NASA/CR—2019-219488

Predictive Kinematic Analysis Methodologies for Use With OpenSim

Brad Humphreys
ZIN Technologies, Inc., Middleburg Heights, Ohio

September 2019

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/CR—2019-219488 DA–ANA–001

Predictive Kinematic Analysis Methodologies for Use With OpenSim

Brad Humphreys
ZIN Technologies, Inc., Middleburg Heights, Ohio

Prepared under Contract NNC14CA02C

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

September 2019

Level of Review: This material has been technically reviewed by NASA expert reviewer(s).

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/

Preface

This document, DA-ANA-001, the “Predictive Kinematics Analysis Methodology for Use with OpenSim,” has been prepared by ZIN Technologies, Inc. to support activities associated with the Digital Astronaut Project at NASA. This document describes the methodologies developed to predict kinematics using OpenSim software and mathematical optimization and techniques.

Contents

Predictive Kinematic Analysis Methodologies for Use With OpenSim

Brad Humphreys
ZIN Technologies, Inc.
Middleburg Heights, Ohio 44130

Abstract

This report focuses specifically on the development of a methodology to generate predictive kinematics in the absence of measured test data or the ability to test in the microgravity environment. The customer for this deliverable is the AEC project.

1.0 Introduction

The NASA Digital Astronaut Project (DAP) implements well-vetted computational models to predict and assess health and performance risks related to spaceflight, and to enhance countermeasure development. The DAP Musculoskeletal Modeling effort is developing computational models to inform exercise countermeasure development and to predict physical performance capabilities after a length of time in space. For example, integrated exercise device-biomechanical models can determine localized loading, which will be used as input to muscle and bone adaptation models to estimate the effectiveness of the exercise countermeasure. In addition, simulations of mission tasks can be used to estimate the astronaut's ability to perform the task after exposure to microgravity and after using various exercise countermeasures. The software package OpenSim (Stanford University, Palo Alto, CA) is being used to create the DAP biomechanical models, and its built-in muscle model is the starting point for the DAP muscle model.

During exploration class missions, such as to asteroids and Mars, astronauts will be exposed to reduced gravity for extended periods. Therefore, the crew must have access to exercise countermeasures that can maintain their musculoskeletal and aerobic health. Exploration vehicles may have very limited volume and power available to accommodate such capabilities, with even more restrictions than the International Space Station (ISS). The exercise devices flown on exploration missions must be designed to provide sufficient load must have the potential to enable the performance of various resistance and aerobic/anaerobic exercises, while meeting possible additional requirements of limited mass, volume and power. Given that it is not practical to manufacture and test (via ground, analog and/or flight test) all candidate devices, nor is it always possible to obtain data such as localized muscle and bone loading empirically, computational modeling can estimate the localized loading during various exercise modalities performed on a given device to help formulate exercise prescriptions and other operational considerations. With this in mind, the DAP is supporting the Advanced Exercise Concepts (AEC) Project, Exercise Physiology and Countermeasures (ExPC) laboratory and National Space Biomedical Research Institute (NSBRI) funded researchers by developing and implementing well-validated computational models of exercises with advanced exercise device concepts.

2.0 Background

Exercise conditions are often able to be fully simulated in a laboratory setting. Using motion capture and force measurements, kinematics (displacements/velocities) and kinetics (forces/moments) are able to be measured. When the laboratory conditions do not match the conditions in which the exercise will be performed, the available data must be extrapolated to predict the kinematics and kinetics. For example, attempting to understand the effects of microgravity on biomechanics is limited by the ability to perform testing in microgravity. Additionally, understanding the effects of changing the force-displacement profile of a conceptual exercise device, limits the conclusions that can be drawn about the biomechanics of using that particular device.

One means to investigate the effect of changing the device load-displacement pattern (free-weight, constant, inertial, resistive) is to utilize measured kinematics from testing a similar device to drive a modified device model, then comparing the difference in forces at the device-human interface. This method assumes that the device user does not change their movement kinematics due to the change in forces. If the kinematic change is not considered as an assumption, the device does generate a different force profile, and the self-induced forces (movement of the user's limbs' centers of mass) will be different.

A significant body of research exists related to predictive gait kinematics, such as the optimization techniques developed by many researchers in the fields of robotics and prosthetics. Instead of assuming that the exerciser does not change their kinematics in response to changes in the force profile, an assumption is made that the exerciser adjusts their kinematics in an optimal manner. One simple assumption is that the exerciser minimizes their torque at the anatomical joints over the movement duration (with torque squared and integrated to yield a rectified total). Optimization cost functions, such as minimization of the total integrated muscle activation (scaled by muscle volume) or metabolic cost are also utilized.

This document describes methods that have been developed to perform predictive kinematics in OpenSim. Using these methodologies, the effect of changing the exercise environment (e.g. device load-displacement, effect of vibration isolation systems and gravity) can be performed using the existing biomechanical models developed in OpenSim. Using this methodology, the relative effect on coupled kinematic and kinetic systems system can be evaluated when laboratory data are not available or for the evaluation of an exercise device concept. This methodology provides a consistent virtual subject that can indicate the effects on user exercise, given changes in the environment and exercise device.

Papers related to this topic are listed in the References section of this report (Refs. 1. to 10).

3.0 ARM28 OpenSim Model

To develop the methodology and techniques specific to interfacing with OpenSim, a two degree of freedom arm model with 8 muscles is being utilized (Figure 1). The Arm26 model is provided with OpenSim; this model has 2 degrees of freedom with 6 arm muscles (hence the 26 in the name of the model). Based on the objectives defined below, two additional muscles were added to the model: anterior and posterior deltoids. The new model has been named Arm28. The additional muscles allow for shoulder flexion and extension, as the Arm26 model muscles did not permit this capability. Limited efforts have been made in adjusting the muscles' parameters to be anatomically representative of a real arm. The muscles were adjusted and sized to permit the model to position the arm horizontally (parallel to the transverse plane) and perform a toss movement.

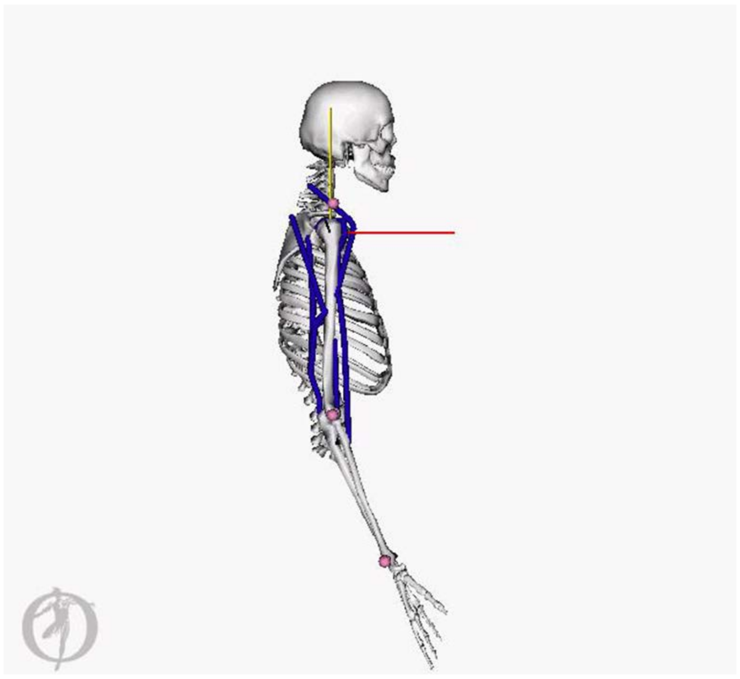

Figure 1.—Arm28 model, two degrees of freedom and eight muscle model.

The optimization goal of this model is a vertical ball toss: maximize the hand's velocity at a specified time for a vertical ball toss movement. No information is given about the muscle activation pattern. The goal of the optimization is to find the muscle activation pattern that achieves the maximum hand velocity after 2 sec.

Constraints for the model include:

1) The hand must be anterior to the torso at the final time.
2) The elbow cannot hyperextend.

The parameters to be varied are the muscles' activation values. Each muscles' activation pattern is modeled as a 5 point cubic spline. The first point in each muscle spline is the initial condition value and is fixed (static pose shown in Figure 1). The remaining four points are then varied by the optimization algorithm. With 8 muscles and 4 spline points to vary, there are 32 parameters to investigate.

4.0 Methodologies

4.1 Shooting Methodology Description

A shooting optimization methodology attempts to ascertain an optimal solution by a gradient search or an interior point solution. A set of parameters is adjusted to minimize the objective function; sometimes referred to as the cost function. Several software libraries exist to perform shooting optimization. For the work described here, the Interior Point Optimizer (IPOPT) (Ref. 11), and MATLAB's native *fmincon* solver have been tested. These software tools rely on the optimization problem to be structured as described below:

Objective Function.—A function that evaluates a set of parameters and provides for a numeric value (scalar) to be minimized. For the Arm28 model, the minimum value of the hand velocity after 2 seconds of movement is utilized. Optimization algorithms typically attempt to minimize an objective function. Therefore the hand velocity is multiplied by -1.

Constraint Function.—A function that evaluates the set of parameters and calculates a numeric set of constraint values (vector). For the Arm28 model, this function returns the final position of the hand and the minimum elbow angle across the entire simulation time.

Objective Gradient.—A function that provides the partial differential of the objective value with respect to each of the parameters. For the Arm28 model, this is the change in the hand velocity with the change in activation of each of the muscles. This is an *n*-by-1 vector, where *n* is the number of parameters. When applied to the Arm28model, *n* = 32.

Constraint Jacobian.—A function that provides the partial differential of the constraint Jacobian with respect to each of the input parameters. This is an *n*-by-*c* matrix, where *c* is the number of constraints When applied to the Arm28model, *c* = 32. The objective gradient and constraint Jacobian can be conceptualized as a linearization around an operating point. Each entry in the "operating point" or base set of parameters is perturbed, and the local derivative of the input is determined. If analytical functions are utilized (see Appendix B), this can occur very efficiently because the Jacobian can be symbolically predetermined functions and then evaluated during optimization. In OpenSim, this process must occur numerically. Each entry of the objective gradient and the constraint Jacobian requires the full simulation to run; in the case of the Arm28 model, this necessitates 32×2 or 64 simulations to compute the constraint Jacobian.

4.2 Simulation

OpenSim provides a high-level MATLAB Application Program Interface (API). Using this API, MATLAB is able to build, modify and control OpenSim models.

The OpenSim model simulation is performed using MATLAB's differential equation solvers. The states of the models are packaged into a plant model of the form:

$$\dot{q} = f(q,u)$$

Where q are the generalized coordinates of the model's joints and u are the muscle activations.
As shown in Figure 2, the Ordinary Differential Equations (ODE) solver then evaluates the OpenSim model at each time step. This allows for full control of the model states. It would also be possible to utilize OpenSim's forward dynamics tool in the MATLAB API, but it was decided to make use of the ODE functions, in order to maintain full control of the simulation. While control values (muscle activations) can be provided as a plant input, it was determined that the more efficient method is to use the OpenSim API to add a prescribed muscle controller with a cubic spline driving function to each muscle in the model. The spline functions are then updated with the values provided by IPOPT's choice of parameters at the beginning of each simulation/integration of the model.

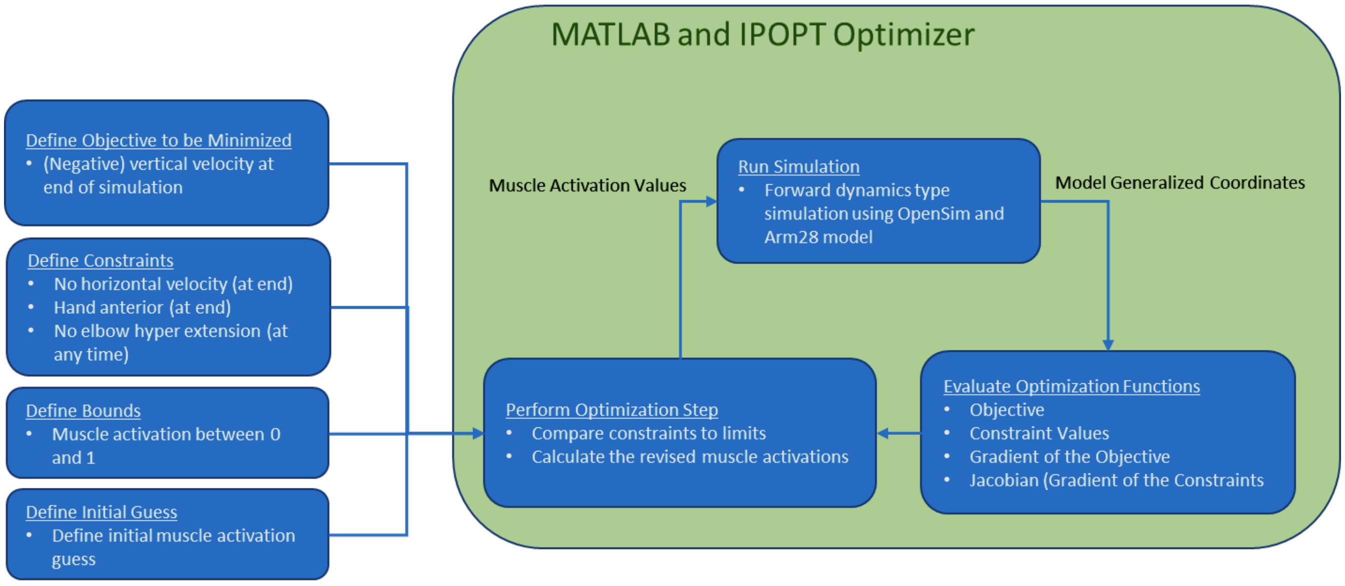
MATLAB and IPOPT Optimizer
Define Objective to be Minimized
• (Negative) vertical velocity at end of simulation
Define Constraints
• No horizontal velocity (at end)
• Hand anterior (at end)
• No elbow hyper extension (at any time)
Define Bounds
• Muscle activation between 0 and 1
Define Initial Guess
• Define initial muscle activation guess
Muscle Activation Values
Run Simulation
• Forward dynamics type simulation using OpenSim and Arm28 model
Model Generalized Coordinates
Perform Optimization Step
• Compare constraints to limits
• Calculate the revised muscle activations
Evaluate Optimization Functions
• Objective
• Constraint Values
• Gradient of the Objective
• Jacobian (Gradient of the Constraints

Figure 2.—OpenSim and IPOPT.

In order to verify the MATLAB function, an arbitrarily chosen arm raise movement was defined and used in OpenSim's Forward Kinematics and Computed Muscle Control (CMC) tool to generate muscle activation patterns. The muscle activations were then used as inputs to the MATLAB plant model of the OpenSim system, and the resulting displacement and velocities were calculated. As shown in Figure 3 through Figure 5, the results from OpenSim and MATLAB match fairly well. Due to a software limitation making it impossible to achieve identical initial conditions in the workflow utilized, the simulations do differ.

The duration of each simulation/integration is important in a numerical shooting algorithm as this determines the feasibility of using this methodology. Using the simpler Arm26 model (no deltoids) and 3 spline points per muscle, the convergence of the optimization can be seen in Figure 6 and Figure 7. Each step in these figures represents a full integration-simulation.

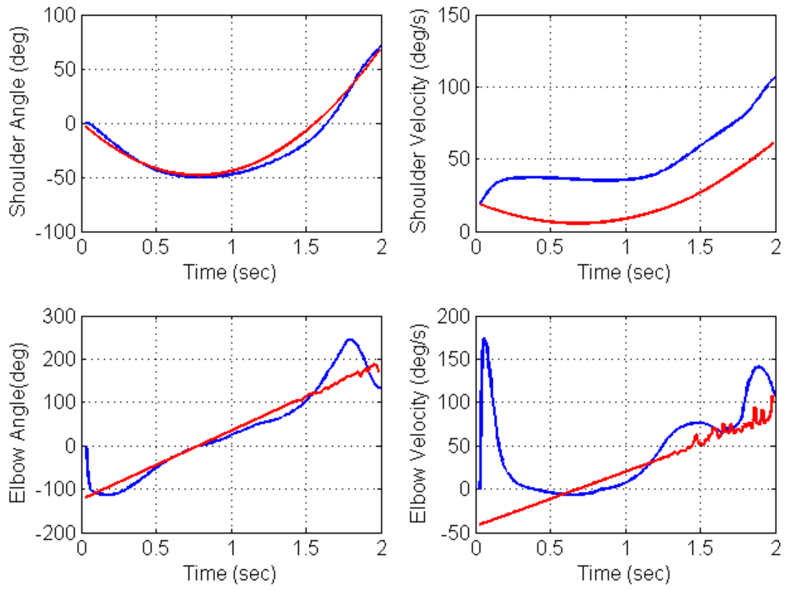

Figure 3.—Joint displacement and velocity as calculated by OpenSim (Blue) and MATLAB simulation (Red).

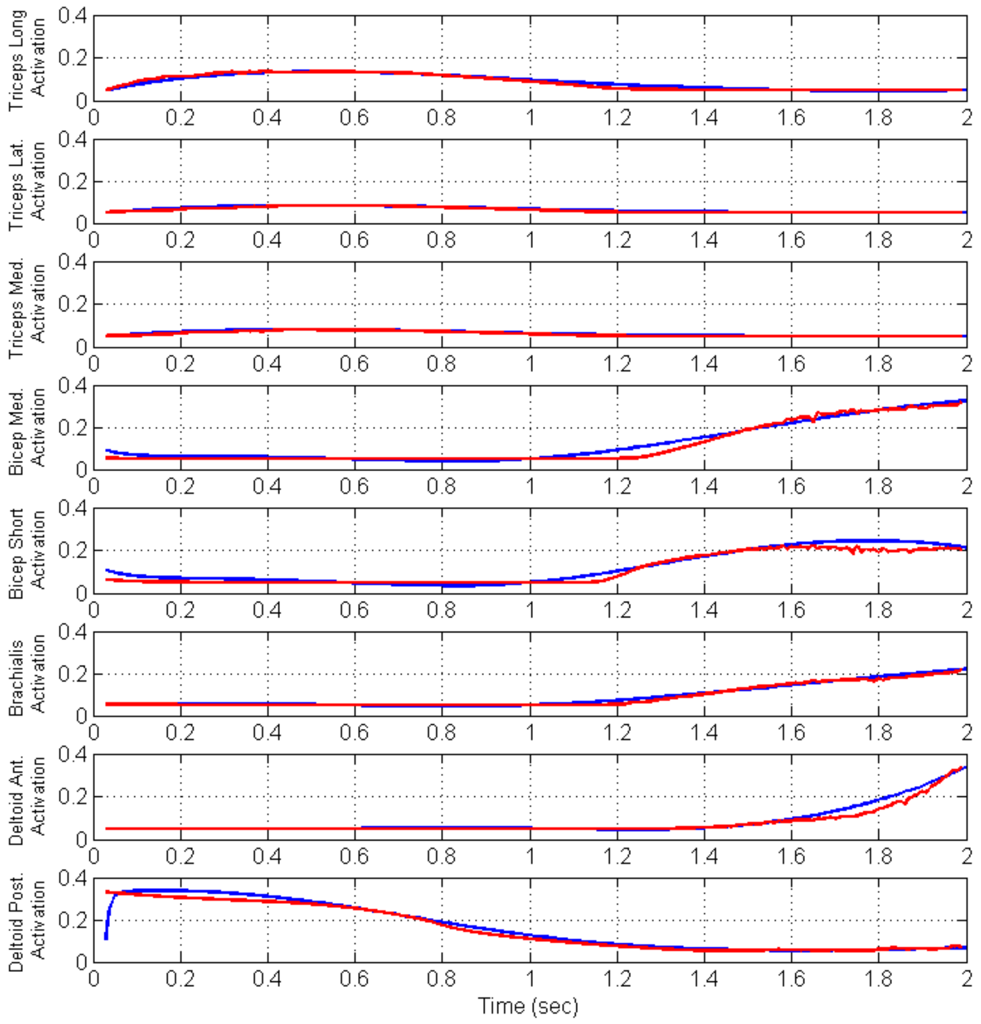

Figure 4.—Muscle activations as calculated by OpenSim (Blue) and MATLAB simulation (Red).

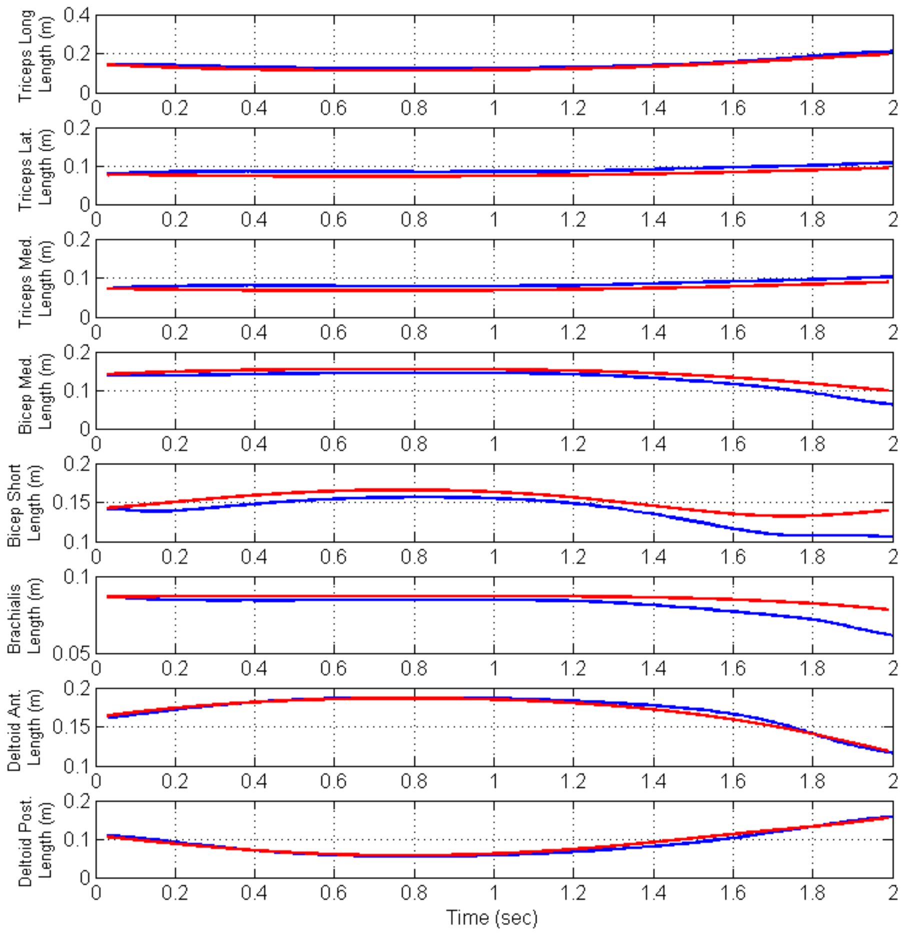

Figure 5.—Muscle lengths as calculated by OpenSim (Blue) and MATLAB simulation (Red).

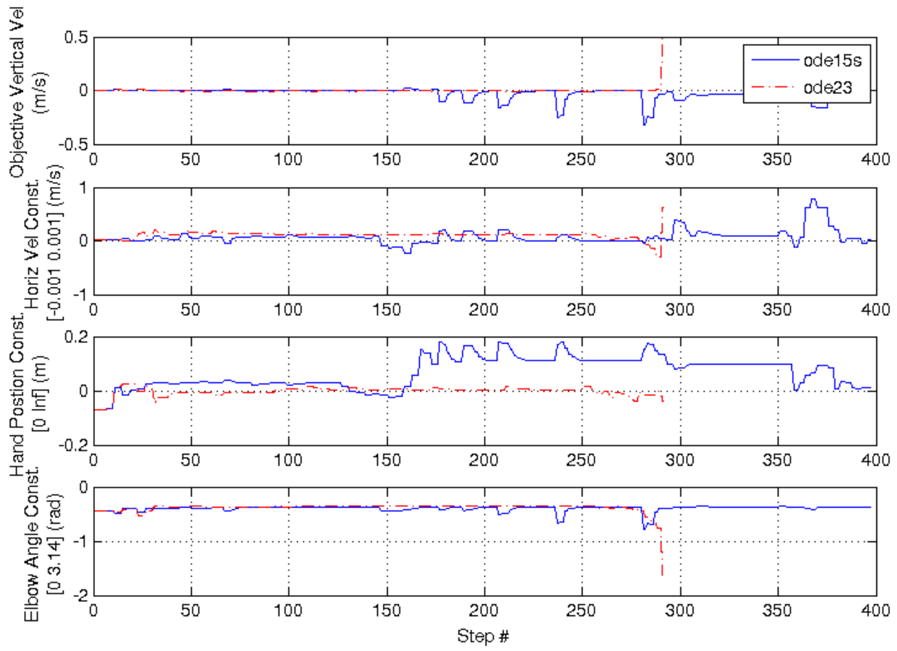

Figure 6.—Arm26 shooting convergence. Top is objective function and lower 3 are constraints.

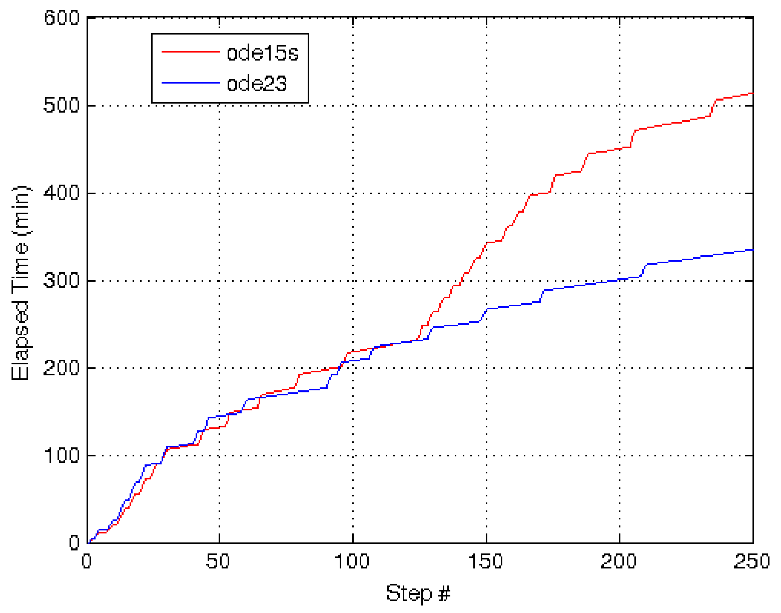

Figure 7.—Shooting method MATLAB integrator performance.

4.2.1 IPOPT Algorithm

The IPOPT (Interior Point OPTimizer) is a general-purpose optimization tool that is not specific to shooting methodology. The shooting methodology is one type of problem that can be solved using IPOPT. The IPOPT algorithm is also being used in the investigation of using Direct Collocation as a means to solve the predictive kinematics problem. Therefore, efforts have been made to characterize IPOPT.

IPOPT does not evaluate the objective function, objective gradient, constraint, and constraint Jacobian function in a serial order. The algorithm determines the order based upon the convergence results at each step. To characterize this, the evaluation order of each function was tracked. It was determined, as can be seen in Figure 8, that objective and constraints functions are evaluated in pairs. For example, during one trial of the shooting method, IPOPT performed 292 steps, but there occurred only 120 unique parameter vectors. The objective gradient and constraint Jacobian are also evaluated in pairs. To further understand the effect of the differential equation solver on performance, the duration of time to investigate each step was also calculated as shown in Figure 9. As can be seen, the performance is roughly equivalent, with MATLAB's *ode23* being slightly faster.

Therefore, to increase the efficiency of the MATLAB code, the following was done:

(1) While the objective and constraint function need be separate entities, calculate both objective and constraint values in the function and save them to global variables.
(2) Do the same for the objective gradient and the constraint Jacobian.
(3) At each step, the current set of parameters is compared to the previous set and if they are the same parameters, the constraint values are available without the need to perform a simulation of the model.

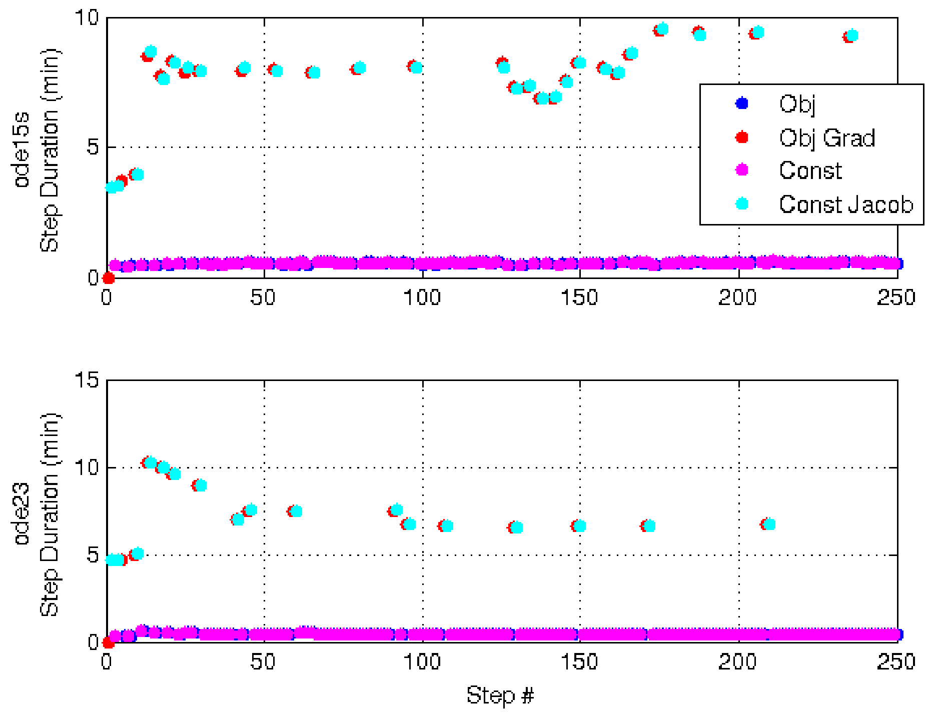

Figure 8.—IPOPT evaluation of objective, objective gradient, constraint, and constraint Jacobian order.

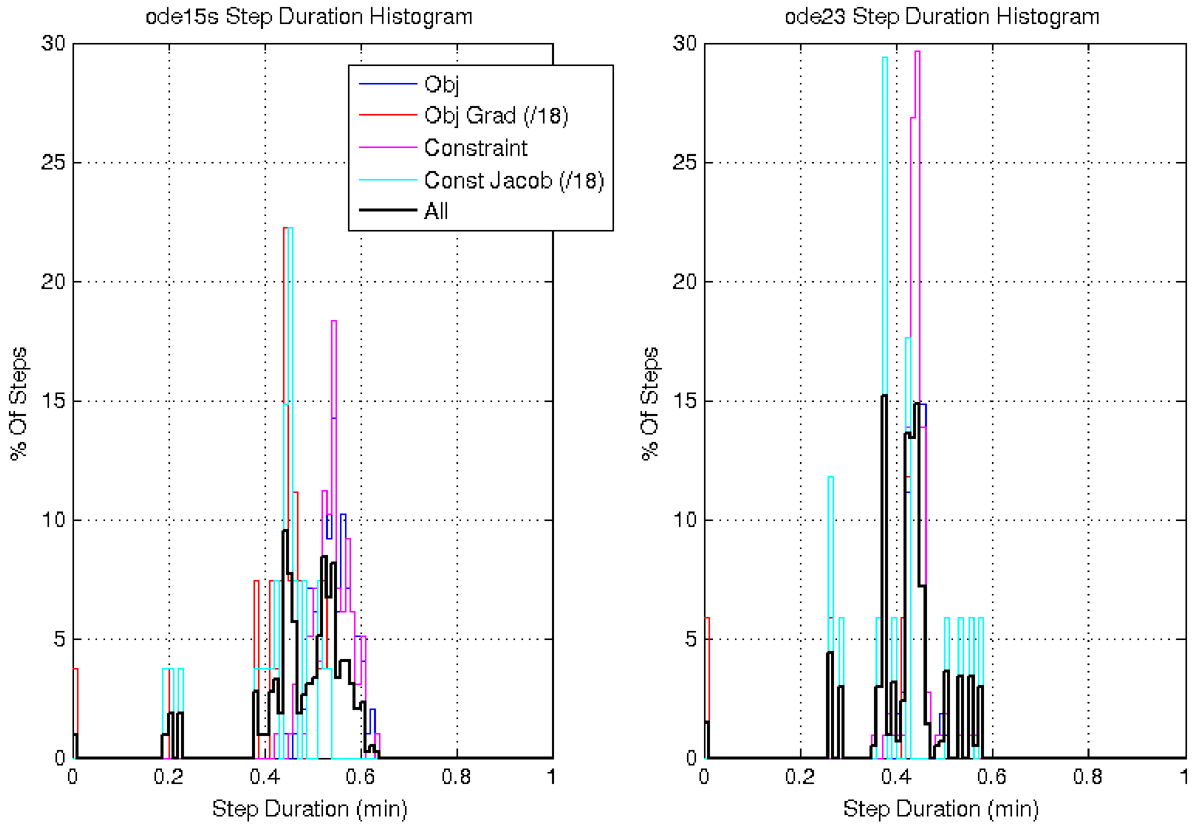

Figure 9.—Differential equation solver performance for 2 candidate MATLAB solvers: ode15s (left) and ode23 (right).

As discussed previously, because the OpenSim model cannot be solved as a closed-form equation, numerical gradients must be calculated. This requires the model to be evaluated at an “operating point” and then perturbed using a finite differences methodology. The choice of this finite differences perturbation value is important, as it affects the “scaling” of the problem. Consider that the objective function has units of m/s (velocity), constraints in m (displacement), and the input parameters are unitless (muscle activation). To account for the difference in units, and provide scaling of constraints relative to one another optimization algorithms “scale” the problems, by applying internal weighting factors. In the case of IPOPT, the model is evaluated at the initial estimate then the results are used to perform scaling of the parameters and functions for the remainder of the optimization run. The IPOPT documentation suggests to attempt to prescale a problem such that the values of the objective, constraints, gradient, and Jacobian are in the range of 0.01 to 100. This also provides some idea of the range in which IPOPT is robust; if the values have a domain much larger than three orders of magnitude across the optimization space, the problem can be considered to be poorly scaled. A detailed investigation of the effects of scaling was performed using a static optimization problem and scaling is discussed in detail in Appendix B.

The shooting method is able to approach a minimum; this minimum was not always found to be the global minimum but was often a local minimum. This is believed to be due to two factors:

(1) Scaling of the optimization problem

(2) Unstable approximation of the objective gradient and constraint Jacobian

To investigate the stability of the derivative calculation, the input parameters (muscle activation) were manually perturbed and the change in the objective function (hand velocity at the end of simulation) was calculated. For example as shown in Figure 10 and Figure 11, the simulation was run and the muscle activation of the triceps was manually increased. The allowable error tolerance of the simulation’s integration can also have an effect on the variance of the results and can also be seen in Figure 10.

The gradient of the objective is the slope of these lines, and can become noisy, as shown in Figure 8. While values of *h* and the integrator absolute tolerance can be found that provide a relatively smooth gradient, there is no guarantee that these values are robust across the entire optimization space.

Several tools exist to perform adaptive gradient calculation. One of these tools, Adaptive Robust Numerical Differentiation (Ref. 16) was applied. These tools require additional evaluations of the simulation to be performed so as to verify convergence on a stable derivative and therefore increase calculation time. As shooting optimization is computationally relatively slow, convergence was not demonstrated to occur within a reasonable period (<40 hr). It should be mentioned and appreciated that if the problem is specified as a set of analytical closed-form equations, the derivatives are exact and IPOPT performs well. However, for a numerically evaluated model in OpenSim, this is not possible. For this reason, it was decided to investigate a non-gradient-based optimization methodology.

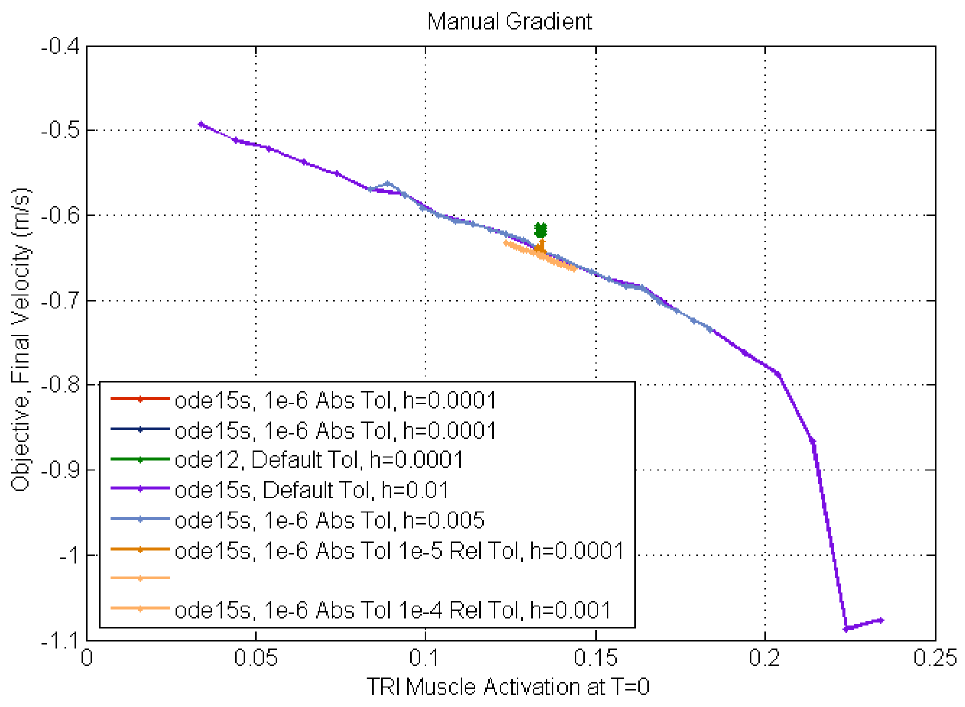

Figure 10.—Iterative calculation of objective gradient, varying ode solver, ode integration tolerance, and derivative finite difference step size (h).

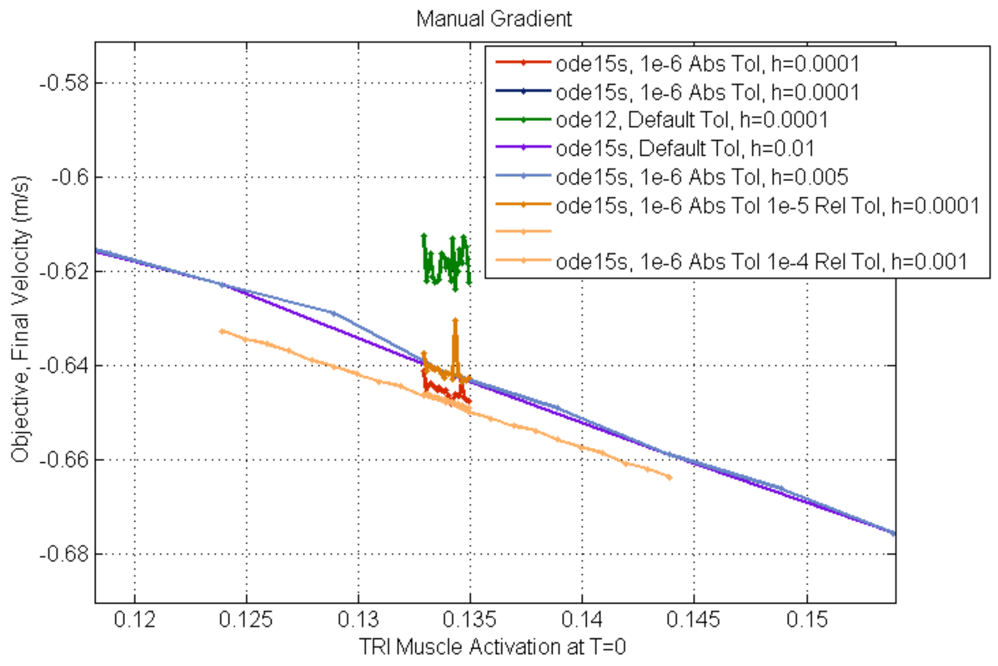

Figure 11.—Iterative calculation of objective gradient, varying ode solver, ode integration tolerance, and derivative finite difference step size (h). Zoomed in on small step size to show noise.

4.2.2 FMINCON

MATLAB's optimization toolbox includes several optimizers. MATLAB's *fmincon* function performs constrained optimization for problems posed in a manner similar to IPOPT. *fmincon* does not require gradient and Jacoboian functions to be provided; it performs internal gradient calculation. *fmincon* was able to converge on a solution in about 22 hr, but this solution was found to be a local minimum. Results are shown in Figure 12.

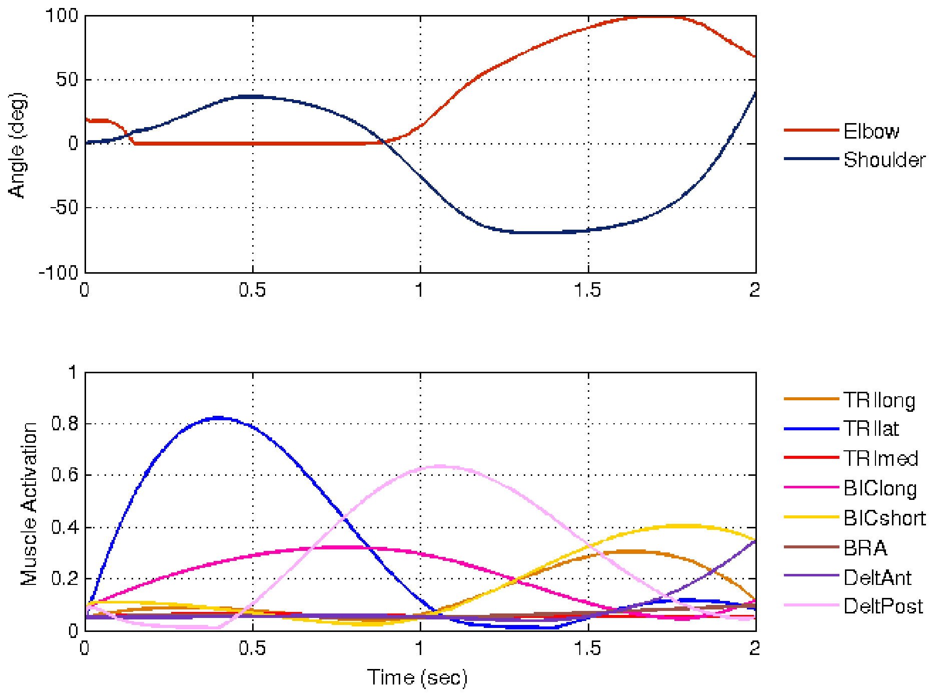

Figure 12.—*fmincon* Arm28 results for joint angles (top) and muscle activation patterns (bottom).

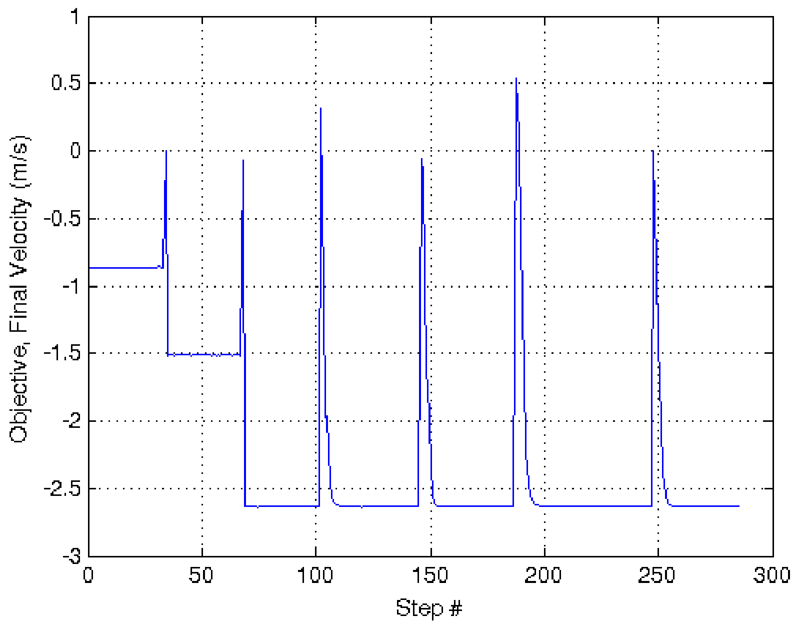

Figure 13.—*fmincon* convergence on final velocity of hand.

4.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) does not directly rely on gradient approximation. A population of particles is randomly placed in the optimization space; each particle is specified by a set of muscle activation parameters. The objective and constraints are then calculated, and the results are communicated between the particles. Each particle then uses the values reported by the other members of the population swarm to move to a new location. Over numerous iterations, the particles will converge upon one another at the solution. PSO algorithms are computationally intensive in that objective and constraint functions need to be evaluated a large number of times thus making it very expensive for our numerical OpenSim functions. Since PSO does not rely on gradient and Jacobian evaluation, it was decided to attempt the use of PSO.

The Arm28 model was investigated with a PSO algorithm and the results were mixed. Using a small population (20 particles), the algorithm converges after 27 generations i.e., with each particle solved 27 iterations or steps in 24 hr of run time (see Figure 15). However, the algorithm converged on a local minimum as seen in Figure 14. Note that the anterior and posterior deltoids are simultaneously active and are therefore antagonistically opposing one another; this is a suboptimal condition.

One method to decrease the likelihood of converging into a local minimum is to use a sufficiently large population size. As a general rule, the population size should be approximately 95[1] for a parameter size of 32. After 36 hr of run time with a population of 95, convergence had not been achieved, and therefore the PSO algorithm is currently not planned for further investigation.

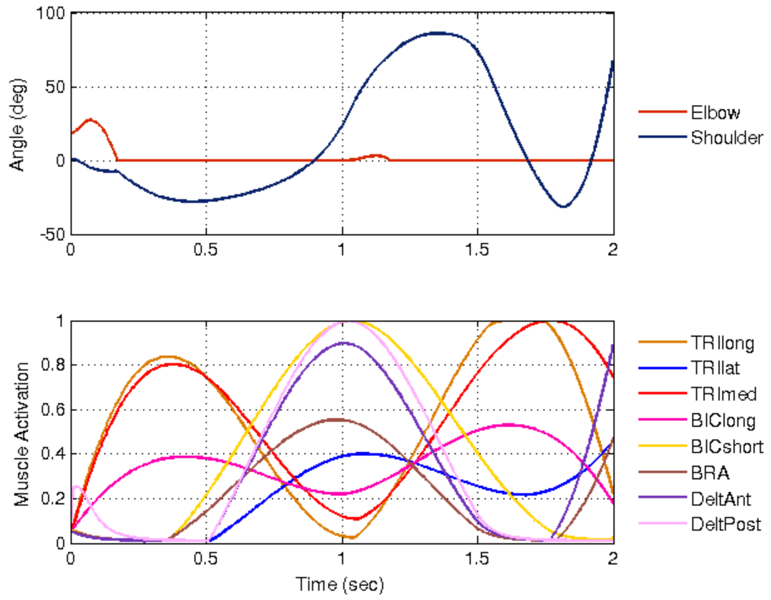

Figure 14.—PSO Arm28 results for joint angles (top) and muscle activation patterns (bottom).

[1] Table 11.1 from D. Simon, "Evolutionary Optimization Algorithms"," 1st Edition, John Wiley & Sons Inc., 2013.

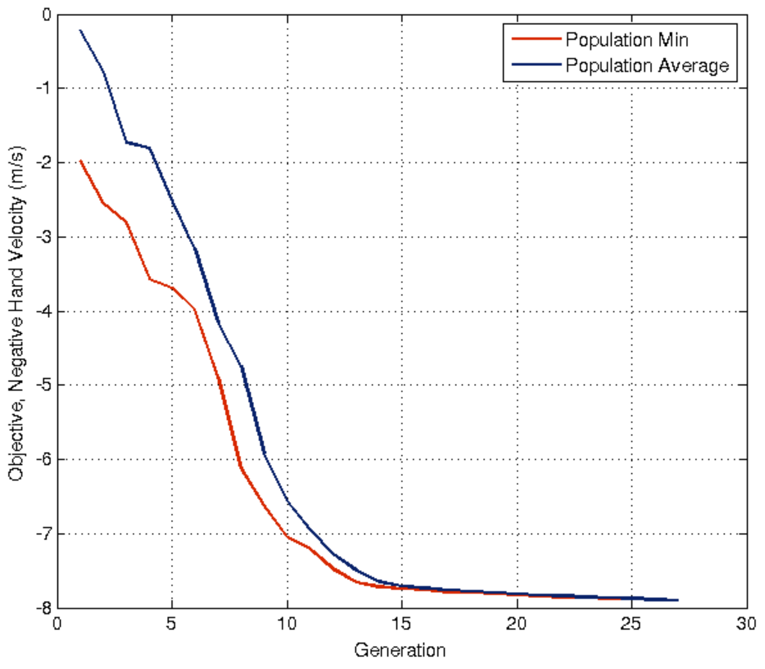

Figure 15.—PSO convergence on final velocity of hand.

4.4 Direct Collocation

Direct collocation is currently planned to be the methodology utilized to perform predictive kinematics in OpenSim. This method does not require the full numerical integration/simulation across the entire time domain. Instead, the problem is divided into N collocation nodes; typically between 20 and 100 during gait analysis. The parameters that are iterated by the optimizer are the states of the system (displacement, velocity, acceleration). The states must be consistent over time; the velocity and acceleration at one state must drive the system to the next state; this is imposed in the constraint function and are referred to as dynamic constraints. Additionally, constraints are added to reflect the desired task. For example, the velocity at the first node and the last node must be a prescribed value and are referred to as task constraints. At each of the N nodes the dynamic equations of the system are evaluated with the parameters (displacement and velocity states), dynamic constraints are evaluated (are the states consistent to achieve the adjacent states), and the task constraints are evaluated. The IPOPT optimizer then adjusts the parameters to better meet the constraints or minimize the objective. It should be noted that the number of collocation nodes is much smaller than the number of integration steps utilized in an ODE solver required by the shooting and particle swarm methodologies. The reduced number of evaluations of the functions is what makes direct collocation advantageous.

The works of Dr. Antoine van Den Bogert provide detailed information on the use of direct collocation with biomechanics models (Ref. 17). A simple pendulum problem often used by Dr. Antoine van den Bogert for illustrative purposes is shown in Figure 16. This model has been adapted to OpenSim, see Figure 17.

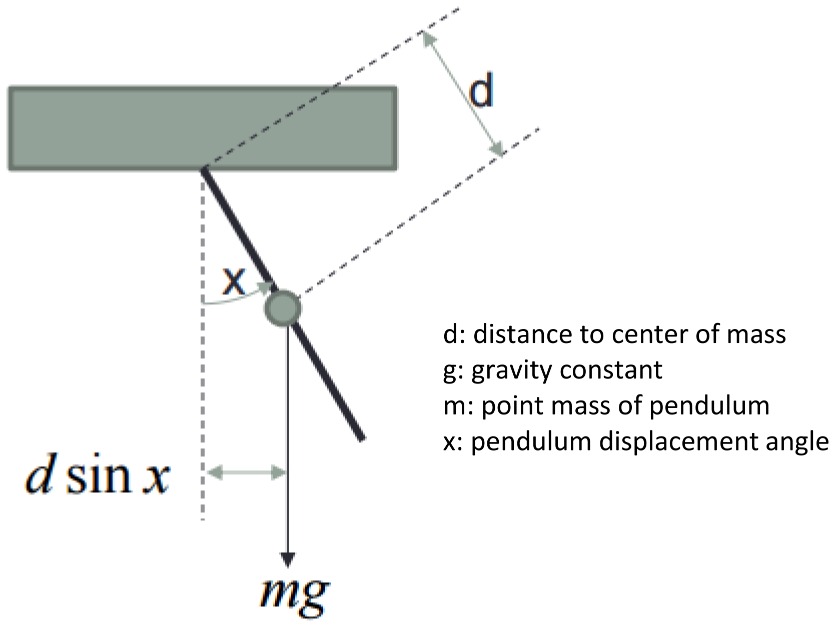
d
x
$d \sin x$
mg
d: distance to center of mass
g: gravity constant
m: point mass of pendulum
x: pendulum displacement angle

Figure 16.—van den Bogert's Pendulum Example.

The equation of motion for the simple pendulum shown in Figure 16 is:

$$I\ddot{x} = -mg\ (dsinx) + u$$

We can then formulate a problem with task constraints. The initial state is hanging at rest, and the final state at time *T* is the pendulum in an inverted position:

$$x(0) = 0 \quad \dot{x}(0) = 0$$
$$x(T) = \pi \quad \dot{x}(T) = 0$$

The cost function to be minimized is the integral of the squared torque, *u*, required to achieve the target position:

$$\int_0^T u(t)^2 dt$$

The problem is temporally discretized into *N* nodes, and the length of each time step is:

$$h = T/(N-1)$$

The system is characterized by position states of x_1, x_2, x_3,…. x_N and controls (torques) of u_1, u_2, u_3,…. u_N.

The equation of motion can then be conformed to algebraic constraints:

$$I\frac{x_{i+1} - 2x_i + i_{i-1}}{h^2} = -mgdsinx_i + u_i$$

and the cost function becomes the algebraic function:

$$\sum_i u_i^2$$

The parameters manipulated by the optimizer algorithm to meet the constraints and to minimize the cost function are:

$$y = (x_1, x_2 \ldots . x_N, u_1, u_2 \ldots . u_N)$$

This pendulum problem has been used with closed-form equations (derivatives of all of the equations above can be symbolically derived and implemented as gradient/Jacobian functions).

This same model has been implemented in OpenSim. Code has been generated to utilize OpenSim's MATLAB API, and the optimization is performed using direct collocation but with numeric evaluation of the functions. This allows validation of the OpenSim direct collocation model and performance base lining with Dr. van den Bogert's closed-form solution.

As can be seen in the two cases presented in Figure 18 and Figure 19, the closed-form analytical solution and the OpenSim numerical results closely match. Both methods fully converge on a solution; the analytical method completes in 0.61 sec and the OpenSim numeric method in 17.38 sec. These results are promising and are currently being applied to the Arm28 model (Ref. 1).

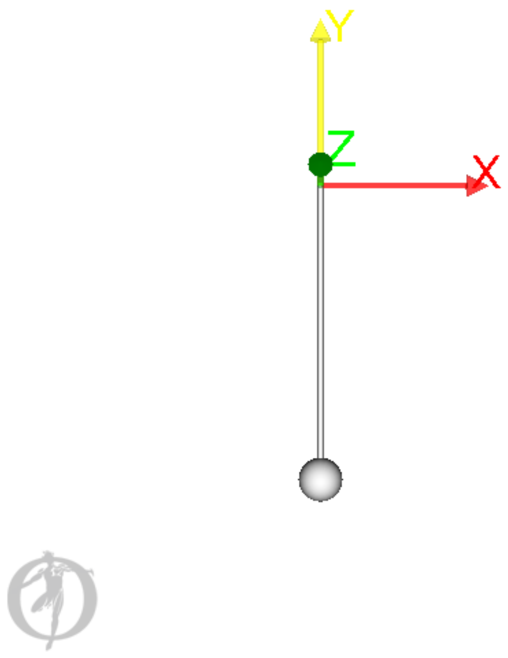

Figure 17.—Pendulum model of Figure 16 implemented in OpenSim.

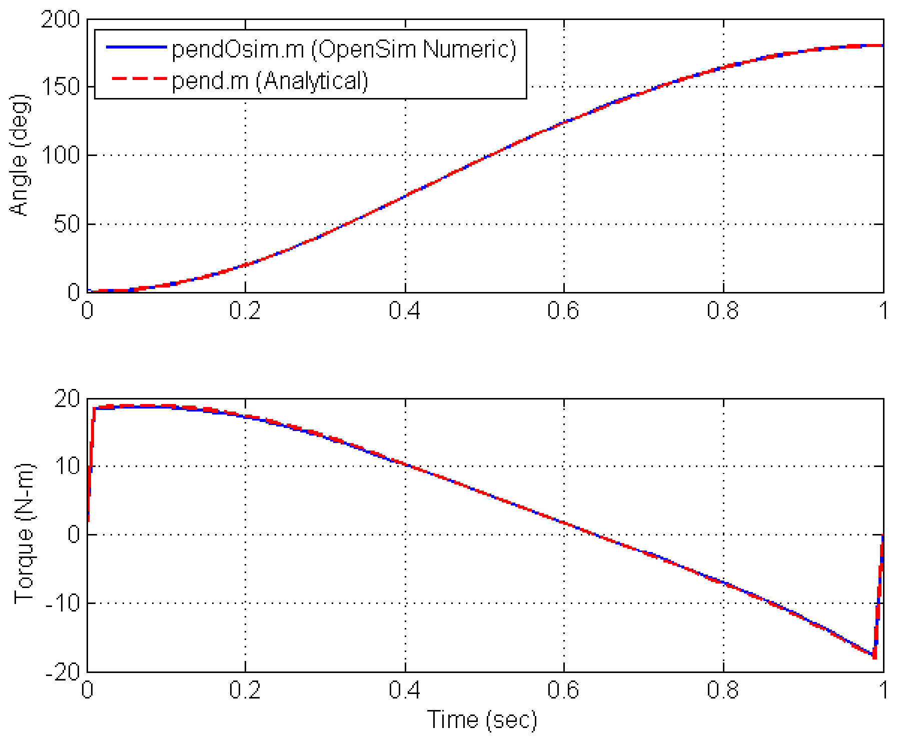

Figure 18.—Pendulum model case 1: Rotation to 180º in 1 sec. Results for joint angles (top) and torque actuation (bottom) for analytical and OpenSim generated solutions. (pendOsim.m and pend.m reference analysis files in Digital Astronaut Project software repository).

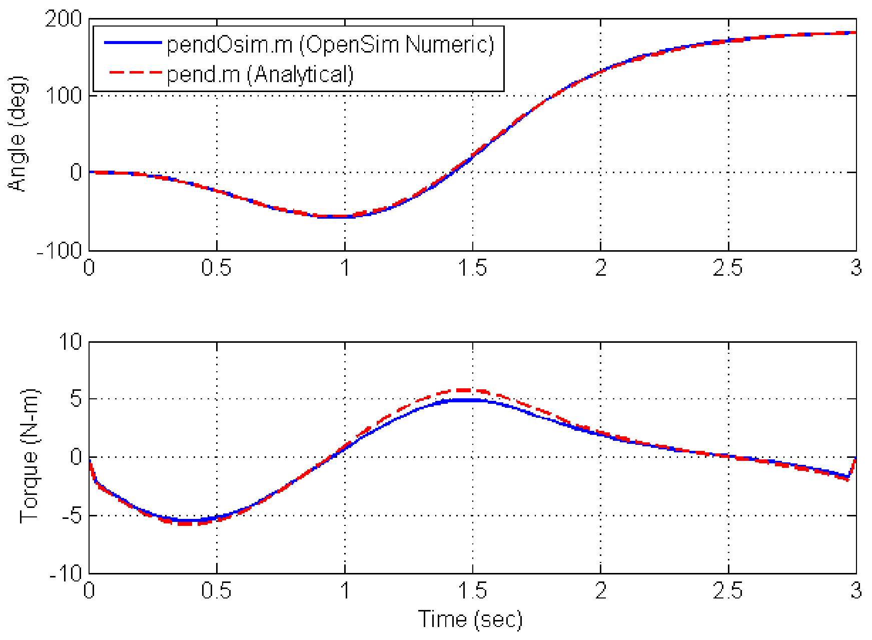

Figure 19.—Pendulum Model Case 2: Rotation to 180º in 3 sec. Results for joint angles (top) and torque actuation (bottom) for analytical and OpenSim generated solutions.

References

1. B. Prilutsky, V. Zatsiorsky, “Optmization-based models of Muscle Coordination,” Exercise Sports Science Review, January 2002.
2. Delp SL, Anderson FC, Arnold AS, Loan P, Habib A, John CT, Guendelman E, Thelen DG. OpenSim: Open-source Software to Create and Analyze Dynamic Simulations of Movement. IEEE Transactions on Biomedical Engineering. (2007).
3. M. Ackermann, A. van den Bogert, “Optimality principles for model-based prediction of human gait,” Journal of Biomechanics 2010 1055-1060.
4. J. Rasmussen, M. Damsgaard, M. Voight, “Muscle recruitment by the min/max criterion – a comparitive numerical study,” Journal of Biomechanics 2001 409-415.
5. M. Ackermann, A. van den Bogert, “Predictive simulation of gait at low gravity reveals skipping as the preferred locomotion strategy,” Journal of Biomechanics, 2013.
6. Alexander RM, “Optimization and gaits in the locomotion of vertebrates,” Physiological Reviews. 1989.
7. Bertram JEA, Ruina A, “Multiple walking speed-frequency relations are predicted by constrained optimization,” Journal of Theoretical Biology, 2001.
8. Crowninshield RD, Brand RA, “Physiologically based criterion of muscle force prediction in locomotion,” Journal of Biomechanics, 1981.
9. Raichlen DA, “The effects of gravity on human walking: a new test of the dynamic similarity hypothesis using a predictive model,” The Journal of Experimental Biology, 2008.
10. Thelen DG, Anderson FC, “Using computed muscle control to generate forward dynamic simulations of human walking from experimental data,” Journal of Biomechanics, 2006.
11. A. Wächter. An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, January 2002.
12. Computational Infrastructure for Operations Research, IPOPT On-line Docuemntation: https://projects.coin-or.org/Ipopt/wiki/HintsAndTricks.
13. Computational Infrastructure for Operations Research, Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT. Revision 2500, August 16, 2014.
14. L. Holmberg, A. Klarbring, “Muscle decomposition and recruitemnt criteria influence muscle force estimates,” Multibody System Dynamics September 2011.
15. J. Betts, “Practical Methos for Optimal Control and Estimation Using Non-linear Programming,” Second Edition, soceiety for Industrial and Applied Mathematics. Pg. 45.
16. J. D’Erricco “Adaptive Robust Numerical Differentiation” MATLAB File Exchanage User Submission, http://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation
17. A. van den Bogert, D. Blana, D. Heinrich, “Implicit methods for efficient musculoskeletal simulation and optimal control,” Procedia, January 2011.
18. K. Schittkowski “Test Examples for Nonlinear Programming Codes,” Lecture Notes in Economics and Mathematical Systems, Vol. 187.

Appendix A.—Example Analytical Optimization Problem

Prof. Klaus Schittkowski of the Department of Computer Science at the University of Bayreuth collected a famous set of optimization problems for use in the research and development of optimization techniques, commonly referred to as the H&S test problems (Ref. 18). One example problem is demonstrated here.

H&S Problem #51 is shown below. There is a differential objective function and three differential constraint functions. Not shown in this example: the parameters can be limited to a range and the constraints can be implemented as an equality or as inequalities. For an analytical function evaluation, the gradient and Jacobian functions could be initially derived using MATLAB's symbolic toolbox. For OpenSim model optimization, each of the functions perform calls to the OpenSim MATLAB API.

PROBLEM:	51
CLASSIFICATION:	QLR-T1-6
SOURCE:	Huang, Aggerwal [34]
NUMBER OF VARIABLES:	$n = 5$
NUMBER OF CONSTRAINTS:	$m_1 = 0$, $m-m_1 = 3(3)$, $b = 0$

OBJECTIVE FUNCTION:

$$f(x) = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2$$

CONSTRAINTS:

$$x_1 + 3x_2 - 4 = 0$$

$$x_3 + x_4 - 2x_5 = 0$$

$$x_2 - x_5 = 0$$

START: $x_0 = (2.5 , .5 , 2 , -1 , .5)$ (feasible)

$f(x_0) = 8.5$

SOLUTION:

$x^* = (1 , 1 , 1 , 1 , 1)$

$f(x^*) = 0$

$r(x^*) = 0$

$e(x^*) = 0$

$\mu = 0$

$I(x^*) = -$

$u^*_{max}/u^*_{min} = 0/0$

$\lambda^*_{max}/\lambda^*_{min} = 3.49/1.90 = 1.84$

H&S Problem #51 is included as an example with IPOPT's MATLAB distribution.

```matlab
% Test the "ipopt" Matlab interface on the Hock & Schittkowski test problem
% #51. See: Willi Hock and Klaus Schittkowski. (1981) Test Examples for
% Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical
% Systems Vol. 187, Springer-Verlag.
%
% Copyright (C) 2008 Peter Carbonetto. All Rights Reserved.
% This code is published under the Eclipse Public License.
%
% Author: Peter Carbonetto
%         Dept. of Computer Science
%         University of British Columbia
%         September 18, 2008
function [x, info] = exampleshs051

  x0         = [ 2.5 0.5 2 -1 0.5 ];  % The starting point.
  options.cl = [ 4 0 0 ];             % Lower bounds on constraints.
  options.cu = [ 4 0 0 ];             % Upper bounds on constraints.

  % Set the IPOPT options.
  options.ipopt.jac_c_constant        = 'yes';
  options.ipopt.hessian_approximation = 'limited-memory';
  options.ipopt.mu_strategy           = 'adaptive';
  options.ipopt.tol                   = 1e-7;

  % The callback functions.
  funcs.objective         = @objective;
  funcs.constraints       = @constraints;
  funcs.gradient          = @gradient;
  funcs.jacobian          = @jacobian;
  funcs.jacobianstructure = @jacobian;

  % Run IPOPT.
  [x info] = ipopt(x0,funcs,options);

% ----------------------------------------------------------------------
function f = objective (x)
  f = (x(1) - x(2))^2 + ...
      (x(2) + x(3) - 2)^2 + ...
      (x(4) - 1)^2 + (x(5) - 1)^2;

% ----------------------------------------------------------------------
function g = gradient (x)
  g = 2*[ x(1) - x(2);
          x(2) + x(3) - 2 - x(1) + x(2);
          x(2) + x(3) - 2;
          x(4) - 1;
          x(5) - 1 ];

% ----------------------------------------------------------------------
function c = constraints (x)
```

```
  c = [ x(1) + 3*x(2);
        x(3) + x(4) - 2*x(5);
        x(2) - x(5) ];

% ----------------------------------------------------------------------
function J = jacobian (x)
  J = sparse([ 1  3  0  0  0;
               0  0  1  1 -2;
               0  1  0  0 -1 ]);
```

```
The Results for running the above code:


This is Ipopt version 3.11.0, running with linear solver ma57.

Number of nonzeros in equality constraint Jacobian...:        7
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        0

Total number of variables............................:        5
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        3
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0 8.5000000e+000 0.00e+000 4.38e+000   0.0 0.00e+000    -  0.00e+000 0.00e+000   0
   1 1.4119822e+000 0.00e+000 2.46e+000 -11.0 4.38e+000    -  1.00e+000 5.00e-001f  2
   2 1.7008769e-004 0.00e+000 2.19e-002 -11.0 6.89e-001    -  1.00e+000 1.00e+000f  1
   3 1.3722572e-006 2.22e-016 1.74e-003 -11.0 8.44e-003    -  1.00e+000 1.00e+000f  1
   4 1.6589961e-010 0.00e+000 2.34e-005 -11.0 9.18e-004    -  1.00e+000 1.00e+000f  1
   5 1.0880086e-013 0.00e+000 6.83e-007 -11.0 7.79e-006    -  1.00e+000 1.00e+000f  1
   6 5.8410676e-020 8.88e-016 4.06e-010 -11.0 1.92e-007    -  1.00e+000 1.00e+000f  1

Number of Iterations....: 6

                                   (scaled)                 (unscaled)
Objective...............:  5.8410675585715588e-020   5.8410675585715588e-020
Dual infeasibility......:  4.0620632498681392e-010   4.0620632498681392e-010
Constraint violation....:  8.8817841970012523e-016   8.8817841970012523e-016
Complementarity.........:  0.0000000000000000e+000   0.0000000000000000e+000
Overall NLP error.......:  4.0620632498681392e-010   4.0620632498681392e-010


Number of objective function evaluations             = 12
Number of objective gradient evaluations             = 7
Number of equality constraint evaluations            = 12
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 1
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 0
Total CPU secs in IPOPT (w/o function evaluations)   =      0.026
Total CPU secs in NLP function evaluations           =      0.008

EXIT: Optimal Solution Found.
ans =
  Columns 1 through 3
         0.999999999938707          1.00000000002043          1.00000000016744
  Columns 4 through 5
         0.999999999873427          1.00000000002043
```

Appendix B.—Static Optimization

The following investigation was developed as part of a class project and is presented here to demonstrate the effects of scaling on an optimization problem.

B.1 Introduction

The polynomial and minmax criteria have been investigated using the static optimization of a simple lower leg model with 3 degrees of freedom and 8 muscles. Numerical aspects of using these criteria, and specifically the effects of scaling, is demonstrated. The effect of muscle discretization on static optimization is also investigated. The static optimization algorithm in OpenSim is reviewed with respect to these considerations.

As discussed in Reference 1, the human body has 244 kinematic degrees of freedom (DoF) and approximately 630 muscles; the human body is redundantly actuated. Due to this redundancy, multiple muscles can provide antagonistic actuation to the same joint. While the required total moment needed to actuate a joint can be calculated (for example, using inverse dynamics), the distribution of this moment over more than one muscle is statically indeterminate; this is referred to as the force distribution problem.

A simple and often cited example of this is:

$$M = F_1 d_1 + F_2 d_2 \tag{1}$$

where *M* is the moment at the anatomical joint, d_1 is the moment arm of muscle 1, and d_2 is the moment arm of muscle 2. *M* is calculated using inverse dynamics with no information required about the muscle force. The muscle moment arms are calculated during inverse dynamics analysis with only muscle geometric information (no muscle force information is required). We are then left with one equation and two unknowns (the muscle forces). The purpose of static optimization is to determine the muscle forces based on a predefined optimization criterion.

As discussed in Reference 1, it is hypothesized that the central nervous system uses a set of control principles to select muscle activation patterns based on some optimization criteria. The polynomial criterion is one of the most utilized criteria and is the criterion utilized by OpenSim (Ref. 2).

The polynomial criterion is:

$$G_{poly} = \sum_{i=1}^{\mathrm{m}} \left(\frac{F_i}{PCSA_i}\right)^p \tag{2}$$

where *m* is the number of muscles, F_i is the force of a muscle, $\mathrm{PCSA_i}$ is the muscle's physiological cross-sectional area, and *p* is the polynomial exponent. During static optimization, an exponent is selected and the optimization routine minimizes the value of (2) while satisfying the constraint that forces must generate the desired moments. The exponent *p* is typically 2 or 3, and the objective can then be stated as the minimization of the sum of squared/cubed muscle stress. For these relatively lower powers of *p*, the criterion is generally referred to as an "effort"-based criterion. At higher values of *p*, the polynomial criterion is considered to be "fatigue"-like (Ref. 1). This is due to the high exponent heavily weighting the muscle with the greatest cross sectional stress; there is consequently minimal cost associated with increasing less stressed muscles. The resulting muscle actuation pattern will tend to load level stress across the muscles capable of contributing. It is much less likely that any one muscle will become highly loaded. It has been demonstrated that this criterion, along with the incorporation of muscle volume scaling, accurately predicts gait (Ref. 3).

The minmax criterion for static optimization is:

$$G_{minmax} = max\left(\frac{F_i}{PCSA_i}\right) \tag{3}$$

The optimizer attempts to minimize the maximum stress in the muscle set. Similar to the polynomial at high values of *p*, the minmax criterion is a fatigue-like criterion. As will be demonstrated, the polynomial criterion approaches the minmax criterion as *p* approaches infinity (Ref. 3).

Details for each of the objective functions and comparison of their results to measured data can be found in (Ref. 1). While the main goal in the use of these objective functions is to mimic the muscle activation patterns utilized in the central nervous system, the numerical stability and consistency of these functions must also be considered. A simplified model has been utilized to demonstrate the following:

- Polynomial objective functions and scaling
- Comparison of minmax to polynomial
- Muscle discretization
- Review of OpenSim's implementation of static optimization

B.2 Methods

To support the above goals, static optimization has been performed on a simple leg model with 3 joints and 8 muscles. A torque of 50 N-m in the directions shown in Figure 20 is specified at each of the joints: ankle, knee, and hip. The muscle complexes include: Iliposoas (Ilipos), Gluteus (Glut), Hamstring (Hams), Rectus (Rect), Vastus (Vast), Gastrocnemius (Gast), Soleus (Soleus), and Tibialis Anterior Muscle. Muscle moment arms were roughly approximated through measurement in OpenSim. The maximum allowable force of each muscle is shown in Equation (5). PCSA is determined by dividing each muscle's maximum isometric force by the maximum allowable muscle fiber stress (σ_{max}), which is assumed to be 25×10^4 N/m^2.

$$D = \begin{array}{r} (meters) \\ Hip \\ Knee \\ Ankle \end{array} \begin{array}{c} \begin{array}{cccccccc} Ilipos & Glut & Hams & Rect & Vast & Gast & Soleus & Tib \end{array} \\ \begin{bmatrix} 0.050 & -0.062 & -0.070 & 0.034 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.034 & 0.050 & 0.042 & -0.020 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.053 & -0.0530 & 0.037 \end{bmatrix} \end{array} \tag{4}$$

$$\begin{array}{rcccccccc} (N) & Ilipos & Glut & Hams & Rect & Vast & Gast & Soleus & Tib \\ F_{max} = & [1917 & 1967 & 3878 & 1718 & 8531 & 2596 & 3734 & 1233] \end{array} \tag{5}$$

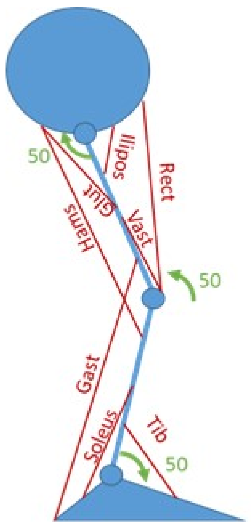

Figure 20.—Lower Body Model. 8 muscle with 3 degrees of freedom. Moments at each joint arbitrarily chosen to be 50N-m.

All optimizations have been performed in MATLAB using the IPOPT optimizer (Ref. 11). Analysis code is provided in the Appendix. Specifics of the software used:

- MATLAB Version: 8.0.0.783 (R2012b)
- IPOPT Version 3.11 with MA57 Linear Solver
- Operating System: Linux 64-bit Ubuntu 14.04 (3.13.0-39-generic)

IPOPT is utilized with all of its default parameters and only the following explicitly set options:

- options.ipopt.hessian_approximation = 'limited-memory';
- options.ipopt.to = 1×10^{-7};

B.3 Results

B.3.1 Polynomial Objective Functions and Scaling

Typically, the polynomial criterion is utilized with the exponents p =2 or p =3 (Ref. 1). If (2) is directly utilized, it suffers from numerical scaling problems at high values of p. For example, if we consider the vastus medialis *PCSA* of 3.4×10^{-3} m^2, at a maximum muscle force of 8531N, the objective function term of the vastus will be:

$$G_{poly,vast} = (\frac{F_{max,vast}}{PCSA_{vast}})^p = (\frac{8531}{0.0034})^p = (25e4)^p \quad (6)$$

Noting that because the F_{max} value of the vastus was utilized for the muscle force in this example, the term becomes equivalent to σ_{max}. From (6), it can be seen that the numerical results become highly sensitive to increasing powers of p. As it is expected that some muscles will provide near-zero force and others will provide force near their maximum, the numerical challenges at higher values of p should be appreciated.

The polynomial form in (2) is typically modified by noting that:

$$PCSA_i = \frac{F_{max,i}}{\sigma_{max}} \quad (7)$$

$$G_{poly} = \sum_{i=1}^{8}(\frac{F_i}{F_{i,max}}\sigma_{max})^p \quad (8)$$

For an optimization problem, the maximum muscle stress can then be factored out, as it is a general parameter for muscle fibers and assumed not to be specific to a particular muscle. The objective function then becomes:

$$G_{poly} = \sum_{i=1}^{8} (\frac{F_i}{F_{i,max}})^p \tag{9}$$

This objective function was implemented as follows in IPOPT:

Objective function:

$$obj = \sum_{i=1}^{8} \left(\frac{F_i}{F_{i,max}}\right)^p \tag{10}$$

Gradient of the objective:

$$grad = \frac{p\left(\frac{F_i}{F_{i,max}}\right)^p}{F_i} \tag{11}$$

Constraint function:

$$cnstr = D * F \tag{12}$$

Jacobian of the constraint:

$$jac = D \tag{13}$$

IPOPT is configured so that the constraint must equal the moment vector M (an equality constraint) and the values of F_i are limited to the bounds of 0 to $F_{max,i}$ (specific to each muscle).

Polynomial objective functions can become numerically problematic if not scaled correctly. This can be seen in Figure 21, where there is no obvious trend in muscle force and the exponent. IPOPT reports a failure to converge (“Restoration Phase Failed”) at $p > 3$. For all exponents, the solution does come close to generating the desired moments.

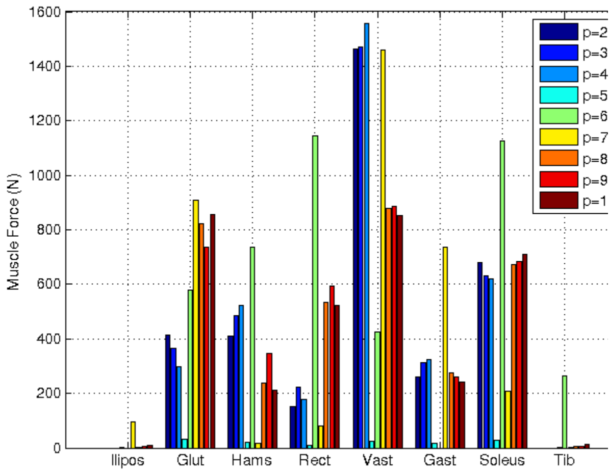

Figure 21.—Muscle force: polynomial criterion and unscaled objective function (Eq. (10)). Data shows no trending with increasing objective function polynomial exponent (p).

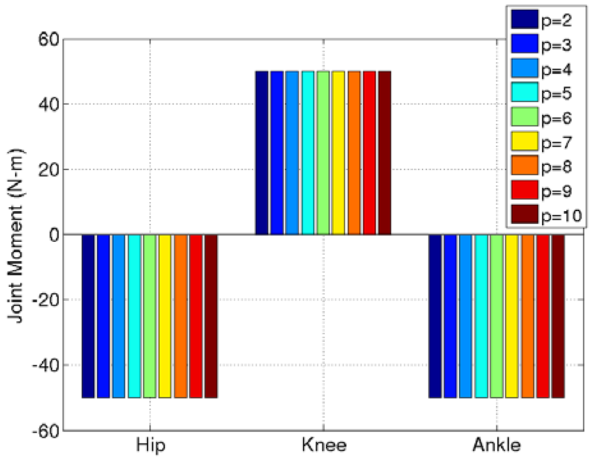

Figure 22.—Joint torques: polynomial criterion and unscaled objective function (Eq. (10)). Note that while muscle force has no trend, the desired joint moments are achieved.

The lack of a discernible trend and IPOPT's warning were determined to be caused by poor scaling in the objective function. It should be appreciated that at very low activation, for example 0.01, and a high power such as p = 10, the term for the muscle would have a value of 1×10^{-20}. In reviewing the IPOPT documentation, the following observations and recommendations were reported:

(1) "You should try to make the "typical" values of the non-zero first partial derivatives of the objective and constraint functions to be on the order of, say, 0.01 to 100" (Ref. 12).
(2) "By default, IPOPT performs some very simple scaling of the problem functions, by looking at the gradients of each function evaluated at the user-provided starting point. If any entry for a given function is larger than 100, then this function is scaled down to make the largest entry in the gradient 100" (Ref. 12).
(3) The default scaling method is "gradient-based". If the maximum gradient is above 100 (this value is adjustable using the *nlp_scaling_max_gradient parameter***)**, then gradient-based scaling will be performed. Scaling parameters are calculated to scale the maximum gradient back to this value. The lower bound for the scaling factors computed by the gradient-based scaling method is 1×10^{-8} (able to be specified by using the *nlp_scaling_min_value parameter***)** (Ref. 13).

The IPOPT scaling parameters were not adjusted (but are planned to be investigated further in future work). Instead, a manual scaling factor of 10 was added to the objective and gradient functions:

$$G_{poly} = \sum_{i=1}^{8} \left(10\,\frac{F_i}{F_{i,max}}\right)^p \tag{14}$$

The addition of the 10× coefficient produced good results, as seen in Figure 23 and Figure 24.

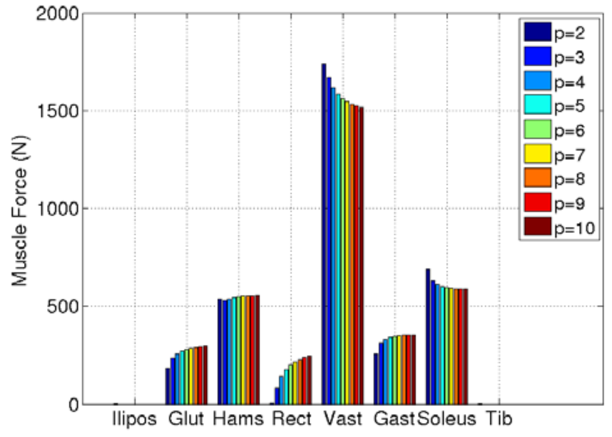

Figure 23.—Muscle force: polynomial criterion and unscaled objective function (Eq. (14)). Note trending of results with increasing objective function polynomial exponent (*p*).

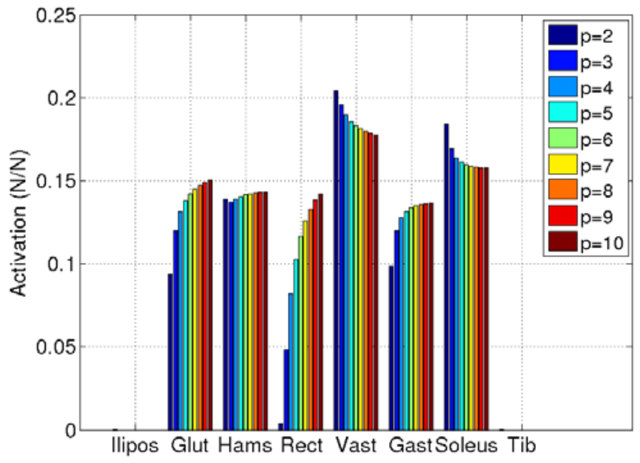

Figure 24.—Muscle activation: polynomial criterion and unscaled objective function (Eq. (14)).

A max-norm formulation (not to be confused with a minmax optimization) was also investigated. This adds an outer 1/*p* root to scale the objective:

$$G_p = \left(\sum_{i=1}^{8}\left(\frac{F_i}{F_{i,max}}\right)^p\right)^{1/p} \tag{15}$$

$$\frac{dG_p}{dF_i} = \frac{\left(\frac{F_i}{F_{i,max}}\right)^p}{F_i\left(\sum\left(\frac{F_i}{F_{i,max}}\right)^p\right)^{\frac{p-1}{p}}} \tag{16}$$

The max-norm formulation converged without the need for a scaling coefficient (Figure 25). In addition to the typical $p = 2$ and $p = 3$ exponents, 5 log-spaced exponents from 5 to 50 were also analyzed. It should be mentioned that for all of the criteria that converged in this study, global convergence was verified by additionally running static optimization with 100 trials of random initial guesses of the muscle force. This indicates strong robustness (at least for the model utilized in this study). Equation (16) is not unitless, however, and the number of muscle terms can affect the scaling; dividing by the number of muscles in the summation term and adding an outer multiplier of the mean F_{max} may provide additional robustness but has not been attempted here.

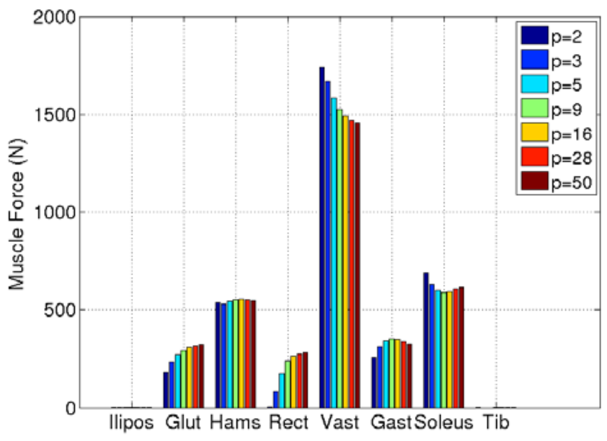

Figure 25.—Muscle force: polynomial criterion and max-norm objective function (15).

B.3.2 Comparison of Minmax to Polynomial

The minmax criterion, minimizing the maximum activation of all muscles, was implemented using the "bound formulation". The objective function is a single parameter: *s*. This parameter is added to the vector of muscle values to be solved for by the optimizer (referred to as a slack variable). In addition to the moment constraints, another set of constraints linking *s* and each of the muscle forces is added. The moment constraints are equity constraints (to the value of the moments), and the additional slack variable constraints are inequity constraints that activation minus *s* must be less than 0.

Objective function:

$$obj = s \tag{17}$$

Gradient of the objective function:

$$grad = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \tag{18}$$

Constraint function:

$$cnstr = [D * F_i, \frac{F_i}{F_{i,max}} - s] \tag{19}$$

Jacobian of the constraint function:

$$jac = [D\ 0\ \frac{1}{F_{i,max}}\ \text{-1}] \tag{20}$$

Bounds on parameters:

$$0 < F_i < F_{i\_max}\ (F_i \text{ must be between 0 and } F_{i\_max}) \tag{21}$$

$$0 < s < \infty \tag{22}$$

Constraints:

$$D * F_i = M_i \tag{23}$$

$$-\infty < \frac{F_i}{F_{i,max}} - s < 0 \tag{24}$$

The static optimization results are shown in Figure 26 and Figure 27. To demonstrate that the polynomial criterion approaches the minmax criterion, in addition to the typical $p = 2$ and $p = 3$ exponents, 5 log-spaced exponents from 5 to 50 were again analyzed (Figure 27). The relationship between p and activation is not monotonic for soleus. This is indicative, at least for this model, that relatively high polynomial orders are needed if the intent is to use the polynomial criterion to perform a minmax fatigue type of optimization. As previously discussed, it can be seen that the activation in the gluteus, vastus, rectus femoris, and soleus become load leveled.

Due to the formulation of the optimization problem, the minmax criterion does not suffer from scaling problems.

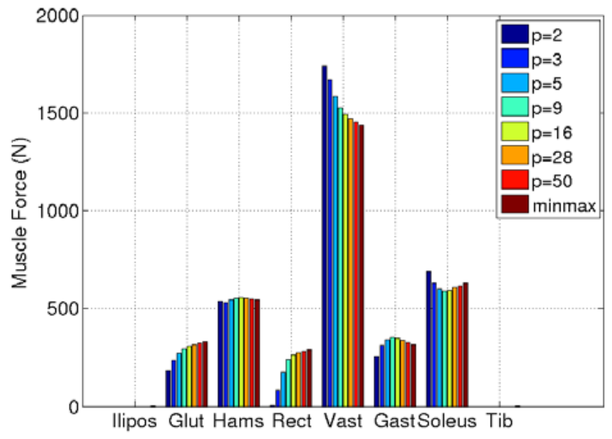

Figure 26.—Muscle force: minmax criterion.

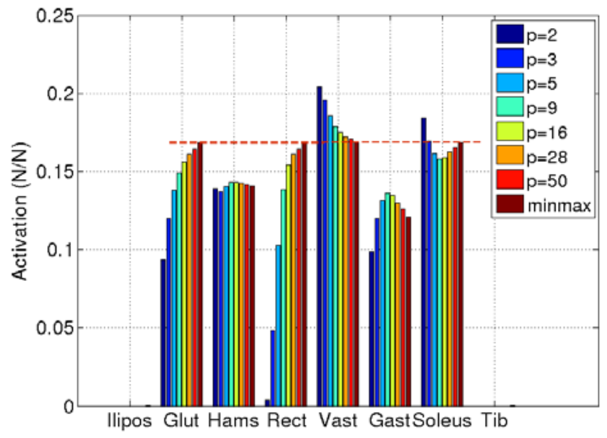

Figure 27.—Muscle activation: minmax criterion. Note activation across muscles reaches limit at high values of objective function polynomial (*p*).

B.3.3 Muscle Discretization

It may not be intuitively obvious that the polynomial objective functions may suffer from a mathematical inconsistency: muscle discretization. If a muscle is divided into two parallel muscles (discretized) having the same moment arm, it would seem that it would be appropriate to scale the muscles by dividing their maximal isometric forces in half. The effect of this incorrect assumption can easily be seen if we replace a single vastus muscle having a maximal isometric force of $F_{max,vast,init}$ with two muscles having maximal forces of $F_{max,vast,init}$ /2. As shown in the static optimization results of

Table 1, the sum of the forces of the 2 half vasti muscles does not equal the force of the single vastus muscle. Accordingly, the results for the other muscles are also affected. Holmberg (Ref. 14) derived a factor for scaling the muscle when using the polynomial criterion:

$$C_{poly,scale} = 2^{\frac{1-p}{p}} \tag{25}$$

TABLE 1.—MUSCLE DISCRETIZATION EFFECTS (FORCE IN *N*)

	Polynomial (p = 2)			Minmax	
	Baseline 1 Vast Muscle	2 Vast Muscle	2 Vast Muscle	*Baseline* 1 Vast Muscle	2 Vast Muscle
C_{scale}	1	0.5	0.707	1	0.5
Vast $F_{\max}$	8,531	4,266	6,031	8,531	4,266
Ilipos	0	0	0	0	0
Glut	184	236	184	331	331
Hams	539	519	539	546	546
Rect	7	59	7	289	289
Vast 1	**1,740**	**820**	**870**	**1,437**	**719**
Vast 2		**820**	**870**	**------**	**719**
Gast	256	210	256	314	314
Soleus	688	733	688	629	629
Tib	0	0	0	0	0

While this example addresses the simple case of dividing a muscle with the same moment arm (same insertion points), it should be appreciated that an additional level of complexity will be present if the muscle insertion points are also changed.

As can be seen in Table 1, the minmax criterion does not require the use of the scaling factor; dividing the muscle maximal isometric force in half produces vasti muscle forces that are half of the single vastus baseline, while the force values of the other 7 muscles remain unchanged.

B.3.4 Review of OpenSim Static Optimization Algorithm

OpenSim is capable of utilizing several optimization engines, and is packaged with IPOPT. The following source code from OpenSim v3.2 was reviewed:

- Optimizer.cpp
- InteriorPointOptimizer.cpp
- StaticOptimization.cpp
- StaticOptimizationTarget.cpp

OpenSim static optimization uses the polynomial criterion:

$$G_{poly} = \sum_{i=1}^{\mathrm{m}} (A_i)^p \tag{26}$$

Where A is muscle activation.

The objective function is the minimization of the sum of the muscle (or actuator) activation, and the polynomial power, p, is selectable by the user. The user is also able to choose whether the muscle force-length-velocity effects should be included. Including this relationship ensures that the results of the static optimization are realizable when forward dynamics analysis is performed. In the previous discussion on the polynomial and minmax criteria, note that the muscle maximum achievable force, $F_{max,I}$, is the maximum isometric force of the muscle. The maximum achievable force, however, decreases when the muscle is not at optimum length or is characterized by high shortening velocities. This ability in OpenSim ensures that static optimization results will not overstate the capability of muscles. The ability to not use the force-length-velocity relationship allows for easier troubleshooting of a static optimization problem that will not converge. In this case, all of the muscles' F_{max} values can be increased (by the same proportion) and the problem solved to determine which muscle complexes are being over actuated.

As seen in Figure 28, the optimization fails to converge above $p = 9$. When a 10× coefficient was added to the activation, the optimization converged through $p = 50$ (Figure 29).

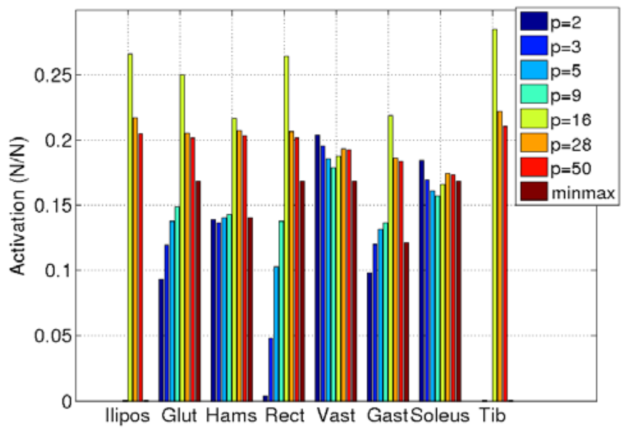

Figure 28.—Muscle activation: polynomial criterion and unscaled activation objective function (Eq. (26)).

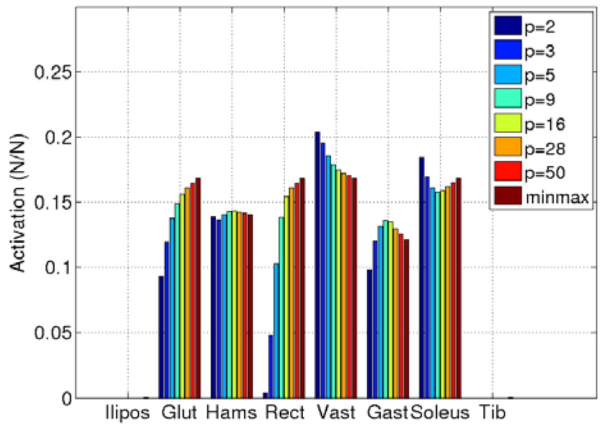

Figure 29.—Muscle activation: polynomial criterion and scaled activation objective function.

It has been my experience that OpenSim static optimization is prone to fail to converge when p is greater than 3. There is no prescaling term in OpenSim; activations are directly used. Given that IPOPT seems to not perform scaling for terms that are less than 10×10^{-8}, a slightly-actuated muscle (e.g., 0.01) will not be considered in scaling at approximately $p = 4$. Moments have been used in the constraint functions developed to support this paper (moments are readily available as an output of forward dynamics). OpenSim uses generalized coordinate acceleration in place of moments in static optimization constraints. From a mathematical standpoint, accelerations are more prone to scaling issues; relatively small mass bodies such as appendages can exhibit high accelerations at low forces. I believe that it would be productive to develop a static optimization analysis (or augment the current OpenSim static optimization analysis tool) to:

(1) Add a manual scaling factor to the objective and constraint (advanced user option)
(2) Investigate the use of generalized moments and forces in place of accelerations
(3) Add a minmax criterion to support fatigue-like optimization (due to both its numerical stability and for better modeling of fatigue-based activities). Over the course of this investigation, the minmax criteria never failed to converge.
(4) OpenSim starts with an initial guess of activation for all muscles of 0. As discussed above, gradient scaling is performed by IPOPT utilizing this guess. However, IPOPT will need to increase the activations to non-zero terms to establish the gradient and scaling. The process that IPOPT uses should be further investigated, along with the benefits of being able to specify the use of a default muscle activation other than 0.

B.4 Discussion

The implementation of the polynomial and minmax criteria has been investigated and demonstrated using a simple model. The polynomial criterion is useful for effort-like optimization when the exponent is small. It can be used for fatigue-like optimization with higher exponents, but becomes problematic due to scaling considerations. The minmax criterion is numerically much better suited to performing fatigue-like optimization and by its nature is better scaled.

The effect of muscle discretization on the polynomial and minmax criteria were also demonstrated. While the polynomial criterion can be corrected using Holmberg’s factor, the minmax criterion can be more intuitively used. As the polynomial and minmax criteria represent relatively different optimization paradigms, effort versus fatigue, both have a purpose. Consideration must be given to the type of static optimization that will be used when building or modifying a muscle-skeletal model’s properties.

Efforts here have emphasized investigating the scaling of objective functions. Scaling of constraint functions needs to be considered, as well.

In reviewing the OpenSim static optimization analysis, it may be possible to better scale problems and allow for improved convergence. Several possible modifications have been presented. Better scaling will not only provide a better chance to achieve convergence, but it can also increase the speed to reach convergence for problems that are currently successful.

I came across this quote at the completion of this project, while researching scaling methodologies, and I believe that it reflects the theme of my findings:

> *“Scaling is everything. Poor scaling can make a good algorithm bad. Scaling changes the convergence rate, termination tests, and numerical conditioning.”* - John T. Betts, Optimization Author and Guru (Ref. 15)