A Predictive Control Method for Human Upper-Limb Motion: Graph-Theoretic Modelling, Dynamic Optimization, and Experimental Investigations

by

Ajay Seth

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2000

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Optimal control methods are applied to mechanical models in order to predict the control strategies in human arm movements. Optimality criteria are used to determine unique controls for a biomechanical model of the human upper-limb with redundant actuators. The motivation for this thesis is to provide a non-task-specific method of motion prediction as a tool for movement researchers and for controlling human models within virtual prototyping environments.

The current strategy is based on determining the muscle activation levels (control signals) necessary to perform a task that optimizes several physical determinants of the model such as muscular and joint stresses, as well as performance timing. Currently, the initial and final location, orientation, and velocity of the hand define the desired task. Several models of the human arm were generated using a graph-theoretical method in order to take advantage of similar system topology through the evolution of arm models. Within this framework, muscles were modelled as non-linear actuator components acting between origin and insertion points on rigid body segments. Activation levels of the muscle actuators are considered the control inputs to the arm model.

Optimization of the activation levels is performed via a hybrid genetic algorithm (GA) and a sequential quadratic programming (SQP) technique, which provides a globally optimal solution without sacrificing numerical precision, unlike traditional genetic algorithms. Advantages of the underlying genetic algorithm approach are that it does not require any prior knowledge of what might be a 'good' approximation in order for the method to converge, and it enables several objectives to be included in the evaluation of the fitness function. Results indicate that this approach can predict optimal strategies when compared to benchmark minimum-time maneuvers of a robot manipulator.

The formulation and integration of the aforementioned components into a working model and the simulation of reaching and lifting tasks represents the bulk of the thesis. Results are compared to motion data collected in the laboratory from a test subject performing the same tasks. Discrepancies in the results are primarily due to model fidelity. However, more complex models are not evaluated due to the additional computational time required. The theoretical approach provides an excellent foundation, but further work is required to increase the computational efficiency of the numerical implementation before proceeding to more complex models.

Acknowledgements

I would like to thank several individuals for their advice and support from the beginning of my graduate studies to the completion of this manuscript. First, I am forever indebted to the support of my supervisor Professor John McPhee for initially accepting my research proposal, letting me run with it, and trusting me to complete it. I have really learned a lot about myself and the world of research under his tutelage; not to mention he has provided me with a solid foundation in multi-body dynamics. Next, I would like to acknowledge my co-supervisor, Professor Richard Wells in the Department of Kinesiology, for providing the opportunity for data collection and for his advice and assistance throughout the process.

My appreciation goes out to my readers, Professors Catherine Burns and Fakhri Karray for their valuable feedback as well as their kind consideration. It was in fact Professor Karray that first presented the basis of the genetic algorithm to me through his intelligent control systems course.

I would like to thank both Systems Design Engineering graduate secretaries, Carol Kendrick, of years past, and Angela Semple, of recent months, for truly being helpful and accommodating even when bureaucratic policies would have dictated otherwise. In the same light, I offer my gratitude to Elaine Garner for her guidance and patience as I have stumbled through rounds of NSERC applications. Your kindness does not go unnoticed.

I must give a sincere thanks to my fellow lab mates of past and present: Ivan Bruulsema, Craig Good, Tim Lahey, Pengfei Shi, Yuping He, Hank Venema and Marcus Scherer, who collectively have contributed to every part of this thesis. I have to thank Tim, especially, for coming up with alternative plans when recently faced with "technical difficulties" and for his wealth of free advice for problems in general, even when no advice was sought. Most of all, thanks guys for founding and contributing to my caffeine addiction — "hey let's go to the C&D". "In five minutes ... really".

Finally, to my mom, my sister and her family, and to my dearest Olga, thank you all for your love and support and for providing me with happiness that is beyond personal status and material wealth.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Current Human Movement Analysis Tools 2

1.2.1 Kinematics and Inverse Kinematics 3

1.2.2 Inverse Dynamics 4

1.2.3 Electromyography 5

1.2.4 Forward Dynamics 6

1.2.5 Optimal Control 7

1.3 Research Objectives 8

1.4 Control Prediction Methodology 9

2 Human Upper Limb Mechanical Model **11**

2.1 Review of Upper-Limb Biomechanics 12

2.1.1 Upper Limb Kinematics 12

2.1.2 Bones and segment models 14

2.1.3 Joint models 14

2.1.4 Muscle Models 16

2.1.5 Remarks on Biomechanical Modelling 18

2.2 Current Mechanical Modelling Methods 18
2.3 Biomechanical Modelling using Graph Theory 19
2.3.1 Component Paradigm 24
2.3.2 DynaFlex: 3D Mechanical System Modeller 25
2.3.3 Generating Equations of Motion with DynaFlex 26
2.3.4 Scalable Framework 31
2.3.5 Adding Components by Introducing New Terminal Equations 31
2.4 Numerical Solution of Biomechanical System Equations 31

3 Optimal Control Determination **33**

3.1 Optimization of Dynamic Systems 33
3.1.1 Static vs. Dynamic Optimization 35
3.2 The Optimal Control Problem 35
3.3 Solution Methods for Dynamic Optimization 37
3.3.1 Necessary Extremum Conditions 38
3.3.2 Sufficient Optimality Conditions 40
3.3.3 Link to Analytical Mechanics 41
3.4 Indirect Numerical Solutions 42
3.4.1 The Shooting Method 42
3.4.2 Multiple Shooting Method 43
3.4.3 Higher-Order Methods 44
3.5 Direct Numerical Methods 45
3.5.1 Control Parameterization 45
3.5.2 Augmenting the Optimal Control Problem 46
3.5.3 Determining Functional Gradients 47

3.6 Numerical Optimization Methods 49
3.6.1 Gradient Based Methods 50
3.6.2 Global Search Methods 57
3.6.3 The Genetic Algorithm 59
3.6.4 Hybrid Methods 64
3.7 Comparison of Methods 65

4 Implementation and Prediction Results **69**

4.1 Performance of a Planar 2-link Manipulator 69
4.1.1 The Optimal Control Problem for the Planar Manipulator 70
4.1.2 Planar 2-link Model Functional Variations 71
4.1.3 Minimum-Time Maneuver Results 72
4.1.4 An Alternate Path Parameterization Method 75
4.2 Upper-Limb Biomechanical Model Performances 76
4.2.1 The Optimal Control Problem for the Biomechanical Arm 77
4.2.2 Biomechanical Arm Model Functional Variations 79
4.2.3 Increasing Computational Efficiency 80
4.2.4 Optimal Control Results for the Upper-Limb Model 83

5 Data Collection and Comparison Methods **90**

5.1 Human Motion Collection Objectives 90
5.2 Task/Movement Definition 91
5.2.1 Parameters 91
5.2.2 Constraints 92
5.2.3 Task Selection Guidelines 92
5.2.4 Task Descriptions 93

5.2.5 Task Collection Protocol 94
5.3 Defining Segment Kinematics 96
5.3.1 Optotrak Marker^{TM} Placements 96
5.3.2 Segment Definitions 98
5.3.3 Optotrak^{TM} System Setup 98
5.3.4 Formulating Segment Kinematics 98
5.4 Electromyography (EMG) Acquisition 105
5.4.1 EMG Electrode Placement 105
5.4.2 EMG Setup 106
5.4.3 Forming Muscle Activation Signals 108
5.5 Collection 109
5.5.1 Laboratory Setup 109
5.5.2 Subject Selection Criteria 109
5.6 Results and Statistics 111
5.6.1 Subject Anthropometry 111
5.6.2 Segment Kinematics 114
5.6.3 Muscle Activation Patterning 124
5.7 Limitations of the Collection 129
5.8 Comparison with Predicted Results 130
5.8.1 Spatial to Planar Projections of Observed Kinematics 130
5.8.2 Planar Trajectories for Tasks 1 and 7 131
5.8.3 Muscle Activations for Tasks 1 and 7 134

6 Conclusions **139**

6.1 Summary of Model Predictive Results 139

6.2 Assessment of Research 140

6.3 Contributions of Research 141

6.4 Future Research 142

6.4.1 Biomechanical Model 142

6.4.2 Optimization Methods 143

6.4.3 Collection Data Processing 144

6.4.4 Human Motion Prediction 144

6.5 Concluding Remarks 145

Bibliography **146**

A Detailed Implementation of Mechanical Models **155**

A.1 MATLAB Implementation of the Planar 2-Link Manipulator 155

A.1.1 planarEqns 155

A.1.2 costateEqns 156

A.1.3 simplePlanar 158

A.1.4 planarGrads 160

A.2 Upper-Limb Biomechanical Model 161

A.2.1 DynaFlex Input File: BioArm.graph 161

A.2.2 bioArm Input for DYNOPT 166

B Project Files: Descriptions and Locations **174**

B.1 Optimization Methods 174

B.2 Collection Data Processing 175

B.2.1 Processing Methods 175

B.2.2 Collection Data 175

List of Tables

3.1 Test function optimization results from the GA-SQP method 68

5.1 Collection trials and associated tasks 95

5.2 Marker numbers and their corresponding placement 97

5.3 Anthropometric measurements from the test subject 113

5.4 Derived segment lengths 114

B.1 Files implementing the optimzation methods 174

B.2 Collection data processing MATLAB functions 175

B.3 Data files generated by collection and processing 176

List of Figures

1.1 Overview of the motion prediction methodology 10

2.1 Joints of the human upper limb 13

2.2 Typical systems model for a muscle 17

2.3 Linear graph representation of a typical muscle model 20

2.4 Example biomechanical model of the upper-limb 21

2.5 Linear graph representation of the upper-limb model 22

2.6 Planar two-link manipulator 27

2.7 Planar two-link manipulator graph 27

3.1 Example of a single paramter q, for velocity profile parametrization 34

3.2 Linear interpolation of a prametrized control signal 46

3.3 Behaviour of penalty functions as $\rho \to 0$ [2] 54

3.4 Benchmark objective function with infinite local optima 66

4.1 The "bang-bang" optimal controls of the planar 2-link manipulator 73

4.2 The optimal trajectory for the planar 2-link manipulator 74

4.3 Feasible paths in the motion space of the planar 2-link manipulator 75

4.4 Functional organization of DYNOPT and GA integration 82

4.5 Triceps muscle with a linear two-point interconnection 84
4.6 Optimal motion of the biomechanical upper-limb model performing a vertical reach. 85
4.7 Model predicted optimal angular trajectories of a vertical reach task. 85
4.8 Optimal controls for the biomechanical arm performing a vertical reach 86
4.9 Optimal motion of the biomechanical arm model performing a vertical lift ($5kg$). . 87
4.10 Model predicted optimal angular trajectories of a vertical lift ($5kg$) 87
4.11 Optimal controls for the biomechanical arm performing a vertical lift ($5kg$) 88

5.1 Skeletal landmarks [47] for marker placement 96
5.2 Musculature of the human upper limb [47] 107
5.3 Camera and workspace layout 110
5.4 Upper limb segment frames defined by Optotrak markers 112
5.5 Movement of humerus segment markers over all trials of Task 6 113
5.6 Reconstructed motion for Task 6 115
5.7 Reconstructed and actual marker trajectories (markers 1, 5, 12) 117
5.8 Mean actual - mean reconstructed marker (1, 5, 12) trajectories 118
5.9 Sternum orientation relative to ground 119
5.10 Clavicle rotation relative to the sternum 120
5.11 Humerus rotation relative to the clavicle 121
5.12 Forearm rotation relative to the humerus 122
5.13 Hand rotation relative to the forearm 123
5.14 Mean and ($\pm$) one standard deviation of Teres major EMG 124
5.15 Mean and ($\pm$) one standard deviation of Anterior Deltoid EMG 125
5.16 Mean and ($\pm$) one standard deviation of Medial Deltoid EMG 125
5.17 Mean and ($\pm$) one standard deviation of Posterior Deltoid EMG 126

5.18 Mean and (±) one standard deviation of Pectoralis Major EMG 126

5.19 Mean and (±) one standard deviation of Biceps Brachii EMG 127

5.20 Mean and (±) one standard deviation of Triceps Brachii EMG 127

5.21 Mean and (±) one standard deviation of Brachioradialis EMG 128

5.22 Planar motion of the test subject performing task 1 132

5.23 Planarized angular trajectories of test subject performing task 1 132

5.24 Planar motion of the test subject performing task 7 133

5.25 Planarized angular trajectories of test subject performing task 7 133

5.26 Test subject deltoid and biceps activation levels from task 1 135

5.27 Test subject brachioradialis, triceps and teres major activations from task 1 135

5.28 Test subject deltoid and biceps activation levels from task 7 136

5.29 Test subject brachioradialis, triceps and teres major activations from task 7 137

6.1 A 3pt. muscle geometry vs. the current 2pt. muscle geometry 143

Chapter 1

Introduction

1.1 Motivation

As we enter a new millennium it is surprising how little we understand the basis by which animals and especially how we, humans, move. Tremendous innovations have come about in replicating movement using artificial neural networks and fuzzy-logic controllers that can imitate human/animal motion and even perform complex dynamic tasks such as walking (Honda robot, [81]), climbing, brachiation [23] and other movements [19], [18]. However, if we ask the question how a robot's or a patient's performance can be enhanced, via training, reconstructive surgery, etc., in order to perform better, these methods are unable to shed any light, because we fundamentally do not know what is "better". These soft systems methods are sophisticated interpolators that can handle highly nonlinear and multivariable relationships typical of biomechanical systems, but they all require a teacher, an example, or large data sets to learn these relationships. To use these methods to predict changes in behavior from alterations to the system model (e.g. from one individual to another or a normal limb to an artificial limb) is to go beyond the confines of their learned behavior and thus these methods would try to control the new system as if it were the old. In other words, they would try to develop a control strategy for an individual based on a norm and produce movement of an amputee based on a non-amputee. Although our goals may be to have these patient's move as normal as possible it may prove futile unless we understand what is "normal".

Is normal a repetitive kinematic pattern? Is it a set of foot plate reaction forces from a normalized cycle time in gait analysis? Are these patterns really an indication of some more fundamental behavior? If so, what is this fundamental nature of movement? It seems quite logical to assume that movement along with the evolution of species has evolved to exhibit qualities that are advantageous to the survival of the species. Perhaps we can think of our principal mode of locomotion (walking) to be an energy efficient and low stress method of transport [67], or running as a method of escaping immediate danger or capturing our prey. In both tasks the high-level goals are very different: one is to get from point A to point B as easily as possible and the other is to get from point A to B as quickly as possible. It is no wonder that both tasks exhibit such different dynamic behavior from the same biomechanical system. By understanding the dominant objectives perhaps we can better prescribe treatments to enable patients to perform more effectively rather than appearing to be more normal.

In the context of robotics, there is a comparable need to analyze and determine actuator controls that meet multiple objectives such as minimizing performance time, power consumption as well as wear and tear from repetitive stresses. By understanding how biomechanical systems solve problems of reliability, these same principles can be applied to the design and control of robots. For example robots can be made more robust by introducing redundancy [35].

1.2 Current Human Movement Analysis Tools

There are several tools available to the biomechanist today. Each provide a particular perspective on the nature of movement that is simplified and quite focused rather than providing a comprehensive description of the decisions of the control system [93]. As will be discussed in greater detail within the specific chapters herein, there are many levels at which predictive models are affected by modelling assumptions.

There is a methodological "tug-of-war" between biomechanical modellers, psycho-physicists, and kinesiologists/physiologists. Most physiology-based kinesiologists with their greater understanding of joint tissue properties, bone articulations, etc., argue that no model can accurately describe the real system and thus model-optimization based predictions are invalid. As a result they rely heavily on empirical studies to deduce control decisions for a specific task and individual and at most a specific group.

Psycho-physicists ([7], [43], [29]) observing the kinematics of movement discover common patterns that they can replicate with idealized models, and then infer rules about our neuro-control system. They seldom concern themselves with the mechanical dynamics. They argue that the dynamics of more complex models would vary from case to case, and that each model would require the derivation and solution of complex equations which in the end, would lose the commonality that they seek.

Modellers are concerned with the dynamics of everything, from a single contractile element ([30], [36]) to the complete movements of highly articulated humanoids ([90], [58], [81], [41], etc.) and generate various constitutive equations for muscles, tissue, and joints, with varying complexity and argue that the type of model and the fidelity required are dependent on the system/task being studied. The question from a psycho-physicist or a neuro-physicist is what is the appropriate model fidelity necessary to replicate the performance of a given task? The kinesiologist would respond by asking how one would measure the replication? Using kinematics, electromyography, or system dynamics independently the degree of replication could vary widely. To understand each perspective it is pertinent to discuss the tools employed in human movement analysis today.

1.2.1 Kinematics and Inverse Kinematics

Human kinematics are generally obtained directly from markers placed on limb segments. These provide a time history of point locations from which the velocity of each marker can be obtained by differentiation and acceleration by double differentiation in time. From this data researchers can plot the displacement, velocity and acceleration trajectories of selected anatomical locations. An inverse kinematics analysis is simply a coordinate transformation operation which enables the reinterpretation of an end effector location, for example, in terms of the coordinates of the system's joints. Because of redundancy — there are more joint coordinates than end-effector coordinates — inherent in biomechanical systems, solutions are seldom unique. A specific solution to the inverse kinematics problem is obtained by adding constraints to the modelled system that account for the articular limitations of joints; however, these are often still insufficient to provide a unique solution. Many researchers, especially those concerned with workspace ergonomics, go further to make assumptions about the "comfort" range of various joints and thus solve the system to maximize this quantity of comfort [43].

Inverse kinematics enables the extrapolation of geometric rules that are interpreted as psychomotor relationships such as Fitt's Law [29] and Donders' Law [76], which govern the motion of a complete kinematic chain to meet an observed (or desired) end effector (hand) trajectory. Once heuristics are developed they are easily encoded for the purposes of animating computer models [16] .

Unfortunately, we cannot effectively predict movement strategies without large repetitious datasets that provide sufficient correlations from which these rules relating joint trajectories to the end-effector path can be derived. In determining unique solutions when faced with redundancy, the resulting measure of comfort remains quite arbitrary and is significant only as an adjustment factor enabling the replication of the observed kinematics.

Fundamentally, this method can only provide superficial insight into underlying movement objectives that are strictly path dependent, such as path "smoothness" and object avoidance. The best possible conclusions are overly simplistic: the path being either smooth or not or the path hits or does not hit an obstacle, for example. The basis for why the path is smooth and the dynamics of avoidance remain elusive.

1.2.2 Inverse Dynamics

Inverse dynamics analysis begins to address the questions involving the generalized forces that are required to produce motion. Given position data through video or infrared markers, this data is converted to the segment accelerations via double differentiation or collected directly from accelerometers. Making assumptions about the nature of the joints (revolute, universal, etc.), the geometric and inertial properties of the segments, and finally with measurements of external forces (i.e. ground reactions) one can solve for the moments (forces) about (at) the joints necessary to drive the observed motion [33], [52].

The inverse dynamics analysis provides information on the net generalized forces that produce the resulting movement of the system given some assumptions of the biomechanical system parameters (i.e inertias, joint centres, etc.) and results are very sensitive to these model assumptions. Validating net joint torque and contact forces derived form link segment models and motion data is currently resolved by performing the inverse dynamics analysis twice — beginning at different ends of the link segment chain, known as top-down and bottom-up mechanical analyses

[39]. Beginning the analysis at the "free" end, i.e. the head or hand, where there are no external forces and again from the feet, where the ground reactions are available, should provide the same information about the necessary forces and moments.

Given the net moment about a joint, a decomposition method is then required to further determine the muscle contributions that produce those moments. Several decomposition assumptions including the minimization of individual muscular effort or some measure of expended energy ([17], [60]) have been used with limited success. In these cases there have been poor correlations with electromyography data [85] because the decomposition employed purely quadratic objective functions such that they were easy to minimize, and used simplified biomechanical models that made implicit assumptions. In addition, the static optimization of muscle force distributions at each instant is not equivalent to the same objective applied to the performance of the task over a continuous length of time [93]. It is difficult to assess which of these assumptions, either the optimization or the model, or both, results in the poor correlations.

As an example of problems that arise from implicit assumptions, we can look at the use of planar models. A planar model implicitly assumes no muscular effort by those synergistic muscles that limit motion to the given plane (e.g. use of oblique muscles controlling medial-lateral position of the torso during gait analysis in the sagittal) [28]. Furthermore, the explicit use of ideal joints implicitly assumes infinite joint reaction forces and thus eliminates the need for muscle co-contraction that may have otherwise contributed to joint integrity and stability. A more detailed analysis of modelling assumptions and their consequences is discussed in Chapter 2.

1.2.3 Electromyography

Electromyography (EMG) is the detectable electrical signal caused by a summation of individual muscle fibre depolarizations at an electrode location on or within the muscle. It is the depolarization of individual muscle fibres that triggers the contractile elements within the fibres to produce tension and it is their temporal and spatial summation which produces the resulting muscle tension ([79]). By recording these electrical signals, coordination and movement strategies are hypothesized and individual muscle forces are estimated by relating activation patterns to observed kinematics and (inverse) dynamics ([54], [59]).

Studies using electromyography are limited to surface muscles since needle electrodes are both too localized and potentially harmful to the test subject which makes it difficult to determine the activation (tension production) occurring internally, especially in large muscles. In addition, there is a highly nonlinear relationship between EMG activation and tension. For these reasons, surface EMG alone cannot provide specific information about every muscle's tension contribution.

EMG along with other methods (Section 1.2.2) can offer a means of physiological validation. Hybrid muscle force decomposition methods where EMG data and optimization methods via inverse dynamics are used in tandem have been proposed ([14]), however their promise have yet to be realized.

1.2.4 Forward Dynamics

The forward dynamics analysis method of human movement is based on producing movement of a suitable model with specific assumptions (i.e. optimal) made about the control signal such that it will replicate human behavior as observed from collected kinematics and EMG, for example.

Rather than try to deduce the underlying control decisions, forward models enable the control hypothesis to be tested directly by generating and applying hypothesized signals to the model actuators (muscles) and observing the results.

A forward dynamics analysis provides more certainty about an underlying control strategy ([90], [58] and [93]) because the control hypothesis is explicit and all model output is clearly deterministic. In inverse dynamics, the required decomposition methods are sensitive to a range of objective function, sampling, and model assumptions which are often implicitly made in the analysis by virtue of measurement devices and idealized simplifications.

Forward dynamics enables the direct testing of control decisions. There are many approaches to generating control inputs, but most notable is the work of Gentaro Taga [81]. His work includes control signal generators that are composed of a hierarchy of limit cycle attractors that are modulated by modelled "sensory" feedback from joints, muscles and contact forces, as well as "command" signals from an emulated central nervous system. The multi-layering permits the transition from attractor to attractor and thus exhibits changes in dynamic behavior such as transition from walking to running. The difficulty of using this approach in general is determining the parameters of the signal generators and especially the attractor shape and signal gain which

requires a lot of trial and error in order to produce motions that even resemble walking and running.

It is apparent that this method in particular is excellent for implementing controllers for walking or for complex multi-tasking robots. The concept of attractors themselves, once they have been refined to produce gait on a human model, encapsulate many key activation relationships of muscle actuators necessary to produce coordinated motion. This in turn enables psycho-physicists and other scientists to hypothesize the control structure in the human central nervous system.

The question that leaps to mind is how could this method produce a walking robot if we knew nothing about gait before hand? The many parameters for the attractors, sensory gains, etc., can only be tuned for a desired specified behavior. If this is the way in which the human control system operates, and we in fact learn by tuning our "parameters" then what is the guiding force or the desired goal that results in gait?

1.2.5 Optimal Control

The application of optimal control methods to human biomechanics is an extension to the forward dynamics approach and borrows the idea of optimization of motion from the resolution of muscle forces. The novelty in the optimal control approach is that optimization is now used to determine control signals or forces over the time continuum to optimally generate the net forces as moments that effect movement rather than statically decompose the moments at discrete instances from an inverse dynamics analysis.

Common to all optimal control methods is the hypothesis that the commands produced by the CNS and sent to the muscles are in some way optimal [15] for performing a specific task such as pointing, lifting or even jumping. Pandy *et al* produced a planar human model that produced the human counter-weighting (from a squat position, first moving down and back before forward and up) jump when trying to find the highest jump possible given fixed maximum muscular strengths and geometric and inertial parameters of the model [58]. Similar results were obtained by Zhao *et al* [94]. Recently Bobrow *et al*, have applied similar techniques to a biomechanical model of a weight lifter to reproduce the "clean-and-jerk" motion used by elite weight lifters to lift weights exceeding twice their body weight over their heads given the objective to maximize the weight lifted.

The results from this method are the control signals to actuators that optimally achieve the desired objective and can be compared to human performance in terms of both kinematics and EMG in order to determine if similar objectives applied to the model are applied by the human CNS in governing motion.

This method has one significant limitation which is the optimization of non-linear dynamic systems with continuous (infinite) design variable basis. The difficulties arise from the complexity of the equations of motion and thus the subsequent complexity of the objective function. As a result, models tend to be as simple as possible which makes comparisons with physical systems less valid. Even for simple models, optimal controls can be difficult to resolve and require sophisticated optimization techniques. This involves some parameterization of the design variables in order to create a finite basis problem to make the problem tractable numerically. A major concern are the initial approximations which are necessary for most optimization methods in order for them to converge to the optimal solution and not just a solution that resembles human motion. Finally, the computational costs are substantial and a significant investment in efficient numerical methods is necessary. For these reasons optimal control methods are not very popular in human biomechanics today although they can provide the most insight into the control decisions made in human movement.

Based on the aforementioned discussion, undoubtedly no single technique can provide all the information necessary to understand the basis for human motion and control. Although forward dynamics using optimal control provides the best correlation of hypothesis to performance, these results will still be limited to the assumptions of the particular model and require validation using any or all of the remaining methods.

1.3 Research Objectives

Formally, the objective is to develop a method for determining the optimal controls of biomechanical systems for varying system assumptions. These assumptions include control hypotheses that optimize efficiency, stress distributions, trajectory smoothness, etc., or any combinations thereof as well as model assumptions such as joint and muscle models, and other tissue properties. The resulting framework should be extensible and reusable to facilitate both changes in the performance objective(s) as well as changes to model components.

Robotic applications will be a direct by-product since the human locomotor system is a superclass of robotic systems. In other words, a method that can deal with the nonlinearities and complexities of human biomechanical models should perform well with idealized joints and joint torque actuators.

It is not our purpose here to discuss control theory in the typical sense of feedback controllers, which follow a particular behavioral pattern, but rather to determine and understand why a particular behavior or motion is chosen by the human control system and to chose the optimal movement when considering robotic systems design.

1.4 Control Prediction Methodology

This thesis embodies a culmination of the aforementioned techniques in order to develop a comprehensive modelling methodology for creating and validating predictive models. The high level overview of the computational components and their interconnectivity (Figure 1.1) will act as a road map for the discussions in future chapters. Briefly, the work has been divided into four computational blocks. The first block is the generation of the upper-limb model and the formulation of the equations of motion necessary to simulate upper-limb motions [1]. The control of the model represents the second computational block. The controls are determined by applying optimal control theory to a suitable model with user-defined objectives. The third block contains the collection of human motion data and the necessary processing methods for comparison with model predicted output. The comparison itself represents the last block. The individual sections of the report correspond to major system blocks of the overview. The order in which they are presented is as follows :

1. Upper-limb dynamic modelling via graph-theoretic methods, Chapter 2
2. Control strategy determination using optimal control theory and evolutionary methods, Chapter 3
3. Integration of these components to produce optimal motion predictions, Chapter 4

[1] Note the bidirectional arrow bewteen the data collection and task database represents parameters derived from the collection process that are necessary for the model and optimization methods

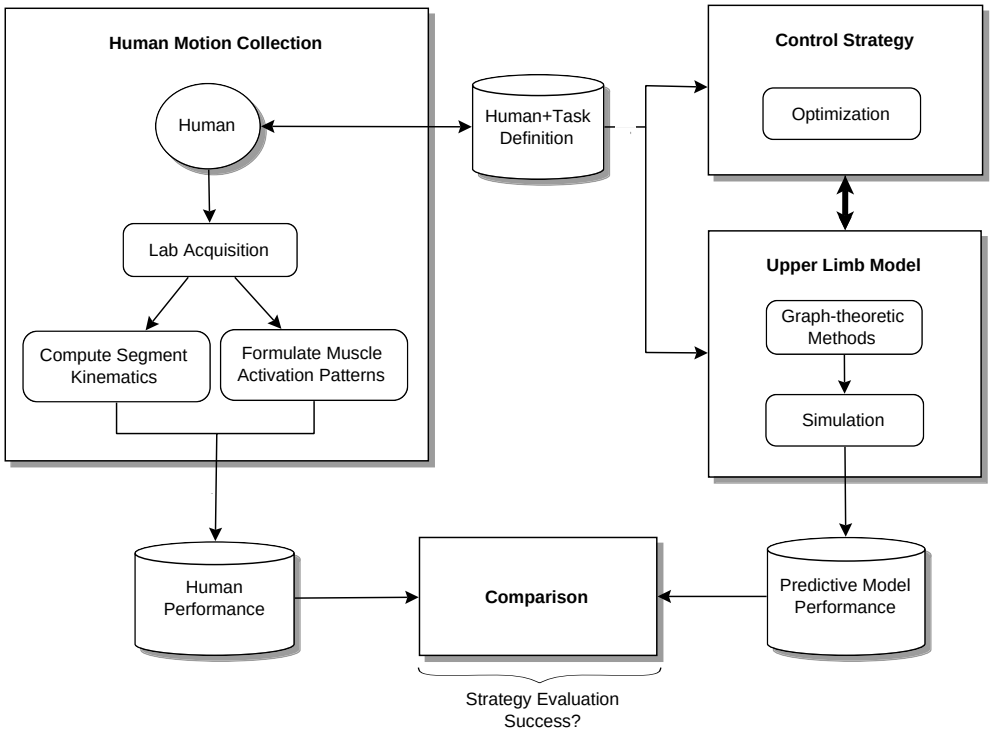

Figure 1.1: Overview of the motion prediction methodology

4. Collection and processing of human motion data for comparison with model predictions, Chapter 5

5. Conclusions are drawn from model predictions and comparisons with human performance data, Chapter 6

Chapter 2

Human Upper Limb Mechanical Model

Previous models of the upper limb have been developed and most notable is the recent work of Maurel et al [47]. They develop a very accurate biomechanical model by reconstructing the left upper limb segments, tissue, and joints from the Visible Human imaging data. This work represents the bulk of the topological aspects necessary for the creation of an upper limb model and includes joint articulations and the resulting degrees of freedom, inertial properties, most upper limb muscles and their anatomical points of origin and insertion as well as their lines of action. The actual dynamics of the problem and specifically the generation of the equations of motion from such complex models remains the focus of current research. The work of Raikova (1992) [65] using a systematic application of Newton-Euler formulations to derive the equations of motion, by hand, for models not nearly as complex of that of Maurel et al, still represents the state of the art in biomechanical modelling.

After a brief review of upper limb biomechanics, we present our approach for generating the equations of motion for complex models.

2.1 Review of Upper-Limb Biomechanics

The biomechanics of the upper-limb presents several fascinating problems in creating models of sufficient fidelity to capture both the dexterity and the physical limitations that human joints both enable and impose. Furthermore, the driving forces of redundant musculo-tendinous actuators adds yet another dimension of complexity especially in determining individual actuator control. The systematic approach to modelling any system begins with making conceptual assumptions and simplifications that enable the creation of a model. Specifically in biomechanical modelling, these assumptions simplify the model such that mathematical relationships can be formulated into solveable equations of motion.

2.1.1 Upper Limb Kinematics

The movement of the arm is the result of the relative motions of several segments of the upper limb, Figure 2.1. From the sternum distally, these include the scapula, clavicle, humerus, radius, ulna, and the hand. The segments of the hand are being lumped into a single body within this study since, for the time being, we are more concerned with determining the gross motions of the arm. If the sternum is taken as the ground-body then all the movement can be reasonably described by the relative changes in orientation of each segment, described by a time-varying relative rotation matrix $\mathbf{R}_i$, except for the motion of the scapula. The scapula can be considered to have three degrees of freedom due to the constraints of the scapulo-thoracic joint, which allows the scapula to translate and rotate in contact with the thorax surface [57]. However, since the segmental chain from the sternum to the hand is completed by the connection of the humerus to the sternum via the clavicle, the movement of the scapula is fully constrained by the acromio-clavicular and the aforementioned scapulo-thoracic joint. Thus, the movement of the scapula can be ignored and approximated at a latter date by modelling the scapulo-thoracic surface constraint and knowing the acromio-clavicular position. This is necessary since scapular movement, which is primarily below the surface of the skin, is impossible to capture using markers (light emitting diodes) that are placed on the surface of the skin.

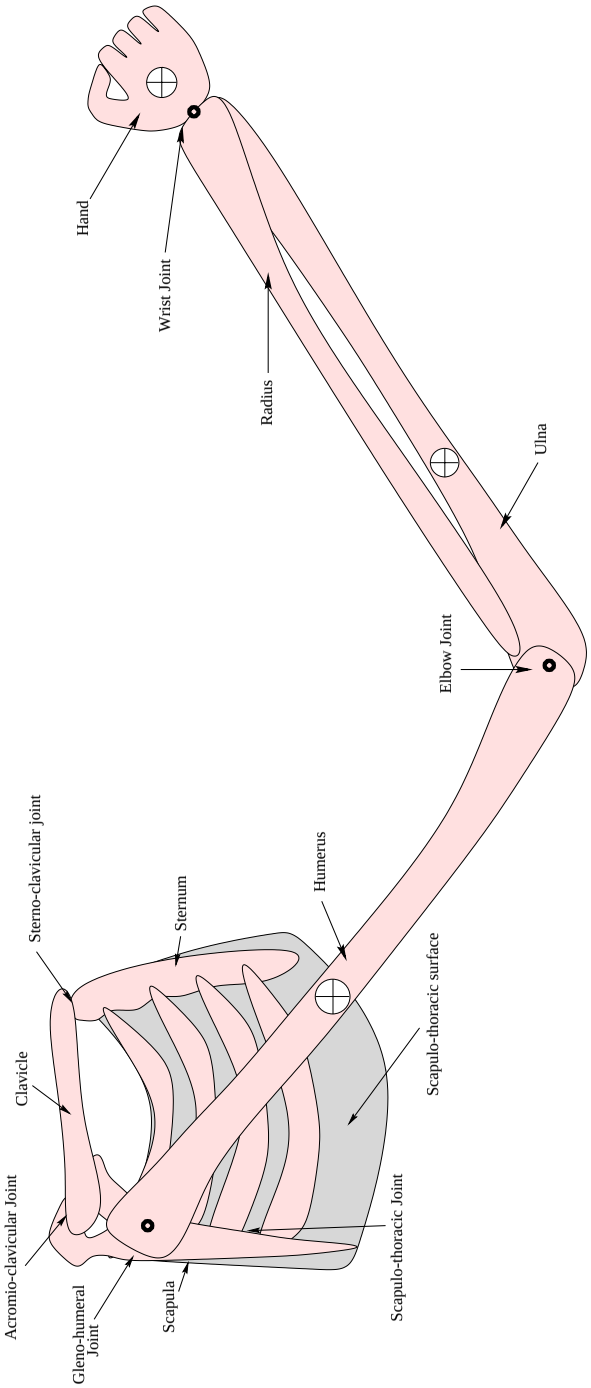

Figure 2.1: Joints of the human upper limb

2.1.2 Bones and segment models

Bones and limb segments are not rigid bodies but then again nothing in reality is truly rigid. In comparison to the forces applied to bones the deflection (deformation) of bone is generally small and considerably smaller than the errors incurred by skin-marker movement for normal tasks. Thus assuming bones to be rigid bodies is reasonable in this context. For specific or extreme loading situations, however, it is necessary to consider bones as being flexible where the mechanical properties such as Young's modulus for the bones must be attained by experimentation, [22]. Although the mass of a segment does not change, a muscle's mass distribution varies with its length and tonicity. Inertial changes due to muscle deformation is considered negligible by most researchers who study whole body movements [3], [33] and [52]. Once again for more specific models, we can assign mass properties to the muscle model which is described in greater detail in the discussion of muscle models, Section 2.1.4.

Inertial properties are generally estimated by the inertia of representative cylinders from the length and other geometrical features of segments or by scaling according to total body mass and height from normative data. Both methods contribute errors in inertia estimates and thus to the resulting (inverse) dynamics analysis with no definitive advantage for either method [40]. Using data from the Visible Human project (VHP) one can produce precise volumetric and approximate density distributions to derive accurate estimates of segment inertias. However, the errors of using any scaling method to then relate the VHP inertias to a particular test subject will still remain.

2.1.3 Joint models

In concerning ourselves with the relative rotations of the limb segments we have in fact assumed that all segments are interconnected via joints that enable rotation only, and thus are either spherical, universal, or revolute joints. In reality, joints can separate and deform tissue, which results in translational degrees of freedom. Yet other joints, like the gleno-humeral joint, act very much like cams where the head of the humerus rolls upon the glenoid surface thus resulting in the translation of the joint centre. Unfortunately, in the case of the head of the humerus, it is not a perfect cam either in that it rolls and slips simultaneously [57], which is a phenomenon which adds significant complexity to a model. For the purpose of replicating human kinematics from a model one has to observe the degrees of freedom that a joint exhibits as well as the degrees of

freedom necessary to perform the task that is to be modelled. This is common practice in human biomechanics and thus the use of planar models are quite popular for capturing the observable kinematics resulting in simpler system equations.

The problem of using over-simplistic models, particularly planar models, are that, often, they implicitly assume all joints to be revolute joints. Revolute joint reaction forces maintain planar equilibrium (no net forces acting out of the plane) however, in comparison to electromyography (EMG) from muscles which do no act in a single plane, there must be counter forces that maintain planar motion which results in the synergy of muscles observed in the real system. For planar models it is unnecessary for the control method to predict appropriate muscle synergy in order to maintain planar motion.

The danger of using ideal joints is that they will not favor the prediction of co-contraction in a dynamic model where minimization of energy or effort is the only objective. This is because co-contraction is inefficient from an energetic point of view and thus if the "pin" of the revolute or the "socket" of a spherical joint can create infinitely large reaction forces to maintain the joint integrity, then there is an unrealistic advantage at selectively loading the largest and strongest muscles to create the motion although it may require unrealistic reaction forces. Therefore, in comparison with EMG data, model predicted controls are seldom similar although the kinematics may be very similar to the observed motion [14], [62]. Glitsch et al [28], concluded that idealized models, especially planar models, could not be used to predict muscular efforts required for joint stabilization unless subsequent compensation was made in the analysis or decomposition methods. Thus, in using idealized joints in general, it is necessary to include some penalty measure of the destabilizing forces in the joints in order to predict realistic muscle control.

In addition to the degrees of freedom of the joints, are the effects of passive structures such as bone articulations, ligaments and cartilage that limit joint motion. Joint articulations and specifically modelling the resulting contact results in discontinuous functions. Translating these functions to the system dynamic equations results in many constraint forces that are state dependant requiring another set of equations to describe the effects when the constraint is active. In other words, when a joint reaches an "extreme" angle, another equation must describe this "locking" state of the joint. The resulting numerical solution is not trivial and can significantly increases the complexity of numerical solutions. The alternate and the more popular method for optimal control problems is to include hard constraints on the system kinematics such that joint

angles are bounded within "realistic" ranges. A more detailed discussion of this method is given in the discussion of optimal control methods, Chapter 3.

Connective tissue such as cartilage, tendons and ligaments have more continuous force curves that can be modelled to a limited degree, by mechanical components such as springs and dampers, which attempt emulate the elastic and viscous properties of tissue. The characteristic equations for these components are generally non-linear with greater responses at extreme postures and at ballistic speeds [83]. These structures typically contribute very little to the joint moments; however, in circumstances of extreme loading they resist joint separation. Alternatively then, knowing the loading thresholds, in terms of tensile and shear forces at which these passive tissues will fail, these thresholds can be used to set limits on joint loads within an optimization method.

For more detailed analysis of tissue mechanics resulting in elastic and inelastic deformations, finite element methods are required. Specific tensile, compressive, shear, and torsional stress-strain relationships are available for various tendons, ligaments [89] and articular cartilage [82] are then necessary for finite element methods.

2.1.4 Muscle Models

The literature on muscle models is extensive; however there are only two fundamental approaches — constitutive and empirical models. Most famous of the empirical models is Hill's model [32]. Hill's model fits a muscle force production curve, which is a function of its length, velocity, and activation, to corresponding measured values for isometric and concentric contractions. Essentially it models the relationship of muscle force (f_m) to length (l) and force to velocity (v) as hyperbolic relationships governed by two corresponding exponents, α and β. These act as "shaping" parameters which change depending on the type of task and the muscle involved, but are generally between 0.5 and 1 [86], with the maximum isometric contraction force (F_{max}) being a constant multiplier modulated by a normalized activation ($0 \leq A \leq 1$) signal:

$$f_m(A, l, v) = F_{max} * A * \left[1 - \left(\frac{|l - l_0|}{l_0}\right)^{\alpha}\right] * \left(\frac{v_{max} - v}{v_{max}}\right)^{\beta} \tag{2.1}$$

On the other side of the spectrum are the biophysical synthesis models of Huxley [36] and later Hatze [30] that attempt to describe the muscle behavior based on the underlying biophysics of

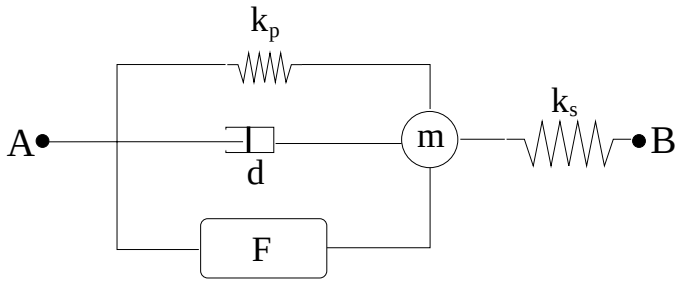

Figure 2.2: Typical systems model for a muscle

muscle fibres. Huxley described the fundamental contractile mechanics of muscles by the sliding filament theory of sarcomeres. Here, contractile dynamics are affected by concentrations of calcium ions, which are modulated by "neural drive", availability of energetic ATP, the overlap of thick and thin fibres of the contractile mechanism and the velocity at which they slide past each other, as well as the number of fibres and their geometrical arrangement in the muscle. In Hatze's biocybernetic model of muscles there are numerous parameters that govern the functional relationships of these variables and describe tissue properties such as elasticity of filaments and tendons and the viscosity of connective tissue which are all determined experimentally. To determine the behavior of the whole muscle requires the summation of the contributions of each contractile element in a way that is similar to a finite element analysis which then results in the solution of partial differential equations (PDEs). The consensus amongst researchers ([87], [93], [92]), is that the numerous parameters and the computationally intensive solution of PDEs make these models impractical and ultimately no more reliable since errors in parameter estimates are propagated to model output.

Generally a compromise between these methods is used in large scale biomechanical models. A systems model is developed using a Hill type contractile element (F) while incorporating additional physiological components such as serial (k_s) and parallel (k_p) elastic elements as well as viscous (d) and inertial (m) components to model higher order effects [27], Figure 2.2. The parallel elastic and viscous elements are the net elastic and viscous effects of the muscle fibres and tissue between fibres and the series elastic element represents the elasticity of connective tissue especially the contribution of the muscle tendon.

The origin and insertion points (A and B) of a muscle and its lines of action (pull) on multiple limb segments can have a large impact on the muscle's contribution to the joint moments [92].

These points are generally determined by normative ratios of segment lengths. Unfortunately, applying scaling methods from normalized data always introduces error into the model.

2.1.5 Remarks on Biomechanical Modelling

It is fair to say that more complex joint models provide better model kinematics capable of matching human kinematics and similarly complex constitutive models for muscle and connective tissues can better emulate dynamic phenomenon. Unfortunately, even if the fundamental biophysics were advanced enough to provide the constitutive equations for tissue deformation, nonlinear elasticity, etc., that hold under varying loading conditions, the solution of the system differential and partial differential equations with available numerical methods and computers would be prohibitively demanding on computational resources. However, given the rapid increases in computational speed and the advancements in numerical methods this will not be the case in the near future. What is required as a result, is a modelling environment that enables the evolution of biomechanical models which parallel technological progress.

2.2 Current Mechanical Modelling Methods

We have introduced the fundamental components for the biomechanical modelling of the upper limb but the problem remains how we can map these individual component models to mathematical equations necessary for simulation and optimization. There are many mechanical modelling and simulation packages such as ADAMS, DADS and Working Model, just to name a few, that internally determine and solve the system differential equations from user's assembly of components via a graphical user interface. In these applications there is a segregation of the assemblage (managed by the user) and mathematical formulation (performed by the application) which has its advantages and disadvantages. The main advantage is that the user need not be an expert in dynamics or numerical methods to create complex models and to produce simulations. The primary disadvantage is that the user has no control over the methods used to formulate the equations nor the numerical techniques used to solve the system equations. For simulation purposes alone this is not a significant problem; however for optimization and especially for optimal control (Chapter 3) the numerical methods used are critical for convergence as well as for providing tractable solution times. Simulation times in commercial packages are slower due to their

generality which requires more internal application communication, and the use of general and robust methods rather than the most efficient methods. Finally commercial packages are limited to the set of components that have been implemented. For example if the user desires to convert a gleno-humeral joint model from a universal joint to a spatial cam with slipping this would be impossible with the typical palette of components consisting of ideal joints.

As a result, many researchers still derive equations of motion for their models by hand using link segement models and Newton-Euler equations [33], [65] or classical analytical methods such as the method of Lagrange [15]. Unfortunately, hand derivation of complex systems is both time consuming and highly prone to errors especially for complex spatial models. Thus a significant motivation for using planar models is merely to simplify the formulation process. The following graph theoretical approach to modelling offers several of the advantages of commercial packages with the freedom of hand derivation.

2.3 Biomechanical Modelling using Graph Theory

The conceptual origins of the graph theory as a unified systems modelling approach stemmed from researchers [38] striving for a systematic method of describing physical systems mathematically that was not limited to the physical laws of a particular discipline. They identified that any discrete (finite number of components) physical system could be described mathematically given: 1) a description of the interconnections of system components, and 2) the physical laws that governed the behavior of a component that could be expressed mathematically. Thus, borrowing from linear graph theory as strictly a tool to solve combinatorial problems since the time of Leonard Euler ($\approx$1600, who used linear graphs to analyze paths on a network of bridges), they described system interconnections by a linear graph where the lines or edges of the graph represented the physical components and the nodes were reference points representing connection points and/or other points of interest.

The linear graph representation (Figure 2.3) of the aforementioned muscle model (Figure 2.2) illustrates the interconnectivity of mechanical components of a muscle; however it could easily represent an electrical system composed of a current source, resistors, a capacitor, etc. In fact the same topological equations would be derived regardless of the components. The interconnectivity is described mathematically by an incidence matrix which is an n (number of nodes) by m (number

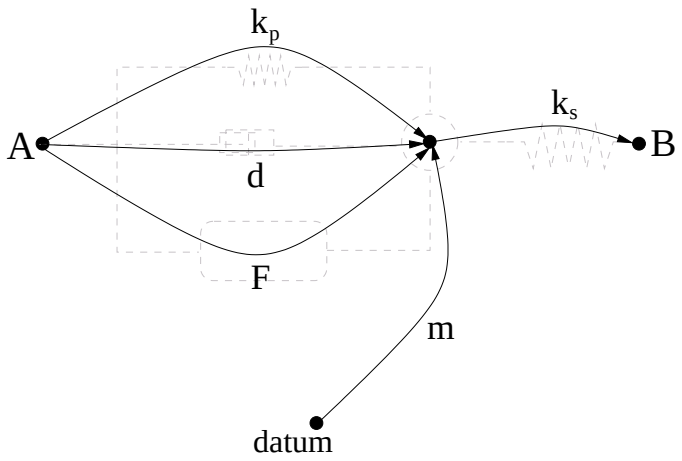

Figure 2.3: Linear graph representation of a typical muscle model

of edges) matrix which describes the direction of incidence of edges at nodes by ± 1; otherwise the entry is 0. From the incidence matrix topological equations resembling nodal balances and loop closures are derived. The individual components relate system variables such as flow and pressure, current and voltage, force and displacement, torque and orientation, etc., associated with the edge. The general concept is that all system components relate "through" variables with "across" variables defined by physical laws, which have been determined empirically. The specific equations describing a component's behavior is known as its terminal equation. The two sets of topological equations (reduced nodal or cutset equations and circuit equations) plus the terminal equations provide a necessary and sufficient set of equations to describe the behavior of any system given a linear graph representation. Details of the general systems theory are described by Chandrashakar and Savage [68].

The graph-theoretic method (GTM) applied to mechanical systems has recently taken significant strides in its evolution and McPhee presents a concise history of the method and a detailed guide to its application to multibody dynamics [48]. From this standpoint, one can see the graph as the mechanical model because of the one-to-one mapping of physical components to the system graph (Figure 2.5) as well as its similarity in appearance to the physical system (Figure 2.4). This example upper-limb model is the base biomechanical model which will be used in later sections to derive the equations of motion used to investigate optimal control strategies. The model consists of four bodies: the scapula as the ground body, the humerus, the forearm (not individual ulna

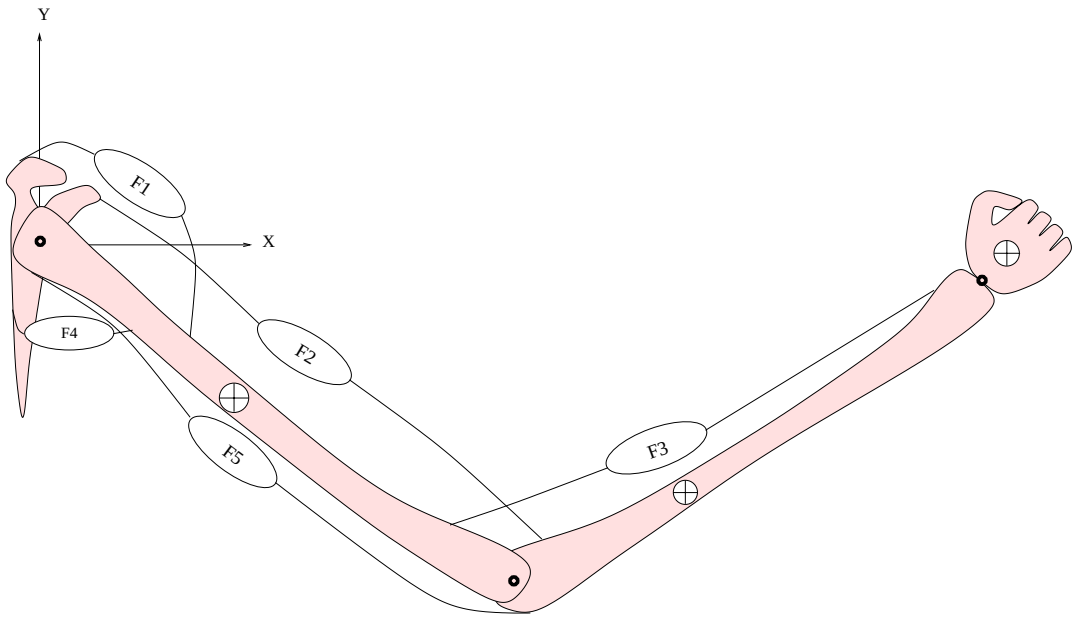

Figure 2.4: Example biomechanical model of the upper-limb

and radius) as well as a lumped hand segment (Section 2.1.2). The type of joints and muscle models are not specified in either the model or graph. This demonstrates the topological nature of the linear graph whereby the specific nature of components such as joints and muscles does not affect the graph representation. In this way the graph is a powerful tool for managing complex mechanisms so that the topology can be fully described without the concern of the behavior of individual components. Especially for spatial systems, the graph can be a significantly easier method of organizing the system topology than the spatial arrangement of components in a computer aided design (CAD) or modelling environment.

For the GTM applied to multi-body systems, nodal equations can be understood as force or torque balance equations. Specifically, the cutset equations represent force (torque) balance equations from free body diagrams of subsystems that provide fewer dynamic equations than performing an analysis at each node.

A significant step in the evolution of the GTM has been its application to spatial systems. Unlike particles or planar kinematics, where rotations are either non-existant or scalars, rotations of spatial mechanisms do not satisfy classical "loop" closure equations. This is because spatial rotations cannot be described as scalars or vectors and thus the sum of rotations (unlike trans-

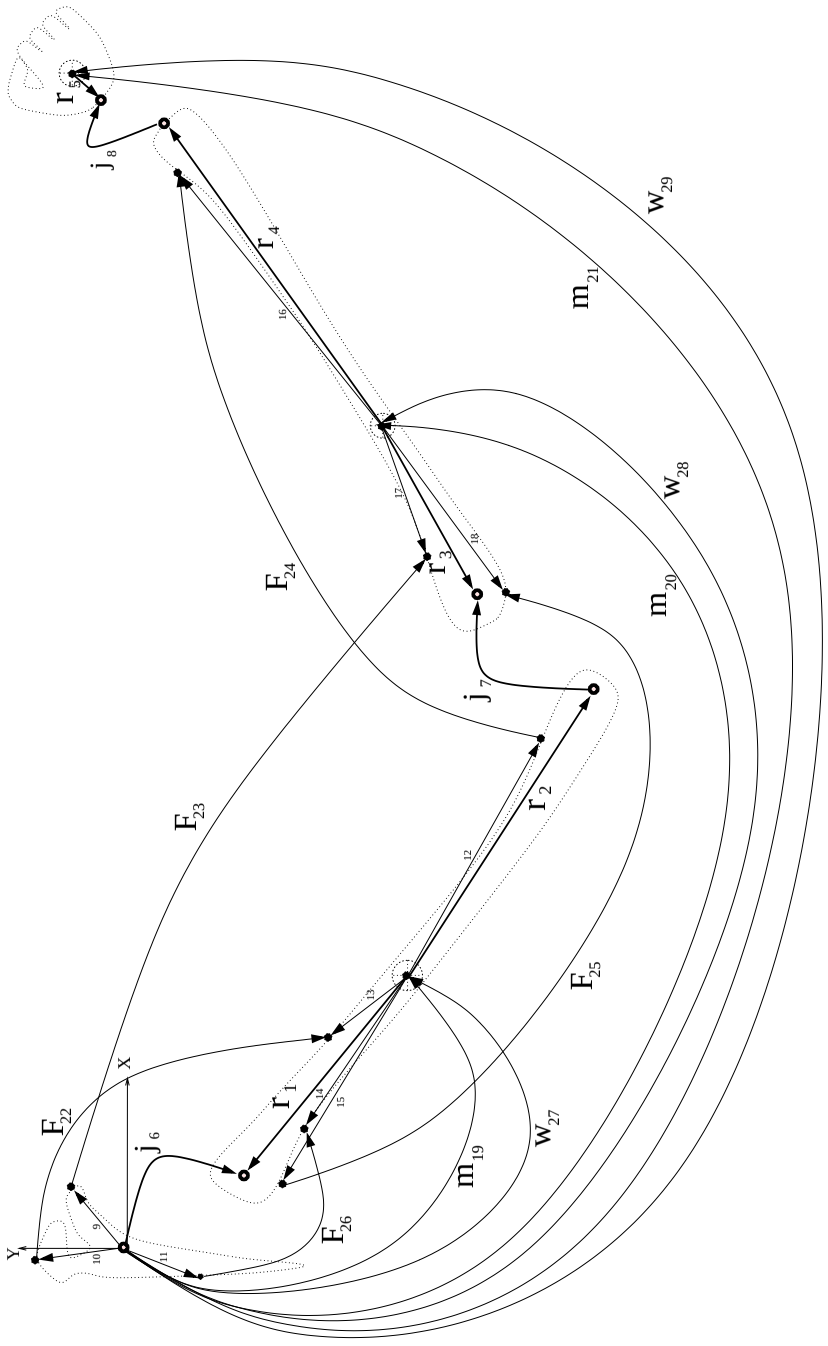

Figure 2.5: Linear graph representation of the upper-limb model

lational displacements) is meaningless. Thus, the idea of reference frame transformations was introduced in re-thinking loop closure equations to be the products of rotation matrices, which are transformations of orientation from frame to frame around a closed circuit that must bring us back to the starting orientation. The translational and rotational loop closure equations, together, result in the system's kinematic equations [88].

Finally the user has control over what form the system equations will take by the selection of edges that compose the spanning tree. A spanning tree is any set of edges that interconnects all nodes such that the path between any two nodes in the graph is unique (i.e. no closed loops). Edges in the tree are known as branches and all non-tree edges are known as chords whose across variables (displacement or orientation) can be resolved in terms of branch coordinates using the branch transformation from the kinematic equations. By selecting mass elements into the tree the resulting equations will have the absolute coordinates (position and orientation) of the bodies as branch coordinates that appear in the final form of the equations (Example [55]). However if joint coordinates are selected instead, then the final equations will contain just the joint coordinates (only one angle for a revolute joint) and thus provides a reduced set of equations.

The final equations of motion are differential algebraic equations (DAEs) of the form :

$$\{\Phi(q)\} = 0 \tag{2.2}$$

$$[M]\{\ddot{q}\} + \{\Phi\}_q^T\{\lambda\} = \{\Im(q, \dot{q}, t)\} \tag{2.3}$$

where $\{q\}$ is the vector of branch coordinates and $[M]$ is the symmetric and positive definite system mass matrix. $\{\Phi\}$ is the vector of kinematic constraint equations where the q subscript is the derivative of the constraints with respect to the branch coordinates which results in a matrix also known as the system Jacobian matrix in robotics, [51] and [78]. The Lagrange multipliers $\{\lambda\}$ are the joint reaction forces or moments that enforce the kinematic constraints. The right hand side contains the generalized forces from system drivers, state dependant forces such as spring-dampers, as well as centripetal and coriolis forces as the case may be. If absolute coordinates are used, by having masses in the tree, then Lagrange multipliers will appear such that there is one λ_i per mass degree of freedom that is eliminated by the joints. However if using joint coordinates, Lagrange multipliers only appear if there are kinematic constraints such that joint coordinates

are not independent and then they correspond to the joint forces or torques necessary to maintain the dependency of the joint coordinates.

2.3.1 Component Paradigm

The individual description of a graph's edges by terminal equations lends itself nicely to a component paradigm. In such a paradigm the analyst need only identify the topology of components in the system and then "plug in" the appropriate component model, via their terminal equation, from a database containing various templates of components. Borrowing from the GTM applied to multibody dynamics, these include masses, rigid and or flexible arm elements, joints, actuators, springs and dampers. Within this framework bones are mass elements consisting of rigid or flexible arm elements, muscles are composite elements composed of nonlinear actuators, springs and dampers (Figure 2.2), and joints can be any kinematic constraint.

In the upper-limb graph (Figure 2.5) we can see that the inertial aspects of the system dynamics are captured by mass edges ($m_{19..21}$) that act at the centre of mass of the segments and include both translational ($\{F\} = -[m]\{\ddot{x}\}$) and rotational ($\{\tau\} = -[I]\{\dot{\omega}\}$) inertial forces (torques). Arm elements (r) identify connection points on the body including joint locations ($r_{1..4}$) and origin and insertion points of muscles (only numbered edges 9..18). Joints have only been classified as elements ($j_{6..8}$) that impose kinematic constraints and thus could be revolute, universal, spherical joints and cams or any user defined kinematic constraint. Muscles have been shown by single edges and can be simple force drivers or thought to encapsulate their own subgraph as in the muscle model graph (Figure 2.3). Note that the terminals A and B are connected to the origin and insertion points of bone segments in the musculo-skeletal system which in turn are connected to the inertial frame. For a composite edge the corresponding terminal equation is a composite equation with contributing terms from each of its elements. The elements used in muscle models are actuators governed by some function of the control input and muscle length and velocity, springs for the elasticity and dampers for viscous properties. These three components are commonly used and are often grouped together in parallel to form a single spring-damper-actuator (SDA) element in the GT modelling of mechanical systems.

Although many component terminal equations are already available, one is not limited to this selection and can certainly introduce any variety of new terminal equations. For example,

a specific terminal equation modelling the scapulo-thoracic joint constraint could be introduced. In this case the joint enables the concave surface of the scapula to slide on the matching convex surface of the thorax while enforcing contact. Similarly compound components, such as muscle models, can be lumped into a single edge with one terminal equation. A discussion on the inclusion of additional terminal equations into the solution method is included with the discussion of the computer automated formulation process, Section 2.3.2.

2.3.2 DynaFlex: 3D Mechanical System Modeller

The most significant reason for using the systematic GTM is to automate the formulation process and to eliminate the manual task of deriving system equations and the errors that come along with it. As mentioned previously this method is fully described in [48] in which the formulation can be broken into systematic procedures that can be implemented on a computer. DynaFlex, developed by Shi and McPhee [75] at the University of Waterloo, is a Maple based application that implements the GTM to produce the symbolic equations of motion from a graph definition of the mechanical system via an input file. Furthemore, this algorithm defines the terminal equations in terms of the virtual work contribution of the component as a function of its virtual displacement. The use of virtual work has the same advantages as its application in classical analytical mechanics where the forces of all non-working joints and other components can be eliminated from the dynamics equations. The final system of equations, therefore, are free of Lagrange multipliers and there is a subsequent reduction in the number of equations — one per system degree of freedom (DOF). The general form of the equations are DAEs free of Lagrange multipliers .

$$[\tilde{M}]\{\ddot{q}\} = \{\tilde{\Im}(q, \dot{q}, t)\} \tag{2.4}$$

$$\{\Phi(q, t)\} = 0 \tag{2.5}$$

The reduced number of equations (1 per DOF) means the subsequent mass matrix $[\tilde{M}]$ must be nonsymetric for dependant generalized coordinates $\{q\}$. The constraint equations $\{\Phi\}$, generally non-linear algebraic equations, provide the remaining equations to fully describe the system. For open loop systems, however, this method provides the minimum number of independent ordinary

differential equations, one per degree of freedom, instead of a system of DAEs which are generally more difficult to solve numerically [75].

Another significant advantage of using energy methods is that it can resolve the dynamics of flexible bodies without resorting to finite element methods. The details of this new method is described in [74]. From this standpoint we will describe how to create the necessary input file such that DynaFlex can generate the symbolic equations of motion.

2.3.3 Generating Equations of Motion with DynaFlex

DynaFlex requires a representation of the graph via an input file. This file defines the number of nodes and edges in the graph as well as the individual components in the graph and their interconnection. Components are identified according to DynaFlex's component list which include a variety of joints, drivers, the SDA, and arm elements.

The formulation of the equations of motion for a planar two-link manipulator, which will be used in later chapters, are derived here using DynaFlex as an illustrative example. The manipulator (Figure 2.6) in the horizontal plane has two motors (torque drivers), one per revolute joint, and a payload (m_3) at the distal end of the second link (l_2).

The corresponding graph (Figure 2.7) of the planar manipulator looks very similar to the mechanical system. The subscript of the elements in the graph correspond to the edge number. The edge r_1 identifies the revolute joint h_5 connecting the two links with respect to the centre of rotation of link one (m_7) and is the location of the revolute joint h_4 fixing the link to the datum (node 7). The inertial effects associated with the second link, edge m_8, have to be associated with its centre of mass since it is not pinned to the inertial frame. Edges r_2 and r_3 identify the joints h_5 and the weld joint w_7, respectively, with their sum equivalent to l_2 of the mechanical model. The weld joint w_7 joins the centre of mass of the payload (m_9) to the second link. The two motors, are shown as edges T_{10} and T_{11} which are torque drivers acting on the the two revolute joints. The selection of the tree is easily made. Considering that the two revolute joints contribute two branch coordinates and the weld joint and rigid arm elements contribute none, we can get the minimum number of equations, 1 per DOF, by selecting these elements into the tree and these edges are shown in bold. Selecting the masses into the tree would result in 18 dynamic equations (6 per body) and 16 constraint equations (5 per revolute joint plus 6 for the weld joint) in 3D

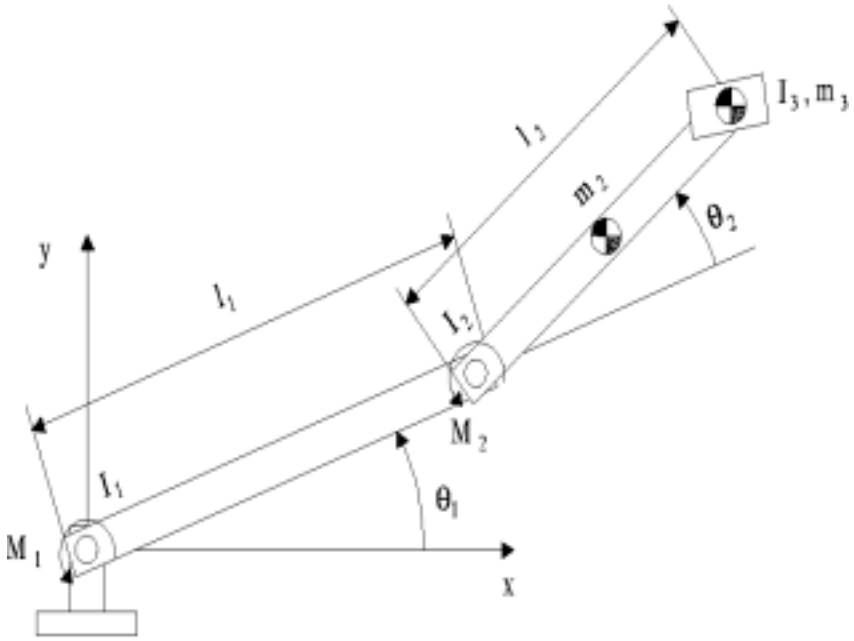

Figure 2.6: Planar two-link manipulator

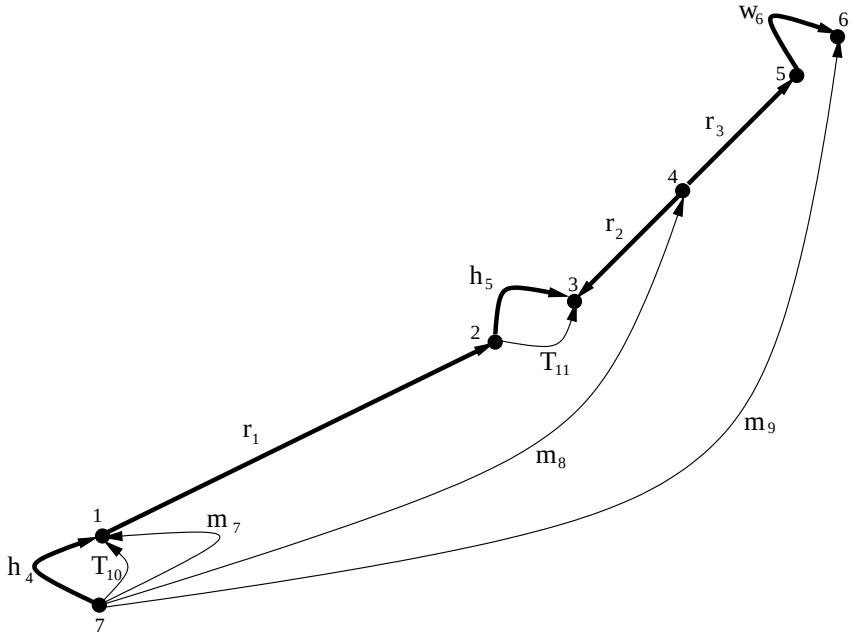

Figure 2.7: Planar two-link manipulator graph

space.

Once the graph is drawn and a tree selected, an input file transcribing the topology and system components must be generated. The following is the transcription of the planar two-link graph into the DynaFlex inputfile.

```
# DynaFlex input for a 2-link planar manipulator.
# ITEM ONE: number of nodes and edges in the system graph
NOofedges:=11;
NOofnodes:=7;
Datum:=7;
# Note that Datum stands for the ground node.

# Identify the edges
edge[1]:=table([(1)=Y,   # identify if edge is in the tree
                (2)=[1,2], # specify start and end node
                (3)=AE_R,  # edge type, i.e. AE_R is a rigid arm element
                           # DynaFlex documentation contains a complete
                           # set of available component types
                (4)=table([coords=[r1,0,0] ])
                           # component specific data
              ]);

edge[2]:=table([(1)=Y,   # edge is in the tree
                (2)=[4,3],
                (3)=AE_R,
                (4)=table([coords=[-r2,0,0] ])
                           # position in mass frame
               ]);

edge[3]:=table([(1)=Y,
                (2)=[4,5],
                (3)=AE_R,
                (4)=table([coords=[r3,0,0] ])
               ]);

edge[4]:=table([(1)=Y,
                (2)=[7,1],
                (3)=JE,    # Joint element
                (4)=RV     # revolute, default rotation about local z-axis
               ]);

edge[5]:=table([(1)=Y,
                (2)=[2,3],
                (3)=JE,
                (4)=RV
```

```
            ]);

edge[6]:=table([(1)=Y,
             (2)=[5,6],
             (3)=JE,
             (4)=WELD   # weld, frames fixed relative to one another
            ]);

edge[7]:=table([(1)=N,   # not in the tree
             (2)=[7,1],
             (3)=BE_R,  # rigid body (mass) element
             (4)=table([inert=[[0,0,0],
                               [0,0,0],
                               [0,0,J1]], # planar rotation about z
                        mass=m1          # J1 is the inertia of body 1
                                         # w.r.t to the pinned end
                       ])
            ]);

edge[8]:=table([(1)=N,
             (2)=[7,4],
             (3)=BE_R,
             (4)=table([inert=[[0,0,0],
                               [0,0,0],
                               [0,0,J2]], # w.r.t to centre of mass
                        mass=m2
                        ])
            ]);

edge[9]:=table([(1)=N,
             (2)=[7,6],
             (3)=BE_R,
             (4)=table([inert=[[0,0,0],
                               [0,0,0],
                               [0,0,J3]],
                        mass=m3
                        ])
            ]);

edge[10]:=table([(1)=N,
             (2)=[7,1],
             (3)=FDE,                # Force driver element
             (4)=table([type=JD,     # joint driver
                       theta=(0,0,T[1](t))
                       ])            # theta specifies torque
            ]);
```

```
edge[11]:=table([(1)=M,
                 (2)=[2,3],
                 (3)=FDE,
                 (4)=table([type=JD,
                            theta=(0,0,T[2](t))
                             ])            # torque about local z-axis

                ]);
Iedge:=[];    # List of independant joint edges used if there
              # are more constraints than joint coordinates
              # Thus system VW is expressed in terms of Iedges
              # Null for open loop systems
```

Reading the inputfile into MAPLE, we can execute the DYNAFLEX procedures which provide the following equations of motion in symbolic form.

$$[M]\{\ddot{q}\} = \{\Im(q, \dot{q}, t)\} \tag{2.6}$$

where,

$$\{q\} = \begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} \tag{2.7}$$

with,

$$[M] = \begin{bmatrix} \begin{Bmatrix} (m_2+m_3)l_1^2 + (m_2+2m_3)\cos(q_2)l_2l_1 + (\frac{1}{4}m_2+m_3)l_2^2 + J_1 + J_2 + J_3 \\ (\frac{1}{2}m_2+m_3)\cos(q_2)l_2l_1 + (\frac{1}{4}m_2+m_3)l_2^2 + J_2 + J_3 \end{Bmatrix}^T \\ \begin{Bmatrix} (\frac{1}{2}m_2+m_3)\cos(q_2)l_2l_1 + (\frac{1}{4}m_2+m_3)l_2^2 + J_2 + J_3 \\ (\frac{1}{4}m_2+m_3)l_2^2 + J_2 + J_3 \end{Bmatrix}^T \end{bmatrix} \tag{2.8}$$

and

$$\{\Im(q, \dot{q}, t)\} = \begin{Bmatrix} T_1 + (\frac{1}{2}m_2+m_3)l_2l_1\sin(q_2)(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) \\ T_2 - (\frac{1}{2}m_2+m_3)l_2l_1\sin(q_2)(\dot{q}_1^2) \end{Bmatrix} \tag{2.9}$$

These results are after the substitutions of $r_1 = l_1$ and $r_2 + r_3 = l_2$ where made. Note there are no Lagrange multipliers for the joint reaction forces and no constraint equations using this formulation. The result are two second order ODEs that can readily be solved via numerical integration methods.

2.3.4 Scalable Framework

The previous exercise is an excellent example of the scaling capabilities of the graph representation, which is capable of describing varying system complexities. Specifically we began with the biomechanical model of the upper-limb and its graph (Figure 2.5) but it is easy to recognize that the planar 2-link manipulator graph (Figure 2.7) is in fact its subgraph. In other words, by removing the muscle edges and their associated arm elements (identifying the origin and insertion points of the muscles) and adding two torque drivers we obtain the identical graphs. Specifying revolute joints further simplified the model to a planar one with no change to the graph. This is a powerful feature which can be exploited and used in reverse to quickly evolve biomechanical models beginning with simpler models and moving towards more complex systems by using different terminal equations. For example, by defining the base shoulder joint in either graph to be a spherical joint we now have a four degree of freedom spatial model from the planar model without a single change to the graph. The difficulties in formulation are relegated to the computer, thus freeing the researcher to focus on greater issues and virtually eliminating formulation errors for very complex systems.

2.3.5 Adding Components by Introducing New Terminal Equations

Adding new components such as more anatomically accurate joints or muscles and other connective tissue can be done by adding to DynaFlex's list of terminal equations. DynaFlex requires terminal equations to be described in terms of the virtual work and virtual displacement of the edge. DynaFlex automatically assigns dependant virtual work elements as necessary [70].

2.4 Numerical Solution of Biomechanical System Equations

Having derived the symbolic equations of motion using the GTM embodied in DynaFlex, the final step in the modelling process is solving the equations and generating simulations of motion. Here we have complete freedom over which ODE solver to use since the Maple equations can be exported as optimized C or Fortran code to make use of available numerical methods. For convenience and rapid prototyping we have exported the equations to Matlab for numerical integration using `ode45` which is a 4^{th} order (5^{th} order error approximation) Runge-Kutta method

[61]. Most numerical ODE solvers ([64], [46]) and all the solvers in MATLAB can only solve a system of first order ODEs. This is a small limitation since it is a simple transformation to convert the second order equations of motion to first order equations. Given the dynamic equations (2.6) we can rewrite them in first order state space form by

$$\{\dot{x}\} = \left\{ \begin{array}{c} \{\dot{q}\} \\ [M]^{-1}\{\Im(q,\dot{q},t)\} \end{array} \right\} \tag{2.10}$$

where $\{x\} = \left\{ \begin{array}{c} \{q\} \\ \{\dot{q}\} \end{array} \right\}$ contains the generalized coordinates and their velocities. A more detailed discussion of the numerical implementation is discussed in Chapter 4.

Chapter 3

Optimal Control Determination

3.1 Optimization of Dynamic Systems

In either the study of human biomechanics or robotics, the concept of optimality plays a major role in performance. In robotics, the concept of optimal control, especially in manipulator task and path planning [25], [37], [31], has resulted in more efficient, time-saving and cost-saving use of robot manipulators. In human biomechanics, muscle force decomposition methods from net joint moments, determined by inverse dynamics (Section 1.2.2), have traditionally used optimization techniques to determine the distribution of redundant muscle forces ([60], [17], [62], [28], etc...). For this application, a static optimization is performed at each point in time to decompose the net joint moment and thus are generally known as quasi-static optimization since the dynamics of the problem are independently captured by the inverse dynamics model.

Yoshimura et al [91], apply a similar optimization procedure, except the problem is cast to determine both the muscular forces and the speed along a desired path such that total muscle energy is minimized in attempts to reproduce human-like behavior. The desired joint trajectory is defined for the task. However, the angular velocity trajectory is assumed to be smooth and bell-shaped such that it can be parameterized by a single value, q, the point of maximum angular velocity of the trajectory where the time period is normalized (Figure 3.1). To solve for the optimal muscle force distribution and velocity profile, the moment and the decomposition of the muscle forces are computed for a fixed value of q, which is then varied as a single design variable of

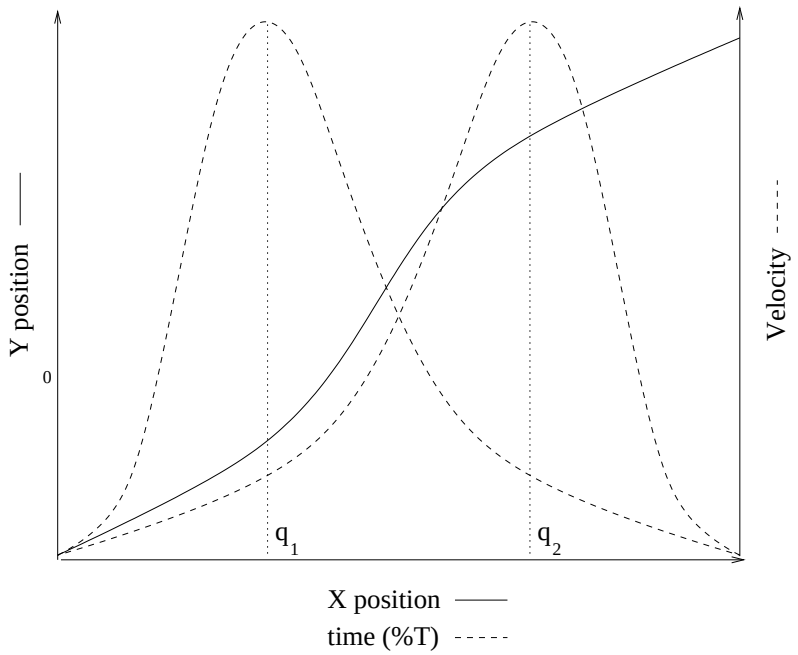

Figure 3.1: Example of a single paramter q, for velocity profile parametrization

another static optimization procedure. This approach is unique in that it introduces the concept of trajectory parameterization. However, this simplified representation cannot be applied for predicting the behavior of large scale (many degrees of freedom) systems because it is precisely the joint kinematics which we wish to predict and is defeated by prescribing joint displacements. In addition, restricting the shape of the velocity profile also constrains the acceleration profile and thus artificially constrains the dynamics of the problem such that the result is specific for the assumed shape. The formulation of the control problem must include the system's dynamics as explicit constraints along with a suitable objective for performance, and the optimization must be performed dynamically across the the performance period and not by independent point evaluations if the true optimum is to be discovered.

More recent applications of optimization in biomechanical systems have been used in a more dynamic sense to answer questions ranging from, "how high we can jump?" (Pandy [58]), to "what is the arrangement of neural feedback circuits in the central nervous system that enables optimal stability?" (Loeb and Levine [45]), to "how to program functional neuromuscular stimulators (FNS) to restore an individual's gait mobility?" (Yamaguchi [90]). Here, controls are sought to

bring about a desired behavior that is optimal according the specific requirements of the task (e.g. maximum height of the centre of mass for jumping). Many methods are used to perform the optimizations, including linear-quadratic programming, grid search, sequential-quadratic programming, penalty methods, etc. The selection of the method is dependent on the approach, the objectives and structure of the optimization problem.

3.1.1 Static vs. Dynamic Optimization

Methods of optimization vary largely with the particular classification of the optimization problem. Two classes of optimization problems are static or parameter optimization and dynamic optimization. In the static case, the design variables are system parameters that are fixed during the evaluation of the objective function. An example of static optimization would be finding the geometrical, stiffness, and damping parameters of an automobile suspension that minimized a performance index such as bounce or driver acceleration [12]. The static classification relates to the nature of the finite parameters and not the dynamic or static nature of the problem. Dynamic optimization for a similar problem would be to determine the forcing functions of system actuators which would result in the minimization of oscillations in the suspension system. The design variables in the dynamic case are functions of time and thus the basis for the space of design variables is infinite.

3.2 The Optimal Control Problem

In the dynamic optimization example, the actuator forcing functions can be thought of as the active control of the system to perform optimally. Determining the control functions is known as an optimal control problem. For predicting the motion of an upper limb biomechanical model it is precisely the control of muscle forces that is required once a dynamic model has been formulated.

Having previously described how to generate the equations of motion (Chapter 2) with either joint torques or muscular actuators as the controlling inputs, we formulate the optimal control problem here before discussing the relevant optimization methods and the specific approach chosen for our research.

Most optimal control problems begin with the description of the scalar objection function that

maps a series of system performance criteria to a single performance value. Unfortunately the specific form of the objective in our study is unknown and is dependent on the criteria we wish to include, such as performance time, joint reaction forces, muscle stress, etc. As a result, a method which takes advantage of the structure of the objective function cannot be applied directly and more general solution methods are required. The general form of the optimal control problem as described by Agrawal and Fabien [2] is presented below.

The objective function can be expressed as

$$J = \Phi(t, \{x\})|_{t_f} + \int_{t_0}^{t_f} F(t, \{x\}, \{u\}, \{p\})dt \tag{3.1}$$

where $\{x\}$ is the vector of system state variables, $\{u\}$ is the vector of control inputs and $\{p\}$ is a vector of system parameters. J is the total performance value to be optimized and Φ is a performance measure of the final system state which can be any scalar function of the final state kinematics. F is a functional that yields an increment value in performance, such that the integral from the start time to the final time yields some total measure for the criteria included in the functional. We are avoiding using the word "cost" and terms like "cost functional " or "running costs" to avoid the inference of minimizing the objective function. It is simply a matter of convenience or by an analogy that a problem is cast as a maximization or minimization for a particular optimization method, Section 3.6.

The objective function J is subject to equality constraints defined by the system state equations $\{f\}$ from the system equations of motion

$$\{\dot{x}\} = \{f(t, \{x\}, \{u\}, \{p\})\} \tag{3.2}$$

and the prescribed initial state $\{x_0\}$ and/or final state $\{x_f\}$

$$\begin{aligned} \{x(t_0)\} &= \{x_0\} \\ \{x(t_f)\} &= \{x_f\} \end{aligned} \tag{3.3}$$

Although the objective functional F will vary in our study, the modelled tasks will be defined by fixed starting and target states. As a result, any function of the final state, Φ, will only contribute a constant to the objective and will not affect the optimal control problem. However, as will be seen in the discussion of direct methods, we can use the final state performance to "free" the final state constraint and to compute the solution to an approximate problem [37].

From the state equations it is evident that the controls $\{u\}$ "lead" the system state and thus the resulting optimization is to determine $\{u\}$ and the additional parameters $\{p\}$ which optimize the total performance measure J. Furthermore, the system is subject to inequality constraints

$$\{C(t, \{x\}, \{u\})\} \leq 0 \tag{3.4}$$

where $\{C\}$ can be any set of nonlinear functions of the state and control, and

$$\{|u|\} \leq \{U(t, \{x\})\} \tag{3.5}$$
$$-(U_i - u_i)(U_i + u_i) \leq 0$$

where $\{U\}$ is a general expression for the maximum magnitudes of the system controls, which can be functions of the system state. Equation 3.5 is rewritten to eliminate taking the magnitude of the elements in $\{u\}$ without doubling the number of constraints.

The precise optimal control problems will be described for a specific upper-limb model in our discussion of the implementation and results, Chapter 4.

3.3 Solution Methods for Dynamic Optimization

General solution methods for dynamic optimization, as is the case for the optimal control problem, fall into two categories. Of course the most elegant of solution methods would be analytical methods which apply the calculus of variations [42] to determine the closed-form functional expressions for the control inputs. Realistically these cannot be applied directly to most problems since the nonlinear differential- algebraic equations (DAEs) of motion have no closed form solution. Indirect methods apply numerical approximations to satisfy the analytical conditions for optimality which define boundary value problems (BVP) involving DAEs (BVP-DAEs), in the general case. The various methods used to solve the BVP-DAEs comprise the indirect solution methods to the optimal control problem. The direct methods are the "brute" force methods that discretize the control inputs which are then systematically propagated to fully parameterize the problem. From this point a static optimization method can be applied to solve the problem numerically.

The results from analytical methods are the foundations, to some degree, for the numerical methods to follow. Here, we briefly highlight the conditions and consequences of optimality, which provide sufficient information to determine the optimal controls [2].

3.3.1 Necessary Extremum Conditions

First, we recognize for our class of problems that the objective is dependent on the functional

$$\int_{t_0}^{t_f} F\{t, \{x\}, \{u\}, \{p\}\}dt \tag{3.6}$$

Assuming F is twice differentiable with respect to its arguments, the state, controls and parameters, $\{z\} = \{\{x\}, \{u\}, \{p\}\}^T$, it is desired to find $\{z\}$ that extremizes the objective (3.1). A change in the objective is defined as $\Delta J = J[\{z\} + \{h\}] - J[z]$, where $\{h\} = \{\{h\}_x, \{h\}_u, \{h\}_p\}^T$ is any admissible increment in $\{z\}$ that satisfies the functional's boundary conditions. Expanding the expression ΔJ in a Taylor series approximation and taking the first term, δJ,such that $\delta J \approx \Delta J$ and integrating appropriately, it can be shown [26], for the first-order necessary optimality condition $\delta J = 0$, that

$$\frac{\partial F}{\partial \{z\}} - \frac{d}{dt}\frac{\partial F}{\partial \{\dot{z}\}} = 0 \tag{3.7}$$

must be satisfied. This equation, known as Euler's equation, is the primary condition for an extremum for the appropriate state and control variables. With additional end point conditions being

$$\left(\frac{\partial L}{\partial \{\dot{z}\}}\right)|_{t_f} = 0 \quad and \quad \left(\frac{\partial L}{\partial \{\dot{z}\}}\right)|_{t_0} = 0 \tag{3.8}$$

which are trivial ($0 = 0$) if the end states are prescribed. Similar results can be derived for variable end times where an additional variation is taken in time such that the integration limits of (3.6) become $t_0 + \delta t_0$ and $t_f + \delta t_f$ and the above derivation is repeated resulting in additional boundary conditions

$$\left[F - \{\dot{z}\}^T \left(\frac{\partial F}{\partial \{\dot{z}\}}\right)\right]_{t_f} = 0 \quad and \quad \left[F - \{\dot{z}\}^T \left(\frac{\partial F}{\partial \{\dot{z}\}}\right)\right]_{t_0} = 0 \tag{3.9}$$

such that (3.7), (3.8) and (3.9) together provide the necessary conditions for an extremum.

In the control of multi-body systems there are alway equality constraints (3.2) which are the equations that govern the motion of the system. Here the Lagrange multiplier method is used to append the dynamic constraints to the objective such that

$$J' = \Phi(t, \{x\})|_{t_f} + \int_{t_0}^{t_f} F\{t, \{x\}, \{u\}, \{p\}) + \{\lambda\}^T\{f'(t, \{x\}, \{u\}, \{p\})\}dt \tag{3.10}$$

where the the Lagrange multipliers $\{\lambda\}$ are functions of time and the constraint

$$\{f'(t,\{x\},\{u\},\{p\})\} = \{f(t,\{x\},\{u\},\{p\}) - \{\dot{x}\}\} = 0 \tag{3.11}$$

is identical to (3.2). Now we can replace the functional F, in the previous results for the necessary conditions, with the augmented functional $H = F + \{\lambda\}^T\{f'\}$, which is also known as the Hamiltonian of the system. Generalizing further, auxiliary equality and/or inequality constraints may be applied in a similar manner. For example with additional kinematic constraints $\{g(t,\{x\})\} = 0$ and control bounds (3.5),

$$H = F + \{\lambda\}^T\{f'\} + \{\mu\}^T\{g\} + \{\nu\}^T\{\{C\} + \{s^2\}\} \tag{3.12}$$

is the resulting general Hamiltonian where $\{\mu\}$and $\{\nu\}$ are Lagrange multipliers as functions of time that correspond to the equality constraints $\{g\}$ and inequality constraints $\{C\}$ (from (3.4)), respectively. Note that this represents the general form of the Hamiltonian for problems addressed in this study. The inequality constraints are included in the augmented functional H by the addition of slack variables $\{s\}$ which convert the inequality constraints (3.4) into equivalent equality constraints by

$$\{C(t,\{x\},\{u\})\} + \{s^2\} = 0 \tag{3.13}$$

which can be included, once again, by the Lagrange multiplier method. The square of the slack variables ensures $\{C\} \leq 0$. The new slack variables are considered as new control variables. Finally, using the general Hamiltonian (3.12) as the objective functional, with $\{z\} = \{\{x\},\{\lambda\},\{u\},\{p\},\{\mu\},\{s\},\{\nu\}\}^T$, then Euler's equation (3.7) yields

$$\{\dot{x}\} = H^T_{\{\lambda\}} \tag{3.14}$$

$$\{\dot{\lambda}\} = -H^T_{\{x\}} \tag{3.15}$$

$$0 = H^T_{\{u\}} \tag{3.16}$$

$$0 = H^T_{\{p\}} \tag{3.17}$$

$$0 = H^T_{\{\mu\}} \tag{3.18}$$

$$0 = H^T_{\{s\}} \tag{3.19}$$

$$0 = H^T_{\{\nu\}} \tag{3.20}$$

as extremum conditions. The boundary conditions for variable final time and final states from (3.9) yields

$$\{\lambda\}|_{t_f} = \Phi^T_{\{x\}}|_{t_f} \tag{3.21}$$

$$0 = \left[\Phi_t + \{\mu\}^T\{g\}_t + \{\nu\}^T\{C\}_t + \{\lambda\}^T\{f\}_t + F\right]_{t_f} \tag{3.22}$$

otherwise the state boundary values ($\{x(t_0)\} = \{x_0\}$ and $\{x(t_f)\} = \{x_f\}$) provide the complete set of necessary conditions for an extremum for the general optimal control problem in this study. These conditions include $2n$ differential equations for the state (3.14) and co-state (3.15) conditions, m ordinary equations of additional extremum control conditions (3.16), l equality (3.18) and $2r$ inequality (3.19-3.20) constraint conditions, plus one final time boundary equation (3.22). The initial and final conditions provide the boundary values for the differential equations. It is interesting to note that the optimal co-state (adjoint) constraints correspond to Pontryagin's minimum (maximum) principle [56] for optimal control, which was derived using a similar procedure. The principle is based on the fact that the velocity of the optimal trajectory $\{\dot{x}\}$ in augmented state space (including the "running" cost as the state x_0) should be perpendicular (dot product is zero) to the upward normal (towards increasing cost) of the surface formed by all feasible trajectories in order to minimize the cost. In this sense the dynamic equations define the surface of all feasible trajectories while the co-states $\{\lambda\}$ from the adjoint (adjoined constraints) equations constrain the augmented state space velocity to remain perpendicular to the surface upward normal [84].

3.3.2 Sufficient Optimality Conditions

The first order variation of the objective functional F (3.6) was used to find the necessary conditions of the control that extremized J. To determine if the control is in fact optimal (maximum or minimum) we look at the second term of the Taylor series expansion, $\delta^2 J$, which dominates the change ΔJ as an extremum is approached ($\delta J \to 0$). Thus the sufficient condition for a minimum is that $\delta^2 J$ is positive definite, or is negative definite for a maximum. For a system with a Hamiltonian $H = F + \{\lambda\}^T\{f'\}$ and state $\{x\}$ and controls $\{u\}$ this has two consequences for a minimum. First that the Legendre condition

$$\frac{\partial^2 H}{\partial \{u\}^2} > 0 \tag{3.23}$$

must be satisfied. Secondly, the solution for S in

$$\dot{S} + H_{xx} + S\{f'\}_x + \left(\{f'\}_x\right)^T S = \left[H_{xu} + S\{f'\}_u\right] H_{uu}^{-1}\left[H_{xu}^T + \{f'\}_u^T S\right] \tag{3.24}$$

where $S(t_f) = 0$ for $\Phi = 0$, must be finite on the interval $t_0 \leq t \leq t_f$. Together they are sufficient conditions for $\{u(t)\}$ to be the optimal control that minimizes J. The necessary finite solution to this Riccati equation is known as the Jacobi condition and is a result of the fact that the state and control in the evaluation of $\delta^2 J$ are not independent and are governed by the differential equations of motion. A detailed derivation of these two conditions is presented in [24] and [2].

These conditions are presented for completeness, however in practice the Jacobi condition cannot be readily verified without numerical resolution since it requires the solution of a set of non-linear differential equations.

3.3.3 Link to Analytical Mechanics

We take the opportunity at this time to consider the functional H, the Hamiltonian of the system, which is commonly used in mechanics. The Hamiltonian in analytical mechanics represents the generalized system momentum and thus the integration of the the functional in time is a scalar measure of the system energy. In analytical mechanics this is used to determine the equations of motion by considering that no work is done to or by the external world. Thus, the change in the total system energy is zero and the same derivation that resulted in the Euler's equation produces Lagrange's equation where the variation is with respect to the system generalized coordinates. This provides the differential equations of motion such that no virtual work is done by the system and thus the trajectories of the generalized coordinates are an extremized path in the configuration space of the mechanism. The resulting equations of motion are in fact the conditions on the trajectories, in terms of the generalized coordinates, in the configuration space of the mechanism [42].

In the optimal control problem the analogy is that the objective function represents the system energy and we desire the generalized coordinates, now the states and controls, that extremizes the energy in the motion space of the system.

Pontryagin's minimum principle links both analytical mechanics with optimal control of a mechanical system for special classes of problems, mainly the minimum time problems. For these problems, time is considered as a generalized coordinate and an appropriate "momentum" term for the time coordinate is assigned to the system Hamiltonian such that the path is extremized for the configuration through time which is the resulting motion space. The results are the

differential co-state and optimality conditions derived above (Section 3.3.1) defining a boundary value problem [84].

For the minimum time problem (fixed end points, $\Phi = 0$, and $F = 1$ in (3.1)), it is generally known *apriori* that the controls will be max/min or "bang-bang" ([25], [21]). In this case, the co-state and state conditions of the two point boundary value problem (TPBVP) can be used directly to determine the switching times for the controls. In the general case, however, the structure of the control is not known *apriori*.

3.4 Indirect Numerical Solutions

The resulting conditions (3.14-3.22) for an extremum, in the general case form a set of BVP-DAEs with nonlinear DAEs. These equations are only resolvable by numerical methods with the exception of a few text book cases used to illustrate analytical methods. Indirect methods are the resulting set of numerical methods used to find the state, co-state and control variables which satisfy the boundary conditions.

For the optimal control problems under investigation in this study, the end point states are known and the class of problems are known as two-point boundary value problems (TPBVPs) which have several known solution techniques. These include finite difference methods, collocation methods, and initial value or shooting methods [2]. The most robust of these methods however, is the multiple shooting method [80].

3.4.1 The Shooting Method

The simple shooting method begins with an initial guess of the state and controls that attempts to satisfy both the state and co-state equations by integrating both the state and co-state conditions from the initial conditions. At the end of the integration an error is computed by taking the distance from the end point of the integrated trajectory to the desired final condition. An adjustment to the guess is made based on the error. The adjusted guess is then used and this is done iteratively until no more improvement can be made or the final condition is satisfied.

This method is highly sensitive to the initial guess, especially if the guess results in unstable behavior, wherein the method simply cannot converge. Second, if the solution method finds

a suboptimal solution where the final conditions can be satisfied, it has difficulties making further improvements that require leaving the suboptimal solution. These methods have very poor convergence for highly nonlinear dynamics, especially where the structure of the controls are unknown.

3.4.2 Multiple Shooting Method

The multiple shooting method subdivides the TPBVP into a series of M initial value problems where the initial points (nodes) of the intermediate problems are initially **not** the end points of the previous integration but are intermediate guesses. The initial point for the first segment is obviously the prescribed initial state. Integrating, from each starting point, an error state vector is calculated taking the distance from the termination of the previous segment with the starting point of the following, where the last segment's error is with respect to the final condition. The integration is done numerically using, for example, a high-order implicit Runge-Kutta method [71]. The resulting error thus measures both continuity as well as the the two point boundary condition. The variation of the error state vector with respect to the initial and intermediate guessed solutions (the state and controls) results in a Jacobian matrix which is then used to compute a change in the initial guesses for the following iteration. This is done repeatedly until continuity and boundary conditions are satisfied within a desired tolerance or there is no more change in the solution.

The multiple shooting method is definitely better suited for nonlinear problems with better convergence characteristics than the simple shooting method. This is because the solution is based on multiple approximations that have varying influence according to the error contribution of each node. The subsequent computation using the Jacobian causes nodes with smaller error to change solutions more slowly compared to nodes with larger errors and thus the method is less sensitive to the initial guesses when compared to just a single guess. With sufficiently good initial guesses, solutions found with this method are very accurate.

In essence the multiple shooting method is the solution to the nonlinear state, co-state, optimality and bounds from the first order optimality conditions at M discrete times within a domain of interest. The fundamental problem is the solution of $(2n + m + p)(M + 1)$ system of nonlinear equations, from the state and co-state equations $(2n)$, control (m) and auxiliary constraints

(p). This requires a robust root-finding technique using an iterative approach such as Newton's method. For Newton's method, for example, an initial solution in a region close to the root is required for convergence, thus the indirect method requires good starting guesses for the states and co-states as well as the controls in order to converge.

Furthermore if we consider a biomechanical system with four degrees of freedom — that is at least eight state variables — and the system is over-actuated (has redundant muscles) with ten muscles and bounds on their maximum output, then the result is a set of 16 state plus co-state equations, 10 control equations, and 20 constraint equations (introducing slack variables), for a total of 46 equations. If we use any reasonable number of intermediate nodes, we need to solve a system of about 500 or more nonlinear equations! With over-actuated systems there is also an increased risk of the Jacobian matrix becoming singular because the state trajectories and subsequent errors are not independent in terms of the controls (muscle forces).

3.4.3 Higher-Order Methods

Recently, Agrawal et al [1] have shown that for open-loop dynamical systems, the substitution of the controls, from the equations of motion, in terms of the n generalized coordinates directly into the objective functional eliminates the Lagrange multipliers (no dynamics constraints) as well as the control variables (m controls), resulting in a BVP with n fourth-order ODEs. This is a significant simplification with many computational advantages, except for solving the fourth-order differential equations. Reducing the order of the system as in Section 2.4 to use the multiple shooting approach, i.e. $4n$ first order-equations, is not straight forward nor is there a significant reduction in the number of equations in comparison to using the Lagrange multiplier method.

Simpler collocation methods [10] have been used successfully to solve this class of problems. These methods approximate the trajectory of the (n) generalized coordinates by the sum of k mode functions, which are polynomials that satisfy the boundary conditions. The weightings of the mode functions are determined for k instances in time for the n generalized coordinates such that they satisfy the fourth order ODEs. The result is a system of nk nonlinear equations that are solved using an iterative technique. The number of modes k, can be increased via a recursive method to generate solutions of desired precision.

Although the general form of the equations of motion in this study are second-order ODEs

(2.4), we cannot explicitly eliminate the controls by direct substitution of the state variables in order to reduce the problem to fewer higher-order BVP-DEs, because in the general case we will have redundant muscles to control. Thus using, the aforementioned reduction technique to eliminate the n Lagrange multipliers and the m control variables from the initial objective functional is not always feasible.

3.5 Direct Numerical Methods

The direct methods are more straight forward methods of determining the optimal controls of dynamic systems. The technique is similar to indirect methods in that an approximation to the solution is made and that approximation is refined iteratively until it converges to a solution. The big difference is that the entire control and state trajectories are parameterized via piecewise polynomials or other discretizing approximations. This approximation is included in the objective functional as well as the dynamic constraints and boundary conditions. The result is a large static optimization problem with the control and, if desired, additional system parameters as the design variables. The direct approach requires the solution to an approximate problem where the final conditions are "free" with an objective measure Φ' that penalizes results that do not meet the actual problems' final state constraints.

3.5.1 Control Parameterization

First we consider a mapping between the continuous controls at discrete points by discretizing the time span $t \in [t_0, t_f]$ such that $t_i = i\Delta t$, where $\Delta t = t_f/(N-1)$ and $i = 0, 1, ..., N-1$. Thus the approximation to $\{u(t)\}$ is $\{\bar{u}(t_i)\}$ which is defined only at the discrete points in time t_i. To determine a continuous approximation suitable for integrating the state equations and objective functional in order to evaluate the objective J', the discrete controls must be interpolated. Many interpolation schemes can be employed such that $\{\tilde{u}(t, t_i, t_{i+1}\} \approx \{u(t)\}$ provides approximate and continuous control values for intermediate time points $t_i \leq t < t_{i+1}$. A simple linear interpolation is popular in the literature and appears to be effective, [84], [37], and [2]. A simple linear approximation is

$$\{u(t)\} \approx \{\tilde{u}(t, t_i, t_{i+1}\} = \{\bar{u}_i\} + \frac{\{\bar{u}_{i+1}\} - \{\bar{u}_i\}}{\Delta t}(t - t_i) \tag{3.25}$$

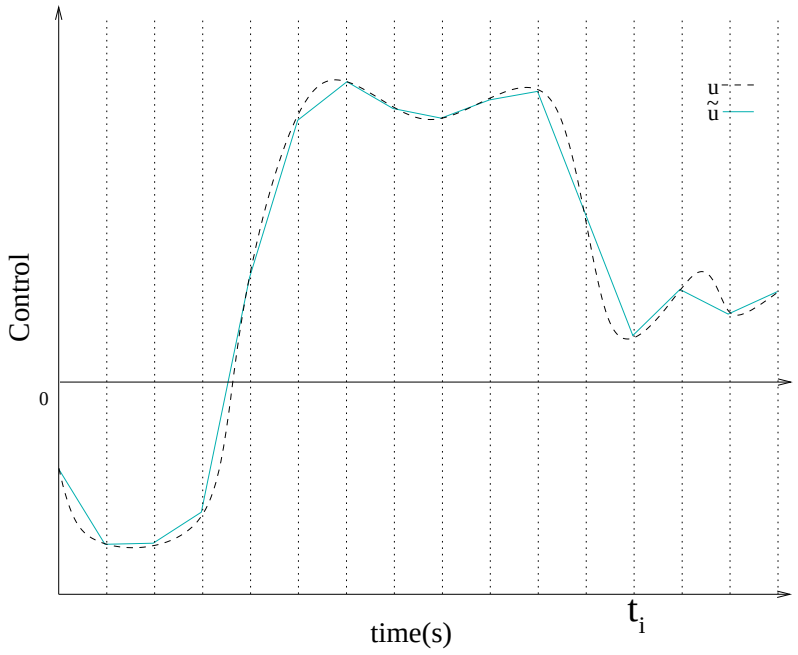

Figure 3.2: Linear interpolation of a prametrized control signal

which generates an approximate control, as shown in Figure 3.2. Additional time-varying system parameters can be appended to the set of controls in exactly the same manner.

3.5.2 Augmenting the Optimal Control Problem

In order to solve the problem using direct solution methods we must convert the original TBVP-DAE problem to an initial value problem. This is done by eliminating the final state constraints and including them via a penalty term in a new measure of the final state performance Φ such that

$$\Phi' = \Phi + \{S\}^T [\tilde{r}] \{S\} \tag{3.26}$$

where $[\tilde{r}]$ is a k dimensional diagonal matrix of weights and $\{S\}$ is a k dimensional vector of the final state violations so that when the constraints are met, $\Phi' = \Phi$.

This now allows us to evaluate the objective functional by integrating the equations of motion forward in time from the initial conditions for any given controls $\{\bar{u}\}$ and parameters $\{p\}$.

3.5.3 Determining Functional Gradients

A large part of static optimization methods (Section 3.6) is determining the gradient of the objective function with respect to the design variables. This can be done numerically but, since one evaluation of the objective function requires a simulation, it is computationally prohibitive to compute the gradients for a series of objective function evaluations.

To determine the gradients with respect to the controls and parameters we consider the objective function J (3.1) which is dependent on a functional of the state trajectory $\{x\}$, control $\{u\}$ and additional state parameters $\{p\}$. Given an initial guess of the control $\{\tilde{u}\}$, we want to modify the control as to get a better control $\{\tilde{u}'\}$. Taking $\{\tilde{u}'\} = \{\tilde{u}\} + \{h_u\}$, where $\{h_u\}$ and $\{h_p\}$ are increments in the controls and parameters guess and $\{h_x\}$ is the subsequent state increment, then we can get an expression for the change in objective

$$\Delta J \approx \delta J = \int_{t_0}^{t_f} (H_x + \{\dot{\lambda}\})^T \{h_x\} + H_u^T \{h_u\} + H_p^T \{h_p\}) dt \tag{3.27}$$

where H is the system Hamiltonian not including any auxiliary constraints. We can choose the co-states $\{\dot{\lambda}\} = -H_x$, from the optimality condition (3.15), and then integrate the remaining terms using the trapezoidal rule for the discretized time span to find

$$\begin{aligned} \Delta J \approx & \textstyle\sum_{i=0}^{N-2} \frac{1}{2} (H_u^T(t_{i+1})\{h_u(t_{i+1})\} + H_u^T(t_i)\{h_u(t_i)\})\Delta t \\ & + \textstyle\sum_{i=0}^{N-2} \frac{1}{2} (H_p^T(t_{i+1}) + H_p^T(t_i))\{h_p\}\Delta t \end{aligned} \tag{3.28}$$

$$\Delta J \approx \sum_{i=0}^{N-1} \left(\frac{\partial J}{\partial \{\bar{u}(t_i)\}} \right)^T \{h_u(t_i)\} + \left(\frac{\partial J}{\partial \{p\}} \right)^T \{h_p\} \tag{3.29}$$

It is evident from (3.28) and (3.29) that the gradient of the objective with respect to the discrete controls and additional parameters are

$$\begin{aligned} \tfrac{\partial J}{\partial \{\bar{u}(t_i)\}} &= \tfrac{1}{2} H_u(t_i)\Delta t, \; i = 0, \, N-1 \\ \tfrac{\partial J}{\partial \{\bar{u}(t_i)\}} &= H_u(t_i)\Delta t, \; i = 1, 2, \ldots N-2 \end{aligned} \tag{3.30}$$

$$\frac{\partial J}{\partial \{p\}} = \sum_{i=0}^{N-2} \frac{1}{2} (H_p^T(t_{i+1}) + H_p^T(t_i))\Delta t \tag{3.31}$$

We can rewrite both the controls and system parameters as one set of design variables $\{y\} = \left\{ \{\bar{u}(t_0)\}^T, \{\bar{u}(t_1)\}^T, ..., \{\bar{u}(t_{N-1})\}^T, \{p\}^T \right\}^T$ and the resulting objective function gradient

$$\nabla J = \left\{ \left\{ \frac{\partial J}{\partial \{\bar{u}(t_0)\}} \right\}^T, \left\{ \frac{\partial J}{\partial \{\bar{u}(t_1)\}} \right\}^T, ..., \left\{ \frac{\partial J}{\partial \{\bar{u}(t_{N-1})\}} \right\}^T, \left\{ \frac{\partial J}{\partial \{p\}} \right\}^T \right\}^T \tag{3.32}$$

can be used to determine the subsequent increment

$$\{h\} = \left\{\{h_u(t_0)\}^T, \{h_u(t_1)^T, ...\{h_u(t_{N-1})^T, \{h_p\}^T\right\}^T$$

of the design variables towards the optimal solution. Thus the evaluation of the gradient with respect to the controls requires the integration of the costate equations (3.15) backward in time knowing (3.21) from $\Phi I_{\{x\}}|_{t_f}$.

Depending on the static optimization method used to solve for the discrete time controls and system parameters, constraints can be appended using the Lagrange multiplier method and/or penalty methods. However, in Section 3.6, we will see that some optimization methods can include constraints implicitly.

Direct methods have come to the forefront of optimal control solution methods for several reasons. First, closed-form analytical solutions for most practical problems are impossible. Next, indirect methods are difficult to solve requiring the application of many numerical techniques that have to be specific to the structure of the objective functional and especially to how the controls appear (i.e. linear, independant, etc.) in the dynamic equations. These methods also require good initial guesses to converge, which for complex systems can be equally as difficult to determine as solving the optimal control problem itself. Finally, the increase in the average computational speed in recent years have enabled more exhaustive parameter optimization methods to be exploited.

Direct methods are very robust at finding solutions, and most methods guarantee at least a feasible suboptimal solution. This is a result of the approximations to the control which can in some instances make it impossible for the optimization to converge to the optimal solution. For example, an optimal switch in the control may occur within the span of Δt, but the approximation to the closest point in time, t_i, is insufficient for an optimal time "bang-bang" control, so instead a suboptimal performance with more gradual control transitions are produced. In these instances only a fine mesh can find the optimal control. Unfortunately, a finer mesh means more design variables and a larger search space which results in a greater probability of local minima. Thus even though precision can be increased, it does not necessarily mean an increase in accuracy for the higher computational cost.

For this reason researchers have suggested using direct methods to generate initial guesses for indirect methods. For certain applications such as robotics this is a feasible strategy. Biomechanical systems, however, are generally more non-linear and have redundant actuators which creates

many instabilities for the indirect methods. As was mentioned earlier, the nonlinear solver for the indirect method must be given an initial starting point that lies sufficiently close to the root of the system; otherwise it may still not converge even though it is feasible and sub-optimal. In the next section we discuss optimization methods in order to solve optimal control problems using the direct method, and compare their abilities to find the global solution.

3.6 Numerical Optimization Methods

It should be apparent, from the aforementioned discussion of solution methods to optimal control problems (Section 3.2), that we have cautiously arrived to the direct approach as the method of choice. This is mainly because the direct approach provides the most general framework for solving optimal control problems without having the requirements of assuming the structure of the control or setting restrictions on how the system is actuated (i.e. over or proper-actuated). The concerns are primarily with the accuracy of the solution method — will it converge to the optimal solution? — and second is the tractability of the computation. If the method fails to provide the optimal solution reliably or is unable to produce results in the order of hours the utility of this work will be questionable.

Optimization methods differ only by the means by which they traverse the objective surface to find better solutions and then, hopefully, the optimal solution. The most common approaches rely on the gradient of the objective function with respect to the design variables to approach an extremum solution quickly and with high precision. However, variable performance criteria with variable weightings and over-actuated dynamic models with many DOFs can dramatically change the objective landscape and, furthermore, considering the large number of design variables from the control parameterization, it soon becomes apparent that local optima are inevitable and gradient information is insufficient.[1] Being able to find the the global optimum amongst many

[1] As an important aside, we might consider using the knowledge at hand, i.e. use the data collected from test subjects to "coordinate" the initial trajectory and make assumptions about the controls based on an inverse dynamics analysis. From our point of view this defeats the purpose of using optimal control methods. Could we say that the human control system optimizes a particular objective function if we used *a priori* information to initialize the results that recreated the desired motions? Even if they were possibly only representative of local extrema? Of course not! Perhaps this is prescisely why muscle force decomposition strategies in inverse dynamics are insensitive to the objective used. If our only desire is to recreate human-like motion then we need not go through the rigors of modelling and constructing the optimal control problem.

local extrema without *a priori* knowledge of the optimal solution is a significant challenge.

First we consider the general optimization problem beginning with the scalar function J

$$J = f(\{y\}) \tag{3.33}$$

where $\{y\} = \{y_1, y_2, ..., y_n\}^T$ is a vector of dependent variables. The goal of any optimization is to find $\{y\}^*$that extremizes the scalar J. Given a starting point $\{y\}_i$, we want to incrementally find $\{y\}_{i+1}$ where $f(\{y\}_{i+1}) < f(\{y\}_i)$ for a minimum (reversed for a maximum) and continue iteratively such that $\lim_{k\to\infty}\{y\}_k = \{y\}^*$. We briefly examine some classes of iterative methods that lead to solutions to the general problem and then present a benchmark optimization problem before discussing our method of choice and our results in solving optimal control problems, Chapter 4.

3.6.1 Gradient Based Methods

The following gradient based methods all use the fact that an extremum lies at a stationary point (gradient is zero) on the objective surface in a valley or atop a hill by using the "steepness" (gradient) to provide the search direction of greatest improvement. Using an analytical function for the gradient or using a numerical approximation from difference equations, the gradient is computed to determine both the direction in the n dimensional space as well as the magnitude of the increment $\{h\}$ for a given iteration. In the following approaches the incrementally better solution is computed by $\{y\}_{i+1} = \{y\}_i + \{h\}$ beginning at a solution sufficiently close to the extremum. The methods differ only in their approach in computing $\{h\}$. At this point we consider the optimization of (3.33) to be a minimization problem for consistency with literature on gradient descent methods.

Unconstrained: Cauchy's (Steepest Descent) Method

Cauchy's method generates an approximation of the surface using a line approximation along the gradient. This can be observed from

$$f(\{y\}_i + \{h\}) \approx f(\{y\}_i) + \nabla f(\{y\}_i)^T\{h\} \tag{3.34}$$

where the second and higher order terms of the Taylor series approximation are ignored. For the optimal control problem formulation it was shown how the gradients can be computed from the

objective functional and dynamic constraints (3.29-3.32) and thus we assume the gradient to be available or computed using a finite difference technique [64]. Rewriting the equation,

$$\Delta J = f(\{y\}_i + \{h\}) - f(\{y\}_i) \approx \nabla f(\{y\}_i)^T \{h\} \tag{3.35}$$

we have an expression for the change in the objective of function J in terms of the gradient at some initial point. This method is also called the steepest descent method because the increment $\{h\}$ is now taken in the opposite direction of the greatest change

$$\{h\} = -\alpha \nabla f(\{y\}_i)^T \tag{3.36}$$

such that $\Delta J < 0$ which is necessary to minimize J. The constant α scales the magnitude of the descent step and is determined such that $\alpha > 0$ (to maintain a descent) and minimizes $f(\{y\}_i + \alpha\{h\})$. The techniques to finding α are known as line search algorithms. These techniques evaluate the objective at several values of α (along the search direction) and then use a quadratic to interpolate three of these points and find the minimum to this approximation. Variations of the basic algorithm exist using a similar technique iteratively for better accuracy ([64]). Having determined α, and the subsequent increment $\{h\}$, Cauchy's method is complete.

This method shows rapid convergence characteristics far from the minimum (but within the same "valley"); however in the near neighborhood of a stationary point (local extremum) the gradient almost vanishes and so does the increment $\{h\}$. Therefore, the method takes subsequently smaller steps close to the optimum and requires many more iterations to converge, especially for high precision.

Unconstrained: Newton's Method

Newton's method differs from Cauchy's method only in its approximation of the objective surface. Here (3.35) is expanded to the second-order term in the Taylor series

$$f(\{y\}_i + \{h\}) \approx f(\{y\}_i) + \nabla f(\{y\}_i)^T \{h\} + \frac{1}{2}\{h\}^T \left[\nabla^2 f(\{y\}_i)\right] \{h\} \tag{3.37}$$

for a quadratic approximation to the objective surface. The second derivative of a scalar function with respect to its dependent variables, $\left[\nabla^2 f(\{y\})\right]$, is known as its Hessian matrix. A necessary condition for a minimum is $\frac{\partial f(\{y\}^*)}{\partial \{y\}^*} = 0$, thus for the next iteration $\{y\}_{i+1}$ to be the minimum of

the quadratic approximation, we substitute $\{h\} = \{y\}_{i+1} - \{y\}_1$into (3.37) and differentiate

$$\frac{\partial f(\{y\}_{i+1})}{\partial \{y\}_{i+1}} \approx \nabla f(\{y\}_i) + \left[\nabla^2 f(\{y\}_i)\right]\{h\} = 0 \tag{3.38}$$

We can resolve (3.38) to determine the desired increment

$$\{h\} = -\left[\nabla^2 f(\{y\}_i)\right]^{-1} \nabla f(\{y\}_i) \tag{3.39}$$

which will find the minimum to the quadratic approximation. The magnitude, or step size, of the increment vector $\{h\}$ is again modulated by a parameter α. The same line search algorithms now apply to Newton's method. With a suitable α, we substitute $\{h\} = \alpha\{h\}$ for the next approximate solution $\{y\}_{i+1}$. This is performed iteratively until either the magnitude of the gradient or the increment is smaller than a desired precision.

This method is very efficient at finding the nearest true minima quickly with the quadratic approximation eliminating slow convergence due to small gradients near the optimum. Difficulties arise in evaluating the Hessian matrix and inverting it. If they can be provided analytically than this method can be used with excellent results. Otherwise the Hessian has to approximated numerically; similar methods which compute an internal representation of the Hessian are called Quasi-Newton methods. Generally these methods require significantly more function evaluations in order for the method to build up a representation of the Hessian matrix, especially if using finite difference approximations. More efficient approximations use matrix updating methods based on the past evaluations of the gradients from the Newton iterations, for example the BFGS matrix updating formula ([11],[73]). These methods have been extended further to avoid matrix inversion by maintaining and updating an inverse representation and are also known as conjugate gradient (1^{st}order) methods [24]. The advantages of the Quasi-Newton methods are that they do not require a user supplied Hessian and they can still provide comparable performance .

Constrained: Penalty Methods

Static optimization problems with constraints can be transformed to unconstrained problems via penalty functions. For example, the constrained problem

$$min \quad J = f(\{y\})$$

subject to

$$g(\{y\}) \leq 0$$

is restated as

$$min \quad J = f(\{y\}) + \phi(g(\{y\}), \rho)$$

The penalty function ϕ is a scalar function that grows as the constraint is approached or violated via an exponential or other growth function. The parameter ρ effects the severity of the penalty such that the penalty becomes less gradual as $\rho \to 0$. There are two basic penalty methods: interior and exterior penalty methods. Interior penalty methods append a continuous penalty to the objective function such that the penalty function grows (within the feasible region) inversely proportional to the distance to the constraint, i.e. $\phi = \rho \frac{-1}{g(\{y\})}$. Here $\rho \to 0$ causes the penalty to have less influence within the feasible space but the penalty increases more sharply when approaching the constraint. Thus when ρ is decreased iteratively, the solution can "slide" closer to the constraint. The external penalty method applies no penalty while the solution is within the feasible space but grows rapidly beyond the constraint boundary. In this case, ρ causes the penalty to grow more severely beyond the constraint bound such that as ρ is decreased solutions are "pushed" to the boundary, Figure 3.3.

The use of either method depends on the particular problem. If violations of the constraints results in errors in a simulation, for example, then an interior penalty would be favored. On the other hand if the constraints represent bounds from conservative limits on performance and where a boundary solution is expected or an interior point initial guess is not guaranteed, then an exterior penalty method would be superior. Of course in the $\lim_{\rho \to 0}$ they are equivalent infinite step functions. When the constraint is an equality constraint then a symmetric (about the constraint) external penalty must be applied, such that the constraint curve has zero penalty. Penalty methods require iterative solutions to the optimization problem with $\rho \to 0$. For the interior penalty functions, this is necessary because it may unduly penalize solutions close to but not violating the constraint. Exterior penalty methods must have sufficiently low values of ρ to ensure solutions have been pushed back to the bound while ensuring there is not a superior internal point solution. Beginning with $\rho \cong 0$ could cause undefined gradients causing the optimization method to fail. Beginning with $\rho \cong 1$ will enable the gradient to be defined while creating an increasingly "steep hill" to push points to the interior. The increasing nonlinearity as $\rho \to 0$ does make it difficult to determine boundary solutions with a high degree of accuracy. Equality constraints are dealt with much more efficiently by the Lagrange-Newton method.

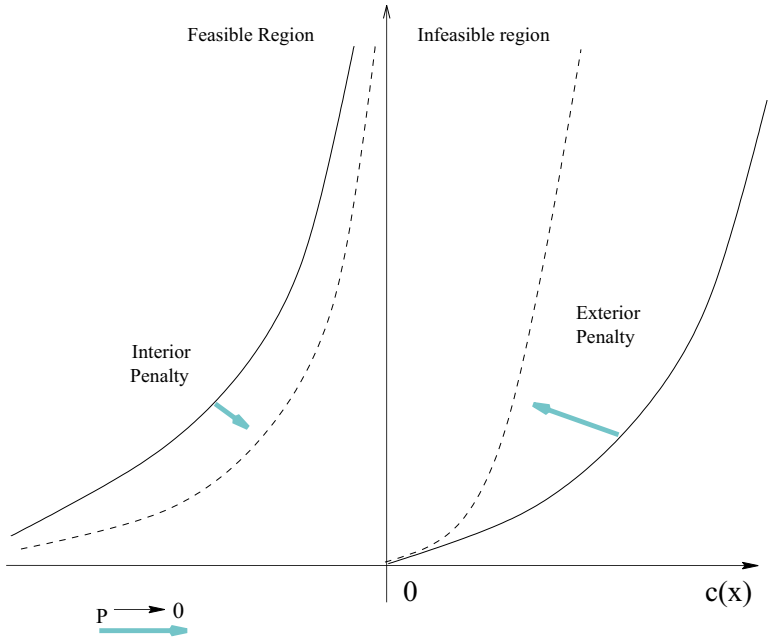

Figure 3.3: Behaviour of penalty functions as $\rho \to 0$ [2]

Constrained: Lagrange-Newton (SQP) Method

This method is based primarily on appending equality constraints by the Lagrange multiplier method, similar to (3.10). Having the resulting scalar function of both the design variables and Lagrange multipliers we under go the same procedure in developing Netwon's method.

For a scalar objective $f(\{y\}$ subject to m equality constraints $\{g(\{y\})\} = 0$ we first define a scalar Lagrangian

$$\Lambda(\{y\}, \{\lambda\}) = f(\{y\}) + \{\lambda\}^T \{g(\{y\})\} \tag{3.40}$$

We assume that the Lagrangian at the next point is the optimal and must satisfy the necessary condition $\nabla\Lambda = 0$. Thus, we take the gradient of the Lagrangian (3.40) at the next point using a second order approximation (excluding higher order terms) from the Taylor series expansion of

$$\Lambda(\{y\}_{i+1}, \{\lambda\}_{i+1}) = \Lambda\left(\left\{\begin{array}{c}\{y\}\\ \{\lambda\}\end{array}\right\}_i + \{\tilde{h}\}\right)$$

where

$$\{\tilde{h}\} = \left\{\begin{array}{c}\{h\}_y\\ \{h\}_\lambda\end{array}\right\} = \left\{\begin{array}{c}\{y\}_{i+1} - \{y\}_i\\ \{\lambda\}_{i+1} - \{\lambda\}_i\end{array}\right\}$$

which yields

$$\nabla\Lambda\left(\left\{\begin{array}{c}\{y\}\\ \{\lambda\}\end{array}\right\}_{i+1}\right) \approx \nabla\Lambda\left(\left\{\begin{array}{c}\{y\}\\ \{\lambda\}\end{array}\right\}_{i}\right) + \left[\nabla^2\Lambda\left(\left\{\begin{array}{c}\{y\}\\ \{\lambda\}\end{array}\right\}_{i}\right)\right]\{\tilde{h}\} = 0 \tag{3.41}$$

similar to (3.38). Using

$$\Lambda = \left\{\begin{array}{c}\nabla f + \nabla\{g\}^T\{\lambda\}\\ \{g\}\end{array}\right\}$$

and

$$\Lambda_{\{y\}\{y\}} = \left[\nabla^2 f + \sum_{j=1}^{m}[\nabla^2 g_j]\lambda_j\right]$$

with the definition of $\{h\}_\lambda$, we can rewrite (3.41) in matrix form

$$\left[\begin{array}{cc}\Lambda_{\{y\}\{y\}_i} & \nabla\{g\}_i^T\\ \nabla\{g\}_i & 0\end{array}\right]\left\{\begin{array}{c}\{h\}_y\\ \{\lambda\}_{i+1}\end{array}\right\} = -\left\{\begin{array}{c}\nabla f_i\\ \{g\}_i\end{array}\right\} \tag{3.42}$$

which can be solved for the current increment in the design variables, $\{h\}_y$, as long as the constraints $\{g\}$ are linearly independent and $\Lambda_{\{y\}\{y\}}$ is not singular.

This method is also called the Sequential Quadratic Programming (SQP) method since $\{h\}_y$ is also the solution to a quadratic programming (QP) problem which minimizes $q_i(\{h\}_y) = \nabla f_i^T\{h\}_y + \frac{1}{2}\{h\}_y^T\Lambda_{\{y\}\{y\}_i}\{h\}_y$, subject to $\{g\}_i + (\nabla\{g\}_i)\{h\}_y = 0$. This problem is equivalent to minimizing the current change (maximum decrease) in the objective J (3.33) from a quadratic surface approximation q (3.33) that is subject to a linear approximation of the constraint curve. The resulting system of equations for the solution of $\{h\}_y$ from the QP problem are identical to the Lagrange-Newton result (3.42). The sequence of solutions to QP approximations in an iterative approach results in the SQP.

As presented above, there is no guarantee that $\{h\}_y$ will in fact result in a decrease in the objective function. To avoid divergence, $\{h\}_y$ is scaled so that $\{y_{i+1}\} = \{y\}_i + \alpha\{h\}_y$, where α is chosen, once again, to ensure $f(\{y\}_{i+1}) < f(\{y\}_i)$. Specifically we construct a penalty function ϕ from which we can determine $\alpha > 0$ such that

$$\phi(\{y\}_i + \alpha\{h\}_y, \{r\}) < \phi(\{y\}_i, \{r\}) \tag{3.43}$$

$$\phi(\{y\}, \{r\}) = f(\{y\}) + \{r\}^T\{|g(\{y\})|\} \tag{3.44}$$

The set of positive parameters $\{r\}$ ensure the constraints at the next point are not violated. The formula

$$r_{(i+1)_j} = \max(|\lambda_{(i+1)_j}|, \frac{1}{2}(|\lambda_{(i+1)_j}| + r_{i_j})),\ i = 1, 2, ..., m \tag{3.45}$$

was demonstrated by Powell [63] to provide good results. Note that $\{\lambda\}_{i+1}$ is determined at the current SQP iteration via (3.42). Beginning with $\alpha = 1$, if condition (3.43) is not satisfied then we decrement α until the condition is met.

The SQP method is far more efficient than penalty methods since the constraints are used to help "guide" the increment towards the feasible optimal rather than "blocking" infeasible regions. The method also does not suffer from the numerical instabilities resulting from penalty functions.

Inequality constraints are not handled explicitly. However converting the inequality constraints to equality constraints by the introduction of slack variables (3.13) or eliminating the constraints via a penalty function (Section 3.6.1) are possible adaptations. The penalty methods have their problems. Slack variables effectively add more design variables and further increase the dimensionality of the problem, which can introduce more complexity to already high dimensional problems.

An additional method can be applied with the SQP method. The method of active constraints takes advantage of the fact that the optimization is iterative in order to determine from a set of inequality constraints which constraints are active in the computation of the next SQP step. The active subset is added to the equality constraints and the subsequent computation of the solution increment includes their influence, while the inactive constraints are ignored. This method requires some additional "book-keeping", but does not affect the complexity of the solution method. To begin, inequality constraints that are violated by the initial solution are considered active. In subsequent iterations the same violation check is performed plus the inclusion of the inequality constraints that have positive Lagrange multipliers $\{\lambda\}_{i+1}$ even if these constraints are currently not violated. $\{\lambda\}$ can be loosely interpreted as a constraint "reaction force", which resists movement normal to the constraint path and that $\{\lambda\}_{i+1}$ can be thought of the "anticipated reaction" at the next point. For inequality constraints the concern with a "reaction" keeping the solution inbounds is represented by a positive value, and thus should be included in the subsequent increment. For an inequality constraint we want to ignore reactions of movement towards an interior point. Thus, we remove previously active inequalities with negative λ_i values.

3.6.2 Global Search Methods

The previous methods are considered local optimizers since they descend quickly towards points on the objective surface where the gradient is zero and where the Hessian is positive definite (for a minimum). However, these conditions can be satisfied at multiple points on the surface, known as local minima. Which local minima is found depends solely on the initial point from which the descent begins. For objective surfaces that are highly nonlinear with "dimples" and "bumps"in the surface, gradient information can be misleading. In noisy or rough surfaces the gradient methods can be trapped very close to the starting point.

Unfortunately, the only method that can guarantee the global optimum is complete enumeration of the objective surface. Although exhaustive and robust the computational requirements make it impractical for most applications. Following from that is the random or Monte-Carlo optimization method, which randomly samples the space keeping the best point to date. After a sufficiently desirable solution is found or the current best has not been surpassed after some terminal number of iterations, the method is stopped. Very simple, but still extremely expensive computationally.

The following methods are two general approaches which have been shown to be more effective by borrowing from natural laws to generate heuristics for stochastic optimum seeking. They are called stochastic methods because they both rely, to some degree, on random perturbations which provide the explorative nature of the techniques. Thus, they take irreproducible (non-deterministic) paths to the optimum while being consistent in finding the optimum, unlike a pure random search. Since these methods sample the solution space and do not require a continuous surface for computing gradients, discontinuous objective functions with undefined or non-existent derivatives do not create a problem.

Simulated Annealing

Simulated annealing is based on a thermodynamic model of the optimization problem such that it is analogous to a liquid metal. In the cooling of hot liquid metal, loosely bound atoms come to align themselves such that the material can obtain a lower energetic state which is a solid. This process is called annealing. When cooled slowly the annealing process results in ordered arrangements of atoms over distances tens of orders larger than the size of an atom which is

a crystalline structure representing the lowest energy state attainable by the arrangement of atoms. If cooled too rapidly the atoms do not align over such large distances thus resulting in the polycrystalline structure. The analogy here is that the gradient methods are like the rapid cooling process that does not enable the system to reach the lowest energetic state, i.e. the optimal state [64]. Thus, in practice the design variables of our optimization problem are analogous to the atomic configurations and the objective function is the scalar measure that is analogous to the energy of the configuration [64]. The temperature and the "annealing schedule" determine the size of the random perturbations (high temperatures generally mean larger potential changes in configuration) and the rate or the increments at which the temperature is lowered. A probabilistic function is used to determine when the system changes configuration (i.e. a new solution) and is 1 if the resulting change will decrease the total energy, otherwise a probability of increasing the total energy is assigned for the same temperature. This enables a "realignment" step to occur even though it may require a transition to a higher energy state before further lowering.

Evolutionary methods

Evolutionary methods and simulated annealing share many similarities. First, evolutionary algorithms are also stochastic, with randomness introduced via mutation of individual solutions. Solutions take the analog of organisms instead of configurations and changes in configuration are performed via reproduction operators instead of a probability distribution. The analogy for the objective function is different in that it casts the general problem as a maximization of a scalar fitness value. The most significant difference is the concept of a population. All evolutionary methods maintain a set of solutions which then compete to reproduce the next generation by a "survival of the fittest" heuristic [34]. This enables multiple branches of organisms (or species perhaps) to evolve in parallel until the fittest one dominates. In simulated annealing most of the exploration occurs early on in the annealing process when the temperature (or randomness) is high, but as the system is cooled a single annealing path is selected perhaps at the cost of another. The evolutionary method, however, because of parallel evolution, can exhaust multiple evolutionary branches [69].

3.6.3 The Genetic Algorithm

The genetic algorithm (GA) is an evolutionary method that we have examined more closely for determining the globally optimal controls with greater reliability [77]. The GA is based on the method of transfer of genetic material in the natural or biological world and the performance and survival of species. Biological systems do not maintain a mathematical model of their world in which to determine the optimal performance, yet most creatures perform very well and perhaps optimally for the environment they live in. A reason for this is the encoding of all features of the organism from the function of a single cell to the complete skeletal structure. This encoding through reproduction ensures that those features that enabled the current organism to survive are passed to its offspring. Through genetic recombination, surviving pairs have increased chances of producing fitter offspring. Life itself is the test if the new genes are worthy to be passed on by surviving to maturity and being able to reproduce as well.

In an artificial systems context, survival is determined by the successful performance evaluated by the designer. Based on the evaluation of performance, known as the fitness function, an "organism's" chances of reproduction are determined. Similarly, an artificial organism's parameters, which define its current performance, are encoded in strings known as chromosomes which are analogous to the biological structures containing genes. After a chromosome is selected according to its fitness, reproduction occurs via crossover. Crossover is a method of combining successful individuals, which interchanges equivalent lengths of their chromosomes. Finally, variation is introduced into the new population by random mutation. Mutation occurs only at a single base (letter or digit) of the coded chromosome for a set of bases that are selected at random within the entire population. The result of selection, crossover and mutation is a new population with potentially more individuals of increased fitness. After several generations, the population will converge on the chromosome that provides the highest fitness value attainable — that is the optimal organism.

The genetic algorithm can be outlined as follows:

1. Design variable encoding and initial population generation
2. Fitness evaluation of each individual in the population
3. Select a surviving portion of the population based on the fitnesses

4. Mate selected individuals using a cross-over technique

5. Apply random mutations to the offspring and replace the old population

6. Repeat 2-5, until the population is dominated by a single chromosome

Encoding Design Variables

Design variables are encoded into string representations where each character is known as a base. The most common method is binary encoding (0/1 bases) such that each design variable has an equivalent binary representation. The binary representation according to the Schema Theorem and the Building Block Hypothesis [44], can best leverage the recombination of fittest features represented as sub-strings. We present an anecdotal argument which supports the Schema Theorem. First we consider any (optimal) value of say 10 (in decimal). The equivalent binary encoding is [001010] and in base 4 representation (0->3) the equivalent encoding is [022]. If we change (by mutation or cross-over) the least significant bit we get [001011] and [023] as the equivalent binary and base 4 representation for 11. If we consider the new chromosomes in terms of a ratio of the bases appearing in the original chromosome, we find that the binary maintains $\frac{5}{6}$ of the original encoding, while the base 4 yields only $\frac{2}{3}$. It has been concluded that the binary encoding can preserve more features through the evolutionary processes and thus possesses stronger convergence characteristics [34].

In our implementation of the genetic algorithm the appropriate number of bits are allocated based on the range of design variables and their desired precision. A chromosome is generated by concatenating the appropriately sized strings for each design variable. An initial population of chromosomes is generated by randomly populating the chromosome bits for a user defined population size, N_p. Population size is an important parameter in any genetic algorithm and selecting the appropriate size is dependent on several factors. A large chromosome typically requires a big population, since the chromosome length defines the size of the search space. If many local optima are anticipated then to ensure that each "hill" is sufficiently explored, the population size should facilitate multiple evolutionary paths.

Fitness Functions

Fitness functions represent the mapping from the chromosome encoding of the design variables to a scalar value of the chromosome's performance and is essentially the objective function. GA's are essentially maximization operations and thus the fitness function should apply the necessary transformation.

Constraints are included by penalizing the fitness of an individual for violating the constraints and for the degree of violation. Thus for casting the general constrained minimization described earlier, we rewrite the scalar objective J (Section 3.33) subject to constraints $\{g(\{y\})\} = 0$ and $\{c(\{y\})\} \leq 0$ by the fitness function

$$Fit = F_{max} - J - \sum_{i=1}^{m} \beta_i |g_i|^{\alpha_i} - \sum_{j=1}^{l} \beta_{m+j} (c_j + |c_j|)^{\alpha_{m+j}} \tag{3.46}$$

F_{max} is a large positive constant that enables all fitness values to be positive, although it is not essential in our algorithm. The constant factors $\{\beta\}$ and exponents $\{\alpha\}$ determine the severity of the penalties for the $m + l$ constraints. The values for penalty constants depend largely on the problem. Consideration should be given to potential tradeoffs that may arise from individuals that perform well but violate constraints, versus those that are feasible but perform poorly. If feasibility is the primary characterization for performance then sufficiently large penalties must be applied. The critical job of the fitness function is to accurately segregate the population such that better solutions have higher fitness values.

Selection Process

There are several selection methods available and include the roulette wheel, ranking, stochastic selection and elitism methods, to name a few. The process we employ is a combination of multiple techniques. Essentially the population is divided about the median fitness value for the current population and those less than the median are discarded. From the remaining chromosomes, their fitnesses are shifted such that all values are above zero, if necessary. An exponential scaling is applied to further separate the fitnesses based on the dominance (homogeneity of the population) and the number of iterations where the fitness has remained the same in order to drive the method to converge.

The total fitness is evaluated so we can express individual fitnesses as a ratio of the total fitness which then defines the individuals' selection probabilities on a "roulette wheel". N_p random "rolls" are used to then select the current population. Elitism is also supported in two ways: first, the selected population is forced to have at least one top performer; second, on the condition that the maximum fitness of the new generation does not equal or exceed the past best it is reinserted in place of the new worst performer.

Genetic Recombination

From the selected population, mating partners are randomly assigned. The method of genetic recombination is via a multiple crossover technique. The mean number of crossover points is a user specified parameter that enables more genetic intermingling for larger chromosomes than can be provided by single crossover [13]. The actual number of crossover points is achieved by "blurring" the mean number of crossovers with a normal distribution centred on the mean value so that for any generation it can be slightly greater or less than specified.

Multiple crossover is achieved by randomly selecting crossover points for a particular couple and then sorting them in ascending order. Our method, for an even number of crossover points, uses sequential pairs of crossover points to define the substrings that are interchanged. Similarly for an odd number of crossover points, the first location of the chromosome is included in the list of crossover points and we proceed as before. The recombined chromosomes then replace the parents in the new population.

Multiple crossover is thought to be more effective, especially for long chromosomes, in order to ensure that short and fit schemata (fundamental substrings) can form and recombine [50].

Mutation Technique

We have found that adjusting the mutation rate dynamically can be a powerful method of ensuring the GA is explorative enough while providing the opportunity for fit schemata to form and recombine so that the method does not degenerate into a random search. The technique is quite simple and requires a measure of dominance, D, and the number of iterations at which the maximum fitness has remained the same, I_{same}. Beginning at a user defined value the mutation rate is scaled as I_{same} approaches the maximum number of iterations, I_{max}, with the same fitness

and furthermore as the population is dominated. Thus, at the point of convergence the mutation rate can reach up to twice the set value and a minimum of half the value if the fittest shows almost no dominance.

This helps to stabilize the algorithm such that when a new fittest is found it can have the opportunity to dominate instead of being lost by random mutations. The opposite is also true; instead of premature convergence the algorithm adapts the mutation rate to increase its explorative capabilities. We have found in our applications that the variable mutation rate has been a key element to the robust performance of the GA.

Convergence Criteria

As mentioned previously, the two criteria which define dominance in our implementation of the GA are: 1) exceeding the maximum number of iterations for which the maximum fitness has remained the same, $I_{same} > I_{max}$ and 2) attaining a dominated population, $D > D_{max}$. Dominance is a normalized measure of every chromosome's Hamming distance to the fittest chromosome, such that a value of 1 means all chromosomes are identical to the fittest and a value of 0 means they are identically not equal to the fittest. For a completely random distribution the dominance ranges between 0.40 and 0.60 and values exceeding 0.9 mean near complete dominance of the population. Due to random mutations the dominance is generally bounded by $D_{max} \leq 1 - 2M_r$, where M_r is the mutation rate.

Lookup Table of Past Performances

Recent GA implementations ([6] and [50]) typically include some form of lookup table in order to eliminate the computational burden of re-evaluating chromosomes that appear multiple times in a generation or reappear in later generations. When computationally intensive simulations are part of the fitness function, this provides marked time savings. However, for less intensive evaluations trade-offs are made with regards to the time to search a large table and time for re-evaluation. In our work we have found that re-evaluations occur with the greatest frequency within the current and the last generations, with chromosomes further than four generations past having very few re-occurrences. Thus searching can be greatly reduced by checking the current as well as the past three generations for possible re-evaluations. Searching of long binary strings

defining the chromosomes also lead to time-consuming inefficiencies. By keying the database by a unique number such as the decimal representation of the binary strings, searching of moderately sized databases becomes very inexpensive.

We implement a lookup table that maintains a database of n_g to $2n_g$ of the most recent generations with pruning occurring only when the database reaches more than $2n_g N_p$ (N_p is the population size), where n_g is a user specified minimum number of generations to be maintained.

3.6.4 Hybrid Methods

Several authors [44],[69] propose hybrid methods combining some or all of the features of a global method such as the genetic algorithm with the speed and precision of gradient based methods such as the Quasi-Newton method. Techniques for hybridization include a simple two step approach using the GA to identify the optimal region and the gradient method to converge on the precise optimum as well as more highly coupled methods. In the first case the GA provides a sufficiently accurate initial guess to the gradient method.

A hybrid method proposed by Renders and Flasse [66] integrates the the two techniques more intimately. In their approach, the application of the Quasi-Newton method for each chromosome at each generation represents the "life" of the individual which is equivalent, for example, to several Newton iterations. At the end of its life, the chromosome will have "grown" to be a fitter individual and therefore it is the mature chromosome that now replaces the infant in the population with its improved fitness value. The method then performs another iteration in the GA using the mature chromosomes and their fitness values, resulting in no fundamental change to the GA. In fact, all that is required is for the fitness function to perform a few Newton steps of a Quasi-Newton optimization and then to return the final fitnesses as well as the mature chromosomes to the genetic algorithm.

The resulting hybrid method offers much faster convergence in comparison to the GA alone if the objective (fitness) function is locally smooth (i.e. continuous double differentiable). Otherwise, the local search method can make little progress and the method defaults to a performance equivalent to the GA. Computation times for locally smooth functions are typically less than N_p times longer than the Quasi-Newton method alone because as the method approaches convergence, fewer and fewer new individuals arise requiring fewer computations thanks to the GA's look-up

table.

With a local SQP optimization method we can also handle constraints more efficiently via the hybrid approach by applying the constraints explicitly in the SQP steps and thus assuring that the resulting solutions are feasible.

The genetic algorithm and the additional functionality to compute SQP steps with constraints was implemented as a MATLAB function in `genetic.m` (Appendix B.1).

3.7 Comparison of Methods

Having briefly discussed local gradient based, global heuristic-stochastic, and hybrid optimization methods, we turn to a sample problem for evaluating the robustness, precision and computational resources (time) required by these methods. As a representative of the gradient methods we chose the Quasi-Newton method with the gradients and Hessian computed via a finite difference method and an updating scheme [11]. This method is available as a function, `fmin`, in MATLAB's optimization toolbox. For the global methods we applied the genetic algorithm which was described in Section 3.6.3 as well as a hybrid GA/SQP from Section 3.6.4.

We have selected a benchmark problem proposed by Schaffer [1989] and represented in the text by Lin and Lee [44], that provides the following unconstrained optimization problem:

$$max \quad f(\{y\}) = 0.5 - \frac{\sin^2\left(\sqrt{\{y\}^T\{y\}}\right) - 0.5}{\left[1.0 + 0.001(\{y\}^T\{y\})\right]^2} \tag{3.47}$$

Where $\{y\} = \{y_1.y_2\}^T$ and the range for the two design variables is $-100 \leq y_i \leq 100$ and the solution is desired to a precision of two decimal points. This problem poses several difficulties: first the surface has infinite local optima (concentric rings) (Figure 3.4); local optima exist (first concentric ring) which have a value very close ($\cong 0.995$) to the global optima of 1; and finally the design variable range is large.

Although the surface is smooth and twice differentiable, the Quasi-Newton applied directly to the minimization of $-f(\{y\})$ provides poor reliability in that only initial guesses within the first concentric ring ($|\{y\} - \{y^*\}| \leq 1.253$) converge to the global optimum. For the desired precision the GA encoding required 22 bits per design variable, and for a population of 100 chromosomes

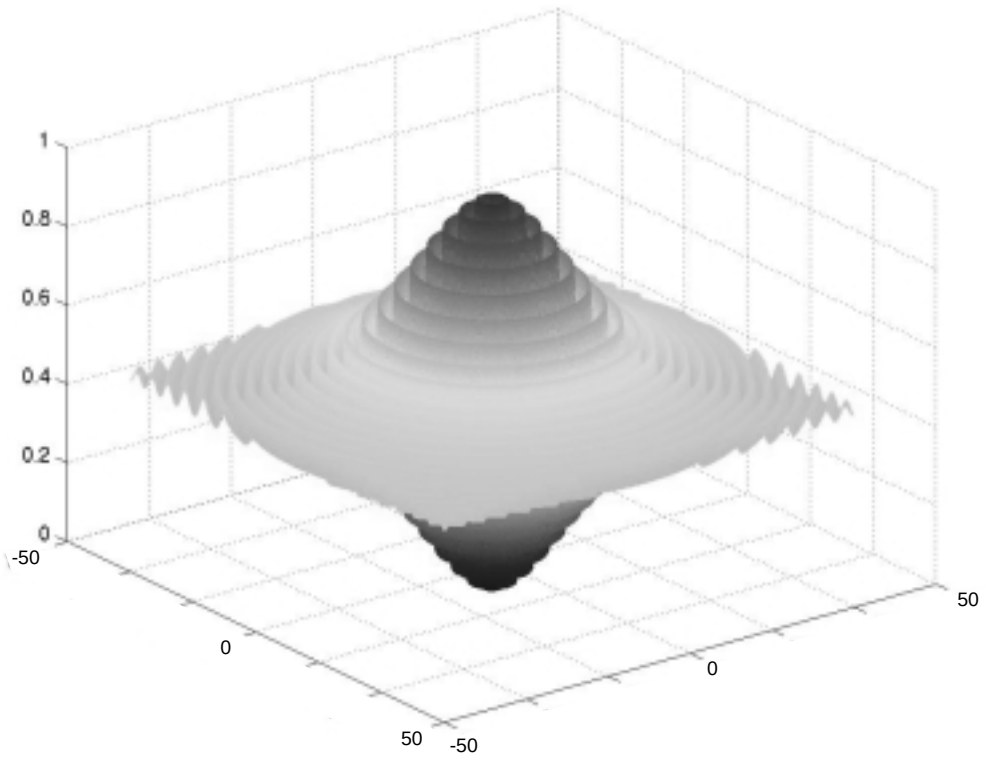

Figure 3.4: Benchmark objective function with infinite local optima

it typically did not converge before 40 generations (less than 4000 function evaluations). Out of 15 trials, the GA found a solution on the global (central) peak only 4 times with a maximum objective value of 0.997. The hybrid method enabling each chromosome to grow on each iteration via five Newton iterations of the Quasi-Newton method provided the precise optimum (to within 10^{-5}) 14 times of the 15 trials. Only 1 trial converged to the local optimum of the first concentric ring with a population size of 20. The number of generations was always within 18 generations for an average less than 1400 function evaluations. Doubling the population increased reliability to 100% with convergence under 17 generations and less than 2400 function evaluations, Table 3.1. The mutation rate was initially defined to be 0.02 (which is varied within the algorithm) and the number of crossover locations was set to 2 for the the GA in both approaches.

This test function has several features that we can assume hold true for optimal control problems in general. First the objective surface is smooth and, second, it is highly non-linear and, more importantly, non-convex (non-concave). For the optimal control problem we can expect the objective to be at least piecewise differentiable [84] with respect to the controls since the equations of motion from the system dynamics are generally nonlinear but continuous. The hybrid method is able to capitalize on the smoothness of the function while using the genetic operators to avoid being trapped in local optima.

The high dimensionality of the optimal control problem was also simulated by increasing the dimension of the solution space by making $\{y\} = \{y_1, y_2, ..., y_{10}\}^T$, i.e. $N_D = 10$. Only the hybrid-method was studied for this case and resulted in all of the 15 trials converging to the optimal solution with a population size of 120 and holding all other parameters the same. Even more impressive convergence was observed with convergence to the precise optimal solution being found within 50 generations with less than 52,000 (and as low as 42,000), function evaluations (Table 3.1). Although the search space increases by the power of the number of design variables, the computational effort only increases in proportion to the number of design variables times the factor increase in population size!

Based on these results we believe a GA-SQP optimization can provide the greatest reliability for the optimal control problem using a direct solution method. Computing gradients internally from Equations 3.30-3.31 will also reduce the computational requirements over the test case which were computed by the Quasi-Newton method using finite difference equations.

The specific implementation and results of this approach applied to robotic and biomechanical

Table 3.1: Test function optimization results from the GA-SQP method

N_D	N_p	Relability (%)	mean $nfevals$
2	20	93	1377
2	40	100	2370
4	80	100	10903
10	120	100	51913

models of the arm are presented in Chapter 4.

Chapter 4

Implementation and Prediction Results

Using the optimal control methods developed in Chapter 3 we investigate the performance of arm models with specified performance objectives. First, we test our algorithm with a benchmark minimum-time maneuver of a planar 2-link manipulator. Second we consider the biomechanical model of the human upper-limb for similar pointing and lifting tasks with the some biologically-inspired objectives. For each model the detailed implementation including the model parameter values for segment inertias, joint locations and muscle attachment locations are included in Appendix A.

4.1 Performance of a Planar 2-link Manipulator

The benchmark minimum-time problem has been investigated by several researchers, [25] and [31]. Its results are well known, which will enable us to validate our algorithm's performance. The planar manipulator was described previously (Section 2.3.3) and the equations of motion for the manipulator are presented in (2.6-2.9) with the specific model parameters included in Appendix A.1.

4.1.1 The Optimal Control Problem for the Planar Manipulator

The minimum time objective is given by

$$min \quad J = \int_0^{t_f} dt \tag{4.1}$$

subject to the dynamic equations in state space form, (2.10)

$$\{f(\{x\},\{u\},t)\} - \{\dot{x}\} = 0 \tag{4.2}$$

the initial and final kinematic conditions

$$\{\phi\}|_0 - \{X_0\} = 0 \tag{4.3}$$

$$\{\phi\}|_{t_f} - \{X_f\} = 0 \tag{4.4}$$

and the bounds on the control torques $\{u\} = \{T_1, T_2\}^T$

$$\{|u|\} \leq \left\{ \begin{array}{c} 25 \\ 9 \end{array} \right\} \tag{4.5}$$

The initial and final end-effector kinematic constraints are defined in state coordinates by

$$\{\phi\} = \left\{ \begin{array}{c} l_1 \cos(x_1) + l_2 \cos(x_1 + x_2) \\ l_1 \sin(x_1) + l_2 \sin(x_1 + x_2) \\ -l_1 \sin(x_1)(x_3) - l_2 \sin(x_1 + x_2)(x_3 + x_4) \\ l_1 \cos(x_1)(x_3) + l_2 \cos(x_1 + x_2)(x_3 + x_4) \end{array} \right\} \tag{4.6}$$

which describes the end-effector position and velocities in Cartesian coordinates.

For comparison with known results we define the the initial position such that the arm is fully extended along the Cartesian X-axis and the final position as the point were the arm is fully extended on the Y-axis. Combining this with the fact that both initial and final velocities are zero,

$$\{X_0\} = \left\{ \begin{array}{c} l_1 + l_2 \\ 0 \\ 0 \\ 0 \end{array} \right\} \quad and \quad \{X_f\} = \left\{ \begin{array}{c} 0 \\ l_1 + l_2 \\ 0 \\ 0 \end{array} \right\} \tag{4.7}$$

In order to use a direct solution method we must rewrite the problem such that the final state is free and penalize J accordingly. Thus the approximate objective becomes

$$min \quad J' = \Phi + \int_0^{t_f} dt \tag{4.8}$$

$$\Phi = \rho\{S\}^T\{S\} \tag{4.9}$$

where $\{S\} = \{\phi\}|_{t_f} - \{X_f\}$ and ρ is a penalty weighting. The new objective is subject to the dynamic equations (4.2) and the bounds on the controls (4.5) only, which can be appended by the method of Lagrange multipliers. Note that the only free variable is the final time of the integration. We can parameterize the final time such $t_f = p$ and normalize the integration time such that $\tau = \frac{t}{p}$. The dynamic equations in normalized time, τ, are $\frac{d\{x\}}{d\tau} = p\{f(\{x\},\{u\},t)\}$. Adding the final time parameter p to the final state cost Φ and appending the dynamic equations we get the final form of the objective

$$min \quad J'' = \Phi' + \int_0^1 \left(\{\lambda\}^T \left\{ p\{f(\{x\},\{u\},\tau)\} - \frac{d\{x\}}{d\tau} \right\} \right) d\tau \tag{4.10}$$

where

$$\Phi\prime = r\{S\}^T\{S\} + p^2 \tag{4.11}$$

subject to

$$-p \leq 0 \tag{4.12}$$

and the limits on the controls (4.5). These limits can be enforced directly by constraining the optimization method to vary the design variables within the prescribed range. We can now evaluate the objective function by integrating the dynamic equations forward in τ by linear interpolation (3.25) of the discrete approximations of the controls $\{\bar{u}\}$ and an estimate of the final time, p.

4.1.2 Planar 2-link Model Functional Variations

To compute the gradients with respect to the design variables — the discrete controls $\{\bar{u}\}$ and parameter p — we compute the variations in the Hamiltonian,

$$H = \{\lambda\}^T \left\{ p\{f(\{x\},\{u\},t)\} - \frac{d\{x\}}{d\tau} \right\} \tag{4.13}$$

and get the optimality conditions

$$H_{\{x\}}: \quad \{\lambda\}^T\{f\}_{\{x\}}p = -\frac{d\{\lambda\}}{d\tau} \tag{4.14}$$

$$H_{\{u\}}: \quad \{\lambda\}^T \{f\}_{\{u\}} p = 0 \tag{4.15}$$

$$H_p: \quad \{\lambda\}^T \{f\} = 0 \tag{4.16}$$

The Jacobian of the state space equations $\{f\}$ with respect to the state $\{x\}$ and controls $\{u\}$ were computed in MAPLE and are easily described analytically in terms of the original dynamic equations (2.6).

$$\{f\}_{\{x\}} = \begin{bmatrix} [0] & [1] \\ ([M]^{-1}\{\Im\})_{\{x\}} & \end{bmatrix}_{(4\times 4)} \tag{4.17}$$

$$\{f\}_{\{u\}} = \begin{bmatrix} [0] \\ [M]^{-1}\{\Im\}_{\{x\}} \end{bmatrix}_{(4\times 2)} \tag{4.18}$$

The final conditions on the co-states can be computed directly from the final state cost

$$\{\lambda\}|_1 = \Phi'_{\{x\}}|_1 \tag{4.19}$$

which provide the initial conditions for integrating the co-state equations backwards in time to determine the co-state trajectories. From the co-state trajectories we can evaluate $H_{\{\bar{u}\}}$ and H_p over the performance time (for the set of design variables used to evaluate J'') and use (3.30) and (3.31) to compute $\nabla J''$, (3.32). The computation is performed in MATLAB via .m files: `planarEqns`, `costateEqns`, `simplePlanar`, and `planarGrads`. The state and co-state equations are coded in `planarEqns` and `costateEqns` respectively. These functions are integrated by their respective calling functions, namely `simplePlanar` and `planarGrads` via a NAG variable step 4^{th} order Runge-Kutta ODE solver [53]. The function `simplePlanar` evaluates the objective value and `planarGrads` computes the gradients according to the most recent state and integrated co-state equations. These files are included in Appendix A.1.

The initial state is chosen such that it satisfies (4.3) which conveniently is $\{x\} = \{0\}$.

4.1.3 Minimum-Time Maneuver Results

The above objective and gradient functions were used in our implementation of the hybrid GA-SQP technique outlined earlier (Section 3.6.4). The control limits were handled explicitly by the

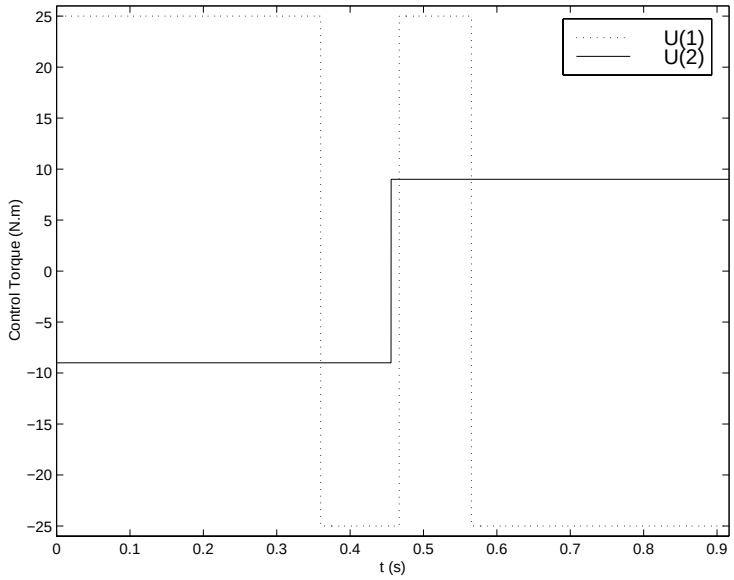

Figure 4.1: The "bang-bang" optimal controls of the planar 2-link manipulator

GA-SQP method and thus do not appear in the objective function. There are no auxiliary constraints. The control mesh was taken with 51 nodes (discrete instances). The hybrid optimization method was configured with: a population size of 80; a baseline mutation rate of 0.02; a mean of two cross-over points; and an upper bound of 10 Newton iterations per fitness function evaluation. Compatibility of the fitness function values (GA maximization) with the objective function values (SQP minimization) for the respective methods was maintained by taking the fitnesses to be $Fitness_i = 2000 - J_i$, where J_i is the value of the objective function after an individual has taken up to 10 Newton iterations. The final dominant individual was then used to run the SQP to completion. The precision on the design variables for the hybrid-method were coarsely set at 0.1 and the final time parameter at 0.01. The final SQP was given a convergence tolerance of 10^{-4} for both the design variables and the objective function.

The following results were generated without using any *a priori* knowledge that the counter revolution of l_2 is a superior strategy. Finding this solution can be attributed to the explorative nature of the GA.

Our minimum time result of $p = 0.9185$ (Figure 4.1) results are virtually identical to the results obtained by Geering [25] ($p = 0.92$) and by Heilig and McPhee [31] ($p = 0.9164$) who

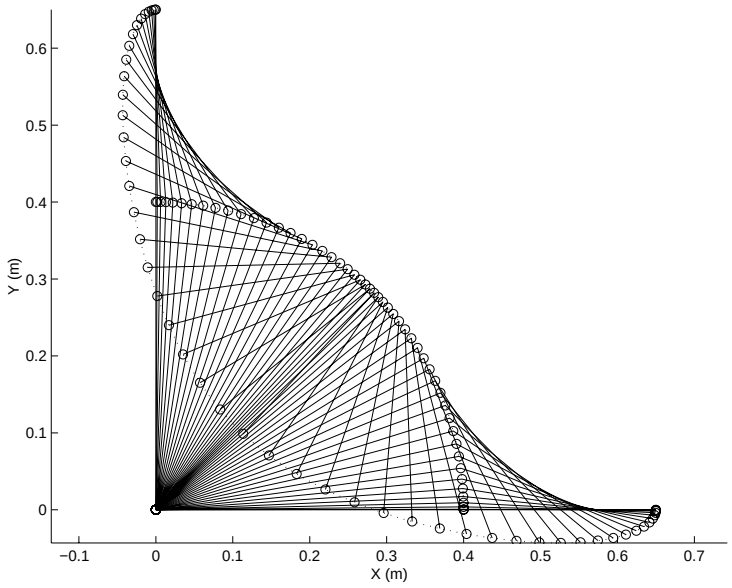

Figure 4.2: The optimal trajectory for the planar 2-link manipulator

were able to get extremely precise results by using a finer mesh in a direct SQP method and experimenting with various initial trajectories. In these previous cases, however, the final state was prescribed such that $x_2 = -2\pi$, whereas our implementation only imposed constraints, $\{\phi\}$, on the end-effector position and velocity. The set of feasible solutions is significantly larger in our definition of the problem with at least one other local minima resulting in $\{x_2 = 0\}$ which has an optimal time of only 1.2371. It is interesting to note that a very coarse encoding of only 10 nodes with a GA optimization provided sub-optimal results but still demonstrated the optimal counter revolution strategy [72].

Our results required 32487 function evaluations and only 10420 gradient evaluations in 143 generations. The integration of the state equations varied between 0.7 and 1.5 seconds to compute, on a Silicon Graphics Indigo 2xZ workstation. Evaluating the gradients ranged from 1.0 to 4.0 seconds taking more time in the initial phases when the terminal constraints are grossly violated and the initial conditions for the co-states are very large. The variation in time can be attributed to the variable step size of the Runge-Kutta method which adapts to the "smoothness" of the dynamic equations which can be very erratic in the initial phases due to purely random controls. The total time for evaluation was just under 15 hours.

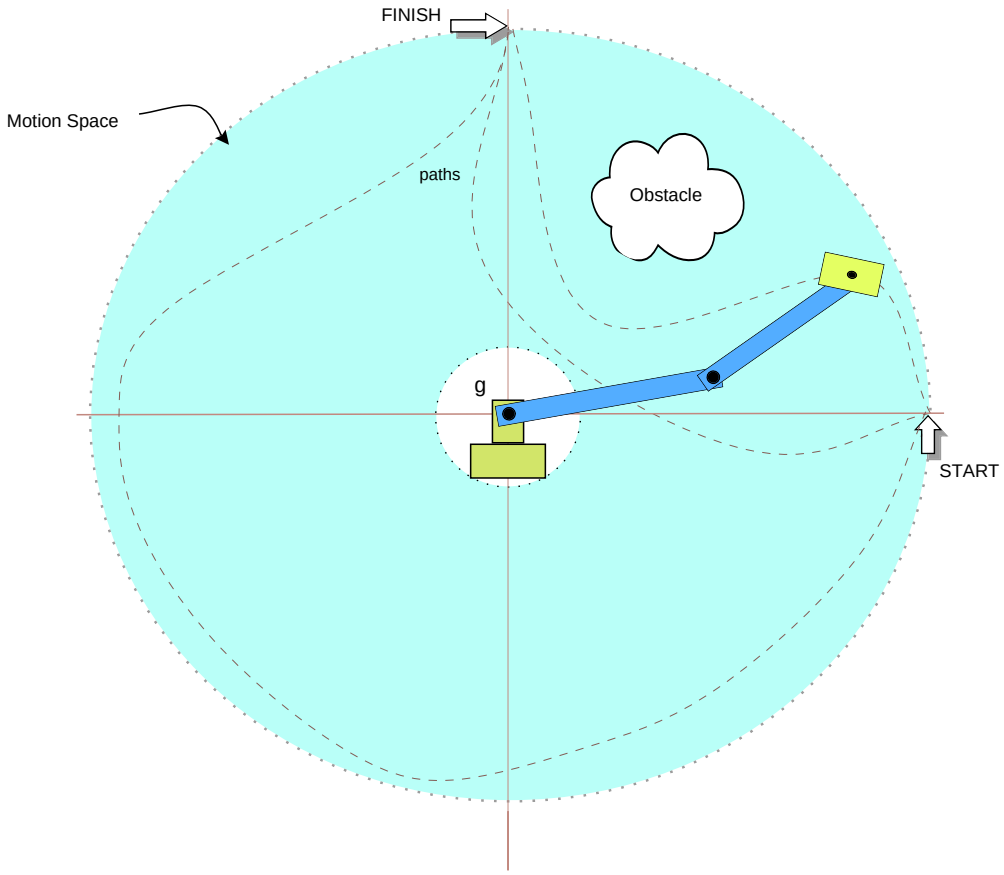

Figure 4.3: Feasible paths in the motion space of the planar 2-link manipulator

4.1.4 An Alternate Path Parameterization Method

In striving to increase the efficiency of our method, various parameterizations were considered. Most significant was the idea of parameterizing the end-effector path. This resulted in solving an optimal control sub-problem in finding the minimum time controls along a specified trajectory. Notably the work of Bobrow [8] was implemented using paths parameterized by points in Cartesian space interpolated by a cubic spline. Bobrow's method resolves the controls by establishing a profile for the maximum acceleration or deceleration along the path. The optimal control is a set of optimal switches which toggles the controls between maximum acceleration or deceleration.

The GA was then used to move the nodes in the end-effector motion space, Figure 4.3. The main advantage of this method is that all solutions are inherently feasible and always satisfy the

terminal state conditions. Almost identical results were obtained using only four nodes (four pairs of coordinates in the horizontal plane) and thus only eight design variables opposed to 103 variables in the above method. Surprisingly, this method was slower due to the time consuming solution of the optimal sub-problem that is evaluated by the fitness function, even though only 982 fitness function evaluations were required.

The path parameterization method remains an attractive method for proper-actuated open-loop systems. However an oversight on our part resulted in a failure to recognize that this method is ineffective for over-actuated systems. In order to resolve the optimal sub-problem a linear combination of the controls must produce a unique acceleration vector on the path which is always true for proper actuated systems. For redundant muscle actuators there are infinite combinations that can provide the same end-effector acceleration, and thus the individual controls must be determined by yet another optimal decomposition method, *a la* the inverse dynamics method. Although possible to resolve the problem this way, the additional complexity was abandoned for the present direct method.

4.2 Upper-Limb Biomechanical Model Performances

The five muscle actuated upper-limb model (Figure 2.4) is used in this investigation with the shoulder and elbow joints modelled as revolute joints and the wrist joint as a weld. Thus the model has two degrees of freedom and five actuators and thus can be considered to be over-actuated. In addition, the bicep (F_2) is a two-joint muscle connection to the ground body (fixed scapula) and the forearm segment. The motion of the arm is in the vertical plane and thus the effects of gravity have been included in the dynamic equations. Hill's equation is used as the muscle actuator model (Section 2.1.4) with the length and velocity of contraction determined directly from the segment kinematics. The normalized ($0 \rightarrow 1$) activation levels to the muscle models represent the five control variables $\{u\} = \{u_1, u_2, \ldots, u_5\}^T$. The graph representation (Figure 2.5) was used to generate the input file (Appendix A.2.1) to DynaFlex . The resulting equations of motion have been implemented in `bioArm` (Appendix A.2.2).

4.2.1 The Optimal Control Problem for the Biomechanical Arm

The objective for the biomechanical arm now includes minimizing time as well as a scalar measure (ζ) of the joint reaction forces and can be described by

$$min \quad J = \int_0^{t_f} (1 + \zeta(\{x\}, \{\dot{x}\}, \{u\}, t)dt \tag{4.20}$$

subject to the dynamic equations in state space form,

$$\{f(\{x\}, \{u\}, t)\} - \{\dot{x}\} = 0 \tag{4.21}$$

auxiliary state constraints that account for the articular limitations of the elbow joint,

$$(x_2 - \pi)x_2 \leq 0 \tag{4.22}$$

as well as initial and final state kinematic constraints

$$\{\phi\}|_0 - \{X_0\} = 0 \tag{4.23}$$

$$\{\phi\}|_{t_f} - \{X_f\} = 0 \tag{4.24}$$

and bounds on the control inputs

$$\{0\} \leq \{u\} \leq \{1\} \tag{4.25}$$

The measure of the joint reactions (ζ) is fundamentally a projection of the joint reaction forces onto the local frame of each joint's distal body. For example, the reaction force at the second joint is expressed in terms of forces parallel to the long axis of the forearm body and perpendicular to its long axis. The perpendicular force is used to represent the shear force on the joint and a negative reaction force along the long axis represents the tensile (pulling) reaction force. We only take into account the joint shear and tensile forces while ignoring compression when computing ζ. The hypothesis is that shear and tensile forces contribute to joint separation and thus muscle forces that work to maintain joint integrity can only be be duplicated if these measures increase the performance cost. Mathematically,

$$\zeta(\{x\}, \{\dot{x}\}, \{u\}, t) = \mu_1 \frac{1}{2}(|A_1'| - A_1') + \mu_2 A_2 + \nu_1 \frac{1}{2}(|B_1''| - B_1'') + \nu_2 B_2 \tag{4.26}$$

where $\{A'\}$ and $\{B''\}$ are the joint reaction forces in the respective distal body frame and the constants $\{\mu\}$ and $\{\nu\}$ are weightings on both the tensile and shear components of the respective

joint reactions. The first component of the reaction forces represents the longitudinal reaction force and using its absolute value minus itself yields twice its magnitude if the component is negative (i.e. tensile); otherwise it is zero. The integration of the dynamic equations (4.21) yields the state trajectory which is used to determine the acceleration of the centre of masses of each segment. The individual muscle forces and accelerations allow us to solve for the the joint reactions at each time step and evaluate $\bar{\zeta}(t_i)$ which is then integrated using the trapezoidal rule. The result is the cumulative cost of the shear and tensile reaction forces. The projection of the reaction forces is dependent on the joint angles $\{x\}$ and the forces require the acceleration of the segments which are dependent on the state velocities and accelerations $\{\dot{x}\}$ as well as the muscle forces which are dependent on $\{x\}$ and the controls $\{u\}$.

We include articular limitations of the joints as explicit joint constraints as in (4.22) which limits the second joint angle between extreme extension ($x_2 = 0$) and extreme flexion ($x_2 = \pi$). These limits may not reflect the precise joint limits but simply demonstrates how this method includes joint limits.

The initial and final state constraints define the particular task. In this case the kinematic constraint equations, $\{\phi\}$ are identical to those of the planar manipulator (4.6), but we have redefined the terminal conditions such that

$$\{X_0\} = \begin{Bmatrix} P_{0_x} \\ P_{0_y} \\ 0 \\ 0 \end{Bmatrix} \quad and \quad \{X_f\} = \begin{Bmatrix} P_{f_x} \\ P_{fy} \\ 0 \\ 0 \end{Bmatrix} \tag{4.27}$$

These terminal states describe a lateral view of a task of raising one's hand from rest in the "at ease" position, P_0, (arm along side) and reaching (lifting) such that the hand (end-effector) comes to rest at the highest vertical point, P_f, to which the individual can reach. We intend to determine the start (P_0) and end (P_f) from kinematic data collected directly from a subject performing a vertical reach task. Similarly, segment lengths, muscle origin and insertion points, and inertial estimates are also to be determined from the test subject (Chapter 5).[1]

As in the previous benchmark example, using the direct method requires the freeing of the final state terminal condition and the addition of an appropriate final state penalty, Φ (4.11).

[1] The initial and final points were approximated from the 3D kinematics before the kinematics were projected on to the most significant plane. See Section 5.8.1.

Using the same definition of the final penalty and parameterizing the final-time as in the previous example we can rewrite the objective.

$$min \quad J' = \Phi' + \int_0^1 \left(1 + \zeta(\{x\}, \frac{d\{x\}}{d\tau}, \{u\}, \tau, p) + \{\lambda\}^T \left\{ p\{f(\{x\}, \{u\}, \tau)\} - \frac{d\{x\}}{d\tau} \right\} \right) d\tau \quad (4.28)$$

subject to the final time parameter (4.12) and control bounds (4.25). The initial state constraints are always satisfied since they are used to initialize the forward integration of the equations of motion using the discrete control approximation $\{\bar{u}\}$ and a guess of the final time parameter, p.

4.2.2 Biomechanical Arm Model Functional Variations

Once again, to compute the gradients with respect to the design variables $\{\{\bar{u}\}, p\}$, we must compute the variations of the system Hamiltonian

$$H = 1 + \zeta(\{x\}, \frac{d\{x\}}{d\tau}, \{u\}, \tau, p) + \{\lambda\}^T \left\{ p\{f(\{x\}, \{u\}, \tau)\} - \frac{d\{x\}}{d\tau} \right\} \quad (4.29)$$

The specific first-order optimality conditions for this system are,

$$H_{\{x\}}: \quad \zeta_{\{x\}} + \{\lambda\}^T \{f\}_{\{x\}} p - \frac{d}{d\tau}(p\zeta_{\{\dot{x}\}}) = -\frac{d}{d\tau}\{\lambda\}^T \quad (4.30)$$

$$H_{\{u\}}: \quad \zeta_{\{u\}} + \{\lambda\}^T \{f\}_{\{u\}} p = 0 \quad (4.31)$$

$$H_p: \quad \zeta_p + \{\lambda\}^T \{f\} = 0 \quad (4.32)$$

The Jacobian of the state space equations $\{f\}$ with respect to the state $\{x\}$ and controls $\{u\}$ can be determined symbolically for relatively simple functions as in the previous example. However for complex system dynamics, both the Jacobian of the state space equations and the gradient of the cost functional cannot be resolved analytically. In the current example, the gradient of the reaction force measure ζ is much more difficult to determine analytically than for the final time functional of the previous example.

For more complex problems, we can approximate the variations, $H_{\{x\}}$, $H_{\{u\}}$ and H_p from finite difference approximations of the Jacobian of the state space equations, $\{f\}_{\{x\}}$and $\{f\}_{\{u\}}$, as well as the gradients of the scalar cost $\zeta_{\{x\}}$, $\zeta_{\{\dot{x}\}}$, $\zeta_{\{u\}}$ and $\zeta_{\{p\}}$. The situation is not so bleak. From

the forward integration of the state space dynamic equations we can simultaneously compute the derivatives of the dynamic equations as well as the cost functional with respect to the states at each node of the integration.

We begin by eliminating a set of first order derivatives, namely $\frac{d}{d\tau}(p\zeta_{\{\dot{x}\}})$, by direct substitution of the state equations, $\{\dot{x}\} = \{f(\{x\},\{u\},t)$, into the scalar cost functional, such that $\zeta_{\{\dot{x}\}} = 0$. Now,

$$H_{\{x\}}: \quad \zeta(\{u\},\{x\},p\{f\}))_{\{x\}} + \{\lambda\}^T\{f\}_{\{x\}}p = -\frac{d}{d\tau}\{\lambda\}^T \tag{4.33}$$

We can approximate $\{f\}_{\{x\}}$ with a finite difference equation using some of the intermediate Runge-Kutta (RK) evaluations used to determine the state, $\{\bar{x}\}$, at the following node [64]. These intermediate evaluations as well as the nodal values are then used in a finite difference approximation of $\{f\}_{\{x\}}$, such that

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\{x_1,\ldots,x_j+\Delta x_j,\ldots\}_i,\{u\}_i,\tau_i) - f_i(\{x_1,\ldots,x_j,\ldots\}_i,\{u\}_i,\tau_i)}{\Delta x_j} \tag{4.34}$$

Similarly, we can evaluate $\zeta_{\{x\}}$, $\{f\}_{\{u\}}$, $\zeta_{\{u\}}$ and ζ_p by the same procedure given sufficient evaluations at adequate increments. By saving all the trajectory points $(\{\bar{x}\}_i, \{x_{RK}\}_i, \{\bar{u}\}_i)$ [2] from the forward integration of the dynamic equations and the corresponding results for $\{f\}$ and ζ we can compute the gradients at each step of the backwards integration of the co-state equations (4.33) as we solve for the co-state trajectories. Furthermore using an adaptive step size RK method we can insure that the variations provide good derivative information since smaller steps are taken through intervals of rapid change in the system dynamics. We obtain the starting conditions by evaluating the finite difference approximation of $\Phi_{\{x\}}$ at the final state (4.19).

4.2.3 Increasing Computational Efficiency

The method of internal differentiation — internal because the computation of the gradients are made within the simulation (or backward ODE solution) — saves a tremendous amount of computational effort. The equivalent solution, performed taking the finite difference of the objective function with respect to the design variables (outside of the simulation), would require an objective function evaluation (a complete simulation) for each variation in a design variable which translates to $Nm+p+1$ simulations per gradient evaluation, neglecting updating schemes, with N,

[2] RK indicates the intermediate Runge-Kutta integration points.

m, and p, being the number of nodes, controls, and parameters in the optimal control problem. Instead, we essentially perform only two integrations (forward and backward in time), similar to the analytical method used in the previous example. In general this procedure provides results on par with using analytical gradients for moderate to low precision with 60-80% savings in computational effort in comparison to external differentiation methods [9].

The coupled RK integrator with an SQP optimization has been implemented by Professor Fabien at the University of Washington [20], and was recently modified for our application. The C source files and PERL scripts are listed in Appendix B.1. The package, DYNOPT, is quite straightforward to use. There are two C source files, `dyn_sqp.c` and `math_opt.c`, included with DYNOPT that perform the SQP functions including the evaluation of the objective functional and its derivatives, as detailed above. A user defined function which returns the cost functional, dynamic equations, and constraints along with parameters defining the number of nodes, states, controls, model parameters, constraints and terminal conditions is generated by the PERL script, `dynopt`. The script parses a user input file containing the system variables, parameters and equations and generates a C file, which includes the `main` function. It then compiles the user defined function and links the object files from `dyn_sqp.c` and `math_opt.c` to form a stand-alone executable. The C executable is over an order of magnitude faster at executing a Newton iteration than the same operation performed in MATLAB. This is very fortunate since solving the optimal control of the biomechanical arm, which is a significantly more difficult problem requires many more iterations.

DYNOPT was applied to solve the optimal control of the biomechanical arm with several arbitrary initial guesses for the controls and final time. However, none of these trials were able to converge even to a sub-optimal solution. It appears that there is an abundance of local minima which violate the final state conditions resulting in all of our test trials to be trapped within five to ten Newton iterations. This case exemplifies the difficulty of resolving the optimal control problem for biomechanical systems. Providing an initial guess such that the SQP method can progress to the optimal solution is not a trivial task.

Modifications were made to both `dynopt` and `dyn_sqp.c` so the that we could integrate our GA with DYNOPT's existing capabilities. These included adding an input argument to the main function to enable us to control the number of iterations and providing functions for exporting the objective and constraint values for use by a fitness function. The GA, originally developed in

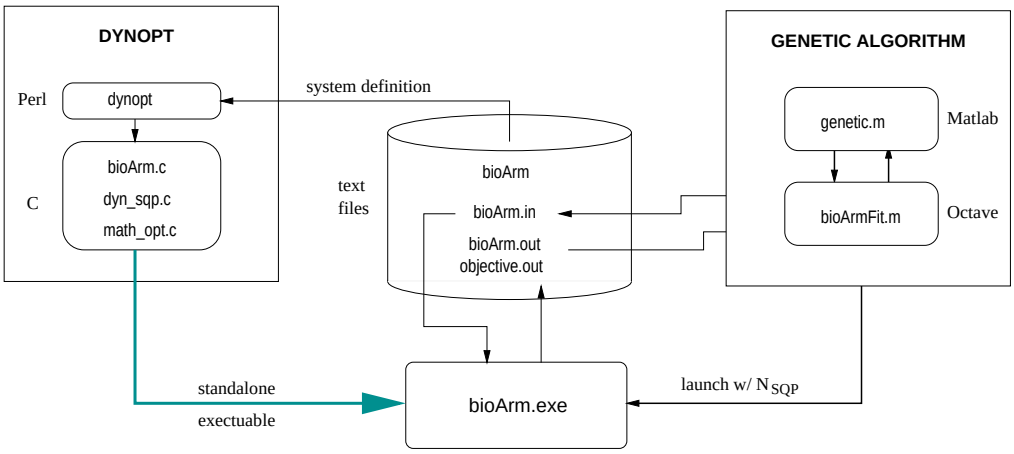

Figure 4.4: Functional organization of DYNOPT and GA integration

MATLAB, was modified to run in OCTAVE on a LINUX platform. OCTAVE provides a platform for rapid prototyping similar to MATLAB and accepts most MATLAB scripts and functions. OCTAVE does not have the overheads of supporting structures and variable sized arrays, thus is significantly faster but comparable to older versions of MATLAB. The conversion was primarily made because OCTAVE is open source software available for free and could run the GA on a PC with a LINUX operating system.

The execution of the DYNOPT generated executable, `bioArm.exe`, is performed by the fitness function implemented in `bioArmFit.m` which is a MATLAB/OCTAVE script. This function acts as a "wrapper" for the stand-alone executable and communicates with it via input and output files (Figure 4.4). The input file includes the controls at the nodes and the final time parameter (decoded from a chromosome in the population). The executable returns the objective function and magnitude of the constraint violations and the new design variable values after attempting N_{SQP} steps set by the calling fitness function. The fitness function writes out the necessary files before executing `bioArm.exe`, then reads in the output file to replace the old chromosome and assign it a fitness value. We define the fitness of an individual as

$$Fit_i = 2000 - J_i^2 - 1000(|\{g\}|_i^2) \tag{4.35}$$

where J_i is the objective function value and $|\{g\}|_i$ is the magnitude of the vector of active constraints for the present solution. Both J and $|\{g\}|$ are returned in the objective output file by `bioArm.exe`. Further speed gains could be made by eliminating file-system communication and

will be addressed in future implementations.

4.2.4 Optimal Control Results for the Upper-Limb Model

Once again a 51 node mesh was used to discretize the problem. The inclusion of muscle models should act to smoothen the dynamic response of the system to oscillating controls such that the discrete controls are always good approximations. In other words, the muscle models act as filters such that "bang-bang" controls produce smoother force profiles and therefore the discretization and finite difference gradients are good approximations[3]. With 51 nodes and and five muscles we now have 256 design variables. To limit the chromosome lengths somewhat, the controls were given a precision of 0.2, but we maintained a final time precision of 0.01. A population size of 120 was used and the remaining GA parameters were held constant from the previous example.

Many attempts were made at solving this optimal control problem with no success. It became evident that the model with the controls bounded $(0 \rightarrow 1)$ was incapable of performing the task due to the definition of the muscles. The muscles in our model linearly connect two points which at certain states causes the moment arm of the muscle to diminish, or even worse, to cause the opposite function. For example, if we examine the moment generation capabilities of the triceps (Figure 4.5) we can observe that the muscle is nearly incapable of producing an extensor moment about the elbow at approximately 90°, and worse yet, at greater angles the triceps cause a flexor moment! In reality, the tendon connecting to the insertion point wraps around the joint such that the triceps always provides an extensor moment. Furthermore, this phenomenon is not limited to the triceps and therefore, this model is biomechanically invalid.

As an exercise for our optimal control method we have continued with this model by relaxing the control bounds $(-1 \leq \{u\} \leq 2)$ such that the actuators can now push as well as pull. This should also take into account the contribution of muscles that were not included such as the posterior deltoid being a strong shoulder extensor.

In the first run we considered no load at the end-effector; thus m_{21} (Figure 2.5) is only the mass of the hand. The results were generated after 285 generation of the GA-SQP with the total number of SQP steps just falling short of 203,000 (including the integration of the co-state

[3]Although 51 nodes were used in the previous example, discretization error was less of a concern because analytical gradients were computed analytically

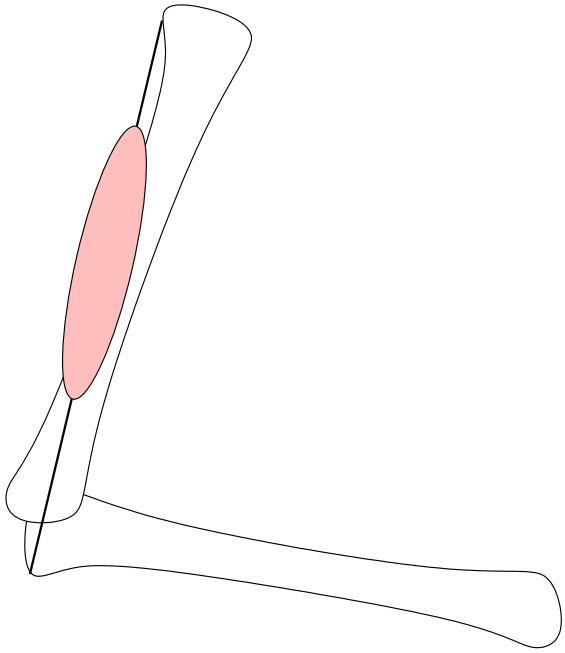

Figure 4.5: Triceps muscle with a linear two-point interconnection

equations). The computation time was approximately 30 hours on a Pentium PC running LINUX. The motion of the arm is traced in Figure 4.6. The height and mass of a test subject together with experimental measures of link lengths were used to estimate the masses and moments of inertia of the segments as well as the muscle origin and insertion locations on the bodies [39]. As noted earlier, the start and end points of the task were approximated directly from the calculated spatial rotations of the bodies and the derived link lengths (Section 5.3.4), prior to the projection of the data to the most significant plane.

The performance time is quick with a final time of $0.328s$. The corresponding joint trajectories are illustrated in Figure 4.7. The identical problem with the same objective function was run with an end-effector load of $5kg$. The following results were computed in 437 generations with 315,055 SQP iterations. The computation of the optimal solution required close to 62 hours. The optimal motion of the upper-limb with a load (Figure 4.9) is surprisingly similar to the performance without the load; however the performance time of $0.715s$ is almost twice as long as without the load.

In both tasks, we can observe a counter-movement preceding the motion where the humerus

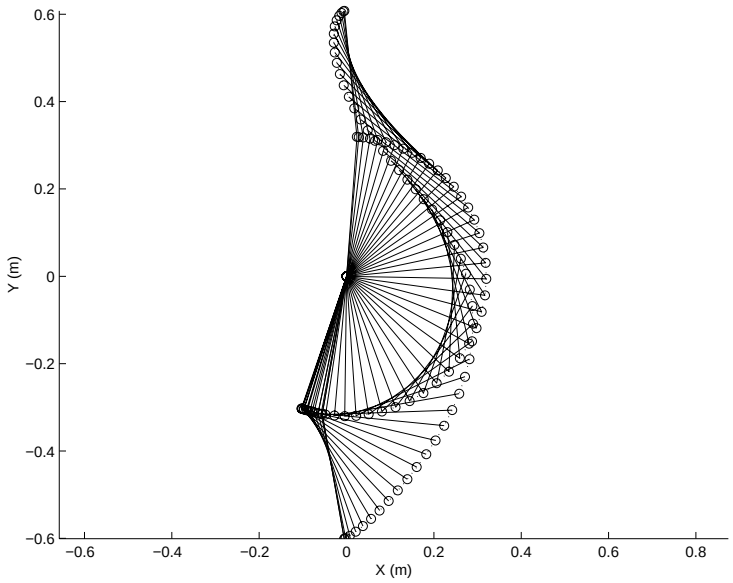

Figure 4.6: Optimal motion of the biomechanical upper-limb model performing a vertical reach.

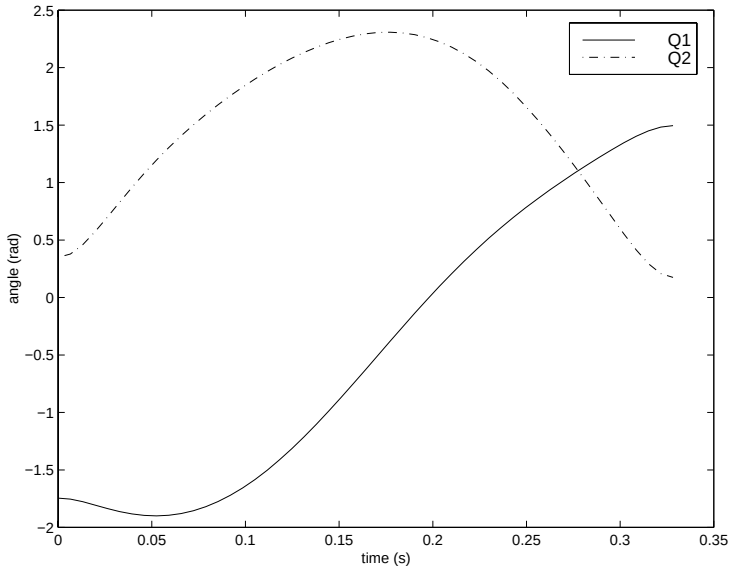

Figure 4.7: Model predicted optimal angular trajectories of a vertical reach task.

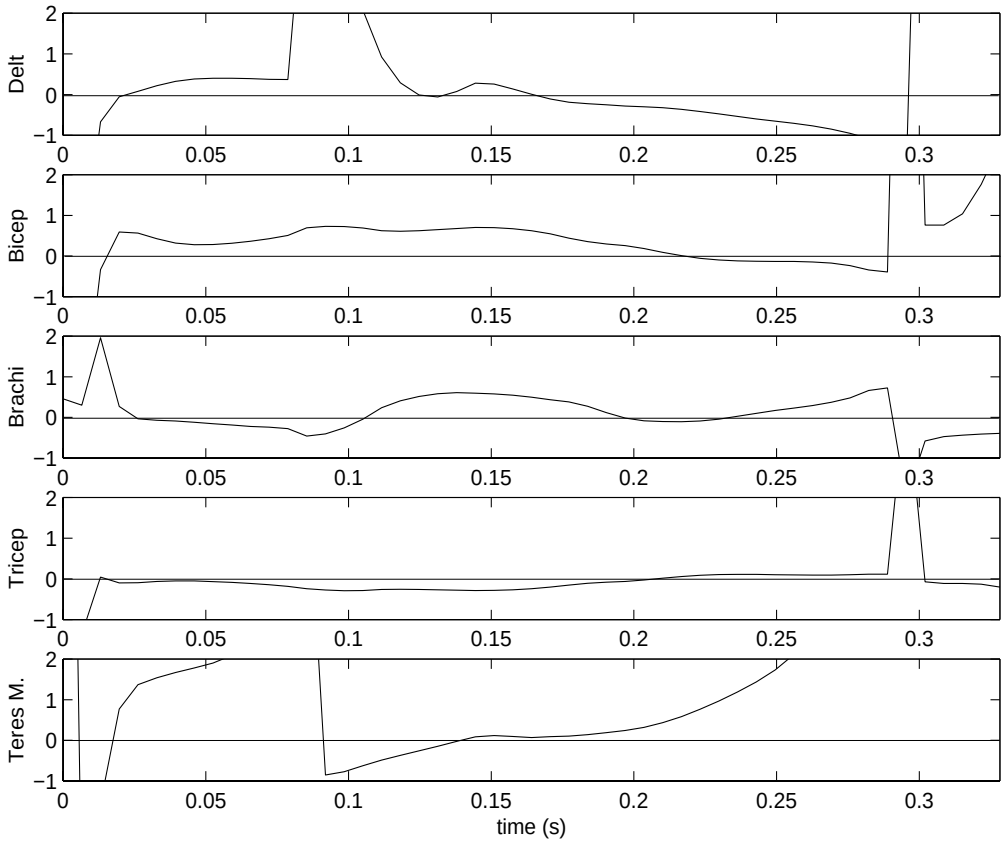

Figure 4.8: Optimal controls for the biomechanical arm performing a vertical reach

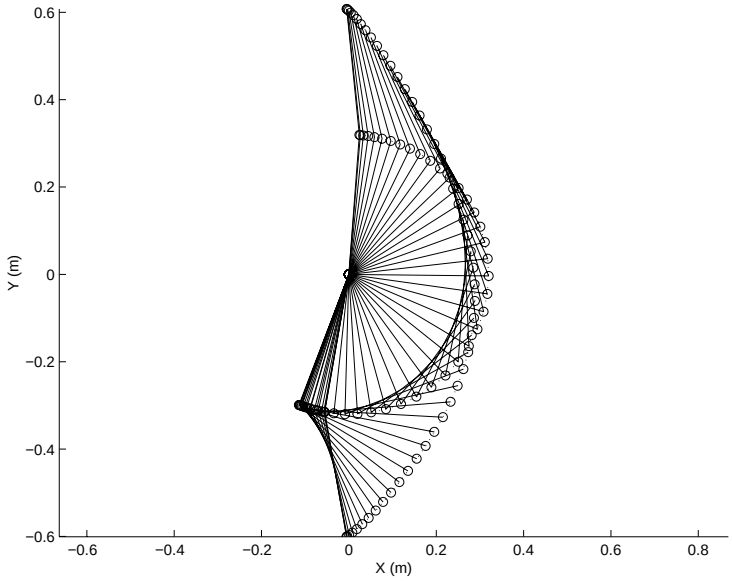

Figure 4.9: Optimal motion of the biomechanical arm model performing a vertical lift ($5kg$).

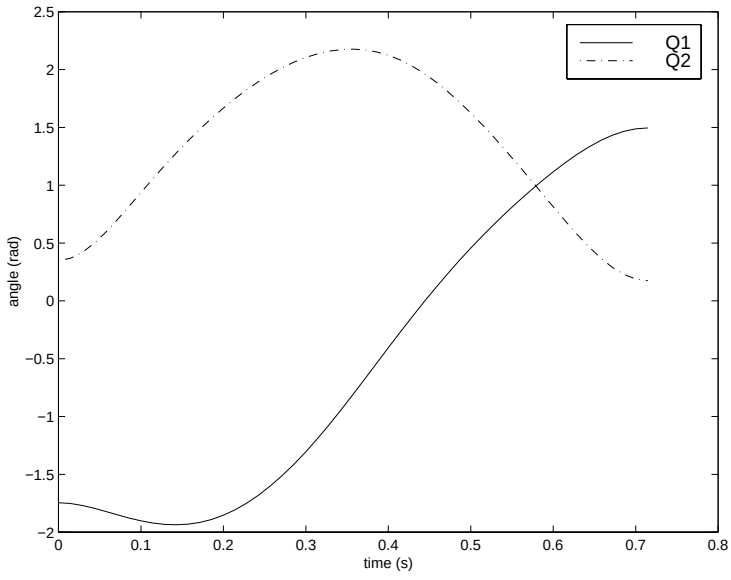

Figure 4.10: Model predicted optimal angular trajectories of a vertical lift ($5kg$)

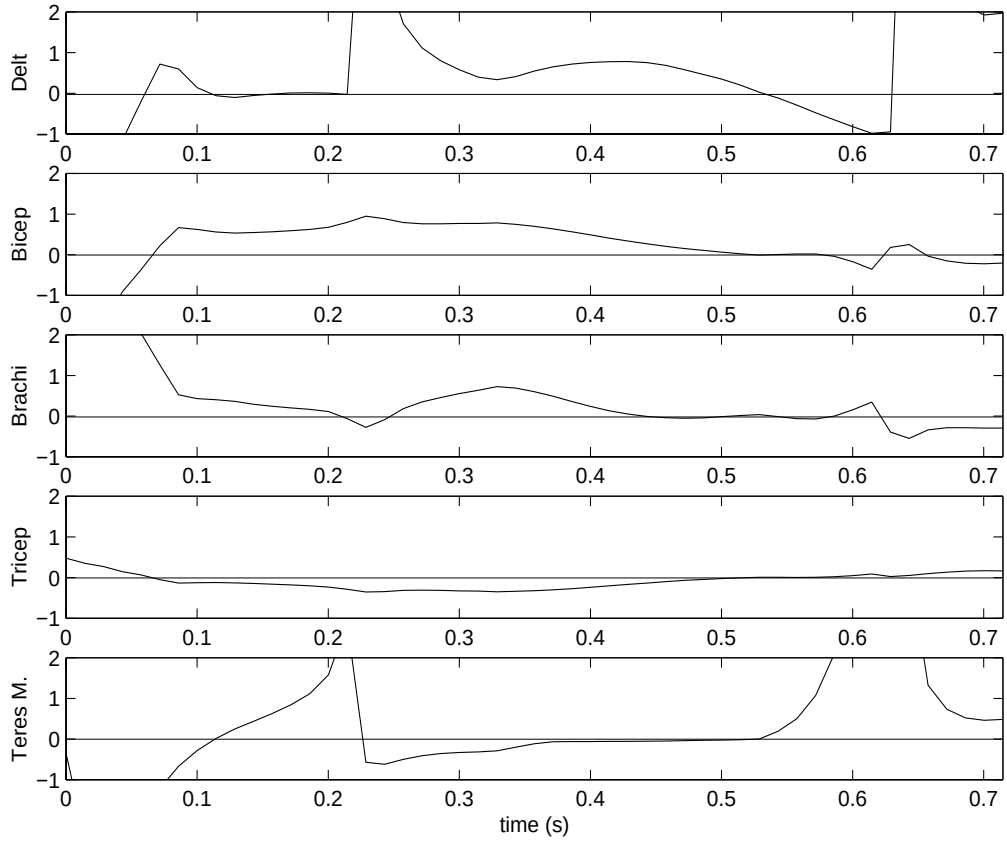

Figure 4.11: Optimal controls for the biomechanical arm performing a vertical lift ($5kg$)

is pulled back before swinging the forearm forward. There are definitely two phases of the motion where the elbow is first flexed to reduce the moment of inertia about the shoulder and then the extension of the elbow as the shoulder reaches maximum flexion. With a load, we see that the elbow becomes nearly fully extended well before reaching the target whereas without a load, the triceps are used more aggressively to "snap" the forearm into the final position.

The controls to the muscles have less physical relevance due to the error in the modelling. In spite of the modelling error there are a few features that should be noted about the results. In the performance of both tasks there is an early "push" by the deltoid which partially explians the initial extension of the shoulder (Figures 4.8and4.11). This indicates that the posterior deltoid, responsible for extension of the humerus, should have been included in the model as well. We see similar behaviour from the biceps.

Although the upper-limb model does not reflect the true nature of human upper-limb dynamics, the results do present some strategies that could be used to understand and evaluate human performances, especially in terms of the motion kinematics. This is verified by the analysis of a test subject's performance of similar tasks, which is presented in the following chapter.

Chapter 5

Data Collection and Comparison Methods

In this chapter, we step back from the predictive control methods and acknowledge that our predictions will have to be validated. Since the goal is to predict human behavior with our model, it is appropriate to collect motion data from a test subject and to compare the results. Herein we capture and analyze the movement of a single subject. The anthropometry of the subject was used to scale our biomechanical model and to generate the movement predictions in Chapter 4. Data collected from our test subject performing various tasks under went several levels of processing to facilitate the comparison with model predictions.

5.1 Human Motion Collection Objectives

The objectives of the collection process were to obtain data that could be readily used to compare the controls and kinematic output from our motion prediction scheme. Thus from pre-defined tasks we wish to determine:

1. Right Upper Limb Kinematics
    - position vs. time history of upper limb segments

- relative orientation changes between segments

2. Muscle activation patterns

   - muscle electromyography (EMG)
   - determine periods of low, medium and high activation

The kinematics of the segments provide a direct comparison with the state trajectories of our model. The EMG from the muscles provides us with both the timing and the relative level of activation which is comparable to the controls in our biomechanical model.

5.2 Task/Movement Definition

Capturing the movement of the arm performing various tasks was the objective of this collection procedure. Tasks were defined by an initial and final location of the hand, while initial and final velocities are implicitly taken to be zero. More tasks are collected here than was necessary for comparison with the current results, because we anticipate making future comparisons as both the model and optimization methods improve. In addition, once a subject and the data collection equipment have been set up, it was much easier to collect a whole battery of trials rather than perform the collection on separate occasions for every task.

5.2.1 Parameters

1. *Initial* and *final location*, orientation and *velocity* of a given segment (hand), within a global reference system, are the parameters that define the kinematic goals that must be accomplished to successfully perform that task.

2. Selected anthropometry of the tested individual (model scaling):

   The amount of anthropometry required is quite small since the marker data can be used to give accurate estimates of the limb segment *lengths*.

   (a) For the purposes of scaling a biomechanical model, the segment lengths along with the *height* and *total body mass*, of the individual, can be used to estimate the inertial properties of the individual segments [39].

(b) The scapular dimensions are unknowns, thus the length of the spine of the scapula (*scapular breadth*) is recorded to scale a future model of the scapula.

(c) In addition, the *girth* of the torso at the xiphoid is necessary to define the thorax and used to determine the scapulo-thoracic surface constraint.

3. Applied or carried loads for tasks requiring an external load to be moved, must be known.

4. Spatial location and volume of obstacles:

(a) The *circumference* of the head and *height* of the top of the head from the sternum is necessary to construct the head volume that acts as an obstacle in some tasks.

(b) The *height*, *width*, and *location* of a square frame forming an obstacle on a table top will define an external obstacle to be avoided.

5.2.2 Constraints

1. The tasks cannot involve the motion of body segments other than the right upper limb, otherwise they may confound results. Artifacts from the movement of the torso will confound any results in future comparisons, thus it is necessary that the sternum be held still.

2. All necessary data for a particular task should be collected simultaneously while the task is being performed to ensure the validity of the measured electromyography with the observed kinematics for the specific individual.

5.2.3 Task Selection Guidelines

Tasks should enable both the verification of the model formulation methods as well as test the effects of specific optimization objectives. The following list categorizes the information we hope to obtain; the general class of motions that fall into that category and other implications on the collection process.

1. Simple tasks for validation of graph-theoretic methods and the derived equations of motion:

(a) Planar Motion: Planar (close to planar) motions can be easily analyzed (i.e. via Lagrange's equation) with simple planar models to validate the GTM model performance.

i. Limit the subject to an upright sitting position

ii. Use supports/restraints to keep sternum and head stationary

2. Three-dimensional taks of motion constraints (Non-planar) :

   (a) Maximum Speed: Optimization should work within system constraints defined by maximal stress (muscular, joint, tissue, etc.) to limit maximum velocity capabilities of the arm.

      i. Task should provide observable synergy or some typical coordination pattern.

   (b) Extreme Flexion/Extension: Similar to maximum speed, violation of possible ranges of motion should not occur.

      i. The assigned task should require extreme postures without prescribing a trajectory.

   (c) Object Avoidance: Verify the object avoidance capabilities.

   (d) Optimal Load Distribution: Moderate/High load at slow/moderate speed should result in the stress criterion being minimized by adequate redistribution

      i. Peak joint and actuator stresses should not be violated.

3. Task reconstruction assumptions:

   (a) All segments behave as rigid bodies and rotate about a fixed point (joint centre) relative to their local frame.

   (b) The mechanical model does not simulate the motion of the fingers, thus tasks should be performed with fingers held in a fist.

5.2.4 Task Descriptions

From the aforementioned guidelines, the following tasks were developed and prescribed to the subject's right arm.

1. *Vertical reach* (*planar*):

   Start from an "at ease" position (arm along side palm inwards) to a maximum vertical reach

2. *Horizontal reach to opposite shoulder (planar)*:

   Start with arm abducted (90°), parallel to floor, to palm touch on opposite (left) acromion

3. *Diagonal reach*:

   Start with the palm on the left hip and move to a target on a shelf directly ahead of, and slightly above, the right shoulder

4. *Diagonal reach in minimum time*:

   Same as 3) but now as fast as possible

5. *Horizontal pass with an obstacle*:

   Start with the hand on a table to the right of a rectangular obstacle and move to a target on the left hand side of the target

6. *Body avoidance*:

   Start with the palm on the left hip (as in 3) to palm touch on a target on the back of the head

7. *Vertical lift*:

   Same as 1) but now with a load in hand

8. *Load placement*:

   Raise a load from the table to a shelf just above shoulder height

9. *Load removal*:

   As in 8) now lowering the load from a shelf to the table

- *The load refererred to in tasks 7,8 and 9 is a 5kg mass.*

5.2.5 Task Collection Protocol

1. Calibration trials:

   (a) Arm relaxed alongside with all markers in view

      i. Identify each marker for defining segment reference frames

1. T6	*2.* T5	*3.* T8	*4.* T1	*5.* T5
6. T3	*7.* T7	*8.* T6	*9.* T9	*10.* T4
11. T3	*12.* T4	*13.* T2	*14.* T5	*15.* T2
16. T4	*17.* T5	*18.* T8	*19.* T9	*20.* T6
21. T2	*22.* T4	*23.* T9	*24.* T1	*25.* T8
26. T7	*27.* T5	*28.* T2	*29.* T9	*30.* T6
31. T7	*32.* T8	*33.* T2	*34.* T1	*35.* T3
36. T1	*37.* T7	*38.* T4	*39.* T3	*40.* T9
41. T6	*42.* T1	*43.* T8	*44.* T7	*45.* T3

Table 5.1: Collection trials and associated tasks

ii. Determine baseline levels of EMG signals

2. Subject preparation before collection with periodic reminders:

   (a) Seated upright with back firmly pressed against the cushioned back rest of the chair

   (b) Keep head stationary throughout the performance of each task

3. Collect five trials for each task and randomize the performance sequence:

   (a) Create a list of 45 trials (9 tasks$\times$5 trials/task)

   (b) Form a random trial collection list (Table 5.1) by randomly selecting tasks from list (a) (without replacement). A random number generator was used to pick both a trial number and task number.

   (c) The purpose of randomizing the tasks is to eliminate the tendency to "consciously control" the action through learning a desirable response. Randomizing will assure that the envelope of movement variability reflects the inherent variability between the subject's performances.

4. End-calibration trials:

   (a) Verify that data derived from initial calibration remain consistent

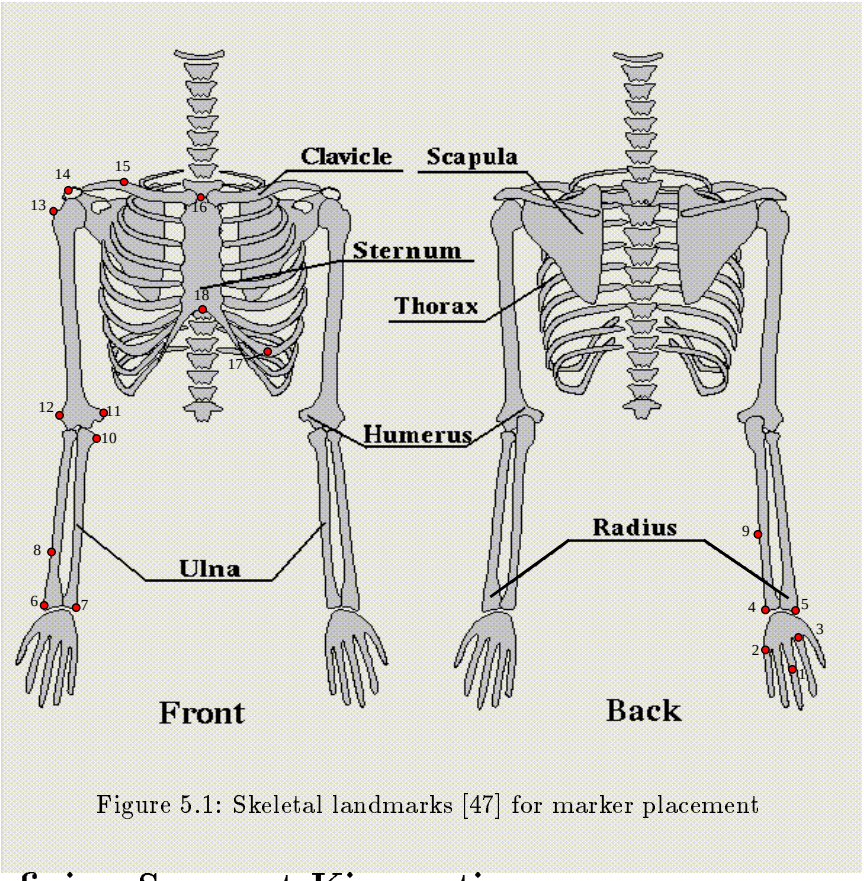

Figure 5.1: Skeletal landmarks [47] for marker placement

5.3 Defining Segment Kinematics

To capture the motions of the arm (minus the scapula), the position and orientation of the segments must be recoverable from the point locations of infra red light emitting diodes (markers) detected by an optoelectronic tracking system (Optotrak^{TM} Northern Digital, Waterloo, Ontario). In general, three non-collinear points are required to identify the location and orientation of any rigid body. From the kinematics of the markers, the relative segment rotations can be determined by assuming these underlying limb segments behave as rigid bodies with idealized rotational joints.

5.3.1 Optotrak Marker^{TM} Placements

Marker placements were selected to minimize contributions of skin and muscle movement, thus majority of the markers are placed on accessible bony landmarks of each segment (Figure 5.1). Eighteen locations were identified for marker placement.

Marker	Anatomical Location
1	Distal end of the middle finger proximal phalanx
2	Head of little finger metacarpal
3	Head of index finger metacarpal
4	Posterior of ulnar styloid process
5	Posterior of radial styloid process
6	Anterior of radial styloid process
7	Anterior of ulnar styloid process
8	Anterior ridge of the radius ($\approx$2/3 proximal to distal distance)
9	Posterior ridge of the ulna ($\approx$1/3 proximal to distal distance)
10	Olecranon (ulna)
11	Medial epicondyle (humerus)
12	Lateral epicondyle (humerus)
13	Greater tubercle (humerus)
14	Acromion (Acromioclavicular junction)
15	Anterior ridge of clavicle ($\approx$1/2 proximal to distal distance)
16	Sternoclavicular junction (on clavicle)
17	Angle of the 7th rib, left side of the thoracic cage
18	Sternum immediately above the xiphoid process

Table 5.2: Marker numbers and their corresponding placement

5.3.2 Segment Definitions

Each segment is defined by at least three points. These points define the motion of a reference frame on each body. The convention used herein is: the origin of the local frame is defined by the first point; the z-axis of each segment frame is defined by an axis along the vector of the second point minus the first, with the y-axis forming the plane in which the third point and the z-axis lie, and finally, the local x-axis being the normal to that plane.

Specifically, each segment is defined by:

1. Sternum = markers {18, 16, 17}
2. Clavicle = markers: {16, 14, 15}
3. Humerus = markers: {13, 12, 11}
4. Radius = markers: {12, 5, 8, 6}
5. Ulna = markers: {10, 7, 9, 4}
6. Hand = markers: {5, 3, 4, 1, 2, 6, 7}
7. (Scapula) = marker {14} and three points on scapulothoracic surface

5.3.3 Optotrak^{TM} System Setup

The Gait and Posture Laboratory was used for the movement collection. The Optotrak^{TM} system was used with three cameras to ensure that a sufficient number of markers were in view at each instant (time frame), in order to define the motion of each segment. For eighteen markers the maximum sampling frequency obtainable was *80Hz* which is sufficient for the majority of the tasks that are of relatively low frequency.

5.3.4 Formulating Segment Kinematics

The nature of the collection is such that several markers disappear and reappear during movement due to limb and workspace elements temporarily blocking the view of the cameras. For this reason, redundant markers were used to identify segments prone to "losing" markers (like the

hand). Leveraging the additional information to provide the best representation of the individual segment kinematics is of primary concern.

Describing the kinematics of the segments by the individual marker movements does not encapsulate movement information in an efficient manner and it does not directly compare to predictive model results. For example, tracking a single marker on the hand does not effectively describe the motion strategy, and when the marker is out of view it does not provide any information. The obvious remedy of analyzing more markers requires more data to be maintained and does not completely overcome the problem of hidden markers since more computational methods are necessary to incorporate and/or switch to available markers. The ideal representation is the relative rotation of each segment which requires only a single time varying matrix to describe a segment's motion rather than N ($\geq$3) marker coordinates that have three components each. These relative rotations can be resolved into the relative time varying angles (three angles describing a segment's orientation) which is considerably easier to analyze and compare than the 21 trajectories (x,y,z components) of the hand segment's seven markers! Relative rotations are possible because each segment is constrained by a joints or joints that only allow for relative rotation (or can be approximated by rotations only) when the scapula is not included.

The method employed herein determines the segment's local frame, which is the set of segment fixed local vectors that define marker locations within the owning segment's reference frame. This is done by using the calibration trials, which have all markers in view, to determine the location and orientation of each segment's local reference frame and then determining the fixed position vectors of each segment marker in that segment's frame. These frames thus represent a local mapping of each segment which can now be used in conjunction with any set of markers in view to determine the "best" rotation that satisfies both the pre-processed local definition and the observed global marker positions.

Segment Reference Frames

The first step in determining the relative motion of each segment is determining the postion and orientation (local rotation) of each segment from the calibration trials. The local rotation of each segment is defined as the matrix ($\Re$) that can transform any vector from the global frame to its equivalent representation in a local frame. Any three non-collinear points on a segment can be used to define a local frame and thus provide the resulting transformation matrix. Using

the previously defined convention (Section 5.3.2), the local rotation is simply the matrix that transforms the local frame unit vectors $\widehat{i}$, $\widehat{j}$, $\widehat{k}$ (columns, corresponding to the direction of the segment's **x**, **y**, **z** axes) defined in global coordinates, to the local frame representation which, by definition, is an identity matrix.

$$[\Re][\,\widehat{i}\;\widehat{j}\;\widehat{k}\,] = [\,\widehat{i'}\;\;\widehat{j'}\;\;\widehat{k'}\,] = [I] \tag{5.1}$$

therefore,

$$[\Re] = [\,\widehat{i}\;\widehat{j}\;\widehat{k}\,]^T \tag{5.2}$$

The unit vectors of the local frame in global coordinates are calculated based on the first three marker locations of a given segment's definition. Thus, if the first three marker positions correspond to p_1, p_2 and p_3, the $\widehat{i}$, $\widehat{j}$, $\widehat{k}$ unit vectors are determined by:

1. p_1, the origin of the segment's (local) frame
2. $\overrightarrow{k} = p_2 - p_1$, the vector along the z-direction of the segment
3. $\overrightarrow{v} = p_3 - p_1$, the vector locating the third point defining the segment plane
4. $\overrightarrow{i} = \overrightarrow{v} \times \overrightarrow{k}$, the normal vector to the segment plane and the segment's x-direction
5. $\overrightarrow{j} = \overrightarrow{k} \times \overrightarrow{i}$, the vector in the segment's y-direction
6. $\widehat{i} = \dfrac{\overrightarrow{i}}{\left\| \overrightarrow{i} \right\|}$, the normalized unit vector, similarly for $\widehat{j}$and $\widehat{k}$

The accuracy of this evaluation is dependent on the accuracy of the vectors used in the cross products. Since for longer vectors, the marker position noise (such as skin movement) have less effect in proportion to shorter vectors, using longer vectors in the cross products acts to preserve segment movement information. Thus, the defined convention deliberately uses the points that define the longest vectors on the body. As well, normalization only occurs after evaluating the cross products.

Unfortunately, this method cannot be used to determine segment rotations for trials other than the calibration trials, because they are the only trials that guarantee that the necessary markers will remain in view for the entire trial period. In the task trials, any marker can be obscured at any instant and therefore the evaluation of the rotation matrix by this method is not sufficient. An alternate method is to describe marker data in both local and global representations and then to determine the necessary rotation (transformation) matrix. However, to define the local frame the orientation of the segment must be known at a given instant. For this purpose, the above method is used on the calibration trials to determine the local segment frames.

Local Segment Frames

The next step is resolving each marker into its owning segment's (or segments') fixed local frame. The memberships of markers to segments are described in the segment definitions (Section 5.3.2). Having established the origin and the local rotation for calibration trials, each member marker can now be resolved into a local vector. The global marker positions are resolved into global vectors originating at the origin of the local reference frame, and are then mapped into the local frame by applying the local rotation, $\Re$. Obviously, the marker defining the segment's origin maps to $\{0,0,0\}^T$, and, according to the defining convention, the second defining point lies on the local z-axis and the third point lies in the local yz-plane. Formally these are computed by:

1. $\overrightarrow{v_i} = p_i - o_s$, global vector from the origin of segment s to its i^{th} marker position
2. $\overrightarrow{l_i} = [\Re]\,\overrightarrow{v_i}$, the local representation of the vector
3. computing the average local vector representation over all instances (N_T time frames) collected in the calibration trials,
$$\overrightarrow{\underline{l_i}} = \frac{1}{N_T}\sum_{n=1}^{N_T} l_{i,n} \tag{5.3}$$
4. yields, the local frame L for segment s with m markers is,
$$[L_s] = [\,\overrightarrow{\underline{l_1}}\ \overrightarrow{\underline{l_2}}\ \ldots\ \overrightarrow{\underline{l_m}}\,] \tag{5.4}$$
where L_s is a matrix containing the fixed local coordinates of the segment's markers as column vectors.

"Best" Local Rotation

The local frame defined in (5.4) now enables any three or more markers in view to determine the orientation of the segment. In particular, when more than three markers are in view, the following method was developed to include the additional markers in order to determine the "best" local rotation matrix, $\Re_B$. Best is used tentatively, because it simply refers to the solution that minimizes the sum of squared differences of local vectors transformed to global vectors and the global vectors obtained directly from the Optotrak markers. In this framework, the more markers available the more rigorous the fitting process and the more accurate the resulting rotation matrix should be. Formally, at each instant the following procedure is followed:

1. For each segment, s, determine the set of global markers in view, M
2. If the number of markers in view, $m \geq 3$, then:
   (a) Select these markers from L_s and form a local frame of visible markers, L
   (b) Generate a set of interconnecting fixed segment vectors from M
      i. $MM_i = M_j - M_k$, for $j \neq k$
      ii. results in $p = (m-1) + (m-2) + \ldots + (m-m)$, column vectors
   (c) Similarly, formulate the local frame vectors, LL
   (d) Find $\Re_B$ such that the function,
   $$J = \sum_{i=1}^{p} \left\{ \left[[MM_i] - [\Re_B]^T [LL_i] \right] \cdot \left[[MM_i] - [\Re_B]^T [LL_i] \right]^T \right\} \tag{5.5}$$
   is minimized. with the constraint $[\Re_B][\Re_B]^T = [I]$to ensure $\Re_B$ is an orthonormal rotation matrix.
   (e) $\Re = \Re_B$, define the local rotation of this segment as the matrix resulting from least squares solution
3. Otherwise, identify the rotation matrix at this instant to be undefined

Centres of Rotation

Thus far we have have described how to obtain the local rotation matrix, $\Re$, of each segment's frame with respect to a global frame, from Optotrak system marker positions. Although it is true

that the rotational motion of a segment can be described by these matrices, we have not indicated about what point in space the rotations are occurring. The local reference frames, thus far, do not identify the point about which the rotation occurs. Using the origin of these local frames is often a suitable approximation; however, for segments, such as the humerus, the distance of the origin-marker (on the skin surface above the greater tubercle) to the actual point of rotation can be several centimeters and this positioning error is often further amplified, by a segment's length, for greater errors at the distal end. To maintain the above definition of the segment-reference frames and their resulting rotational information, a method was devised to identify the joint centres rather than settling for the closest marker positions.

In idealized multibody dynamics, the rotational joint centre is a point belonging to both bodies and thus does not move relative to each segment's local reference frame. Of course this is only true if the joint only allows for pure rotation, which is a simplifying assumption that was made earlier for all upper limb joints except for the scapulothoracic joint. Under this assumption, a good approximation of the joint-centre can be found by determining two body-fixed vectors (one affixed to each segment coincident on the joint-centre) that minimize the differences in the global joint-centre location transformed from each body's local definition, over all time.

Thus, given two adjacent local frames with their origins o_{i-1} and o_i, with rotations $\Re_{i-1}$ and $\Re_i$, the global joint-centre location $\overrightarrow{C}_{jc}$, can be determined by either of two local fixed vectors, $\overrightarrow{c}_{i-1}$ and $\overrightarrow{c}_i$ with respect to their local reference frames, such that:

$$\overrightarrow{C}_{jc} \;=\; o_{i-1} + [\Re_{i-1}]^T \cdot \overrightarrow{c}_{i-1} \;=\; o_i + [\Re_i]^T \cdot \overrightarrow{c}_i \tag{5.6}$$

thus the solution for $\overrightarrow{c}_{i-1}$and $\overrightarrow{c}_i$ is that which minimizes the objective:

$$J = \sum_{n=1}^{\aleph} \left\{ \left[(o_{i-1} + [\Re_{i-1}]^T \cdot \overrightarrow{c}_{i-1}) - (o_i + [\Re_i]^T \cdot \overrightarrow{c}_i) \right]^2 \right\}_n \tag{5.7}$$

This result is the least squares approximation of the joint-centre locations. Here, $\aleph$ is the total number of frames over <u>all</u> trials, since this is the best approximation of the limb movements for all time. This expression can be reduced into a simple linear regression equation of form $y = X\beta + \upsilon$, by reordering the equation.

$$o_{i-1} - o_i \;=\; -[\Re_{i-1}]^T \cdot \overrightarrow{c}_{i-1} + [\Re_i]^T \cdot \overrightarrow{c}_i \tag{5.8}$$

then,

$$\underbrace{(o_{i-1} - o_i)}_{y_{(3N\times 1)}} = \underbrace{\left[-[\Re_{i-1}]^T \;\; [\Re_i]^T\right]}_{X_{(3N\times 6)}} \cdot \underbrace{\left\{ \begin{array}{c} \overrightarrow{c}_{i-1} \\ \overrightarrow{c}_i \end{array} \right\}}_{\beta_{(6\times 1)}} + v_{(3N\times 1)} \tag{5.9}$$

In this context the linear "fitting" parameters β contain the resulting local vectors $\overrightarrow{c}$, for two adjacent segments. This now provides the joint-centre locations to be used as fixed parameters for a given subject and are added to the local segment definition which identifies the joint-centre "virtual" marker position. The residual, v, indicates the error in the fitting which is the distance of the joint centre postion between the transformation of the two adjacent segments.

Relative Rotations, R_i

We define the relative rotation R_i as the rotation from a proximal frame ($i^{th} - 1$ segment) to a desired adjacent and distal frame (the i^{th} segment). Knowing the local rotation, $\Re$, of each segment, the relative rotation of each segment can be determined as follows:

From (5.1),

$$[\Re_{i-1}] \cdot \overrightarrow{a} = \overrightarrow{a}'$$

$$[\Re_i] \cdot \overrightarrow{a} = \overrightarrow{a}''$$

are both representations of the global vector $\overrightarrow{a}$ in two adjacent segment frames. Which can now be written as:

$$[R_i][\Re_{i-1}] \cdot \overrightarrow{a} = \overrightarrow{a}'', \quad or$$

$$[R_i][\Re_{i-1}] \cdot \overrightarrow{a} = [\Re_i] \cdot \overrightarrow{a} \tag{5.10}$$

Then by dotting each side by $\overrightarrow{a}$ and dividing by $\|\overrightarrow{a}\|^2$, yields

$$[R_i] = [\Re_i][\Re_{i-1}]^T \tag{5.11}$$

Thus, using the sternum as the most proximal segment (the beginning of the kinematic chain), its relative rotation is equivalent to its local rotation, since $\Re_0$ represents the local rotation of the global reference which is simply the identity matrix. The subsequent relative rotations describing the movement of each segment can now be similarly computed using equation (5.11).

With the local segment frames defined including the joint centre locations, the entire kinematics of the upper limb can be replayed using the relative rotations, R_i, to drive the motion of each segment. Furthermore, the relative rotation matrices can be decomposed into an angular representation, such as Euler angles, to produce the angular trajectories for analysis and comparison purposes.

5.4 Electromyography (EMG) Acquisition

Due to complications encountered with the EMG electrode leads to a battery pack and its interface with a sixteen channel bio-amplifier, only eight channels could be collected using two portable EMG bio-amplifiers. Being limited to eight channels, the muscles selected were based on their capacity to produce detectable signals representing unique motions. For example, the trapezius was neglected because, although it provides considerable activity, the activity of the trapezius does not vary significantly with the variety of movements since it is a major stabilizer of the scapula for most upper limb movements.

5.4.1 EMG Electrode Placement

The following is the list of muscles that were selected for EMG electrode placement:

1. Teres major

   A specialized back muscle that adducts posteriorly and medially rotates the humerus and assists to extend the flexed arm[1]

2. Deltoideus

   The functional fibres are easily accessible on the surface and highly specialized

   (a) Anterior fibres

   Strong flexor and medial rotator of the humerus

(b) Middle fibres

Principle abductor of the arm at the glenohumeral joint

(c) Posterior fibres

Strong extensor and lateral rotator of the humerus

3. Pectoralis Major

Powerful anterior adductor and medial rotator of the humerus at the shoulder

4. Biceps Brachii

Primary flexor of the elbow joint and powerful supinator of the forearm (radius rotation about the ulna)

5. Triceps Brachii

Primary extensor of the elbow joint

6. Brachioradialis

A flexor of the elbow joint and chief elbow joint stabilizer

These muscles (Figure 5.2) are quite accessible and represent the larger actuators in the system as indicated. As per the experimental objectives, the important features from the EMG data are the firing patterns and general level of activation. Thus, data will be recorded in raw A/D units or microvolts as opposed to a percentage of maximum voluntary contraction (%MVC), which would require establishing a protocol for measuring the MVC for all the aforementioned muscles.

5.4.2 EMG Setup

The bellies of the aforementioned muscles were found via palpation and two electrodes were placed with each being perpendicular to the muscle fibres and directly adjacent to its partner along the line of the muscle (as a differential pair). For each of the two bio-amplifiers one lead also contained a ground electrode which was placed at the top of the spine for the amplifier containing the teres

[1] Teres major along with the latimus dorsi and pectoralis major muscles are major climbing muscles which lift the trunk when the arms are fixed. These muscles in conjunction are also major stabilizers of the shoulder joint [57].

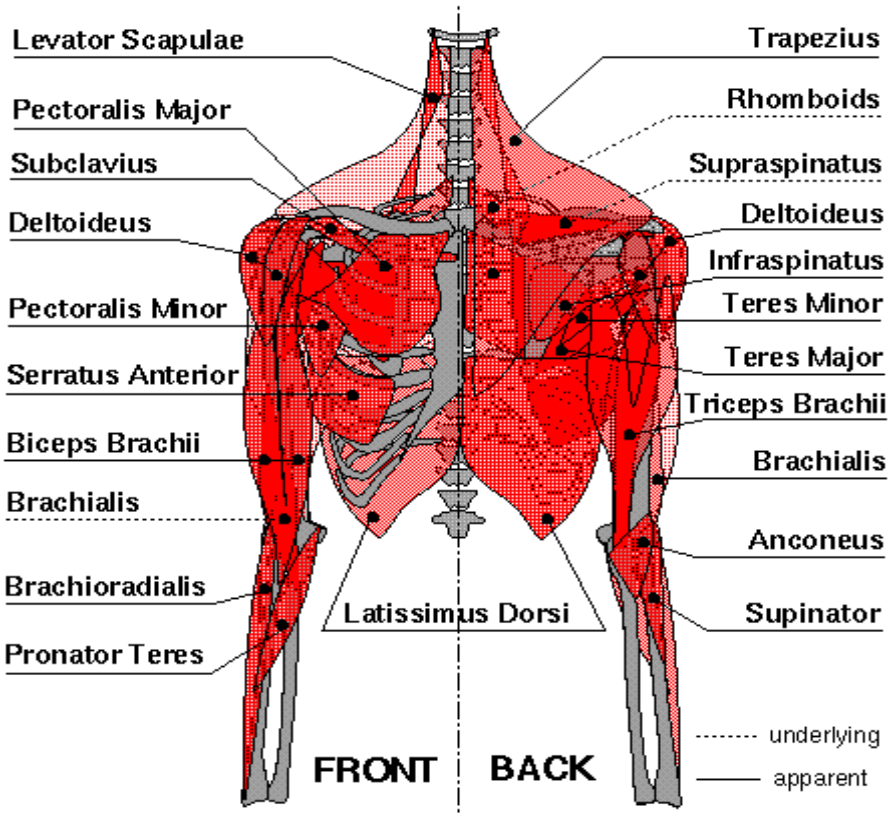

Figure 5.2: Musculature of the human upper limb [47]

major and deltoideus muscles, and on the olecranon for the amplifier containing the pectoralis major, biceps brachii, triceps brachii and brachioradialis muscles. These grounding sites were selected because they are are large and bony sites upon which an electrode can be placed whilst minimizing the voltage contribution arising from surrounding muscles. The ground appears in each differential pair and thus acts to subtract background noise. The gains of each channel were adjusted such that most submaximal movements involving the particular muscle would occupy about half of the available discretization range. The analog to digital sampling frequency was set to *2400Hz,* an integer multiple of the Optotrak sampling frequency, so that motion and EMG data could be easily aligned.

5.4.3 Forming Muscle Activation Signals

In the comparative study, the biomechanical model has fewer actuators than the human upper limb. Therefore, it is unlikely that model actuator forces would compare well with %MVC (therefore not measured) as an indicator of actuator output. However, the temporal features of the selected muscle activation signals should provide a basis for comparison of actuators with similar function. Thus, the temporal changes of relative levels of activation are of primary importance for comparison with predictive control signals. In addition, the actual activation levels in %MVC are considerably more difficult to acquire since a protocol for evaluating the MVC of each muscle would be required. Thus, for the time being, the processing of the collected EMG signals is quite simple.

1. Rectify each signal by taking the absolute value of every point of the EMG waveform

2. Low pass filter the signals with a second order bidirectional discrete Butterworth filter with a $10Hz$ cut-off frequency.

   The movements are anticipated to be between 0-$5Hz$ (individual movements lasting atleast $0.2s$) while backgound noise is anticipated to be from electrical devices operating at much higher freqencies, e.g. $60Hz$ light bulbs. Thus, $10Hz$ ensures to capture all the movement features but eliminating the high frequency content. The bidirectional filter ensures the phase shift introduced by the second order difference equation employed by the filter does not artificially introduce lag.

The result is a set of signals that represent a smooth time varying activation. This allows several muscles to be observed in parallel (over the same trial and time scale) to establish temporal activation patterns that may shed light on coordinative strategies which may or may not be predicted by the optimization driven model.

5.5 Collection

Collection was performed on Saturday November 7th, 1998 between 9:30 a.m. and 2:30 p.m. A young male athlete with no history of neuromuscular disorders was selected as our subject with his written consent obtained prior to collection. The experimental protocol was in accordance with, and received the approval from, the Office of Human Research at the University of Waterloo.

5.5.1 Laboratory Setup

The Gait and Posture laboratory was arranged with three Optotrak^{TM} cameras, a table, a chair and an obstacle such that the subject could be seated comfortably and interact with the workspace in order to perform the prescribed tasks (Section 5.2.4).

Figure 5.3 is the top view (from positive to negative Y-axis) of the experimental setup. The corresponding heights of the table and obstacle are described herein. The global reference origin is, in fact, elevated off the floor ($34cm$). Thus, in global coordinates the table top lies at $40.2cm$ in the positive Y-direction. The top of the obstacle stands $23.5cm$ above the table top (or $63.7cm$ Y from the origin). The shelf used in the load placement was suspended during data collection. However, its location was not recorded because it was designed specifically to act as a visual target for the placement (the final or initial height) of the hand which can be determined directly from hand-marker data (described in Figure 5.1). The suspended shelf was adjusted to provide the subject with an appropriate target height within their reaching capacity without requiring the movement of the sternum.

5.5.2 Subject Selection Criteria

The following criteria were used in the selection of our test subject.

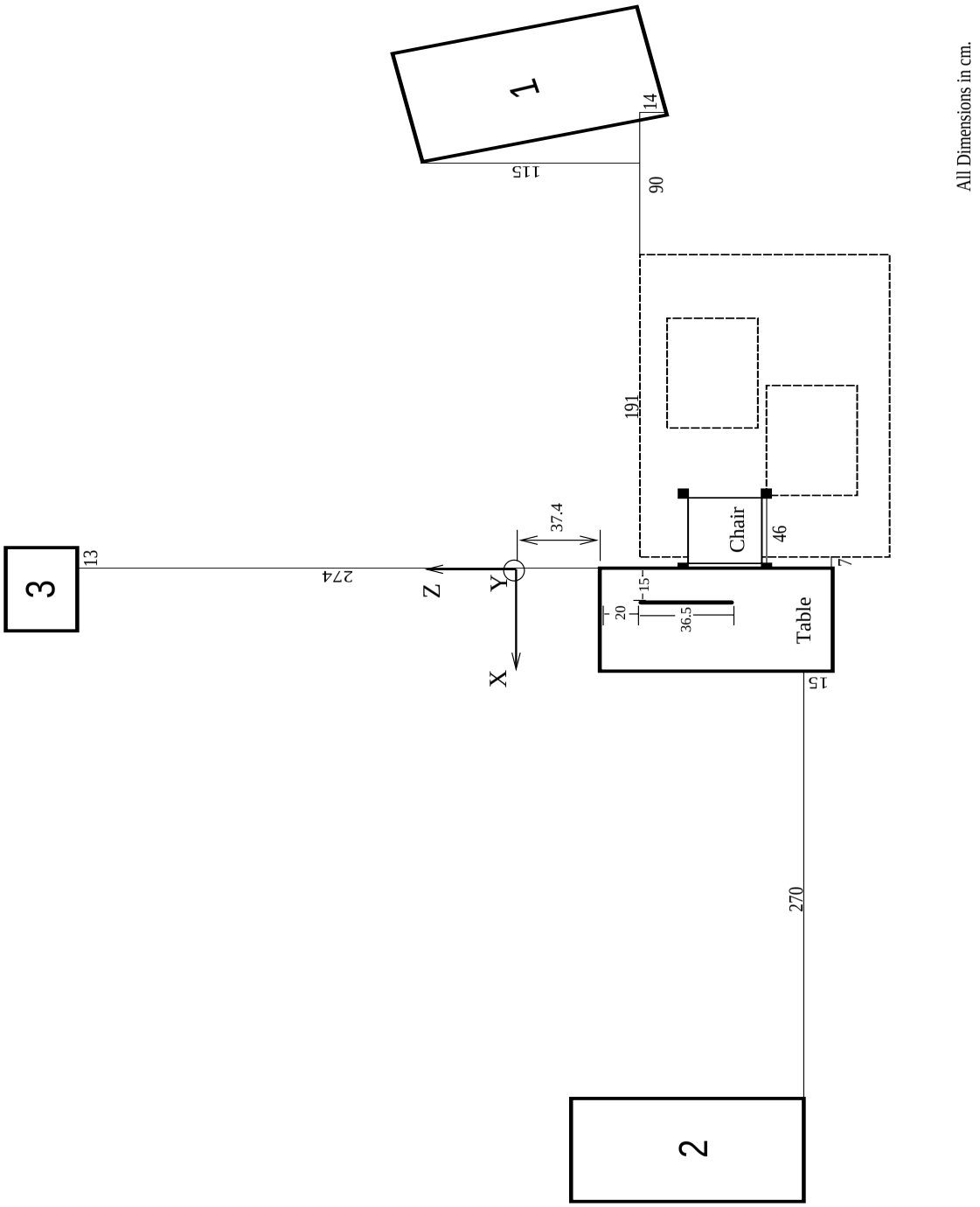

Figure 5.3: Camera and workspace layout

1. A fit volunteer with his/her consent
   (a) Fit individuals generally have better muscle development
   (b) Male was selected because of ease of palpation for EMG electrode placement
2. No known (or past history of a) neuromuscular disorder
3. Minimal subcutaneous fat and body hair
   (a) Enables accurate detection of segmental landmarks and less skin marker distortion
   (b) Capable of higher voltage detection of myoelectric signals by the EMG electrodes
   (c) Subject must agree to shave his right arm prior to collection

5.6 Results and Statistics

The aforementioned specifications and collection methods were used to determine anthropometry, three dimensional kinematics, and muscle activation patterns from the selected subject.

5.6.1 Subject Anthropometry

Direct Measurements

The following measurements were taken directly from the subject using a measuring tape and a scale.

Derived Measurements

Segment definitions were constructed and used to compute the rotation of each segment. Figure 5.4 depicts the resulting generation of the segment (or local) reference frames from the calibration data.

It is important to note that relative marker movement for some segments are quite high. In particular, the humerus segment and especially the distance between its markers that are furthest

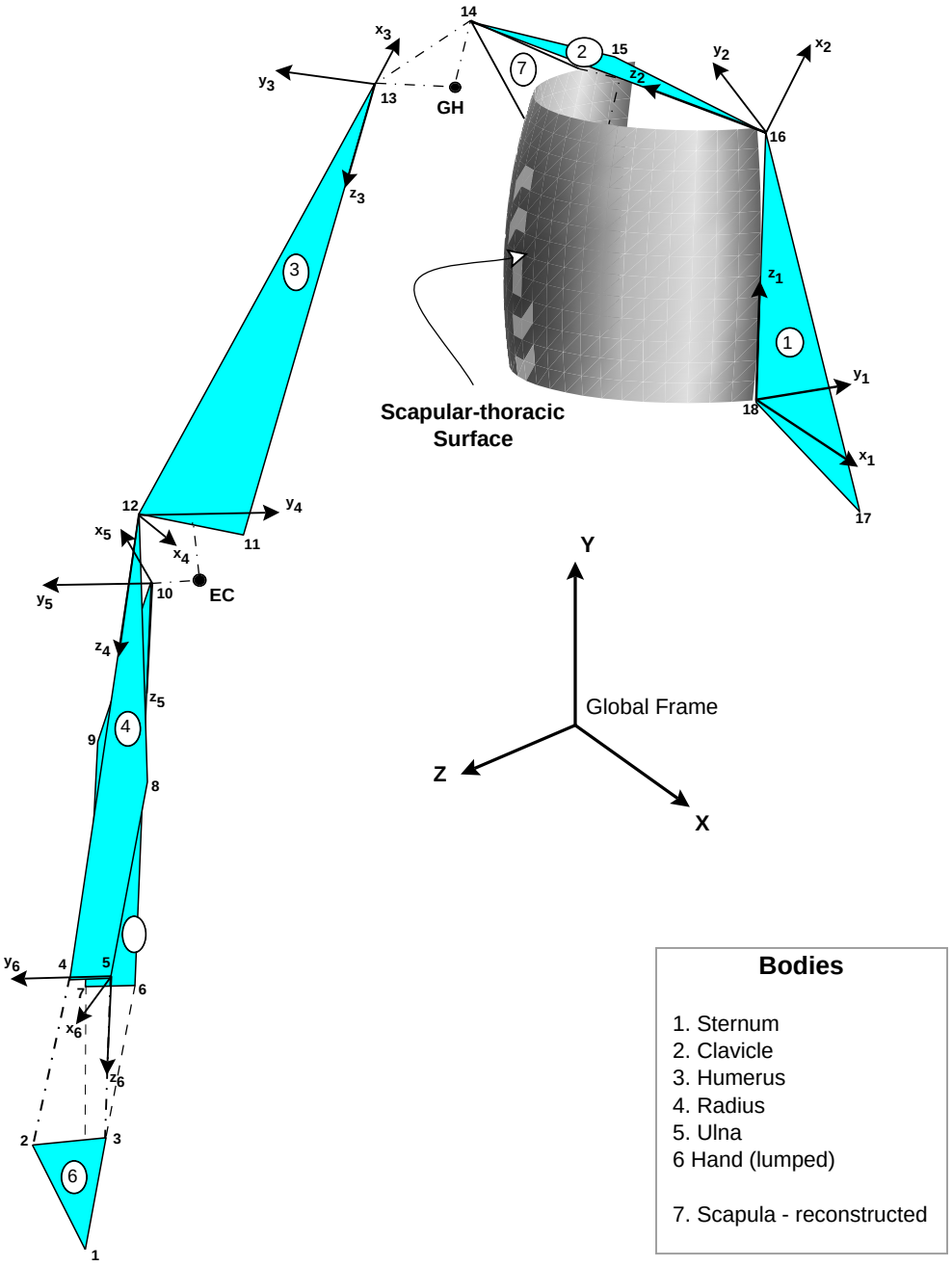

Figure 5.4: Upper limb segment frames defined by Optotrak markers

The segment dimensions, depicted above, use the mean segment inter-marker distances for the markers that define the respective segments (Section 5.3.2). The vertices are numbered to correspond with the marker number.

Anthropometric Measure	Value
Height	$177\,cm$
Total Body Mass	$73.1kg$
Head circumference	$53\,cm$
Distance of top of head from sternum (*marker* 16)	$37.5\,cm$
Chest girth (at xiphoid elevation)	$90\,cm$
Length of the spine of the scapula (starting at the acromion)	$14\,cm$

Table 5.3: Anthropometric measurements from the test subject

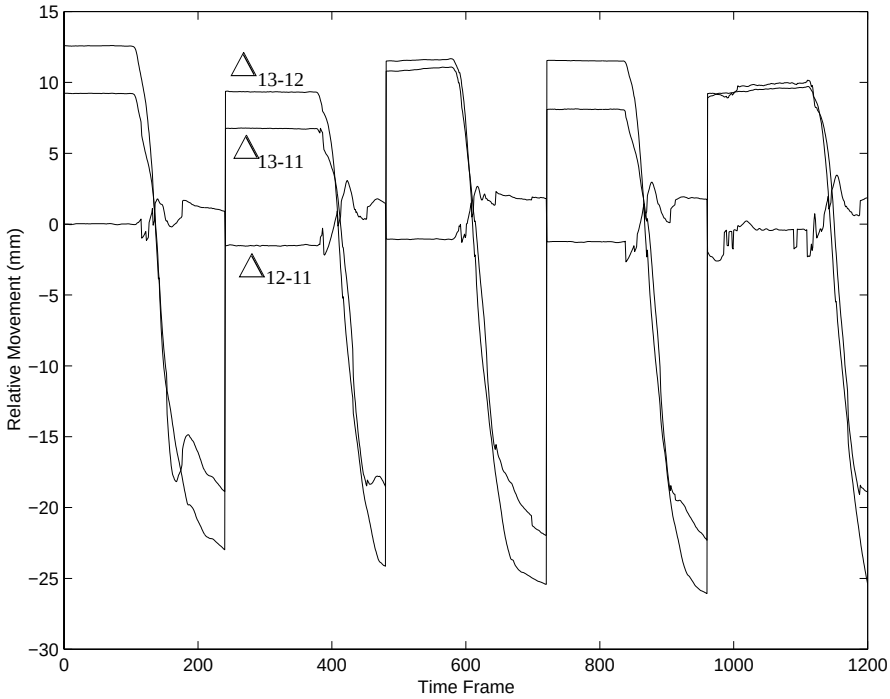

Figure 5.5: Movement of humerus segment markers over all trials of Task 6

This figure depicts the relative movement between markers (11, 12, and 13) that define the humerus segment (Section 5.3.2). The traces Δ_{13-12}, Δ_{13-11}, Δ_{12-11} are the deviations from mean distance of the identified markers at each frame over all five trials (and thus the periodic form).

Segment Measure	Length
Sternum (sternum origin, marker 18, to sternoclavicular joint centre)	$15.07\,cm$
Clavicle (sternoclavicular joint centre to glenohumeral joint centre)	$17.00\,cm$
Humerus (glenohumeral joint centre to elbow joint centre)	$32.57\,cm$
Forearm (elbow joint centre to wrist joint centre)	$28.45\,cm$
Hand (wrist joint centre to distal hand location, marker 1)	$10.06.cm$

Table 5.4: Derived segment lengths

apart, are greatly affected by the movement of the marker above the greater tubercle due to the skin and muscle movement which cover the greater tubercle (Fig 5.5).

Quantitatively, the average distance between the greater tubercle (marker 13) and the lateral epicondyle (marker 12) is $285.01mm$ while the standard deviation is $14.10mm$ with largest variation (max-min) being $38.68mm$. Thus, in reconstructing the segment kinematics, these variations will lead to errors in the computation of segmental rotations and joint centre locations which assumes the markers are affixed rigidly to the body segments. In light of the large amount of marker movement and the potentially problematic collinearity of several of the markers on both the ulna and radius segments, large errors are likely. Therefore the markers from the ulna and radius were lumped into a single body, for the time being, to form the forearm segment.

The resulting segment lengths were determined by taking the magnitude of the vector joining the joint centres (Section 5.3.4) of each segment or, in the case of the sternum and hand, between a joint centre and the origin and the most distal marker, respectively. These parameters were used to define the link segment lengths for the predictive model (Section 4.2.4).

5.6.2 Segment Kinematics

The methods used to derive the relative rotations of all the segments (Section 5.3.4) were encoded in MATLAB (Appendix B.2.1) to determine the joint centres and generate the relative rotation matrices for each segment for all task trials. Each trial was then reconstructed to produce an animation to visually validate the numerical results. Figure 5.6 depicts the resulting animation of trial 1, from task 6 (Section 5.2.4), with the "$*$" trajectory describing the Optotrak trajectory obtained from marker1 .

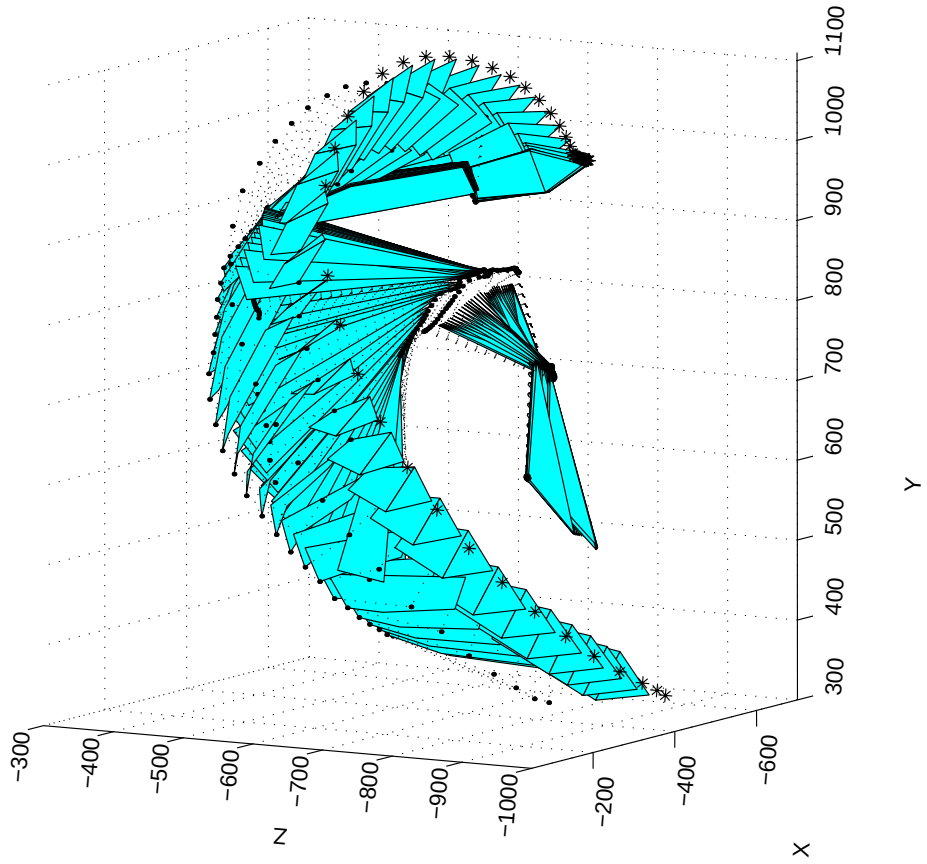

Figure 5.6: Reconstructed motion for Task 6

This animation graphically portrays the performance of the calculated relative rotations (R_i) in reconstructing the movement of Task 6, with the right hand (triangular segment at the end of the kinematic chain) starting at the left hip and ending behind the subject's head. The tip of the triangular hand segment represents the reconstructed location of marker 1, located at the distal end of the middle finger's proximal phalanx. The '*' trajectory represents the actual trajectory of marker 1. The Y-axis is the vertical axis and the positive X-axis is the forward direction.

For comparison purposes, the marker positions for trials of the same task were temporally aligned by observing the start and end of motions from the marker trajectories of the hand. This was performed by identifying the points in time at which the slope of the movement surpassed a small threshold (start) and then fell below the threshold (end) for more than five continuous time frames. These start and end times were then used to "clip" the relative rotation trajectories which were then normalized in time and scaled to the mean performance time.

The derived segment models (segment frames) were driven by the derived relative rotation matrices (R_i) and the trajectories of key marker positions (markers 1, 5, 12) averaged over the task trials were compared to the actual averaged and standard deviation of the unprocessed marker positions. Figure 5.7 illustrates the comparison for task 6 which had a mean performance time of $1.21s$.

For the majority of the duration of the task, the average rotation driven markers follow the average actual marker trajectory quite closely, but towards the end of the task, the reconstructed marker trajectories are close to the standard deviation boundary for markers 5 and 12, and almost $3.5cm$ off the mean for the motion of marker 1. The residual differences between between the actual and reconstructed averages (Figure 5.8) provides a clearer indication of the magnitude of their differences.

These errors can be explained by the aforementioned relative marker movement (Figure 5.5) since the segment frame definitions assume that the marker locations are fixed with respect to the segment's frame. In addition, larger errors result at the "end effector", because marker 1 and the fingers, incidentally, may not remain equidistant to the wrist (i.e. slight flexion and extension of the fingers), thus the fixed locations defining the hand are violated. Thus, the least squares method can do little to eliminate this error due to the assumption of fixed segment marker coordinates within the segment frame.

The relative rotations for each trial were decomposed into three sequential angles of rotation either by Euler angles or by an X and Y space fixed (first body's axes) rotations and a body fixed (second body's axes) local z rotation, for the purposes of presenting angular information rather than rotation matrices, which are difficult to interpret. An alternative to the Euler angles was sought in order to provide a better anatomical interpretation of the angular trajectories. However, since the coordinate frames of each segment neither coincide with the joint location nor align with the axes joining subsequent joint-frames, the angular information does not have

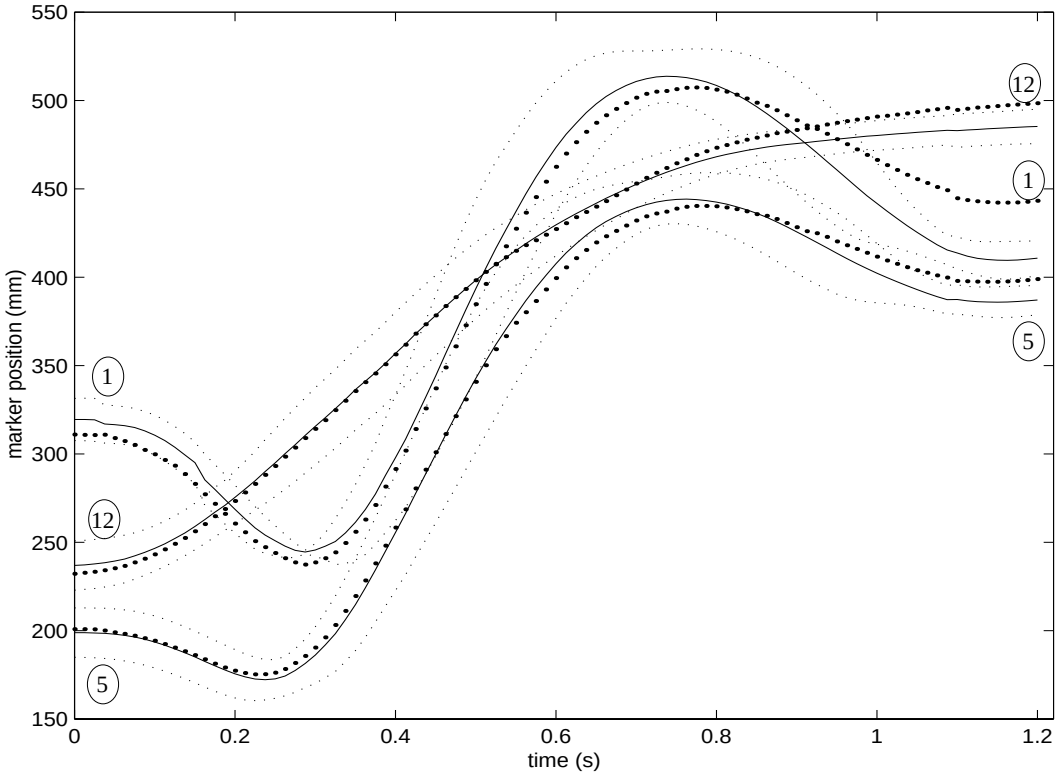

Figure 5.7: Reconstructed and actual marker trajectories (markers 1, 5, 12)

The labels 1, 5, and 12 identify the positions of markers 1 (tip of hand) , 5 (posterior of radial styloid), and 12 (lateral epicondyle) in terms of their distance from the sternum origin while performing Task 6. The solid lines depict the mean marker positions while the fine dotted lines indicate a standard deviation boundary about the mean determined over the five trials of Task 6. The bold dotted lines are the average of the reconstructed positions from the derived relative rotations.

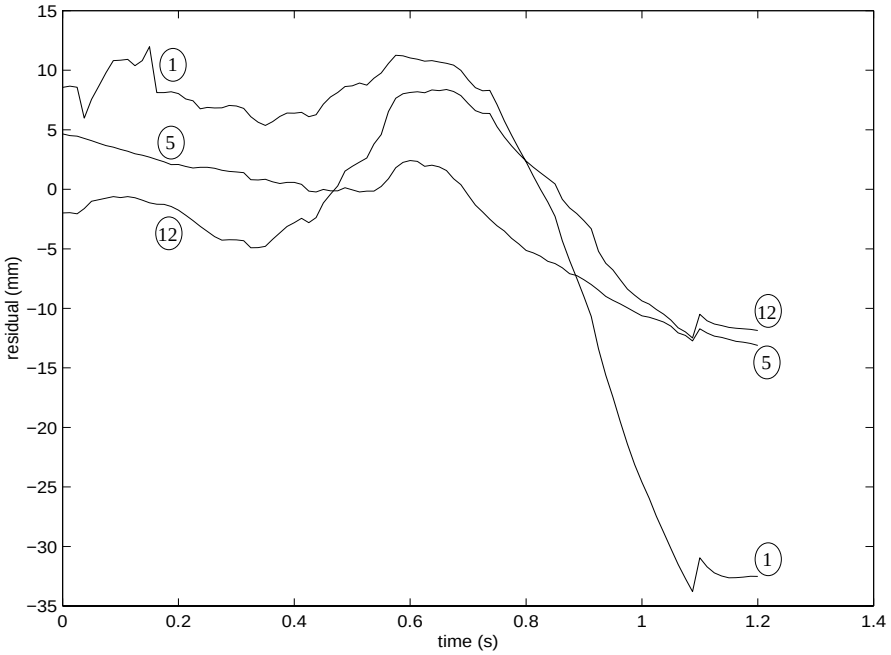

Figure 5.8: Mean actual - mean reconstructed marker (1, 5, 12) trajectories

The curves numbered 1, 5, and 12 represent the residual values of the mean actual position minus the average reconstructed position for the corresponding markers as described in Figure 5.8. Note, the largest errors occur at the end of the task and coincide with the phase of the movement where the deviation of the humerus length from its mean length is also the largest, Figure 5.5 (approaching the end of each trial).

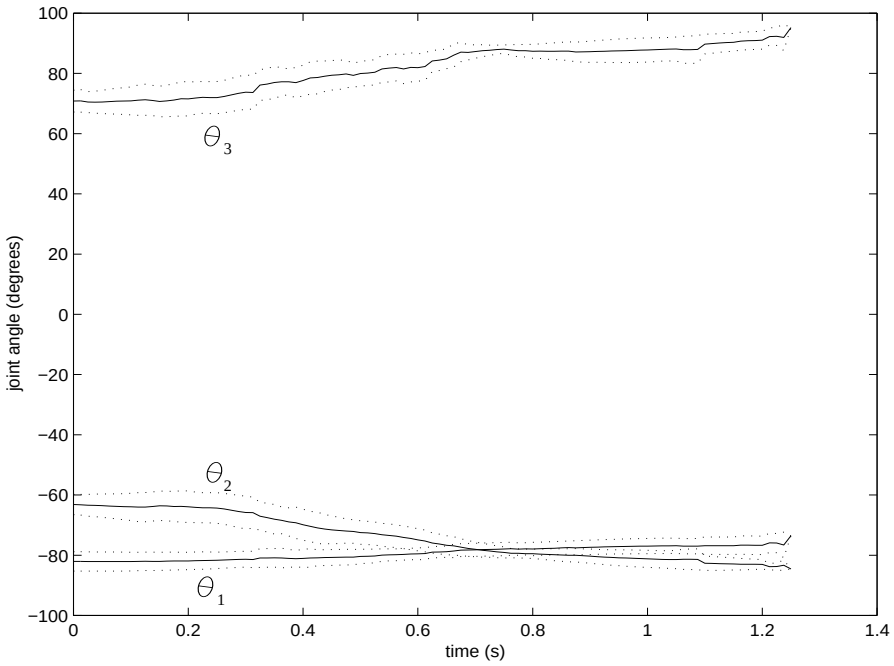

Figure 5.9: Sternum orientation relative to ground

The relative rotation, for Task 6, of the sternum relative to the ground (global coordinate) frame was decomposed into the angles θ_1 (about global X), θ_2(global Y) and θ_3 (sternum z). The solid line represents the mean angular trajectory over all trials of Task 6, while the dotted line represents a standard deviation upper and lower bound.

obvious anatomical interpretations.

Across trials for the same task, these angular trajectories were averaged to form prototypical angular trajectories in performing a particular task. The following figures describe the prototypical motion and their variability for each segment.

In Figure 5.9, θ_1 determines a rotation about the global X-axis, θ_2 about the global Y-axis, and θ_3 is the spin angle about the local z-axis. The first rotation ($\approx$-80^o) about X remains relatively constant rotating the local z-axis such that it is almost coincident on the global Y and resulting in the two subsequent rotations being about two very similar axes. Since the rotations are almost equal in magnitude and opposite in direction, this suggests that they tend to cancel out to form very small net rotation about z.

The relative rotation angles of the clavicle with respect to the sternum (Figure 5.10) was determined by computing the Euler angles, which is the sequence of rotations about a space fixed

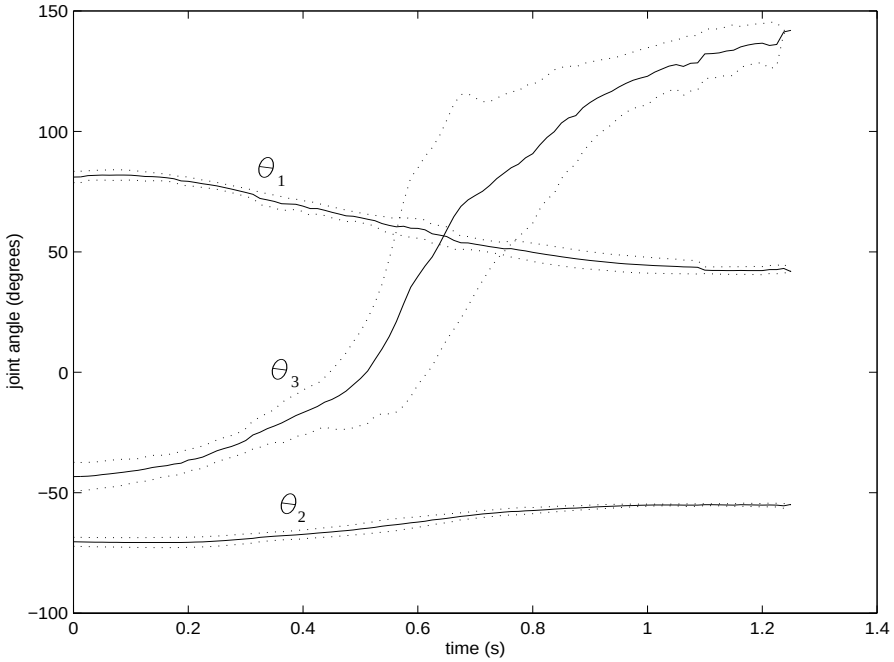

Figure 5.10: Clavicle rotation relative to the sternum

The relative rotation, for Task 6, of the clavicle relative to the sternum frame was decomposed into Euler angles: θ_1 (sternum Z), θ_2(clavicle y) and θ_3 (clavicle z). The solid line represents the mean angular trajectory over all trials of Task 6, while the dotted line represents a standard deviation upper and lower bound.

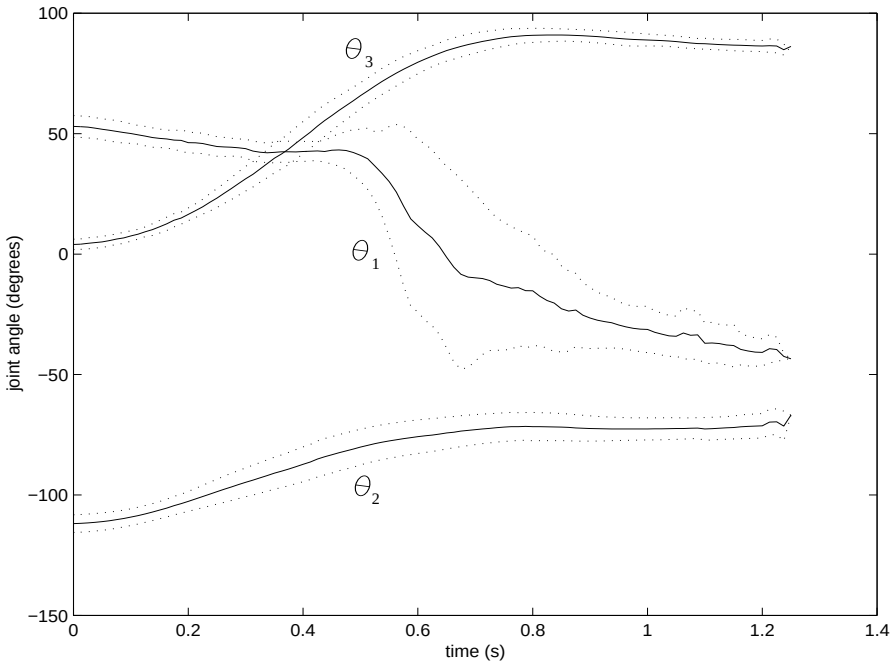

Figure 5.11: Humerus rotation relative to the clavicle

The relative rotation, for Task 6, of the humerus relative to the clavicle frame was decomposed into Euler angles: θ_1 (clavicle Z), θ_2 (humerus y) and θ_3 (humerus z). The solid line represents the mean angular trajectory over all trials of Task 6, while the dotted line represents a standard deviation upper and lower bound.

Z-axis (the sternum's long axis) then about the new body fixed y-axis (on clavicle) and finally about the local z-axis. The first rotation aligns the respective y axes such that the larger the angle the more protracted the clavicle and the smaller the more retracted, with 60^o being about midway between both extremes. The local y rotation now aligns the respective z-axes which is now an indication of the elevation of the clavicle. The final angle is the axial rotation (about its long z-axis) and does not in fact go through the range of motion indicated in Figure 5.10, and can be explained by skin movement. Because the clavicle is a particularly thin long bone, any skin movement at marker 15 is accounted for by a phantom axial rotation. The systematic increase of the angle indicates that the skin movement is a function of the movement which is to be expected. However, unlike the previous rotations, the dominant skin movement, which is a noisy process, also produces larger variations (Figure 5.10, θ_3).

Euler angles were used for the rotation angles of the humerus relative to the clavicle (Figure 5.11). θ_1, which is the initial rotation about the clavicle's z-axis, determines to some degree the

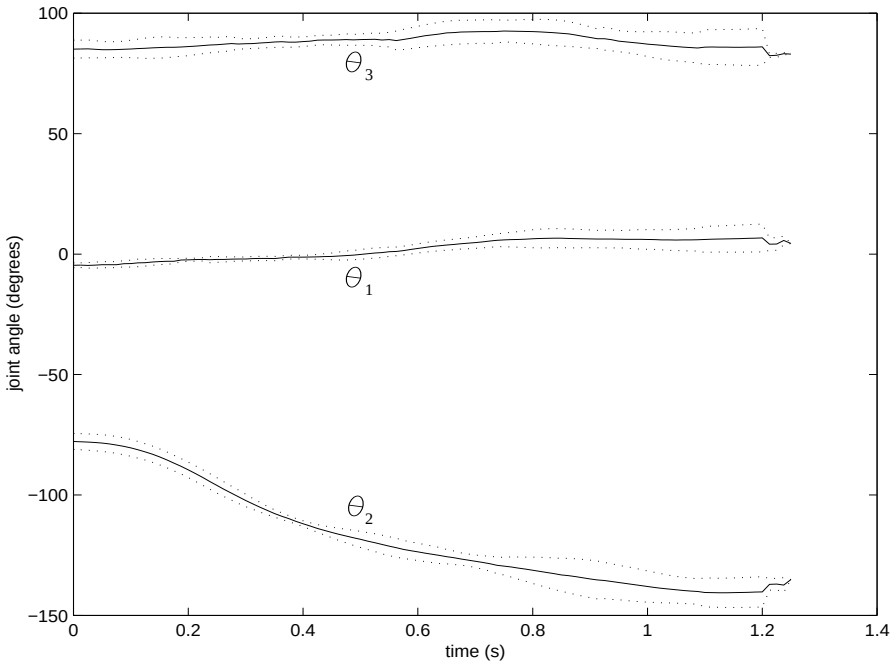

Figure 5.12: Forearm rotation relative to the humerus

The relative rotation, for Task 6, of the forearm relative to the humerus frame was decomposed into Euler angles: θ_1 (humerus Z), θ_2(forearm y) and θ_3 (forearm z). The solid line represents the mean angular trajectory over all trials of Task 6, while the dotted line represents a standard deviation upper and lower bound.

abduction/adduction of the humerus. This angle is greatly affected by the noise and errors in the clavicle axial orientation (since it follows the clavicle's axial rotation). None the less, the subsequent rotation θ_2, affecting a hybrid of flexion/extension and abduction/adduction, and θ_3, which is partially responsible for the medial/lateral rotation, are much more consistent.

Again Euler angles were used to describe the rotation of the forearm relative to the humerus (Figure 5.12). Here θ_1, almost a constant, 0, reveals that there are only two degrees of freedom about the elbow joint (as expected). θ_2 can be clearly interpreted as the flexion/extension of the elbow, and θ_3 responsible for the pronation/supination of the forearm. Again due to the segment frame axes there is some coupling between the anatomical angles and therefore θ_1 is not quite zero and perhaps θ_3 does not reflect the complete supination angle of the forearm.

Using a Y, Z, X (forearm axes) rotation sequence provides some anatomical information about the rotation of the hand relative to the forearm. The angle θ_1 about Y is the flexion (positive) /extension (negative) which tends to move from extended to flexed during the task. θ_2 is almost

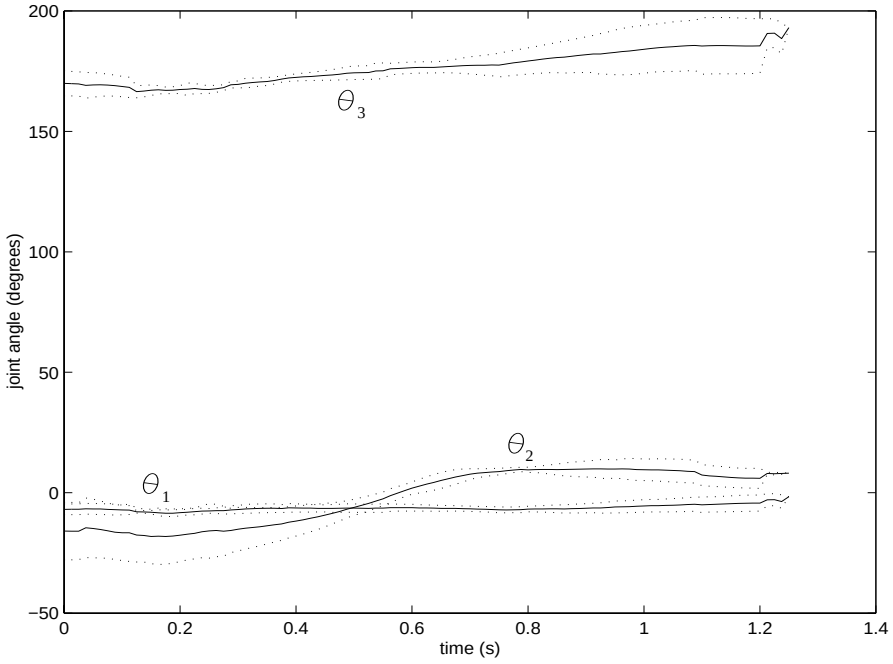

Figure 5.13: Hand rotation relative to the forearm

The relative rotation, for Task 6, of the hand relative to the forearm frame was decomposed into the angles θ_1 (forearm Y), θ_2 (forearm Z) and θ_3 (forearm X). The solid line represents the mean angular trajectory over all trials of Task 6, while the dotted line represents a standard deviation upper and lower bound.

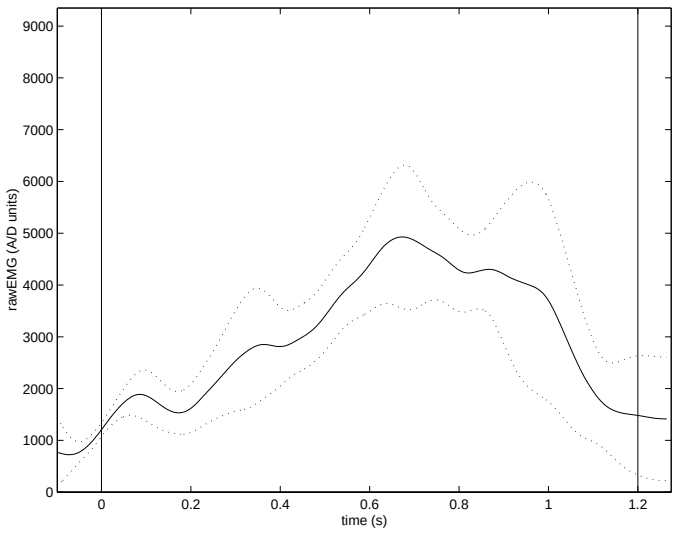

Figure 5.14: Mean and ($\pm$) one standard deviation of Teres major EMG

180^o because the two frames are defined such that their respective X and Y-axes are almost anti-parallel to one another. This angle should remain constant since there is no "spin" of the hand relative to the forearm. Finally, θ_3 is the abduction (positive)/adduction (negative) angle, and appears to be minimal throughout the performance of the task.

5.6.3 Muscle Activation Patterning

The methods to condition the collected EMG waveforms (Section 5.4.3) from the individual muscles were also encoded in MATLAB. The time "clips", used in Section 5.6.2, were again used to clip the corresponding EMG signals. EMG signals from all the trials of the same task and muscle were scaled to the average performance period and then sampled and averaged at a 1000 discrete intervals to produce a mean activation signal. Similarly the standard deviation at each interval was determined to form the boundaries (range) of the observed activation levels.

The mean and standard deviation signals, for each of the eight muscles recorded are presented in Figures 5.14-5.21 for Task 6, to illustrate the output of this process. The vertical lines indicate the start and end times of the task. Additional regions before and after were included for the purpose of observing any transitions from inactive to active and *vice-versa.*

For task 6, these activation patterns are sensible. The rapid rise in bicep activation, which

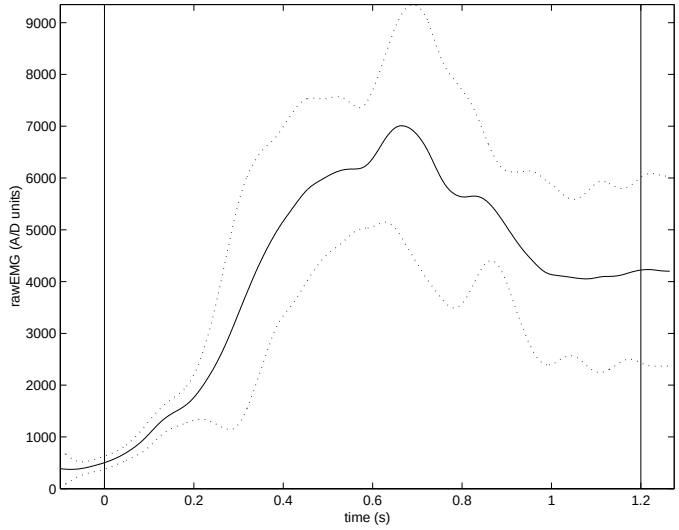

Figure 5.15: Mean and ($\pm$) one standard deviation of Anterior Deltoid EMG

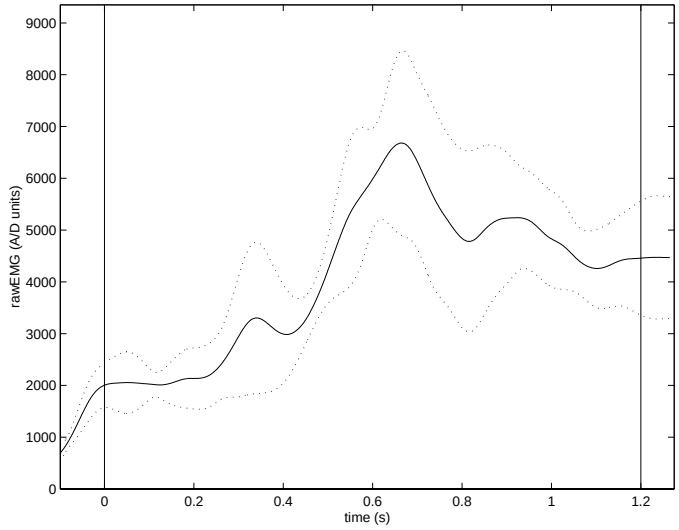

Figure 5.16: Mean and ($\pm$) one standard deviation of Medial Deltoid EMG

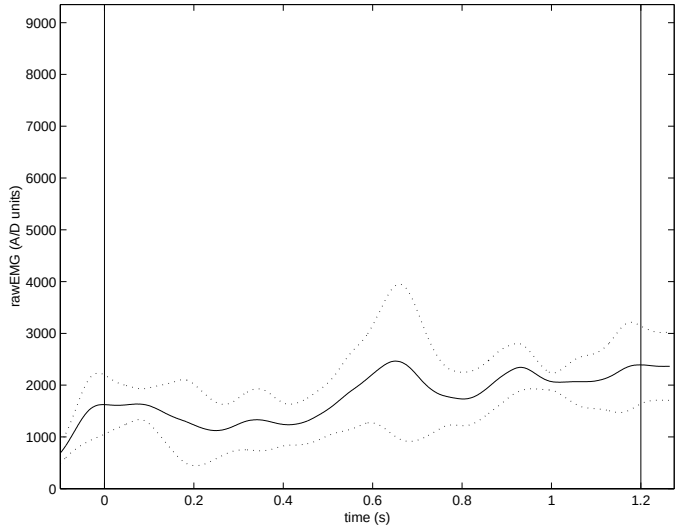

Figure 5.17: Mean and (±) one standard deviation of Posterior Deltoid EMG

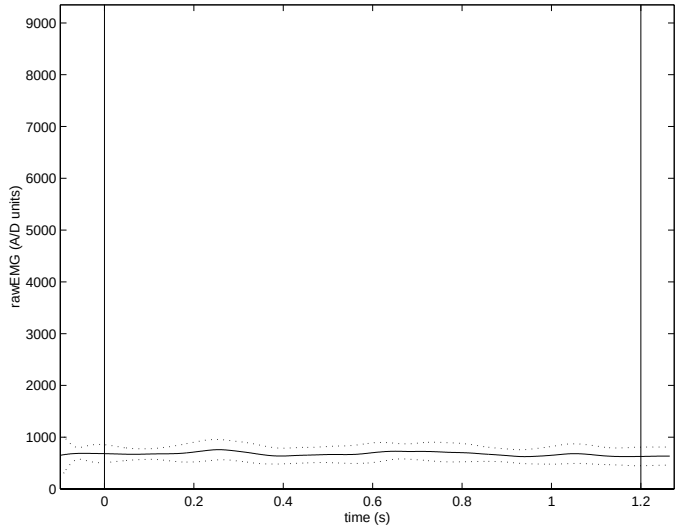

Figure 5.18: Mean and (±) one standard deviation of Pectoralis Major EMG

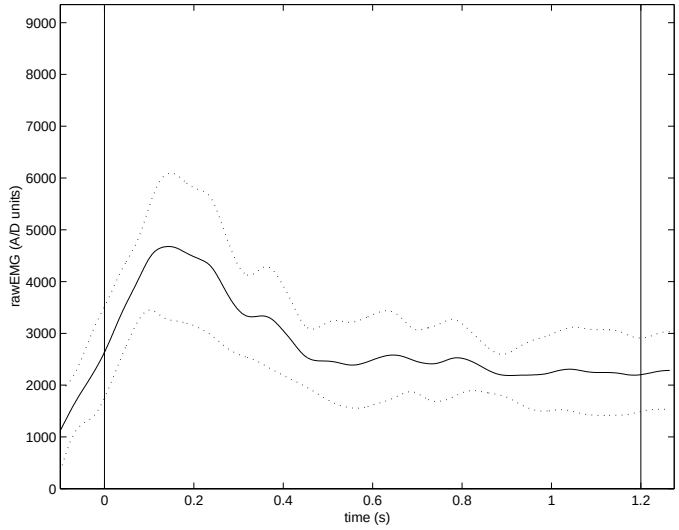

Figure 5.19: Mean and (±) one standard deviation of Biceps Brachii EMG

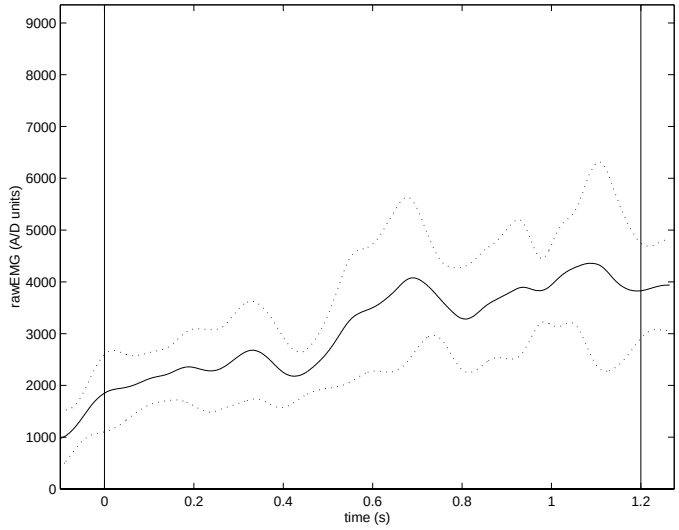

Figure 5.20: Mean and (±) one standard deviation of Triceps Brachii EMG

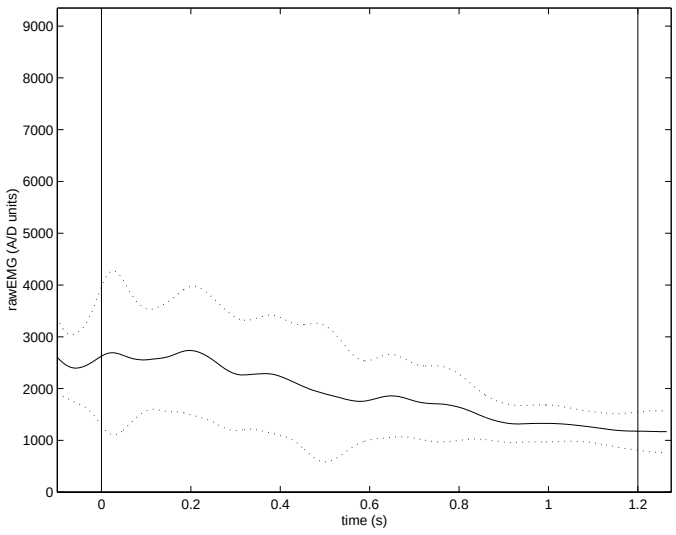

Figure 5.21: Mean and ($\pm$) one standard deviation of Brachioradialis EMG

works to flex the elbow joint, corresponds well with the detection of movement initiation. This followed by the larger output of the anterior and medial deltoid account for the flexion and then abduction necessary to lift the entire arm against gravity. Finally, the activity of the triceps, increasing to match that of the biceps, seems necessary to stabilize the forearm relative to the humerus.

The large variability over the five trials is perhaps an indication of both the variability in EMG detection (i.e. electrode pick-up) as well as the possible sensitivity of the dynamics (or driving forces) necessary to produce very similar kinematics (i.e. variuos muscle deployments generating simlilar kinematics). Specifically, the temporal scaling assumes that the activation sequence scales linearly with performance time such that slow and fast movements differ only in execution speed. Although the five trials varied by less than 6% in performance time, there is no accounting for the possibility that the nervous system employed some form of nonlinear scaling which was more concerned with the dynamics of individual phases (or features) of the movement.

5.7 Limitations of the Collection

Formulating the relative rotations of limb segments has proven far more difficult than the computational methods themselves describe. Primarily, the markers do not represent the location of fixed positions on a segment and thus their relative movement contributes significantly to the error in calculating the limb segment frames. Thus, even a least squares evaluation of each segment rotation, which attempts to minimize any error, cannot account for translations between markers that occur over the time to perform a task. A potential improvement may be to recompute the rotations based on frames originating on the previously computed centres of rotation. In fact a "dual" least squares optimization between the fixed segment lengths over all time and the best rotation at each instant can be performed in a cyclical fashion, each improving the evaluation of the other. As well, originating each frame at a joint centre will result in angular decomposition via Euler angles, etc., which have more significance in terms of the functional anatomy of the upper limb.[2]

In analyzing the EMG signals, the simple linear scaling of activation levels, based on start and end times, may in fact contribute to the observation of greater variability than truly exists, by incurring higher variability across regions of transition (i.e. rising edge in activation) due to small temporal shifts. However, for the most part, the data does not reflect this (regions of transition do not have larger deviations, e.g. Figure 5.19) and therefore suggests the resulting variability may be a result of varying signal gain (i.e. changes in EMG electrode "pick-up" during the collection period). Alignment of EMG trials based on outstanding signal features such as peaks and slopes of characteristic edges may provide a more representative composite of the EMG signals. These methods, however, would require significantly more sophisticated pattern recognition and signal processing techniques and would further increase the processing time.

For the time being, the current methods are sufficient to produce prototypical kinematic and muscle activation patterns for the purposes of comparison with predictive model results. Naturally, limiting the variability of the prototypical data would serve to strengthen the legitimacy of predicted results that were to fall within the boundaries of variability.

[2] As it stands now, the joint angular rotations are with respect to the segment frame as initially defined by three non-colinear markers (Section 5.3.2) with the first being the origin of the frame.

5.8 Comparison with Predicted Results

The biomechanical model results were generated for vertical reach task with and without a load (Section 4.2.4). These results correspond to tasks 1 and 7. In order to compare the performance of these tasks with the planar biomechanical model results, the captured kinematics must be presented in terms of planar motion. In addition, the biomechanical model essentially models the movement of two segments: the humerus and the forearm, since the hand is welded to the forearm in our simple biomechanical model. Therefore, we will concern ourselves with acquiring the planar rotation of these bodies.

5.8.1 Spatial to Planar Projections of Observed Kinematics

In order to leverage the spatial data we have collected, rather than select an arbitrary plane, such as the sagittal plane, to project the kinematics we determine the most significant plane of motion. In other words we are striving to obtain the best planar representation by finding the plane that minimizes the segment kinematics that are out of the plane. Since we have obtained the global rotations for each body in 3D space this task is greatly simplified. Given the individual rotation matrices we wish to find the single matrix P that can transform all the rotations to equivalent planar rotations. This means all rotation matrices must have one row and column (the same across the matrices) that are identical to zero except for the diagonal entry which should be one. Given only the two bodies of interest, we need only find P such that

$$[P][A] \approx \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad and \quad [P][B] \approx \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which yields:

$$\begin{aligned} \{P_{1\ldots}\}\{A_{\cdot:3}\} &= 0 \\ \{P_{2\ldots}\}\{A_{\cdot:3}\} &= 0 \\ \{P_{3\ldots}\}\{A_{\cdot:1}\} &= 0 \\ \{P_{3\ldots}\}\{A_{\cdot:2}\} &= 0 \\ \{P_{3\ldots}\}\{A_{\cdot:3}\} - 1 &= 0 \end{aligned}$$

where $\cdots$ and $\vdots$ represent the remaining row and column entries respectively. Similarly for rotation matrix B we get five additional equations. We assemble these 10 equations for each instant in time and solve for the best P, in a least squares sense, which satisfies the above system. P is in fact a rotation matrix that is being applied to the global frame and we are in essence rotating the frame such that the new $X'Y'$ plane minimizes the movement out of the plane. Given that P is a rotation matrix we assemble the matrix from three Euler angles which ensures that P is orthonormal. With this constraint, we can use the equations from either the last row or the last column of the product resulting in three equations per segment per instant. Assembling three equations per segment through all instances in time yields a $3\aleph \times 2$ error matrix, which the nonlinear least squares method, in this case MATLAB's `lsqnonlin`, utilizes to determine the best fit Eulerian angle parameters.

We then apply the rotation P to all the rotations in time and ignore the last row and column to get planar rotation matrices and the corresponding joint angles θ_1 and θ_2. Applying the resulting joint angle trajectory to the derived measurements of the humerus and forearm lengths (assuming the mass of the hand and load is located at the distal end) we can recreate the planar animation.

5.8.2 Planar Trajectories for Tasks 1 and 7

The planar motion of our subject performing the vertical reach without a load (Figure 5.22) appears to be very similar to that of the predictive model result (Figure 4.6). The mean performance time of our subject of 1.13 seconds, however, is close to three times slower than the predictive results.

The kinematic performance of the lifting task with a load of $5kg$ by our test subject (Figure 5.24) does not differ greatly with his no-load performance, task 1. Even the mean performance time of $1.318s$ represents an increase of only 16.6% whereas the predictive model time of $0.715s$ is closer to being twice as slow than without the load. Still, the predictive model is considerably faster than our test subject. The major reason for this discrepancy can be attributed to the relaxation of the control bounds used in the model prediction, Section 4.2.4. Although this was necessary simply to get the model to perform the task, it also has the effect of making the model potentially "stronger" since muscles that would normally be antagonistic might be perfoming in concert by virtue of both moment-arm inversion and by allowing the actuators to push. Because

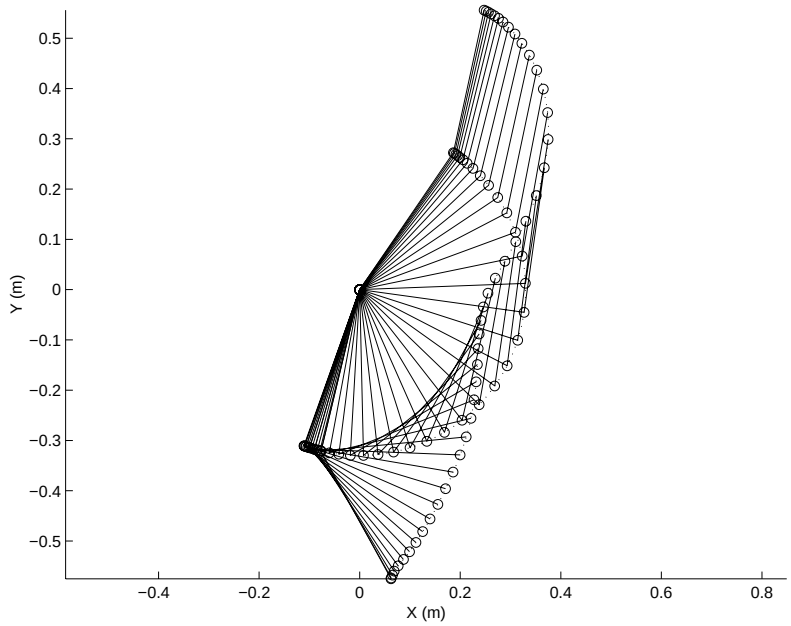

Figure 5.22: Planar motion of the test subject performing task 1

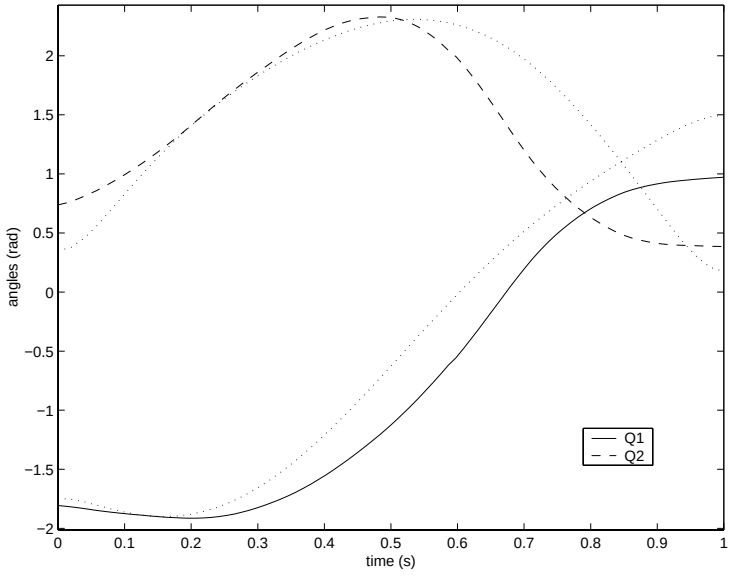

Figure 5.23: Planarized angular trajectories of test subject performing task 1

The final times have been normalized to enable comparison with model predicted angular trajectories (dotted).

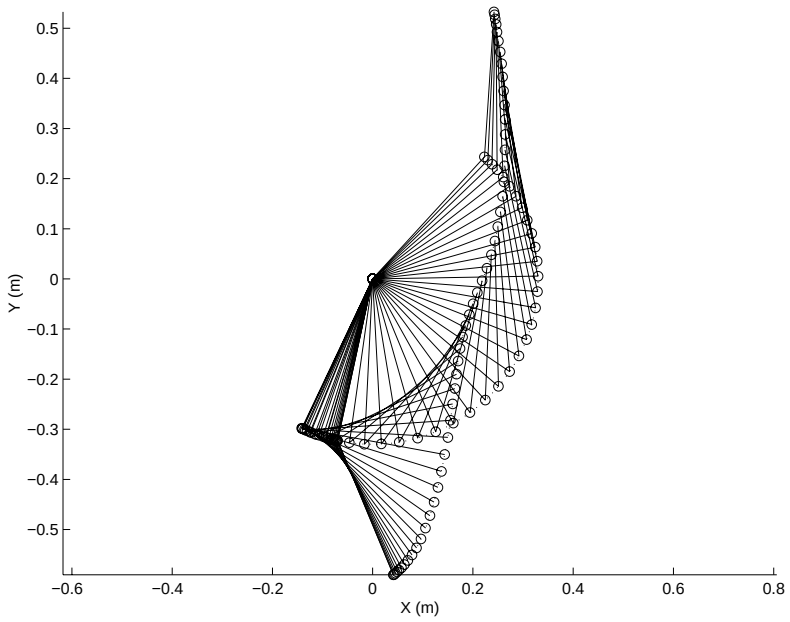

Figure 5.24: Planar motion of the test subject performing task 7

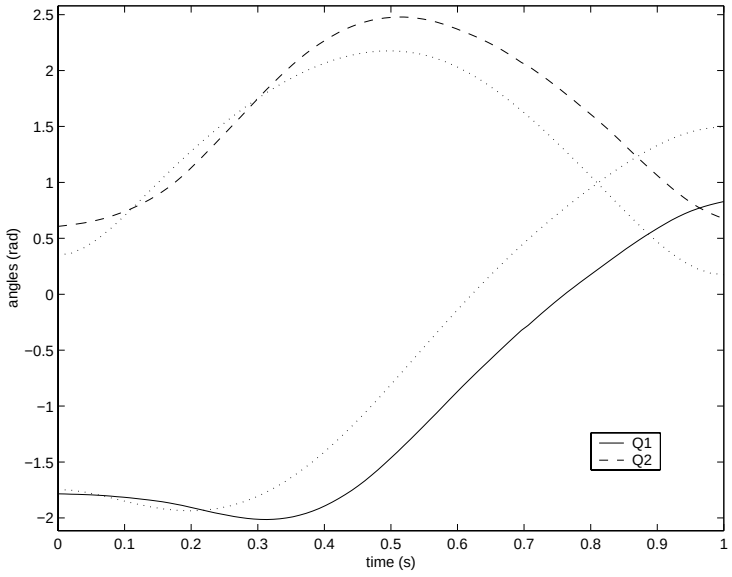

Figure 5.25: Planarized angular trajectories of test subject performing task 7

The final times have been normalized to enable comparison with model predicted angular trajectories (dotted).

of our modelling error, it is impossible to ascertain if inertial parameters were under estimated or that maximal force limits were over estimated.

An interesting point to note, however, is the small counter movement of the humerus prior to shoulder flexion — more so in the loaded case — which was also observed in the predictive model results. This counter movement exploits the effects of gravity by providing additional kinetic energy (from the initial increase in potential) through the up swing of the humerus while the elbow flexion reduces the system moment of inertia about the shoulder. Surprisingly, even though our model actuation had little anatomical relevance, the joint trajectories when compared in normalized in time (Figures 5.23 and 5.25) are quite similar to one another. We believe this is principally due to our optimization criteria. Although both models differ greatly in how net joint moments (and thus inherently different capabilities) the general strategies remain the same. Regardless of how the system is actuated, it is always an effective strategy to reduce the inertia about the shoulder and to exploit conservative forces when trying to minimize time.

5.8.3 Muscle Activations for Tasks 1 and 7

Due to the invalidity of our biomechanical model, resulting from the two-point muscle interconnections (Section 4.2.4), the activations of the predictive model are not anticipated to be similar to the muscle activations collected from the test subject. We include the muscle activation levels from our test subject for the sake of completeness.

In both tasks, the activity of the biceps and brachioradialis lead the motion by both flexing the elbow and slightly extending the shoulder (counter-movement), interestingly enough, by pure momentum conservation. The deltoids (mostly the anterior and medial) then activate as the elbow reaches its maximum flexion and they reach maximum activation as the hand approaches shoulder elevation. This corresponds well with the larger moment generated by the weight of the arm and the external load about the shoulder at that position. The temporal features of the muscle activations for both tasks are very similar, with the loaded task stretched out over a slightly longer period. As expected, the raw activation levels are also higher for the loaded task. Also interesting is the activity of the bicep well before movement initiation, which is the necessary muscle activation required to hold the load still and perhaps to counteract joint separation.

As expected the actuator control signals from the five modelled actuators in our predictions

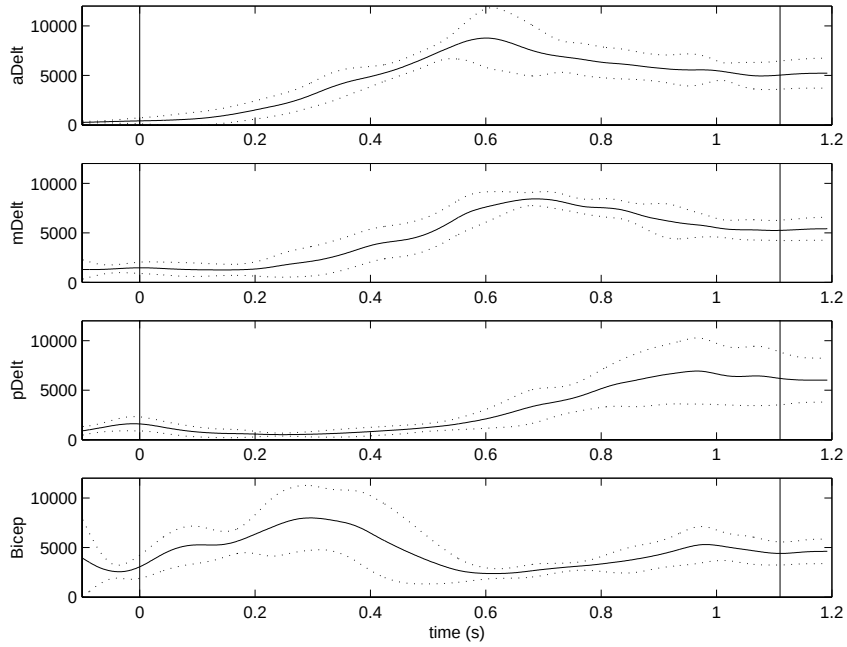

Figure 5.26: Test subject deltoid and biceps activation levels from task 1

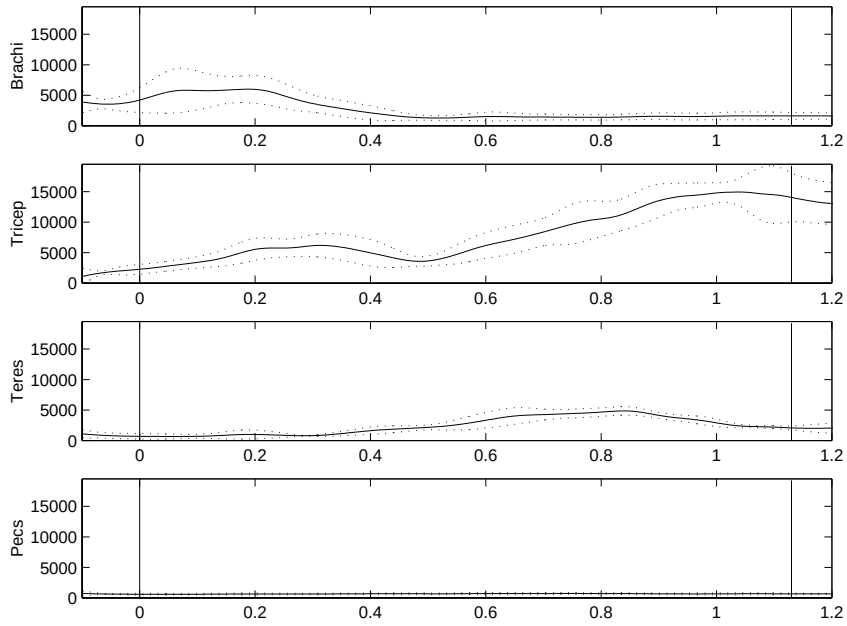

Figure 5.27: Test subject brachioradialis, triceps and teres major activations from task 1

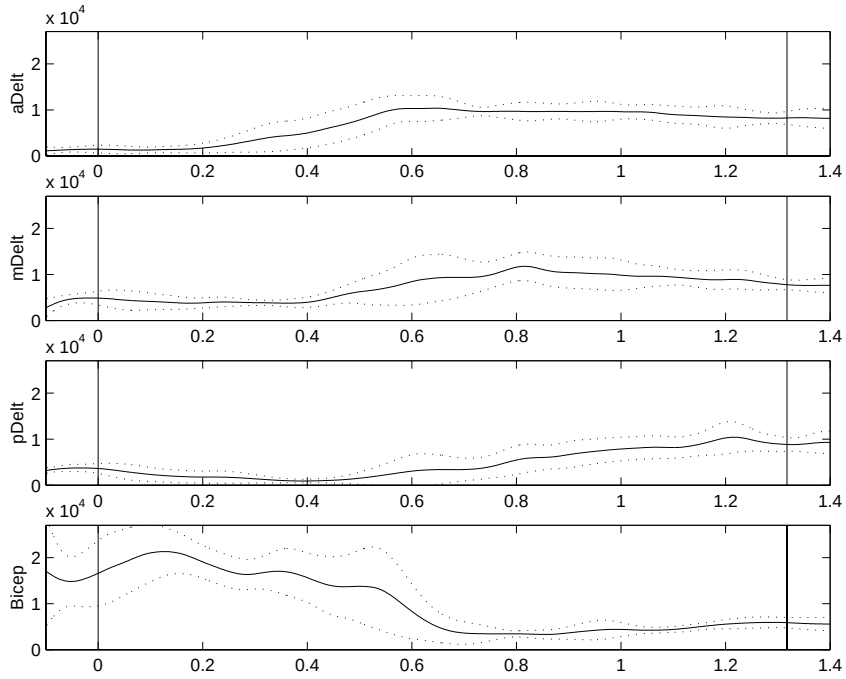

Figure 5.28: Test subject deltoid and biceps activation levels from task 7

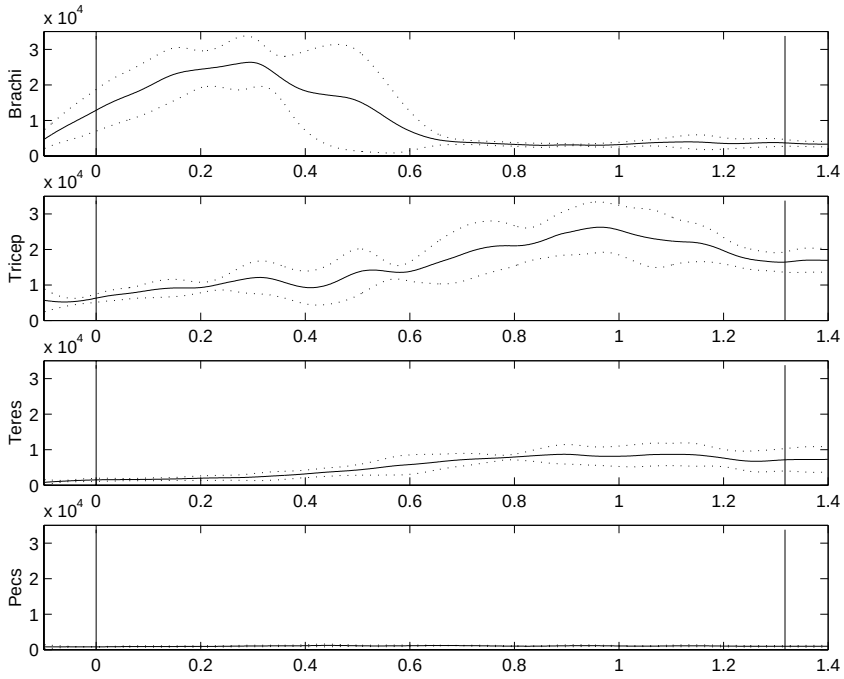

Figure 5.29: Test subject brachioradialis, triceps and teres major activations from task 7

(Figures 4.8and4.11) and our subject's activation signals for the anterior deltoid, biceps, brachio-radialis, triceps and teres major muscles, bear no resemblance.

Chapter 6

Conclusions

6.1 Summary of Model Predictive Results

The model predicted results, at this stage, are similar to the subject's performance in terms of segmental kinematics. Several key features of performing the reaching and lifting tasks effectively are produced by both the model prediction and our subject. These include: an immediate elbow flexion coupled with a small counter-movement via an extension of the shoulder and then fairly rapid flexion of the shoulder. This strategy was determined by the hybrid GA-SQP method without any guiding initial approximation.

Unfortunately, our results do not match in terms of task dynamics considering the predicted controls in comparison with the muscle activations captured by EMG. This was expected for several reasons: first, the present geometric model of muscle attachment, with a single line interconnection of the origin and insertion points, is anatomically unrealistic. It causes the model to be unduly sensitive to the degree of flexion/extension, which can cause the moment arm to be non-existent or such that the moment generated contradicts the function of the muscle (Section 4.2.4).

Second, the human upper limb has dozens of muscles with multiple functions and is not limited to only five actuators. The five muscles were modelled as an attempt to capture the most significant actuators for the given task and to hopefully provide encouraging similarities when

compared to the same actuators in the human system. Further, dissimilarity arises from the fact that the upper-limb also has many more degrees of freedom. Even though the simple vertical reach is kinematically close to planar motion, this is achieved by muscle synergy that cannot be captured by the current biomechanical model (Chapter 2). Another significant consequence is that some muscle (i.e. Teres Major) lengths become unrealistically small in two dimensions which renders the muscle ineffective and further confounding results.

Finally, the effect of the objective function is not known. For example, if the importance of minimizing stress was increased relative to performance time would we have predicted slower and more similar performances. Of course these are the types of questions we hoped to answer with our model but this cannot be determined until the previous issues are resolved.

6.2 Assessment of Research

In hindsight, the human data collection should have been held off until the biomechanical model and optimization methods had advanced well beyond the prototype stage. This time could have been invested in more rigorous testing of the biomechanical model as well as encoding the ODE solver and SQP methods in a compiled language such as C or Fortran.

Fortunately, the methods implemented by DYNOPT [20] have given us a head start in that direction. The biomechanical model of the upper-limb used in our investigation of optimal controls provided several important features that conventional solvers (such as DYNOPT) alone could not resolve. Primarily, the increased number of actuators and their non-linear appearance, via muscle models in the dynamic equations, creates many local minima which gradient-based methods cannot overcome. In addition, the over-actuated system tends to create long "valleys" in the objective space causing difficulties in computing a new search direction in an SQP method. This raises the issue of "designing" a better objective function in order to minimize these regions on the objective surface while maintaining biomechanical relevance.

Even the hybrid GA-SQP method has its difficulties with this phenomenon which causes many unique chromosomes (along the long valley bottom) to have identical fitnesses thus making dominance low which keeps the mutation low in our adaptive mutation implementation. This may result in a great deal of computational time spent exploring these valleys until, by chance mutation, a significantly fitter individual appears. Although the hybrid GA-SQP method is more

effective than the SQP, it cannot be concluded that the hybrid method is the most efficient method. The use of coarser (shorter) chromosomes (less encoded precision) can avoid this altogether, but at the risk at finding a solution which is optimal only for that encoding.

6.3 Contributions of Research

The main contributions of this research include the application of graph-theory and genetic algorithm methods to biomechanical simulation as well as the development of data processing tools for comparing human performance with simulation results.

Although the graph-theoretical approach to multi-body dynamics has undergone rapid development since the 1970's [4], the application of this research to biomechanical modelling has yet to be recognized. Applying these methods via automated tools such as DynaFlex goes beyond the state of the art in terms of formulation methods used by biomechanists today.

By far the most significant contribution of this research is the introduction of the genetic algorithm into the realm of dynamic optimization. Genetic algorithms have found their way into solving many static optimization problems including complex configuration and pattern recognition problems. Newton's method (or its variants), however, remains the dominant method for the direct optimization approach to the optimal control problem. For the most part, the use of genetic algorithms has been limited to determining signal parameters for problems where the general form of the control is known and can be exploited. In these instances, the controls can be encapsulated by either function parameters or neural network weightings (when there is inherent uncertainty about the inputs to the control system). In either case, the genetic algorithm is used to "tune" the model parameters or train the network so that differences in model performance with measured human performance is minimized [5].

For optimal control problems, gradient methods are often sought because the objective functions are usually smooth, at least piecewise continuous, and doubly- differentiable due to the (non-linear) dynamic equations which are appended using Lagrange multipliers. In addition, optimality conditions (specifically, the co-state equations) can be exploited to compute the variation of the objective function with respect to discretized control inputs, and thus gradient-based methods appear to be the most efficient solution method. The fundamental assumption that the initial point lies on a surface that monotonically approaches the optimal solution is usually overlooked.

Alternatively, the initial guess is set to a solution that is already a good representative of the desired performance. Few researchers would consider abandoning the efficiency of gradient approaches for time consuming explorative methods such as the GA. Those concerned with finding the globally optimal solution often find themselves trying many initial points in hopes that they all converge to the same optimum. Often, getting these methods to converge is difficult because of the abundance of local optima where the current solution can be trapped far from satisfying terminal constraints. In spite of this, given that an initial guess from experimentation or manual calculation of a simplified case does converge, this is taken as evidence that the solution is the global optimum since the handful of other attempts did not converge. At the very least then, the hybrid GA-SQP we have developed frees the designer from having to find suitable initial guesses and presents an automated and systematic method of trying numerous "initial guesses".

Finally, we have developed a suite of data processing tools that facilitate the analysis of three-dimensional marker kinematics. These include finding both the most representative rotations and joint centre locations such that they minimize discrepancies with marker positions in a least squares sense. Furthermore, we address planar motions not by simply selecting arbitrary planes, but by mathematically identifying the plane which is most representative of the motion.

6.4 Future Research

6.4.1 Biomechanical Model

It is apparent that accurate muscle geometry is critical to reproducing human-like behavior. The present linear interconnection of muscles from origin to insertion points is insufficient. Intermediate points must be introduced such that the muscles provide the appropriate function regardless of the degree of flexion or extension (Figure 6.1). This evolution in the biomechanical model should eliminate muscles that had to "push" in the current model due to the erroneous moment-arm lengths.

The most significant advancement in the model has to be the transition to a spatial model. From the modelling perspective, it is no more complex to do thanks to the GT methods and DynaFlex. The difficulty lies in acquiring the appropriate model parameters such as the origin and insertion coordinates given the sensitivity of the joint moments to these parameters.

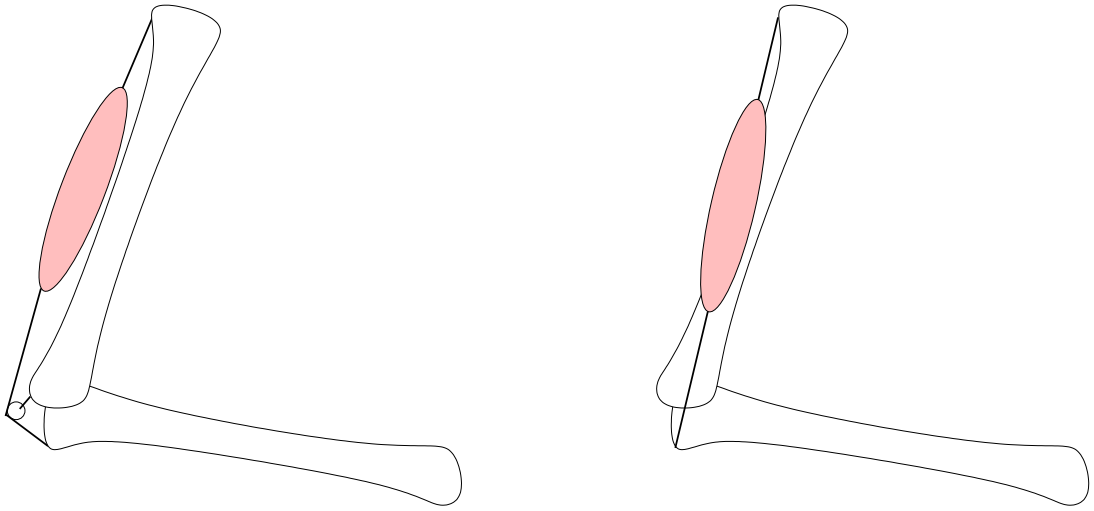

Figure 6.1: A 3pt. muscle geometry vs. the current 2pt. muscle geometry

6.4.2 Optimization Methods

Spatial mechanisms typically have more degrees of freedom and therefore have more state and control variables. Moving to spatial models thus increases both simulation time (more differential equations to integrate) as well as the search space of the optimal control problem (more control variables). Thus for practical solution of large scale optimal control problems efficient implementation is critical. For example, the DYNOPT implementation of the Runge-Kutta integration and SQP optimization is on the order of 24 times faster than the identical implementation in MATLAB. MATLAB and other scientific computing environments (OCTAVE, SCILAB, etc.) provide superior prototyping environments, but for pure execution speed they cannot match compiled C or Fortran code.

The implementation as it stands now, is an ad hoc assembly of MATLAB/OCTAVE functions, PERL scripts, and the DYNOPT C executable, whereby communication is performed primarily by flat files which are continually written and read to/from the file-system. Eliminating communication via flat files should alleviate at least a tenth (and up to a third) of the time costs per call to DYNOPT, which occurs approximately N_p (a population size) times per generation.

Following suit, the GA should be translated into C to reduce the GA cycle time as well as homogenize the implementation and provide some degree of portability.

Portability may be a significant issue when considering the potential speed gains that can be achieved via computational parallelization. The underlying GA is easily distributed such that several processors can each be given a portion of the population for evaluating the fitness function. There is an almost linear reduction in computational time with an increase in the number of processors [49] as long as the number of processors is less than the population size. Distributed systems, such as the BEOWULF system, are rivaling and in many cases out-performing the computational capabilities of mainframes at the fraction of the cost. They are quickly becoming accessible to research scientists and there is no doubt that the solution of optimal control problems will benefit from this as well.

6.4.3 Collection Data Processing

The reconstruction of limb segment kinematics from collected marker kinematics was performed with a significant amount of mathematical rigor in order to get better estimates of limb segment geometries as well as their rotations. We should continue in this way by coupling the least squares operations in a dual strategy which runs the joint centre location optimization and then the best fit rotations one after the other in a cyclical approach to find even better sets of rotations and segment geometries.

6.4.4 Human Motion Prediction

After making improvements to the model and the optimization methods we can consider finding an objective function with appropriate weightings that can reproduce the performances for the broad variety of tasks we have already collected. We should devise a recursive method of determining the weightings of the individual objective function criteria such that the resulting model movements best match the human movement observations. This would require cycling through many optimal control problems and therefore the computational costs would increase several fold. At the current rate of increase in computational speed it will not be very long before such a process would be feasible.

6.5 Concluding Remarks

In conclusion, we have demonstrated a feasible methodology for determining control strategies for the movement of complex mechanical systems and how they can be applied to the performance of the human upper-limb. We have demonstrated the advantages of biomechanical modelling via a graph-theoretic approach in order to generate reliable formulations. This method also lends itself to a component paradigm that enables us to reuse the topology of previous models and easily interchange components.

The most significant development has been employing a genetic algorithm within the direct optimization method. Employing the more exhaustive GA with the faster SQP techniques eliminates the requirement for an initial guess while providing accurate results within the desired precision. This is a significant improvement since the alternative requires manual trials with varying guesses. Otherwise, we risk obtaining only a local optima in the near neighborhood of the initial guess. Furthermore, the hybrid GA-SQP enables the solution of over-actuated systems which are generally plagued with regions where the Hessian is near singular, which would normally cause an SQP method to fail.

Finally, we are now in the position to provide a test bed for human movement researchers to experiment with various model and optimality assumptions and provide the tools for comparing model results with data collected from motion capture systems. We hope this will lead to a greater understanding of human motion control strategies and help to identify methods of restoring and/or improving human performance. Future method developments will seek to increase the ease of transition between the modelling, optimization and comparison methods as well as the overall efficiency of the computations.

Bibliography

[1] S. K. Agrawal, P. Claewplodtook, and B. C. Fabien. Optimal trajectories of open-chain mechanical systems: a new solution without lagrange multipliers. *Journal of Dynamic Systems, Measurement, and Control, Transactions of the ASME*, 120:134–136, 1998.

[2] Sunil K. Agrawal and Brian C. Fabien. *Optimization of Dynamic Systems*. Kluwer Academic Publishers, Netherlands, 1999.

[3] James G. Andews and Sean P. Mish. Methods for investigating the sensitivity of joint resultants to body segment parameter variations. *Journal of Biomechanics*, 29(5):651–654, 1996.

[4] G. C. Andrews. The vector network model: A new approach to vector dynamics. *Mechanism and Machine Theory*, 10:57–75, 1975.

[5] Mehran Armand. *Adaptation in Bidel Locomotion: Insights from Dynamic Modelling, Numerical Optimization, and Neuro-Fuzzy-Genetic Programming*. PhD thesis, University of Waterloo, 1998.

[6] Aaron E. Baumal. Automated design of mechanical systems through numerical optimization of dynamic behaviour. Master's thesis, University of Waterloo, 1997.

[7] P. J. Beek and R. J. Bootsma. Physical and informational principles in modelling coordinated movements. *Human Movement Science*, 10(1), 1991.

[8] J. E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along a specified path. *International Journal of Robotics Research*, 4(3):3–17, 1985.

[9] H. G. Bock. Recent advances in the parameteridentification techniques for o.d.e. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations, Proceedings of an International Workshop*, pages 339–340, Heidelberg, Germany, August 30 - September 3 1982. Birkhauser.

[10] C. A. Brebbia. *The Boundary Element Method for Engineers.* Pentech Press, 1978.

[11] C. G. Broyden. Quasi-newton mehods and their applications to function minimization. *Mathematics of Computation*, 21:368–381, 1967.

[12] Ivan T. Bruulsema and John J. McPhee. Dynamic modelling and optimization of automitive suspension systems. Technical report, Systems Design Engineering, University of Waterloo, December 1999. submitted to CSME Forum 200, Montreal, May 16-19, 2000.

[13] L. Chambers. *Practical Handbook of Genetic Algorithms: New Frontiers.* CRC Press, Inc., 1995.

[14] Jacek Cholewicki, Stuart M. McGill, and Robert W. Norman. Comparison of muscles forces and joint load from an optimization and emg assited lumbar spine model: Towards development of a hybrid approach. *Journal of Biomechanics*, 28(3):321–331, 1995.

[15] C. K. Chow and D. H. Jacobson. Studies of human locomotion via optimal programming. *Mathematical Biosciences*, 10(1):239–306, 1971.

[16] Eric Coutu. Demonstration of genicom's safework software, December 1998. Held at the Department of Kinesiology, University of Waterloo.

[17] R. D. Crowninshield and R. A. Brand. A physiologically based criterion of muscle force prediction in locomotion. *Journal of Biomechanics*, 14(11):793–801, 1981.

[18] Wolfgang J. Daunicht. Development of a stance-gait-neuroprosthesis for paraplegic patients. In *Biomechanics Meets Robotics: Modelling and Simulation of Motion*, Heidelberg, Germany, November 1999.

[19] J. Dean, H. Cruse, M. Bruwer, and U. Steinkuhler. Studies of human arm movements using three joints: control strategies for a manipulator with redundant degrees of freedom. In Madan M. Gupta and Naresh K. Sinha, editors, *Intelligent Control Systems: Theory and Applications*, chapter 24. IEEE Press, New York, 1996.

[20] Brian C. Fabien. Some tools for the direct solution of optimal control problems. *Advances in Engineering Software*, 29:45–61, 1998. Software available at ftp site: abs-5.me.edu/pub/book/code.

[21] Reza Fotouhi-C and Walerian Szyszkowski. An algorithm for time-optimal control problems. *ASME Journal of Dynamic Systems, Measurement and Control*, 120(1):414–418, 1998.

[22] H. M. Frost. *Bone Modelling and Skeletal Modelling Errors.* Charles C. Thomas, Springfield, IL, 1973.

[23] T. Fukuda. Modelling and control of brachiation robots. In *Biomechanics Meets Robotics: Modelling and Simulation of Motion*, Heidelberg, Germany, November 1999.

[24] Rafail F. Gabasov and Raina M. Kirillova. *Methods of Optimization.* Optimization Software Inc. Publications Division, New York, 1988.

[25] H. P. Geering, L. Guzzella, S. A. R. Hepner, and C. H. Onder. Time-optimal motions of robots in assembly tasks. *IEEE Transactions on Automatic Control*, 31(1):512–518, 1996.

[26] I. Gelfand and S. V. Fomin. *Calculus of Variations.* Prentice-Hall, 1963.

[27] Stanton A. Glantz. A three-element description for muscle with viscoelastic passive elements. *Journal of Biomechanics*, 10:5–20, 1977.

[28] U. Glitsch and W. Baumann. The three-dimensional determination of internal loads in the lower extremity. *Journal of Biomechanics*, 30(11/12):1123–1131, 1997.

[29] Y. Guiard. On fitt's and hooke's laws: simple harmonic movement in upper-limb cyclical aiming. *Acta psychologica*, 82(1), 1993.

[30] H. Hatze. A myocybernetic control model of skeletal muscle. *Biological Cybernetics*, 25(1):103–119, 1977.

[31] J. Heilig and J. J. McPhee. Determination of minimum-time maneuvers for a robotic manipulator using numerical optimization methods. *Mechanics of Structures and Machines*, 27(2):185–201, 1999.

[32] A V. Hill. The heat of shortening and the dynamic constants of muscle. In *Proceedings of the Royal Society of Biology*, volume 126, pages 136–195, 1938.

[33] At L. Hof. An explicit expression for the moment in multibody systems. *Journal of Biomechanics*, 25(10):1209–1211, 1992.

[34] J. H. Holland. *Adapatation in Natural and Artificial Systems.* MIT Press, 1992.

[35] Li Huang and Leila Notash. Design of parallel manipulators for high reliability and minimal failure rate. In S. Ziada and D. S. Weaver, editors, *Proceedings of the 17th Canadian Congress of Applied Mechanics*, pages 339–340, Hamilton, Ontario, May 30-June 3 1999. Canadian Congress of Applied Mechanics.

[36] A. F. Huxley. Proposed mechanism of force generation in striated muscle. *Nature*, 233(1):533–538, 1971.

[37] M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated chains. *Journal of Dynamic Systems, Measurement, and Control, Transactions of the ASME*, pages 164–172, September 1971.

[38] H. K. Kesavan, H. E. Koenig, and Y. Tokad, editors. *Analysis of Discrete Physical Systems.* McGraw-Hill, 1967.

[39] I. Kingma, M. P. De Looze, H. M. Toussaint, H. G. Klinjsma, and T. B. M. Bruijnen. Validation of a full body 3-d dynamic link segment model. *Human Movement Sciences*, 15(1):833–860, 1993.

[40] I. Kingma, H. M. Toussaint, M. P. De Looze, and J. P. Van Dieen. Segment inertial parameter evaluation in two anthropometric models by application of dynamic linked segment model. *Journal of Biomechanics*, 29(5):693–704, 1996.

[41] Arthur D. Kuo. A mechanical analysis of force distribution between redundant, multiple degree- of-freedom actuators in the human: implications for the central nervous system control. *Human Movement Science*, 13:635–663, 1994.

[42] Cornelius Lanczos. *The Variational Principles of Mechanics.* University of Toronto Press, Toronto, 1970.

[43] Mark L. Latash. *Control of Human Movement.* Human Kinetics, Windsor, Ontario, 1989.

[44] C. T. Lin and C. S. G. Lee. *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems.* Prentice-Hall, Inc., 1996.

[45] Gerald E. Loeb and William S. Levine. Linking musculoskeletal mechanics to sensorimotor neurophysiology. In J. M. Winters and S.L-Y Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 10. Springer-Verlag, 1990.

[46] R. M. M. Mattheij and J. Molenaar. *Ordinary Differential Equations in Theory and Practice.* John Wiley and Sons, Inc., Toronto, ON, 1996.

[47] W. Maurel, D. Thalman, P. Hoffmeyr, P. Beylot, P. Gingins, and N. M. Thalmann. A biomechanical musculoskeletal model of the human upper limb for dynamic simulation. In R. Boulie and G. Hegron, editors, *Proceedings of the Eurographics Workshop*, pages 121–136, Poitiers, France, August 31-September 1 1996. Eurographics, Speinger Wien New York. http://www.epfl.ch/ maurel/CHARM/WP3.

[48] John J. McPhee. Graph-theoretic modelling of mechanical systems. Technical report, Systems Design Engineering, University of Waterloo, 1996.

[49] Travis Metcalfe and ed Nather. Natural slection in a linux universe. *Linux Journal*, pages 58–61, September 1999.

[50] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1992.

[51] Yoshiko Nakamura. *Advanced Robotics: Redundancy and Optimization.* Addison-Wesley Publishing Company, Don Mills, 1991.

[52] B. M. Nigg and A. J. van den Bogert. Simulation. In B. M. Nigg and W. Herzog, editors, *Biomechanics of the Musculo-skeletal System*, chapter 4.8. Wiley, 1995.

[53] The Numerical Algorithms Group Limited. *The NAG Fortran Library Manual, Mark 15*, first edition, 1991.

[54] Sandra J. Olney and David A. Winter. Predictions of knee and ankle moments from emg and kinematic data. *Journal of Biomechanics*, 18(1):9–20, 1985.

[55] Olayiwola Oshinowo. Automated analysis of planar mechanical systems with user-defined coordinates: A graph theoretic approach. Master's thesis, University of Waterloo, 1996.

[56] B. Pagurek. The classical calculus of variations in optimum control problems: An introduction to the maximum principle of pontryagin. Technical report, Controls Systems Laboratory, Department of Electrical Engineering, University of Toronto, 1962.

[57] Nigel Palastanga, Derek Field, and Roger Soames. *Anatomy and Human Movement: Structure and Function.* Heinemann Medical Books, Oxford, 1989.

[58] M. G. Pandy, F. C. Anderson, and D. G. Hull. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *ASME Journal of Biomechanical Engineering*, 114(4):343–363, 1992.

[59] A. E. Patla, B. S. Hudgins, P. A. Parker, and R. N. Scott. Myolectric signal as a quantitative measure of muscle mechanical output. *Medical and Biological Engineering and Computing*, 20(1):319–328, 1982.

[60] A. Pedotti, V. V. Krishnan, and L. Stark. Optimization of muscle-force sequencing in human locomotion. *Mathematical Biosciences*, 38(1):57–76, 1978.

[61] G. Lindfield J. Penny. *Numerical Methods Using MATLAB.* Ellis Horwood Limited, Toronto, 1995.

[62] D. D. Penrod, D. T. Davy, and D. P. Singh. An optimization approach to tendon force analysis. *Journal of Biomechanics*, 7(1):123–129, 1974.

[63] M. J. D. Powell. Algorithms for functions that use lagrangian functions. *Math. Prog.*, 14:224–248, 1978.

[64] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 1986.

[65] R. Raikova. A general approach for modelling and mathematical investigation of the human upper limb. *Journal of Biomechanics*, 25(8):857–867, 1992.

[66] J-M. Renders and S. P. Flasse. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, pages 243–258, April 1996.

[67] Andy L. Ruina. Robots without motors, people without muscles. In *Biomechanics Meets Robotics: Modelling and Simulation of Motion*, Heidelberg, Germany, November 1999.

[68] G. Savage and M. Chandrashekar. Engineering systems - modelling, analysis and design. SYDE 361 Course Notes, University of Waterloo, 1995.

[69] Hans-Paul Schefel. *Evolution and Optimum Seeking.* John Wiley and Sons, Inc., Toronto, 1994.

[70] Marcus Scherrer. Adding elemnts to dynaflex by inclusion of their virtual work. Technical report, Systems Design Engineering, University of Waterloo, 1999.

[71] Adam Lowell Schwartz. *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems.* PhD thesis, University of California at Berkeley, 1996.

[72] A. Seth and J. McPhee. Prediction of optimal arm motions using a genetic algorithm. In S. Ziada and D. S. Weaver, editors, *Proceedings of the 17th Canadian Congress of Applied Mechanics*, pages 339–340, Hamilton, Ontario, May 30-June 3 1999. Canadian Congress of Applied Mechanics, McMaster University.

[73] D. F. Shanno. Recent advances in gradient based unconstrained optimization techniques for large problems. *ASME Journal of Mechanics, Transportation, and Automation*, 105:155–159, 1983.

[74] P. Shi. *Flexible Multibody Dynamics: A New Approach using Virtual Work and Graph Theory.* PhD thesis, University of Waterloo, 1998.

[75] P. Shi and J. McPhee. Dynamics of flexible multibody systems using virtual work and linear graph theory. *to appear in Multibody System Dynamics*, 2000.

[76] J. F. Soechting, C. A. Buneo, U. Herrmann, and M. Flanders. Moving effortlessly in three dimensions: does donders' law apply to arm movement? *Journal of Neuroscience*, 9(15):6271–6280, 1995.

[77] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. *Computer*, pages 17–26, 1994.

[78] Wolfram Stadler. *Analytical Robotics and Mechatronics.* McGraw-Hill, Inc., Toronto, 1995.

[79] Dan. Stashuk. Biomedical engineering: Course and lecture notes. Systems Design Engineering 444, 1996.

[80] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Spinger-Verlag, New York, 1980.

[81] G. Taga. A model of neuro-musculo-skeletal system for human locomotion. *Biological Cybernetics*, 73(1):97–111, 1995.

[82] A. F. van der Voet, N. G. Shrive, and N. S. Scharar. Numerical modelling of articular cartilage in synovial joints: Poroelasticity and boundary conditions. In J. Middleton and G. N. Pande, editors, *Proceedings of the International Conference on Computer Methods in Biomechanics and Biomedical Engineering*, pages 200–209, Swansea, 1993. Books and Journals International.

[83] G. J. van Ingen Schenau. An alternate view of the concept of the utilisation of elastic energy in human movement. *Human Movement Science*, 3:301–336, 1984.

[84] T. L. Vincent and W. J. Grantham. *Nonlinear and Optimal Control Systems.* John Wiley and Sons, Inc., New York, 1997.

[85] S. C. White and D. A. Winter. Predicting muscle forces in gait from emg signals and musculotendon kinematics. *Journal of Electromyography and Kinesiology*, 4(1):217–231, 1993.

[86] J. M. Winters. Hill-based muscle models: a systems engineering perspective. In J. M. Winters and S.L-Y Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 5. Springer-Verlag, 1990.

[87] J. M. Winters and L. Stark. Muscle models: what is gained and what is lost by varying model complexity. *Biological Cybernetics*, 55(1):403–420, 1987.

[88] J. Wittenburg. *Dynamics of Systems of Rigid Bodies.* B. G. Teubner, Stuttgart, 1977.

[89] S. L. Y. Woo, M. A. Gomez, Y. K. Woo, and W. H. Akeson. Mechanical properties of tendons and ligaments i: quasi-static and non-linear viscoelastic properties. *Biorheology*, 19:385–396, 1982b.

[90] Gary T. Yamaguchi. Performing whole-body simulations of gait with 3-d, dynamic musculoskeletal models. In J. M. Winters and S.L-Y Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 43. Springer-Verlag, 1990.

[91] M. Yoshimura and H. Masui. Prediction of muscular force sharing in a human upper limb and determination of optimal limb motion. *JSME International Journal*, 35(4):574–581, 1992.

[92] F. E. Zajac and M. E. Gordon. Determining muscle's force and action in multi-articular movement. *Exrercise Sport Science Review*, 17(1):187–230, 1989.

[93] Felix E. Zajac and Jack M. Winters. Modeling musculoskeletal movement systems: Joint and segmental dynamics, musculoskeletal actuataion, and neuro muscular control. In J. M. Winters and S.L-Y Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 8. Springer-Verlag, 1990.

[94] X. Zhao, D. Tolani, B. Ting, and N. I. Badler. Simulating human movements using optimal control. In R. Boulie and G. Hegron, editors, *Proceedings of the Eurographics Workshop*, pages 109–120, Poitiers, France, August 31-September 1 1996. Eurographics.

Appendix A

Detailed Implementation of Mechanical Models

A.1 Matlab Implementation of the Planar 2-Link Manipulator

A.1.1 planarEqns

```
function Qdot = planarEqns(tau, Q, U, p)
% Integrate the state {Q} equations for the
% planar 2-link manipulator with controls U = [U1; U2] in
% normalized time, tau, and p as the final time parameter
% Q = [Q1; Q2; Q1dot; Q2dot]
% USAGE: Qdot = planarEqns(t, Q, U, p)
% Ajay Seth

% Manipulator Parameters
l = [0.4, 0.25];
J = [1.6, 0.195625, 0.01];
m = [NaN, 15, 6];

c2 = cos(Q(2));
s2 = sin(Q(2));
```

```
% Mass Matrix of the planar manipulator
M(1,1) = (m(2)+m(3))*l(1)*l(1)+(m(2)+2*m(3))*c2*l(2)*l(1)+(m(3)+m(2)/4)*l(2)*l(2)+...
         J(2)+J(3)+J(1);
M(1,2) = (m(3)+m(2)/2)*c2*l(2)*l(1)+(m(3)+m(2)/4)*l(2)*l(2)+J(2)+J(3);
M(2,1) = M(1,2);
M(2,2) = (m(3)+m(2)/4)*l(2)*l(2)+J(2)+J(3);

dtM = M(1,1)*M(2,2)-M(1,2)*M(1,2);

% Inverse of Mass matrix M = W;
W = [M(2,2) -M(1,2); -M(1,2) M(1,1)]/dtM;

% Generalized force in Dynamic EQuation
F(1,1) = -((-m(2)/2-m(3))*Q(4)*Q(4)+(-m(2)-2*m(3))*Q(4)*Q(3))*l(2)*l(1)*s2+U(1);
F(2,1) = -(m(3)+m(2)/2)*Q(3)*Q(3)*l(2)*l(1)*s2+U(2);

Qdot(1:2,1)=Q(3:4)*p;
Qdot(3:4,1) = W*F*p;
```

A.1.2 costateEqns

```
function Cdot = costateEqns(tau, C, Q, U, p)
% Integrate the costate {c} equations from final condition of
% the planar 2-link manipulator with controls U = [U1; U2] in
% normalized time, tau, and p as the final time parameter
% Q = [Q1; Q2; Q3; Q4]];  C= C1; C2; C3; C4;
% USAGE: Cdot = planarEqns(tau, C, Q, U, p)
% Ajay Seth

% Manipulator Parameters
l = [0.4, 0.25];
J = [1.6, 0.195625, 0.01];
m = [NaN, 15, 6];

% Compute trig funs only once
c2 = cos(Q(2));
s2 = sin(Q(2));

% Mass Matrix of the planar manipulator
```

```
M(1,1) = (m(2)+m(3))*l(1)*l(1)+(m(2)+2*m(3))*c2*l(2)*l(1)+(m(3)+m(2)/4)*l(2)*l(2)+...
         J(2)+J(3)+J(1);
M(1,2) = (m(3)+m(2)/2)*c2*l(2)*l(1)+(m(3)+m(2)/4)*l(2)*l(2)+J(2)+J(3);
M(2,1) = M(1,2);
M(2,2) = (m(3)+m(2)/4)*l(2)*l(2)+J(2)+J(3);

dtM = M(1,1)*M(2,2)-M(1,2)*M(1,2);

dtM_Q2 = (m(2)*m(2)/2+2*m(3)*m(3)+2*m(3)*m(2))*s2*c2*l(2)*l(2)*l(1)*l(1);
M11_Q2 = -(m(2)+2*m(3))*s2*l(2)*l(1);

% Inverse of Mass matrix M = W;
W = [M(2,2) -M(1,2); -M(1,2) M(1,1)]/dtM;

% Partial of W w.r.t Q2
W_Q2(1,1) = -1/dtM^2*M(2,2)*dtM_Q2;
W_Q2(1,2) = 1/dtM^2*M(1,2)*dtM_Q2-1/dtM*M11_Q2/2;
W_Q2(2,1) = 1/dtM^2*M(1,2)*dtM_Q2-1/dtM*M11_Q2/2;
W_Q2(2,2) = -1/dtM^2*M(1,1)*dtM_Q2+1/dtM*M11_Q2;

% Generalized force in Dynamic EQuation
F(1,1) = -((-m(2)/2-m(3))*Q(4)*Q(4)+(-m(2)-2*m(3))*Q(4)*Q(3))*l(2)*l(1)*s2+U(1);
F(2,1) = -(m(3)+m(2)/2)*Q(3)*Q(3)*l(2)*l(1)*s2+U(2);

% Gradient of {F} w.r.t {Q}
FQ(1,1) = 0;
FQ(1,2) = -((-m(2)/2-m(3))*Q(4)*Q(4)+(-m(2)-2*m(3))*Q(4)*Q(3))*l(2)*l(1)*c2;
FQ(1,3) = -(-m(2)-2*m(3))*Q(4)*l(2)*l(1)*s2;
FQ(1,4) = -(2*(-m(2)/2-m(3))*Q(4)+(-m(2)-2*m(3))*Q(3))*l(2)*l(1)*s2;
FQ(2,1) = 0;
FQ(2,2) = -(m(3)+m(2)/2)*Q(3)*Q(3)*l(2)*l(1)*c2;
FQ(2,3) = -2*(m(3)+m(2)/2)*Q(3)*l(2)*l(1)*s2;
FQ(2,4) = 0;

% Gradient of First order function of the dynamic (equation) constraints
FF_Q = [zeros(2,2), eye(2,2); [[zeros(2,1), W_Q2*F, zeros(2,2)]+W*FQ]];

Cdot(1:4,1) = -[C'*FF_Q*p]';
```

A.1.3 simplePlanar

```
function [Phi, const] = simplePlanar(U, nodes, A, B)
% Function for solving the dynamics of the
% planar 2-link manipulator for an optimization routine
% Uses a Runge-Kutta integrator from the NAG library
% USAGE: [Final_time, Contraints] = simplePlanar(Y, nodes)
% Where Y is m*nodes+1 vector of m controls concatenated and
% The final time parameter is the last entry
% nodes are the number of nodes in control mesh
% Ajay Seth

% Fist Identify global variables required by ODE solver
global ODE_x_end ODE_step_size ODE_steps ODE_out ODE_aux_fun
% ODE_aux_fun is an auxilliar function that ODEoutput calls if it exists

% Parameters required by State, Costate and gradient equations
global Tau_time Q_states Controls Final_time N_nodes Q_states

output ='ODEoutput';
fcn   ='planarNAG';

ODE_aux_fun = [];
N_nodes = nodes;

% Integration tolerance
tol = 1e-5;

%Penalty paramter
Rho = 1000;

% Nomalized Control scalings and inetgration time scaling
Umax = [25; 9];
l = [0.4; 0.25];
%Final State of end-effector
Xf = 0; Yf = l(1)+l(2);

% Number of controls
m = 2;
```

```
Controls = [Umax(1)*U(1:nodes),Umax(2)*U(nodes+1:2*nodes)];
Final_time = U(m*nodes+1);

% Normalized time, Tau
tau_end = 1;
dTau = tau_end/(nodes-1);
Tau_time = 0:dTau:tau_end;

% Starting time and initial state
tau = 0.0;
Q = zeros(4,1);

%tic;
% Set globals for ODE solver
ODE_x_end = tau_end;
ODE_steps = nodes-2;
ODE_step_size = dTau;

% Global matrix for trajectory output
ODE_out = [];

% Use the NAG RK integrator
[tau,Q,tol,ifail] = d02bbf(tau,tau_end,Q,tol,fcn,output);
%toc;
% Take global output and return to local variables
tau = ODE_out(:,1);  Q = ODE_out(:,2:5);
%animateAngles(Q(:,1:2),1);

% Compute trig functions at final state
c1 = cos(Q(nodes,1));   c12 = cos(Q(nodes,1)+Q(nodes,2));
s1 = sin(Q(nodes,1));   s12 = sin(Q(nodes,1)+Q(nodes,2));

% Evaluate the final state violations in order to evaluate Phi
pen = [l(1)*c1+l(2)*c12-Xf; l(1)*s1+l(2)*s12-Yf; Q(nodes,3); Q(nodes,4)];
Phi = (Rho*pen'*pen+Final_time^2); % Rho is a scalar penalty value 2000-
const= []; % Do uncostrained problem for now
Q_states = Q;
```

A.1.4 planarGrads

```
function [gJ, gC] = planarGrads(Y,A,B,nodes)
% Gradient of the objective J for a planar 2-link manipulator
% with controls and final time paramater
% Y = {U1o, ..., U1n, U2o,...,U2n, P}
% USAGE: [gJ, gC] = planarGrads(Y, nodes)
% Ajay Seth

% Fist Identify global variables required by ODE solver
global ODE_x_end ODE_step_size ODE_steps ODE_out ODE_aux_fun
% ODE_aux_fun is an auxilliar function that ODEoutput calls if it exists

% Parameters required by State, Costate and gradient equations
global Tau_time Q_states Controls Final_time N_nodes
global ControlVariational ParameterVariational PlanarGradient

Rho = 1000;
l = [0.4; 0.25];
%Final State of end-effector
Xf = 0; Yf = l(1)+l(2);

output ='ODEoutput';
fcn = 'costatesNAG';
% Integration tolerance
tol = 1e-5;

Q = Q_states;

tbk_end = 0; tbk =1;

% Compute trig functions at final state
c1 = cos(Q(nodes,1));   c12 = cos(Q(nodes,1)+Q(nodes,2));
s1 = sin(Q(nodes,1));   s12 = sin(Q(nodes,1)+Q(nodes,2));

% Evaluate the final time costates for integration from Phi_Q at Tf
C=2*Rho*[(l(1)*c1+l(2)*c12-Xf)*(-l(1)*s1-l(2)*s12)+(l(1)*s1+l(2)*s12-Yf)*(l(1)*c1+l(2)*c12
         (l(1)*c1+l(2)*c12-Xf)*(-l(2)*s12)+(l(1)*s1+l(2)*s12-Yf)*l(2)*c12;
         Q(nodes,3);
         Q(nodes,4)];
```

```
% Costate equations are dependant on the states
Q_states = Q;

dTau = ODE_step_size;
% Set globals for ODE solver
ODE_x_end = tbk_end;
ODE_steps = nodes-2;
ODE_step_size = -dTau;
% Compute variational functionals simultaneosly with integration
ODE_aux_fun = 'planarHuHp';

% Reset Global matrix for costate trajectory output
ODE_out = [];  ControlVariational = []; ParameterVariational =[];
PlanarGradient =[];

[tbk,C,tol,ifail] = d02bbf(tbk,tbk_end,C,tol,fcn,output);

% Compute Gradient dCost/dU
dCost_dU = ControlVariational*dTau;
dCost_dU([1,nodes],:) = 0.5*dCost_dU([1,nodes],:); %Half end points

% Compute Gradient dCost/dp
Hp = ParameterVariational;
dCost_dp = 2*Final_time+dTau*(sum(Hp)-(Hp(1)+Hp(nodes))/2);

% Overal design variable gradient
gJ = [dCost_dU(1:nodes,1)',dCost_dU(1:nodes,2)', dCost_dp];
gC = [];
```

A.2 Upper-Limb Biomechanical Model

A.2.1 DynaFlex Input File: BioArm.graph

```
# bioArm.graph
# Input file for a biomechanical arm with redundant actuators. This arm
# is driven by 5 SDA elements accross two joints.

# ITEM ONE: number of nodes and edges in the system graph
```

```
NOofedges:=28;
NOofnodes:=18;
Datum:=1;
# Note that Datum stands for the ground node.

edge[1]:=table([(1)=Y,
                (2)=[1,16],
                (3)=AE_R,
                (4)=table([coords=[r1[x],r1[y],0] ])
               ]);

edge[2]:=table([(1)=Y,
                (2)=[3,2],
                (3)=AE_R,
                (4)=table([coords=[r2,0,0] ])
               ]);

edge[3]:=table([(1)=Y,
                (2)=[3,4],
                (3)=AE_R,
                (4)=table([coords=[r3,0,0] ])
               ]);

edge[4]:=table([(1)=Y,
                (2)=[6,5],
                (3)=AE_R,
                (4)=table([coords=[r4,0,0] ])
               ]);

edge[5]:=table([(1)=Y,
                (2)=[6,7],
                (3)=AE_R,
                (4)=table([coords=[r5,0,0] ])
               ]);

edge[6]:=table([(1)=Y,
                (2)=[3,12],
                (3)=AE_R,
                (4)=table([coords=[r6[x],r6[y],0] ])
               ]);
```

```
edge[7]:=table([(1)=Y,
                (2)=[3,13],
                (3)=AE_R,
                (4)=table([coords=[r7[x],r7[y],0] ])
               ]);

edge[8]:=table([(1)=Y,
                (2)=[1,17],
                (3)=AE_R,
                (4)=table([coords=[r8[x],r8[y],0] ])
               ]);

edge[9]:=table([(1)=Y,
                (2)=[6,10],
                (3)=AE_R,
                (4)=table([coords=[r9[x],r9[y],0] ])
               ]);

edge[10]:=table([(1)=Y,
                 (2)=[6,18],
                 (3)=AE_R,
                 (4)=table([coords=[r10[x],r10[y],0] ])
                ]);

edge[11]:=table([(1)=Y,
                 (2)=[3,11],
                 (3)=AE_R,
                 (4)=table([coords=[r11[x],r11[y],0] ])
                ]);

edge[12]:=table([(1)=Y,
                 (2)=[3,14],
                 (3)=AE_R,
                 (4)=table([coords=[r12[x],r12[y],0] ])
                ]);

edge[13]:=table([(1)=Y,
                 (2)=[6,9],
                 (3)=AE_R,
```

```
              (4)=table([coords=[r13[x],r13[y],0] ])
             ]);

edge[28]:=table([(1)=Y,
              (2)=[1,15],
              (3)=AE_R,
              (4)=table([coords=[r28[x],r28[y],0] ])
             ]);

edge[14]:=table([(1)=Y,
              (2)=[1,2],
              (3)=JE,
              (4)=RV
             ]);

edge[15]:=table([(1)=Y,
              (2)=[4,5],
              (3)=JE,
              (4)=RV
             ]);

edge[16]:=table([(1)=Y,
              (2)=[7,8],
              (3)=JE,
              (4)=WELD
             ]);

edge[17]:=table([(1)=N,
              (2)=[1,3],
              (3)=BE_R,
              (4)=table([inert=[[0,0,0],
                                [0,0,0],
                                [0,0,J1]],
                         mass=m1 ])
             ]);

edge[18]:=table([(1)=N,
              (2)=[1,6],
              (3)=BE_R,
              (4)=table([inert=[[0,0,0],
```

```
                                [0,0,0],
                                [0,0,J2]],
                           mass=m2 ])
              ]);

edge[19]:=table([(1)=N,
                 (2)=[1,8],
                 (3)=BE_R,
                 (4)=table([inert=[[0,0,0],
                                [0,0,0],
                                [0,0,J3]],
                           mass=m3 ])
                ]);

edge[20]:=table([(1)=N,
                 (2)=[16,12],
                 (3)=FDE,
                 (4)=table([type=LSD,
                           character=(k20,d20,f20(t)) ])
               ]);

edge[21]:=table([(1)=N,
                 (2)=[17,10],
                 (3)=FDE,
                 (4)=table([type=LSD,
                           character=(k21,d21,f21(t)) ])
               ]);

edge[22]:=table([(1)=N,
                 (2)=[11,18],
                 (3)=FDE,
                 (4)=table([type=LSD,
                           character=(k22,d22,f22(t)) ])
               ]);

edge[23]:=table([(1)=N,
                 (2)=[14,9],
                 (3)=FDE,
                 (4)=table([type=LSD,
                           character=(k23,d23,f23(t)) ])
```

```
            ]);

edge[24]:=table([(1)=N,
                 (2)=[15,13],
                 (3)=FDE,
                 (4)=table([type=LSD,
                            character=(k24,d24,f24(t)) ])
                ]);

edge[25]:=table([(1)=N,
                 (2)=[1,3],
                 (3)=FDE,
                 (4)=table([type=PD,
                            fy=W1,
                            force=gl  ])
                ]);

edge[26]:=table([(1)=N,
                 (2)=[1,6],
                 (3)=FDE,
                 (4)=table([type=PD,
                            fy=W2,
                            force=gl  ])
                ]);

edge[27]:=table([(1)=N,
                 (2)=[1,8],
                 (3)=FDE,
                 (4)=table([type=PD,
                            fy=W3,
                            force=gl  ])
                ]);

Iedge:=[];
```

A.2.2 bioArm Input for Dynopt

The equations of motion produced by DynaFlex were first simplified in Maple. To further simplify the problem the parallel elastic and viscous elements were assigned 0 valued coefficients.

The resulting equations were implemented in `bioArm`. The minimum stress objective is also encoded herein. DYNOPT converts the file into C and compiles and links with `dyn_sqp.c` and `math_opt.c` to produce the standalone executable `bioArm.exe`.

```
# bioArm input file for Dynopt
# Planar Biomechanical Model of a 2DOF  Arm with 5 muscle actuators
# Now including the joint stress objective
# Equations generated in Maple from the following files:
# 1. bioArm_gen.mws + bioArm.graph => bioArm.raw               {unsimplified}
# 2. bioArm_simp.mws + bioArm.par => (M)ass, (G)amma, (F)orces {w/ symb subs}
# The final symbolic equations are implemented herein
# [M]{dotdot{q}}+{F} = 0; => {x3,x4}^T= inv[M]*{-F}!
# Ajay Seth

state x1 x2 x3 x4
control u1 u2 u3 u4 u5

method = dyn_sqp
input_file = bioArm2Guess
output_file = bioArm2
nodes = 21
epsilon = 1.0e-5

# Optimization parameters, i.e. p is time parameterization
parameter p

# Initial and terminal  variables
real X1_0 X2_0 X1_f X2_f
# Maximum muscle forces
real FM1 FM2 FM3 FM4 FM5
# Constants
real pi g alpha beta

# Trig functions
real c1 s1 c2 s2 c12 s12

# Masses and inertias, Mass matrix, and generalized forces
real eff_load m1 m2 m3 m23 J1 J2 J3 detM M_11 M_12 M_21 M_22 F1 F2
# General expresssions substitued into F including muscle forces
```

```
real G1 G2 G4 G5 G6 f1 f2 f3 f4 f5
# Rigid arm elements
real r1x r1y r2x r2y r3x r3y r4x r4y r5x r5y r6x r6y r7x r7y r8x r8y
real r9x r9y r10x r10y r11x r11y r12x r12y r13x r13y r28x r28y
real rc1o1 rc2o2 l1 l2 lo1 lo2 lo3 lo4 lo5
real vmax v1 v2 v3 v4 v5 v1x v1y v2x v2y v3x v3y v4x v4y v5x v5y

# Connection vectors from proximal to distal and their unit vectors
real R1x R1y R2x R2y R3x R3y R4x R4y R5x R5y R1 R2 R3 R4 R5
real d1x d1y d2x d2y d3x d3y d4x d4y d5x d5y

# Variables for computing Cost functional, L
real a1x a1y ajx ajy a2x a2y a3x a3y dx1 dx2 dx3 dx4
real A_X A_Y B_X B_Y Ax Ay Bx By

eff_load = 0.0
# or 5.0
pi = 3.14159265358975
g = -9.81

X1_0 = -pi/1.8
X2_0 = pi/10
X1_f = -pi/2.1
X2_f = 0.9*pi

# Muscle parameters: rest lengths and maximu velocity and shaping parms
lo1 = 0.15;   lo2 = 0.2845;   lo3 = 0.3257;   lo4 = 0.31;   lo5 = 0.065;  vmax = 20
alpha = 0.5;  beta = 0.5

# Write arm parameter values here
r1x = 0.005;    r1y = 0.03;     r2x = -0.155;    r2y = 0.0;      r3x = 0.175;     r3y = 0.0
r4x = -0.11;    r4y = 0.0;      r5x = 0.17;     r5y = 0;        r6x = -0.02;    r6y = 0.015
r7x = r2x+0.01; r7y = -0.0567;  r8x = 0.005;    r8y = 0.02;     r9x = -0.085;   r9y = 0.007
r10x = r5x;     r10y = 0.01;    r11x = 0.12;    r11y = 0.015;   r12x = -0.135;  r12y= -0.015
r13x = -0.135;  r13y = -0.019;  r28x = -0.065;  r28y = 0.0
rc1o1 = 0.15;   rc2o2 = 0.11;   l1 = 0.33;      l2 = 0.28

m1 = 2.7;    m2 = 1.8;     m3 = 0.4+eff_load;   m23 = m2+m3
J1 = (1/12)*m1*(3*pow(2*r11y,2)+pow(r3x-r2x,2))
J2 = (1/12)*m2*(3*pow(1.5*r10y,2)+pow(r5x-r4x,2))
```

```
J3 = (1/2)*m3*pow(0.004,2)

# Maximum isometric force
FM1 = 1800;     FM2 = 1500;     FM3 = 600;      FM4 = 2600;     FM5 = 4400

initial_condition:

      x1 = X1_0
      x2 = X2_0
      x3 = 0.0
      x4 = 0.0

dynamic_equation:

#Evaluate trig functions
c1 = cos(x1);   c2 = cos(x2);   c12 = cos(x1+x2)
s1 = sin(x1);   s2 = sin(x2);   s12 = sin(x1+x2)

# Get global vector connecting origin to insertion points
R1x = c1*(-r2x+r6x)-s1*(-r2y+r6y)-r1x
R1y = s1*(-r2x+r6x)+c1*(-r2y+r6y)-r1y
R2x = c1*(-r2x+r3x)-s1*(-r2y+r3y)+c12*(-r4x+r9x)-s12*(-r4y+r9y)-r8x
R2y = s1*(-r2x+r3x)+c1*(-r2y+r3y)+s12*(-r4x+r9x)+c12*(-r4y+r9y)-r8y
R3x = c1*(-r2x+r3x)-s1*(-r2y+r3y)+c12*(-r4x+r10x)-s12*(-r4y+r10y)-(c1*(-r2x+r11x)-s1*(-r2y+r11y))
R3y = s1*(-r2x+r3x)+c1*(-r2y+r3y)+s12*(-r4x+r10x)+c12*(-r4y+r10y)-(s1*(-r2x+r11x)+c1*(-r2y+r11y))
R4x = c1*(-r2x+r3x)-s1*(-r2y+r3y)+c12*(-r4x+r13x)-s12*(-r4y+r13y)-(c1*(-r2x+r12x)-s1*(-r2y+r12y))
R4y = s1*(-r2x+r3x)+c1*(-r2y+r3y)+s12*(-r4x+r13x)+c12*(-r4y+r13y)-(s1*(-r2x+r12x)+c1*(-r2y+r12y))
R5x = c1*(-r2x+r7x)-s1*(-r2y+r7y)-r28x
R5y = s1*(-r2x+r7x)+c1*(-r2y+r7y)-r28y

# Their magnitudes (lengths)
R1 = sqrt(R1x*R1x+R1y*R1y);     R2 = sqrt(R2x*R2x+R2y*R2y);     R3 = sqrt(R3x*R3x+R3y*R3y)
R4 = sqrt(R4x*R4x+R4y*R4y);     R5 = sqrt(R5x*R5x+R5y*R5y)

# Corresponding unit (direction) vectors
d1x = R1x/R1;  d1y = R1y/R1;  d2x = R2x/R2;  d2y = R2y/R2;  d3x = R3x/R3;  d3y = R3y/R3
d4x = R4x/R4;  d4y = R4y/R4;  d5x = R5x/R5;  d5y = R5y/R5

# Compute the velocity of insertion points relative to origin points
v1x = (-s1*(-r2x+r6x)-c1*(-r2y+r6y))*x3
```

```
v1y = (c1*(-r2x+r6x)-s1*(-r2y+r6y))*x3
v2x = (-s1*(-r2x+r3x)-c1*(-r2y+r3y))*x3+(-s12*(-r4x+r9x)-c12*(-r4y+r9y))*(x3+x4)
v2y = c1*x3*(-r2x+r3x)-s1*x3*(-r2y+r3y)+c12*(x3+x4)*(-r4x+r9x)-s12*(x3+x4)*(-r4y+r9y)
v3x = -s1*x3*(-r2x+r3x)-c1*x3*(-r2y+r3y)-s12*(x3+x4)*(-r4x+r10x)-c12*(x3+x4)*(-r4y+r10y)+ @
       s1*x3*(-r2x+r11x)+c1*x3*(-r2y+r11y)
v3y = c1*x3*(-r2x+r3x)-s1*x3*(-r2y+r3y)+c12*(x3+x4)*(-r4x+r10x)-s12*(x3+x4)*(-r4y+r10y)- @
      c1*x3*(-r2x+r11x)+s1*x3*(-r2y+r11y)
v4x = -s1*x3*(-r2x+r3x)-c1*x3*(-r2y+r3y)-s12*(x3+x4)*(-r4x+r13x)-c12*(x3+x4)*(-r4y+r13y)+ @
       s1*x3*(-r2x+r12x)+c1*x3*(-r2y+r12y)
v4y = c1*x3*(-r2x+r3x)-s1*x3*(-r2y+r3y)+c12*(x3+x4)*(-r4x+r13x)-s12*(x3+x4)*(-r4y+r13y)- @
      c1*x3*(-r2x+r12x)+s1*x3*(-r2y+r12y)
v5x = -s1*x3*(-r2x+r7x)-c1*x3*(-r2y+r7y)
v5y = c1*x3*(-r2x+r7x)-s1*x3*(-r2y+r7y)

# Project velcities on to connecting lines to get contraction velocity +ve = lengthening
v1 = v1x*d1x+v1y*d1y
v2 = v2x*d2x+v2y*d2y
v3 = v3x*d3x+v3y*d3y
v4 = v4x*d4x+v4y*d4y
v5 = v5x*d5x+v5y*d5y

# Determine individual control forces
f1 = (FM1*u1)*(1-pow(abs(R1-lo1)/lo1,alpha))*pow(abs(vmax+v1)/vmax, beta);
f2 = (FM2*u2)*(1-pow(abs(R2-lo2)/lo2,alpha))*pow(abs(vmax+v2)/vmax, beta);
f3 = (FM3*u3)*(1-pow(abs(R3-lo3)/lo3,alpha))*pow(abs(vmax+v3)/vmax, beta);
f4 = (FM4*u4)*(1-pow(abs(R3-lo4)/lo4,alpha))*pow(abs(vmax+v4)/vmax, beta);
f5 = (FM5*u5)*(1-pow(abs(R4-lo5)/lo5,alpha))*pow(abs(vmax+v5)/vmax, beta);

# Define the Mass Matrix
M_11 = m2*(l1*l1+r4x*r4x+2.0*r4x*l1*c2)+m3*(l2*l2+l1*l1+2.0*l1*l2*c2)+m1*r2x*r2x+J1+J2+J3
M_12 = m3*(l1*l2*c2+l2*l2)+m2*(r4x*l1*c2+r4x*r4x)+J2+J3
M_21 = M_12
M_22 = m2*r4x*r4x+m3*l2*l2+J2+J3
detM = M_11*M_22-M_12*M_21

# First solve for the sub-expressions used to simplify the Force equations
G1 = sqrt((2*s1*r2x-2*s1*r6x-2*c1*r6y)*r1y+(2*c1*r2x+2*s1*r6y-2*c1*r6x)*r1x+ @
          (r6x*r6x+r6y*r6y)-2*r2x*r6x+(r1x*r1x+r1y*r1y)+r2x*r2x)
G2 = sqrt((-2*r2x-2*c1*r28x-2*s1*r28y)*r7x+(-2*c1*r28y+2*s1*r28x)*r7y+(r28x*r28x+r28y*r28y)+ @
            2*c1*r2x*r28x+2*s1*r2x*r28y+(r7x*r7x+r7y*r7y)+r2x*r2x)
```

```
G4 = sqrt((-2*r4x-2*r11y*s2+2*r3x*c2-2*r11x*c2)*r10x+(-2*r11y*c2-2*r3x*s2+2*r11x*s2)*r10y+ @
         (r11x*r11x+r11y*r11y)+2*r4x*r11y*s2+(2*r4x*c2-2*r3x)*r11x+(r10x*r10x+r10y*r10y)+ @
          r4x*r4x+r3x*r3x-2*r3x*r4x*c2)
G5 = sqrt((2*r4x*s2-2*r13y*c2-2*r13x*s2)*r12y+(2*r4x*c2+2*r13y*s2-2*r3x-2*r13x*c2)*r12x+ @
          (r13x*r13x+r13y*r13y)+(-2*r4x+2*r3x*c2)*r13x-2*r3x*r13y*s2+(r12x*r12x+r12y*r12y)+ @
           r4x*r4x+r3x*r3x-2*r3x*r4x*c2)
G6 = sqrt((2*s12*r4x+2*s1*r2x-2*s12*r9x-2*c12*r9y-2*s1*r3x)*r8y+ @
          (2*c1*r2x+2*s12*r9y+2*c12*r4x-2*c12*r9x-2*c1*r3x)*r8x+(r9x*r9x+r9y*r9y)+ @
          (-2*r2x*c2-2*r4x+2*r3x*c2)*r9x+(2*r2x*s2-2*r3x*s2)*r9y+(r8x*r8x+r8y*r8y)+ @
            2*r2x*r4x*c2-2*r2x*r3x+r2x*r2x+r4x*r4x-2*r3x*r4x*c2+r3x*r3x)

F1 = ((-r7x+r2x)*(r28y*c1-r28x*s1)+(r28x*c1+r28y*s1)*r7y)*f5/G2- @
      x4*s2*(2.0*x3+x4)*(r2x-r3x)*(r4x*m23-m3*r5x)+ @
      g*((c12*r4x+c1*(r2x-r3x))*m23+c1*m1*r2x-m3*c12*r5x)+ @
     (((-r9x+r4x)*r8y+r8x*r9y)*c12+((r9x-r4x)*r8x+r8y*r9y)*s12- @
      (-r8y*c1+r8x*s1)*(r2x-r3x))*f2/G6+ @
      ((r1x*r6y+r1y*(r2x-r6x))*c1+(-r1x*(r2x-r6x)+r1y*r6y)*s1)*f1/G1

F2 = f4*((r12x*r13x-r3x*r13x+r12y*r13y+r4x*r3x-r12x*r4x)*s2+ @
     c2*(r12y*r4x-r3x*r13y+r12x*r13y-r12y*r13x))/G5+ @
     x3*x3*s2*(r2x-r3x)*(r4x*m23-m3*r5x)+g*c12*(r4x*m23-m3*r5x)+ @
     ((r11y*(r4x-r10x)+r10y*(-r3x+r11x))*c2+(r11y*r10y-(-r3x+r11x)*(r4x-r10x))*s2)*f3/G4+ @
     (((-r9x+r4x)*r8y+r8x*r9y)*c12+((r9x-r4x)*r8x+r8y*r9y)*s12- @
     (r2x-r3x)*((-r9x+r4x)*s2-r9y*c2))*f2/G6

# Remember Dynaflex: [M]{qdotdot}+{F_col}=0 thus, {qdotdot}=inv([M])*{-F_col}
F1 = -F1
F2 = -F2

      dx1 = x3*p
      dx2 = x4*p
      dx3 = (1/detM)*( M_22*F1-M_12*F2)*p
      dx4 = (1/detM)*(-M_21*F1+M_11*F2)*p

      ddt x1 = dx1
      ddt x2 = dx2
      ddt x3 = dx3
      ddt x4 = dx4

cost_functional:
```

```
        # Get the accelerations of the centre of masses in global coords
        a1x = (c1*(-dx1*dx1)-s1*dx3)*rc1o1
        a1y = (s1*(-dx1*dx1)+c1*dx3)*rc1o1
        ajx = (c1*(-dx1*dx1)-s1*dx3)*l1
        ajy = (s1*(-dx1*dx1)+c1*dx3)*l1
        a2x = ajx+(c12*(-dx2*dx2)-s12*dx4)*rc2o2
        a2y = ajy+(s12*(-dx2*dx2)+c12*dx4)*rc2o2
        a3x = ajx+(c12*(-dx2*dx2)-s12*dx4)*l2
        a3y = ajy+(s12*(-dx2*dx2)+c12*dx4)*l2

        # Compute joint reaction forces in global coords
        # Muscle force is is in the opposite direction of connection vector
        B_X = m2*a2x+m3*a3x-f2*(-d2x)-f3*(-d3x)-f4*(-d4x)
        B_Y = m2*(a2y-g)+m3*(a3y-g)-f2*(-d2y)-f3*(-d3y)-f4*(-d4y)
        # Muscles that act on both segments act in the positive direction on their origin body
        A_X = m1*a1x-(-B_X)-f1*(-d1x)-f5*(-d5x)-f3*d3x-f4*d4x
        A_Y = m1*(a1x-g)-(-B_Y)-f1*(-d1y)-f5*(-d5y)-f3*d3y-f4*d4y

        # Rotate back into local body coordinates
        Ax = c1*A_X+s1*A_Y
        Ay = -s1*A_X+c1*A_Y
        Bx = c12*B_X+s12*B_Y
        By = -s12*B_X+c12*B_Y

        initial_time = 0.0
        final_time = 1.0
        L = 1.0e-5*pow(abs(Ax)-Ax,2)+4.0e-5*pow(Ay,2)+1.5e-5*pow(abs(Bx)-Bx,2)+1.0e-4*pow(By,2)

terminal_condition:

        phi = p
        psi = x1-X1_f
        psi = x2-X2_f
        psi = x3
        psi = x4

inequality_constraint:

        d = -p
        d = (x2-pi)*x2
```

```
d = -(1-u1)*u1
d = -(1-u2)*u2
d = -(1-u3)*u3
d = -(1-u4)*u4
d = -(1-u4)*u5
```

Appendix B

Project Files: Descriptions and Locations

B.1 Optimization Methods

All files for performing the optimization are available on the Silicon Graphics 2xZ workstation, "real", in the Motion Research Group laboratory. The locations presented in Table B.1 extend from my home directroy '/u/aseth/'.

Filename	Description	Location
genetic.m	genetic algorithm matlab	Arm/GeneticAlgorithm/
GA_DYNOPT	PERL script for parsing model file	Numerical/Dynopt/GA_Dynopt/dynopt-script
GA_PRINTC_CODE	PERL script to generate C code	Numerical/Dynopt/GA_Dynopt/dynopt-script
dyn_sqp.c	C source for DYNOPT routines	Numerical/Dynopt/GA_Dynopt/dynopt-source
math_opt.c	DYNOPT C utility functions	Numerical/Dynopt/GA_Dynopt/dynopt-source

Table B.1: Files implementing the optimzation methods

Filename	Description
makeSegs.m	Form local segment definitions from calibration data
pos2Rot.m	Calculate rotations for each segment from marker coordinates
getJointCentres.m	Calculate the joint centre locations on respective bodies
reconTrial.m	Compute the relative rotations to generate an animation of a trial
reconTaskStats.m	Generate normalized kinematics with mean and s.d. angular trajectories
planarAnimation.m	Planarize rotations and compute the joint trajectories
emgStats.m	Process EMG and generate mean and s.d. activations for a task

Table B.2: Collection data processing MATLAB functions

B.2 Collection Data Processing

All data and processing utilities are available on the Motion Research Group's laboratory PC.

B.2.1 Processing Methods

The MATLAB function files described in Table B.2 can be found in 'e:/ajay/ArmModel/Reconstruct/'.

B.2.2 Collection Data

The various marker and EMG data as well as post-processed data are described in Table B.3. These files are mainatined in sub directories of 'e:/ajay/ArmModel/Reconstruct/'.

File names	Sub-directory	Description
c#*.tsk	Trials	Raw marker data for tasks
c#*.cal	Trials	Calibration marker data
c#*.end	Trials	End calibration markers
o#*.tsk	Trials	Raw emg data for tasks
o#*.cal	Trials	EMG during calibration
o#*.end	Trials	EMG during end calibration
o#*.0mg	Trials	Background EMG
trial*.mat	CleanData	Filtered and interpolated marker data
T*_segments.mat	L_frames	Segment definitions for each task
rot*.mat	RotData	Global rotation matrices for each trial
reconT*_*.mat	Results	Relative rotations and key marker traces
T*_start_end.mat	Results	Start and end times of trials of a task

Table B.3: Data files generated by collection and processing