Documents

OpenSim Workshop

September 20-22, 2011

HYPER Summer School, La Alberca, Spain

Jeff Reinbolt, Jen Hicks

OpenSim Workshop Agenda

14:00 – 14:15	Welcome and goals of the workshop – *Jen Hicks*
14:15 – 14:35	Inverse dynamics and static optimization: how it works, exercises, & practice – *Jeff Reinbolt*
14:35 – 15:00	Guided analyses and exploration on your own – *Jeff Reinbolt & You*
15:00 – 15:20	Forward dynamics simulation: how it works, exercises, & practice – *Jeff Reinbolt*
15:20 – 15:45	Guided simulation and exploration on your own – *Jeff Reinbolt & You*
15:45 – 16:05	Interfacing OpenSim models with MATLAB/Simulink – *Jeff Reinbolt*
16:05 – 16:55	Guided control system design and exploration on your own – *Jeff Reinbolt & You*
16:55 – 17:00	Closing remarks – *Jen Hicks*

Acknowledgments

OpenSim was developed as a part of SimTK and funded by the Simbios National Center for Biomedical Computing through the National Institutes of Health and the NIH Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers can be found at http://nihroadmap.nih.gov/bioinformatics.

Trademarks and Copyright and Permission Notice

Workshop Description

Participants who attend this workshop will learn key features of OpenSim through real-life illustrations and exercises. Participants will create simulations on their computers and analyze those simulations to understand muscle actions.

Audience

This workshop is recommended for those who desire a broader knowledge of musculoskeletal simulation with OpenSim and how to begin using this software.

Learning Objectives

- Discuss the need for simulations to understand the complex neuromusculoskeletal system
- Perform inverse dynamics to determine joint torques from joint motion
- Perform forward dynamics to determine joint motion from muscle excitations
- Edit muscle excitations in a forward dynamics simulation to understand muscle actions
- Interfacing OpenSim models with MATLAB/Simulink to create closed-loop control systems

Table of Contents

1 Introduction

1.1 What is OpenSim?

OpenSim is a freely available software package that enables you to build, exchange, and analyze computer models of the musculoskeletal system and dynamic simulations of movement. OpenSim version 1.0 was introduced at the American Society of Biomechanics Conference in 2007. Since then, many people have begun to use the software in a wide variety of applications, including biomechanics research, medical device design, orthopedics and rehabilitation science, neuroscience research, ergonomic analysis and design, sports science, computer animation, robotics research, and biology and engineering education.

The software provides a platform on which the biomechanics community can build a library of simulations that can be exchanged, tested, analyzed, and improved through multi-institutional collaboration. The underlying software is written in C++ and the graphical user interface (GUI) is written in Java. OpenSim plug-in technology will make it possible to develop customized controllers, analyses, contact models, and muscle models among other things. These plug-ins can be shared without the need to alter or compile source code. You can analyze existing models and simulations and develop new models and simulations and visualize them within the GUI.

OpenSim is built using SimTK, an open-source simulation toolkit developed to create mathematical models of biological dynamics. SimTK is being developed by Simbios, an NIH National Center for Biomedical Computation based at Stanford University. Open-source, third-party tools are used for some basic functionality, including the Xerces Parser from the Apache Foundation for reading and writing XML files (xml.apache.org/xerces-c) and the Visualization Toolkit (VTK) from Kitware for visualization (www.vtk.org). Use of plug-in technology will allow low-level computational components such as integrators and optimizers to be updated as appropriate without extensive restructuring.

1.2 Capabilities

OpenSim includes a wide variety of features. You can find out about them by completing the tutorials and browsing the user guide and this handout. Some of the most useful features include:

- Scaling a Model
- Performing Inverse Kinematics Analyses
- Performing Inverse Dynamics Analyses
- Performing Static Optimization Analyses
- Generating Forward Dynamics Simulations
- Analyzing Dynamic Simulations
- Plotting Results
- Creating Snapshots and Making Animations

1.3 Model and Simulation Repository

You can create your own models of musculoskeletal structures and dynamic simulations of movement in OpenSim, as well as take advantage of computer models and dynamic simulations that other users have developed and shared. For example, you can use existing computer models of the human lower limb, upper limb, cervical spine, and whole body which have already been developed and posted at https://simtk.org/home/nmblmodels. You can also use dynamic simulations of walking and other activities that have been developed, tested and posted on Simtk.org. We encourage you to share your models and simulations with the research community by setting up a project on SimTK.org.

1.4 Compatibility with SIMM

SIMM (Software for Interactive Musculoskeletal Modeling) from Motion Analysis Corp. is a widely used software application for biomechanical simulation, surgical planning, and ergonomic analysis. The joint (*.jnt) and muscle (*.msl) files used by SIMM to describe models of the musculoskeletal system can be converted into OpenSim models (*.osim) and

brought into the OpenSim framework, thus allowing users of OpenSim to build on the wealth of models built and validated in SIMM. In this way, OpenSim complements SIMM by enabling forward dynamics simulations of models without third party software or the need to compile your own source code.

1.5 Additional Resources and Help

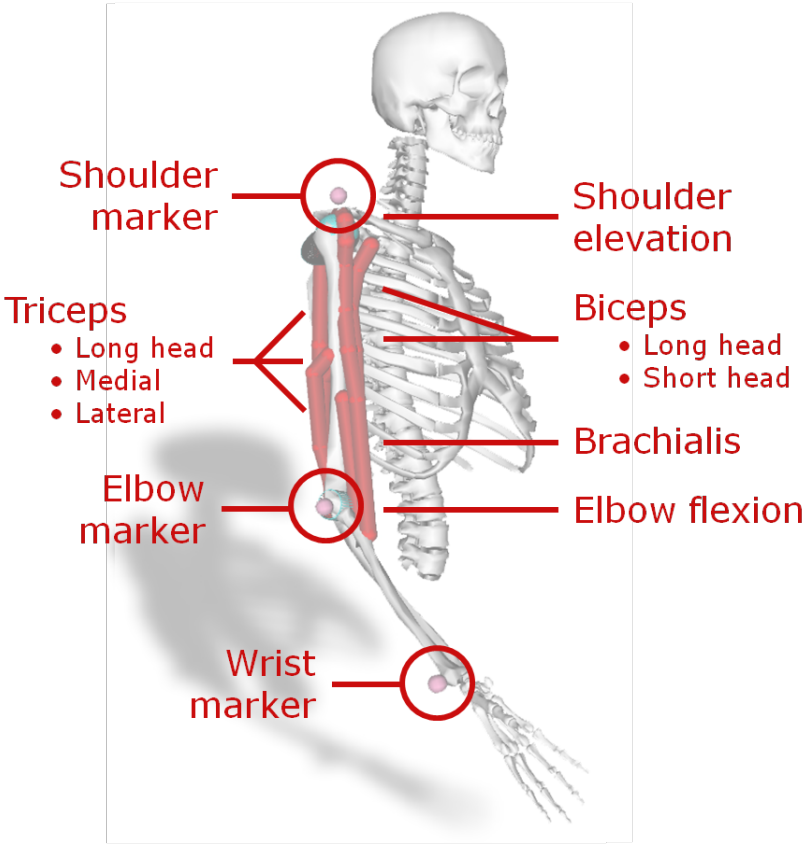
You can learn more at the OpenSim project site at http://simtk.org/home/opensim. The project site provides a forum for users to ask questions and share expertise. You can also get additional information in the following article: Delp, S.L., Anderson, F.C., Arnold, A. S., Loan, P., Habib, A., John, C., Guendelman, E.G., Thelen, D.G., OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940-1950, 2007.

2 Workshop Overview

This workshop provides an introduction to some of the many tools available within OpenSim, a freely available package for musculoskeletal modeling and dynamic simulation of movement. For more information on OpenSim, visit the OpenSim project site at http://simtk.org/home/opensim. The project site provides a forum for users to ask questions and share expertise.

This workbook serves as a summary and guide for the workshop. In the first hour, we will focus on inverse dynamics to compute the joint torques necessary to produce joint angles from inverse kinematics analyses; in addition, we will cover static optimization to compute muscle forces necessary to produce joint torques from inverse dynamics. In the second hour, we will focus on forward dynamic simulations that help us understand muscle actions. In the final hour, we will focus on interfacing OpenSim models with MATLAB/Simulink for closed-loop control of the simulated movement.

To present some of the tools and capabilities of OpenSim, we will use a simplified model (arm26) throughout this workbook. The model consists of the torso, upper arm, and forearm segments along with the triceps brachii (long, medial, and lateral heads), biceps brachii (long and short heads), and brachialis muscles. This simple model is not intended for research.



arm26 (2 coordinates & 6 muscles) model

3 Inverse Dynamics

Key Concepts

- Kinematics: coordinates and their velocities and accelerations
- Kinetics: forces and torques
- Dynamics: equations of motion

3.1 How it Works

The Inverse Dynamics Tool steps through each time frame of a motion and computes the net forces and/or torques at each joint in the model that generate the experimental kinematics. The equations of motion relate the model accelerations to the forces and/or joint torques applied to the model.

3.1.1 Kinematics: Coordinates and their Velocities and Accelerations

A *coordinate* is a joint angle or distance that specifies the relative orientation or location of two body segments in the model. The derivative (rate of change) of a coordinate with respect to time is the coordinate's *velocity*. In turn, the time derivative of its velocity is the coordinate's *acceleration*. The collection of coordinates and their velocities and accelerations describe the *kinematics* of the model.

3.1.2 Kinetics: Forces and Torques

Forces and *torques* cause the model to accelerate according to Newton's second law. A force can be applied to points (e.g., ground reactions) or between points (e.g., muscles) on the model. A torque can be applied to a coordinate (e.g., joint torque) to accelerate that joint angle directly.

3.1.3 Dynamics: Equations of Motion

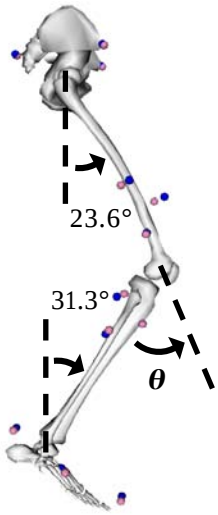
From Newton's second law, we can determine the kinetics necessary to accelerate the model by treating the skeleton as a set of interconnected rigid-bodies with inertial properties such that:

$$\underbrace{\boldsymbol{\tau}}_{\text{unknowns}} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{F}}_{\text{knowns}}$$

where $\boldsymbol{\tau}$ is the set of joint torques, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ are the coordinates and their velocities and accelerations, respectively; $\boldsymbol{M}(\boldsymbol{q})$ is the mass matrix, which depends on the coordinates and inertial properties of the model; $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the combination of Coriolis and centrifugal forces, which depend on the coordinates and their velocities; $\boldsymbol{G}(\boldsymbol{q})$ is the gravitational force, which depends on the coordinates; and $\boldsymbol{F}$ is other forces applied to the model.

The motion of the model is completely defined by the coordinates and their velocities and accelerations. Consequently, all of the terms on the right-hand side of the equations of motion are known. The remaining set of joint torques on the left-hand side is unknown. The Inverse Dynamics Tool uses the known motion of the model to solve the equations of motion for the unknown joint torques.

Exercise



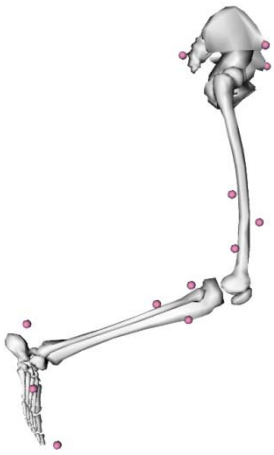
1. For the model shown on the right, what is the value (θ) of the knee coordinate (*Note: extension is* +)?

   A. 23.6° B. -54.9° C. 31.3° D. -125.1°

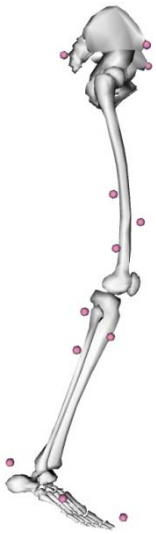
2. Given that the model shown on the right is at rest, what is the velocity of the knee?

   A. 23.6°/s B. -54.9°/s C. 3.89°/s D. 0°/s

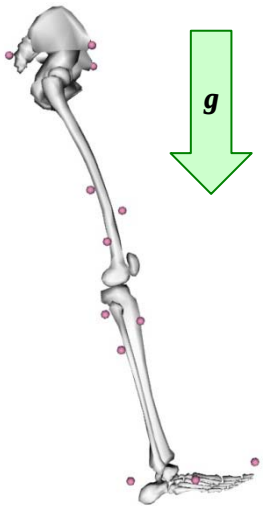
3. For the model poses shown below at rest and with gravity (***g***) as the only force acting on the model, which pose requires the largest torque at the knee joint?



A.



B.



C.

3.2 Inverse Dynamics Tool

To launch the Inverse Dynamics Tool select **Inverse Dynamics...** from the **Tools** menu. The **Inverse Dynamics Tool** dialog, like all other OpenSim tools, operates on the *Current Model* open and selected in OpenSim.

3.2.1 Inputs

Two items are required as input for the Inverse Dynamics Tool:

- The current model loaded in OpenSim.
- Motion file containing the time histories of coordinates that describe the movement of the model. This file may be generated by the Inverse Kinematics Tool.

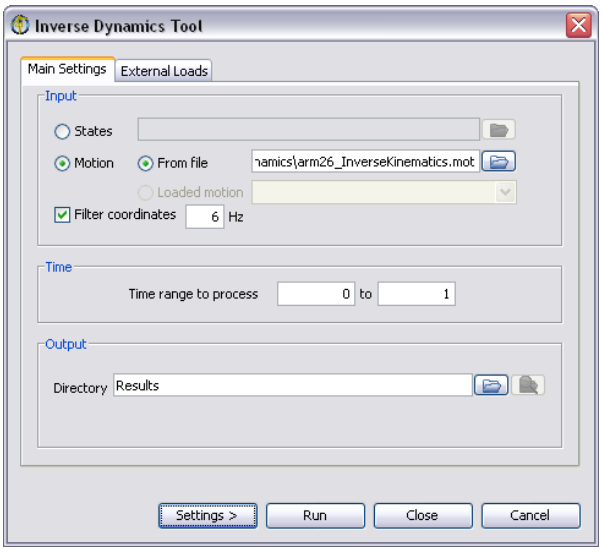
One *additional* item may be provided to the Inverse Dynamics Tool: motion file containing external loads (e.g., ground reaction forces).

3.2.2 Outputs

The Inverse Dynamics Tool generates a single storage file containing the time histories of the the net forces and/or torques at each joint responsible for the movement of the model.

Practice: Elbow moment and filtering

Main Settings



1. What is the subject's elbow flexion moment at peak elbow flexion during the motion?
2. By changing the filter cutoff frequency from 6 Hz to 2Hz, how does this change the subject's elbow flexion moment at peak elbow flexion?
3. Without filtering the motion, what is the subject's elbow flexion moment at peak elbow flexion?
4. Is it practical to perform inverse dynamics without filtering the motion?

4 Static Optimization

Key Concepts

- Kinematics: coordinates and their velocities and accelerations
- Kinetics: muscle forces
- Muscle physiology: muscle activation-contraction and force-length-velocity relations
- Dynamics: equations of motion
- Musculoskeletal geometry: muscle moment arm
- Optimization: the “distribution” problem

4.1 How it Works

The Static Optimization Tool steps through each time frame of a motion and computes the muscle activations in the model that generate the experimental kinematics. The equations of motion relate the model accelerations to the muscle moments applied to the model. The musculoskeletal geometry relates the muscle moments to the muscle forces generating those moments. The muscle contraction dynamics relate the muscle forces to the muscle activations.

4.1.1 Kinematics: Coordinates and their Velocities and Accelerations

A *coordinate* is a joint angle or distance that specifies the relative orientation or location of two body segments in the model. The derivative (rate of change) of a coordinate with respect to time is the coordinate’s *velocity*. In turn, the time derivative of its velocity is the coordinate’s *acceleration*. The collection of coordinates and their velocities and accelerations describe the *kinematics* of the model.

4.1.2 Kinetics: Muscle Forces

Muscle forces cause the model to accelerate according to Newton's second law. A muscle force is applied between origin and insertion points on the model.

4.1.3 Muscle Physiology: Muscle Activation-Contraction and Force-Length-Velocity Relations

Muscle activation (a_m) triggers a biochemical reaction that causes a muscle's fibers to *contract*. An inactive muscle has an *activation* $a_m = 0$. As more muscle fibers *contract*, the muscle *contraction* produces more *force*. The maximum *force* is produced with an *activation* $a_m = 1$. Therefore, *muscle activation* determines the relative *muscle contraction* to produce various *muscle forces*.

Muscle forces are also affected by the *muscle length* and *muscle velocity*. The optimal fiber *length* (l_o) is where a muscle actively produces maximum *force*. The force production varies as a function of fiber *length*. When the muscle fiber *length* is shorter than $0.5 l_o$ or longer than $1.5 l_o$, the active muscle *force* diminishes to zero. Zero velocity is where a muscle actively produces peak isometric *force* (F_0^M). As a muscle lengthens with a lengthening velocity, the active muscle *force* asymptotically increases to $1.8 F_0^M$. On the contrary, as a muscle shortens with a shortening velocity, the active muscle *force* decreases and the *force* becomes zero at the muscle's maximum contraction *velocity*.

4.1.4 Dynamics: Equations of Motion

From Newton's second law, we can determine the kinetics necessary to accelerate the model by treating the skeleton as a set of interconnected rigid-bodies with inertial properties such that:

$$\underbrace{\boldsymbol{\tau}}_{\text{unknowns}} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{F}}_{\text{knowns}}$$

where $\boldsymbol{\tau}$ is the set of joint torques, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ are the coordinates and their velocities and accelerations, respectively; $\boldsymbol{M}(\boldsymbol{q})$ is the mass matrix, which depends on the coordinates and inertial properties of the model; $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the combination of Coriolis and centrifugal forces, which depend on the coordinates and their velocities; $\boldsymbol{G}(\boldsymbol{q})$ is the gravitational force, which depends on the coordinates; and $\boldsymbol{F}$ is other forces applied to the model.

The motion of the model is completely defined by the coordinates and their velocities and accelerations. Consequently, all of the terms on the right-hand side of the equations of motion are known. The remaining set of joint torques on the left-hand side is unknown. The Static Optimization Tool uses the known motion of the model to solve the equations of motion for the unknown joint torques.

4.1.5 Musculoskeletal Geometry: Muscle Moment Arm

A *muscle moment arm* is the perpendicular distance from the line of action of a muscle to the joint center of rotation. The *moment arm* transforms the linear force of muscle into angular moment about a joint center.

4.1.6 Optimization: The "Distribution" Problem

As described earlier, the motion of the model is completely defined by the coordinates and their velocities and accelerations. The "distribution" problem arises from the fact there are more unknown muscle forces than the number of coordinates. The Static Optimization Tool uses the known motion of the model to solve the equations of motion for the unknown generalized forces (e.g., joint torques) subject to one of the following constraints:

$$\underbrace{\sum_{m=1}^{nm}\left(a_m F_m^0\right) r_{m,j} = \tau_j}_{\text{ideal force generators}} \quad \text{or} \quad \underbrace{\sum_{m=1}^{nm}\left[a_m f\left(F_m^0, l_m, v_m\right)\right] r_{m,j} = \tau_j}_{\text{constrained by force-length-velocity properties}}$$

while minimizing the objective function:

$$J = \sum_{m=1}^{nm} \left(a_m\right)^p$$

where *nm* is the number of muscles in the model; a_m is the activation level of muscle *m* at a discrete time step; F_m^0 is its maximum isometric force; l_m is its length; v_m is its shortening velocity; $f\left(F_m^0, l_m, v_m\right)$ is its force-length-velocity surface; $r_{m,j}$ is its moment arm about the j^{th} joint axis; τ_j is the generalized force acting about the j^{th} joint axis; and *p* is a user defined constant.

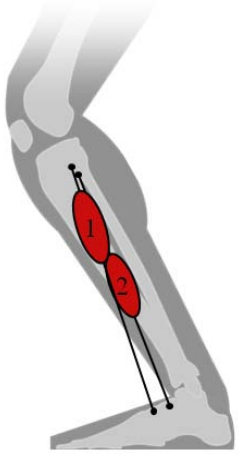
Exercise

1. Given that the rectus femoris muscle has a peak isometric force of 1169 N and it is at its optimal fiber length and zero velocity, what is the force generated for an activation of 0.86?

   A. 164 N B. 952 N C. 1005 N D. 1058 N

2. For the model shown on the right, which muscle has the largest moment arm about the ankle joint?

   A. 1 B. 2 C. Neither (are identical)



3. For the model shown on the right, which muscle has the largest moment arm about the knee joint?

   A. 1 B. 2 C. Neither (are identical)

4. For the model shown above, muscle 1 and 2 have the following properties:

Muscle	Peak Isometric Force (N)	Moment Arm (cm)
1	905	3.6
2	512	3.0

   To solve the "distribution" problem minimizing the sum of squared activations, which muscle would be activated more for a given dorsiflexion moment?

   A. 1 B. 2 C. Neither (are identical)

4.2 Static Optimization Tool

To launch the Static Optimization Tool, select **Static Optimization...** from the **Tools** menu. The **Static Optimization Tool** dialog, like all other OpenSim tools, operates on the *Current Model* open and selected in OpenSim.

4.2.1 Inputs

Two items are required as input for the Static Optimization Tool:

- The current model loaded in OpenSim.
- Motion file containing the time histories of coordinates that describe the movement of the model. This file may be generated by the Inverse Kinematics Tool.

One *additional* item may be provided to the Static Optimization Tool: motion file containing external loads (e.g., ground reaction forces).

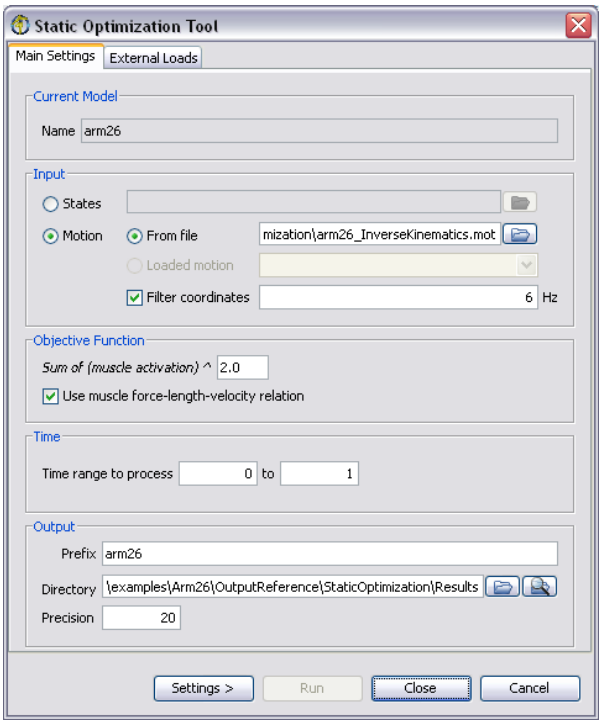
4.2.2 Outputs

The Static Optimization Tool generates three files:

- Controls file containing the time histories of muscle activations. These controls were minimized by the Static Optimization Tool.
- Storage file containing the time histories of muscle activations.
- Storage file containing the time histories of muscle forces.

Practice: Biceps muscle force and physiology

Main Settings



1. What is the force produced by biceps brachii long head at peak elbow flexion during the motion?
2. By changing the objective function to ignore the muscle force-length-velocity relation, how does this change the force produced by biceps brachii long head at peak elbow flexion?
3. Does the change (or not much change) in muscle force produced without the muscle force-length-velocity relation make sense in this case?

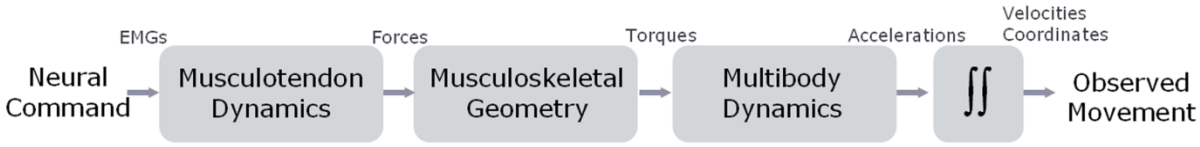
5 Forward Dynamics

Key Concepts

- Musculoskeletal model dynamics
- States of a musculoskeletal model
- Controlling a musculoskeletal simulation
- Numerical integration of dynamical equations

5.1 How it Works

The Forward Dynamics Tool uses the model together with the initial states and controls to run a muscle-driven forward dynamics simulation. A forward dynamics simulation is the solution (integration) of the differential equations that define the dynamics of a musculoskeletal model.



5.1.1 Musculoskeletal Model Dynamics

In contrast to inverse dynamics where the motion of the model was known and we wanted to determine the forces and torques that generated the motion, in forward dynamics, a mathematical model describes how coordinates and their velocities change due to applied forces and torques (moments).

From Newton's second law, we can describe the accelerations (rate of change of velocities) of the coordinates in terms of the inertia and forces applied on the skeleton as a set of rigid-bodies:

$$\ddot{\boldsymbol{q}} = \left[\boldsymbol{M}(\boldsymbol{q})\right]^{-1}\left\{\boldsymbol{\tau} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}\right\} \qquad \text{Multibody dynamics}$$

where $\ddot{\boldsymbol{q}}$ is the coordinate accelerations due to joint torques, $\boldsymbol{\tau}$, Coriolis and centrifugal forces, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$, as a function of coordinates, $\boldsymbol{q}$, and their velocities, $\dot{\boldsymbol{q}}$, gravity, $\boldsymbol{G}(\boldsymbol{q})$, and other forces applied to the model, $\boldsymbol{F}$, and $\left[\boldsymbol{M}(\boldsymbol{q})\right]^{-1}$ is the inverse of the mass matrix.

$$\boldsymbol{\tau}_m = \left[\boldsymbol{R}(\boldsymbol{q})\right]\boldsymbol{f}(\boldsymbol{a},\boldsymbol{l},\dot{\boldsymbol{l}}) \qquad \text{Moments due to muscle forces}$$

$$\dot{\boldsymbol{l}} = \Lambda(\boldsymbol{a},\boldsymbol{l},\boldsymbol{q},\dot{\boldsymbol{q}}) \qquad \text{Muscle contraction dynamics}$$

$$\dot{\boldsymbol{a}} = \mathrm{A}(\boldsymbol{a},\boldsymbol{x}) \qquad \text{Muscle activation dynamics}$$

The net muscle moments, $\boldsymbol{\tau}_m$, in turn, are a result of the moment arms, $\boldsymbol{R}(\boldsymbol{q})$, multiplied by muscle forces, $\boldsymbol{f}$, which are a function of muscle activations, $\boldsymbol{a}$, and muscle fiber lengths, $\boldsymbol{l}$, and velocities, $\dot{\boldsymbol{l}}$. Muscle fiber velocities are governed by muscle contraction dynamics, Λ, which is dependent on the current muscle activations and fiber lengths as well as the coordinates and their velocities. Activation dynamics, A, describes how the activation rates, $\dot{\boldsymbol{a}}$, of the muscles respond to input neural excitations, $\boldsymbol{x}$, generally termed the model's controls. These form a set of differential equations that model *musculoskeletal dynamics*.

5.1.2 States of a Musculoskeletal Model

The *state* of a model is the collection of all model variables defined at a given instant in time that are governed by dynamics. The model dynamics describe how the model will advance from a given state to another. In a *musculoskeletal model* the states are the coordinates and their velocities and muscle activations and muscle fiber lengths. The dynamics of a model require the state to be known in order to calculate the rate of change of the model states (joint accelerations, activation rates, and fiber velocities) in response to forces and controls.

5.1.3 Controlling a Musculoskeletal Model

The forces (e.g., muscles) in a *musculoskeletal model* are governed by dynamics and have inputs that affect their behavior. In OpenSim, these inputs are called the *controls* of a model, which can be excitations for muscles or torque generators. Ultimately, controls determine the forces and/or torques applied to the model and therefore determine the resultant motion.

5.1.4 Numerical Integration of Dynamical Equations

A simulation is the integration of the musculoskeletal model's dynamical equations starting from a user-specified initial state. After applying the controls, the activation rates, muscle fiber velocities, and coordinate accelerations are computed. Then, new states at a small time interval in the future are determined by numerical integration. A 5^{th}-order Runge-Kutta-Feldberg integrator is used to solve (numerically integrate) the dynamical equations for the trajectories of the musculoskeletal model states over a definite interval in time. The Forward Dynamics Tool is an open-loop system that applies muscle/actuator controls with no feedback, or correction mechanism, therefore the states are not required to follow a desired trajectory.

Exercise

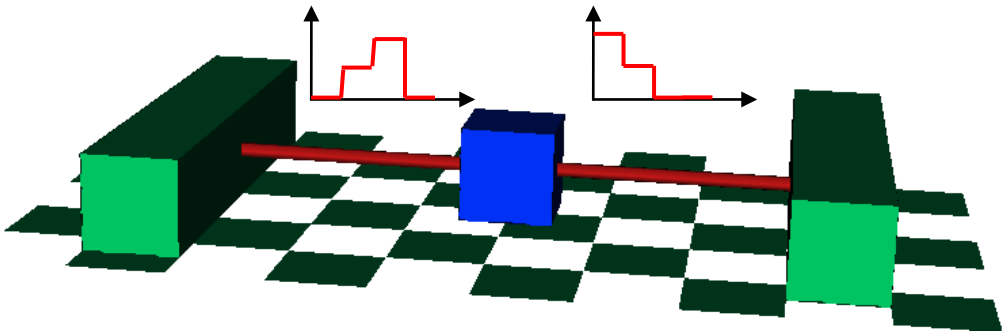
1. A forward dynamics simulation is

   A. a musculoskeletal model
   B. muscle-driven
   C. a simulation that uses feed-back
   D. the integration of dynamical equations

2. The musculoskeletal model for the workshop (arm26) has how many states?

   A. 3 B. 9 C. 12 D. 16

3. Given the model below with two identical muscles and their levels of control plotted versus time, which way will the block initially move if starting from rest?



A. To the left B. To the right C. Does not move D. Upward

4. Given initial coordinates and their velocities and muscle activations and muscle fiber lengths, how are these states determined at a small instant ahead in time?

   A. Specify controls and compute muscle activation rates, fiber velocities, and coordinate accelerations from model dynamics
   B. Numerically integrate forces and controls from model differential equations
   C. Numerically integrate muscle activation rates, fiber velocities, and coordinate accelerations
   D. Numerically differentiate forces and controls from the dynamical equations
   E. A & C

5.2 Forward Dynamics Tool

To launch the Forward Dynamics Tool select **Forward Dynamics...** from the **Tools** menu. The **Forward Dynamics Tool** dialog like all other OpenSim tools operates on the *Current Model* open and selected in OpenSim.

5.2.1 Inputs

Three items are required by the Forward Dynamics Tool:

- The current model loaded in OpenSim
- XML file containing the time histories of the model controls (e.g., muscle excitations) to the muscles and/or joint torques. This file may be generated by the user, Static Optimization Tool, or Computed Muscle Control Tool.
- Storage file containing the initial states of the musculoskeletal model that includes coordinates and their velocities and muscle activations and muscle fiber lengths

Two *additional* items may be provided to the Forward Dynamics Tool:

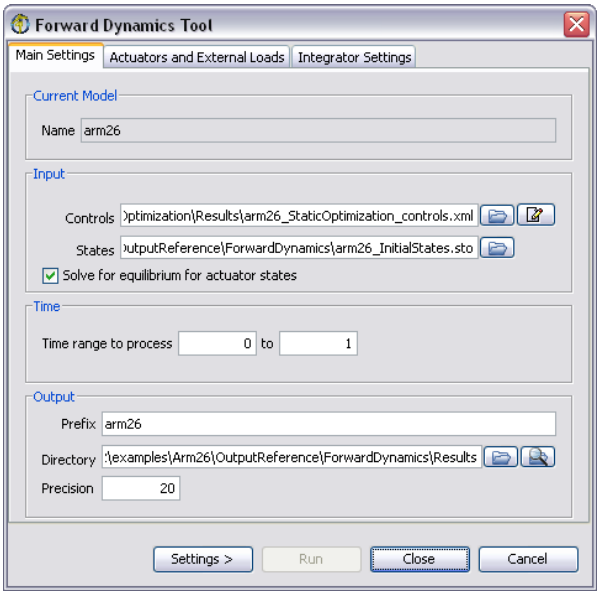
- Motion file containing external loads (e.g., ground reaction forces)
- Settings for numerical integration

5.2.2 Outputs

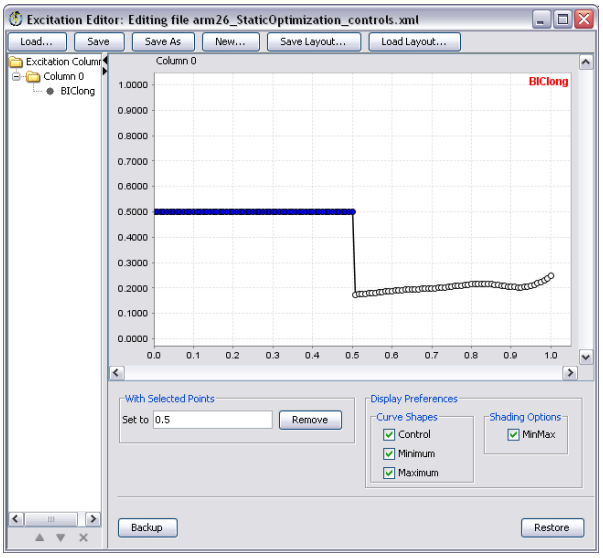
The Forward Dynamics Tool generates a storage file containing the time histories of the model's controls and states that result from integration of the model's dynamical equations.

Practice: Muscle-driven elbow flexion simulation

Main Settings



Excitation Editor



1. Using activations from static optimization as controls, what are the shoulder and elbow angles at end of the motion? How do they compare to angles determined from inverse kinematics?
2. By exciting a single muscle early in the motion, can you exceed the elbow flexion determined from inverse kinematics? If yes, what muscle?
3. By exciting a second muscle later in the motion, can you achieve the shoulder elevation and elbow flexion from inverse kinematics? If so, which muscle?
4. What methods could you use to automate the process of creating a forward dynamics simulation from the inverse kinematics results?

6 Interfacing OpenSim with MATLAB/Simulink

Key Concepts

- S-function: system function
- Simulink block: inputs, states, and outputs
- Forward dynamics: controls, states, and integration
- Feedback: closed-loop control

6.1 How it Works

The OpenSim interface block interacts with Simulink similar to built-in Simulink blocks by using the S-function Application Programmer Interface (API); moreover, the block's underlying computer language description interacts with OpenSim's dynamically linked libraries using the OpenSim API. Simulink solvers (integrators) generate the forward dynamics simulation by solving (integrating) the differential equations generated by OpenSim that define the dynamics of a musculoskeletal model.

6.1.1 S-Function: System Function

An *S-function* is a system function that extends the capabilities of Simulink through a computer language description of a Simulink block written in MATLAB, C, C++, or Fortran and compiled as a MEX-file. An S-function is compiled as a dynamically linked subroutine that MATLAB can automatically load and execute. After an S-function is compiled, the user can customize it using Simulink's block masking tools.

6.1.2 Simulink Block: Inputs, States, and Outputs

A *block* is the element used by Simulink to build its models and represents elementary dynamic systems that Simulink knows how to simulate. Every *block* consists of a set of *inputs*, a set of *states*, and a set of *outputs*. The *inputs* are variables whose values come from other Simulink blocks, the MATLAB workspace, or a file. The *states* are variables that determine a *block's outputs* and whose current values are a function of the previous values of the *block's states* and/or *inputs*. The *block's outputs* are a function of the simulation time, the *inputs*, and the *block's states*.

6.1.3 Forward Dynamics: Controls, States, and Integration

The *controls* of a model are inputs that can be excitations for muscles or torque generators. The forces (e.g., muscles) in a musculoskeletal model are governed by dynamics and the *controls* are the inputs that affect their behavior. Ultimately, *controls* determine the forces and/or torques applied to the model and therefore determine the resultant motion. The *controls* for an OpenSim model are identical to the input signal for the OpenSim interface block in Simulink.

The *states* of a model are the collection of all model variables defined at a given instant in time that are governed by dynamics. In a musculoskeletal model, the *states* are the coordinates and their velocities and muscle activations and muscle fiber lengths. The dynamics of a model require the state to be known in order to calculate the derivative of the model states (joint accelerations, activation rates, and fiber velocities) in response to forces and controls. The *states* for an OpenSim model are identical to the states for the OpenSim interface block in Simulink.

The *integration* of the musculoskeletal model's dynamical equations generates a simulation. After applying the *controls*, *state derivatives* (i.e., activation rates, muscle fiber velocities, and coordinate accelerations) are computed. Then, new *states* at small time interval in the future are determined by *numerical integration*. In OpenSim, a 5^{th}-order Runge-Kutta-Feldberg *integrator* is used to solve (*numerically integrate*) the dynamical equations for the trajectories of the musculoskeletal model *states* over a definite interval in time. On the other hand, the OpenSim interface block in Simulink relies on MATLAB *integrators* to solve these equations. A forward

dynamic *integration* alone is an open-loop system that applies muscle/actuator *controls* with no feedback, or correction mechanism, therefore the *states* are not required to follow a desired trajectory.

6.1.4 Feedback: Closed-Loop Control

In a *closed-loop control* system, the input variable is adjusted by the controller in order to minimize the error between the measured output variable and its reference set point. This *control* design is synonymous to *feedback control*, in which the deviations between the measured variable and a reference set point are fed back to the controller to generate appropriate *control* actions. The controller takes the difference between the reference set point and the output to change the inputs to the system in order to have the measured outputs go to the reference set point value.

Exercise

1. What computer language can an S-function be written in?

   A. MATLAB B. C C. C++ D. All of the above and Fortran

2. What are the three basic parts of every Simulink block?

   A. Coordinates, velocities, and accelerations
   B. Muscle force, length, and velocity
   C. Inputs, states, and outputs
   D. Controls, states, and integration

3. Which software provides the integrator to solve the dynamical equations for the model states when using the OpenSim interface block in Simulink?

   A. OpenSim B. MATLAB C. Simulink D. None of the above

4. What control system design feature allows a controller to take the difference between the reference set point and the block output to change the inputs to the system?

   A. Closed loop B. Open loop C. Feedback D. A and C

6.2 OpenSim interface block in Simulink

To launch Simulink, you must first start the MATLAB technical computing environment. You can then start the Simulink software in two ways: 1) enter **simulink** in the MATLAB command window; or 2) click the Simulink icon ( ) on the MATLAB toolbar. To open an existing model, select **Open** from the Simulink library window's **File** menu and then choose or enter the file name for the model to edit. Alternatively, the **S-Function block** (found in Simulink Library > User-Defined Functions) may be included in a new model block diagram to access the S-function named as the S-function name parameter.

6.2.1 Inputs

Two items are required by the OpenSim interface block in Simulink:

- An OpenSim model file
- The initial states of the musculoskeletal model that includes coordinates and their velocities and muscle activations and muscle fiber lengths

One *additional* item may be provided to the OpenSim interface block in Simulink:

- Motion file containing external loads (e.g., ground reaction forces)

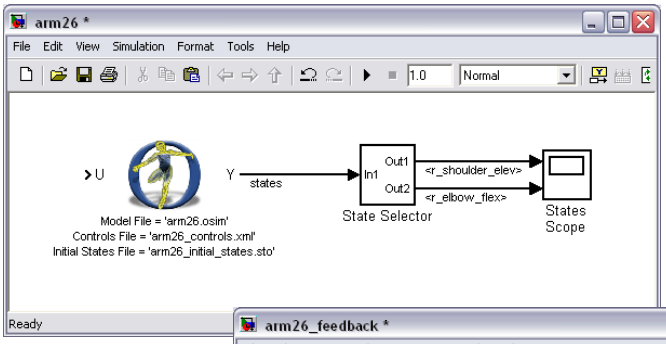
6.2.2 Outputs

The OpenSim interface block in Simulink generates a storage file containing the time histories of the model's states that result from integration of the model's dynamical equations.

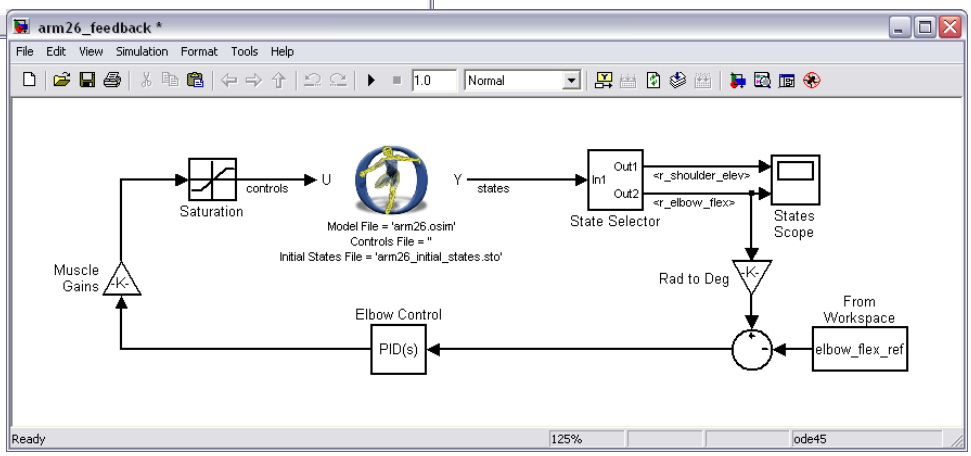
Practice: Elbow flexion feedback control

OpenSim Interface Block in Simulink

Open-Loop System



Closed-Loop System



1. Which file is a necessary input for the OpenSim interface block in Simulink in the open-loop system case, but not the closed-loop system case?
2. Which state of the model should be compared to the elbow flexion reference signal from the workspace?
3. Is a proportional controller with a gain of 1 sufficient to control the elbow flexion? If not, what type of controller would you suggest?
4. Can you develop and tune a control system that tracks the elbow flexion angle well while maintaining a shoulder elevation angle equal to 0°? If so, what does the control system involve?

7 Answers to Exercises and Practices

Inverse Dynamics

Exercise:	1. B	2. D	3. A	
Practice:	1. 2.73 Nm	2. 2.77 Nm	3. 2.45 Nm	4. No

Static Optimization

Exercise:	1. C	2. A	3. C	4. A
Practice:	1. 140.94 N	2. 141.06 N	3. Yes, the muscle is not short or long and does not approach maximum contraction velocity	

Forward Dynamics

Exercise:	1. D	2. D	3. B	4. E

Practice:

1. -15° (shoulder diminished elevation), 85° (elbow diminished flexion)
2. Yes, biceps brachii long head (note, other elbow flexors may work as well)
3. Yes, triceps brachii lateral head (note, other elbow extensors may work as well)
4. Optimization, computed muscle control, feedback control, etc.

Interfacing OpenSim with MATLAB/Simulink

Exercise: 1. D 2. C 3. B 4. D

Practice:

1. Controls file (in the closed-loop system Simulink determines controls)
2. State 1 (r_elbow_flex)
3. No, a proportional, integral, and derivative controller is a better choice (note, correct answers may vary)
4. Various answer may be correct