curve,

[temp,elements] = size(theta);
for i = 1:(elements+1)/2;
 theta(i) = theta(2*i-1);
 ascending(i) = ascending(2*i-1);
 descending(i) = descending(2*i-1);
end
theta = theta(1:i);
ascending = ascending(1:i);
descending = descending(1:i);
lastpoint(1) = i;

curve = zeros(lastpoint(1),curves*2+2);
curve(:,1) = theta';
curve(:,2) = descending';
curve(1:lastpoint(2),3:(2+curves*2)) = curveset(:,:);
curves = curves+1;
```

```
% reduce the grid size to 10 x 10

for i = 1:curves
  gridpoint(i) = curve(lastpoint(curves+1-i), 2*(curves+1-i)-1);
end


clear newgrid;
for curveindex = 1:curves-1

  newgrid(1, 2*curveindex-1) = curve(1, 2*curveindex-1);
  newgrid(1, 2*curveindex) = curve(1, 2*curveindex);

  for pointindex = 2:curves-curveindex+1
    target = gridpoint(pointindex-1);

    huntindex = 1;
    while ~( ((target > curve(huntindex,   2*curveindex-1)) & (target <
curve(huntindex+1, 2*curveindex-1))))
      huntindex = huntindex + 1;
    end

    fnew = (curve(huntindex+1, 2*curveindex) - curve(huntindex,
2*curveindex)) * (target - curve(huntindex, 2*curveindex-1)) /
(curve(huntindex+1, 2*curveindex-1) - curve(huntindex, 2*curveindex-1)) +
curve(huntindex, 2*curveindex);

    newgrid(pointindex, 2*curveindex-1) = target;
    newgrid(pointindex, 2*curveindex) = fnew;

  end

% tack on the last point

  newgrid(pointindex+1, 2*curveindex-1) = gridpoint(pointindex);
  newgrid(pointindex+1, 2*curveindex) =
curve(lastpoint(curveindex),2*curveindex);

end

newgrid(1, 2*curves-1) = curve(1, 2*curves-1);
newgrid(1, 2*curves) = curve(1, 2*curves);
newgrid(2, 2*curves-1) = curve(lastpoint(curves), 2*curves-1);
newgrid(2, 2*curves) = curve(lastpoint(curves), 2*curves);

newgrid(1, 2*(curves+1)-1) = curve(1,1);
newgrid(1, 2*(curves+1)) = curve(1,2);

for i = 1:curves
  lastpoint(i) = curves+2-i;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Compute C's for interpolating f(alpha, beta) in all cells, by row

rows = curves;

% Open a file to record the mesh

fid = fopen('elbow2.mesh','w');

for row = 1:rows

 thetaindex = 2*row-1;
 findex = 2*row;

 alphamax(row) = newgrid(lastpoint(row), thetaindex);
 if row == rows
  alphamin(row) = theta(1);
 else
  alphamin(row) = newgrid(lastpoint(row+1), thetaindex+2);
 end

 fprintf(fid,'%f\t%f\t',alphamin(row), alphamax(row));

 betamin(row) = theta(1);
 betamax(row) = alphamax(row);



% Cell = [betamin betamax c1 c2 c3 c4]

 clear cells;

 if ~(row == rows)

  for cell = 1:(lastpoint(row+1)-1)

   cells(cell,1) = newgrid(cell,   thetaindex);
   cells(cell,2) = newgrid(cell+1, thetaindex);

   bmin = cells(cell,1);
   bmax = cells(cell,2);

   fab(1,1) = newgrid(cell,   findex);
   fab(2,1) = newgrid(cell+1, findex);
   fab(3,1) = newgrid(cell,   findex+2);
   fab(4,1) = newgrid(cell+1, findex+2);

   M = [1 alphamax(row) bmin (alphamax(row))*bmin ;
        1 alphamax(row) bmax (alphamax(row))*bmax ;
        1 alphamin(row) bmin (alphamin(row))*bmin ;
        1 alphamin(row) bmax (alphamin(row))*bmax];


   % solve fab = Mc

   c = inv(M)*fab;
   cells(cell,3:6) = c';

  end
  cell = cell + 1;


% Do the triangle

  clear fab;
  cells(cell,1) = newgrid(cell, thetaindex);
  cells(cell,2) = newgrid(cell+1, thetaindex);
  fab(1,1) = newgrid(cell, findex);
  fab(2,1) = newgrid(cell+1, findex);
  fab(3,1) = newgrid(cell, findex+2);

  bmin = cells(cell,1);
  bmax = cells(cell,2);

  M = [1 alphamax(row) bmin ;
       1 alphamax(row) bmax ;
       1 alphamin(row) bmin ];

 % solve fab = Mc
  c = inv(M)*fab;
  cells(cell,3:5) = c';

  end

  if (row == rows)

   clear fab;
   cells(1,1) = newgrid(1, thetaindex);
   cells(1,2) = newgrid(2, thetaindex);
   fab(1,1) = newgrid(1, findex);
   fab(2,1) = newgrid(2, findex);
   fab(3,1) = newgrid(1, findex+2);

   bmin = cells(1,1);
   bmax = cells(1,2);

   M = [1 alphamax(row) bmin ;
        1 alphamax(row) bmax ;
        1 alphamin(row) bmin ];

  % solve fab = Mc
  c = inv(M)*fab;
  cells(1,3:5) = c';
  cells(1,6) = 0.0;

end
```

```
% Print the mesh

 [length,width] = size(cells);
 fprintf(fid,'%d\n',length);
 for index = 1:length

fprintf(fid,'%f\t%f\t%f\t%f\t%f\t%f\n',cells(index,1),cells(index,2),cells(i
ndex,3),cells(index,4),cells(index,5),cells(index,6));
 end
 fprintf(fid,'\n');

end

fclose(fid);
```

Appendix B: Test Function for f(α,β) Mesh

This appendix contains a test function written in C, implementing a simple test procedure that calculates f(a,b) at set intervals and writes the values to a file. The resulting mesh of values is processed and visualized by the code in Appendix C.

```
#include <stdio.h>
#include <stdlib.h>


/* Each cell is in a row bounded by alphamin and alphamax.
   The cells have bounds betamin and betamax, and four
   coefficients for interpolating f(a,b) within.

   A mesh is made up of rows, and each row is made up of cells;
   rows and cells are allocated dynamically.
*/


typedef struct cell_struct {
  double betamin;
  double betamax;
  double c1;
  double c2;
  double c3;
  double c4;
} Cell;

typedef struct row_struct {
  double alphamin;
  double alphamax;
  int    numcells;
  Cell   *cells;
} Row;

typedef struct mesh_struct {
  int numrows;
  Row *rows;
} Mesh;
```

```
void
main(void)
{
  FILE *fid;
  void loadmesh(char *filename, Mesh *mesh);
  double calc_fab(Mesh mesh,double i,double j);
  Mesh elbowmesh;
  double i,j;
  double start_b, end_b, start_a, end_a;

  loadmesh("elbow2.mesh", &elbowmesh);

  start_b = elbowmesh.rows[0].cells[0].betamin;
  start_a = elbowmesh.rows[elbowmesh.numrows-1].alphamin;
  end_a = elbowmesh.rows[0].alphamax;

  fprintf(stderr,"%lf\n",end_b);


  if((fid = fopen("elbowtest2.dat", "w")) == NULL) {
    fprintf(stderr, "Can't open dat file to write\n");
    exit(-1);
  }


/* Go through and calculate f(a,b) at 5 degree intervals */

  i = start_a+0.5;
  while( i < end_a ) {

    j = start_b+0.5;
    while( j < i) {
      fprintf(fid,"%lf %lf %lf\n", i, j, calc_fab(elbowmesh,i,j));
      j+=5;
    }

    i+=5;
  }

  close(fid);

}
```

```
/* Interpolates f(a,b), given a mesh and coordinates (a,b)*/

double
calc_fab(Mesh mesh, double a, double b)
{
  int i,j;
  int triangle = 0;

  for(i=0; mesh.rows[i].alphamin > a ; i++)
    ;
  for(j=0; mesh.rows[i].cells[j].betamax < b ; j++)
    ;
  if( j == mesh.rows[i].numcells-1)
    triangle = 1;

  if( !triangle )
    return(mesh.rows[i].cells[j].c1     +
         mesh.rows[i].cells[j].c2*a   +
         mesh.rows[i].cells[j].c3*b   +
         mesh.rows[i].cells[j].c4*a*b);
  else
    return(mesh.rows[i].cells[j].c1     +
         mesh.rows[i].cells[j].c2*a   +
         mesh.rows[i].cells[j].c3*b);
}


/* Loads a mesh into a large structure from a data file */

void
loadmesh(char *filename, Mesh *mesh)
{

  FILE *fp;
  int items_read, ncells, i;
  double amin, amax;
  Row *current_row;
  Cell *current_cell;

  if((fp = fopen(filename, "r")) == NULL) {
    fprintf(stderr, "Can't open mesh file\n");
    exit(-1);
  }

  mesh->numrows = 0;
  mesh->rows = NULL;

  while((items_read = fscanf(fp,"%lf%lf%d",&amin,&amax,&ncells)) == 3) {

    mesh->numrows++;
    mesh->rows = realloc(mesh->rows, mesh->numrows*sizeof(Row));

    current_row = &((mesh->rows)[mesh->numrows-1]);

    current_row->alphamin = amin;
    current_row->alphamax = amax;
    current_row->numcells = ncells;
    current_row->cells = malloc(ncells*sizeof(Cell));

    for(i=0; i < ncells ; i++) {
      fscanf(fp,"%lf%lf%lf%lf%lf%lf",
           &((current_row->cells)[i].betamin),
           &((current_row->cells)[i].betamax),
           &((current_row->cells)[i].c1),
           &((current_row->cells)[i].c2),
           &((current_row->cells)[i].c3),
           &((current_row->cells)[i].c4));
    }
  }
}
```

Appendix C: Visualization Processing for Mesh Test

This appendix contains the function "process_test," used to create a 3-dimensional surface from the surface data produced in Appendix A. The surface can be quickly scanned for continuity and accuracy.

```
function [] = process_test(data)

clear X ;
clear Y;
clear Z;
clear xindex;
clear ymax;
tolerance = 0.001;
xindex = 1;
ymax = 1;
found_a_j = 0;

X(xindex) = data(1,1) ;
Y(ymax) = data(1,2) ;

[rows cols] = size(data);

for i = 1:rows

  if abs(data(i,1) - X(xindex)) > tolerance

    xindex = xindex + 1 ;
    X(xindex) = data(i,1) ;

  end



  for j = 1:ymax

    if abs(data(i,2) - Y(j)) < tolerance
      found_a_j = 1;
      break;

    end

  end

  if found_a_j == 0
    ymax = ymax + 1 ;
    Y(ymax) = data(i,2) ;
    j = j+1;
end

  Z(j,xindex) = data(i,3);
  found_a_j = 0;

end

[rows cols] = size(Z);
```

```
for i = 1:rows
 for j = 1:cols
  if abs(Z(i,j)) < 0.0001
   Z(i,j) = NaN;
  end
 end
end

hold off
clg
surf(X,Y,Z)
view(20,30)
```

Appendix D: Elbow Joint Test

This appendix contains code to test the elbow joint hysteresis model. An input datafile is read, which contains a sequence of elbow joint positions and a velocity indication (the model needs to know if the input is increasing or decreasing). The compose_force() function then creates the hysteresis torque for the elbow joint, based on a stored history of extrema (the M[] and m[] arrays in mesh_struct). Output is written to a file. Note the difference between the mesh_struct here that must maintain a full input history, and the simplified struct in Appendix B.

```
#include <stdio.h>
#include <stdlib.h>

#define UP 0
#define DOWN 1

typedef struct cell_struct {
  double betamin;
  double betamax;
  double c1;
  double c2;
  double c3;
  double c4;
} Cell;

typedef struct row_struct {
  double alphamin;
  double alphamax;
  int    numcells;
  Cell   *cells;
} Row;

typedef struct mesh_struct {
  int last_direction;
  double last_angle;
  int num_trapezoids;
  double M[100];
  double m[100];
  double max_saturation;
  int numrows;
  Row *rows;
} Mesh;
```

```
void
main(void)
{
  void loadmesh(char *filename, Mesh *mesh);
  void initialize_mesh(Mesh *mesh, double angle);
  double calc_fab(Mesh mesh,double i,double j);
  double compose_force(Mesh *mesh, double theta, int direction);

  Mesh elbowmesh;
  FILE *fid, *outfid;
  int i;
  double input_data[60];
  int input_dir[60];

  loadmesh("elbow3.mesh", &elbowmesh);
  initialize_mesh(&elbowmesh, 35.0);

  if((fid = fopen("testinput.txt", "r")) == NULL) {
    fprintf(stderr, "Can't open dat file to read\n");
    exit(-1);
  }
  for (i=0;i<60;i++) {
    fscanf(fid, "%lf",&(input_data[i]));
    fscanf(fid, "%d",&(input_dir[i]));
  }
  close(fid);

  if((outfid = fopen("testout.dat", "w")) == NULL) {
    fprintf(stderr, "Can't open dat file to write\n");
    exit(-1);
  }

  for (i=0;i<60;i++) {
    fprintf(outfid, "%lf %lf\n",
          input_data[i],
          compose_force(&elbowmesh, input_data[i], input_dir[i]));
  }
  close(outfid);
}
```

```
void
initialize_mesh(Mesh *mesh, double angle)
{
  int i;

  mesh->num_trapezoids = 1;
  mesh->last_direction = UP;
  mesh->last_angle = angle;
  for (i=0; i<100 ; i++) {
    mesh->M[i] = 0.0;
    mesh->m[i] = 0.0;
  }

}


/* Compose_force is the central function for the hysteresis model.
   It maintains a history of input extrema (the interface I(t))
   and implements the wiping-out property when previous extrema
   are surpassed; it also calculates and returns the net
   hysteresis function value f(t) based on equation (3.10).
   Each mesh keeps its own interface record, so the function
   remains generic for all meshes.
*/

double
compose_force(Mesh *mesh, double theta, int direction)
{
  double calc_fab(Mesh mesh, double a, double b);
  double sum = 0.0;
  int k;

  /*  fprintf(stderr,"%d " ,mesh->num_trapezoids);
  fprintf(stderr,"%lf %lf %lf %lf %lf %lf " ,mesh->M[0],mesh->m[0],mesh-
>M[1],
        mesh->m[1],mesh->M[2], mesh->m[2]);
        */

  if (direction == UP) {

    /* If we're going up, check to see if we just changed direction */
    if (mesh->last_direction == DOWN) {
      mesh->m[mesh->num_trapezoids] = mesh->last_angle;
      mesh->num_trapezoids += 1;
      mesh->last_direction = UP;
    }
    mesh->last_angle = theta;

    /* Check for wiping-out property */
    if (mesh->num_trapezoids > 1) {
      if (theta > mesh->M[mesh->num_trapezoids-1]) {
      mesh->M[mesh->num_trapezoids-1] = 0;
      mesh->m[mesh->num_trapezoids-1] = 0;
      mesh->num_trapezoids -= 1;
      }
    }
    /* Compute the sum */
    for (k=1; k <= mesh->num_trapezoids-1; k++)
      sum += calc_fab(*mesh, mesh->M[k], mesh->m[k])
      - calc_fab(*mesh, mesh->M[k], mesh->m[k-1]);

    return (- mesh->max_saturation
          + sum
          + calc_fab(*mesh, theta, theta)
          - calc_fab(*mesh, theta, mesh->m[mesh->num_trapezoids-1]));
  }


  else if (direction == DOWN) {

    /* If we're going down, check to see if we just changed direction */
    if (mesh->last_direction == UP) {
      mesh->M[mesh->num_trapezoids] = mesh->last_angle;
      mesh->last_direction = DOWN;
    }
    mesh->last_angle = theta;

    /* Check for wiping-out property */
    if (mesh->num_trapezoids > 1) {
    if (theta < mesh->m[mesh->num_trapezoids-1]) {
      mesh->M[mesh->num_trapezoids] = 0;
      mesh->m[mesh->num_trapezoids-1] = 0;
      mesh->num_trapezoids -= 1;
    }
    }
    /* Compute the sum */
    for (k=1; k <= mesh->num_trapezoids-1; k++)
      sum += calc_fab(*mesh, mesh->M[k],
                mesh->m[k]) - calc_fab(*mesh, mesh->M[k], mesh->m[k-1]);

    return (- mesh->max_saturation
          + sum
          + calc_fab(*mesh, mesh->M[mesh->num_trapezoids], theta)
          - calc_fab(*mesh,
                 mesh->M[mesh->num_trapezoids],
                 mesh->m[mesh->num_trapezoids-1]));
  }


}

double
calc_fab(Mesh mesh, double a, double b)
{
  int i,j;
  int triangle = 0;

  for(i=0; mesh.rows[i].alphamin > a ; i++)
    ;
  for(j=0; mesh.rows[i].cells[j].betamax < b ; j++)
    ;
  if( j == mesh.rows[i].numcells-1)
    triangle = 1;
  if( !triangle )
    return(mesh.rows[i].cells[j].c1     +
          mesh.rows[i].cells[j].c2*a   +
          mesh.rows[i].cells[j].c3*b   +
          mesh.rows[i].cells[j].c4*a*b);
  else
    return(mesh.rows[i].cells[j].c1     +
          mesh.rows[i].cells[j].c2*a   +
          mesh.rows[i].cells[j].c3*b);
}



void
loadmesh(char *filename, Mesh *mesh)
{

  FILE *fp;
  int items_read, ncells, i;
  double amin, amax;
  Row *current_row;
  Cell *current_cell;

  if((fp = fopen(filename, "r")) == NULL) {
    fprintf(stderr, "Can't open mesh file\n");
    exit(-1);
  }

  mesh->numrows = 0;
  mesh->rows = NULL;

  fscanf(fp, "%lf",&(mesh->max_saturation));
  while((items_read = fscanf(fp,"%lf%lf%d",&amin,&amax,&ncells)) == 3) {

    mesh->numrows++;
    mesh->rows = realloc(mesh->rows, mesh->numrows*sizeof(Row));

    current_row = &((mesh->rows)[mesh->numrows-1]);

    current_row->alphamin = amin;
    current_row->alphamax = amax;
    current_row->numcells = ncells;

    current_row->cells = malloc(ncells*sizeof(Cell));

    for(i=0; i < ncells ; i++) {
      fscanf(fp,"%lf%lf%lf%lf%lf%lf",
          &((current_row->cells)[i].betamin),
          &((current_row->cells)[i].betamax),
          &((current_row->cells)[i].c1),
          &((current_row->cells)[i].c2),
          &((current_row->cells)[i].c3),
          &((current_row->cells)[i].c4));
    }
  }
}
```

Appendix E: Simulation Code

This appendix containts the C code that runs the SD/FAST simulation (compliant lower body). The other simulation cases are simplified versions of this code; unsuited conditions remove the suit constraints and the rigid body simulation prescribes the lower body to be still. For complete code of the other simulations, contact the EVA Dynamics Research Section of the Man-Vehicle Laboratory , MIT Rm. 37-219.

```
/*****************************************************************
With *extreme* gratitude to the programming efforts of:
Original author of simulation framework: Grant Schaffner, May 1995
Modified by Grant Schaffner Nov. 1996.
****************************************************************

Modified for EMU model and new task: David Rahn, 1996-1997

This simulation is modeled on the Spartan 204 mass manipulation
EVA performed on STS-63.
****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <jointcontrol.h>

#define UP 0
#define DOWN 1

/* Structure definitions for hysteresis model */
typedef struct cell_struct {
  double betamin;
  double betamax;
  double c1;
  double c2;
  double c3;
  double c4;
} Cell;

typedef struct row_struct {
  double alphamin;
  double alphamax;
  int    numcells;
  Cell   *cells;
} Row;

typedef struct mesh_struct {
  int last_direction;
  int num_trapezoids;
  double last_angle;
  double M[100];
  double m[100];
  double max_saturation;
  int numrows;
  Row *rows;
} Mesh;
```

```
/* Global meshes for hysteresis */
Mesh elbowmesh, shouldermesh, kneemesh;

/* Bodies (Frames) */
#define GND     -1
#define LLEGR    0
#define ULEGR    1
#define LLEGL    2
#define ULEGL    3
#define PELVR    4
#define PELVL    5
#define TRUNK    6
#define HEAD     7
#define UARMR    8
#define FARMR    9
#define HANDR   10
#define UARML   11
#define FARML   12
#define HANDL   13
#define HNDRG   14
#define HNDLG   15
#define PLSS    16
#define SPARTAN 17

/* Loop Joint Body Numbers */
#define LPELVR  18
#define LHANDR  19
#define LHANDL  20

/* State Variables */
#define NQ      54
#define NU      43
#define NEQ     (NQ+NU)
#define NJNT    21

/* Constraints */
#define NC      61
#define NLC     18

/* Integration Parameters */
#define DT .05
#define TOL 1e-7
#define CTOL 1e-5

/* Simulation Parameters */
#define MAX_VELOCITY 0.025
#define MOTION_TIME 10.0
#define NSTEP 200

/* Simple Linear fit for suit ankle */
#define ANKLE_A (-.068)

/* Declare external variable types */
int step, motionphase;
double xstart, ystart, pi, dtr, theta;
double ystore[NSTEP][NEQ], accel[NSTEP][NQ];
double time,state[NEQ];
```

```
/* Joint kinematics files */
char rank[30]="spart.rank",rkne[30]="spart.rkne",rhip[30]="spart.rhip",\
     lank[30]="spart.lank",lkne[30]="spart.lkne",lhip[30]="spart.lhip",\
     lumb[30]="spart.lumb",rshl[30]="spart.rshl",relb[30]="spart.relb",\
     rwrs[30]="spart.rwrs",lshl[30]="spart.lshl",lelb[30]="spart.lelb",\
     lwrs[30]="spart.lwrs";
FILE *rank_ptr,*rkne_ptr,*rhip_ptr,*lank_ptr,*lkne_ptr,*lhip_ptr,*lumb_ptr,\
     *rshl_ptr,*relb_ptr,*rwrs_ptr,*lshl_ptr,*lelb_ptr,*lwrs_ptr;

/* Joint torque files */
char
rankt[30]="spart.rankt",rknet[30]="spart.rknet",rhipt[30]="spart.rhipt",\

lankt[30]="spart.lankt",lknet[30]="spart.lknet",lhipt[30]="spart.lhipt",\

lumbt[30]="spart.lumbt",rshlt[30]="spart.rshlt",relbt[30]="spart.relbt",\

rwrst[30]="spart.rwrst",lshlt[30]="spart.lshlt",lelbt[30]="spart.lelbt",\
     lwrst[30]="spart.lwrst";
FILE
*rankt_ptr,*rknet_ptr,*rhipt_ptr,*lankt_ptr,*lknet_ptr,*lhipt_ptr,*lumbt_ptr
,\
     *rshlt_ptr,*relbt_ptr,*rwrst_ptr,*lshlt_ptr,*lelbt_ptr,*lwrst_ptr;

char *body[12] =
{"llegl","ulegl","llegr","ulegr","pelvis","trunk","uarml","farml",
     "handl","uarmr","farmr","handr"};

void printvals(int nval, double array[]);



void
main (void)
{
  void loadmesh(char *filename, Mesh *mesh);
  void initialize_mesh(Mesh *mesh, double angle);
  double compose_force(Mesh *mesh, double theta, int direction);

  extern double xstart, ystart;
  extern double pi, dtr, theta;

  int i, lock[NU], fcnt, err, counter;
  int j, flag;
  double t, y[NEQ], yinit[NEQ], dy[NEQ], qd[NQ], ud[NU];
  double com[3], pos[3], vel[3];
  double perrs[NC], verrs[NC];
  double q1,q2,q3,q4,q5,q6;

  loadmesh("elbow2.mesh", &elbowmesh);
  initialize_mesh(&elbowmesh, 10.0);

  loadmesh("shoulder2.mesh", &shouldermesh);
  initialize_mesh(&shouldermesh, 0.0);

  loadmesh("knee2.mesh", &kneemesh);
  initialize_mesh(&kneemesh, 47.0);
  pi = acos(-1.0);
  dtr = pi/180.0;

  openfiles();      /* For writing data */
  sdinit();         /* SD/FAST init.    */

/* Initial Configuration */

  /* Find xstart, ystart */
  q1 = 90.0*dtr; /* ankle    */ /* angle wrt RHS horizontal */
  q2 =  47.0*dtr; /* knee     */
  q3 = -52.0*dtr; /* hip      */
  q4 = 270.0*dtr; /* shoulder */
  q5 =  5.0*dtr; /* elbow    */
  q6 =  0.0*dtr; /* wrist    */

  xstart = 0.43*cos(q1)+0.43*cos(q1+q2)+0.575*cos(q1+q2+q3)
          +0.300*cos(q1+q2+q3+q4)+0.28*cos(q1+q2+q3+q4+q5)
          +0.05515*cos(q1+q2+q3+q4+q5+q6);
  ystart = 0.43*sin(q1)+0.43*sin(q1+q2)+0.575*sin(q1+q2+q3)
          +0.300*sin(q1+q2+q3+q4)+0.28*sin(q1+q2+q3+q4+q5)
          +0.05515*sin(q1+q2+q3+q4+q5+q6)-0.79985;

  initconds(t,y);  /* For astronaut    */

  preshandson();   /* Prescribed motion for assembly phase */
  motionphase = 1; /* Assembly/inverse kinematics */

  lockjoints(lock); /* locks all joints except arms */
  printf("done.\n"); /* end of initial conditions */

  t = 0.0;
  step = 0;
/* Do position and velocity analysis */

  printf("Assembly and Velocity Analysis......");

/* This function returns compatible position values in the state vector. */
/* Note: sdassemble calls sdumotion to obtain the prescribed motions. */

  sdassemble(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0){
     printf("\a\n Assembly failed! Error no.: %d\n",err);
     exit(-1);
   }

/* This function returns compatible velocity values in the state vector. */
/* Note: sdinitvel calls sdumotion to obtain the prescribed motions. */

  for (i = 0 ; i < NU ; i++) lock[i] = 0;
  sdinitvel(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0) printf("\a\n Velocity analysis failed! Error no.:
%d\n",err);

#ifdef DEBUG
  printf("\nPosition errors for the NC constraints (ctol=%f):\n",CTOL);
  printvals(NC,perrs);
  printf("\nVelocity errors for the NC constraints (ctol=%f):\n",CTOL);
  printvals(NC,verrs);
#endif

  printf("done.\n"); /* end of assembly & velocity analysis */

/* Kinematic Analysis. Loop calls sdmotion during each time step and stores
   the state vector in an array (neq x nsteps) until the desired duration
   has been achieved. sdmotion integrates the state vector over dt. */

  printf("Kinematics Analysis................");


  for (i = 0 ; i < NEQ ; i++) ystore[0][i] = y[i];
  for (i = 0 ; i < NEQ ; i++) dy[i] = 0.0;

  flag = 0;


  /* Kinematics loop */

  for(step = 1 ; step < NSTEP ; step++)
    {
      for (i = 0 ; i < NEQ ; i++) ystore[step][i] = y[i];
      for (i = 0 ; i < NU ; i++) accel[step][i] = dy[NQ+i];

      printkinem(t,y,dy);

      /* Perform motion integration. */

      sdmotion(&t,y,dy,DT,CTOL,TOL,&flag,&err);
      if(err != 0){
      printf("\a\n Problem with integrator! Error no.: %d\n",err);
      exit();
      }
    }

  printf("done.\n"); /* end of kinematics analysis */


  /* Inverse Dynamics Analysis */

  printf("Inverse Dynamics...................");

  /* Calculation of joint torques from prescribed kinematics. */

  t=0.0;

  /* Turn prescribed motion ON for internal body joints. */

  presbodyon();
  motionphase = 2;  /* indicates inverse dynamics phase for sdumotion */

  for(step = 0 ; step < NSTEP ; step++){
    for (i = 0 ; i < NEQ ; i++) y[i] = ystore[step][i];

    for(i = 0 ; i < NQ ; i++) q[i]=y[i];
    for(i = 0 ; i < NU ; i++) u[i]=y[NQ+i];
    sdstate(t,&y[0],&y[NQ]);

    sduforce(t,q,u);
    sdumotion(t,q,u);

    sdderiv(dy,&dy[NQ]);

    /* Obtain joint torque values and print to data files */

    torqvals(t);

    t=t+DT;
  }
  fprintf(stderr,"done.\n\n\a\a\a"); /* end of inverse dynamics */
}




/*--------------------------------------------------------------------------
-*/
/*                              Functions
*/
/*--------------------------------------------------------------------------
-*/

/* Opens all the joint kinematics and torque data files.
   (G. Schaffner)
*/
openfiles()
{
  /* Joint kinematics data files */
  rank_ptr = fopen(rank,"w");
  if(rank_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  rkne_ptr = fopen(rkne,"w");
  if(rkne_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  rhip_ptr = fopen(rhip,"w");
  if(rhip_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lank_ptr = fopen(lank,"w");
  if(lank_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lkne_ptr = fopen(lkne,"w");
  if(lkne_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lhip_ptr = fopen(lhip,"w");
```

```
if(lhip_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
lumb_ptr = fopen(lumb,"w");
if(lumb_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
rshl_ptr = fopen(rshl,"w");
if(rshl_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
relb_ptr = fopen(relb,"w");
if(relb_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
rwrs_ptr = fopen(rwrs,"w");
if(rwrs_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
lshl_ptr = fopen(lshl,"w");
if(lshl_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
lelb_ptr = fopen(lelb,"w");
if(lelb_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
lwrs_ptr = fopen(lwrs,"w");
if(lwrs_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}

/* Joint torque data files */

rankt_ptr = fopen(rankt,"w");
if(rankt_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
rknet_ptr = fopen(rknet,"w");
if(rknet_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
rhipt_ptr = fopen(rhipt,"w");
if(rhipt_ptr == NULL) {
  printf("Oops. Unable to open file. Exiting...\n");
  exit(-1);
}
lankt_ptr = fopen(lankt,"w");
```

```
  if(lankt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lknet_ptr = fopen(lknet,"w");
  if(lknet_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lhipt_ptr = fopen(lhipt,"w");
  if(lhipt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lumbt_ptr = fopen(lumbt,"w");
  if(lumbt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  rshlt_ptr = fopen(rshlt,"w");
  if(rshlt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  relbt_ptr = fopen(relbt,"w");
  if(relbt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  rwrst_ptr = fopen(rwrst,"w");
  if(rwrst_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lshlt_ptr = fopen(lshlt,"w");
  if(lshlt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lelbt_ptr = fopen(lelbt,"w");
  if(lelbt_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
  lwrst_ptr = fopen(lwrst,"w");
  if(lwrst_ptr == NULL) {
    printf("Oops. Unable to open file. Exiting...\n");
    exit(-1);
  }
}


/*******************************************************
 Print the first line (headers) of each output data file.
 (G. Schaffner)
 *******************************************************/

printhdrs()
```

```
{

  /* Kinematics headers */


fprintf(rank_ptr,"Time\t\tRankRolP\tRankYawP\tRankPchP\tRankRolV\tRankYawV\t
RankPchV\tRankRolA\tRankYawA\tRankPchA\n");
  fprintf(rkne_ptr,"Time\t\tRknePchP\tRknePchV\tRknePchA\n");

fprintf(rhip_ptr,"Time\t\tRhipRolP\tRhipYawP\tRhipPchP\tRhipRolV\tRhipYawV\t
RhipPchV\tRhipRolA\tRhipYawA\tRhipPchA\n");

fprintf(lank_ptr,"Time\t\tLankRolP\tLankYawP\tLankPchP\tLankRolV\tLankYawV\t
LankPchV\tLankRolA\tLankYawA\tLankPchA\n");
  fprintf(lkne_ptr,"Time\t\tLknePchP\tLknePchV\tLknePchA\n");

fprintf(lhip_ptr,"Time\t\tLhipRolP\tLhipYawP\tLhipPchP\tLhipRolV\tLhipYawV\t
LhipPchV\tLhipRolA\tLhipYawA\tLhipPchA\n");

fprintf(lumb_ptr,"Time\t\tlumbRolP\tlumbYawP\tlumbPchP\tlumbRolV\tlumbYawV\t
lumbPchV\tlumbRolA\tlumbYawA\tlumbPchA\n");

fprintf(rshl_ptr,"Time\t\tRshlRolP\tRshlYawP\tRshlPchP\tRshlRolV\tRshlYawV\t
RshlPchV\tRshlRolA\tRshlYawA\tRshlPchA\n");
  fprintf(relb_ptr,"Time\t\tRelbPchP\tRelbPchV\tRelbPchA\n");

fprintf(rwrs_ptr,"Time\t\tRwrsRolP\tRwrsYawP\tRwrsPchP\tRwrsRolV\tRwrsYawV\t
RwrsPchV\tRwrsRolA\tRwrsYawA\tRwrsPchA\n");

fprintf(lshl_ptr,"Time\t\tLshlRolP\tLshlYawP\tLshlPchP\tLshlRolV\tLshlYawV\t
LshlPchV\tLshlRolA\tLshlYawA\tLshlPchA\n");
  fprintf(lelb_ptr,"Time\t\tLelbPchP\tLelbPchV\tLelbPchA\n");

fprintf(lwrs_ptr,"Time\t\tLwrsRolP\tLwrsYawP\tLwrsPchP\tLwrsRolV\tLwrsYawV\t
LwrsPchV\tLwrsRolA\tLwrsYawA\tLwrsPchA\n");


  /* Torque headers */

  fprintf(rankt_ptr,"Time\t\tRankRolT\tRankYawT\tRankPchT\n");
  fprintf(rknet_ptr,"Time\t\tRknePchT\n");
  fprintf(rhipt_ptr,"Time\t\tRhipRolT\tRhipYawT\tRhipPchT\n");
  fprintf(lankt_ptr,"Time\t\tLankRolT\tLankYawT\tLankPchT\n");
  fprintf(lknet_ptr,"Time\t\tLknePchT\n");
  fprintf(lhipt_ptr,"Time\t\tLhipRolT\tLhipYawT\tLhipPchT\n");
  fprintf(lumbt_ptr,"Time\t\tlumbRolT\tlumbYawT\tlumbPchT\n");
  fprintf(rshlt_ptr,"Time\t\tRshlRolT\tRshlYawT\tRshlPchT\n");
  fprintf(relbt_ptr,"Time\t\tRelbPchT\n");
  fprintf(rwrst_ptr,"Time\t\tRwrsRolT\tRwrsYawT\tRwrsPchT\n");
  fprintf(lshlt_ptr,"Time\t\tLshlRolT\tLshlYawT\tLshlPchT\n");
  fprintf(lelbt_ptr,"Time\t\tLelbPchT\n");
  fprintf(lwrst_ptr,"Time\t\tLwrsRolT\tLwrsYawT\tLwrsPchT\n");
}


/*************************************************************
   Print the position, velocity, and acceleration values of
   joint degrees of freedom to joint data files. (G. Schaffner)
   *********************************************************/

printkinem(t,y,dy)
double t,y[],dy[];
{
  sdst2ang(y,q_ang); /* ok to use y[], rest of state variable is ignored */

  fprintf(rank_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(LLEGR,0)],q_ang[sdindx(LLEGR,1)],q_ang[sdindx(LLEGR,2)],\
        y[NQ+sdindx(LLEGR,0)],y[NQ+sdindx(LLEGR,1)],y[NQ+sdindx(LLEGR,2)],\

dy[NQ+sdindx(LLEGR,0)],dy[NQ+sdindx(LLEGR,1)],dy[NQ+sdindx(LLEGR,2)]);

  fprintf(rkne_ptr,"%f\t%f\t%f\t%f\n",t,\
        y[sdindx(ULEGR,0)],y[NQ+sdindx(ULEGR,0)],dy[NQ+sdindx(ULEGR,0)]);

  fprintf(rhip_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(PELVR,0)],q_ang[sdindx(PELVR,1)],q_ang[sdindx(PELVR,2)],\
        y[NQ+sdindx(PELVR,0)],y[NQ+sdindx(PELVR,1)],y[NQ+sdindx(PELVR,2)],\

dy[NQ+sdindx(PELVR,0)],dy[NQ+sdindx(PELVR,1)],dy[NQ+sdindx(PELVR,2)]);

  fprintf(lank_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(LLEGL,0)],q_ang[sdindx(LLEGL,1)],q_ang[sdindx(LLEGL,2)],\
        y[NQ+sdindx(LLEGL,0)],y[NQ+sdindx(LLEGL,1)],y[NQ+sdindx(LLEGL,2)],\

dy[NQ+sdindx(LLEGL,0)],dy[NQ+sdindx(LLEGL,1)],dy[NQ+sdindx(LLEGL,2)]);

  fprintf(lkne_ptr,"%f\t%f\t%f\t%f\n",t,\
        y[sdindx(ULEGL,0)],y[NQ+sdindx(ULEGL,0)],dy[NQ+sdindx(ULEGL,0)]);

  fprintf(lhip_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(PELVL,0)],q_ang[sdindx(PELVL,1)],q_ang[sdindx(PELVL,2)],\
        y[NQ+sdindx(PELVL,0)],y[NQ+sdindx(PELVL,1)],y[NQ+sdindx(PELVL,2)],\

dy[NQ+sdindx(PELVL,0)],dy[NQ+sdindx(PELVL,1)],dy[NQ+sdindx(PELVL,2)]);

  fprintf(lumb_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(TRUNK,0)],q_ang[sdindx(TRUNK,1)],q_ang[sdindx(TRUNK,2)],\
        y[NQ+sdindx(TRUNK,0)],y[NQ+sdindx(TRUNK,1)],y[NQ+sdindx(TRUNK,2)],\

dy[NQ+sdindx(TRUNK,0)],dy[NQ+sdindx(TRUNK,1)],dy[NQ+sdindx(TRUNK,2)]);

  fprintf(rshl_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(UARMR,0)],q_ang[sdindx(UARMR,1)],q_ang[sdindx(UARMR,2)],\
        y[NQ+sdindx(UARMR,0)],y[NQ+sdindx(UARMR,1)],y[NQ+sdindx(UARMR,2)],\

dy[NQ+sdindx(UARMR,0)],dy[NQ+sdindx(UARMR,1)],dy[NQ+sdindx(UARMR,2)]);

  fprintf(relb_ptr,"%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(FARMR,0)],y[NQ+sdindx(FARMR,0)],dy[NQ+sdindx(FARMR,0)]);
```

```
  fprintf(rwrs_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(HANDR,0)],q_ang[sdindx(HANDR,1)],q_ang[sdindx(HANDR,2)],\
        y[NQ+sdindx(HANDR,0)],y[NQ+sdindx(HANDR,1)],y[NQ+sdindx(HANDR,2)],\

dy[NQ+sdindx(HANDR,0)],dy[NQ+sdindx(HANDR,1)],dy[NQ+sdindx(HANDR,2)]);

  fprintf(lshl_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(UARML,0)],q_ang[sdindx(UARML,1)],q_ang[sdindx(UARML,2)],\
        y[NQ+sdindx(UARML,0)],y[NQ+sdindx(UARML,1)],y[NQ+sdindx(UARML,2)],\

dy[NQ+sdindx(UARML,0)],dy[NQ+sdindx(UARML,1)],dy[NQ+sdindx(UARML,2)]);

  fprintf(lelb_ptr,"%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(FARML,0)],y[NQ+sdindx(FARML,0)],dy[NQ+sdindx(FARML,0)]);

  fprintf(lwrs_ptr,"%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n",t,\

q_ang[sdindx(HANDL,0)],q_ang[sdindx(HANDL,1)],q_ang[sdindx(HANDL,2)],\
        y[NQ+sdindx(HANDL,0)],y[NQ+sdindx(HANDL,1)],y[NQ+sdindx(HANDL,2)],\

dy[NQ+sdindx(HANDL,0)],dy[NQ+sdindx(HANDL,1)],dy[NQ+sdindx(HANDL,2)]);
}


/*****************************************************************************
*
   Extract the joint torque values (using sdgetht()) and prints
   these to the joint torque data files. (G. Schaffner)
*****************************************************************************
**/

torqvals(t)
double t;
{
double trq_rank[3],trq_rkne,trq_rhip[3],trq_lank[3],trq_lkne,trq_lhip[3],\
       trq_lumb[3],trq_rshl[3],trq_relb,trq_rwrs[3],trq_lshl[3],\
       trq_lelb,trq_lwrs[3];

/* double tmn[NQ], tmx[NQ]; */

  sdgetht(LLEGR,0,&trq_rank[0]);
  sdgetht(LLEGR,1,&trq_rank[1]);
  sdgetht(LLEGR,2,&trq_rank[2]);

  sdgetht(ULEGR,0,&trq_rkne);

  sdgetht(PELVR,0,&trq_rhip[0]);
  sdgetht(PELVR,1,&trq_rhip[1]);
  sdgetht(PELVR,2,&trq_rhip[2]);

  sdgetht(LLEGL,0,&trq_lank[0]);
  sdgetht(LLEGL,1,&trq_lank[1]);
  sdgetht(LLEGL,2,&trq_lank[2]);
  sdgetht(ULEGL,0,&trq_lkne);

  sdgetht(PELVL,0,&trq_lhip[0]);
  sdgetht(PELVL,1,&trq_lhip[1]);
  sdgetht(PELVL,2,&trq_lhip[2]);

  sdgetht(TRUNK,0,&trq_lumb[0]);
  sdgetht(TRUNK,1,&trq_lumb[1]);
  sdgetht(TRUNK,2,&trq_lumb[2]);

  sdgetht(UARMR,0,&trq_rshl[0]);
  sdgetht(UARMR,1,&trq_rshl[1]);
  sdgetht(UARMR,2,&trq_rshl[2]);

  sdgetht(FARMR,2,&trq_relb);

  sdgetht(HANDR,0,&trq_rwrs[0]);
  sdgetht(HANDR,1,&trq_rwrs[1]);
  sdgetht(HANDR,2,&trq_rwrs[2]);

  sdgetht(UARML,0,&trq_lshl[0]);
  sdgetht(UARML,1,&trq_lshl[1]);
  sdgetht(UARML,2,&trq_lshl[2]);

  sdgetht(FARML,2,&trq_lelb);

  sdgetht(HANDL,0,&trq_lwrs[0]);
  sdgetht(HANDL,1,&trq_lwrs[1]);
  sdgetht(HANDL,2,&trq_lwrs[2]);


fprintf(rankt_ptr,"%f\t%f\t%f\t%f\n",t,trq_rank[0],trq_rank[1],trq_rank[2]);
  fprintf(rknet_ptr,"%f\t%f\n",t,trq_rkne);

fprintf(rhipt_ptr,"%f\t%f\t%f\t%f\n",t,trq_rhip[0],trq_rhip[1],trq_rhip[2]);

fprintf(lankt_ptr,"%f\t%f\t%f\t%f\n",t,trq_lank[0],trq_lank[1],trq_lank[2]);
  fprintf(lknet_ptr,"%f\t%f\n",t,trq_lkne);

fprintf(lhipt_ptr,"%f\t%f\t%f\t%f\n",t,trq_lhip[0],trq_lhip[1],trq_lhip[2]);

fprintf(lumbt_ptr,"%f\t%f\t%f\t%f\n",t,trq_lumb[0],trq_lumb[1],trq_lumb[2]);

fprintf(rshlt_ptr,"%f\t%f\t%f\t%f\n",t,trq_rshl[0],trq_rshl[1],trq_rshl[2]);
  fprintf(relbt_ptr,"%f\t%f\n",t,trq_relb);

fprintf(rwrst_ptr,"%f\t%f\t%f\t%f\n",t,trq_rwrs[0],trq_rwrs[1],trq_rwrs[2]);

fprintf(lshlt_ptr,"%f\t%f\t%f\t%f\n",t,trq_lshl[0],trq_lshl[1],trq_lshl[2]);
  fprintf(lelbt_ptr,"%f\t%f\n",t,trq_lelb);

fprintf(lwrst_ptr,"%f\t%f\t%f\t%f\n",t,trq_lwrs[0],trq_lwrs[1],trq_lwrs[2]);
}


/************************************************************
   Turns prescribed motion ON for 6-dof joints connecting
   hands to ground, and OFF for internal body joints.
   (G. Schaffner)
**************************************************************/

preshandson()
{

  /* hand 6-dof joints ON */

  sdpres(HNDRG,0,ON); sdpres(HNDRG,1,ON); sdpres(HNDRG,2,ON);
  sdpres(HNDRG,3,ON); sdpres(HNDRG,4,ON); sdpres(HNDRG,5,ON);

  sdpres(HNDLG,0,ON); sdpres(HNDLG,1,ON); sdpres(HNDLG,2,ON);
  sdpres(HNDLG,3,ON); sdpres(HNDLG,4,ON); sdpres(HNDLG,5,ON);
  /*
  sdpres(HNDRG,0,OFF); sdpres(HNDRG,1,OFF); sdpres(HNDRG,2,OFF);
  sdpres(HNDRG,3,OFF); sdpres(HNDRG,4,OFF); sdpres(HNDRG,5,OFF);

  sdpres(HNDLG,0,OFF); sdpres(HNDLG,1,OFF); sdpres(HNDLG,2,OFF);
  sdpres(HNDLG,3,OFF); sdpres(HNDLG,4,OFF); sdpres(HNDLG,5,OFF);
  */
  /* internal body joints OFF */

  sdpres(LLEGR,0,OFF); sdpres(LLEGR,1,OFF); sdpres(LLEGR,2,OFF);
  sdpres(ULEGR,0,OFF);
  sdpres(LLEGL,0,OFF); sdpres(LLEGL,1,OFF); sdpres(LLEGL,2,OFF);
  sdpres(ULEGL,0,OFF);
  sdpres(PELVR,0,OFF); sdpres(PELVR,1,OFF); sdpres(PELVR,2,OFF);
  sdpres(PELVL,0,OFF); sdpres(PELVL,1,OFF); sdpres(PELVL,2,OFF);
  sdpres(TRUNK,0,OFF); sdpres(TRUNK,1,OFF); sdpres(TRUNK,2,OFF);
  sdpres(UARMR,0,OFF); sdpres(UARMR,1,OFF); sdpres(UARMR,2,OFF);
  sdpres(FARMR,0,OFF);
  sdpres(HANDR,0,OFF); sdpres(HANDR,1,OFF); sdpres(HANDR,2,OFF);
  sdpres(UARML,0,OFF); sdpres(UARML,1,OFF); sdpres(UARML,2,OFF);
  sdpres(FARML,0,OFF);
  sdpres(HANDL,0,OFF); sdpres(HANDL,1,OFF); sdpres(HANDL,2,OFF);
}

/***************************************************************************
*
   Turns prescribed motion ON for internal body joints,
   and OFF for joints connecting hands to ground.
   (G. Schaffner)
***************************************************************************
*/

presbodyon()
{

  /* internal body joints ON */
  sdpres(LLEGR,0,ON); sdpres(LLEGR,1,ON); sdpres(LLEGR,2,ON);
  sdpres(ULEGR,0,ON);
  sdpres(LLEGL,0,ON); sdpres(LLEGL,1,ON); sdpres(LLEGL,2,ON);
  sdpres(ULEGL,0,ON);
  sdpres(PELVR,0,ON); sdpres(PELVR,1,ON); sdpres(PELVR,2,ON);
  sdpres(PELVL,0,ON); sdpres(PELVL,1,ON); sdpres(PELVL,2,ON);
  sdpres(TRUNK,0,ON); sdpres(TRUNK,1,ON); sdpres(TRUNK,2,ON);
  sdpres(UARMR,0,ON); sdpres(UARMR,1,ON); sdpres(UARMR,2,ON);
  sdpres(FARMR,0,ON);
  sdpres(HANDR,0,ON); sdpres(HANDR,1,ON); sdpres(HANDR,2,ON);
  sdpres(UARML,0,ON); sdpres(UARML,1,ON); sdpres(UARML,2,ON);
  sdpres(FARML,0,ON);
  sdpres(HANDL,0,ON); sdpres(HANDL,1,ON); sdpres(HANDL,2,ON);

  /* hand 6-dof joints OFF (loop joints) */

  sdpres(HNDRG,0,OFF); sdpres(HNDRG,1,OFF); sdpres(HNDRG,2,OFF);
  sdpres(HNDRG,3,OFF); sdpres(HNDRG,4,OFF); sdpres(HNDRG,5,OFF);

  sdpres(HNDLG,0,OFF); sdpres(HNDLG,1,OFF); sdpres(HNDLG,2,OFF);
  sdpres(HNDLG,3,OFF); sdpres(HNDLG,4,OFF); sdpres(HNDLG,5,OFF);

}


/****************************************************************************
*

   sduforce

   This routine computes and applies the forces acting in the system. It
must
   always be included, when the Simplified Analysis Routines are used,
because
   they will make calls to sduforce. This is the case even if the sdforce
   function is unused (empty).

****************************************************************************
*/

sduforce(t,q,u)
double t,q[NQ],u[NU];
{

  double compose_force(Mesh *mesh, double theta, int direction);

  double qang[NU];
  double trq_rank[3],trq_rkne,trq_rhip[3],trq_rshl[3],trq_relb,trq_rwrs[3];
  double trq_lumb[3];
  double trq_lank[3],trq_lkne,trq_lhip[3],trq_lshl[3],trq_lelb,trq_lwrs[3];

  double suit_elbow_torque,suit_elbow_damping;
  double suit_shoulder_torque,suit_shoulder_damping;
  double suit_knee_torque, suit_knee_damping;
  double uarm_pos,farm_pos,uleg_pos,ankle_pos;
  double suit_ankle_torque;

  sdst2ang(q,qang);

  /* Assume symmetric for now */
  farm_pos = qang[sdindx(FARMR,0)]*180.0/M_PI ;
  uarm_pos = (qang[sdindx(UARMR,2)]*180.0/M_PI)-90.0 ;
  uleg_pos = qangsdindx(ULEGR,0)]*180.0/M_PI ;
  ankle_pos = qang[sdindx(LLEGR,2)]*180.0/M_PI ;

  /* Control of Lower Body Joints */
  /* ANKLE */

  /* The passive ankle torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

       /* X-axis */

  if(qang[sdindx(LLEGR,0)] < ANK_X_MIN*dtr)
    trq_rank[0] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,0)]-(ANK_X_MIN)*dtr);
  if(qang[sdindx(LLEGR,0)] > ANK_X_MIN*dtr && qang[sdindx(LLEGR,0)] <
ANK_X_MAX*dtr)
    trq_rank[0] = -KROT_ANK*(qang[sdindx(LLEGR,0)]-(BIAS_RANK_X))
               -BDAMP_ANK*u[sdindx(LLEGR,0)];
  if(qang[sdindx(LLEGR,0)] > ANK_X_MAX*dtr)
    trq_rank[0] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,0)]-(ANK_X_MAX)*dtr);

  if(qang[sdindx(LLEGL,0)] < ANK_X_MIN*dtr)
    trq_lank[0] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,0)]-(ANK_X_MIN)*dtr);
  if(qang[sdindx(LLEGL,0)] > ANK_X_MIN*dtr && qang[sdindx(LLEGL,0)] <
ANK_X_MAX*dtr)
    trq_lank[0] = -KROT_ANK*(qang[sdindx(LLEGL,0)]-(BIAS_LANK_X))
               -BDAMP_ANK*u[sdindx(LLEGL,0)];
  if(qang[sdindx(LLEGL,0)] > ANK_X_MAX*dtr)
    trq_lank[0] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,0)]-(ANK_X_MAX)*dtr);

  sdhinget(LLEGR,0,trq_rank[0]);
  sdhinget(LLEGL,0,trq_lank[0]);

       /* Y-axis */

  if(qang[sdindx(LLEGR,1)] < ANK_Y_MIN*dtr)
    trq_rank[1] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,1)]-(ANK_Y_MIN)*dtr);
  if(qang[sdindx(LLEGR,1)] > ANK_Y_MIN*dtr && qang[sdindx(LLEGR,1)] <
ANK_Y_MAX*dtr)
    trq_rank[1] = -KROT_ANK*(qang[sdindx(LLEGR,1)]-(BIAS_RANK_Y))
               -BDAMP_ANK*u[sdindx(LLEGR,1)];
  if(qang[sdindx(LLEGR,1)] > ANK_Y_MAX*dtr)
    trq_rank[1] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,1)]-(ANK_Y_MAX)*dtr);

  if(qang[sdindx(LLEGL,1)] < ANK_Y_MIN*dtr)
    trq_lank[1] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,1)]-(ANK_Y_MIN)*dtr);
  if(qang[sdindx(LLEGL,1)] > ANK_Y_MIN*dtr && qang[sdindx(LLEGL,1)] <
ANK_Y_MAX*dtr)
    trq_lank[1] = -KROT_ANK*(qang[sdindx(LLEGL,1)]-(BIAS_LANK_Y))
               -BDAMP_ANK*u[sdindx(LLEGL,1)];
  if(qang[sdindx(LLEGL,1)] > ANK_Y_MAX*dtr)
    trq_lank[1] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,1)]-(ANK_Y_MAX)*dtr);

  sdhinget(LLEGR,1,trq_rank[1]);
  sdhinget(LLEGL,1,trq_lank[1]);

       /* Z-axis */

  if(qang[sdindx(LLEGR,2)] < ANK_Z_MIN*dtr)
    trq_rank[2] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,2)]-(ANK_Z_MIN)*dtr);
  if(qang[sdindx(LLEGR,2)] > ANK_Z_MIN*dtr && qang[sdindx(LLEGR,2)] <
ANK_Z_MAX*dtr)
```

```
    trq_rank[2] = -KROT_ANK*(qang[sdindx(LLEGR,2)]-(BIAS_RANK_Z))
                -BDAMP_ANK*u[sdindx(LLEGR,2)];
  if(qang[sdindx(LLEGR,2)] > ANK_Z_MAX*dtr)
    trq_rank[2] = -KROT_ANK_STOP*(qang[sdindx(LLEGR,2)]-(ANK_Z_MAX)*dtr);

  if(qang[sdindx(LLEGL,2)] < ANK_Z_MIN*dtr)
    trq_lank[2] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,2)]-(ANK_Z_MIN)*dtr);
  if(qang[sdindx(LLEGL,2)] > ANK_Z_MIN*dtr && qang[sdindx(LLEGL,2)] <
ANK_Z_MAX*dtr)
    trq_lank[2] = -KROT_ANK*(qang[sdindx(LLEGL,2)]-(BIAS_LANK_Z))
                -BDAMP_ANK*u[sdindx(LLEGL,2)];
  if(qang[sdindx(LLEGL,2)] > ANK_Z_MAX*dtr)
    trq_lank[2] = -KROT_ANK_STOP*(qang[sdindx(LLEGL,2)]-(ANK_Z_MAX)*dtr);

  sdhinget(LLEGR,2,trq_rank[2]);
  sdhinget(LLEGL,2,trq_lank[2]);

#ifdef DEBUG
  printf("\nt=%f\tR Ankle Trq = %f\tL Ankle Trq =
%f",t,trq_rank[2],trq_lank[2]); */
#endif
  /* KNEE */

  /* The passive knee torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  if(qang[sdindx(ULEGR,0)] < KNEEMIN*dtr)
    trq_rkne = -KROT_KNE_STOP*(qang[sdindx(ULEGR,0)]-(KNEEMIN)*dtr);
  if(qang[sdindx(ULEGR,0)] > KNEEMIN*dtr && qang[sdindx(ULEGR,0)] <
KNEEMAX*dtr)
    trq_rkne = -KROT_KNE*(qang[sdindx(ULEGR,0)]-(BIAS_RKNE))
                -BDAMP_KNE*u[sdindx(ULEGR,0)];
  if(qang[sdindx(ULEGR,0)] > KNEEMAX*dtr)
    trq_rkne = -KROT_KNE_STOP*(qang[sdindx(ULEGR,0)]-(KNEEMAX)*dtr);

  if(qang[sdindx(ULEGL,0)] < KNEEMIN*dtr)
    trq_lkne = -KROT_KNE_STOP*(qang[sdindx(ULEGL,0)]-(KNEEMIN)*dtr);
  if(qang[sdindx(ULEGL,0)] > KNEEMIN*dtr && qang[sdindx(ULEGL,0)] <
KNEEMAX*dtr)
    trq_lkne = -KROT_KNE*(qang[sdindx(ULEGL,0)]-(BIAS_LKNE))
                -BDAMP_KNE*u[sdindx(ULEGL,0)];
  if(qang[sdindx(ULEGL,0)] > KNEEMAX*dtr)
    trq_lkne = -KROT_KNE_STOP*(qang[sdindx(ULEGL,0)]-(KNEEMAX)*dtr);

  sdhinget(ULEGR,0,trq_rkne);
  sdhinget(ULEGL,0,trq_lkne);

#ifdef DEBUG
    printf("\nt=%f\tR Knee Trq = %f\tL Knee Trq = %f",t,trq_rkne,trq_lkne);
*/
#endif

  /* HIP */

  /* The passive hip torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  /* X-axis */
  if(qang[sdindx(PELVR,0)] < HIP_X_MIN*dtr)
    trq_rhip[0] = -KROT_HIP_STOP*(qang[sdindx(PELVR,0)]-(HIP_X_MIN)*dtr);
  if(qang[sdindx(PELVR,0)] > HIP_X_MIN*dtr && qang[sdindx(PELVR,0)] <
HIP_X_MAX*dtr)
    trq_rhip[0] = -KROT_HIP*(qang[sdindx(PELVR,0)]-(BIAS_RHIP_X))
               -BDAMP_HIP*u[sdindx(PELVR,0)];
  if(qang[sdindx(PELVR,0)] > HIP_X_MAX*dtr)
    trq_rhip[0] = -KROT_HIP_STOP*(qang[sdindx(PELVR,0)]-(HIP_X_MAX)*dtr);

  if(qang[sdindx(PELVL,0)] < HIP_X_MIN*dtr)
    trq_lhip[0] = -KROT_HIP_STOP*(qang[sdindx(PELVL,0)]-(HIP_X_MIN)*dtr);
  if(qang[sdindx(PELVL,0)] > HIP_X_MIN*dtr && qang[sdindx(PELVL,0)] <
HIP_X_MAX*dtr)
    trq_lhip[0] = -KROT_HIP*(qang[sdindx(PELVL,0)]-(BIAS_LHIP_X))
               -BDAMP_HIP*u[sdindx(PELVL,0)];
  if(qang[sdindx(PELVL,0)] > HIP_X_MAX*dtr)
    trq_lhip[0] = -KROT_HIP_STOP*(qang[sdindx(PELVL,0)]-(HIP_X_MAX)*dtr);

  sdhinget(PELVR,0,trq_rhip[0]);
  sdhinget(PELVL,0,trq_lhip[0]);

  /* Y-axis */

  if(qang[sdindx(PELVR,1)] < HIP_Y_MIN*dtr)
    trq_rhip[1] = -KROT_HIP_STOP*(qang[sdindx(PELVR,1)]-(HIP_Y_MIN)*dtr);
  if(qang[sdindx(PELVR,1)] > HIP_Y_MIN*dtr && qang[sdindx(PELVR,1)] <
HIP_Y_MAX*dtr)
    trq_rhip[1] = -KROT_HIP*(qang[sdindx(PELVR,1)]-(BIAS_RHIP_Y))
               -BDAMP_HIP*u[sdindx(PELVR,1)];
  if(qang[sdindx(PELVR,1)] > HIP_Y_MAX*dtr)
    trq_rhip[1] = -KROT_HIP_STOP*(qang[sdindx(PELVR,1)]-(HIP_Y_MAX)*dtr);

  if(qang[sdindx(PELVL,1)] < HIP_Y_MIN*dtr)
    trq_lhip[1] = -KROT_HIP_STOP*(qang[sdindx(PELVL,1)]-(HIP_Y_MIN)*dtr);
  if(qang[sdindx(PELVL,1)] > HIP_Y_MIN*dtr && qang[sdindx(PELVL,1)] <
HIP_Y_MAX*dtr)
    trq_lhip[1] = -KROT_HIP*(qang[sdindx(PELVL,1)]-(BIAS_LHIP_Y))
               -BDAMP_HIP*u[sdindx(PELVL,1)];
  if(qang[sdindx(PELVL,1)] > HIP_Y_MAX*dtr)
    trq_lhip[1] = -KROT_HIP_STOP*(qang[sdindx(PELVL,1)]-(HIP_Y_MAX)*dtr);

  sdhinget(PELVR,1,trq_rhip[1]);
  sdhinget(PELVL,1,trq_lhip[1]);

  /* Z-axis */

  if(qang[sdindx(PELVR,2)] < HIP_Z_MIN*dtr)
    trq_rhip[2] = -KROT_HIP_STOP*(qang[sdindx(PELVR,2)]-(HIP_Z_MIN)*dtr);
  if(qang[sdindx(PELVR,2)] > HIP_Z_MIN*dtr && qang[sdindx(PELVR,2)] <
HIP_Z_MAX*dtr)
    trq_rhip[2] = -KROT_HIP*(qang[sdindx(PELVR,2)]-(BIAS_RHIP_Z))
               -BDAMP_HIP*u[sdindx(PELVR,2)];
  if(qang[sdindx(PELVR,2)] > HIP_Z_MAX*dtr)
    trq_rhip[2] = -KROT_HIP_STOP*(qang[sdindx(PELVR,2)]-(HIP_Z_MAX)*dtr);

  if(qang[sdindx(PELVL,2)] < HIP_Z_MIN*dtr)
    trq_lhip[2] = -KROT_HIP_STOP*(qang[sdindx(PELVL,2)]-(HIP_Z_MIN)*dtr);
  if(qang[sdindx(PELVL,2)] > HIP_Z_MIN*dtr && qang[sdindx(PELVL,2)] <
HIP_Z_MAX*dtr)
    trq_lhip[2] = -KROT_HIP*(qang[sdindx(PELVL,2)]-(BIAS_LHIP_Z))
                -BDAMP_HIP*u[sdindx(PELVL,2)];
  if(qang[sdindx(PELVL,2)] > HIP_Z_MAX*dtr)
    trq_lhip[2] = -KROT_HIP_STOP*(qang[sdindx(PELVL,2)]-(HIP_Z_MAX)*dtr);

  sdhinget(PELVR,2,trq_rhip[2]);
  sdhinget(PELVL,2,trq_lhip[2]);

#ifdef DEBUG
    printf("\nt=%f\tR Hip Trq = %f\tL Hip Trq =
%f",t,trq_rhip[2],trq_lhip[2]); */
#endif

  /* LUMBAR */

  /* The passive lumbar torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  /* X-axis */

  if(qang[sdindx(TRUNK,0)] < LUM_X_MIN*dtr)
    trq_lumb[0] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,0)]-(LUM_X_MIN)*dtr);
  if(qang[sdindx(TRUNK,0)] > LUM_X_MIN*dtr && qang[sdindx(TRUNK,0)] <
LUM_X_MAX*dtr)
    trq_lumb[0] = -KROT_LUM*(qang[sdindx(TRUNK,0)]-(BIAS_LUM_X))
                -BDAMP_LUM*u[sdindx(TRUNK,0)];
  if(qang[sdindx(TRUNK,0)] > LUM_X_MAX*dtr)
    trq_lumb[0] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,0)]-(LUM_X_MAX)*dtr);

  sdhinget(TRUNK,0,trq_lumb[0]);

  /* Y-axis */

  if(qang[sdindx(TRUNK,1)] < LUM_Y_MIN*dtr)
    trq_lumb[1] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,1)]-(LUM_Y_MIN)*dtr);
  if(qang[sdindx(TRUNK,1)] > LUM_Y_MIN*dtr && qang[sdindx(TRUNK,1)] <
LUM_Y_MAX*dtr)
    trq_lumb[1] = -KROT_LUM*(qang[sdindx(TRUNK,1)]-(BIAS_LUM_Y))
                -BDAMP_LUM*u[sdindx(TRUNK,1)];
  if(qang[sdindx(TRUNK,1)] > LUM_Y_MAX*dtr)
    trq_lumb[1] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,1)]-(LUM_Y_MAX)*dtr);

  sdhinget(TRUNK,1,trq_lumb[1]);

  /* Z-axis */

  if(qang[sdindx(TRUNK,2)] < LUM_Z_MIN*dtr)
    trq_lumb[2] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,2)]-(LUM_Z_MIN)*dtr);
  if(qang[sdindx(TRUNK,2)] > LUM_Z_MIN*dtr && qang[sdindx(TRUNK,2)] <
LUM_Z_MAX*dtr)
    trq_lumb[2] = -KROT_LUM*(qang[sdindx(TRUNK,2)]-(BIAS_LUM_Z))
                -BDAMP_LUM*u[sdindx(TRUNK,2)];
  if(qang[sdindx(TRUNK,2)] > LUM_Z_MAX*dtr)
    trq_lumb[2] = -KROT_LUM_STOP*(qang[sdindx(TRUNK,2)]-(LUM_Z_MAX)*dtr);

  sdhinget(TRUNK,2,trq_lumb[2]);
  /*  printf("\nt=%f\tLumbar Z  Trq = %f",t,trq_lumb[2]); */

  /* Control of Arm Joints */

  /* SHOULDER */

  /* The passive elbow torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  if(qang[sdindx(UARMR,0)] < RSHL_X_MIN*dtr)
    trq_rshl[0] = -KROT_SHL_STOP*(qang[sdindx(UARMR,0)]-(RSHL_X_MIN)*dtr);
  if(qang[sdindx(UARMR,0)] > RSHL_X_MIN*dtr && qang[sdindx(UARMR,0)] <
RSHL_X_MAX*dtr)
    trq_rshl[0] = -KROT_SHL*(qang[sdindx(UARMR,0)]-(BIAS_RSHL_X))
               -BDAMP_SHL*u[sdindx(UARMR,0)];
  if(qang[sdindx(UARMR,0)] > RSHL_X_MAX*dtr)
    trq_rshl[0] = -KROT_SHL_STOP*(qang[sdindx(UARMR,0)]-(RSHL_X_MAX)*dtr);

  if(qang[sdindx(UARML,0)] < LSHL_X_MIN*dtr)
    trq_lshl[0] = -KROT_SHL_STOP*(qang[sdindx(UARML,0)]-(LSHL_X_MIN)*dtr);
  if(qang[sdindx(UARML,0)] > LSHL_X_MIN*dtr && qang[sdindx(UARML,0)] <
LSHL_X_MAX*dtr)
    trq_lshl[0] = -KROT_SHL*(qang[sdindx(UARML,0)]-(BIAS_LSHL_X))
               -BDAMP_SHL*u[sdindx(UARML,0)];
  if(qang[sdindx(UARML,0)] > LSHL_X_MAX*dtr)
    trq_lshl[0] = -KROT_SHL_STOP*(qang[sdindx(UARML,0)]-(LSHL_X_MAX)*dtr);

  if(qang[sdindx(UARMR,1)] < RSHL_Y_MIN*dtr)
    trq_rshl[1] = -KROT_SHL_STOP*(qang[sdindx(UARMR,1)]-(RSHL_Y_MIN)*dtr);
  if(qang[sdindx(UARMR,1)] > RSHL_Y_MIN*dtr && qang[sdindx(UARMR,1)] <
RSHL_Y_MAX*dtr)
    trq_rshl[1] = -KROT_SHL*(qang[sdindx(UARMR,1)]-(BIAS_RSHL_Y))
               -BDAMP_SHL*u[sdindx(UARMR,1)];
  if(qang[sdindx(UARMR,1)] > RSHL_Y_MAX*dtr)
    trq_rshl[1] = -KROT_SHL_STOP*(qang[sdindx(UARMR,1)]-(RSHL_Y_MAX)*dtr);

  if(qang[sdindx(UARML,1)] < LSHL_Y_MIN*dtr)
    trq_lshl[1] = -KROT_SHL_STOP*(qang[sdindx(UARML,1)]-(LSHL_Y_MIN)*dtr);
  if(qang[sdindx(UARML,1)] > LSHL_Y_MIN*dtr && qang[sdindx(UARML,1)] <
LSHL_Y_MAX*dtr)
    trq_lshl[1] = -KROT_SHL*(qang[sdindx(UARML,1)]-(BIAS_LSHL_Y))
               -BDAMP_SHL*u[sdindx(UARML,1)];
  if(qang[sdindx(UARML,1)] > LSHL_Y_MAX*dtr)
    trq_lshl[1] = -KROT_SHL_STOP*(qang[sdindx(UARML,1)]-(LSHL_Y_MAX)*dtr);

  sdhinget(UARMR,0,trq_rshl[0]);
  sdhinget(UARMR,1,trq_rshl[1]);
  sdhinget(UARML,0,trq_lshl[0]);
  sdhinget(UARML,1,trq_lshl[1]);

  /* ELBOW */

  /* The passive elbow torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  if(qang[sdindx(FARMR,0)] < ELBOWMIN*dtr)
```

```
    trq_relb = -KROT_ELB_STOP*(qang[sdindx(FARMR,0)]-(ELBOWMIN)*dtr);
  if(qang[sdindx(FARMR,0)] > ELBOWMIN*dtr && qang[sdindx(FARMR,0)] <
ELBOWMAX*dtr)
    trq_relb = -KROT_ELB*(qang[sdindx(FARMR,0)]-(BIAS_RELB))
               -BDAMP_ELB*u[sdindx(FARMR,0)];
  if(qang[sdindx(FARMR,0)] > ELBOWMAX*dtr)
    trq_relb = -KROT_ELB_STOP*(qang[sdindx(FARMR,0)]-(ELBOWMAX)*dtr);

  if(qang[sdindx(FARML,0)] < ELBOWMIN*dtr)
    trq_lelb = -KROT_ELB_STOP*(qang[sdindx(FARML,0)]-(ELBOWMIN)*dtr);
  if(qang[sdindx(FARML,0)] > ELBOWMIN*dtr && qang[sdindx(FARML,0)] <
ELBOWMAX*dtr)
    trq_lelb = -KROT_ELB*(qang[sdindx(FARML,0)]-(BIAS_LELB))
               -BDAMP_ELB*u[sdindx(FARML,0)];
  if(qang[sdindx(FARML,0)] > ELBOWMAX*dtr)
    trq_lelb = -KROT_ELB_STOP*(qang[sdindx(FARML,0)]-(ELBOWMAX)*dtr);

  sdhinget(FARMR,0,trq_relb);
  sdhinget(FARML,0,trq_lelb);

#ifdef DEBUG
    fprintf(stderr,"\nt=%f\tR Elbow Trq = %f\tL Elbow Trq =
%f",t,trq_relb,trq_lelb); */
#endif

  /* WRIST */

  /* The passive wrist torques include a lower joint stop, a nominal range,
     and an upper joint stop. */

  if(qang[sdindx(HANDR,0)] < RWRS_X_MIN*dtr)
    trq_rwrs[0] = -KROT_WRS_STOP*(qang[sdindx(HANDR,0)]-(RWRS_X_MIN)*dtr);
  if(qang[sdindx(HANDR,0)] > RWRS_X_MIN*dtr && qang[sdindx(HANDR,0)] <
RWRS_X_MAX*dtr)
    trq_rwrs[0] = -KROT_WRS*(qang[sdindx(HANDR,0)]-(BIAS_RWRS_X))
               -BDAMP_WRS*u[sdindx(HANDR,0)];
  if(qang[sdindx(HANDR,0)] > RWRS_X_MAX*dtr)
    trq_rwrs[0] = -KROT_WRS_STOP*(qang[sdindx(HANDR,0)]-(RWRS_X_MAX)*dtr);

  if(qang[sdindx(HANDL,0)] < LWRS_X_MIN*dtr)
    trq_lwrs[0] = -KROT_WRS_STOP*(qang[sdindx(HANDL,0)]-(LWRS_X_MIN)*dtr);
  if(qang[sdindx(HANDL,0)] > LWRS_X_MIN*dtr && qang[sdindx(HANDL,0)] <
LWRS_X_MAX*dtr)
    trq_lwrs[0] = -KROT_WRS*(qang[sdindx(HANDL,0)]-(BIAS_LWRS_X))
               -BDAMP_WRS*u[sdindx(HANDL,0)];
  if(qang[sdindx(HANDL,0)] > LWRS_X_MAX*dtr)
    trq_lwrs[0] = -KROT_WRS_STOP*(qang[sdindx(HANDL,0)]-(LWRS_X_MAX)*dtr);

  if(qang[sdindx(HANDR,1)] < RWRS_Y_MIN*dtr)
    trq_rwrs[1] = -KROT_WRS_STOP*(qang[sdindx(HANDR,1)]-(RWRS_Y_MIN)*dtr);
  if(qang[sdindx(HANDR,1)] > RWRS_Y_MIN*dtr && qang[sdindx(HANDR,1)] <
RWRS_Y_MAX*dtr)
    trq_rwrs[1] = -KROT_WRS*(qang[sdindx(HANDR,1)]-(BIAS_RWRS_Y))
               -BDAMP_WRS*u[sdindx(HANDR,1)];
  if(qang[sdindx(HANDR,1)] > RWRS_Y_MAX*dtr)
    trq_rwrs[1] = -KROT_WRS_STOP*(qang[sdindx(HANDR,1)]-(RWRS_Y_MAX)*dtr);

  if(qang[sdindx(HANDL,1)] < LWRS_Y_MIN*dtr)
```

```
    trq_lwrs[1] = -KROT_WRS_STOP*(qang[sdindx(HANDL,1)]-(LWRS_Y_MIN)*dtr);
  if(qang[sdindx(HANDL,1)] > LWRS_Y_MIN*dtr && qang[sdindx(HANDL,1)] <
LWRS_Y_MAX*dtr)
    trq_lwrs[1] = -KROT_WRS*(qang[sdindx(HANDL,1)]-(BIAS_LWRS_Y))
                -BDAMP_WRS*u[sdindx(HANDL,1)];
  if(qang[sdindx(HANDL,1)] > LWRS_Y_MAX*dtr)
    trq_lwrs[1] = -KROT_WRS_STOP*(qang[sdindx(HANDL,1)]-(LWRS_Y_MAX)*dtr);

  sdhinget(HANDR,0,trq_rwrs[0]);
  sdhinget(HANDR,1,trq_rwrs[1]);
  sdhinget(HANDL,0,trq_lwrs[0]);
  sdhinget(HANDL,1,trq_lwrs[1]);

#ifdef DEBUG
  printf("\nt=%f\tR Wrist Trq = %f\tL Wrist Trq =
%f",t,trq_rwrs[0],trq_lwrs[0]);
#endif
     }

     /* SUIT -- assume symmetric */

  /* ELBOW */
  suit_elbow_damping = 10.0; /* To ensure smooth motion */
  if(u[sdindx(FARMR,0)] > 0)
    suit_elbow_torque = compose_force(&elbowmesh,
                              farm_pos,
                              UP);
  else
    suit_elbow_torque = compose_force(&elbowmesh,
                              farm_pos,
                              DOWN);
  /* Apply the torque */
  sdhinget(FARMR,0,
         -suit_elbow_torque
         -suit_elbow_damping*u[sdindx(FARMR,0)]);
  sdhinget(FARML,0,
         -suit_elbow_torque
         -suit_elbow_damping*u[sdindx(FARML,0)]);


  /* SHOULDER */
  suit_shoulder_damping = 10.0;
  if(u[sdindx(UARMR,2)] > 0)
    suit_shoulder_torque = compose_force(&shouldermesh,
                              uarm_pos,
                              UP);
  else
    suit_shoulder_torque = compose_force(&shouldermesh,
                              uarm_pos,
                              DOWN);
  /* Apply the torque */
  sdhinget(UARMR,2,
         -suit_shoulder_torque
         -suit_shoulder_damping*u[sdindx(UARMR,2)]);
  sdhinget(UARML,2,
         -suit_shoulder_torque
         -suit_shoulder_damping*u[sdindx(UARML,2)]);
  /* KNEE */
  suit_knee_damping = 10.0;
  if(u[sdindx(ULEG,0)] > 0)
    suit_knee_torque = compose_force(&kneemesh,
                                     uleg_pos,
                                     UP);
  else
    suit_knee_torque = compose_force(&kneemesh,
                                     uleg_pos,
                                     DOWN);
#ifdef DEBUG
  fprintf(stderr,"\t%lf %lf\n\n",
        uleg_pos,-suit_knee_torque);
#endif
  sdhinget(ULEGR,0,-suit_knee_torque-
         suit_knee_damping*u[sdindx(ULEGR,0)]);
  sdhinget(ULEGL,0,-suit_knee_torque-
         suit_knee_damping*u[sdindx(ULEGL,0)]);

/* Compute 1st order approximation to suit ankle torque */
  suit_ankle_torque = ANKLE_A*ankle_pos;
  sdhinget(LLEGR,2,-suit_ankle_torque);
  sdhinget(LLEGL,2,-suit_ankle_torque);
#ifdef DEBUG
  fprintf(stderr,"theta: %f\tTorque: %f\n",ankle_pos,suit_ankle_torque);
#endif

}

/***************************************************************************
*

  sdumotion

  If enabled, this function prescribes the motion of the hand.
  This function moves the endpoint of the arm along a desired trajectory (a
  linear trajectory with a sinusoidal velocity profile) by means of
  prescribed motion in each of the two slider joints attached between the
  hand and ground.  In addition, the hand is maintained in a horizontal
  orientation, by prescribing the angle of hslide wrt the slider to be 0.

****************************************************************************
**/

sdumotion(t,q,u)
double t,q[NQ],u[NU];
{
  double xe, xde, xdde, ye, yde, ydde;
  double pathlength,pathvelocity,pathaccel;
  double pharm, vharm, aharm, qd[NQ], ud[NU];
  int i;
  double theta;
  extern double xstart, ystart;


  /* Kinematics */
  theta = 0.0;
```

```
pathlength = MAX_VELOCITY*t
  + (MAX_VELOCITY*MOTION_TIME/(2*pi)) * sin((2*pi/MOTION_TIME)*t+pi);

pathvelocity = MAX_VELOCITY
  + MAX_VELOCITY * cos((2*pi/MOTION_TIME)*t+pi);

pathaccel = (-2*pi*MAX_VELOCITY/MOTION_TIME)
  * sin((2*pi/MOTION_TIME)*t+pi);

/* position */

xe = xstart - pathlength * cos(theta);
ye = ystart - pathlength * sin(theta);

/* velocity */

xde = - pathvelocity * cos(theta);
yde = - pathvelocity * sin(theta);

/* acceleration */

xdde = -pathaccel * cos(theta);
ydde = -pathaccel * sin(theta);

/* Prescribed motion for inverse kinematics phase. */

if (motionphase = 1){

  /* prescribe hand accelerations */

  sdpresacc(HNDRG,0,xdde);
  sdpresacc(HNDRG,1,ydde);
  sdpresacc(HNDRG,2,0.0);
  sdpresacc(HNDRG,3,0.0);
  sdpresacc(HNDRG,4,0.0);
  sdpresacc(HNDRG,5,0.0);
  sdpresacc(HNDLG,0,xdde);
  sdpresacc(HNDLG,1,ydde);
  sdpresacc(HNDLG,2,0.0);
  sdpresacc(HNDLG,3,0.0);
  sdpresacc(HNDLG,4,0.0);
  sdpresacc(HNDLG,5,0.0);

  /* prescribe hand velocities */
  /* Note: in order to utilize these, sdstab() must be set */

  sdpresvel(HNDRG,0,xde);
  sdpresvel(HNDRG,1,yde);
  sdpresvel(HNDRG,2,0.0);
  sdpresvel(HNDRG,3,0.0);
  sdpresvel(HNDRG,4,0.0);
  sdpresvel(HNDRG,5,0.0);
  sdpresvel(HNDLG,0,xde);
  sdpresvel(HNDLG,1,yde);
  sdpresvel(HNDLG,2,0.0);
  sdpresvel(HNDLG,3,0.0);
  sdpresvel(HNDLG,4,0.0);
  sdpresvel(HNDLG,5,0.0);
  /* prescribe hand positions
     /* NOTE 1: in order to utilize these, sdstab() must be set */
  /* NOTE 2: sdprespos() IS NOT ALLOWED FOR BALL JOINTS !!
     Thus, only the translational position values of the
     6dof joints are set. */

  sdprespos(HNDRG,0,xe);
  sdprespos(HNDRG,1,ye);
  sdprespos(HNDRG,2,ZRIGHT);

  sdprespos(HNDLG,0,xe);
  sdprespos(HNDLG,1,ye);
  sdprespos(HNDLG,2,ZLEFT);

}

/* Prescribed motion for inverse dynamics phase. */

if (motionphase = 2){

  /* Prescribe accelerations for all body joints. */

  sdpresacc(LLEGR,0,accel[step][sdindx(LLEGR,0)]);
  sdpresacc(LLEGR,1,accel[step][sdindx(LLEGR,1)]);
  sdpresacc(LLEGR,2,accel[step][sdindx(LLEGR,2)]);

  sdpresacc(ULEGR,0,accel[step][sdindx(ULEGR,0)]);

  sdpresacc(PELVR,0,accel[step][sdindx(PELVR,0)]);
  sdpresacc(PELVR,1,accel[step][sdindx(PELVR,1)]);
  sdpresacc(PELVR,2,accel[step][sdindx(PELVR,2)]);

  sdpresacc(LLEGL,0,accel[step][sdindx(LLEGL,0)]);
  sdpresacc(LLEGL,1,accel[step][sdindx(LLEGL,1)]);
  sdpresacc(LLEGL,2,accel[step][sdindx(LLEGL,2)]);

  sdpresacc(ULEGL,0,accel[step][sdindx(ULEGL,0)]);

  sdpresacc(PELVL,0,accel[step][sdindx(PELVL,0)]);
  sdpresacc(PELVL,1,accel[step][sdindx(PELVL,1)]);
  sdpresacc(PELVL,2,accel[step][sdindx(PELVL,2)]);

  sdpresacc(TRUNK,0,accel[step][sdindx(TRUNK,0)]);
  sdpresacc(TRUNK,1,accel[step][sdindx(TRUNK,1)]);
  sdpresacc(TRUNK,2,accel[step][sdindx(TRUNK,2)]);

  sdpresacc(UARMR,0,accel[step][sdindx(UARMR,0)]);
  sdpresacc(UARMR,1,accel[step][sdindx(UARMR,1)]);
  sdpresacc(UARMR,2,accel[step][sdindx(UARMR,2)]);

  sdpresacc(FARMR,0,accel[step][sdindx(FARMR,0)]);

  sdpresacc(HANDR,0,accel[step][sdindx(HANDR,0)]);
  sdpresacc(HANDR,1,accel[step][sdindx(HANDR,1)]);
  sdpresacc(HANDR,2,accel[step][sdindx(HANDR,2)]);

  sdpresacc(UARML,0,accel[step][sdindx(UARML,0)]);
```

```
    sdpresacc(UARML,1,accel[step][sdindx(UARML,1)]);
    sdpresacc(UARML,2,accel[step][sdindx(UARML,2)]);

    sdpresacc(FARML,0,accel[step][sdindx(FARML,0)]);

    sdpresacc(HANDL,0,accel[step][sdindx(HANDL,0)]);
    sdpresacc(HANDL,1,accel[step][sdindx(HANDL,1)]);
    sdpresacc(HANDL,2,accel[step][sdindx(HANDL,2)]);

  }
}


/***************************************************************************
*
   This function sets up the initial conditions. The hand position is set to
   xstart, ystart values. This is referenced to the original position of the
   hand, with the arm hanging straight down. The x and y velocities of the
   hand are also set according to the value of pathvelocity value. Pass
   in y with an initial guess to control the assembly solution and improve
   convergence. On return, y should be a fully compatible state vector,
unless
   an error is received.
   (G. Schaffner)
****************************************************************************
**/

initconds(t,y)
double t, y[];
{
int i;
double q_init[NQ];

/* Setup initial conditions - body positions and velocities */

  printf("\nInitial Conditions.................");

/* Initial Configuration */

  for (i = 0 ; i < NU ; i++) q_ang[i] = 0.0;

/* Assign initial angles to joint degrees of freedom in terms of Euler
   angles. Last step is to convert to Euler parameters (quaternions).
   Note: Angles are measured as rotation of the segment coord. system
   relative to the inboard segment's coord. system. Also, these values
   get the body close to the desired configuration, and then sdassemble
   completes the job by varying the arm joint angles until the desired
   hand positions and orientations are obtained. */

  /* start out with simple init config, REFINE ANGLES LATER */

  q_ang[sdindx(LLEGR,0)]  =   0.0*dtr; /* r-ankle X rot */
  q_ang[sdindx(LLEGR,1)]  =   0.0*dtr; /* r-ankle Y rot */
  q_ang[sdindx(LLEGR,2)]  =   0.0*dtr; /* r-ankle Z rot */

  q_ang[sdindx(ULEGR,0)]  =  47.0*dtr; /* r-knee  Z rot */

  q_ang[sdindx(PELVR,0)]   =   0.0*dtr; /* r-hip X rot */
  q_ang[sdindx(PELVR,1)]   =   0.0*dtr; /* r-hip Y rot */
  q_ang[sdindx(PELVR,2)]   = -52.0*dtr; /* r-hip Z rot */

  q_ang[sdindx(LLEGL,0)]  =   0.0*dtr; /* l-ankle X rot */
  q_ang[sdindx(LLEGL,1)]  =   0.0*dtr; /* l-ankle Y rot */
  q_ang[sdindx(LLEGL,2)]  =   0.0*dtr; /* l-ankle Z rot */

  q_ang[sdindx(ULEGL,0)]  =  47.0*dtr; /* l-knee  Z rot */

  q_ang[sdindx(PELVL,0)]   =   0.0*dtr; /* l-hip X rot */
  q_ang[sdindx(PELVL,1)]   =   0.0*dtr; /* l-hip Y rot */
  q_ang[sdindx(PELVL,2)]   = -52.0*dtr; /* l-hip Z rot */

  q_ang[sdindx(TRUNK,0)]  =   0.0*dtr; /* lumbar X rot */
  q_ang[sdindx(TRUNK,1)]  =   0.0*dtr; /* lumbar Y rot */
  q_ang[sdindx(TRUNK,2)]  =   0.0*dtr; /* lumbar Z rot */

  q_ang[sdindx(UARMR,0)]  =   0.0*dtr; /* r-shoulder X rot */
  q_ang[sdindx(UARMR,1)]  =   0.0*dtr; /* r-shoulder Y rot */
  q_ang[sdindx(UARMR,2)]  =  90.0*dtr; /* r-shoulder Z rot */

  q_ang[sdindx(FARMR,0)]  =   5.0*dtr; /* r-elbow Z rot */

  q_ang[sdindx(HANDR,0)]  =   0.0*dtr; /* r-wrist X rot */
  q_ang[sdindx(HANDR,1)]  =   0.0*dtr; /* r-wrist Y rot */
  q_ang[sdindx(HANDR,2)]  =   0.0*dtr; /* r-wrist Z rot */

  q_ang[sdindx(UARML,0)]  =   0.0*dtr; /* l-shoulder X rot */
  q_ang[sdindx(UARML,1)]  =   0.0*dtr; /* l-shoulder Y rot */
  q_ang[sdindx(UARML,2)]  =  90.0*dtr; /* l-shoulder Z rot */

  q_ang[sdindx(FARML,0)]  =   5.0*dtr; /* l-elbow Z rot */

  q_ang[sdindx(HANDL,0)]  =   0.0*dtr; /* l-wrist X rot */
  q_ang[sdindx(HANDL,1)]  =   0.0*dtr; /* l-wrist Y rot */
  q_ang[sdindx(HANDL,2)]  =   0.0*dtr; /* l-wrist Z rot */

  q_ang[sdindx(HNDRG,0)]  =  xstart; /* r-hand X pos */
  q_ang[sdindx(HNDRG,1)]  =  ystart; /* r-hand Y pos */
  q_ang[sdindx(HNDRG,2)]  =  0.0; /* r-hand Z pos */
  q_ang[sdindx(HNDRG,3)]  =  0.0*dtr; /* r-hand X rot */
  q_ang[sdindx(HNDRG,4)]  =  0.0*dtr; /* r-hand Y rot */
  q_ang[sdindx(HNDRG,5)]  =  0.0*dtr; /* r-hand Z rot */

  q_ang[sdindx(HNDLG,0)]  =  xstart; /* l-hand X pos */
  q_ang[sdindx(HNDLG,1)]  =  ystart; /* l-hand Y pos */
  q_ang[sdindx(HNDLG,2)]  =  0.0; /* l-hand Z pos */
  q_ang[sdindx(HNDLG,3)]  =  0.0*dtr; /* l-hand X rot */
  q_ang[sdindx(HNDLG,4)]  =  0.0*dtr; /* l-hand Y rot */
  q_ang[sdindx(HNDLG,5)]  =  0.0*dtr; /* l-hand Z rot */

  /* convert Euler angles to quaternions and assign vals to state array. */

  sdang2st(q_ang,q_init);

  for (i = 0 ; i < NQ ; i++) y[i] = q_init[i]; /* assign position vals to
state */
```

```
  for (i = 0 ; i < NU ; i++) y[NQ+i] = 0.0; /* assign velocity vals to state
*/

  printf("done.\n"); /* end of initial conditions */

}

/***************************************************************************
*
   This function sets the lock array so that only the arm joints
   are allowed to move during the sdassemble routine.
   (G. Schaffner)
****************************************************************************
**/
void
lockjoints(int lock[])
{
  int i;

  for (i=0 ; i < NU ; i++) lock[i] = ON; /* First, lock ALL joints. */

  /* Next, unlock the arm joints so that they can move. */

  lock[sdindx(UARMR,0)]  = OFF;
  lock[sdindx(UARMR,1)]  = OFF;
  lock[sdindx(UARMR,2)]  = OFF;

  lock[sdindx(FARMR,0)]  = OFF;

  lock[sdindx(HANDR,0)]  = OFF;
  lock[sdindx(HANDR,1)]  = OFF;
  lock[sdindx(HANDR,2)]  = OFF;

  lock[sdindx(UARML,0)]  = OFF;
  lock[sdindx(UARML,1)]  = OFF;
  lock[sdindx(UARML,2)]  = OFF;

  lock[sdindx(FARML,0)]  = OFF;

  lock[sdindx(HANDL,0)]  = OFF;
  lock[sdindx(HANDL,1)]  = OFF;
  lock[sdindx(HANDL,2)]  = OFF;

  /* Finally, unlock the hand-to-ground joints so that they can achieve
     the prescribed motions. */
  /*
  lock[sdindx(HNDRG,0)]  = OFF;
  lock[sdindx(HNDRG,1)]  = OFF;
  lock[sdindx(HNDRG,2)]  = OFF;
  lock[sdindx(HNDRG,3)]  = OFF;
  lock[sdindx(HNDRG,4)]  = OFF;



  lock[sdindx(HNDRG,5)]  = OFF;

  lock[sdindx(HNDLG,0)]  = OFF;
  lock[sdindx(HNDLG,1)]  = OFF;
  lock[sdindx(HNDLG,2)]  = OFF;
  lock[sdindx(HNDLG,3)]  = OFF;
  lock[sdindx(HNDLG,4)]  = OFF;
  lock[sdindx(HNDLG,5)]  = OFF;
  */
}



/* Print function */
void
printvals(int nval, double array[])
{
  int i;
  for (i = 0 ;  i< nval ; i++) printf("%f\t", array[i]);
  printf("\n");
}




/****************************************/
/*     Functions for hysteresis model   */
/****************************************/

void
initialize_mesh(Mesh *mesh, double angle)
{
  int i;

  mesh->num_trapezoids = 1;
  mesh->last_direction = UP;
  mesh->last_angle = angle;
  for (i=0; i<100 ; i++) {
    mesh->M[i] = 0.0;
    mesh->m[i] = 0.0;
  }
  mesh->m[0] = ((mesh->rows[0]).cells[0]).betamin;
}


double
compose_force(Mesh *mesh, double theta, int direction)
{
  double calc_fab(Mesh mesh, double a, double b);
  double sum = 0.0;
  int k;

#ifdef DEBUG
  /* Print out the current Interface I(t), at vertices M[k],m[k] */

  fprintf(stderr,"%lf ",theta);
  fprintf(stderr,"%d " ,mesh->num_trapezoids);
  for (k=1; k < mesh->num_trapezoids+1; k++)
    fprintf(stderr,"%lf %lf\t",mesh->M[k],mesh->m[k]);
#endif

  if (direction == UP) {
    /* If we're going up, check to see if we just changed direction */
    if (mesh->last_direction == DOWN) {
      mesh->m[mesh->num_trapezoids] = mesh->last_angle;
      mesh->num_trapezoids += 1;
      mesh->last_direction = UP;
    }
    mesh->last_angle = theta;

    /* Check for wiping-out property */
    if (mesh->num_trapezoids > 1) {
      if (theta > mesh->M[mesh->num_trapezoids-1]) {
      mesh->M[mesh->num_trapezoids-1] = 0;
      mesh->m[mesh->num_trapezoids-1] = 0;
      mesh->num_trapezoids -= 1;
      }
    }

    /* Compute the sum */
    for (k=1; k <= mesh->num_trapezoids-1; k++)
      sum += calc_fab(*mesh, mesh->M[k], mesh->m[k])
      - calc_fab(*mesh, mesh->M[k], mesh->m[k-1]);

    return (- mesh->max_saturation
          + sum
          + calc_fab(*mesh, theta, theta)
          - calc_fab(*mesh, theta, mesh->m[mesh->num_trapezoids-1]));
  }


  else if (direction == DOWN) {

    /* If we're going down, check to see if we just changed direction */
    if (mesh->last_direction == UP) {
      mesh->M[mesh->num_trapezoids] = mesh->last_angle;
      mesh->last_direction = DOWN;
    }
    mesh->last_angle = theta;

    /* Check for wiping-out property */
    if (mesh->num_trapezoids > 1) {
      if (theta < mesh->m[mesh->num_trapezoids-1]) {
      mesh->M[mesh->num_trapezoids] = 0;
      mesh->m[mesh->num_trapezoids-1] = 0;
      mesh->num_trapezoids -= 1;
      }
    }

    /* Compute the sum */
    for (k=1; k <= mesh->num_trapezoids-1; k++)
      sum += calc_fab(*mesh, mesh->M[k],
                  mesh->m[k]) - calc_fab(*mesh, mesh->M[k], mesh->m[k-1]);

    return (- mesh->max_saturation
          + sum
          + calc_fab(*mesh, mesh->M[mesh->num_trapezoids], theta)
          - calc_fab(*mesh,
                   mesh->M[mesh->num_trapezoids],
                    mesh->m[mesh->num_trapezoids-1]));
  }


}

double
calc_fab(Mesh mesh, double a, double b)
{
  int i,j;
  int triangle = 0;

  for(i=0; mesh.rows[i].alphamin > a ; i++)
    ;
  for(j=0; mesh.rows[i].cells[j].betamax < b ; j++)
    ;
  if( j == mesh.rows[i].numcells-1)
    triangle = 1;

  if( !triangle )
    return(mesh.rows[i].cells[j].c1     +
         mesh.rows[i].cells[j].c2*a   +
         mesh.rows[i].cells[j].c3*b   +
         mesh.rows[i].cells[j].c4*a*b);
  else
    return(mesh.rows[i].cells[j].c1     +
         mesh.rows[i].cells[j].c2*a   +
         mesh.rows[i].cells[j].c3*b);
}



void
loadmesh(char *filename, Mesh *mesh)
{

  FILE *fp;
  int items_read, ncells, i;
  double amin, amax;
  Row *current_row;
  Cell *current_cell;

  if((fp = fopen(filename, "r")) == NULL) {
    fprintf(stderr, "Can't open mesh file %s\n",filename);
    exit(-1);
  }

  mesh->numrows = 0;
  mesh->rows = NULL;

  fscanf(fp, "%lf",&(mesh->max_saturation));
  while((items_read = fscanf(fp,"%lf%lf%d",&amin,&amax,&ncells)) == 3) {

    mesh->numrows++;
    mesh->rows = realloc(mesh->rows, mesh->numrows*sizeof(Row));

    current_row = &((mesh->rows)[mesh->numrows-1]);
    current_row->alphamin = amin;
    current_row->alphamax = amax;
    current_row->numcells = ncells;

    current_row->cells = malloc(ncells*sizeof(Cell));

    for(i=0; i < ncells ; i++) {
      fscanf(fp,"%lf%lf%lf%lf%lf%lf",
           &((current_row->cells)[i].betamin),
           &((current_row->cells)[i].betamax),
           &((current_row->cells)[i].c1),
           &((current_row->cells)[i].c2),
           &((current_row->cells)[i].c3),
           &((current_row->cells)[i].c4));
    }
  }
}
```

Appendix F: Processing for Simulation Results

This appendix contains the code used to process the simulation results in MATLAB. The input files were umatpos.dat, umatvel.dat, and umattrq.dat for the unsuited joint position, velocity, and torque data files; the corresponding suited data files were smatpos.dat, smatvel.dat, and smattrq.dat. This file contains plot axes appropriate for the compliant lower body simulation, but can be used on any data set. The axes may have to be reformatted.

```
clear

load umatpos.dat -ascii
load umatvel.dat -ascii
load umattrq.dat -ascii
load smatpos.dat -ascii
load smatvel.dat -ascii
load smattrq.dat -ascii

% Shorten the names of the data files, and convert angles
% from radians to degrees

upos(:,2:7) = umatpos(:,2:7).*180.0/pi;
upos(:,1) = umatpos(:,1);
[L,w] = size(umatvel); uvel = umatvel(1:(L-1),:);
[L,w] = size(umattrq); utrq = umattrq(2:L,:);
spos(:,2:7) = smatpos(:,2:7).*180.0/pi;
spos(:,1) = smatpos(:,1);
[L,w] = size(smatvel); svel = smatvel(1:(L-1),:);
[L,w] = size(smattrq); strq = smattrq(2:L,:);

% Plot the joint angles as a function of time

figure(1);
clg;
hold on;

subplot(3,2,1);hold on;
plot(upos(:,1),upos(:,2),'w-');
plot(spos(:,1),spos(:,2),'w--');
title('Ankle Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');

subplot(3,2,2);hold on;
plot(upos(:,1),upos(:,3),'w-');
plot(spos(:,1),spos(:,3),'w--');
title('Knee Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');

subplot(3,2,3);hold on;
plot(upos(:,1),upos(:,4),'w-');
plot(spos(:,1),spos(:,4),'w--');
```

```
title('Hip Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');

subplot(3,2,4);hold on;
plot(upos(:,1),upos(:,5),'w-');
plot(spos(:,1),spos(:,5),'w--');
title('Shoulder Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');

subplot(3,2,5);hold on;
plot(upos(:,1),upos(:,6),'w-');
plot(spos(:,1),spos(:,6),'w--');
title('Elbow Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');

subplot(3,2,6);hold on;
plot(upos(:,1),upos(:,7),'w-');
plot(spos(:,1),spos(:,7),'w--');
title('Wrist Joint Position');
xlabel('Time (sec)');
ylabel('Joint Angle (deg)');
legend('Unsuited','Suited');


% Plot the joint torques as a function of time

figure(2);
clg;
hold on;

subplot(3,2,1);hold on;
plot(utrq(:,1),utrq(:,2),'w-');
plot(strq(:,1),strq(:,2),'w--');
title('Ankle Joint Torque');
axis([0 10 -15 15]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');

subplot(3,2,2);hold on;
plot(utrq(:,1),utrq(:,3),'w-');
plot(strq(:,1),strq(:,3),'w--');
title('Knee Joint Torque');
axis([0 10 -15 15]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');

subplot(3,2,3);hold on;
plot(utrq(:,1),utrq(:,4),'w-');
plot(strq(:,1),strq(:,4),'w--');
title('Hip Joint Torque');
```

```
axis([0 10 -15 15]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');

subplot(3,2,4);hold on;
plot(utrq(:,1),utrq(:,5),'w-');
plot(strq(:,1),strq(:,5),'w--');
title('Shoulder Joint Torque');
axis([0 10 -5 5]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');

subplot(3,2,5);hold on;
plot(utrq(:,1),utrq(:,6),'w-');
plot(strq(:,1),strq(:,6),'w--');
title('Elbow Joint Torque');
axis([0 10 -5 5]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');

subplot(3,2,6);hold on;
plot(utrq(:,1),utrq(:,7),'w-');
plot(strq(:,1),strq(:,7),'w--');
title('Wrist Joint Torque');
axis([0 10 -5 5]);
xlabel('Time (sec)');
ylabel('Joint Torque (N-m)');
legend('Unsuited','Suited');
```

```
% Calculate joint power

upwr(:,2:7) = abs(uvel(:,2:7).*utrq(:,2:7));
upwr(:,1) = uvel(:,1);

spwr(:,2:7) = abs(svel(:,2:7).*strq(:,2:7));
spwr(:,1) = svel(:,1);


% Integrate over the timestep (0.05 sec) to get work

uwrk(:,2:7) = upwr(:,2:7).*0.05;
swrk(:,2:7) = spwr(:,2:7).*0.05;


[L,w] = size(uwrk);
ucumwrk = zeros(L,w);
scumwrk = zeros(L,w);
ucumwrk(1,2:7) = uwrk(1,2:7);
scumwrk(1,2:7) = swrk(1,2:7);

% Create cumulative work by accumulating work
for i = 2:L
  for j = 2:7
    ucumwrk(i,j) = uwrk(i,j)+ucumwrk(i-1,j);
      scumwrk(i,j) = swrk(i,j)+scumwrk(i-1,j);
  end
end

ucumwrk(:,1) = uvel(:,1);
scumwrk(:,1) = uvel(:,1);

% Plot the cumulative work

figure(3);
clg;
hold on;

subplot(3,2,1);hold on;
plot(ucumwrk(:,1),ucumwrk(:,2),'w-');
plot(scumwrk(:,1),scumwrk(:,2),'w--');
axis([0 10 0 2]);
title('Cumulative Work at the Ankle Joint');
ylabel('Cumulative Work (Joules)');
xlabel('Time (sec)');
legend('Unsuited','Suited');

subplot(3,2,2);hold on;
plot(ucumwrk(:,1),ucumwrk(:,3),'w-');
plot(scumwrk(:,1),scumwrk(:,3),'w--');
axis([0 10 0 2]);
title('Cumulative Work at the Knee Joint');
ylabel('Cumulative Work (Joules)');
xlabel('Time (sec)');
legend('Unsuited','Suited');

subplot(3,2,3);hold on;
```

```
plot(ucumwrk(:,1),ucumwrk(:,4),'w-');
plot(scumwrk(:,1),scumwrk(:,4),'w--');
axis([0 10 0 2]);
title('Cumulative Work at the Hip Joint');
ylabel('Cumulative Work (Joules)');
xlabel('Time (sec)');
legend('Unsuited','Suited');

subplot(3,2,4);hold on;
plot(ucumwrk(:,1),ucumwrk(:,5),'w-');
plot(scumwrk(:,1),scumwrk(:,5),'w--');
axis([0 10 0 2]);
title('Cumulative Work at the Shoulder Joint');
ylabel('Cumulative Work (Joules)');
xlabel('Time (sec)');
legend('Unsuited','Suited');

subplot(3,2,5);hold on;
plot(ucumwrk(:,1),ucumwrk(:,6),'w-');
plot(scumwrk(:,1),scumwrk(:,6),'w--');
axis([0 10 0 2]);
title('Cumulative Work at the Elbow Joint');
ylabel('Cumulative Work (Joules)');
xlabel('Time (sec)');
legend('Unsuited','Suited');

subplot(3,2,6);hold on;
plot(ucumwrk(:,1),ucumwrk(:,7),'w-');
plot(scumwrk(:,1),scumwrk(:,7),'w--');
title('Cumulative Work at the Wrist Joint');
xlabel('Time (sec)');
ylabel('Cumulative Work (Joules)');
legend('Unsuited','Suited');
```