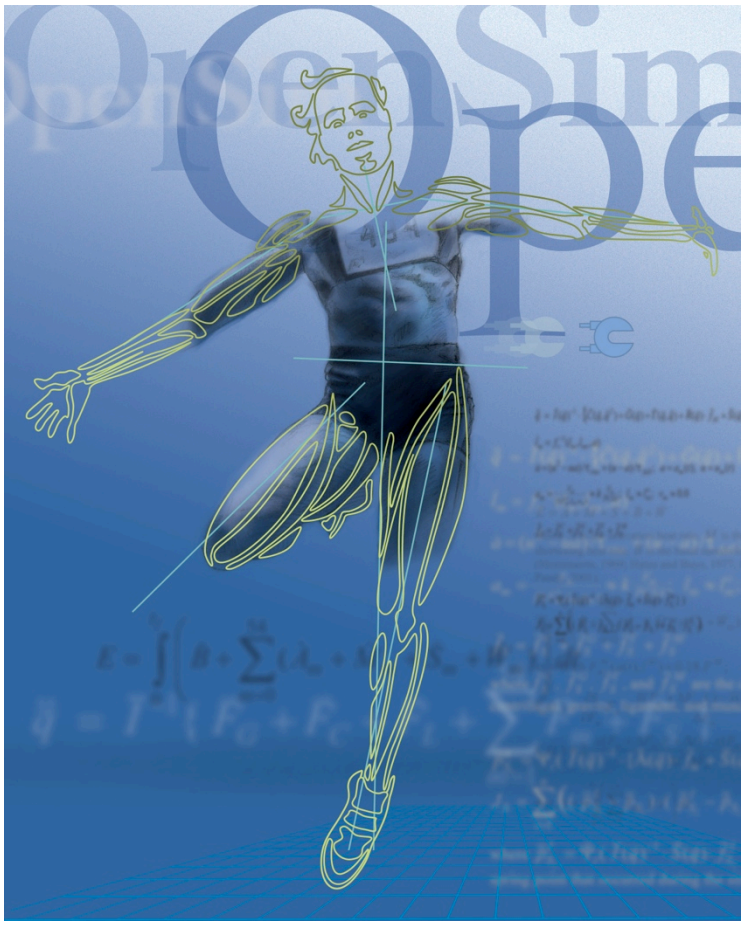
OpenSim



User's Guide

Release 2.4

April 4, 2012

Website: https://opensim.stanford.edu

User's Guide

Welcome to the OpenSim User's Guide! To begin learning more about how to use the OpenSim software, step through the pages in this user guide. The chapters are listed below. Or if you know what you would like to read about, use the search bar in the top right to expedite the process.

Chapters

Introducing OpenSim

This chapter gives an overview of the OpenSim software's capabilities and what is included in this user's guide. The topics covered in the section include:

Next: Welcome to OpenSim

What’s Covered in this Manual

Organization of the User's Guide

This manual is organized into four distinct sections. The first section covers installation of OpenSim. The next section describes how to use the basic components of the OpenSim graphical user interface (GUI). The third section covers the tools available within OpenSim, including plotting, scaling, inverse kinematics, inverse dynamics, forward dynamics, and computed muscle control. The last section describes how to prepare your data for use within OpenSim. The OpenSim Developer's Guide (available from the OpenSim webpages on Simtk.org), provides examples and tutorials for using OpenSim's C++ API.

Conventions in the User's Guide

Below are some of the conventions used within this document:
<- Arrows are used to indicate cascading menu selections
bold Bold-faced text is used to highlight menu choices, mouse button selections, and other actions
*italics*Names of panels and sections within an OpenSim window appear in italics
brown italics Brown italics are used to identify new vocabulary

Many of the OpenSim files use an Extensible Markup Language (XML) file format. In describing these files, the following formats are used:
<TagName> Tags appear in brown text surrounded by blue brackets
attribute Attributes associated with XML tags appear in red text
value Valid values for a set of tags are indicated in bold green

Next: Additional Resources and Help

Previous: Welcome to OpenSim

Home: Introducing OpenSim

Welcome to OpenSim

This page provides an introduction to the capabilities and features of OpenSim and how to get started:

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, and with version 2.0, an application programming interface (API) has been added, allowing researchers to access and customize OpenSim core functionality. Since the initial release, thousands of 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, biology, and 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 core software is written in C++, and the graphical user interface (GUI) is written in Java. OpenSim plug-in technology makes it possible to develop customized controllers, analyses, contact models, and muscle models among other things. These plugins 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 from within the GUI.

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 from Kitware for visualization (www.vtk.org). Use of plug-in technology allows computational components such as integrators and optimizers to be updated as appropriate without extensive restructuring. The GUI is written in Java and is built on the Netbeans Platorm (Netbeans.org).

Capabilities

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

- Taking pictures of musculoskeletal models and making animated movies: Snapshots and Movies
- Plotting results of your analysis: Plotting
- Scaling the size of a musculoskeletal model: Scaling
- Performing inverse kinematics analyses to calculate joint angles from marker positions: Inverse Kinematics
- Performing inverse dynamics analyses to calculate joint moments from joint angles and external forces: Inverse Dynamics
- Generating forward dynamics simulations of movement: Forward Dynamics
- Analyzing dynamic simulations: Analyses

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 Simtk.org. You can also use dynamic simulations of walking and other activities that have been developed, tested and posted on Simtk.org.

A collection of many of these models on Simtk.org can be found in the Neuromuscular Models Library.

We encourage you to share your models and simulations so that other researchers can build on your results. This will greatly accelerate research. Please set up a project on Simtk.org to share your results.

Compatibility with Simbody

OpenSim is built on top of the Simbody library, an open-source multibody dynamics engine developed to create mathematical models of biological dynamics. Simbody is being developed by Simbios, an NIH National Center for Biomedical Computation based at Stanford University. The purpose of Simbios is to enable groundbreaking biomedical research by providing open access to high-quality tools for modeling and simulating biological structures.

Simbody comes bundled with the LAPACK linear algebra library, IPOpt optimizer, in addition to the multibody dynamics engine. Simbody software and documentation are available at http://simtk.org/home/simbody.

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

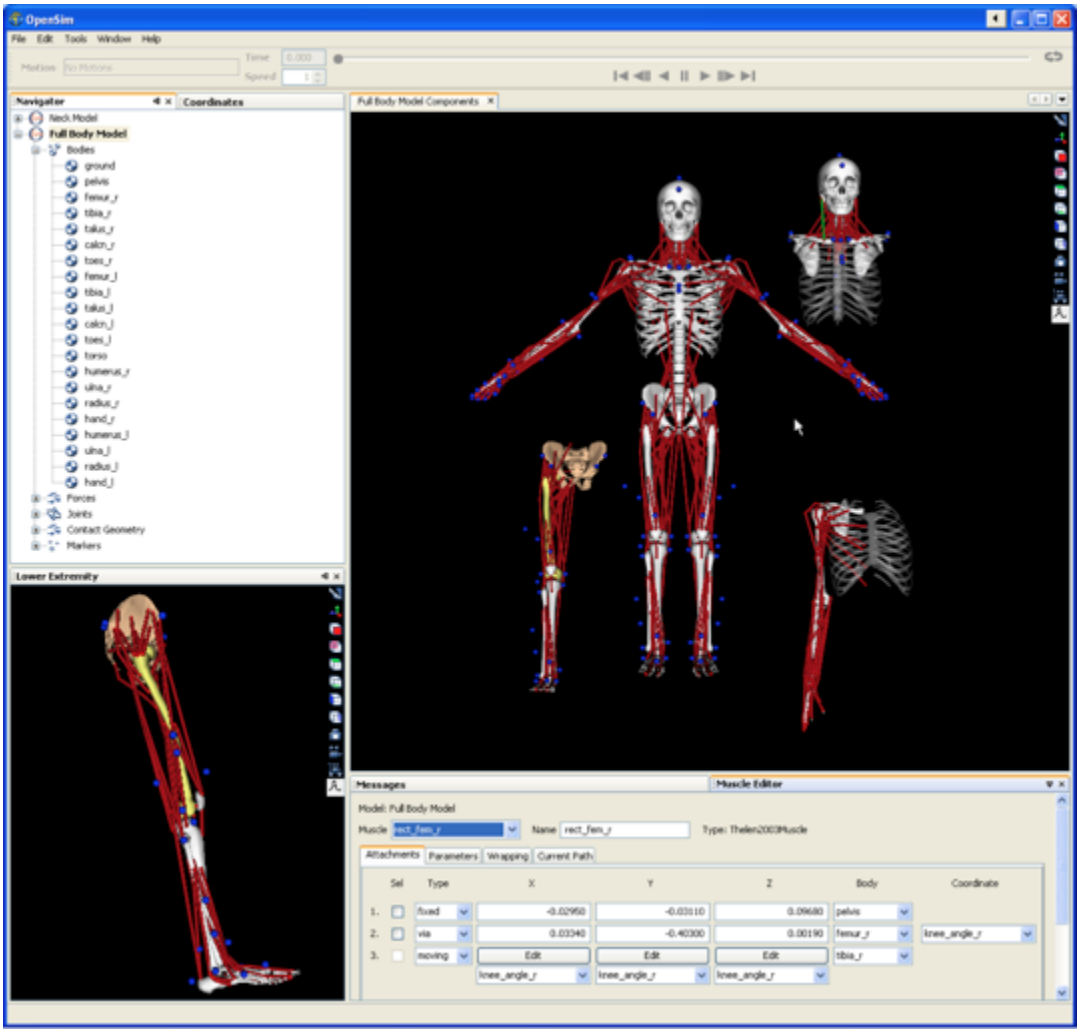
musculoskeletal system can be converted into OpenSim models **(*.osim)** and brought into the OpenSim framework.

OpenSim complements and augments the functionality of SIMM and the SIMM Dynamics Pipeline by providing advanced simulation and control capabilities. In addition, the object-oriented, modular design of OpenSim allows users to extend its functionality and share functionality with other OpenSim users.

OpenSim is a self-contained modeling and simulation environment that does not require additional software components or licenses to generate dynamic simulations.

OpenSim provides a graphical user interface (GUI) that provides access to many of the software features. For example, you can import motion analysis data, scale a computer model of the musculoskeletal system, perform inverse dynamics analysis, and plot results all from the graphical interface.

The OpenSim GUI



Screenshot from OpenSim.Models of many different musculo-skeletal structures, including the lower extremity, upper extremity, and neck, can be loaded, viewed and analyzed. Muscles are shown as red lines; virtual markers are shown as blue spheres.

How to Get Started

It's easy to get started with OpenSim. The instructions in the next section (Installation Guide) explain how to download and install the software on your computer.

The installation process generally takes less than few minutes. The tutorials, which come with the software, are an easy way to get started. Each tutorial takes about an hour to complete and covers a different aspect of the software and biomechanics.

Hundreds of people, from high school biology students to university professors, have completed these tutorials. More resources are also available on the website http://opensim.stanford.edu.

Next: What's Covered in this Manual

Home: Introducing OpenSim

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., Thelen, D.G., OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 1940-1950, 2007.

Next: Installation Guide

Previous: What's Covered in this Manual

Home: Introducing OpenSim

Installation Guide

This section covers how to install OpenSim on a computer running one of the following Microsoft® Windows® operating systems: Vista®, XP, Windows 7.

Next: Getting OpenSim

Supported Platforms

Supported Platforms

The name OpenSim refers to two distinct entities, the OpenSim API (which is a C++ library of classes used for modeling and simulation of biological structures in Biomechanics) and the OpenSim application (GUI) which builds on the OpenSim API. As of version 2.2, the only distribution of the GUI is on Windows (32Bit). Although there has been experimental builds of the GUI on Mac/Linux there's no public distribution.

The API (headers, libraries and Doxygen documentation) is available for both Windows (32Bit), Mac & Linux (32Bit) and is available on the downloads page. To use the API, users will need to write a main program or a plugin or use the various tools provided as command line executables to perform modeling/simulation.

Mac users, who want to use the GUI, have successfully ran OpenSim using Bootcamp, Virtual Machines or other environments that simulate Windows on Mac.

Mac and Linux users will need to add the installation-folder/bin to their environment (LD_LIBRARY_PATH on linux, DYLD_LIBRARY_PATH on mac) and can use the command line utilities or build main programs using the API following instructions similar to those given in the developer's guide for Windows.

If you're interested in helping to build a GUI distribution on Mac, Linux or any other platform, please contact the development team using the developers forum at https://simtk.org/forum/forum.php?forum_id=283

Next: Getting OpenSim

Home: Installation Guide

Getting OpenSim

OpenSim is available for download from Simtk.org. The following sections describe how you access OpenSim from the website:

- Log In to SimTK.org
- Download Installation Program

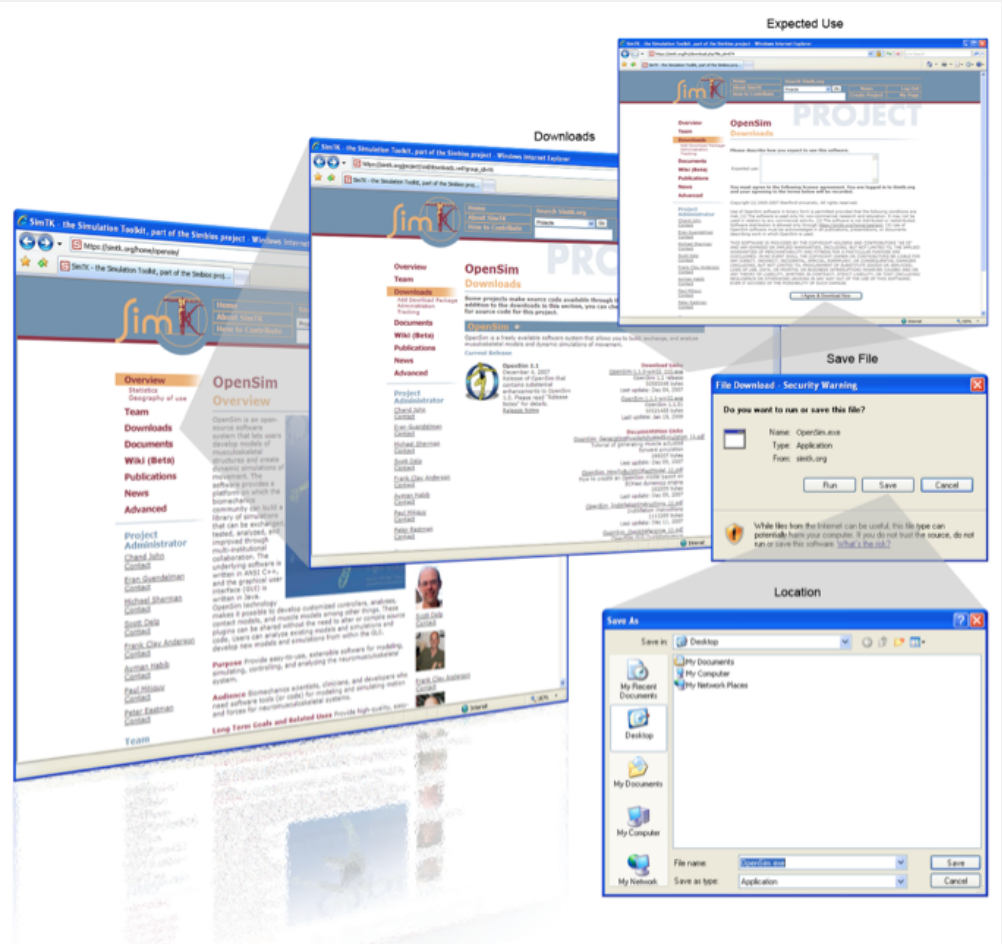
Log In to SimTK.org

1. Connect your browser to the Simtk.org (https://simtk.org) home page
2. If you are already a member of Simtk, you can log in by clicking the Log In link in the top-right corner of the Simtk.org home page. Otherwise, to register as a new member, click on the Register link in the top-right corner of the Simtk.org home page.

Additional details related to Simtk.org registration and login are available by clicking on the **About SimTK** link in the top-left border of the Simtk.org home page.

Download Installation Program

Download the OpenSim software from the Simtk.org website by following the steps below:



1. Connect your browser to t OpenSim project home pa https://simtk.org/home/ope).

2. Click on the **Downloads** li the left-hand column of the OpenSim project home pa You will jump to the Downloads page, where y can view current and previ releases of OpenSim.

3. From the Downloads page click on the link for the OpenSim executable, loca in the Download Links sec There will be different vers of OpenSim available for different API programming environments; however, if are only going to be using OpenSim GUI (and not the API), you can download ar the versions to use. This w be the latest release of OpenSim.

4. If you did not already log i new page will appear, requ you to log in. Visit Log In t SimTK.org to learn how to in.

5. A page will then appear as you to describe how you expect to use the software presenting the license agreement for OpenSim. I text field, describe your intended use of OpenSim then click the button to acc the license agreement.

6. A dialog box will then appe asking if you want to run o save the installation progra Click the **Save** button, choosing a convenient loc (e.g., your Desktop) to sav the installation program. If dialog box does not appea check your Browser securi settings, which may be set block pop-ups.

Next: Installing OpenSim

Previous: Supported Platforms

Home: Installation Guide

Installing OpenSim

To install OpenSim you will need to:

- Exit All Programs
- Run Installation Program

Exit All Programs

It is recommended that you exit all programs, especially existing copies of OpenSim, before you run the installation program. Also, you should temporarily disable virus detection software.

Run Installation Program

A Setup Wizard will guide you through the installation process.

1. To begin installing OpenSim, go to the location where you previously saved the installation program (e.g., your Desktop) and double click the file.
2. An Open File – Security Warning will appear. Confirm you want to run this software by clicking the **Run** button.
3. The OpenSim Setup Wizard will launch. Close all other applications, and then click the **Next** button to continue.
4. The license terms for OpenSim will be displayed. It is recommended that you review these terms before installing. If appropriate, click the **I Agree** button to continue.
5. A list of Install Options will appear. Choose "Add OpenSim to the system PATH for all users" and click the **Next** button to continue.
6. The next screen will ask you to select the folder in which to install OpenSim. Choose the folder and click the **Next** button to continue.
7. You will then be asked to choose a Start Menu folder for the OpenSim shortcuts. Select the folder and click the **Install** button to continue.
8. It may take several minutes to install OpenSim. Once OpenSim has been installed, click the **Finish** button to close the Setup Wizard and complete the installation.

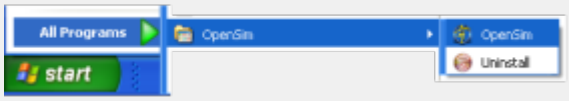
Next: Running OpenSim

Previous: Getting OpenSim

Home: Installation Guide

Running OpenSim

You can begin using OpenSim immediately after installation. To launch OpenSim:



1. Go to the folder you chose for the OpenSim shortcuts (e.g., All Programs > OpenSim).
2. Click on the icon for OpenSim.

Next: Graphical User Interface

Previous: Installing OpenSim

Home: Installation Guide

Graphical User Interface

This chapter introduces the basic components of the OpenSim Graphical User Interface (GUI), including menus, windows, and the toolbar:

- Menus
- Windows
- Toolbar

Next: Menus

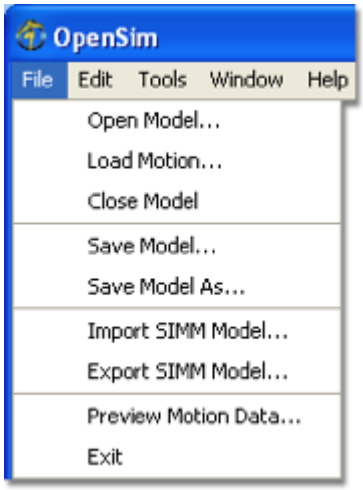
Menus

There are five drop-down menus available from the OpenSim main menu bar:

- File
- Edit
- Tools
- Window
- Help

File

The **File** drop-down menu includes the following options which allow you to input and output information about models and motions:

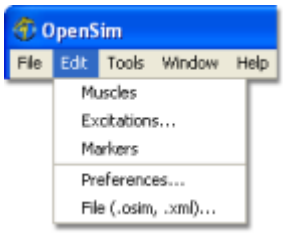


For more information about models (including SIMM models) and motions visit Loading and Saving Models and Motions

- For more information about motion data visit Previewing Motion Capture (Mocap) Data

Edit

The **Edit** drop-down menu is used to modify the following:

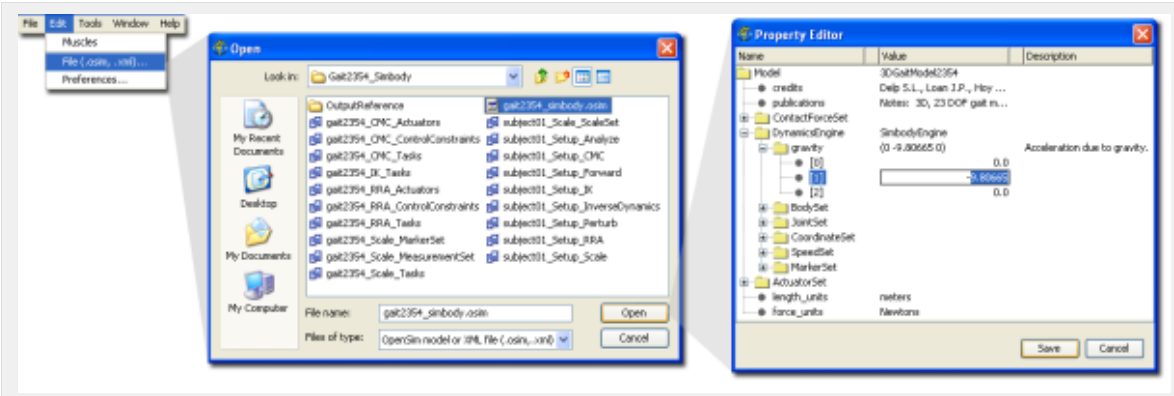


- For more information on editing muscles visit Muscle Editor
- For more information on editing excitations visit Excitation Editor
- For more information on editing markers visit Marker Editor
- For more information on changing preferences visit Preferences Editor
- For more information on editing files visit File Editor

File Editor

The File Editor allows you to edit the properties of the *.osim* and *.xml* files used in OpenSim. OpenSim model (*.osim)* files contain the actuators, bodies, joints, coordinates, and speeds of a model and use the XML (Extensible Markup Language) format. For more information about the model file formats visit Getting OpenSim. OpenSim settings (*.xml*) files contain, for example, settings for tools or additional data for models including marker sets, actuator sets, or control values. These also use the XML format.

To open the file editor:



1. Click **Edit File (.osim, .xml)...**
2. In the window that appears, locate and select the file you wish to edit.
3. A Property Editor window will open, listing all of the editable parameters, along with any corresponding descriptions, for that file. To change the value of a property, **double click** on the value, enter a new value, and press the **Enter** key.
4. To save the edited file when you are finished, click the **Save** button in the Property Editor window.

In the image above, the gravity property of the model is being edited.

Preferences Editor

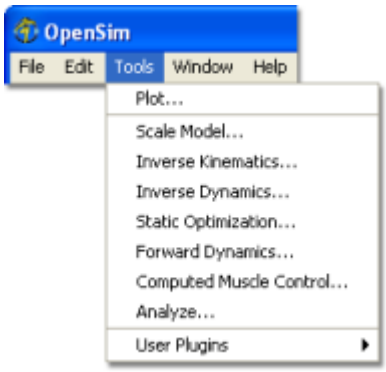
The Preferences Editor allows you to change user preferences for the 3D View window using the OpenSim GUI. To read about these properties described in detail visit the User Preferences page which includes the following options:

Background Color	Anti-Aliasing Frames
Markers Color	Non-current Model Opacity
Geometry File Path	Marker Display Radius
Model Display Offset Direction	Muscle Display Radius
Persist Models	Debug

- The "Persist Models" allows you to determine whether the OpenSim environment settings (what models have been loaded, views, coordinates, etc.) should be maintained upon closing OpenSim. If this is set to "On," then when you launch OpenSim again, you will see the exact same set-up as before. This is the default. To turn off this feature, set the value for "Persist Models" to "Off."
- An option that is helpful in troubleshooting operations is the "Debug" option. Setting it to a value more than 1 will cause more verbose messages to be printed to the message area when loading models.

Tools

The Tools menu enables you to easily perform the tasks needed to generate and analyze musculoskeletal simulations. The following tools are available in OpenSim:



- For more information on the plot tool visit Plotting
- For more information on scaling your model visit Scaling
- For more information on the inverse kinematics tool visit Inverse Kinematics
- For more information on the inverse dynamics tool visit Inverse Dynamics
- For more information on the static optimization tool visit Static Optimization
- For more information on the forward dynamics tool visit Forward Dynamics
- For more information on computing muscle control visit Computed Muscle Control
- For more information on the analyze tool visit Analyses
- For more information on plugins visit OpenSim User Plug-ins

OpenSim User Plug-ins

You can use the Tools menu to load in plugins. The plugin architecture allows you and other users to extend the OpenSim functionality. It is straightforward to use a plugin that you have written or has been shared with you (some existing plugins are available on http://simtk.org – search for opensim plugin):

1. Make sure the plugin has been built and tested on your platform. A plugin comes in the form of a dynamically loaded library (.dll on Windows platform, .dylib on MAC, .so on Linux).
2. Place the plugin under the *plugins* folder within your OpenSim installation folder (if you installed this in the default location on Windows, this will be *C:\Program Files\OpenSim <version number>\plugins*). You should then see the plugin as a menu option under the **User Plugins** menu.
3. From the **User Plugins** menu, click the name of the plugin to load it into OpenSim. You will be given the option to always preload the plugin each time OpenSim is launched. It is advised that you not do this until the plugin has been tested.
4. To remove the plugin from the menu or disable the loading of it by the GUI, remove the plugin file from the *plugins* folder.

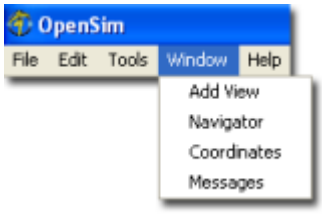
While it is outside the scope of this guide to provide details about writing plugins and supported platforms, a separate document (the OpenSim Developer's Guide) has been provided on the OpenSim downloads page specifically for this purpose.

GUI Plugins

The GUI is made up of a set of modules as well and these can be individually enabled/disabled to customize the application. The details of this are outside the scope of this guide.

Window

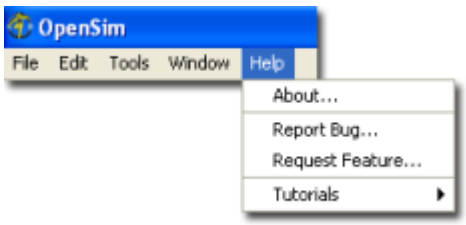
The Window menu controls what windows are displayed:



- For more information on adding views visit Creating and Naming 3D Views
- For more information on the navigator window visit Navigator Window
- For more information on the coordinates window visit Coordinates Window
- For more information on the messages window visit Messages Window

Help

The Help menu includes the following options:



- The **About** option opens a window giving more details about the software and includes acknowledgements and references.
- The **Report Bug…** and **Request Feature…** options will direct your default Internet browser to the Simtk.org website in order to report a bug or send a feature request to the OpenSim developers, respectively.
- Selecting a tutorial from the **Tutorials** list opens a PDF of an educational tutorial that introduces users to musculoskeletal modeling and simulation in OpenSim.

Next: Windows

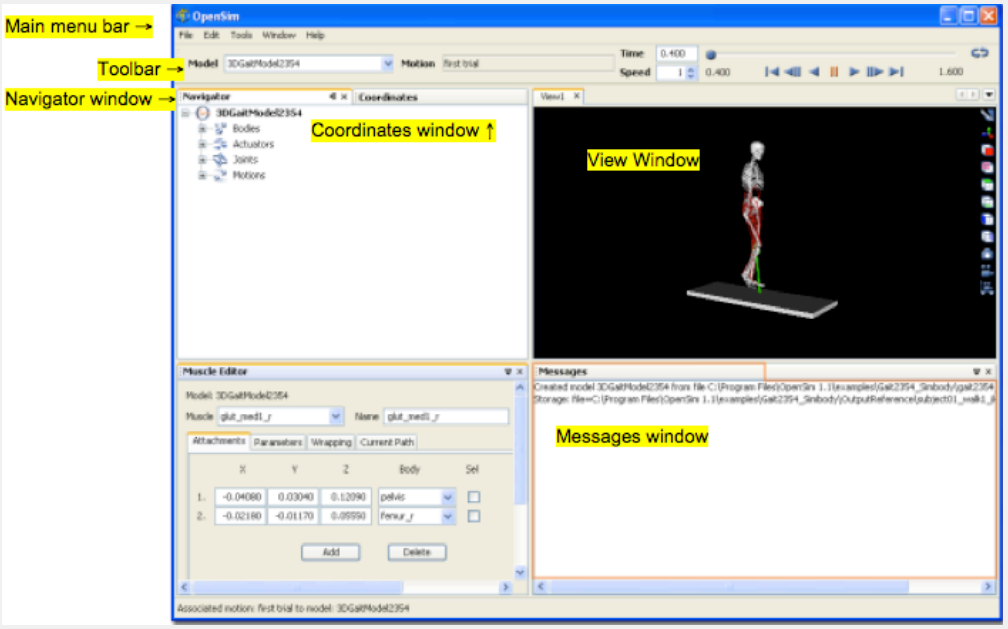
Home: Graphical User Interface

Windows

This section contains information about the four parts of the OpenSim window and one topic on formatting:

- Parts of the OpenSim Window
- View Window
- Navigator Window
- Coordinates Window
- Messages Window
- Docking

Parts of the OpenSim Window

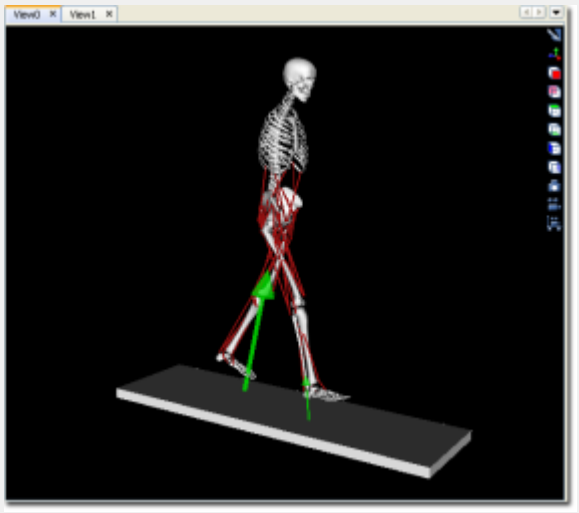


- For more information on the View Window visit View Window
- For more information on the Navigator Window visit Navigator Window
- For more information on the Coordinates Window visit Coordinates Window
- For more information on the Messages Window visit Messages Window
- For more information on Docking windows visit Docking
- For more information on the Main Menu Bar visit Menus
- For more information on the Toolbar visit Toolbar

View Window

The View window provides 3D visualization and animation of musculoskeletal models and simulations in the OpenSim GUI.

To add multiple views of the same model:



1. Go to the OpenSim main menu bar and click Window Add View.

To learn more about the features of the 3D View Window please visit Navigating the 3D View Window

Navigator Window

The Navigator window provides information about a loaded model, such as bodies, actuators, joints, and associated motions, in a tree-style format.

To see the Navigator window:

1. Go to the OpenSim main menu bar and select **Window Navigator**.

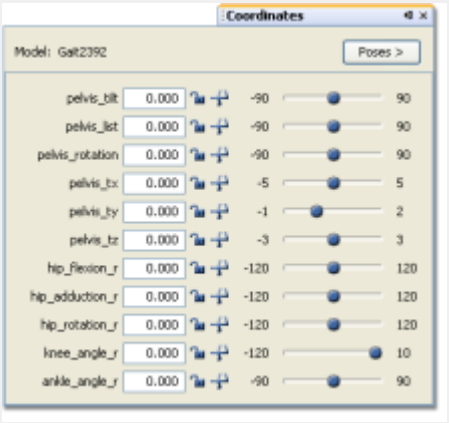
To expand or reduce branches in the Navigator window, click the **plus** or **minus** icon next to the branch heading.

To learn more about the Navigator Window please visit Navigator Window.

Coordinates Window

The Coordinates Window allows a user to interactively modify the joint coordinates in the model.

To see the Coordinates Window:



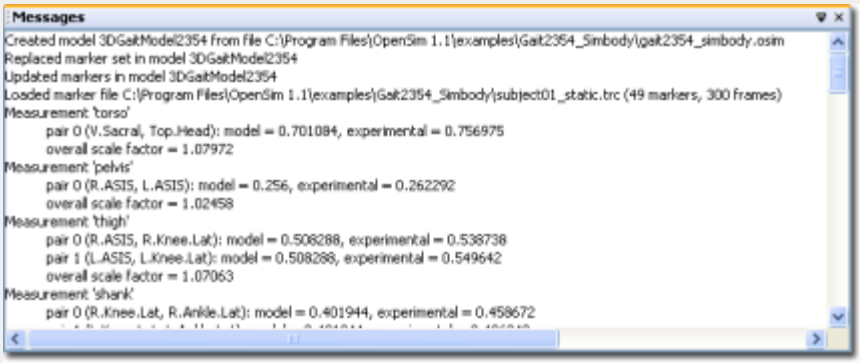
1. Select **Window Coordinates** from the OpenSim main menu bar.

> To learn more about the Coordinates Window please visit Coordinates Window.

Messages Window

The Messages window prints details of commands and analyses performed within OpenSim.

To see the Messages window:

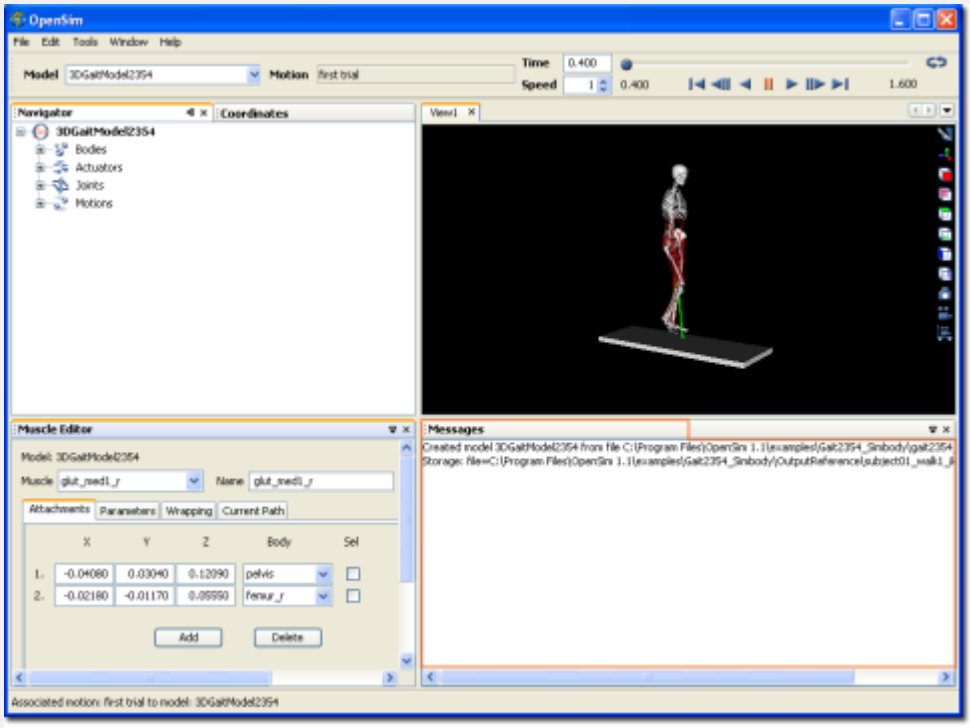


1. Click **Window Messages** from the OpenSim main menu bar.

Docking

The layout of the windows in the OpenSim GUI can be reconfigured by docking.

To dock a window:



1. **Click** the title bar of the window and **drag** it to the desired location.

An orange outline will appear, previewing where the window will be docked.

Next: Toolbar

Previous: Menus

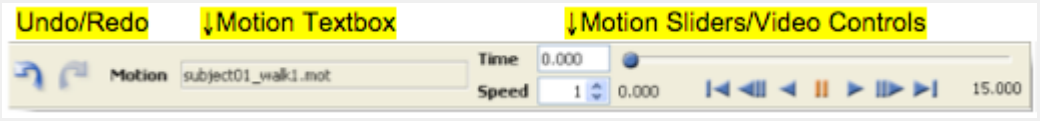
Home: Graphical User Interface

Toolbar

The information about the OpenSim toolbar covered in this section includes:

Parts of the OpenSim Toolbar

The toolbar is located at the top of the GUI and contains the model drop down menu, the motion textbox, the motion slider, and the video controls. There are also redo/undo buttons for enabling and disabling forces and constraints.



- For more information on the Undo/Redo Buttons visit Redo/Undo Buttons
- For more information on the Motion Textbox visit Motion Textbox
- For more information on the Motion Slider/Video Controls visit Motion Slider/Video Controls

Redo/Undo Buttons

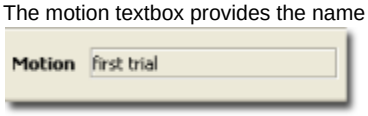
Redo/Undo buttons have been provided in the toolbar to make it easier to enable and disable constraints and forces. The history of enable and disable actions is recorded.

- You can move backwards through this history (reverse the actions) by clicking the **undo** button (the left-pointing arrow) multiple times.
- To move forward through the history and reinstate actions, click the **redo** button (the right-pointing arrow).

For instructions on enabling and disabling constraints and forces please visit Object-Specific Commands

Motion Textbox

The motion textbox provides the name of the motion currently associated with a model.



Motion Slider / Video Controls

The motion slider and video controls are used for the animation of the model:

- To animate the model use the video control buttons (e.g., play, pause, and loop).
- The motion slider corresponds to the current time point in the motion file, which describes how the model moves.

The value in the Speed textbox can be adjusted to change the playback speed of the animation

Next: Loading and Saving Models and Motions

Previous: Windows

Home: Graphical User Interface

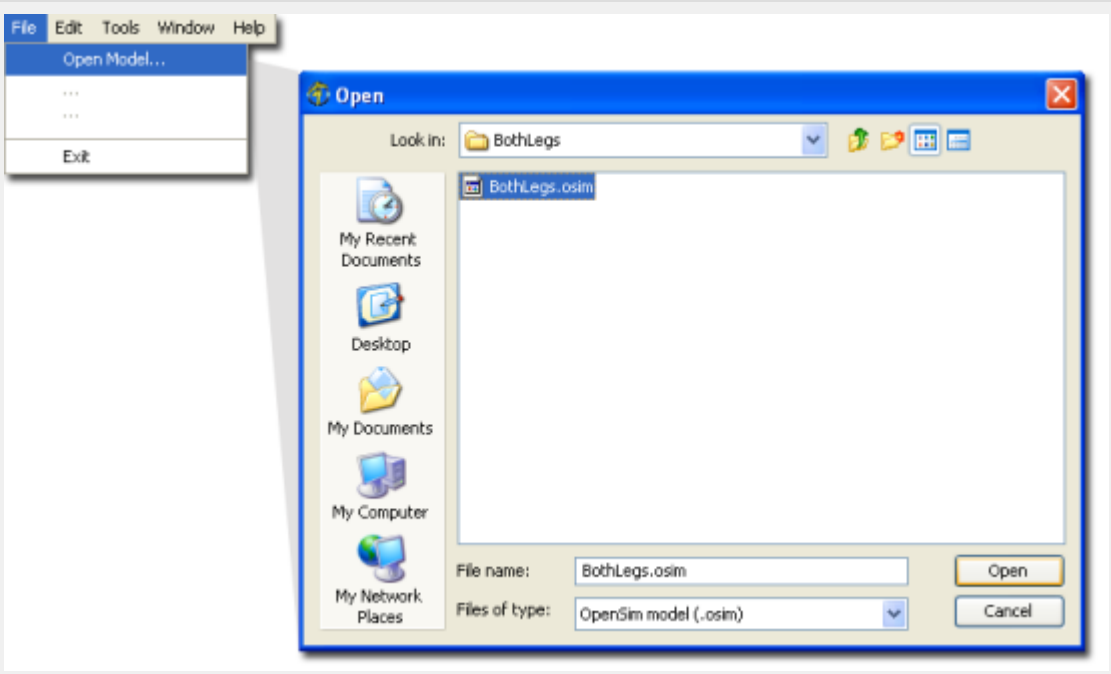
Loading and Saving Models and Motions

This section covers the basics of how to open and save an OpenSim model, how to load a motion into OpenSim, and how to import and export SIMM models within the OpenSim GUI:

Next: Opening a Model

Opening a Model

To open a musculoskeletal model written in the OpenSim format (file type: *.osim*)**:**



1. Click **File Open Model**… from the OpenSim main menu bar.
2. In the window that appears, locate and select the desired model, and then click **Open** as shown in the figure above.
3. After loading a model, its name will appear in the model drop down menu, located in the toolbar, and in the Navigator window.

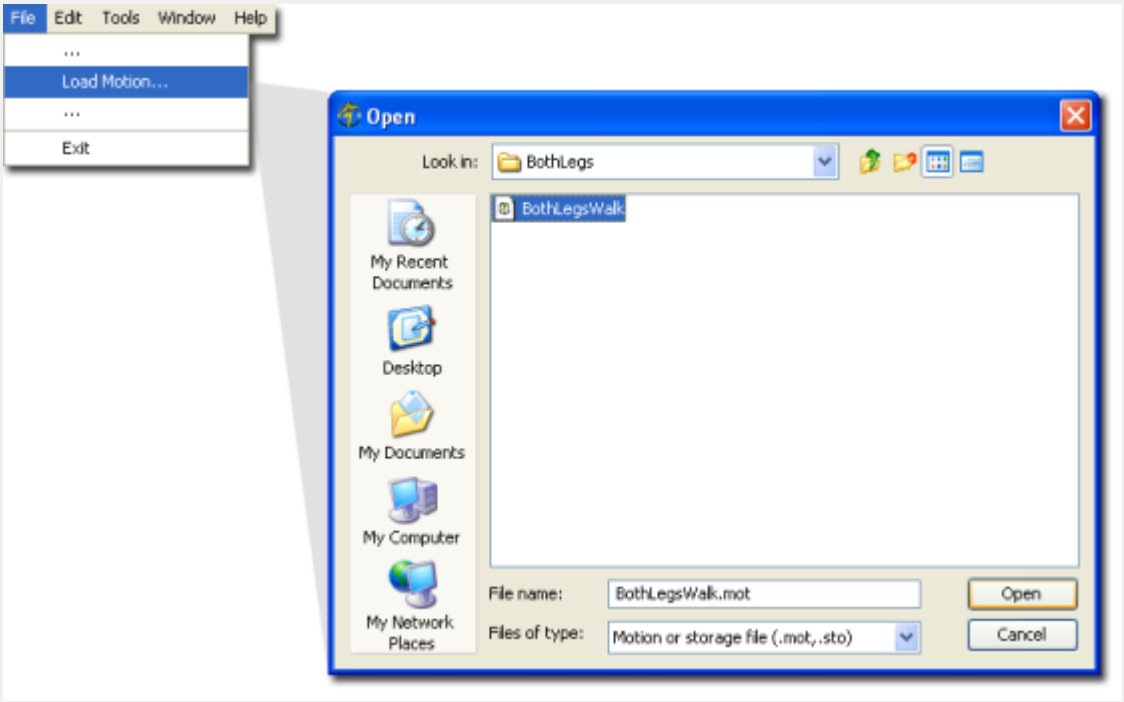
SimmKinematicsEngine models are also supported, provided their base body/segment is called "ground."

Next: Loading a Motion

Home: Loading and Saving Models and Motions

Loading a Motion

To animate a model, load an associated motion file (file type: **.mot)** into OpenSim**:**



1. Select **File Load Motion** … from the OpenSim main menu bar.
2. In the window that appears, locate and select the desired motion file, and then click **Open**.
3. After loading a motion, its name will appear in the motion textbox, located on the toolbar.

You can find the motions associated with a model in the Navigator window:

- A new branch titled the name of the file just loaded will appear in the Navigator titled Motions.
- Expand the Motions branch by clicking the plus icon next to it to see all motions loaded for a particular model.

Next: Appending Data to a Motion

Previous: Opening a Model

Home: Loading and Saving Models and Motions

Appending Data to a Motion

To append data from a motion file to a motion that is already loaded into OpenSim:

1. Expand the **Motions** branch in the Navigator.
2. **Right mouse** click on the desired motion and select **Append Data**.
3. OpenSim will open a file browser, which you can use to select the desired motion file.

To append data from this file to an existing motion, the data must satisfy two criteria:

1. The data must contain all of the time frames in the existing motion, and the time of each frame must match exactly (to within the precision used in the two motion files). If there are one or more time frames in the existing motion that are not in the file, OpenSim will print an error message and cancel the append command. If the file contains time frames not in the existing motion, they will be ignored.
2. The data cannot contain values for coordinates or markers that are in the existing motion. If it does, OpenSim will print an error message and cancel the append command. If the file contains other data columns (*e.g.*, ground-reaction forces) with names that are used in the existing motion, they will be appended, resulting in duplicate column names in the existing motion.

Once the data has been appended to the existing motion, it can be animated and plotted just like the original data. If you save the motion to a file, the original data and the appended data will both be written to the file.

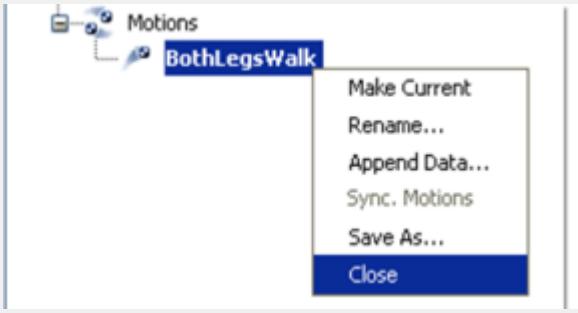
Next: Closing a Motion

Previous: Loading a Motion

Home: Loading and Saving Models and Motions

Closing a Motion

To close a motion:



1. **Expand** the Motions branch in the Navigator.
2. **Right mouse** click on the desired motion and select **Close**.

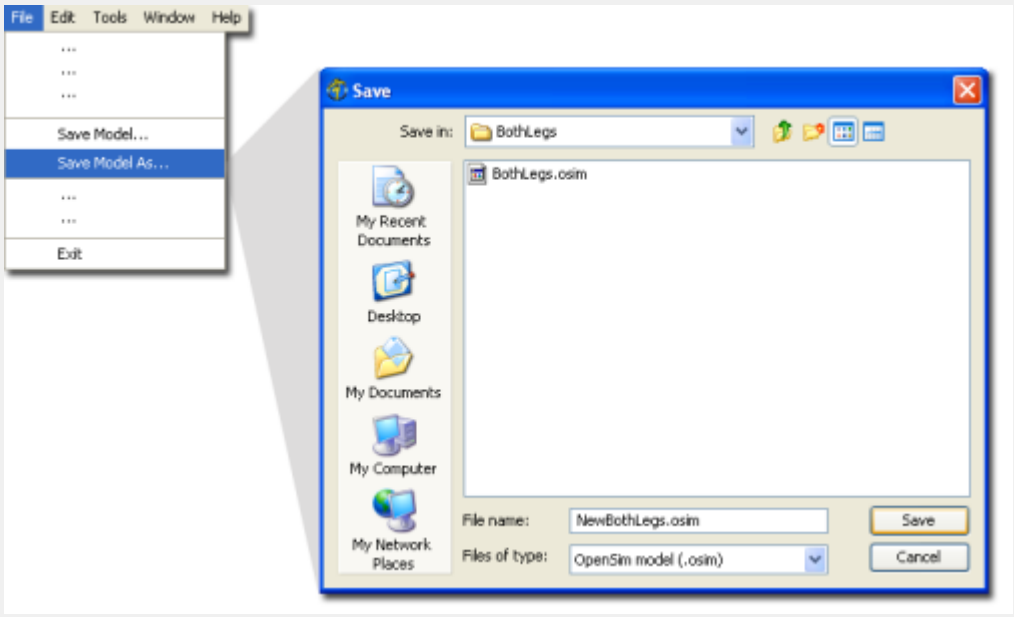
Next: Saving a Model

Previous: Appending Data to a Motion

Home: Loading and Saving Models and Motions

Saving a Model

To save a copy of a model:



1. Select **File Save Model As**… from the OpenSim main menu bar.
2. In the dialog box that appears, enter a name into the File name textbox, and then click **Save.**

⚠ Be sure to include the file extension **.osim** when entering the file name.

Next: Importing a SIMM Model

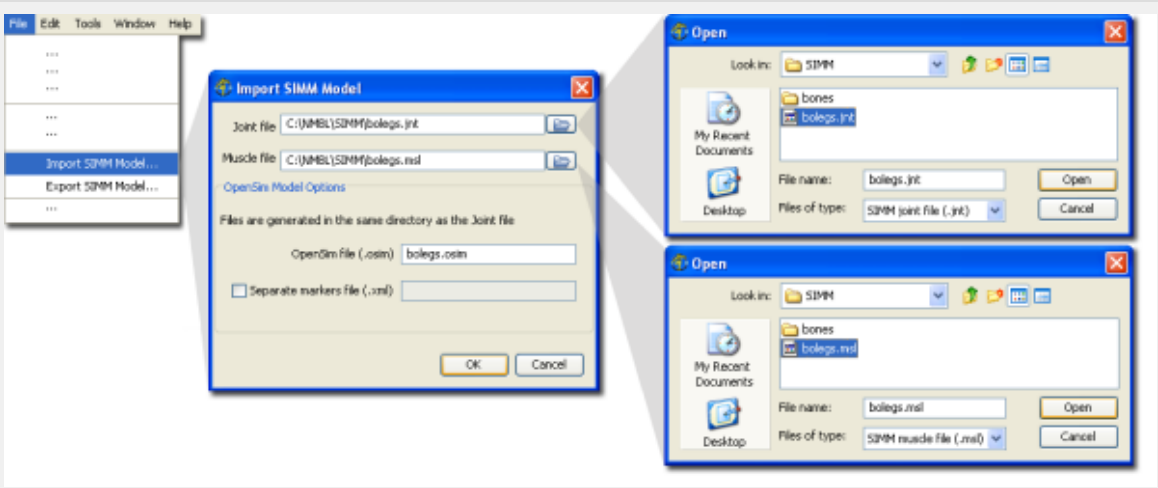
Previous: Closing a Motion

Home: Loading and Saving Models and Motions

Importing a SIMM Model

Models that were created using Musculographics' SIMM (Software for Interactive Musculoskeletal Modeling) toolkit can be imported into OpenSim.

To import a SIMM model:



1. Select **File Import SIMM Model**… from the OpenSim main menu bar.
2. The Import SIMM Model window will appear and ask for the following information:

    a. The SIMM model to be imported: specify this by clicking the folder icons to select the joint file (.jnt) and muscle file (.msl).

    b. The new OpenSim model name: enter the name in the OpenSim file (.osim) textbox.

    c. Whether or not a separate XML file for the model markers should be created when the model is imported: to create the XML file, select the check box next to Separate markers file (.xml) and enter a name into the textbox.

3. After specifying the information in the Import SIMM Model window, click **OK**.

⚠ Be sure to include the extensions when entering file names.

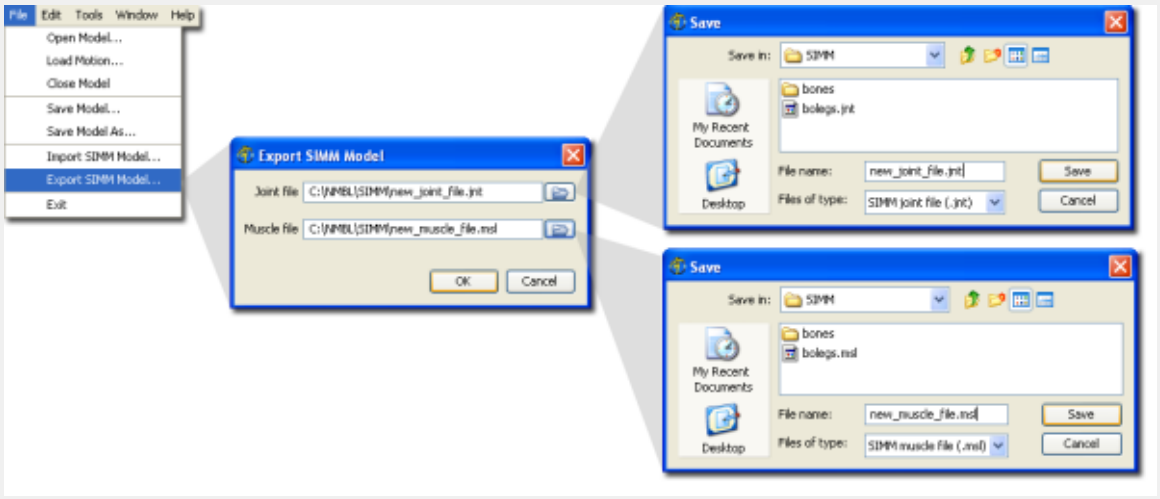
Next: Exporting a SIMM Model

Previous: Saving a Model

Home: Loading and Saving Models and Motions

Exporting a SIMM Model

To export an OpenSim model as a SIMM model:



1. Select **File Export SIMM Model**… from the OpenSim main menu bar.
2. From the Export SIMM Model window that appears, **click** the folder icons to select the location to save the joint file (.jnt) and the muscle file (.msl) for the SIMM model.
3. When you click the folder icon, a Save dialog box will appear. In this window, enter a name for the file in the File name textbox, and then click **Save**.
4. The Save dialog box will close. Then click **OK** in the Export SIMM Model window.

Next: 3D Views

Previous: Importing a SIMM Model

Home: Loading and Saving Models and Motions

3D Views

The 3D View windows allow you to visualize your models within OpenSim to inspect your models' topologies or their kinematics during simulations. Topics covered in this section include:

- Creating and Naming 3D Views
- Navigating the 3D View Window
- User Preferences

Next: Creating and Naming 3D Views

Creating and Naming 3D Views

You can have as many 3D Views as needed in order to examine your models from various angles. Specific uses of the 3D View windows include:

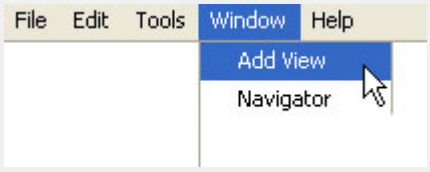
- Visualization of objects associated with models, for example, forces and moments applied to a model
- Visualization of motions that are pre-recorded or which result from an analysis
- A visual check of the validity of the model, for example, the correctness of muscle wrapping which requires visual inspection in different configurations

The topics on this page include:

Creating 3D Views

You do not have to explicitly create 3D Views. A new 3D View window opens automatically when the first model is loaded into OpenSim. Subsequent models are displayed in the same 3D View window. An offset is computed by OpenSim based on the dimensions of the loaded models to place the new model in the 3D View so that it does not overlap with previously loaded models. You can control this offset using the **Display Model Offset…** option, accessed from the node in the Navigator window that represents the model visit Object-Specific Commands for more information.

To explicitly open a new 3D View window (without loading a model):



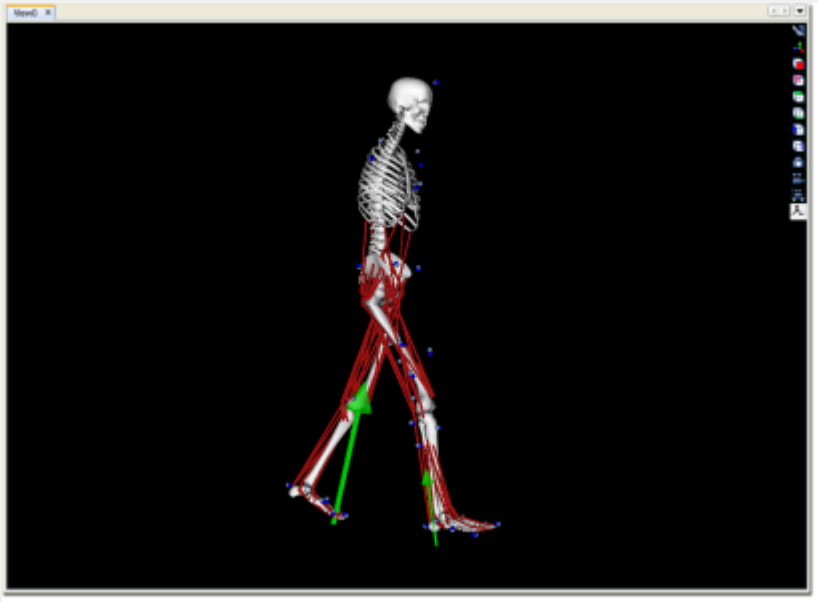
1. Click the **Window** menu from the main OpenSim menu bar and select the **Add View** option.

If you have a 3D View window already open, you can also create a new 3D View window:

1. Go to the existing 3D View window.
2. Right mouse click the tab label and select the **Add**.

If another 3D View window is already open, the newly created window will be a tabbed window overlaid on the previously open 3D View window. You can drag these windows apart as needed. If the new window was created from an existing 3D View window, the camera in the new window will be in the same position as in the original window. OpenSim will not create a new 3D View if you close the 3D View window that was automatically created when the first model was loaded.

The figure below shows a 3D view window with loaded model:

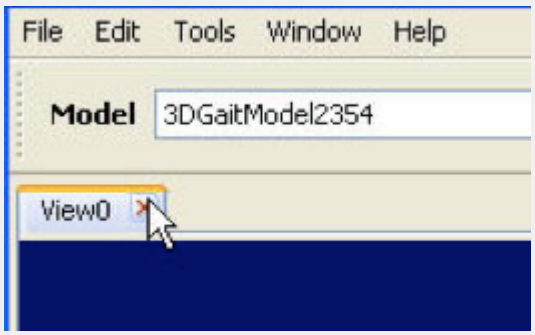


The two tabs in the upper-left indicate that there are two 3D View windows, one of which was created from the other.

Opening Multiple 3D Views

You can open many 3D View windows simultaneously to obtain multiple views. Multiple views can come in handy if a user wants to see a model from multiple angles at the same time. This can be useful, for example, when running a gait simulation or when moving a muscle attachment point in order to place it accurately on a bony landmark.

Closing 3D Views

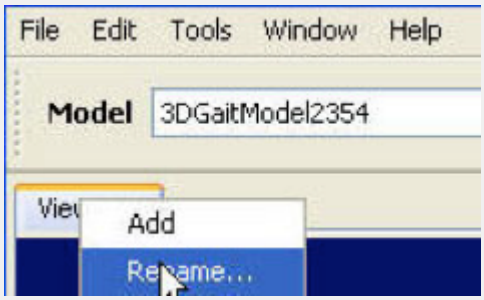


- 3D View windows can be closed by clicking the "**x**" next to the tab label on the 3D View window.

Naming 3D Views

OpenSim creates 3D View windows with default names that start with the prefix "view" followed by a number (e.g., view0, view1).

You can change the names to more meaningful names:



1. **Right mouse** click on the tab label of the 3D View window and selecting **"Rename…".**
2. This brings up the dialog box, populated with the current 3D View name (see figure below)

3. Enter the new name in this dialog box then click **OK**.

> Duplicate names are not allowed.

Next: Navigating the 3D View Window

Home: 3D Views

Navigating the 3D View Window

The topics in this section include:

Navigating the 3D View Window

Each OpenSim 3D View window is a VTK window. VTK is an open source visualization package distributed by Kitware Inc. (www.kitware.com). VTK windows come with a built-in set of key bindings that you can use to navigate within the window and to perform rotations, pan, and zoom operations. The key bindings differ, depending on whether you are running OpenSim on a Windows, Mac, or Linux platform.
On Windows, you can translate and rotate in the 3D View window using the following:

rotate	Rotate by clicking the **left mouse** and dragging.
pan	Pan by clicking the **center mouse** and dragging left or right, orby clicking the **left mouse** with **shift** and dragging left or right.
zoom	Zoom by clicking the **right mouse** and dragging up or down.
refit	Refit the models in the view window by typing **"r"**.
recenter	Center an object in the window by selecting the object (**ctrl**+**left mouse**) and typing **"r"**.

The 3D View Toolbar

The 3D View window comes equipped with a set of standard buttons to make it convenient to orient the objects in the 3D View according to your preferences. These buttons appear in a Toolbar, which is located by default along the right edge of the 3D View window but can be dragged to any of the four sides of the 3D View window. The following buttons are available in the Toolbar:

	Change background color. A color chooser is brought up when this button is clicked and the background of the current 3D View is set to the new color you select. This new color is saved in User Preferences so that future 3D View windows come up with this new background color.
	Look at the model from the world's –X direction. Note that multiple models could be loaded and each model could have its own concept of "front," but there's one unambiguous –X direction within the world's coordinate system. Similarly,  = +X,  = -Y,  = +Y,  = -Z,  = +Z.
	Toggle the display of world coordinate axes on and off.
	Take a snapshot of the view. When you click this button, a file browser is brought up and you are prompted for the name of a .tiff file in which to save the 3D View.
	Record a movie. Clicking this icon toggles movie recording on and off.
	Create or edit a Camera Dolly for custom camera positioning. Details are covered in Snapshots and Movies.
	Annotate objects in the 3-D view.

Hot Keys

The following set of hot keys are available and can be used to achieve some of the same functionality that is available through the ToolBar:

Button	Function
i	Zoom in
o	Zoom out
x	View along the X axis (of the world coordinate system)
y	View along the Y axis (of the world coordinate system)
z	View along the Z axis (of the world coordinate system)
r	Refit selected objects in the view window. All models are fit inside the view window if nothing is selected.

Selecting Objects in the 3D View

Selecting objects in the 3D View is a fundamental part of the graphical editing capabilities available in OpenSim. It makes it easier to perform tasks, such as moving muscle attachment points. The following operations are supported:

select	Select an object using **ctrl**+**left mouse**. An object that has been selected will be colored yellow, and its name will appear in the status bar. Selection is possible only on the current model.
add	Add another object to the current selection group using **shift+ctrl+left mouse**. Re-selecting an object toggles its selection.
clear	Clear all selections by using **ctrl**+**left mouse** in an empty area of the view.
drag	Drag objects by selecting them and then use the **left mouse** to click and drag the object to its new location. Currently only dragging of muscle points is supported.

Annotations

OpenSim allows you to annotate objects in the 3D View. This is accomplished by selecting the button  . This is a toggle button that reverses colors when activated. While activated, picking any object in the 3D View (e.g., bodies, markers or muscle points) will cause the object's name to be displayed in the 3D View in white text. Multiple objects can be selected using the **SHIFT** key. Deselecting the button will remove all annotations from the 3D View.

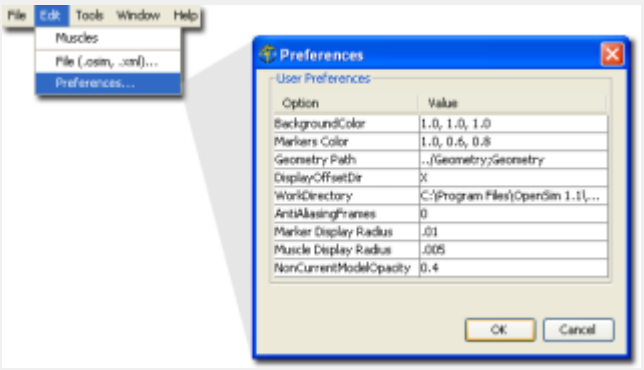
Next: User Preferences

Previous: Creating and Naming 3D Views

Home: 3D Views

User Preferences

You can alter display preferences for the 3D View window:



1. Selecting **Edit Preferences…** from the main OpenSim menu bar.
2. In the Preferences window that appears, **double click** on the value of the option you want to change and enter the new value. To save changes to the preferences, click **OK.**

Details about the available preferences are given in 3D Window Display Preferences, the table below. These preferences are saved and will be applied during the current session of OpenSim, as well as any future sessions of OpenSim.

Figure: Editing User Preferences

3D View Window Display Preferences

Option Name	Default Value	Description	When Applied
BackgroundColor	0.0, 0.0, 0.0	RGB (red, green, blue) values for the color to be used for the 3D View window background. Values range from 0.0 to 1.0.	When a new 3D View is created. Can be changed either here or using the Toolbar of the 3D View window.
Markers Color	1.0, 0.6, 0.8	RGB (red, green, blue) values of the color used for the virtual markers in the model. Values range from 0.0 to 1.0.	When a new model is created or loaded
Geometry Path	../Geometry; Geometry	A list of directories separated by a semi-colon that specify where OpenSim should look for geometry files (.vtp files). When a model is loaded, the directory containing the .osim file is searched for geometry .vtp files, as well as a directory with the name "Geometry" that lives underneath it. If a file is not found in these 2 locations, the directories specified in "Geomerty Path" are searched in order. The paths are specified using the Unix format and are defined relative to the OpenSim installation directory.	Instantly, as well as to subsequently loaded models
DisplayOffsetDir	X	Direction to offset display of models after the first one is loaded (X, Y or Z)	Applied when the next model is loaded
AntiAliasingFrames	0	An integer number between 0 and 4 indicating how many frames are overlaid to achieve the impression of anti-aliasing (blurring of edges to reduce jaggies on straight lines)	Instantly. Be aware that a value of 2 or higher slows the display significantly.

NonCurrentModelOpacity	0.4	A number between 0 and 1.0 indicating the opacity of models that are not designated the current model. A value of 1.0 means that non-current models are opaque, so current and non-current models look the same. A value of 0.0 makes non-current models completely transparent, so only the current model shows in the 3D View window.	Effective the first time the current model changes after setting this value
Debug	0	Debug level. Setting value greater than 0 (max =3) outputs verbose text to the Messages window, also when an exception is thrown a detailed stack-trace to help with troubleshooting is shown.	
Persist Models	On	Whether to keep track of open models, motions and camera views so that they are restored on opening the application.	Exit/Re-entry into the application.

Next: Navigator Window

Previous: Navigating the 3D View Window

Home: 3D Views

Navigator Window

The Navigator window shows you the set of models that have been loaded into OpenSim, along with their associated objects (such as motions), in a hierarchical, or tree, representation. Topics covered in this section include:

- Opening, Closing, and Using the Navigator Window
- Navigator Tree Nodes
- Node Commands (Context Menus)
- Object-Specific Commands

Next: Opening, Closing, and Using the Navigator Window

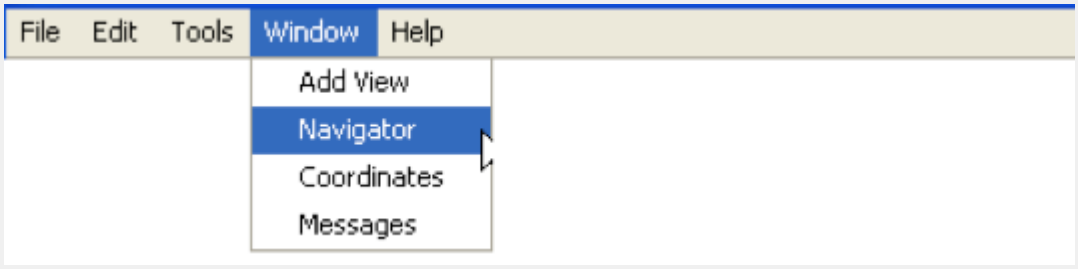
Opening, Closing, and Using the Navigator Window

The Navigator window shows you the set of models that have been loaded into OpenSim, along with their associated objects (such as motions), in a hierarchical, or tree, representation. This is particularly useful for large biomechanical models, which would be overwhelming if presented in a flat structure. For example, a simplified model used for gait analysis is made up of 92 muscles and 23 segments.

By presenting the currently loaded models in a hierarchical format, you are able to visualize how different objects are related to one another. The Navigation window also allows you to focus on the objects needed for the task at hand by collapsing tree nodes for objects that are currently irrelevant. The topics covered in this section include:

Opening and Closing the Navigator Window

When you launch OpenSim, the Navigator window appears but is blank since no models have been loaded yet. As models are loaded into OpenSim, they show up as sub-trees (folders) in the Navigator window. As with other OpenSim windows, you can decide to close the Navigator window to save space on the screen and reopen it again later. However, at most you can have only one Navigator window displayed at a time. Similar to other windows opened by the OpenSim application, the Navigator window can be dragged and docked in various places on the screen.



- To open: select **Window Navigator** from the main OpenSim menu bar.
- To close: click on the "**x**" on the tab of the Navigator window.

The Current Model

OpenSim allows you to load in more than one model at a time. This feature is distinctive to OpenSim and is very useful for comparing models and investigating the effects of various model changes. However, OpenSim functions can only be applied to one model at a time.

OpenSim designates the one model that is being worked on and to which menu commands apply (e.g., Save, Close) as "current." The current model is indicated in the Navigator window in **bold**. When a model is made current, all menu commands operate on it. To make a model current:

screen shot of right clicking option

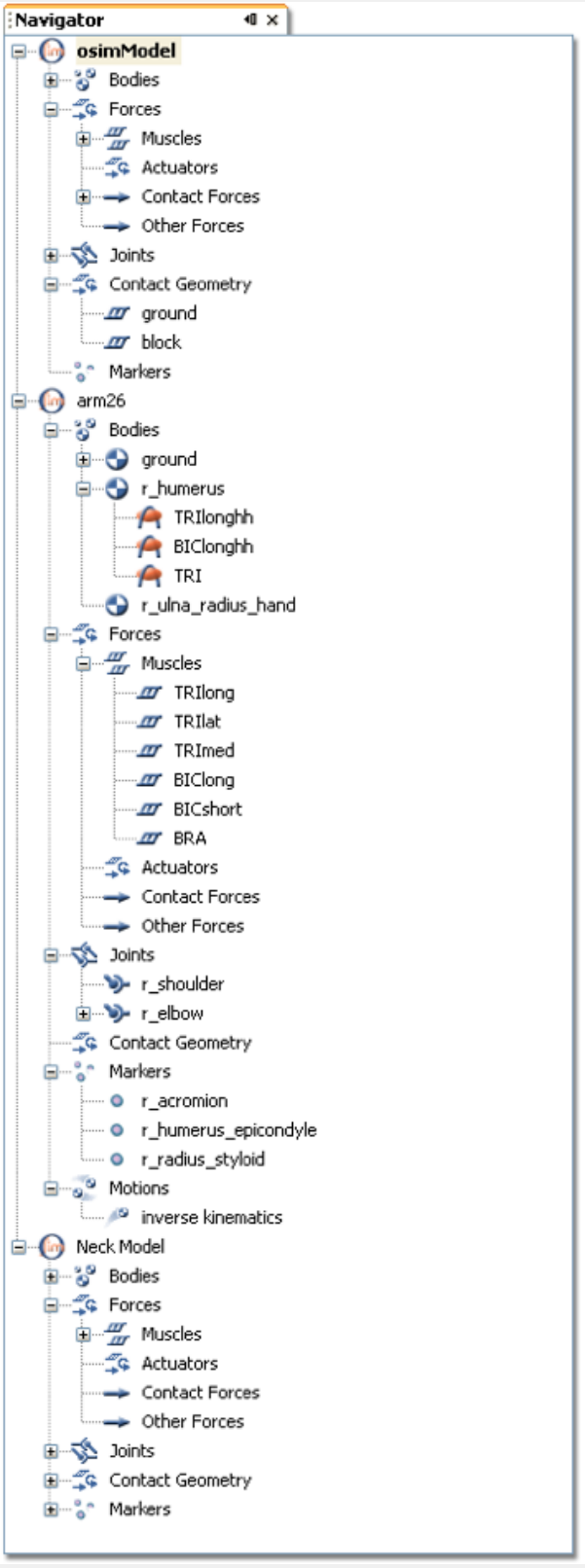
There are several ways you can make a model current:

1. **Double click** the name of the model you wish to make current in the navigator window.
2. **Right click** the name of the model you wish to make current and select **Make Current.**

When a model is current, the colors in the view window are vivid and the name in the navigator window is bold. Non-current models appear transparent in color in the view window and the names are not bold in the navigator window.

The Current Motion

You can also load multiple motions into OpenSim. If a single motion is made current, then it is reflected in the motion textbox in the toolbar at the top of the OpenSim GUI. The current motion is also marked in **bold** in the Navigator window. If more than one motion is marked as current (for example when synchronizing multiple motions), then all the nodes corresponding to current motions are displayed in **bold**.



- Example Navigator View with Two Models (Sample Model and Gait Model).
- Note: the current model is marked in bold, which is Gait Model in this case.

Next: Navigator Tree Nodes

Home: Navigator Window

Navigator Tree Nodes

The tree displayed in the Navigator window has nodes of many different types. Nodes can represent 1) a single object in the model or associated with the model or 2) collections of objects (for example, the node corresponding to the set of rigid bodies in a model). Nodes that correspond to multiple objects have a folder-like representation in the navigator tree.

The following table describes the types of objects that can be represented in the tree and their corresponding icons. Some objects, such as markers, are not represented in the tree in order to present a less cluttered view when dealing with large models.

Node Label	Icon	What the node represents
[Model Name]		A full OpenSim model
• Bodies		The set of rigid bodies in the model
• [Body Name]		A single rigid body
• [Wrap Object Name]		A single muscle wrap object
• Display Geometry		The set of geometry files for a body
• [Geometry Name]		A geometry (.vtp) Ayman, does this have to be a .vtp file?file with editable display properties
Forces		All forces in the model
• Muscles		All Muscles
• [Muscle Group Name]		A muscle group
• [Muscle Name]		A single muscle
• Actuators		All non-muscle actuators
• TorqueActuator		A torque
• CoordinateActuator		A Generalized force
• Contact Forces		A force
• Other Forces		A force
Joints		All joints in the model

• [Joint Name]		A single joint
• Dofs		All degrees of freedom for a joint
• [Dof Name]		A function describing dof changes
Constraints		A list of constraints associated with the model
• [Constraint Name]		A specific constraint
Contact Geometry		A list of contact geometry associated with the model
Markers		A list of markers associated with the model
• [Marker Name]		A specific marker
Motions		All motions associated with a model
• [Motion Name]		A specific motion

Key Bindings

Selection of nodes in the Navigator tree follows the standard conventions:

- Select objects by clicking the **left mouse** button once on the tree node
- Select a region of contiguous nodes in the tree using **shift+left mouse**
- Clicking the **left mouse** button on a node that has already been selected causes it to be deselected
- Add objects to a group of selected objects by using **ctrl+left mouse**

In addition, clicking the **left mouse button twice** (double clicking) a node toggles the display of its corresponding object. This is useful when quickly navigating the model to investigate where different objects are in the 3D View window.

Next: Node Commands (Context Menus)

Previous: Opening, Closing, and Using the Navigator Window

Home: Navigator Window

Node Commands (Context Menus)

This section describes the commands that are accessible through the different tree nodes. You can select the commands by right-mouse-clicking on a tree node to bring up the associated context menu.

Since multiple nodes, potentially representing objects of different types, can be selected, OpenSim decides what commands make sense for the combination of selected nodes and only displays those. For example, the "**Edit..**" command only makes sense for individual objects, and hence, is not shown when multiple objects are selected in the tree.

The sections below describe commands common to most objects, while the section about Object-Specific Commands details object-specific commands:

- Display Menu
- Edit
- Property Viewer

Display Menu

The Display menu includes a set of commands that controls whether objects corresponding to selected tree nodes are shown in the 3D View window, and if shown what representation to use for them. Objects presented in the Navigator tree may not have a corresponding visual representation in the 3D View window, in which case, the nodes for these objects would not have a Display Menu.

If the user selects a combination of nodes that contains some objects that do not have a visual representation, the Display Menu is not available. It is possible, however, to pick multiple objects that have visual representations in the 3D View window and change their visual properties together using the commands in Display Menu.

The following commands in the Display Menu control what objects appear in the 3D View window:

- **Show**: This option is dimmed out if the object is shown already in the 3D View window; otherwise executing the command shows the representation of the object(s) in the 3D View window.
- **Show Only**: Same as "**Show**" except that it hides all other objects of the same type from the 3D View window. Multiple objects can be selected to show only a few bodies or a collection of muscle groups or individual muscles.
- **Hide**: This option is dimmed out if the object is already hidden in the 3D View window; otherwise it hides the representation of the selected object(s) from the 3D View window.

The remaining commands in the Display Menu control the visual display of the object(s) in the 3D View window:

- **Flat-Shaded:** This option changes the visual representation of the selected objects to flat shaded representation. Rendered geometry is polyhedral in general. In the flat shaded representation, a color is used for each face of the polyhedron based on normals at all the vertices of the face. Edges would still show since neighboring faces may not have the same normal at the edges.
- **Smooth-Shaded:** This option changes the visual representation of the selected objects to smooth shaded representation. Rendered geometry is polyhedral in general. Smooth shading averages the normals at the polyhedral edges and vertices of different faces, so the shading appears smooth and continuous.
- **Wireframe:** This option changes the visual representation of the selected objects to a wireframe representation. Rendered geometry is polyhedral in general. The wireframe mode only displays the edges of the polyhedron.
- **Color…:** This option brings up a color chooser window. Selected colors are applied after the color chooser window is closed.
- **Opacity……:** This option brings up a window with a slider displaying the current value of Opacity for the selected object. You can change this value to control how transparent or opaque an object is. The changes are applied instantly to the selected objects and can be cancelled by pressing the **Cancel** button in the window.

Some navigator tree nodes have additional commands that appear under their Display Menu. These are unique to the specific object and are explained in Object-Specific Commands.

Edit

For nodes corresponding to objects that have an editor associated with their types, for example, Muscle Editor to edit muscles, the command **Edit…** is added to the context menu. This option brings up a window for advanced editing of the selected object.

The specialized editors that are available are listed below:

- Muscle Editor
- Function Editor
- Excitation Editor

Editing is only supported for single objects. The editing option will not appear when multiple objects are selected.

Property Viewer

This option brings up a window similar to the File Editor described in the File Editor section. However, it is not possible to change values in this window.

Next: Object-Specific Commands

Previous: Navigator Tree Nodes

Home: Navigator Window

Object-Specific Commands

The following Object-Specific Commands are available:

Model Node

When a model is loaded into OpenSim, OpenSim creates a tree representing the objects comprising this model and adds it to the existing tree displayed in the Navigator window. The node for the model will show the model name (as specified inside the .osim file). If the model is marked as "current," the model name is displayed in **bold**.

The following options are available from the context menu associated with the node for the model. You can access the menu by clicking the right mouse button on the node.

- **Make Current**: Picking this option makes the model corresponding to the selected node the current model within OpenSim. All functions, such as save and close, are applied to the current model. With the default settings, the current model becomes opaque, while any other models are dimmed out. Also, the node corresponding to the current model is displayed in bold in the Navigator window.
  This option is disabled if the model is already marked as current or if multiple model nodes have been selected. This option for changing the current model is equivalent to picking an entry from the model drop down menu in the OpenSim Toolbar.

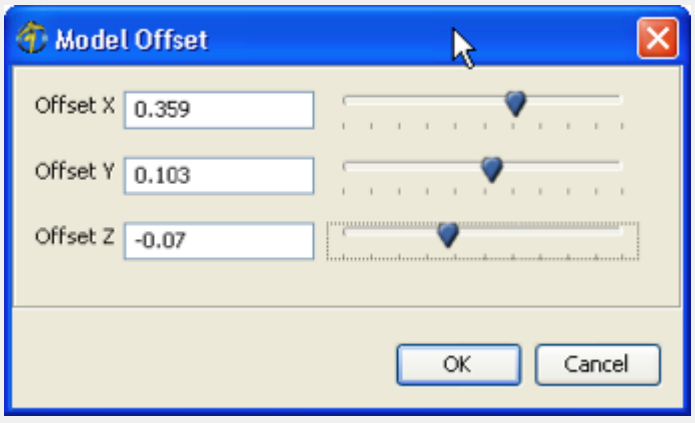
- **Rename…:** Picking this option brings up a window populated with the current name of the model. You type in the new name, and then click **OK**.

Names are not checked for duplication, so you can load multiple instances of the same model into OpenSim. The only requirements for names are that they begin with an alphanumeric character and that they do not include spaces.

> Note that the name change is not permanent. The model must be saved with the new name (using **File -> Save Model…** or **File -> Save Model As…)**, or else the name change is lost.

- **Display**: This option brings up a cascade menu containing options to control the display of the model. The menu contains a subset of the commands described in the Node Commands (Context Menus) section as well as a **Model Offset…** option that allows you to move where the model is displayed on the screen, relative to the world coordinate system.

- **Model Offset:**



To set the model offset value, select the Display -> Model Offset… option from the main OpenSim menu bar. This brings up the dialog box shown below. The default display offset is chosen so that the new model does not overlap with existing models, and is based on the size of a bounding box containing all the models already loaded in OpenSim. You can change this offset, for example, to overlay models on top of each other or to put them far apart, by entering new values in the Model Offset dialog box.

> This display offset is a display only property. The model itself is never affected by the changes to the display offset.

- **Info**: This option brings up a dialog box that displays information about the model. The values are extracted from the .osim file itself. In

particular, the dialog box shows:

- *Model Name*: The name of the model as specified in the model's .osim file.
- *Model File*: The full path to the model's .osim file (This is useful if you load multiple models that have the same display name into OpenSim).
- *Authors*: The list of authors who created the model. Users making significant changes to the model should add their name to this list to get credit.
- *References*: The list of publications that describe the creation and/or modification of the model. These references should include model limitations and assumptions made to generate and validate the model.

- **Close**: Picking this option causes the selected model to be closed and unloaded from OpenSim. This is identical to making the model current and pickingFile -> Closefrom the OpenSim main menu bar, following all the same steps (e.g., you will be prompted to save the model and any associated poses before closing).

Motion Node

When a motion is loaded into OpenSim, it is added to the existing tree in the Navigator window under the **Motions** node. The node for the motion will show the motion name. If the motion is marked as "current," its name is displayed in **bold**.

The following options are available from the drop down (context) menu associated with the node for the motion. You can access the menu by clicking the right mouse button on the node.

Make Current

Picking this option makes the motion corresponding to the selected node the current motion within OpenSim. The name of the current motion appears in the motion textbox of the OpenSim toolbar. All video control button functions, such as play and advance, are applied to the current motion. Also, the node corresponding to the current motion is displayed in bold in the Navigator window. More information about current models is covered in Opening, Closing, and Using the Navigator Window.

You can make more than one motion current using the **Sync. Motions** command below.

Rename

Picking this option brings up a dialog box populated with the current name of the motion. You type in the new name, and then click **OK**.

Names are not checked for duplication, so you can load multiple instances of the same motion into OpenSim. The only requirements for names are that they begin with an alphanumeric character and that they do not include spaces.

Note that the name change is not permanent. The "Save As…" option from the motion context menu.model must be saved with the new name or else the name change is lost.

Sync. Motions

This option is enabled only if you select multiple motions (at most one per model), and it has the effect of displaying all motions concurrently. The display is controlled using the common motion slider embedded in OpenSim's toolbar. As the slider moves, OpenSim puts all models in tandem in their proper time of their respective motions. If a motion does not have a frame of data corresponding to the toolbar's time, a frame is created by interpolating the data. This functionality is useful when comparing different motions.

Displaying Coordinate Axes for Body and Joint Nodes

The display of the coordinate axes for each body is controlled by clicking the right mouse button on the associated node for that body and selecting the **Show Axes** option. The **Show Axes** option is a toggle. A check next to it indicates that the coordinate axes are already being displayed. In this case, clicking on the option will turn off the coordinate axes display. If the axes are not being displayed and the **Show Axes** option is clicked, then the axes will appear.

For joint nodes, there are two similar options: **Toggle Child Frame** and **Toggle Parent Frame**. As with the **Show Axes** option for a body node, the **Toggle Child Frame** and the **Toggle Parent Frame** options are toggle buttons, as their names indicate, accessible via the right mouse button. You click the option to turn it on and click it again to turn it off. The two frames (child and parent) are associated with the two bodies of the joint.

The coordinate axes for both body and joint nodes appear as a set of three orthogonal lines, colored yellow, green, and red.

Enabling and Disabling Muscle and Constraint Nodes

All muscle and constraint nodes can be enabled or disabled via their **Enable** option. This is a toggle button, accessed by clicking the right mouse button on the node.

If the muscle or constraint is enabled, a check appears next to the **Enable** option. To disable the muscle or constraint, just click on **Enable**. To

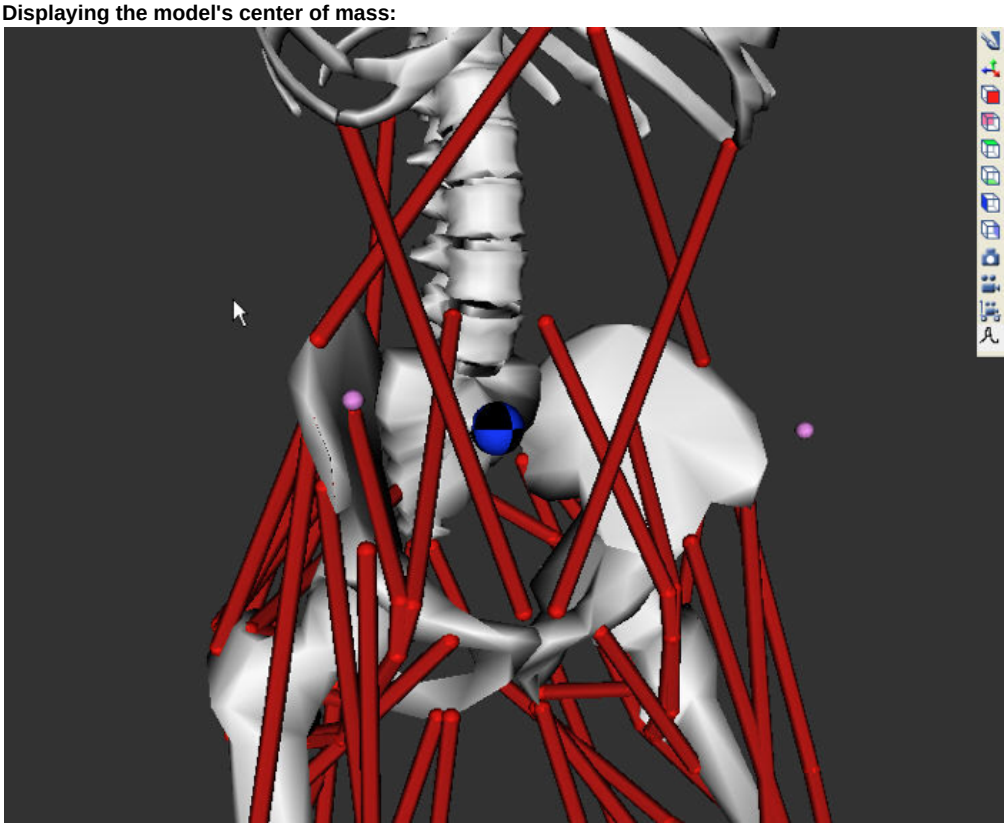
turn the muscle or constraint back on, click on **Enable** again.

You can also enable and disable muscles and constraints via the Redo and Undo buttons in the Toolbar.

Displaying the Center of Mass of Individual Bodies and the Model

Nodes corresponding to individual bodies have another option to toggle the display of the center of mass. Turning this toggle menu choice on will cause the Center-of-mass of the corresponding body to be displayed as a solid green sphere Similar option is available for the node corresponding to the Body-Set, in this case the System's Center of mass is displayed as a blue checkered ball as shown below.

Displaying the model's center of mass:



Next: Coordinates Window

Previous: Node Commands (Context Menus)

Home: Navigator Window

Coordinates Window

The Coordinates window displays all of the joint coordinates (degrees of freedom) in a model, and provides an interface for changing their values. It displays only the coordinates in the current model, which can be set using the Navigator window please visit Opening, Closing, and Using the Navigator Window for more information. This chapter covers:

- Coordinate Controls and Poses

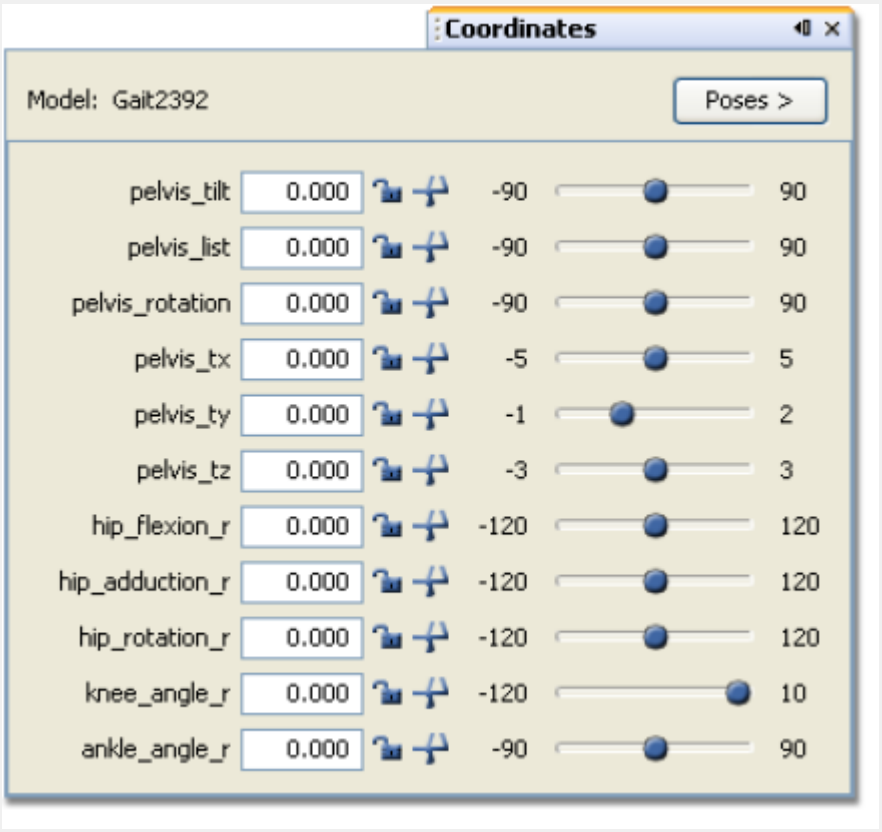
Next: Coordinate Controls and Poses

Coordinate Controls and Poses

The topics included in this section are:

Opening and Closing the Coordinates Window

When you launch OpenSim, the Coordinates window is opened in the same panel as the Navigator window. As with other windows in OpenSim, you can move the Coordinates window to other locations in the application window by dragging the title bar and docking it in another panel of windows.



The Coordinates window allows you to set the value of each coordinate in the current model, as well as to lock and clamp it.

- **To close: click the 'x' in the top right of the tab.**
- **To open:**
  select Window Coordinates from the OpenSim main menu bar.

Coordinate Control

For each coordinate in the model, there is one line in the Coordinates window, specifying the name of the coordinate and the following items that control its value:

value: 30.000

This number field displays the current value. If you type in a new value that is outside the range of motion of a coordinate that is clamped, the current value will not be changed. If the coordinate is locked, the number field will be grayed out, but will still display the current value.

clamp: clamped = unclamped =

This toggle lets you clamp and unclamp the coordinate. When a coordinate is clamped, it cannot be set to a value outside its range of motion, either by typing in a new value, moving the slider, playing back a motion, or running a dynamic simulation. The minimum and maximum values of the range of motion are shown on either side of the coordinate's slider.

lock: locked = unlocked =

This toggle lets you lock and unlock the coordinate. When a coordinate is locked, its value cannot be changed by typing in a new value, moving the slider, playing back a motion, or running a dynamic simulation.

slider: 

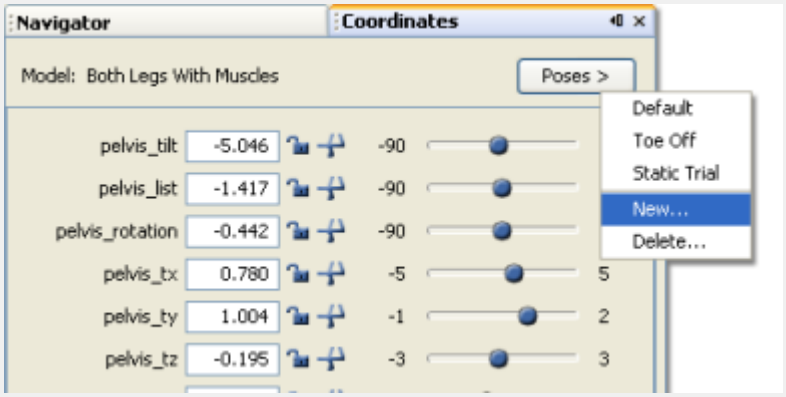
The slider bar allows you to move a coordinate continuously through some or all of its range of motion. Click on the dot and drag it to a new position to change the coordinate smoothly between the minimum and maximum values.

Poses

A pose is a set of values for all of the coordinates in a model. Saving and applying poses is an efficient way of putting the model in different configurations, or poses, rather than entering coordinate values individually.

Create a New Pose

To create a new pose:



1. Set all the coordinate values to put the model in the desired configuration by entering values, moving the sliders, or applying a particular frame of motion.
2. Click **Poses >** and choose **New...** from the drop down menu
3. Enter a name in the dialog box that appears and click **OK**.

Note that a pose does not include a camera position or the locked/unlocked or clamped/unclamped states of the coordinates.

Applying a Pose

To apply a pose, click on the **Poses >** button and choose the desired pose from the list in the menu. "Default" is always the first pose in the list, followed by the user-defined poses 2). "Default" is the pose containing the default coordinate values stored in the model file.
When you apply a pose to a model, each coordinate is set to its value in the pose only if the coordinate is currently unlocked, and, if clamped, only if the value is within the coordinate's range of motion.

Deleting a Pose

To delete a pose:

1. Click on the **Poses >** button and choose **Delete...** from the drop down menu.
2. Select the pose you which to delete by clicking on its **name**.
3. When you click **OK**, the poses will be deleted from the drop down menu.

use ctrl+left mouse or shift+left mouse to select multiple poses

Next: Snapshots and Movies

Home: Coordinates Window

Snapshots and Movies

Exporting models as images and animated movies easily is an important part of a modeling and simulation environment. OpenSim provides this capability, enabling the creation of snapshot images and custom movies with a click of a button. This chapter covers:

- Taking Snapshots and Making Movies

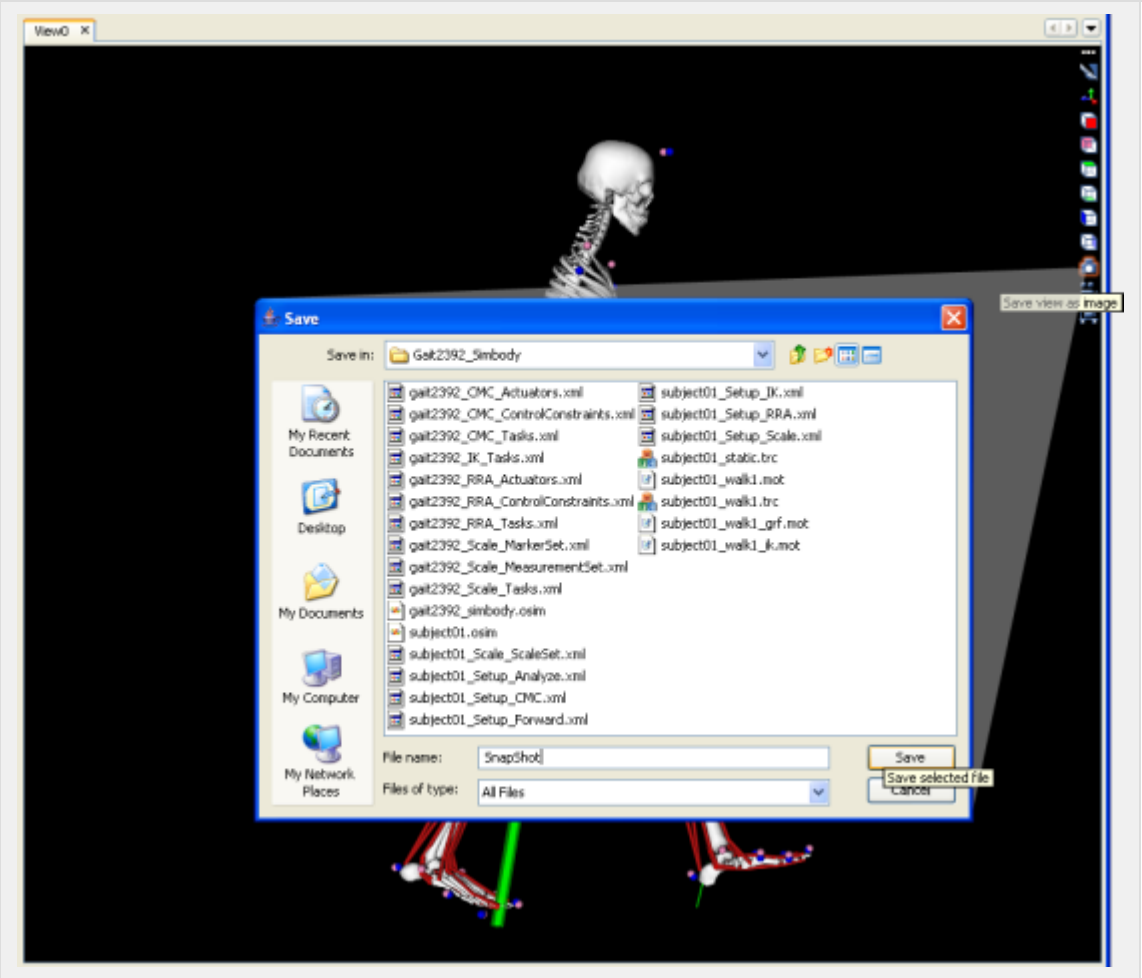
Next: Taking Snapshots and Making Movies

Taking Snapshots and Making Movies

The topics covered in this section are:

Taking a Snapshot

To save the current view as an image:



1. Click the **camera button**  in the current 3D View Toolbar.
2. Enter a file name for the image in the dialog box that appears and click **Save**.

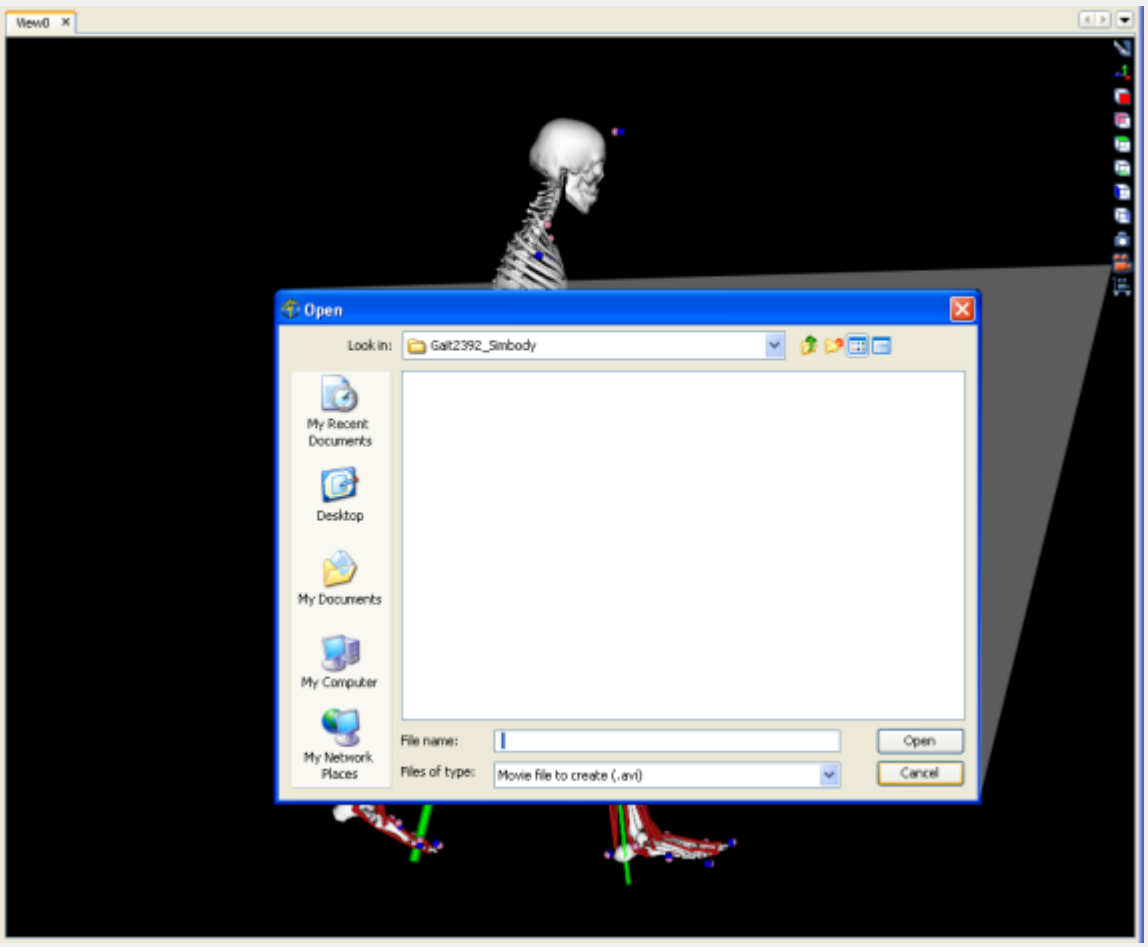
The current view will be saved as a *.tiff* image.

Making a Movie

OpenSim allows you to save the simulation playback in an AVI format movie file using the movie camera button see Recording a Movie for more information. You can change the camera angle or its position with respect to the model during the recording session manually or if more precision is required, use a pre-defined camera path see Use a Pre-Defined Camera for more information. Posing the model with the coordinate viewer can also be recorded.

Recording a Movie

To record a motion as a movie:



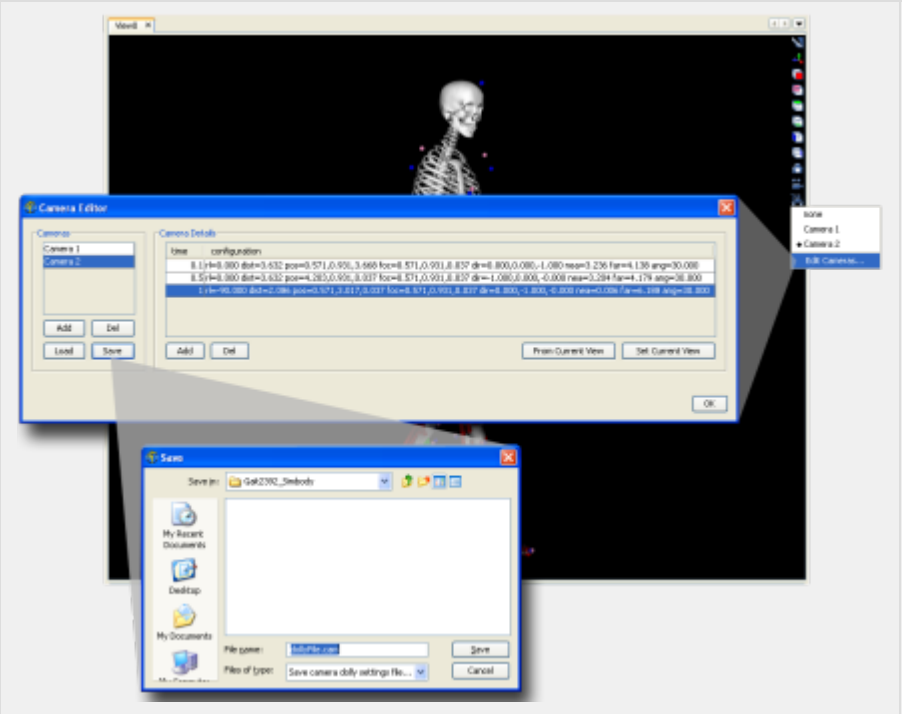
1. Click the **movie camera** button in the 3D View Toolbar.
2. Enter a file name for the movie to be recorded and click the **Open** button which will cause recording to be toggled "on."
3. What is displayed in the 3D View window will then be recorded. So any changes you make to the camera view or to the model, for example, changing coordinates or playing back a simulation through the play or reverse play buttons in the video controls, will be saved as a movie to the specified file. The movie camera button will be highlighted while recording.
4. To stop recording and write out the movie file, press the **movie camera** button again.

Pre-defined Camera Positioning

Changing the user's (camera's) perspective during playback of a motion can be very difficult to do precisely and repeatedly. To assist you in preparing movies with a variety of camera angles, OpenSim provides a camera dolly and editing tools to construct the desired path of the camera during a recording session.

Create/Edit a Pre-defined Camera

To create or edit a camera path:



1. Click on the **camera dolly** button in the 3D View window.
2. Select the **Edit Cameras…** option from the drop down menu that appears.

The **Camera Editor** window will appear with a list of available cameras in the *Cameras* pane on the left. Press the **Add** button underneath the *Cameras* pane to add a new camera. Remove a camera from the list by clicking on the camera name and then pressing the **Del** button underneath the *Cameras* pane.

To view and/or modify a camera's pre-defined configuration(s), select the camera in the *Cameras* pane with the **left mouse** button. Its configurations at various instances in time appear as a table in the *Camera Details* pane. The following list describes how to work with the *Camera Details* table:

add configuration to table	To add a new configuration to the *Camera Details* pane, press the **Add** button. The current view and time, according to the time in the motion player, will be added to the table. If you want a different view to be saved, change the camera view (visit Navigating the 3D View Window on interacting with the 3D View window) before pressing the **Add** button.
edit time field	When a configuration is added to the *Camera Details* pane, the time that is saved to the table is the time in the motion player. You can manually edit the time field by double-clicking on it and entering a new value.
update configuration	You can update a configuration by selecting the desired row in the table using the **left mouse** button. Change the camera view in the 3D View window to the desired perspective and then press the **From Current View** button. Note that only the configuration of the camera (view) is changed; the time remains unchanged.
view configuration	A camera configuration can be previewed in the 3D View window by selecting the desired configuration using the **left mouse** button and pressing the **Set Current View** button. Note that only the configuration of the camera (view) is changed; the motion is not advanced or rewound.
delete configuration from list	To delete a configuration from the list, select the desired row using the **left mouse** button. Then, press the **Del** button.

Specify a list of times and corresponding configurations. During motion playback, the camera path will be generated by interpolating the position and orientation of the camera using the specified configurations and times. Therefore, to produce a smooth path, you need to consider the time intervals and configuration changes between entries in the list. For instance, when the camera is expected to undergo a large change in its orientation (e.g., 90^{o}), you should specify several transitional configurations with short time intervals between them to ensure a smooth path, rather than just providing the first and last configurations.

Use a Pre-Defined Camera

To use a pre-defined camera path, clicking on the camera dolly button  in the 3D View window. A drop down menu appears, listing the available preset cameras, if any. "None" defaults to the current view. Select the desired preset camera. Then, press the play  or reverse play  button in the video controls to see the simulation as viewed with the pre-defined camera.

Next: Muscle Editor

Home: Snapshots and Movies

Muscle Editor

The Muscle Editor gives you access to all of the parameters of the muscles and other actuators in the model. The paths of the muscles can be altered by selecting and moving attachment points in the 3D View window, and the force-generating parameters can be viewed and modified in a set of tabbed panels. The topics covered in this chapter include:

Next: Selecting Models and Muscles

Selecting Models and Muscles

The topics covered in this section include:

Needed: graphic that defines the different parts: muscles/muscle points

Muscle Paths and Muscle Points

The path of a muscle is defined by a series of attachment points. In the simplest case, each attachment point is fixed to a body, and the path of the muscle is the set of straight lines connecting each pair of adjacent points. These attachment points are called *fixed points*.

There are three other types of muscle points that can be used to define a muscle path. *Via points* are attachment points that are fixed to a body, but they are used in the muscle path only when a specified coordinate is in a certain range. These points can be used to implement simple cases of wrapping, such as the quadriceps wrapping over the distal femur when the knee flexes beyond a certain angle.

Another type of attachment point is called a *moving muscle point*. These are points whose X, Y, and/or Z offsets in a body's reference frame are functions of coordinates, rather than simple constants. This type of point is useful when you want the muscle path to move as a joint flexes, but wrap objects are not suitable for implementing the proper motion.

The last type of attachment point is a *wrap point*. Wrap points are attachment points whose XYZ offsets are calculated automatically by OpenSim in order to wrap a muscle over the surface of a wrap object. Wrap objects are geometric shapes (spheres, ellipsoids, cylinders, and torii) that you can use to constrain the paths of the muscles. When the straight-line path of a muscle intersects a wrap object, an algorithm calculates a new path between the two points that wraps smoothly over the object. To define the new path, two wrap points are introduced: one at the tangent point where the path initiates contact with the object, and one at the tangent point where the path breaks contact. The muscle path between these two wrap points is a curved path that follows the surface of the object.

Selecting a Model to Edit

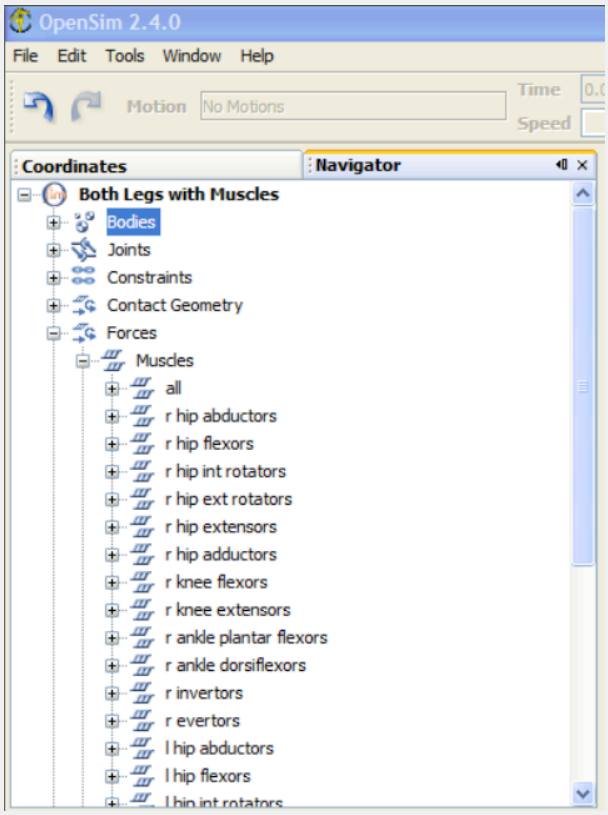
Like all of the model editing tools, the Muscle Editor operates only on muscles in the current model. When you change the current model, the Muscle Editor automatically switches to operate on this model, and displays the first muscle in the model.

Selecting a Muscle

To select a muscle for editing, you must first make sure that the model the muscle is in is the current model visit Opening, Closing, and Using the Navigator Window for more information on how to make a model current.

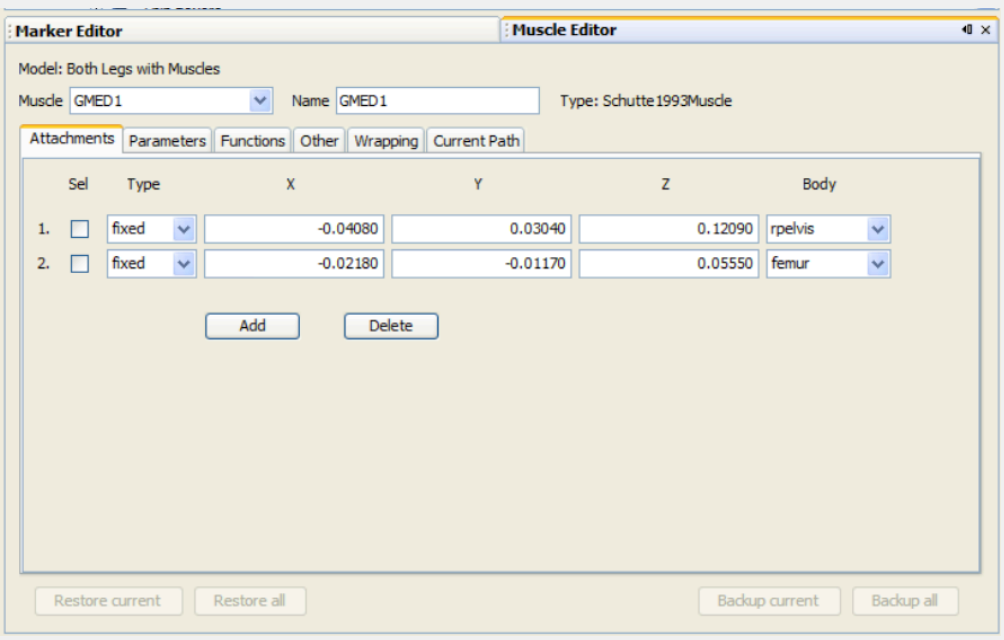
Needed: graphic illustrating the steps below

There are two methods to select a muscle. The first uses the Navigator window:



1. Expand the tree for the model to access the muscle of interest. You can do this by clicking on the **plus (+)** sign next to the name of the current model to display the model components. Click on the **plus (+)** sign next to **Forces**, and then click on the **plus (+)** sign next to **Muscles** to display the list of muscle groups. If the muscles in your model are not organized into groups, you will see a list of the muscles instead of groups. Expand the group of the muscle you want to edit by clicking on the **plus (+)** sign next to the group name.
2. **Right click** on the name of the muscle.
3. Choose **Edit…** from the drop down menu, and the Muscle Editor will update to show the properties of that muscle.

The second method of selecting a muscle for editing is from within the Muscle Editor window.



1. In the top-left corner of the window, just below the model name, there is a box containing a list, sorted alphabetically, of all of the muscles in the current model.
2. The selected muscle is shown in the box.
3. To switch to a different muscle, left click on the box and choose another muscle.

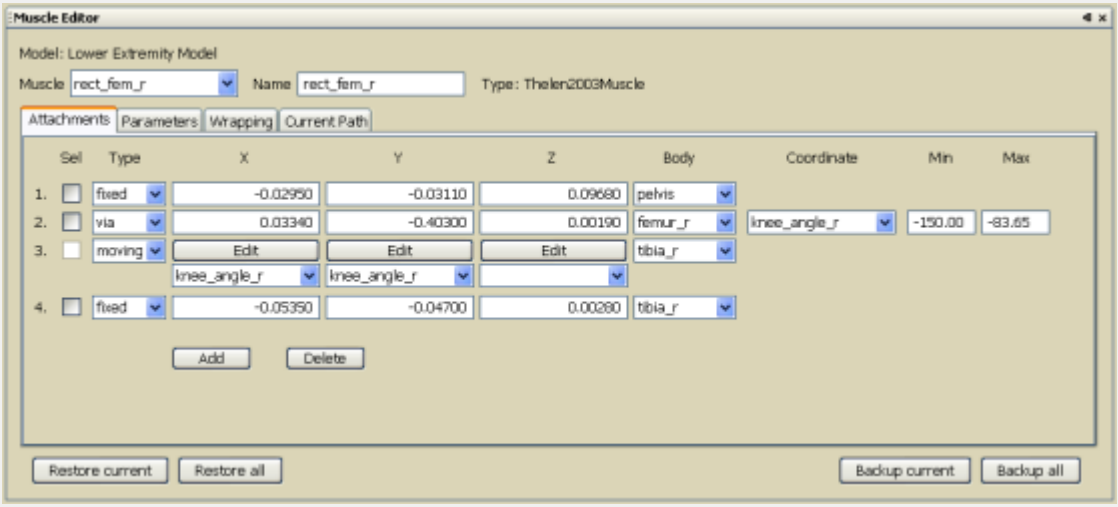
Next: Muscle Editor Panels

Home: Muscle Editor

Muscle Editor Panels

The topics included in this section include:

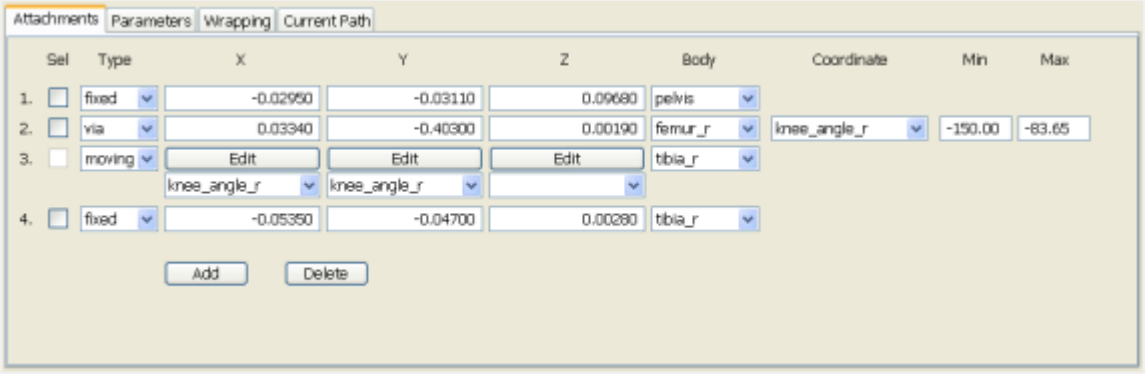
The properties of a muscle are organized into panels within the Muscle Editor window, figure below. The Attachment Panel shows all of the attachment points (fixed, via, and moving) that define the muscle path.



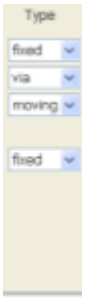
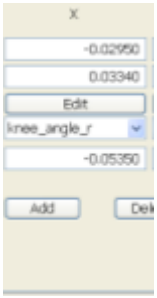
- The Attachment Panel, displays the fixed, via, and moving muscle points that define the muscle path.
- The Parameters Panel contains the force-generating parameters, such as optimal fiber length, pennation angle, and the activation time constants. The exact set of parameters in this panel depends on the type of muscle. There are currently two types of muscle supported in OpenSim. Thelen2003Muscle was developed by Darryl Thelen, Ph.D. [1], and Schutte1993Muscle was developed by Lisa Schutte, Ph.D. [2].
- The Functions Panel contains the force-generating parameters that are functions, such as the force-length curve in a Schutte1993Muscle. Muscles that have no function parameters, such as muscles of type Thelen2003Muscle, do not have a *Functions* panel.
- The Wrapping Panel displays the list of wrap objects that are currently associated with the muscle.
- The Current Path Panel shows the list of attachment points that currently define the muscle path.

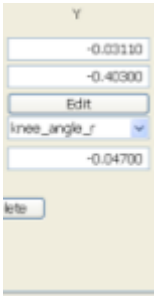
Attachments

The *Attachments* panel gives you access to all of the fixed, via, and moving muscle points that define the path of the muscle. Here is a description of each column of information:

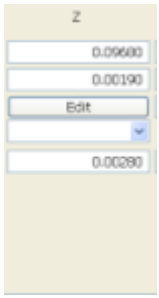


<muscle point index> – the index of the attachment point. This index is used by the Add and Delete buttons, described below, as well as the start point and end point parameters for wrapping. These indices do not change when via points turn on or off or when wrapping points are added to the path. They change only when you add or delete fixed, via, or moving attachment points.

	**Sel** – this checkbox indicates whether or not the muscle point is selected in the model window. Selected attachment points can be dragged to new locations using the mouse.
	**Type** – the type of the attachment point: fixed, via, or moving. Fixed points are points whose XYZ offsets are fixed in a body's reference frame. Via points are points that are fixed to a body, but they are used in the muscle path only when a specified coordinate is in a certain range. Moving muscle points are points whose X, Y, and/or Z offsets in a body's reference frame are functions of coordinates, rather than simple constants.
	**X** – the X offset of the attachment point in the body's reference frame. For fixed and via points, this offset is a constant. You can change the value by clicking in the number field and entering a new value. For moving muscle points, the offset is a function of a coordinate, rather than a constant. To edit the function, click the **Edit** button to display it in the Function Editor window.



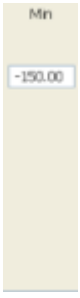
Y – the Y offset of the attachment point in the body's reference frame. For fixed and via points, this offset is a constant. You can change the value by clicking in the number field and entering a new value. For moving muscle points, the offset is a function of a coordinate, rather than a constant. To edit the function, click the **Edit** button to display it in the Function Editor window.



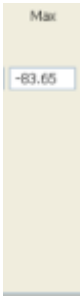
Z – the Z offset of the attachment point in the body's reference frame. For fixed and via points, this offset is a constant. You can change the value by clicking in the number field and entering a new value. For moving muscle points, the offset is a function of a coordinate, rather than a constant. To edit the function, click the **Edit** button to display it in the Function Editor window.

Body – the body to which the muscle point is attached.

Coordinate – For via points, this is the coordinate that controls whether or not the point is included in the muscle path (active). The point is active only when the coordinate is between the specified minimum (Min) and maximum (Max) values.



Min – For via points, the lower bound of coordinate values for which the point is active. If the coordinate has a value that is below this minimum, the attachment point is not included in the path of the muscle.



Max – For via points, the upper bound of coordinate values for which the point is active. If the coordinate has a value that is above this maximum, the attachment point is not included in the path of the muscle.

Parameters

This panel shows the fixed, via, and moving muscle points that define the muscle path. In this example, the path is defined by four attachment points. The first point is fixed to the pelvis and the second point is a via point on the femur that is active only when the knee is flexed between -83.65 and -150.0 degrees. The third point is attached to the tibia, but its location changes as the knee flexes. The fourth point is a point fixed to the tibia.

Below the list of attachment points are buttons for adding and deleting points. To add an attachment point, click on **Add** and then choose where you would like to place the point in the list of existing ones. The indices in the drop down menu refer to the indices listed in the left-most column next to each point in the *Attachments* panel. To delete a point, click on **Delete** and then choose the index of the point to be deleted.

The *Parameters* panel gives you access to all of the parameters that define the force-generating behavior of the muscle. The specific set of parameters displayed depends on the type of muscle.

For muscles of type Thelen2003Muscle, this panel will display the length parameters such as optimal_fiber_length and tendon_slack_length, the activation time constants, and the coefficients describing the force-length properties, such as KshapeActive. To modify any of these parameters, just enter a new value into the number field next to the name of the parameter.

For muscles of type Schutte1993Muscle, the panel will display the length, activation, and pennation angle parameters. Again, these can be modified by entering a new value into the number field next to the parameter name. The force-length parameters, however, are normalized functions instead of constants, so they are displayed in the *Functions* panel.

Functions

The *Functions* panel displays the force-generating properties of the muscle that are functions, rather than constants. For muscles of type Schutte1993Muscle, these functions are: tendon_force_length_curve, active_force_length_curve, passive_force_length_curve, and force_velocity_curve. To modify one of these functions, click on the **Edit** button next to its name and then use the Function Editor to move, add, and delete the function's control points visit Function Editor for more information.

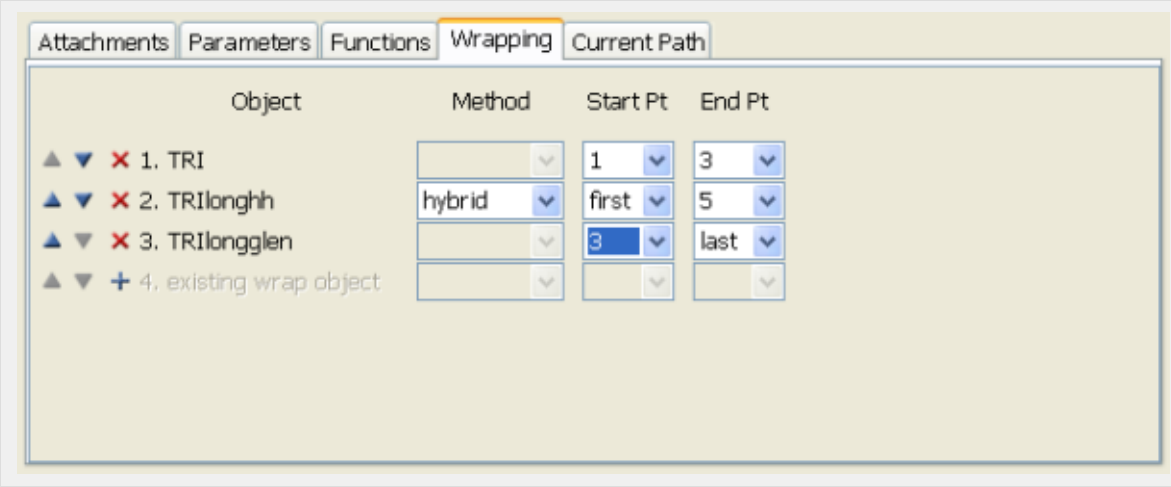
Muscles of type Thelen2003Muscle do not have any parameters that are functions, so the *Functions* panel is not shown.

Wrapping Panel

The *Wrapping* panel shows the wrap objects that are currently associated with the muscle, and provides an interface for adding, deleting, and modifying these associations. When more than one wrap object is associated with a muscle, the order in which the objects are applied to the muscle is the order in which they are listed in the *Wrapping* panel. In many cases, the ordering of wrap objects does not affect the resulting path of the muscle, but when the objects are close to or intersect each other, changing the order may change the muscle path.

To move a wrap object up in the order, click on the blue up arrow to the far left of the object's name. To move it down in the order, click on the blue down arrow. To delete the association, click on the red "x". This does not delete the wrap object from the model; it merely removes it from the list of objects applied to this one muscle. To add a wrap object association, click on the blue + sign and choose a wrap object from the drop down menu.

If the wrap object is an ellipsoid, you can select one of three methods to calculate the muscle path over its surface: midpoint, axial, and hybrid. These methods are described below in the section on the Current Path.

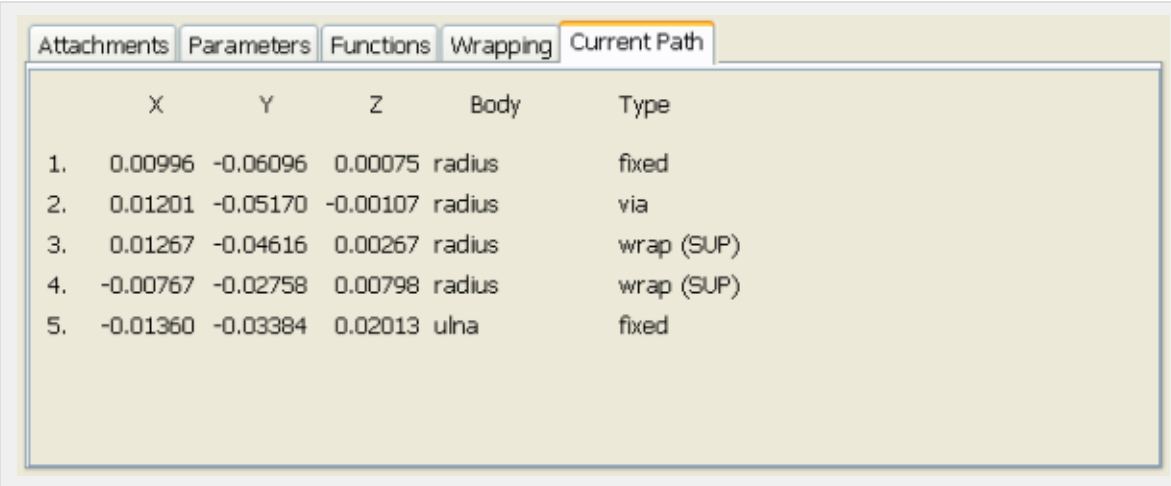


This panel shows all of the wrap objects that are associated with the muscle. You can add to, delete from, or reorder the list, as well as restrict the wrapping of each object to a subset of the muscle path. For ellipsoid wrap objects, you can also specify the algorithm used to calculate the wrapping path.

The **Start Pt** and **End Pt** parameters let you control which sections of the muscle are allowed to wrap over the wrap object. For example, if Start Pt = 1 and End Pt= 4, then only the segments of the muscle path between the first and fourth attachment points will be checked for possible wrapping over the wrap object. These indices correspond to the indices of the points listed in the *Attachments* panel. If Start Pt = first, then the starting point will always be the first attachment point in the muscle. Similarly, if End Pt = last, then the ending point will always be the last attachment point. In contrast, if Start Pt = 1, and then you add a new attachment point to the muscle before point 1, Start Pt will automatically be adjusted to 2 so that it still corresponds to the same attachment point in the list. In most cases, you will want to set Start Pt = first and End Pt = last, so that the entire path is checked for possible wrapping. However, for muscles which have more than one wrap object association, or are associated with wrap objects that are constrained (i.e., only half the object is active), it is sometime helpful to restrict the wrapping to a subset of the muscle path.

Current Path

The *Current Path* panel shows the list of attachment points that currently define the muscle path. The list includes all fixed and moving muscle points, via points that are currently active, and wrap points on each wrap object that is currently constraining the path. The XYZ coordinates of the points in their body's reference frame are shown, as well as the body to which they are attached and the type of point. This list of points is updated as the joints of the model move, so it can be a useful debugging tool when you are defining or modifying a muscle path.



Current Path Panel: This panel shows the list of attachment points that currently define the muscle path.

Backing Up and Restoring

When you modify any element of a muscle in the Muscle Editor, the modification occurs immediately to the muscle stored in the model. You do not need to apply the change to make it happen, nor can you cancel the change before it takes effect. To allow you to undo changes made to the muscles, there are four buttons at the bottom of the Muscle Editor window that backup and restore the state of the muscles. When you first load a model into OpenSim, the Muscle Editor makes a backup copy of every muscle, so you can restore them without having to back them up first. Here is a description of each button:

- **Backup All** – This button makes a backup copy of every muscle in the current model, overwriting the previous copies.
- **Backup Current** – This button makes a backup copy of the current muscle, overwriting the previous copy.

- **Restore All** – This button restores all of the muscles in the model from their backup copies, thereby erasing all modifications you have made since the last time each muscle was backed up.
- **Restore Current**– This button restores the current muscle from its backup copy, thereby erasing all modifications you have made since the last time the muscle was backed up.

References

[1] Thelen, D.G., Anderson, F.C., and Delp, S.L. "Generating dynamic simulations of movement using computed muscle control," Journal of Biomechanics, 2003, 36: 321–328.
[2] Schutte, L.M., Rodgers, M.M., Zajac, F.E., "Improving the efficacy of electrical stimulation-induced leg cycle ergometry: an analysis based on a dynamic musculoskeletal model," IEEE Transactions on Rehabilitation Engineering, 1993, 1: 109-125.

Next: Editing Attachment Points

Previous: Selecting Models and Muscles

Home: Muscle Editor

Editing Attachment Points

The paths of the muscles can be modified by selecting attachment points and moving them to new locations in the reference frames of the bodies to which they are attached. You can select multiple attachment points on multiple muscles and move them at the same time. Fixed points and via points can be selected and moved interactively in the model window. Moving muscle points cannot be selected, but the functions defining their movement can be modified with the Function Editor. Wrap points cannot be selected or moved, because their positions are calculated by the muscle wrapping algorithms. The topics covered in this section include:

Selecting Attachment Points

There are two methods to select muscle attachment points. The first method operates in the 3D View window. Press the **ctrl** key to enter the selection mode. You will see the cursor change into a small crosshair to indicate that you are in selection mode. Keep the **ctrl** key pressed and **left click** on an attachment point to select it. If you also hold down the **shift** key, you can select multiple attachment points. To unselect all points, press **ctrl** and **left click** on a point away from the model.

The second method of selecting attachment points is to click on the **Sel** checkbox in the *Attachments* panel. Choose the muscle whose points you want to select (Selecting Models and Muscles), and then click on the *Attachments* tab. Click on the **Sel** box of the desired point(s). You can select points on additional muscles by switching to each of those muscles in turn, and selecting their points in the *Attachments* panel. This method of selecting attachment points is useful if there are several coincident points in the model (e.g., the insertion point of soleus, lateral gastrocnemius, and medial gastrocnemius on the calcaneous), because selecting coincident points in the 3D model view is problematic.

When an attachment point is selected, it is displayed in yellow on the 3D model, and its name is shown in the text bar at the bottom of the OpenSim window.

Moving Attachment Points

There are two methods for moving muscle attachment points. The first method is to move points interactively in the 3D View window. Select the points you want to move visit Selecting Attachment Points for information on how to select attachment points. Move the cursor over any of the selected points, and press the **left mouse** button. While holding the button down, you can drag the attachment points within the plane of the screen. All of the selected points will move the same amount, so it does not matter which one you click on. To move the points in a different plane, release the left mouse button, rotate the model view as desired, and then press the left mouse button on any selected point to resume dragging.

The second method of moving attachment points is to type their exact XYZ offsets into the number fields in the *Attachments* panel. For this method, the muscle points do not need to be selected, but you can only modify the attachments of the current muscle. Click on the appropriate X, Y, or Z number field, and type in the desired value. These offsets are expressed in the reference frame of the body to which the point is attached, which is shown just to the right of the Z field. For the moving muscle points, the number fields are replaced by **Edit** buttons. When you click on an **Edit** button, the function for that offset will be displayed in the Function Editor so you can modify it.

Adding and Deleting Attachment Points

The *Attachments* panel contains buttons for adding and deleting muscle attachment points. These are located below the list of current attachments.

When you click the left mouse button on the **Add** button, a drop down menu is displayed, giving you a choice of where to add the new point. If you add the new point between two existing points, it will be added halfway between the two. If you add the point before the first or after the last point, the new point will be added a short distance away from the appropriate end point. You can then select the point and move it to the desired location, or type its exact XYZ coordinates into the number fields in the *Attachments* panel.

To delete an attachment point, click on the **Delete** button and choose the index of the point you want to delete. For both the add and delete functions, the indices shown in the drop down menus are the same as the indices of the muscle points shown in the *Attachments* panel. They do not depend on whether or not via points are active, and do not take into account any wrap points which may be currently included in the path.

Next: Ellipsoid Wrapping Algorithms

Previous: Muscle Editor Panels

Home: Muscle Editor

Ellipsoid Wrapping Algorithms

To overcome numerical instabilities involved with computing the optimal path over the surface of an ellipsoid, OpenSim allows you to choose between three different algorithms for calculating muscle paths over ellipsoidal wrap objects. The hybrid method works well for most muscles. However, if you notice erratic muscle motion (e.g., skips or jumps when wrapping over an ellipsoid), you can often achieve a smooth motion with the midpoint or axial algorithms. The topics covered in this section include:

Hybrid Wrapping Method

The hybrid method determines a wrapping path by computing a weighted average of the results of the midpoint and axial methods, described below. The accuracy of each result is used to determine its contribution to the overall path. In most cases, this results in a smooth transition between the two wrapping methods. Occasionally you may experience an artificial change in slope in plots generated from muscles that use the hybrid wrapping method. This is a side effect of the hybrid method that can occur as the weighting shifts from favoring the midpoint result to favoring the axial result or vice versa. In these cases you will want to switch to either the midpoint or axial method rather than the hybrid.

Midpoint Wrapping Method

The midpoint method chooses a wrapping plane (i.e., the plane the muscle path will occupy when wrapped around the ellipsoid) by finding the point on the surface of the ellipsoid closest to the midpoint of the imaginary muscle line passing straight through the ellipsoid. This method produces smooth changes in the wrapping path unless the muscle line through the ellipsoid passes close to the center of the ellipsoid. In this situation the midpoint method becomes numerically unstable, creating erratic jumps in the wrapping path it computes. If you are able to orient your wrap objects such that muscles do not pass near the ellipsoid's center, then the midpoint method will produce well-behaved wrapping paths.

Axial Wrapping Method

The axial method chooses one of the ellipsoid's principal axes, X, Y, or Z, and then computes a wrapping path based on the point where the imaginary muscle line passing straight through the ellipsoid intersects the plane perpendicular to that axis (i.e., the X=0, Y=0, or Z=0 planes). For example, if the X-axis is chosen, then the intersection of the muscle line with the x = 0 plane is used to find a point on the surface of the ellipsoid. The surface point is then used to determine the wrapping plane.

The axial method works best when the muscle line is nearly parallel to the chosen axis. The accuracy of the axial method decreases as the angle between the muscle line and the chosen axis increases. Therefore, the axial method automatically chooses the principal axis that is most parallel to the muscle line. This method produces a good result unless the "most parallel" axis happens to change during the range of motion of the muscle. In this situation a discontinuity in wrapping path will occur when the choice of axis switches from one principal axis to another. The hybrid method described above automatically detects when an axis switch is about to take place, and shifts its weighted average to favor the midpoint method to conceal the effect of the axis switch. If you are able to orient your wrap object such that muscles that wrap over it remain mostly parallel to the same axis throughout their range of motion, then the axial method will produce well-behaved wrapping paths.

Next: Function Editor

Previous: Editing Attachment Points

Home: Muscle Editor

Function Editor

The Function Editor allows you to view and modify the parameters of a model that are functions, such as the force-length curve of a muscle or a joint constraint function. The functions are defined using control points and a function type. You can add, delete, and move control points, as well as change the type of the function (e.g., from a natural cubic spline to a GCV spline). The topics covered in this chapter are:

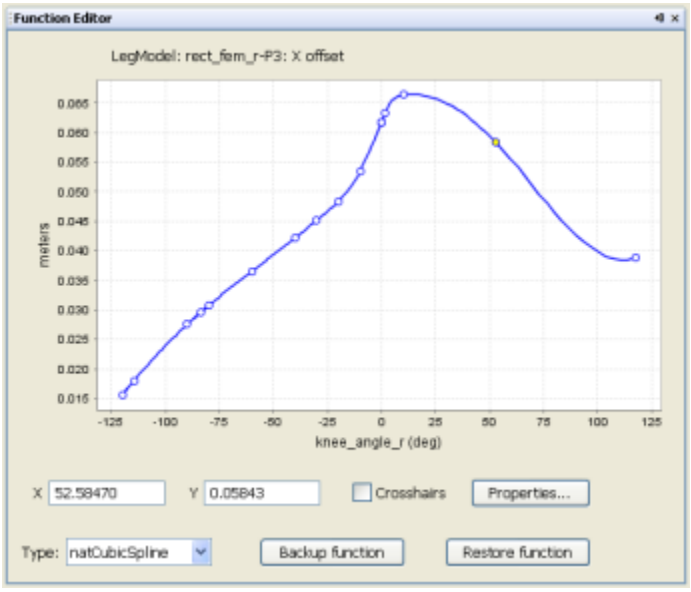
Next: Opening and Restoring Function Editor

Opening and Restoring Function Editor

The topics covered in this section include:

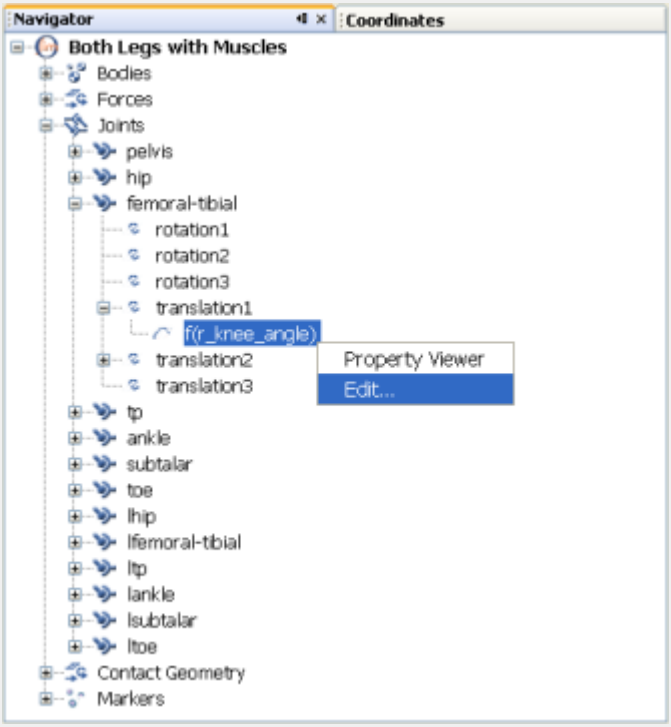
Opening the Editor

The Function Editor window is opened by locating the function you want to modify and choosing the edit option. This will open (or pop, if it is already open) the Function Editor window and load the function into it. Once the window is open, you cannot select another function to edit from within the Function Editor window. You must locate the other function in the Navigator or other model editing tool and choose the edit option for that function. Doing so will close the function that is currently in the Function Editor (saving all changes), and will load the new function.



Opening the Editor for a Joint Constraint Function

To edit a joint constraint function associated with a degree of freedom (DOF) follow the steps below. In this example, the translation1 DOF (the X translation) in the femoral-tibial joint is a function of the coordinate r_knee_angle. Right-clicking on the coordinate name brings up a drop-down menu. Choosing Edit… allows you to load the function into the Function Editor.



1. Locate the DOF in the Navigator window
2. Click on the **plus** sign next to the name of the current model to display the model components.
3. Click on the **plus** sign next to Joints**.**
4. Click on the **plus** sign next to the name of the appropriate joint.
5. Click on the **plus** sign next to Dofs.
6. **Right click** on the name of the coordinate and choose **Edit...** from the drop down menu.
7. The Function Editor window will be opened where you can load the function you want to view and modify.

Opening the Editor for Muscle Property Functions

To edit the force-length curve or other property functions of a muscle, first load the muscle into the Muscle Editor. In the Muscle Editor, click on the *Functions* tab to display the list of muscle properties that are functions. Click on the **Edit** button to load the function into the Function Editor.

Opening the Editor for Moving Muscle Points

To edit the functions that define the movement of moving muscle points, first load the muscle into the Muscle Editor. Then, click on the *Attachments* tab to display the list of the muscle's attachment points. For moving muscle points, the XYZ columns will contain **Edit** buttons. Click on one to load that component's function into the Function Editor.

Backing Up and Restoring

When you modify a function in the Function Editor, the modification occurs immediately to the function stored in the model. You do not need to apply the change to make it happen, nor can you cancel the change before it takes effect. To allow you to undo changes made to a function, there are two buttons at the bottom of the Function Editor window that backup and restore it. When you first load a function into the Function Editor, a backup copy of it is made, so you can restore it without having to back it up first. Pressing the **Backup** button makes a backup copy of the function, overwriting the previous copy. Pressing the **Restore** button restores the function from its backup copy.

Next: Editing Control Points

Home: Function Editor

Editing Control Points

Controls points are used to define the shape of the function. The following operations are possible when working with control points:

select: To select a control point, use **ctrl+left mouse** on the point. To select multiple points, use **ctrl+shift+left mouse** on the points. You can also "box select" points by holding down the **ctrl+left mouse** button and dragging the cursor to form the selection box. You can also hold down the **shift** key while box selecting to select multiple sets of control points. Selected control points are displayed in yellow.

add: To add a control point to the function, put the cursor where you would like to add the point and press the **right mouse** button. Choose **Add control point** from the drop down menu that appears, and a new point will be added to the function.

delete: To delete a control point, put the cursor over the point and press the **right mouse** button. Choose **Delete control point** from the drop down menu.

duplicate point: To duplicate a control point, put the cursor over the point you would like to duplicate, and press the **right mouse** button. Choose **Duplicate control point** from the drop down menu. A very small offset will be added to the point in the X direction, so that it is not exactly coincident with the chosen point. This is done because many types of functions require the control points to be monotonically increasing in the X direction.

move point: Select one or more control points. Then, put the cursor over any of the selected points and press the **left mouse** button. While holding the button down, **drag** the set of points within the plot area. You cannot move the points over one of the adjacent points (i.e., the points in the X direction must always be monotonically increasing).

While you are dragging control points, crosshairs are displayed along with the XY coordinates of the control point under the cursor. Because the displayed coordinates are for the control point, not the actual location of the tip of the cursor, they can help you more accurately position the control point while dragging.

To more precisely move a control point, you can select it and then type its XY coordinates into the **X and Y number fields** below the plot area. When there are multiple points selected, you can type in a Y value to set all selected points to that value, but you cannot set the X value (because that would create coincident points).

Next: Changing the Function Type

Previous: Opening and Restoring Function Editor

Home: Function Editor

Changing the Function Type

The **Type** box lets you change the type of function used to interpolate between the control points. Below is a description of each type.

- **natCubicSpline** – A natural cubic spline is a third-order function with second derivatives equal to zero at the first and last control points. Moving one of the control points has the potential to modify the function over the entire range of control points. Although a natural cubic spline normally requires at least four control points, the implementation in OpenSim supports as few as two points, to be compatible with SIMM. If there are only two control points, linear interpolation is used between them, and if there are three control points, quadratic interpolation is used.

- **GCVSpline** – A GCVSpline is a function that uses generalized cross-validation. The spline can be of degree 1, 3, 5, or 7, and may use smoothing when interpolating the control points. Most GCVSplines in OpenSim are fifth order, and do not include smoothing. Moving one of the control points has the potential to modify the function over the entire range of control points. When you change the type of a function to GCVSpline, the Function Editor will create a fifth-order spline with no smoothing, and thus requires a minimum of six control points. If there are fewer than six points when the type is changed, additional points will be added so that the function contains six control points.

- **PiecewiseLinearFunction** – A linear function interpolates the control points by connecting each pair of adjacent points with a straight line. Moving one of the control points affects only the region of the function between the point and its two neighboring points.

- **StepFunction** – A step function (also called piecewise constant, or zero-order hold) interpolates the control points using horizontal lines between each pair of adjacent points. That is, the value of the function between control points n and n+1 is the Y value of point n. Moving one of the control points affects only the region of the function between that point and the next point.

- **Constant**– A constant is not actually a type of function, but is included for generality. It is implemented as a single Y value for all X values. This type is useful when defining model parameters that are usually functions. For example, in moving muscle points, each of the X, Y, and Z components of the attachment must be a function. If you want one component (e.g., Y) to remain constant, you can set its function type to **Constant**, and enter a single number for it (e.g., Y = 5).

Next: Zooming in the Plot Area

Previous: Editing Control Points

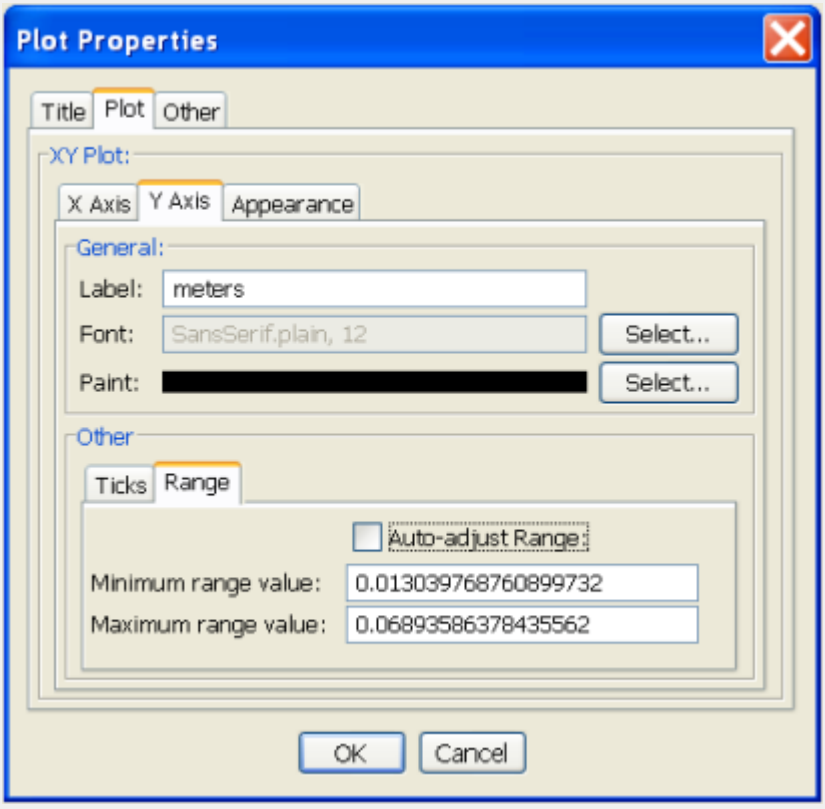
Home: Function Editor

Zooming in the Plot Area

The plot area that displays the function and its control points normally auto-scales to show all of the points. However, if you want to pan or zoom the view to focus on a particular area, there are several ways you can do this.

- **Select a rectangular area** – To select a rectangular area to zoom in on, press the **left mouse** button at the upper-left corner of the rectangle, and hold it down while sweeping down (**dragging**) to the lower-right corner. When you release the left mouse button, the rectangle you defined will be expanded to fit the entire plot area.
- **Hot keys** – Press the **i** and **o** keys to zoom the view in or out, respectively. The zooming is centered on the current cursor location. To pan the view of the function, use the **l, r, u,** and **d** keys to move it left, right, up, and down, respectively.
- **Dialog box**– For more precise control over the zooming of the plot, click on the **Properties** button at the bottom of the Function Editor. In the dialog box for **Plot Properties** that appears, click on the ***Plot*** tab and then on either the ***X Axis*** or the ***Y Axis*** tab. In the lower panel labeled *Other*, click on the ***Range*** tab to display the minimum and maximum values for the axis. Turn off the **Auto-adjust Range** checkbox if it is not already off, and then enter the exact minimum and maximum values you would like to use for that axis.

The Plot Properties dialog box gives you access to many properties for plotting the function. Shown here are the minimum and maximum values for the Y axis. To make the Function Editor auto-scale the Y axis to display all of the control points, turn on the **Auto-adjust Range** checkbox.



- To undo any of the user-defined zooming and panning commands and return the plot to auto-scaling mode, you can either turn on the **Auto-adjust Range** checkboxes for the X axis and the Y axis (see details for the Plot Properties dialog box above), or you can **rectangle-zoom in the opposite direction** in the plot area—that is, while holding down the left mouse button, drag the cursor from the lower-right to the upper-left.
- When you release the left mouse button, the plot will auto-scale to show all of the control points.

Next: Excitation Editor

Previous: Changing the Function Type

Home: Function Editor

Excitation Editor

The Excitation Editor allows you to visually inspect and edit muscle excitation patterns. This can be useful when specifying the input for a forward dynamic simulation (Forward Dynamics) or when examining the outputs of a control algorithm that solves for muscle excitations, for example, the Computed Muscle Control (Computed Muscle Control). In cases where excitations are solved for, the Excitation Editor can also be used to provide an initial guess of the solution.

Although the tool is called the Excitation Editor, it can be used to view and edit any controls waveform described using the OpenSim settings (*.xml)* file format. For example, reserve actuators calculated during a Computed Muscle Control run are written to a controls file along with muscle excitations and can also be viewed and edited using the Excitation Editor tool. The topics covered in this chapter are:

- Opening and Restoring Excitation Editor
- Excitation Editor Command Panel
- Excitation Tree and Excitation Grid Panel
- Excitation Editor Control Panel

Next: Opening and Restoring Excitation Editor

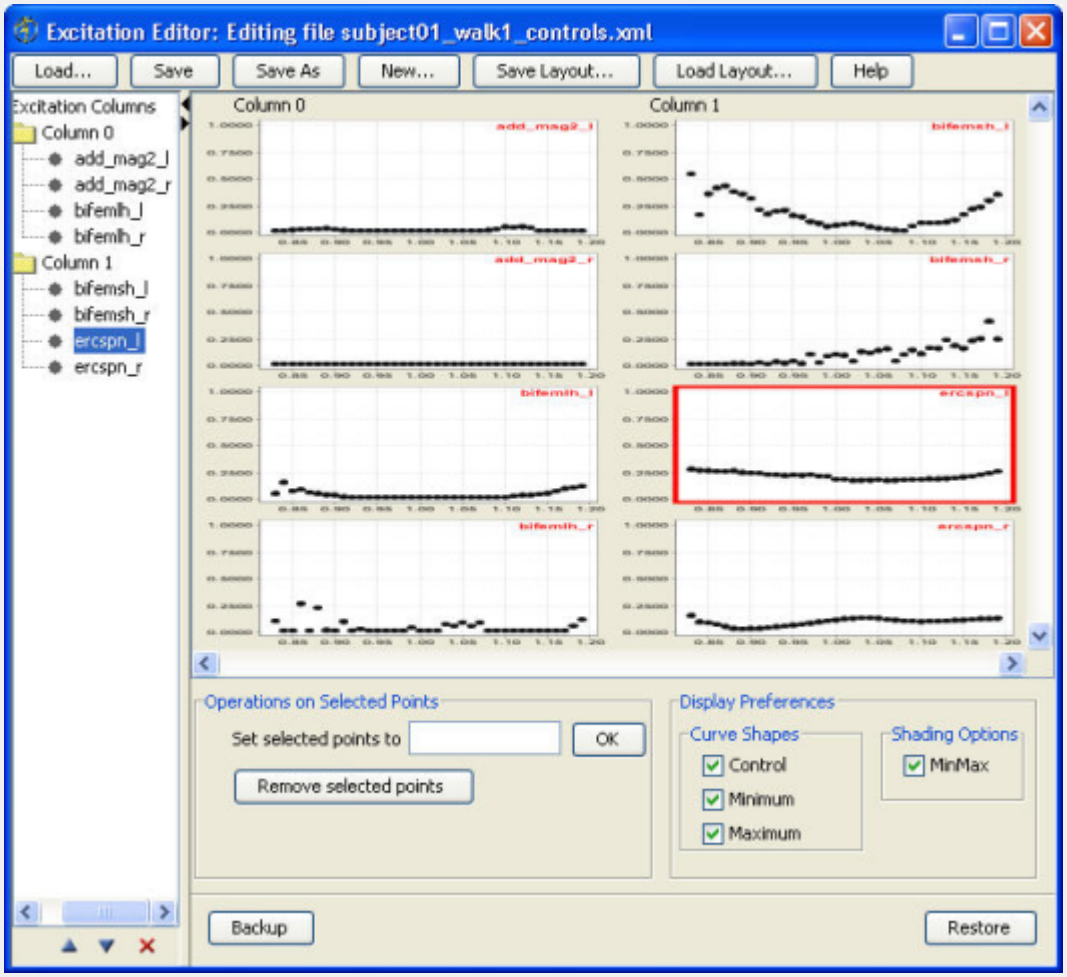
Opening and Restoring Excitation Editor

The topics covered in this section include:

- Opening the Excitation Editor Window
- Backup/Restore

Opening the Excitation Editor Window

The Excitation Editor allows the display and editing of muscle excitations and other control waveforms. In this example, the muscle excitations are organized into two columns. One of the muscle excitations (ercspn_l) is currently selected in the *excitation tree* along the left-hand side of the Excitation Editor. The plot also appears in the *excitation grid* panel with a red border.



- Select **Edit -> Excitations...** from the OpenSim main menu bar.

You can open multiple instances of the Excitation Editor to compare different excitation patterns. The Excitation Editor window remains active and visible until you explicitly close it, so you can run simulations, for example, while the window is up.

Backup/Restore

When you modify an excitation in the Excitation Editor, the modification occurs immediately to the excitation. You do not need to apply the change to make it happen, nor can you cancel the change before it takes effect. To allow you to undo changes made to an excitation, there are two buttons in the *control* panel of the Excitation Editor that backup and restore it. When you first load an excitation into the Excitation Editor, a backup copy of it is made, so you can restore it without having to back it up first. Pressing the **Backup** button makes a backup copy of the excitation, overwriting the previous copy. Pressing the **Restore** button restores the excitation from its backup copy.

The Excitation Editor is a file editor, not a live object editor, so it reads and writes only to a file. If you make changes to a set of muscle excitations, you need to save it to a file before the changes are visible to the rest of OpenSim. Note also that some tools allow the specification of storage files (.sto) for control input (ForwardTool for example). In these cases the excitations are created and can be edited however they need to be saved into an xm-file.

Next: Excitation Editor Command Panel

Home: Excitation Editor

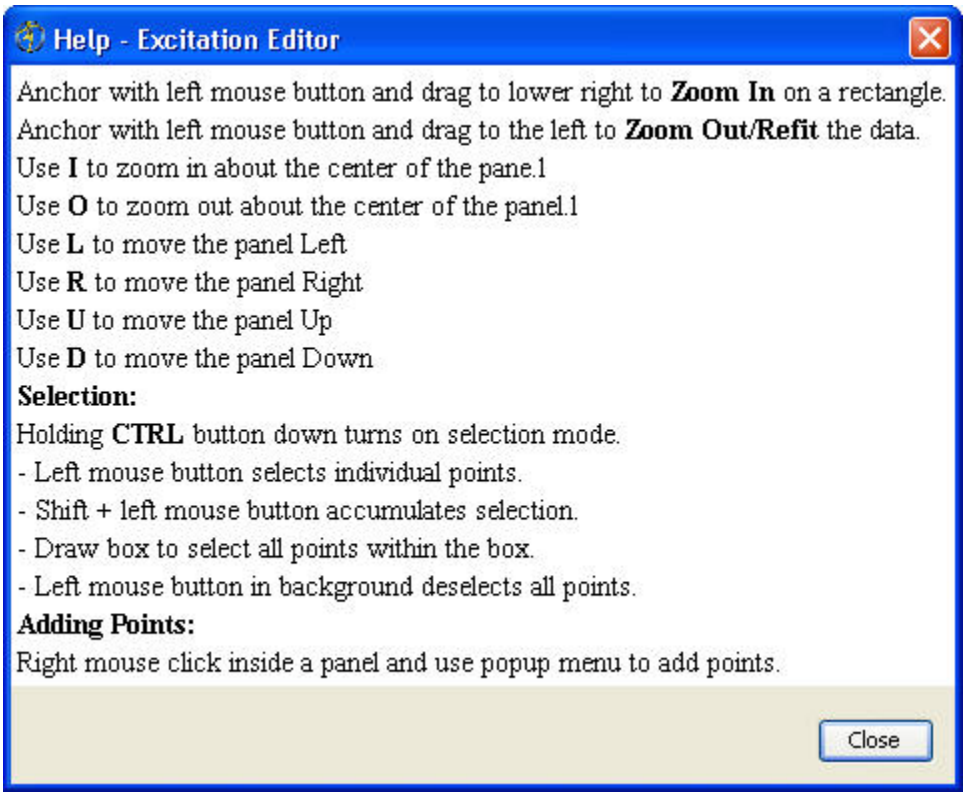
Excitation Editor Command Panel

The *command* panel in the top of the Excitation Editor window allows you to load excitations from an XML file, save them to files, or create a new set of excitations for the current model. Details about the specific commands are described below:

- **Load…**: When you press this button, you are prompted to browse for an XML file containing controls. Upon selecting the file, a filtering dialog box appears (see Selection Filtering Window), from which you can select the controls to be displayed in the Excitation Editor for inspection and editing.
- **Save:** Modified excitations, if any, are saved to the file that they came from
- **New:**This option is available only when a model is currently loaded into the application. Picking this option creates a new set of muscle excitations, with the following properties:
  - Time is between [0.0-1.0]
  - Min value of 0.0
  - Max value of 1.0
  - Excitation value of 0.1

After a new set of muscle excitations is created, the filtering window appears for you to select the excitation patterns you want to edit visually in the Excitation Editor.

- **Save Layout…** and **Load Layout…:** Due to the effort put into laying out excitations in the Excitation Editor (what muscles show and in which column), OpenSim offers the ability to save the layout to an external file (**SaveLayout…**) and to read a saved layout back into OpenSim (**LoadLayout**…).
- **Help:** Hovering over this button brings up a bubble help with short description of the buttons and mouse clicks needed to perform various operations including zooming in/out, panning, selection, box-selection and deselection. Clicking on the button makes the help info persistent (stays up until closed by the user) as shown below.



Next: Excitation Tree and Excitation Grid Panel

Previous: Opening and Restoring Function Editor

Home: Excitation Editor

Excitation Tree and Excitation Grid Panel

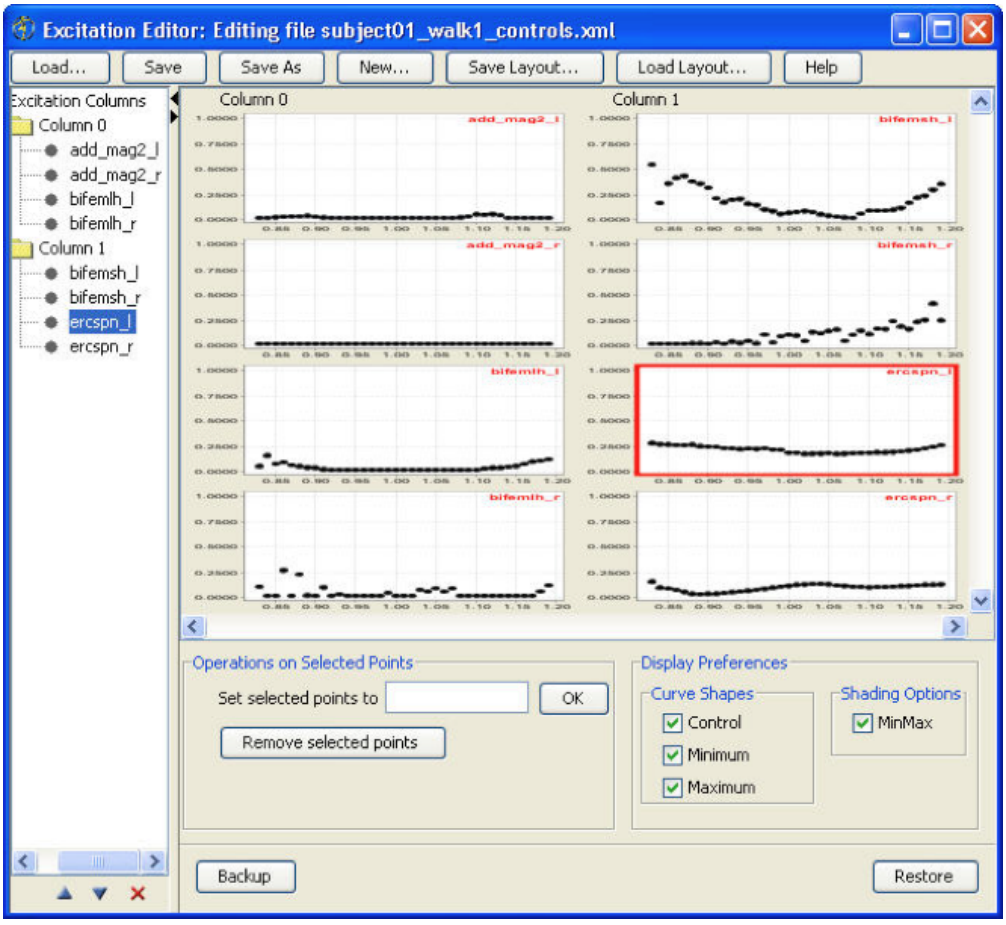
The topics covered in this section include:

Overview

The *excitation tree* and the *excitation grid* panel are the parts of the Excitation Editor that allow you to control the layout of the muscle excitations (or controls, in general) that are displayed. They are discussed together in this section because they represent the same thing. The *excitation tree* , which occupies the left side of the window, shows the names and layout of the excitations displayed in the *excitation grid* panel in a manageable format. You can collapse and expand the *excitation tree* by clicking the arrows at the top of the vertical divider between the *excitation tree* and the *excitation grid* panel.

Excitations are arranged into columns so that you can display excitations in groups based on name or functionality. For example, one common arrangement when studying walking is to arrange excitations into two columns, one for each leg. Another reasonable arrangement is to put different muscle groups of interest into different columns. Columns can have different number of excitations.
Columns have labels, which are displayed at the top of these columns in the *excitation grid* panel. Default names for these columns are "Column 0", "Column 1," etc., but you can change these names (see Excitation Tree Drop Down (Context) Menu).

The *excitation tree* is a simple tree structure with a node/folder for each column and leaf nodes for individual excitations. The order of the nodes in the tree corresponds to the display order of the excitations in the *excitation grid* panel. Columns are displayed left to right, and excitations within a column are displayed in order from top to bottom. The figure below shows an example with two columns, so you could for example use one column for the left glut muscles and the other one for the right glut muscles for easier visual comparison.



Excitation Tree Drop Down (Context) Menus

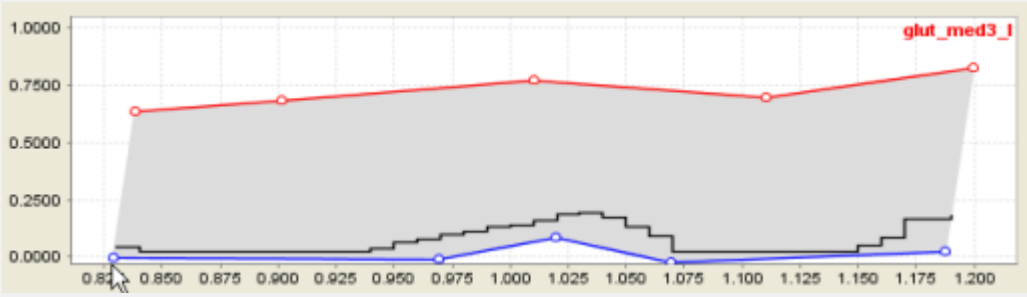
Modifications to the excitation layout can be made by using the drop down (context) menus associated with the *excitation tree.*

add new column	Right mouse click on the root node of the excitation tree and select Add Column…. A new column will be added to the excitation grid panel with the default name "Column i." A filtering dialog box (see Selection Filtering Window) will then appear, so that you can pick excitations to display in the newly added column. Currently displayed excitations are not offered for selection in the filtering dialog box.
add excitation	To pick more excitations and append them to an existing column, **right mouse click** on the node corresponding to that column. Select **Append**. Although the number of excitations per column is originally limited to 8, you can use this option to have more than 8 entries per column.
rename column	To change the display name of a column to a more expressive name (e.g., "Left side" or "Extensors"), **right mouse click** on the node in the *excitation tree* that corresponds to that column and select **Rename**.
reorder excitations	Excitations can be moved up and down inside the same column or across columns using standard drag-and-drop operations in the *excitation tree*.
select excitation	Excitations are selected in the *excitation tree* by just clicking on them. Selected excitations are indicated by a red border (see figure above).
delete excitation	To delete an excitation, select the excitation(s) and then press the  button at the bottom of the Excitation Editor, below the *excitation tree*. These excitations are then removed from the display in the Excitation Editor.

Each entry of the *Excitation Grid* is an *excitation panel* (figure below). An excitation panel is very similar to the panel used by the Function Editor with the difference that at most one set of curves is displayed (excitation curve plus min and max curves, which describe the allowed minimum and maximum values, respectively, for the excitation curve). To add control points to either the excitation or the min or max curves:

Excitation Panel

Excitation panels, such as the one in this figure, are organized in columns in the *Excitation Grid*. Each *excitation panel* shows a control curve (black). It may also show a maximum curve (red) and a minimum curve (blue). The control points for these curves appear as white circles. The display of the control points is controlled by checkboxes in the *Curve Shapes* section of the *Display Preferences* panel. In this example, the checkbox for *Control* is unchecked, so control points do not appear for the control curve. The gray shading between the minimum and maximum curves appears, as shown here, if the *MinMax* checkbox in the *Shading Options* section is checked.



1. Move the cursor to the desired location for the new point.
2. **Right-click** to bring up a drop down (context) menu.
3. Select **Add Control Point to Max** to add a point for the max curve.
4. **Add Control Point to Min** to add a point for the min curve.
5. **Add Control Point to Excitation** to add a point to the excitation. For actual muscle excitations, the allowed range defaults to 0-1 if no min/max curves were specified.

One more option available in the context menu, which is accessed by right clicking in an excitation panel, is the ability to load data from an external file and overlay it on top of an excitation panel. This could be useful when comparing excitations to recorded EMG data.

Next: Excitation Editor Control Panel

Previous: Excitation Editor Command Panel

Home: Excitation Editor

Excitation Editor Control Panel

The *control* panel, located in the bottom-right of the Excitation Editor, allows you to apply operations to selected points in multiple curves, as well as to excitations as a whole. The *control* panel display reflects these different pieces of functionality, which are described below:

Control Point Selection

Control points are selected in one of two ways, either individually or using the box-select functionality. The selection follows a scheme similar to that described in Editing Control Points for control points in the Function Editor.
In particular, to select a control point, hold down the **ctrl** key and **left click** on the point. To select multiple points, hold down the **ctrl** + **shift** keys while **left clicking** on the points.

You can also "box select" points by holding down the **ctrl** key, and then press the **left mouse** button and drag the cursor to the right form the selection box. If you hold down the **shift** key while box selecting, you can select multiple sets of control points.

Selected control points are displayed in blue. It is possible to select points in different panels. Every panel, however, has its own list of selected points, so clicking on a point without holding the **ctrl** key clears the selection but only for the panel that was clicked.

Excitation Selection

Excitations are selected by clicking on the nodes representing them in the *excitation tree* representation. Selected excitations appear in the *excitation grid* panel with a thick red border as shown on the Excitation Tree and Excitation Grid Panel page.

Operations on Control Points

Once you have selected one or more control points, you can drag them to a new location. Put the cursor over any of the selected points, and then press the **left mouse** button. While holding the button down, you can **drag** the set of points within the plot area. You cannot move the points in such a way that any point moves over one of the adjacent points (i.e., the points in the X direction must always be monotonically increasing).

While you are dragging control points, crosshairs are displayed along with the XY coordinates of the control point under the cursor. Because the displayed coordinates are for the control point, and not the location of the tip of the cursor, they can help you more accurately position the control point while dragging.

You can also set control points (across multiple panels) to a fixed value. Select the control points and type in the new value in the text box labeled **Set selected points to**.
You can also remove all selected points in all panels. Select the points and then press the **Remove Selected Points** button in the sub-panel labeled **Operations on Selected Points**.

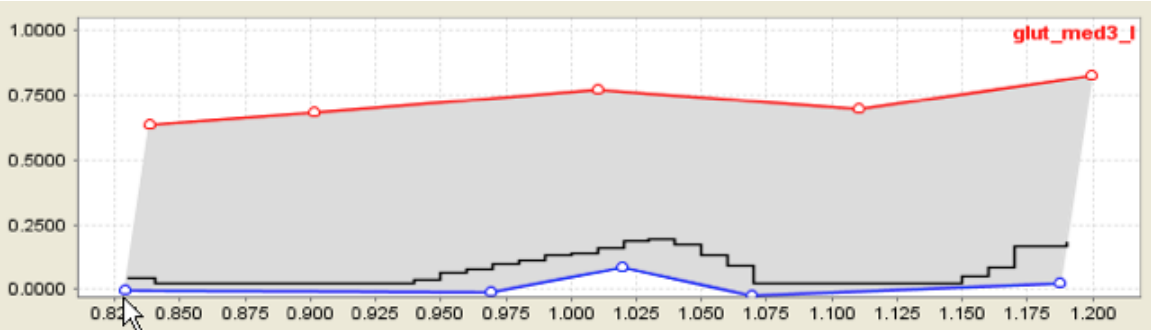
Operations on Excitations

The commands for excitations are located in the *control* panel, the bottom-right panel of the Excitation Editor. Currently, one function is available for excitations: saving them. To save excitations, select them and then click the **Export…** button in the *control* panel. This brings up a dialog box that prompts you for the name of the .xml file in which to save the excitations.

Display Preferences

The *Display Preferences* section provides options to control how the control signal, e.g., a muscle excitation, and/or the min and max curves are displayed. The default is that control points of all the curves are displayed as white circles in the figure below. However, it is possible to turn these off for better visibility by unchecking the **checkbox** next to the appropriate curve type (**Control, Minimum, Maximum**) in the *Curve Shapes* section. Note that turning the circles off disables selection of these control points.

Another display preference is the min/max shading, which controls if the area between the min and max curves, if specified, is shaded. If the checkbox labelled **MinMax** in the *Shading Options* section is checked, shading is used.



Next: Marker Editor

Previous: Excitation Tree and Excitation Grid Panel

Home: Excitation Editor

Marker Editor

The Marker Editor gives you access to all of the markers in the current model. Markers are used by the Scale Tool to scale a generic model to fit a particular subject, and by the Inverse Kinematics Tool to solve for coordinate values corresponding to experimentally recorded marker positions. The Marker Editor lets you add and delete markers, as well as change their names, offsets, and the bodies to which they are attached.

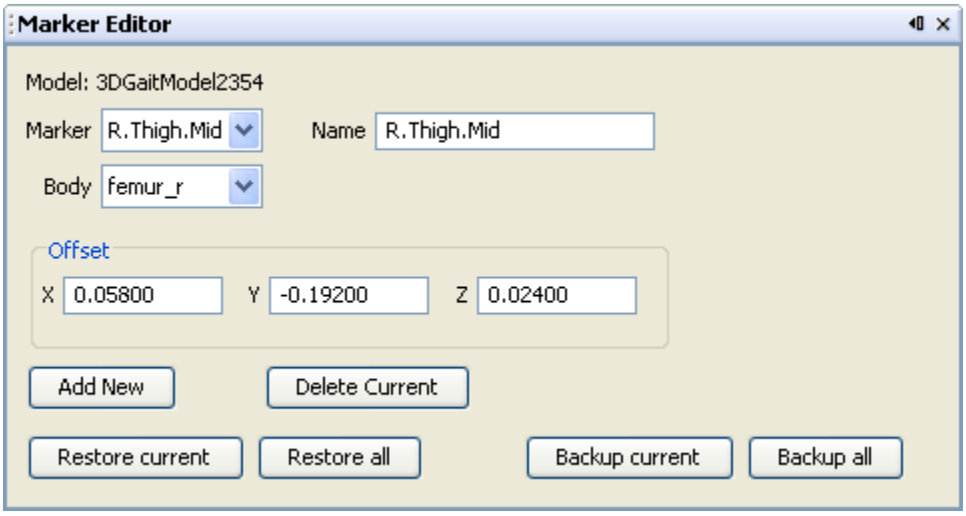
In most cases, you will want to store a model's markers in a separate file. This allows you to use one musculoskeletal model with different marker sets. The Marker Editor is useful when you are first defining a marker set for a project, and for adjusting markers after the Scale Tool has placed them on a model. The topics covered in this chapter are:

Next: Using Markers

Using Markers

The topics covered in this section include:

The Marker Editor



Selecting a Model to Edit

Like all of the model editing tools, the Marker Editor operates only on markers in the current model, visit Opening, Closing, and Using the Navigator Window for more information about how to make a model current. When you change the current model, the Marker Editor automatically switches to operate on this model, and displays the first marker in the model.

Selecting a Marker to Edit

To select a marker for editing, you must first make sure that the model containing the marker is the current model visit Opening, Closing, and Using the Navigator Window for more information about how to make a model current.

There are three methods to select a marker. The first uses the Navigator window:

1. Expand the tree for the model to access the marker of interest. You can do this by clicking on the **plus** sign next to the name of the current model to display the model components, and then clicking on the **plus** sign next to **Markers.** In some models, the markers are organized into groups, in which case you will also have to click on the **plus** sign next to the group containing the marker.
2. **Right click** on the name of the marker.
3. Choose **Edit…** from the drop-down menu, and the Marker Editor will update to show the properties of that marker.

The second method operates in the 3D View window. Press the **ctrl** key to enter selection mode. You will see the cursor change into a small crosshair to indicate that you are in selection mode. Keep the **ctrl** key pressed and **left click** on a marker to select it. If you also hold down the **shift** key, you can select multiple markers. The last marker you select will be displayed in the Marker Editor window and used as the current marker. To unselect all markers, press **ctrl** and **left click** on a point away from the model.

The third method of choosing a marker for editing is from within the Marker Editor window. In the top-left corner of the window, just below the model name, there is a box containing a list, sorted alphabetically, of all of the markers in the current model. The current marker is shown in the box. To switch to a different marker, left click on the box and choose another marker.

When a marker is selected, it is displayed in yellow on the 3D model, with a yellow line connecting it to the origin of the body to which it is attached. Also, its name is shown in the text bar at the bottom of the OpenSim window. If you select multiple markers, the last one selected will be made the current marker in the Marker Editor.

Editing Markers

To change the name of a marker, left click in the **Name** text field and enter a new name. If you enter a name that is already being used by another marker, an error message is printed. To change the body to which the marker is attached, left click on the **Body** box list and choose a new body.

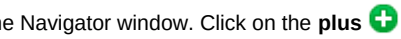
The Marker Editor does not have a mechanism for changing the weights of the markers because this functionality is provided by the various tools that use markers. For example, the Scale Tool and Inverse Kinematics Tool contain **Weights** panels for specifying the weight of each marker during the scaling or inverse kinematics process.

Moving Markers

There are two methods for moving markers. The first method is to move them interactively in the 3D View window. Select the markers you want to move visit Selecting a Marker to Edit for more information on selecting markers. Move the cursor over any of the selected markers, and press the **left mouse** button. While holding the button down, you can drag the markers within the plane of the screen. All of the selected markers will move the same amount, so it does not matter which one you click on. To move the markers in a different plane, release the left mouse button, rotate the model view as desired, and then press the left mouse button on any selected marker to resume dragging.

The second method of moving markers is to type their exact XYZ offsets into the number fields in the *Offset* section. For this method, the markers do not need to be selected, but you can only modify the offset of the current marker. Click on the appropriate X, Y, or Z number field, and type in the desired value. These offsets are expressed in the reference frame of the body to which the marker is attached, which is shown above the *Offset* section.

Adding and Deleting Markers

There are two methods of adding markers to the current model. The first uses the Navigator window. Click on the **plus**  sign next to the name of the current model to display its components, then right-click on the **Markers** category and choose **Add New** from the drop-down menu. A new marker attached to the ground body will be added to the model.

The second method of adding a marker is to click on the **Add New** button in the Marker Editor window. This will also create a new marker attached to the ground body.
To delete a marker, first make sure it is the current marker in the Marker Editor window. Then click on the **Delete Current** button. This will delete only the current marker, not any other markers that are selected in the 3D model window.

Backing Up and Restoring

When you modify the name, body, or offset of a marker in the Marker Editor, the modification occurs immediately to the marker stored in the model. You do not need to apply the change to make it happen, nor can you cancel the change before it takes effect. To allow you to undo changes made to the markers, there are four buttons at the bottom of the Marker Editor window that backup and restore the states of the markers.

When you first load a model into OpenSim, the Marker Editor makes a backup copy of every marker, so you can restore them without having to back them up first. Here is a description of each button:

- **Restore current** – This button restores the current marker from its backup copy, thereby erasing all modifications you have made since the last time the marker was backed up. If the marker was added but not yet backed up, it will be deleted.
- **Restore all** – This button restores all of the markers in the model from their backup copies, thereby erasing all modifications you have made since the last time each marker was backed up. All markers that have been added but not yet backed up will be deleted.
- **Backup current** – This button makes a backup copy of the current marker, overwriting the previous copy.
- **Backup all** – This button makes a backup copy of every marker in the current model, overwriting the previous copies.

Next: Plotting

Home: Marker Editor

Plotting

The plot tool allows you to run a set of standard analyses on a model and plot the results either for visual inspection within OpenSim or to export the results into other applications. In addition to plotting quantities, the tool also allows you to print your plots to a printer, to Postscript files for later inclusion into publications and other electronic forms of publication, or to ASCII data files.

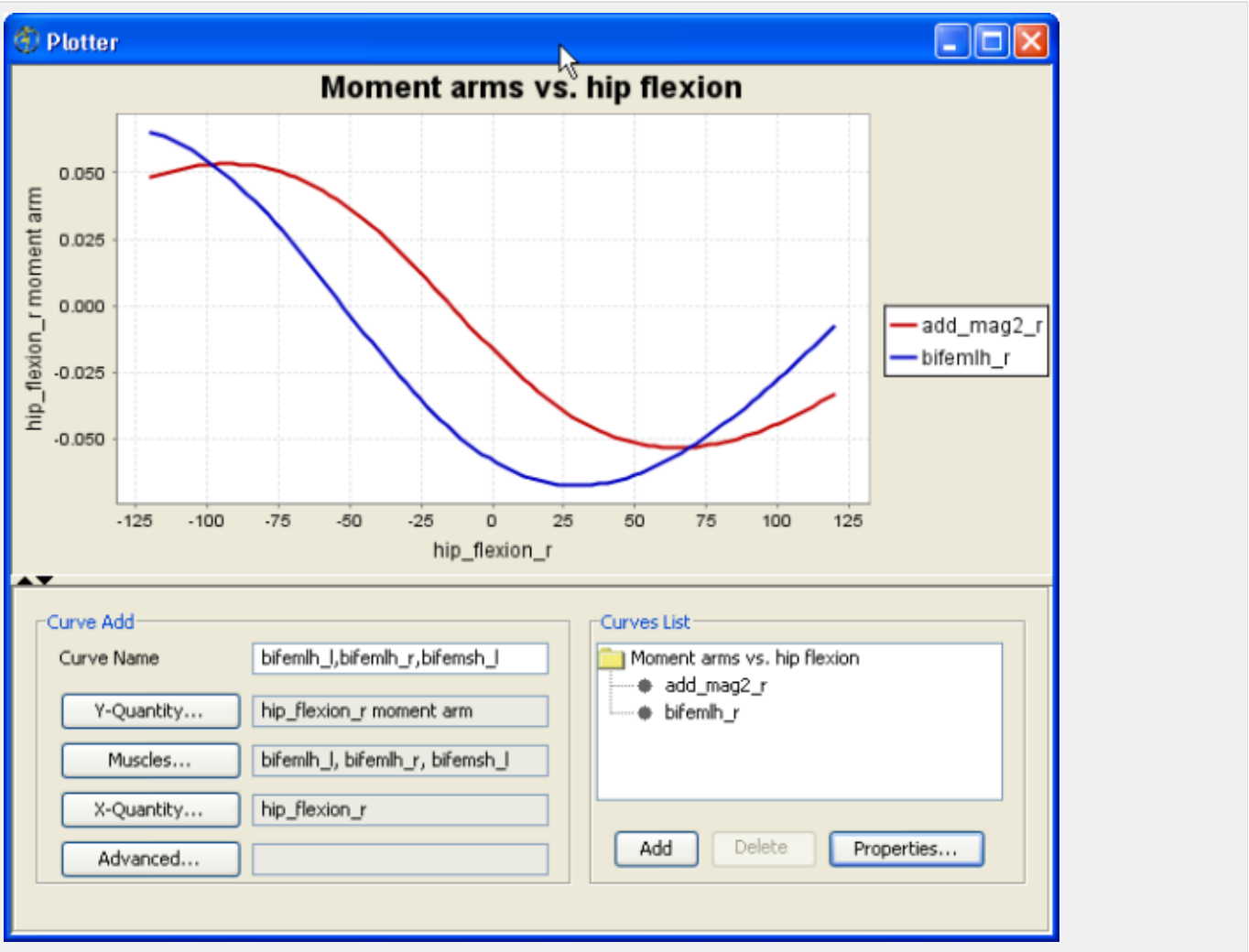
This chapter provides instructions on using the plotting tool, explains the analyses that are built into the plotter, and describes how to tune the plotter to meet your needs and preferences.

Next: Plot Window

Plot Window

Opening and Closing the Plotter Window

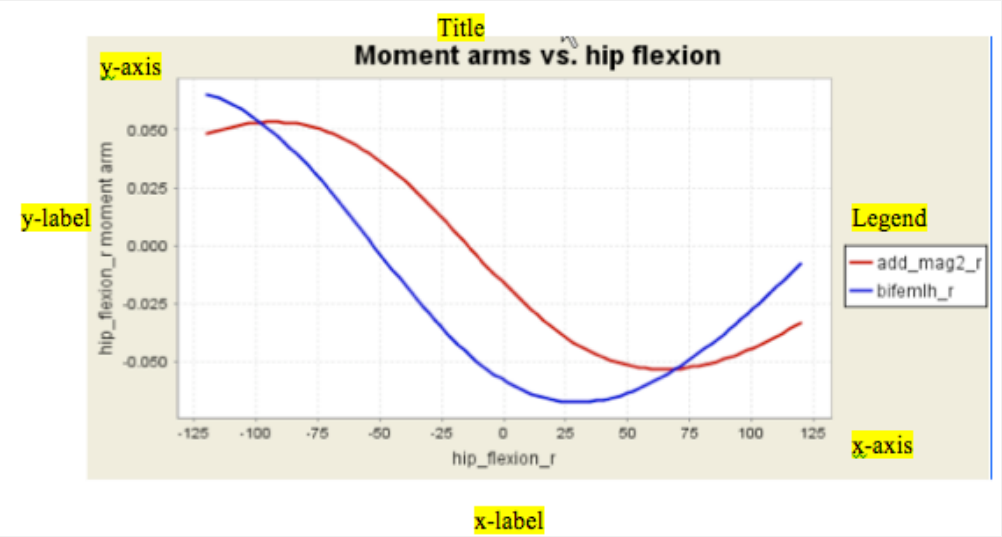
To open/close the Plotter window:



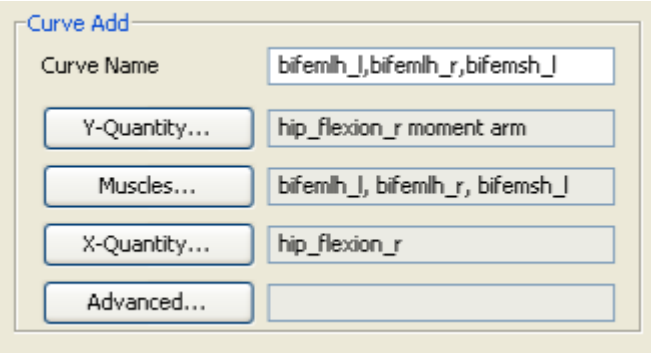
1. To open: select **Tools -> Plot…** from the OpenSim main menu bar. This brings up a blank Plotter window. An example of the plotter (with data) is shown on the left. Multiple instances of the Plotter window can be opened simultaneously.
2. To close: select the standard **"x"** in the upper-right corner of the Plotter window.

The Plotter Window Layout

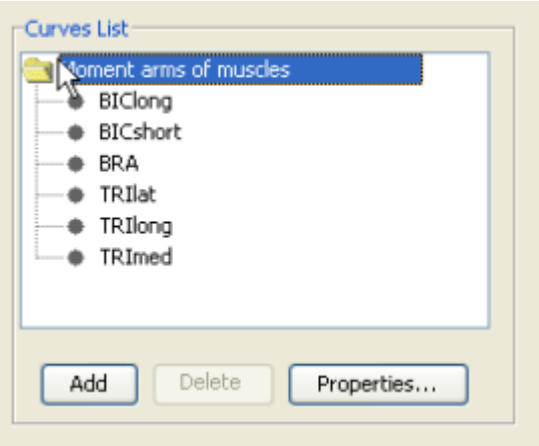
The Plotter window is made up of three panels as shown in the figure above.



- The *plot* panel at the top of the Plotter window displays curves and shows the plot title, x-label, x-axis, y-label, y-axis and legend.



- The *curve creation* panel appears in the lower-left part of the Plotter window and is used to specify the parameters needed to create and add new curves to the *plot* panel. The types of curves that can be displayed are discussed in the Curve Creation Panel section.



- The Plot Summary Panel is the lower-right part of the Plotter window and is used to display the name and summary information for curves that are currently shown. There are also buttons to **add** and **delete** curves from the *plot* panel, as well as to change the display **properties** of the *plot* panel. Any changes made will be reflected in the *curves list* seen to the left.

The bottom half of the Plotter window (containing the *curve creation* and the *plot summary* panels) can be collapsed and expanded using the arrows on the horizontal divider immediately under the *plot* panel. This can be useful for better utilization of the space on the screen.

Next: Plot Summary Panel

Home: Plotting

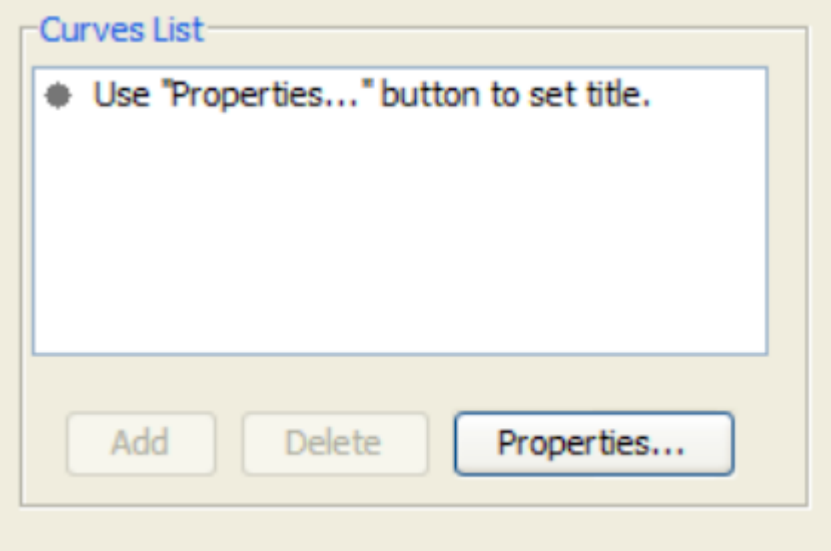
Plot Summary Panel

The topics covered in this section include:

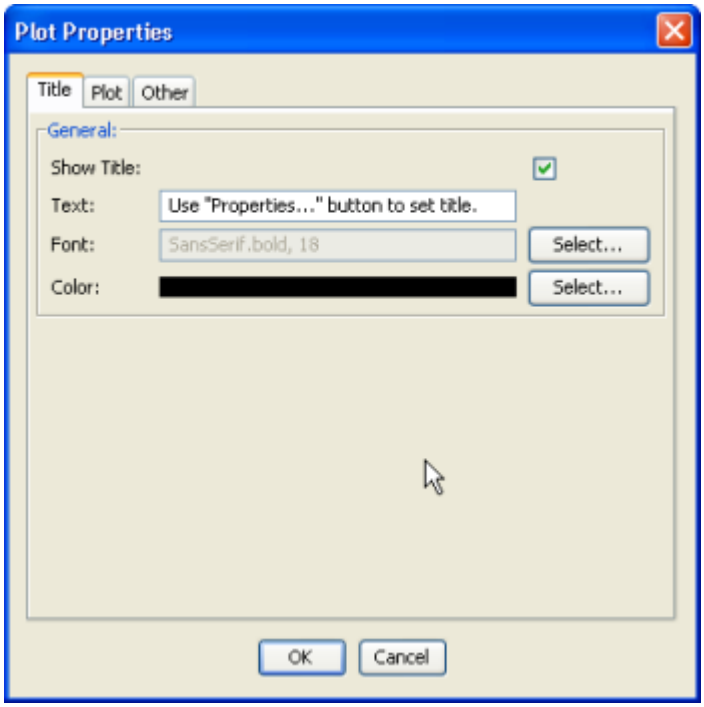
The *plot* panel is the top part of the Plotter Window and includes the drawing area, title, legend, and axis labels shown in the section on the Plot Window. Customization of this area is done through a pop-up menu (context menu) that can be accessed by clicking the **right mouse** button anywhere in the *plot* panel. The context menu is shown in the figure below. The options available through the context menu are discussed in the following sections.

Summary Panel

The *plot summary* panel shows a list of the curves displayed in the *plot* panel, arranged in a tree structure. The buttons at the bottom of the panel control what appears in the *plot* panel:



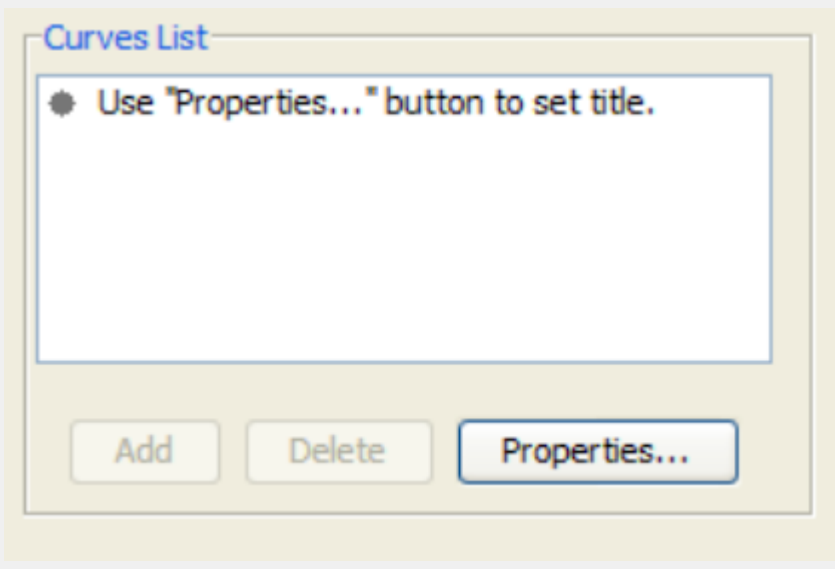
- add: To add a curve to the display, specify the curve in the Curve Creation Panel and then press the **Add** button. The name of the curve will appear at the bottom of the Curves List.
- delete: To remove a curve from the display, **left click** on the name of the curve to be deleted. Then, press the **Delete** button. The name of the curve will be removed from the Curves List. If no curves are selected, the Delete button is disabled.
- edit plot properties: To control the properties of the *plot* panel (e.g., title, colors, axis names and scale), press the **Properties…** button at the bottom of the *plot summary* panel. The Plot Properties window (below) will appear. This is the same window used in the Function Editor section.
- You can also access the Plot Properties window by clicking the right mouse button in the *plot* panel and selecting **Properties…** from the drop down menu that appears.



- Use this panel to edit the title, x-axis or y-axis labels and scales.

Plot Drop Down Menu

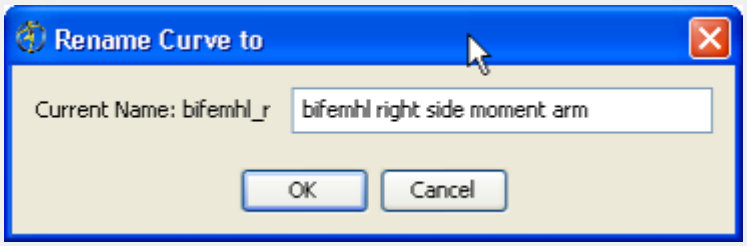
The plot drop down menu is accessed by clicking the **right mouse** button on a given plot name in the *plot summary* panel. The plot names usually appear next to a folder icon. The following options are available from the plot drop down menu:



- For more information on properties see Plot Properties
- For more information on printing and exporting plots see Exporting and Printing
- Use the **Zoom In** and **Zoom Out** options to zoom on different areas of the plot.
- **Auto Range** formats the plot on a "best fit" set of axes.

Curve Drop Down Menu

The curve drop down menu is accessed by clicking the **right mouse** button on on a given curve name in the *plot summary* panel. The curve names usually appear in the list next to a gray dot. The following options are available from the curve drop down menu:



- **Rename** – Changes the name of the selected curve both in the *plot summary* panel and in the legend of the plot, which is shown in the *plot* panel. Picking this option brings up the dialog box on the left, which is populated with the current name of the selected curve. You can change the name to something more meaningful by typing a new name in the box and clicking the **OK** button.

- **Info** – Displays the dialog box on the left, with information about the data source and attributes. This helps distinguish curves from one another, since the auto-generated name for curves is generally not unique. For example, a plot of the muscle-tendon length of "rect_fem" and a tendon length of "rect_fem" will both appear in the *plot summary* panel as "rect_fem" by default. Using the **Info** option enables you to disambiguate which is which.

Next: Curve Creation Panel

Previous: Plot Window

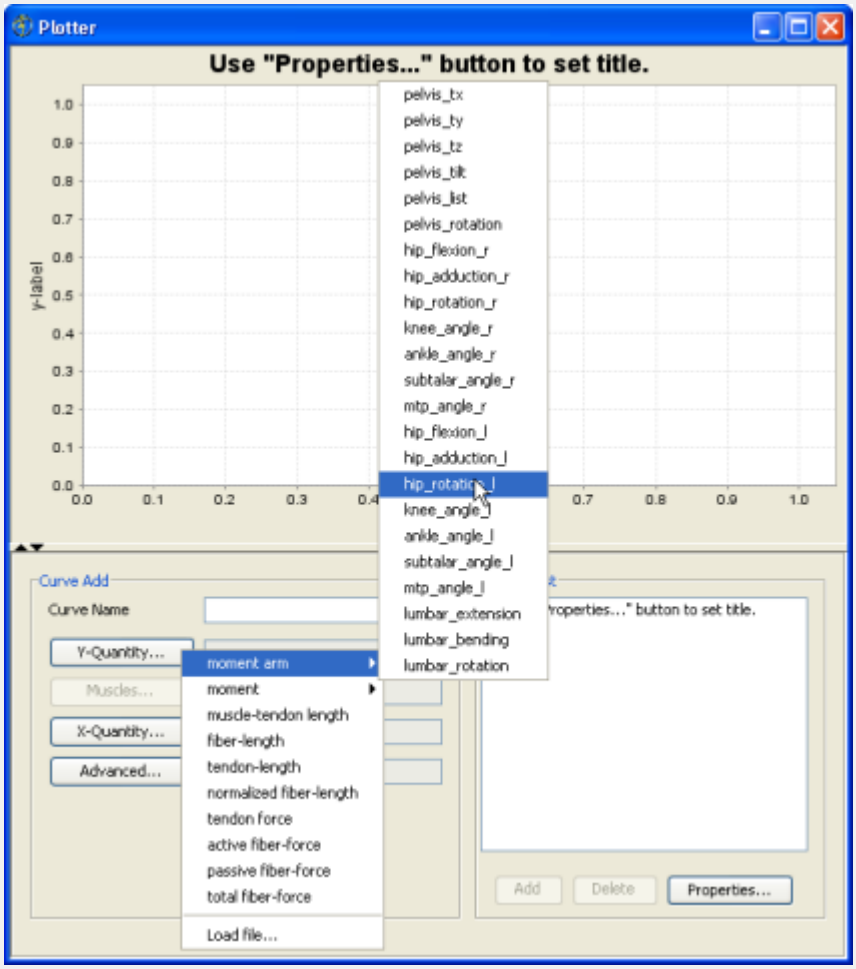
Home: Plotting

Curve Creation Panel

The topics covered in this section include:

Creating a Curve in the Plotter

Curves displayed in the *plot* panel are either based on data generated by OpenSim (*built-in curves*) or read from an external source (*external curves*). The general process for creating a curve is as follows:



1. Select the quantity that you want to plot along the Y-axis by clicking the **Y-Quantity…** button. A drop down menu will appear, displaying available options (figure left). The options will vary, depending upon the data source (i.e., from an OpenSim analysis or an external source).

2. Additional information may be needed to fully specify the quantity of interest. In this case, a small triangle appears to the right of the quantity and can be used to display a second menu. For example, the moment arm has to be defined relative to a specific generalized coordinate, so if you pick "moment arm" from the Y-Quantity menu, a second menu appears so that you can select the set of generalized coordinates.

3. If the Y-Quantity selection requires muscle specification, the **Muscles…** button is enabled. You can select multiple muscles from the current model, as well as sum over subsets of muscles. If no muscles are selected, the **Add** button in the *plot summary* panel to add new curves is dimmed out.

4. Pressing the **Muscles…** button brings up the filtering dialog box used to select muscles visit Selecting Models and Muscles for more information about selecting muscles. In this context, you are able to keep the dialog box open and still make changes in other windows, for example, to change your Y-Quantity selection. This is useful when plotting different quantities for the same set of muscles.

5. Select the quantity that you want to plot along the X-axis by clicking the **X-Quantity…** button. A drop down menu will appear, displaying available options.

6. You may have noticed that an auto-generated name was created when you selected your **Y-Quantity**. If you want, you can specify a new name for the curve by entering the name in the box for **Curve Name**. This name can also be changed after the curve has been generated by using the curve drop down menu visit the section on the Plot Summary Panel for more information about the curve drop down menu. This name will be used to identify the curve in the *plot summary* panel, as well as in the plot legend.

7. Press the **Add** button in the *plot summary* panel to run the desired analyses and display the resulting curves in the *plot* panel.

Built-in Curves

OpenSim provides built-in analyses for some commonly computed quantities, such as moments and moment arms. The curves produced by

these analyses are referred to as *built-in curves.* Built-in curves are computed for the current model. The following built-in curves are available:

- **Moment-arm**: A moment arm curve of a specific muscle (m) about a generalized coordinate (g_y) against a generalized coordinate (g_x) varies g_x along its range of valid values in equal increments (100 by default). As the model is moved, the moment-arm of the selected muscle about g_y is reported.
- **Moment**: A moment curve of a specific muscle (m) relative to a generalized coordinate (g_y) against a generalized coordinate (g_x) is similar to the moment-arm curve above, except that the moment of the selected muscle about g_y is reported instead.
- **Muscle-tendon length**: A muscle-tendon length curve for a specific muscle (m) relative to a generalized coordinate (g) varies g along its range of valid values in equal increments (100 by default). As the model is moved, the total length of the muscle-tendon complex is reported.
- **Fiber-length, Tendon-length, Normalized Fiber-length**: These curves are drawn against a generalized coordinate (g). OpenSim generates these curves by varying g along its range of valid values in equal increments. For each position, the muscles are left to reach equilibrium based on current configuration and muscle properties (muscle tendon force, muscle force, and pennation angle are accounted for).
- **Tendon-force, Active fiber-force, Passive fiber-force, Total force**: These are similar to the fiber-length, tendon-length, and normalized fiber-length set of curves. In this case, the generalized coordinate selected for the X-Quantity is varied, and the model is left to reach equilibrium. Forces in the tendon, active force and passive force (e.g., based on force-length relation of the muscle), as well as the total force in the fiber are reported.

Motion Curves

Instead of varying a generalized coordinate to compute the curves in Built-in Curves, you can use a motion file to change the model configuration and compute the same quantities as described in Built-in Curves. In this case, the drop down menu associated with the **X-Quantity…** button contains the names of the columns of the motion file. Motion files have to already be associated with the model in order to be used by the plotter.

As new motions are added, they will appear in the drop down menu associated with the **Y-Quantity**… button.

Although it is possible to pick any column of the motion to plot against, it is customary to select "time" so that the quantities of interest are plotted through a given motion. This can be useful, for example, to plot moments generated during a complete motion trajectory.

External Curves

You can also plot data generated by other tools in OpenSim. For example, running the forward tool (How Forward Dynamics Works) generates a set of storage (.sto) files recording model states and the forces generated by various actuators in the model during a forward simulation. These can be plotted in the Plotter window.

To plot these results, select **Y-Quantity -> Load file…**. A dialog window appears for you to select the file containing the data. Once the file has been specified, you will be prompted to select which column in the file is to be used for the X-axis (domain) and which for the Y-axis. You can select more than one column to plot along the Y-axis. Once the file has been, it is available until the Plotter window is closed.

Next: Exporting and Printing

Previous: Plot Summary Panel

Home: Plotting

Exporting and Printing

The topics covered in this section include:

- Writing Data to External Files
- Exporting Images
- Exporting to PostScript
- Printing

Writing Data to External Files

You can save the data representing the curves to use at a later time within OpenSim or to import into another application, such as Excel. The saved file uses the .sto format, text files that contain time-sequence data commonly encountered in motion capture systems. The data are arranged in columns with each column representing, for example, a joint angle or a component of the ground reaction force. Each row in a motion file corresponds to a different time point. These are very similar to SIMM motion files (.mot) from MusculoGraphics, Inc., except they have the added flexibility of handling non-uniform time spacing between rows.

There are two ways to save the curves. The first way is click the **right mouse** button in the *plot* panel and select **Export Data…** from the drop down menu that appears. The second way is to **right click** the root node of the tree in the *plot summary* panel and select **Export Data…** from the menu that appears. Picking this option causes a file browser to appear. Using the file browser, select the name of the file and the directory in which to save the file. Press **Save**.

The **Export Data…** option saves all the curves in the displayed plot to the file. Since multiple curves with different domains (x-axes) can be displayed concurrently in the same plot, it is possible that a single storage file would not be adequate to represent this data. In this scenario, a separate file is created for every distinct domain (x-axis). A domain is considered distinct if it has a different number of points.

Exporting Images

You can save your plots as image files for example, for e-mailing or posting on websites. Currently, only the PNG image format is supported. Third-party software tools are freely available to convert PNG images to other formats.

To export a plot as a .png image file, **right mouse click** in the *plot* panel and select **Export Image…** from the drop down menu that appears. A file browser will appear and you will be prompted to select the name of the file and the directory in which to save the file. Press **Save**.

In cases where the image needs to be included in a publication, a better option would be to export the image to the PostScript format (see Preparing Your Data), since advanced editing tools (e.g., Adobe Photoshop) can be used to modify the resulting files (e.g., changing line styles, labels and annotations).

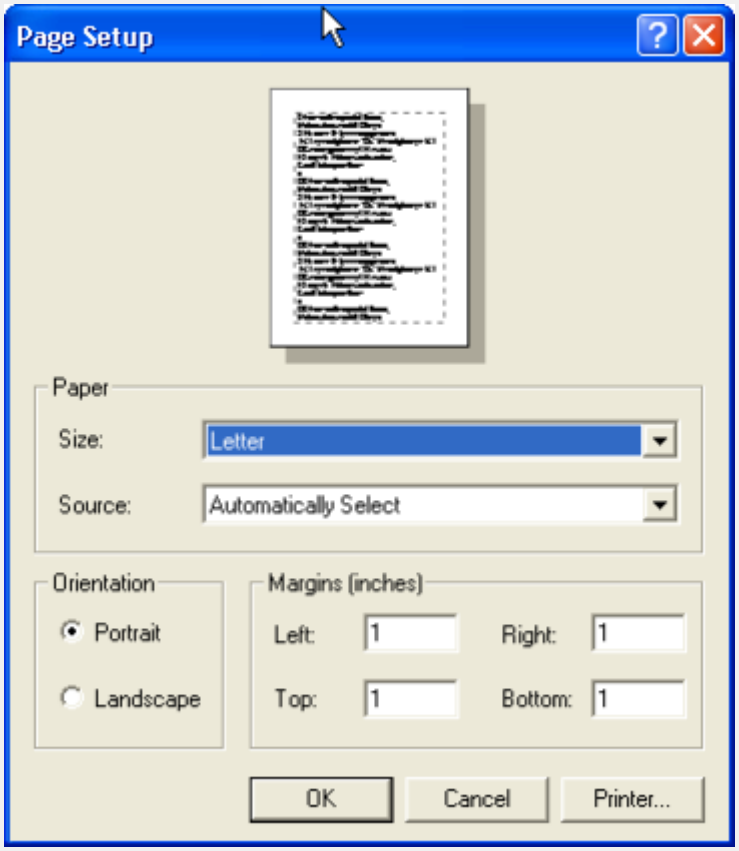
Exporting to PostScript

You can export your image file to the PostScript file format. This is particularly useful if you want to make changes to the plot (e.g., changing the line style or adding annotations) beyond the capabilities provided within OpenSim, since software tools like Adobe Photoshop are able to operate on individual components of a PostScript file.

To export your image file to PostScript, **right mouse click** in the *plot* panel and select **Export PostScript…** from the drop down menu that appears. A file browser will appear and you will be prompted to select the name of the file and the directory in which to save the file. Press **Save**.

Printing

You can print the plot to a printer by:



1. Click the **right mouse** button in the *plot* panel and selecting **Print…** from the drop down menu that appears.
2. The **Page Setup** dialog box shown on left will appear. Use this to pick printing options (e.g., paper size, printing orientation, print margins). Then, press **OK**.

Next: Selection Filtering Window

Previous: Curve Creation Panel

Home: Plotting

Selection Filtering Window

When you press the **Muscles…** button from the *curve creation* panel, the dialog window shown below in Filtering Options is displayed. This window is used in many places throughout OpenSim and allows users to select from a potentially long list of names using different grouping mechanisms. In this case, the window contains the names of all the muscles in the model. To select a few muscles, you can scroll down the list and check the checkboxes next to the muscles of interest. If you want to add the quantity over all selected muscles, check the **sum only** box at the bottom of the window.

For multiple selections, however, the window offers more powerful filtering mechanisms: pattern filtering and muscle group filter (both explained below). If a name is selected and the filtering rules change (e.g., switched from pattern to muscle group or vice versa), selected entries remain selected. You can show all the names available for selection by pressing the **Show All** button. The topics covered in this section include:

- Pattern Filtering
- Muscle Group Filtering
- Selecting Items from the List

Pattern Filtering

In cases where the names of interest have some common suffix, prefix, or any common substring, select the filter option for **pattern** and start **typing the pattern** into the text box next to the word "pattern." As letters are typed in, the list of available names shortens so that only the names matching the typed-in pattern (case insensitive) are displayed. For example, if the user types the letter "g," then *only* muscles whose names contain the letter "g" are displayed in the list.

This comes in handy when selecting muscles from the left side of the body if a naming convention (e.g., using the suffix "*l*" *or the prefix "l*") was used. Another example would be when selecting generalized coordinates from a model that contains the left and right knees. In this case, typing "kn" will likely filter out all but knee-related coordinates.

OpenSim uses regular expressions, a search pattern language, to filter out items in the list of available names. Some characters have special meaning in regular expressions and can be used to speed up the filtering. These characters are explained below using examples:

Pattern	*Filtered quantities*
^l_	Matches all names that **start** with the substring l_
X$	Matches all names **ending** in letter X
.*	Wildcard, matches any sequence of 0 or more letters

Muscle Group Filtering

You can select muscles based on their function rather than their name. This can be done by defining a muscle group within the OpenSim model (.osim) file which contains muscles that perform a higher level function. For example, some of the models distributed along with OpenSim contain a muscle group with the name _R_knee_ext_ (for right knee extensors) that contains only the muscles rect_fem_r and vas_int_r. To limit display of muscle names to this muscle group, choose the radiobutton for model, and then pick the model and the specific muscle group in the model from the drop down menus. Only muscles belonging to the selected muscle group will be shown in the list. You still need to select the muscle(s) out of the filtered list.

Selecting Items from the List

After filtering the names, you can *select* from the shown entries by marking individual boxes, selecting all shown entries using the **select all shown** checkbox, or deselect all shown entries using the **deselect all shown** checkbox.

In some contexts, the **sum only** checkbox is enabled. If this option is checked, you will add the quantity over all selected muscles. At all times, a current tally of the number of selected items is displayed in the bottom-right corner of the window.

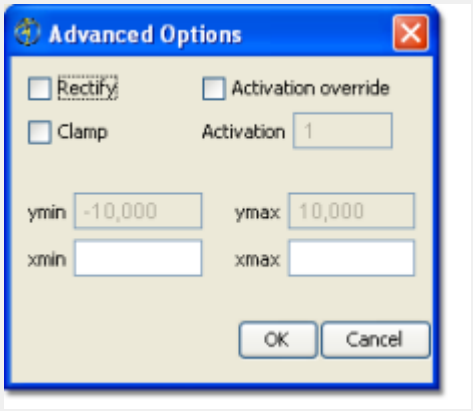
Next: Advanced Options

Previous: Exporting and Printing

Home: Plotting

Advanced Options

A few advanced options are also available through the **Advanced…** button of the *curve creation* panel. These are explained as follows:



- **Rectify**: Takes the absolute value of the quantity to be plotted
- **Clamp**: The values to be plotted are clamped to the range between the **ymin** and **ymax** values specified in the corresponding text boxes
- **xmin, xmax**: Used so the curve is not generated along the full domain values
- **Activation override:** When the **Activation override** box is checked, the activation of each muscle being plotted is set to the value in the **Activation** field. When it is not checked, the plotter uses an activation value of 1.0 for each muscle (or, when plotting data from motions, whatever activation value is specified in the motion).

Once advanced options have been set, they stay in effect until changed by the user.

Next: Overview of the OpenSim Workflow

Previous: Selection Filtering Window

Home: Plotting

Overview of the OpenSim Workflow

OpenSim has a broad range of capabilities for generating and analyzing musculoskeletal models and dynamic simulations. This chapter provides an overview of these capabilities and a list of resources to find more information about each component of the OpenSim workflow.

The OpenSim Model

One of the major goals of the OpenSim project is to provide a common platform for creating and sharing models of the musculoskeletal system. Thus the first component of any analysis is an OpenSim model. An OpenSim model represents the dynamics of a system of rigid bodies and joints that are acted upon by forces to produce motion. The OpenSim model file is made up of components corresponding to parts of the physical system. These parts include bodies, joints, forces, constraints, and controllers.

Additional information is also available in the section on OpenSim Models. A large repository of existing models is available at SimTK.org (https://simtk.org/home/nmblmodels). This library includes models of the lower extremity, head and neck, spine, and wrist. We encourage you to contribute your own models to this library to enable other researchers to build on your work and further advance the field.For example, in a model used for simulation of human walking (above), the bodies represent the geometry and inertial properties of the body segments. The joints specify the articulations at the pelvis, hip, knee, and ankle joints, while a constraint could be used, for example, to couple the motion of the patella with the model's knee flexion angle. The forces in the model include both internal forces from muscles and ligaments and external forces from interaction with the ground. Finally, the model's controller determines the activation of muscles (e.g. computed muscle control).

Importing Experimental Data

In many cases, you will use OpenSim to analyze experimental data that you have collected in your laboratory. This data typically includes:

- Marker trajectories or joint angles from motion capture
- Force data, typically ground reaction forces and moments and/or centers of pressure
- Electromyography

See Preparing Your Data for detailed information about preparing and importing your experimental data.

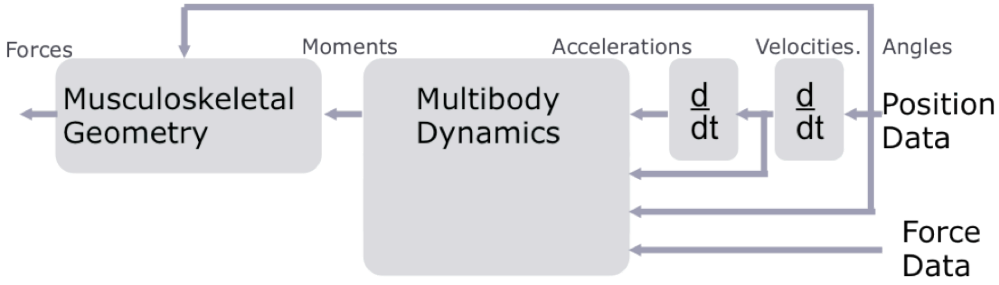
Scaling

If you are using a generic model from the existing library of models, the next step is to scale the model to match the experimental data collected for your subject, functionality provided by the Scale tool in OpenSim. The purpose of scaling a generic musculoskeletal model is to modify the anthropometry, or physical dimensions, of the generic model so that it matches the anthropometry of a particular subject. Scaling is one of the most important steps in solving inverse kinematics and inverse dynamics problems because these solutions are sensitive to the accuracy of the scaling step. In OpenSim, the scaling step adjusts both the mass properties (mass and inertia tensor), as well as the dimensions of the body segments.

See the section on Scaling for more details. Tutorial 3 - Scaling, Inverse Kinematics, and Inverse Dynamics includes an example using the Scale tool. This tutorial is also accessible from the OpenSim application Help menu.

The Inverse Problem

OpenSim enables researchers to solve the Inverse Dynamics problem, using experimental measured subject motion and forces to generate the kinematics and kinetics of a musculoskeletal model (see figure below).



In inverse dynamics, experimentally measured marker trajectories and force data are use to estimate a model's kinematics and kinetics.

Inverse Kinematics

The Inverse Kinematics (IK) Tool in OpenSim finds the set of generalized coordinates (joint angles and positions) for the model that best match the experimental kinematics recorded for a particular subject (figure below). The experimental kinematics targeted by IK can include experimental marker positions, as well as experimental generalized coordinate values (joint angles). The IK tool goes through each time step of motion and computes generalized coordinate values which positions the model in a pose that "best matches" experimental marker and coordinate values for

that time step. Mathematically, the "best match" is expressed as a weighted least squares problem, whose solution aims to minimize both marker and coordinate errors.

Experimental markers are matched by model markers throughout the motion by varying the generalized coordinates (e.g., joint angles) through time. See Inverse Kinematics for full documentation for running IK in OpenSim. Tutorial 3 - Scaling, Inverse Kinematics, and Inverse Dynamics walks through an example of using Inverse Kinematics for human walking.

Inverse Dynamics

Dynamics is the study of motion *and* the forces and moments that produce that motion. The Inverse Dynamics (ID) tool determines the generalized forces (e.g., net forces and torques) that cause a particular motion, and its results can be used to infer how muscles are utilized for that motion. To determine these internal forces and moments, the equations of motion for the system are solved with external forces (e.g., ground reactions forces) and accelerations given (estimated by differentiating angles and positions twice). The equations of motion are automatically formulated using the kinematic description and mass properties of a musculoskeletal model in Simbody™.

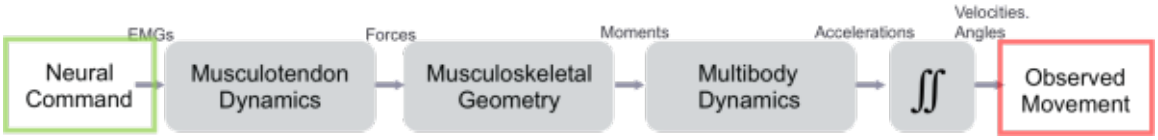
See Inverse Dynamics for full documentation for running ID in OpenSim. Tutorial 3 - Scaling, Inverse Kinematics, and Inverse Dynamics walks through an example of using ID for human walking.

Static Optimization

Static optimization is an extension of inverse dynamics that further resolves the net joint moments into individual muscle forces at each instant in time based on some performance criteria, like minimizing the sum of squared muscle forces. See Static Optimization for more details.

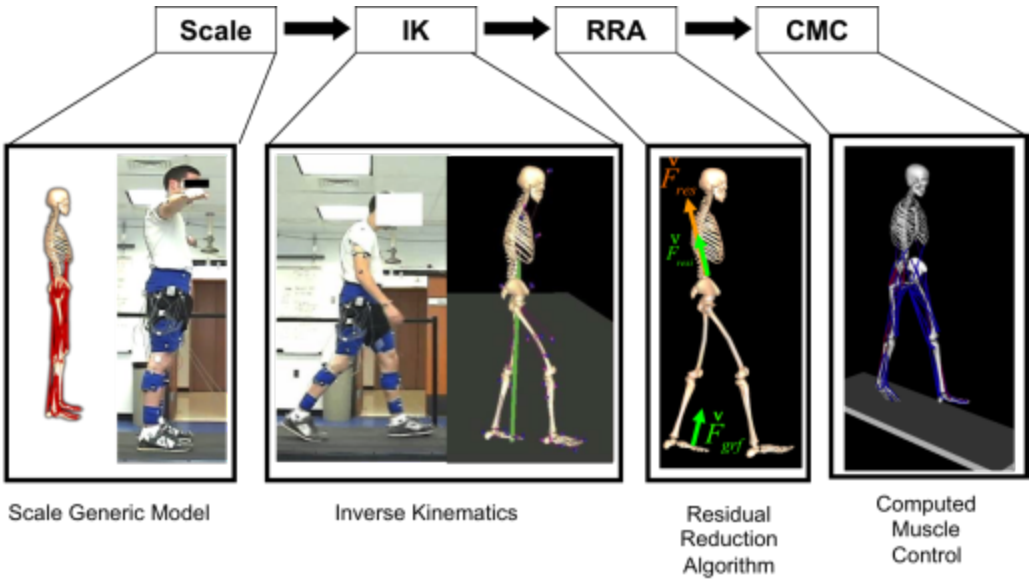
The Forward Problem

OpenSim is also capable of generating muscle-driven forward simulations of gait and other movements (figure below).



In a forward dynamic simulation of motion, simulated muscle excitations are used to drive the motion of a model to follow some observed movement.

The Forward Dynamics tool takes a set of controls (e.g., muscle excitations) to drive a model's motion by integrating forward in time. Typically, muscle excitations are generated using the Computed Muscle Control (CMC) tool. As a pre-cursor to running CMC, the Residual Reduction Algorithm (RRA) is used to minimize the effects of modeling and marker data processing errors that aggregate and lead to large nonphysical compensatory forces called residuals. Specifically, RRA alters the torso mass center of a subject-specific model and permits the kinematics of the model from inverse kinematics to vary in order to be more dynamically consistent with the ground reaction force data. Thus the typical workflow for generating a muscle-driven simulation after importing experimental data is Scale->IK->RRA->CMC->Forward Dynamics (figure below).



Full documentation of Forward Dynamics, the Residual Reduction Algorithm, and Computed Muscle Control is available in the respective sections.

Analyzing Simulations

Often, answering your research questions requires delving deeper into the details of a simulation. Thus OpenSim includes an Analyze tool that allows you to estimate, for example, muscle fiber or tendon lengths during a motion or the loads on the knee joint. The Analyze Tool enables you to analyze a model or simulation based on a number of inputs that can include time histories of model states, controls, and external loads applied to the model. The following analyses are available in OpenSim:

1. **Body Kinematics**: Reports the spatial kinematics (position and orientation, linear and angular velocity, linear and angular acceleration) of specified bodies for the duration of the analysis.
2. **Point Kinematics**: Reports the global position, velocity and acceleration of a point defined local to a body during a simulation.
3. **Muscle Analysis**: Reports all attributes of all muscles. This includes: fiber length and velocity, normalized fiber length, pennation angle, active-fiber force, passive-fiber force, tendon force, and more.
4. **Joint Reactions**: Reports joint reaction forces. These are forces that enforce the motion of the joint. The force applied to either parent or child and expressed in ground, parent or child can be reported.
5. **Induced Acceleration:** Computes accelerations caused or "induced" by individual forces acting on a model, for example, the contribution of individual muscle forces to the mass center acceleration.
6. **Force Reporter:** Reports all forces acting in the model. For ligaments and muscles, the tension along the path is reported and for ideal actuators the scalar force or torque is reported. For all other forces, the resultant body forces (force and moment acting at the center of mass of the body) are reported. For example, contact forces from an ElasticFoundationForce element yields the resultant body force on the contacting bodies separately, expressed in ground. For constraints, the same is true, except the forces are expressed in the most distal common ancestor body. Whenever a constraint involves ground, this is the ground body; however, if for example a model of the arm has a hand with fingers touching via a point constraint, then the forces are expressed in the nearest common ancestor, which would be the palm (if modeled as a single body).

More details about the analyses available in OpenSim are available in the sections Analyses. Joint Reactions Analysis, and Induced Acceleration Analysis.

Next: Scaling

Scaling

In this section, we provide a conceptual review of the inputs and outputs of the Scale tool and a set of troubleshooting tips and best practices for scaling. Carefully scaling your model to match your subject is essential for getting good results from later tools, like Inverse Kinematics and Inverse Dynamics.

Next: Getting Started with Scaling

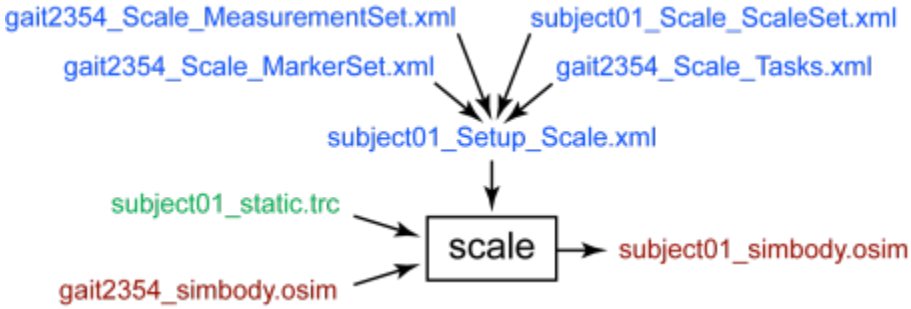
Getting Started with Scaling

The Scale Tool alters the anthropometry of a model so that it matches a particular subject as closely as possible. Scaling is typically performed based on a comparison of experimental marker data with virtual markers placed on a model. In addition to scaling a model, the scale tool can be used to adjust the locations of virtual markers so that they better match the experimental data.

The Scale Tool is accessed by selecting **Tools Scale Model…** from the OpenSim main menu bar. Like all tools, the operations performed by the Scale Tool apply to the current model.

Overview

The figure below shows the required inputs and outputs for the Scale Tool. Each is described in more detail in the following sections.



Inputs and Outputs of the Scale Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue.

> The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Settings Files

The **subject01_Setup_Scale.xml** file is the setup file for the Scale Tool. It contains settings, described in detail in How Scaling Works, and refers to other files that contain additional settings. These other files are listed below:

gait2354_Scale_MarkerSet.xml: Marker set for the Scale Tool. It contains the set of virtual markers that are placed on the body segments of the model.

gait2354_Scale_Tasks.xml: Inverse kinematics tasks for the Scale Tool. In addition to scaling the model, the Scale Tool moves the virtual markers on the model so that their positions match the experimental marker locations. To do this, the Scale Tool must position the model so that it best matches the position of the subject. This requires an inverse kinematics problem to be solved. This file contains the inverse kinematics tasks (i.e., a specification of which virtual and experimental markers should be matched up during the inverse kinematics solution) and their relative weightings. This file contains the inverse kinematics tasks describing which virtual and experimental markers should be matched up during the inverse kinematics phase. The file also contains marker weights, which are relative and determine how "well" the virtual markers track experimental markers (i.e., a larger weight will mean less error between virtual and experimental marker positions).

gait2354_Scale_MeasurementSet.xml: Measurement set for the Scale Tool. It contains pairs of experimental markers, the distance between which are used to scale the generic musculoskeletal model.

AND/OR

subject01_Scale_ScaleSet.xml: Scale set for the Scale Tool. It contains a set of manual scale factors to be applied to the generic musculoskeletal model.

Inputs

Two data files are required by the Scale Tool:

subject01_static.trc: Experimental marker trajectories for a static trial. A static trial is usually several seconds of data with the subject posed in a known static position. A segment of a regular motion file can be used as a static trial if desired, but this is not typically done. The static pose should include the subject wearing the full marker set. The marker trajectories are specified in global frame.

gait2354_simbody.osim: OpenSim musculoskeletal model. This generic model will be scaled to match the anthropometry of your subject.

You can also provide an additional, optional file:

subject01_static.mot: Experimental generalized coordinate values (joint angles) for a trial obtained from alternative motion capture devices or other specialized algorithms. You can specify coordinate weights in the Tasks file, if joint angles are know a priori. Coordinate weights are also relative and determine how "well" a joint angle will track the specified angle.

Outputs

The Scale Tool generates a single file:

subject01_simbody.osim: OpenSim musculoskeletal model scaled to the dimensions of the subject.

Best Practices and Troubleshooting

Data Collection and Other Preparation:

1. When collecting data, take pictures of your subjects in the static pose. These picture are valuable for evaluating the results of the Scale tool
2. Measure subject specifics, like height, mass, body segment lengths, mass distribution (if DXA is available), and strength (if a Biodex is available). You can use this data, along with marker positions, to best match the generic model to a specific subject.
3. Have your subjects perform movements to calculate functional joint centers at the hip, knee, ankle, and/or shoulders and append the joint centers to your static trial data (see Appending Data to a Motion).

Scale Settings:

1. Rely on markers that correspond to anatomical landmarks and functional joint centers (FJC) to position and scale the generic model.
    a. See Scale Factors Pane for information about defining the measurement set for scaling.
    b. See Scale Static Pose Weights Panel for information about setting weights when positioning the model in the static pose
2. Some segments, like the pelvis and torso, are often best scaled non-uniformly. For example, see the torse scale settings in the Scale Factors Pane.
3. Review How Scaling Works, The Control Panel, and Scale Factors Pane for more information about Scale Settings.

Evaluating your Results:

1. Scaling a model is an iterative process. Use the "preview static pose" option in the GUI. See the section on "Previewing Scale" in the Settings Pane section for more information. After running preview, perform steps 2 to 5 described below.
2. Check the messages window, which has information about the results of scaling, including the overal RMS marker error and the maximum marker error.
    a. In general, maximum marker errors for bony landmarks should be <2 cm.
    b. RMS error should typically be less than 1 cm.
    c. Pay close attention to errors in the bony landmark and FJC markers when assessing the quality of your scaling results.
3. Visualize the scaled model's anatomical marker positions relative to the corresponding experimental markers to see how well the model "fits" the data. Use the pictures you took to assess the results, comparing the joint angles in the "Coordinates" window to the angles you observe in the pictures.
    a. Do the hip, knee, and ankle angles from scale match what you observe in the picture?
    b. Are there any large mismatches between experimental and model markers? Can these mismatches be explained by examining the pictures you took?
    c. If pictures aren't available, use what you know about a typical static pose capture. For example, the ankle angle is generally less than 5º and hip flexion angle is less than 10º.
    d. Again, pay close attention to errors in the landmark and FJC markers when assessing the quality of your scaling results.
4. After examining the messages window and performing a visual comparison, adjust the virtual markers and marker weightings to improve your results:
    a. Again, avoid adjusting the positions of the landmark and FJC virtual markers to match the experimental markers.
5. Once you've adjusted the virtual marker positions and the scale settings, preview the new static pose. Re-assess your results using steps 2 to 4 above. Once you are happy with your results, hit "Run" to generate a scaled model and adjust the virtual markers on the model to match all of the experimental markers.

Troubleshooting Tips:

1. It is common to iterate through Scale and Inverse Kinematics to fine-tune segment dimensions and marker positions that yield low marker errors for the task of interest.
2. Use coordinate tasks (Static Pose Weights) to set joint angles for troublesome joints that are very sensitive to how the markers are placed (commonly the ankle joint and lumbar joint). For example if it is known that the foot is flat, an ankle angle can be provided and

then the markers adjusted in order to match the known pose.
3. If using coordinates from a motion capture system make sure that the joint/coordinate definitions match otherwise you may cause more harm than good.
4. The model has a built in assumption that the global Y axis is up. If your data doesn't fit this, then consider transforming it. You can use Previewing Motion Capture (Mocap) Data to determine the proper transform to apply.

Interpreting Your Results

To see a video on evaluating your results, please visit the Video Gallery page.

Next: How Scaling Works

Home: Scaling

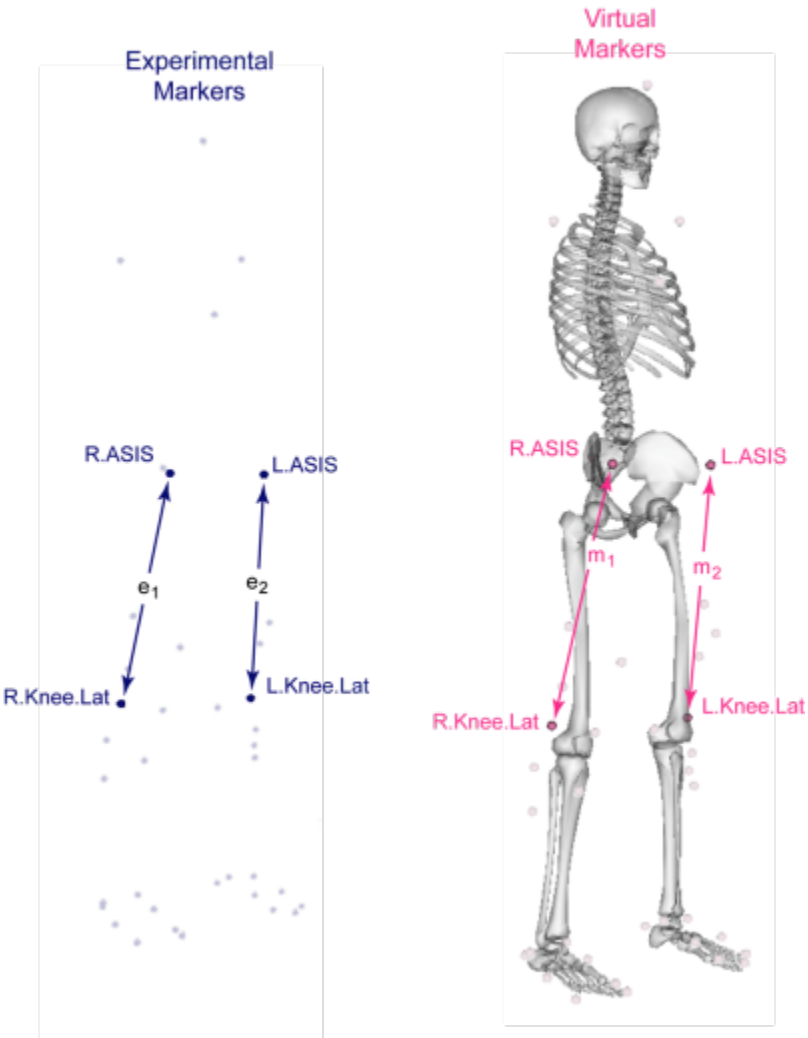
How Scaling Works

The topics covered in this section include:

Overview

Scaling is performed based on a combination of measured distances between x-y-z marker locations and manually-specified scale factors. The marker locations are usually obtained using motion capture equipment. The unscaled model has a set of virtual markers placed in the same anatomical locations as the experimental markers.

The dimensions of each segment in the model are scaled so that the distances between the virtual markers match the distances between the experimental markers. Manual scale factors, which may come from other anthropometric analyses, can also be used as an alternative to the measurement-based scaling for any body segment. Once the dimensions of the segments have been scaled to match the subject, the Scale Tool can be used to move some or all of the virtual markers on the model so that they coincide with the experimental marker locations.



Experimental and Virtual Markers. Experimental marker positions are measured with motion capture equipment (dark blue). Virtual markers are placed on a model in anatomical correspondence (e.g., over the right anterior superior iliac spine (R.ASIS)). Distances between experimental markers (e_i) relative to the distances between corresponding virtual markers (m_i) are used to compute scale factors.

Scaling

The scaling step scales both the mass properties (mass and inertia tensor), as well as the dimensions of the body segments. The main task involved in this step is computing the scale factors for each body segment. This is accomplished using a combination of measurement-based and

manual scaling.

Measurement-based Scaling

In measurement-based scaling, scale factors are determined by comparing distances between markers on the model and experimental marker positions provided in a .trc file (see Marker (.trc) Files for more information about .trc files). A single scale factor is computed using one or more marker pairs.

For example, suppose two marker pairs are used: p_1={R.ASIS, R.Knee.Lat} and p_2={L.ASIS, L.Knee.Lat}. The distance for pair 1 on the model (m_1) is computed by placing the model in its default configuration (all joint angles get assigned their default values, as specified in the <default_value> property of <Coordinate> in the OpenSim model (.osim) file). The experimental distance between pair 1 (e_1) is computed by looking at each frame of experimental marker data in the given .trc file, computing the distance between the pair for that frame, and taking the average across all frames in a user-specified time range. The scale factor due to pair 1 is then $s_1=e_1/m_1$. If the markers were further apart in the .trc file than on the model, this indicates that the segment(s) in the model supporting these markers are too small. The overall scale factor is then the average of the scale factors computed due to all of the pairs (e.g., $s=(s_1+s_2)/2$ in this case, where s_2 is the scale factor due to pair 2). This overall scale factor s can then be used to scale any segments, and along any combination of the X, Y, and Z axes.

Manual Scaling

As an alternative to computing scale factors using measured marker positions, it is possible to specify the x-y-z scale factors for a segment manually. This is useful if the actual scale factors for segments are known, or were computed using some alternative algorithm.

Mass Scaling

The scale factors computed using measurement-based or manual scaling are used to scale the sizes of the segments. In addition, the masses of the segments are adjusted so that the total mass of the body equals the specified subject mass.

There are two different ways the individual segment masses may be adjusted. One approach is to *preserve the mass distribution,* which ensures that the masses of the subject-specific model segments are in the same proportion as they were in the generic model. This scales the masses using a constant factor independent of the scale factors that were used to scale the individual segment sizes. The alternative approach incorporates the size scale factors, still ensuring the total mass equals the subject mass, but having the mass of the scaled segments reflect their scale in size.

In any case, the inertia tensor of each segment is updated to reflect its new size and mass.

Marker Placement

After scaling the model, the next step is to move the model's markers to match experimental marker locations in a static pose. The static pose is computed by trying to match some combination of experimental marker positions and generalized coordinate values, as in the inverse kinematics (IK) step (How Inverse Kinematics Works). The marker locations corresponding to the static pose are computed by averaging the marker positions in a given .trc file across a user-specified time range. Just like IK, marker and coordinate weights are used to determine how strongly the algorithm should try to match them. Once a static pose is computed using the IK-based algorithm, all model markers (except for those designated as fixed) are moved to the averaged "static pose" positions of the experimental markers.

Next: How to Use the Scale Tool

Previous: Getting Started with Scaling

Home: Scaling

How to Use the Scale Tool

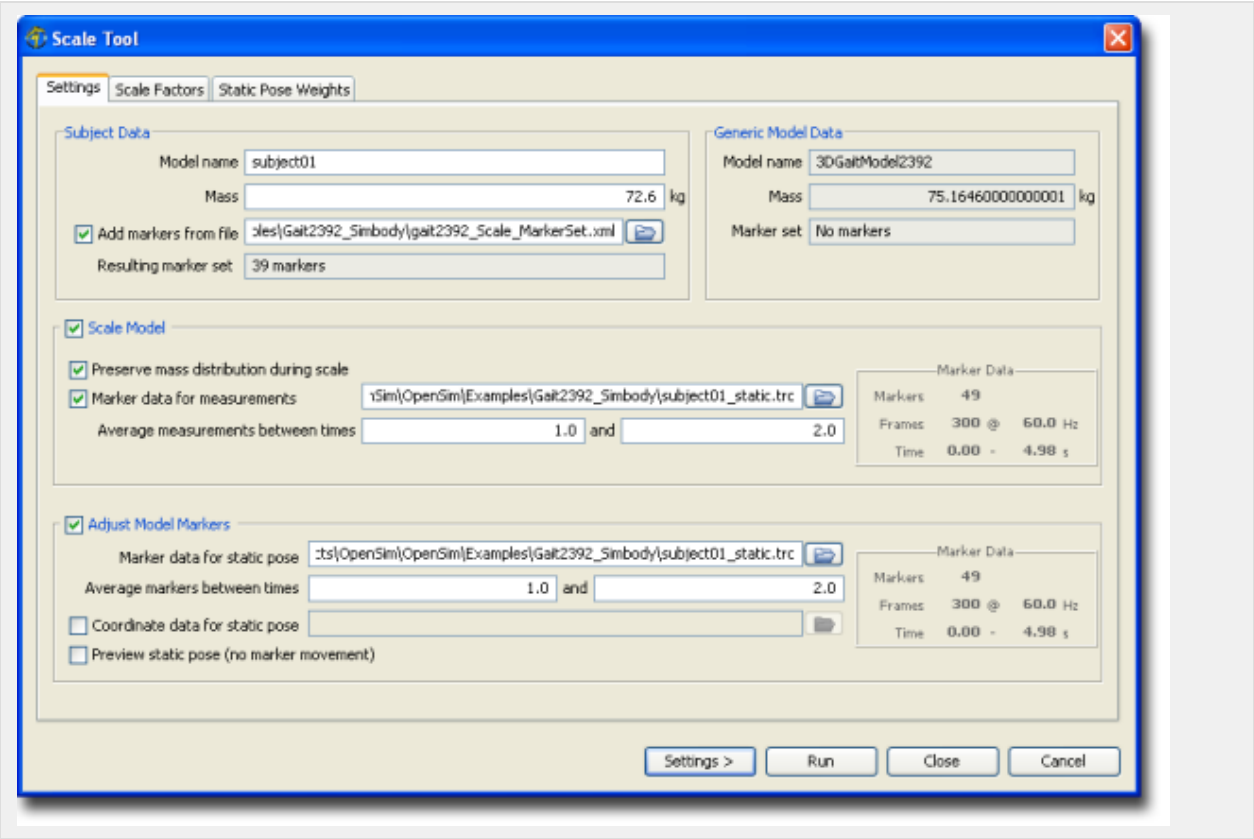
The topics covered in this section include:

- How to Use the GUI
- Command-line Execution
- Videos

How to Use the GUI

The Scale Tool is accessed by:

- Select **Tools Scale Model...** from the OpenSim main menu bar.
- You need the model you want to be scale to be current. To learn how to make a model current visit Opening, Closing, and Using the Navigator Window.



The Scale Tool is controlled by a window with three tabbed panes. The Settings Pane is used to specify parameters relating to the subject data, the generic model, and how the model is to be scaled. The Scale Factors Pane is used to specify the scale factors for each segment. The Scale Static Pose Weights Panel pane is used to specify weights on marker positions and joint angles for solving an inverse kinematics problem for the static pose. The inverse kinematics solution for the static pose is used to place the virtual markers on the model so that they coincide with the measured marker locations on the subject.

Command-line Execution

The Scaling Tool can also be run using the command scale -S <setup file name>, for example:

```
scale -S subject01_Setup_Scale.xml
```

Videos

To watch a video on how to set up the scale tool, please visit the Video Gallery page.

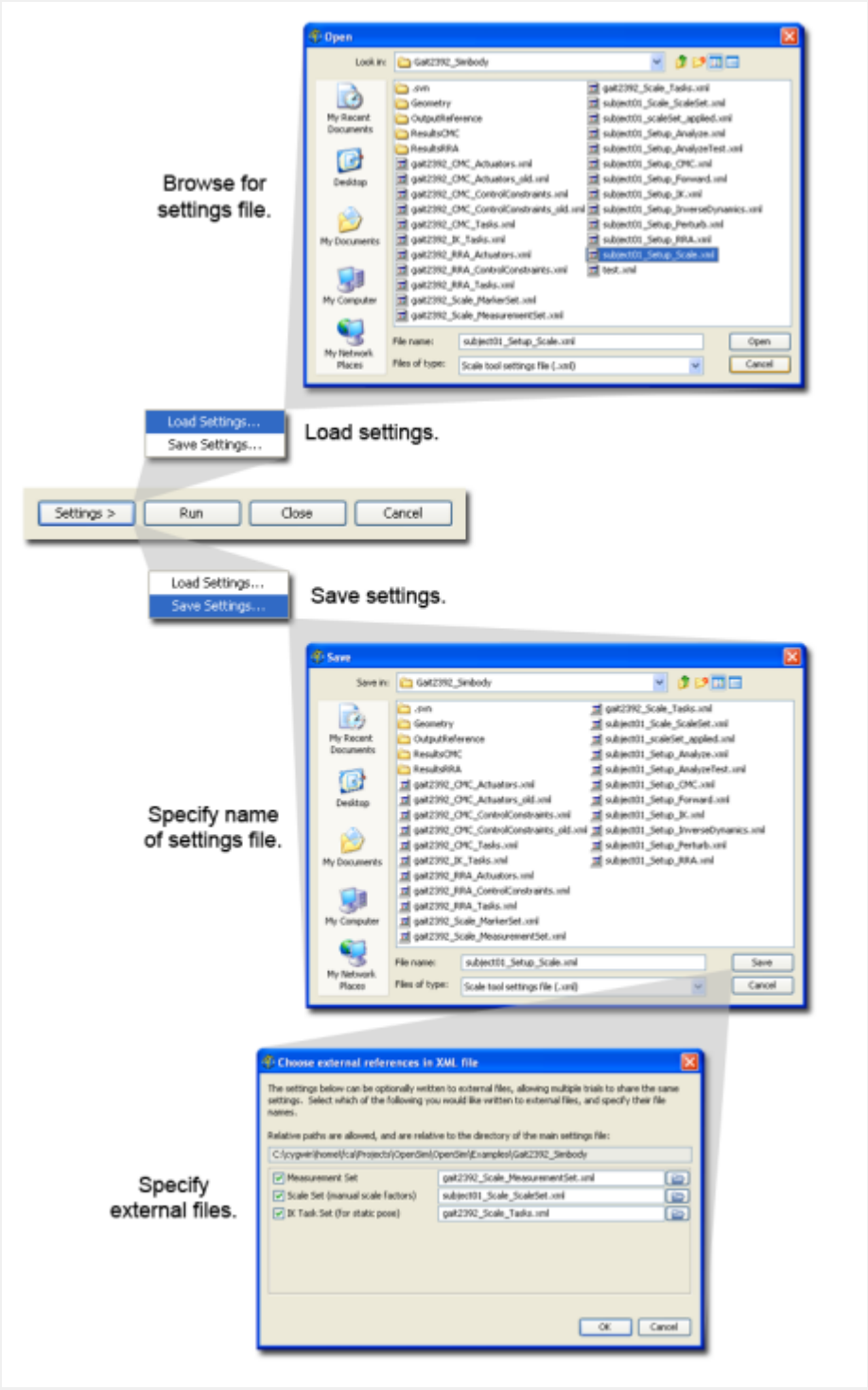
Next: The Control Panel

Previous: How Scaling Works

Home: Scaling

The Control Panel

At the bottom of all the Tool dialog windows are four buttons, located in what we call the *Control Panel*.



- The **Settings >** button is used to load or save settings for the tool.
- The **Run** button starts execution.
- The **Close** button closes the window.

⚠ The **Close** button can be clicked immediately after execution has begun; the execution will complete even though the window has been closed. The **Cancel** button closes the window and cancels all operations that have not yet been completed.

- When the **Settings >** button is clicked, you are presented with the choice of loading or saving settings for the tool. This feature can be very useful as most tools require many parameters to be set before they can be run.
- If you click **Load Settings…**, you will be presented with a file browser that displays all files ending with the **.xml** suffix. You may browse for an appropriate settings file (e.g., **subject01_Settings_Scale.xml**) and click **Open**. The tool (Scale Tool, in this case) will then be populated with the settings in that setup file.
- If you have manually entered or modified settings, you may save those settings to a file for future use. If you click **Save Settings…** , a Save dialog box will come up in which you can specify the name of the settings file. The name you specify for the file should have a suffix of **.xml**. Click **Save** to save the settings to the file. After you click **Save**, you may be presented with another dialog box that asks you whether or not you would like to save some of the settings to separate external files. This can be useful if you would like to reuse those settings for other trials or subjects. Check the boxes of the settings that you'd like to save to external files and specify the names of these files. All of these files should have a suffix of **.xml**.

Next: Settings Pane

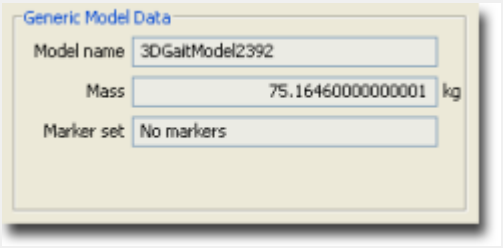
Previous: How to Use the Scale Tool

Home: Scaling

Settings Pane

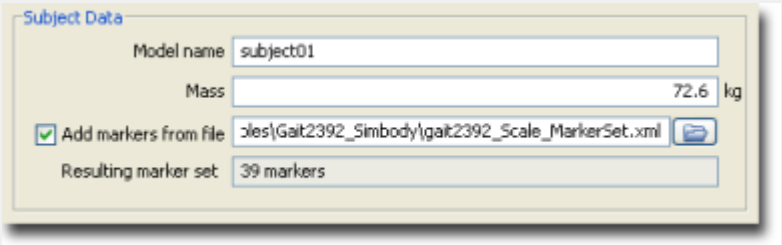
The *Settings* pane is used to specify parameters related to the subject data, the generic model, and how the model is to be scaled. The pane is organized into four main sections:

Generic Model Data



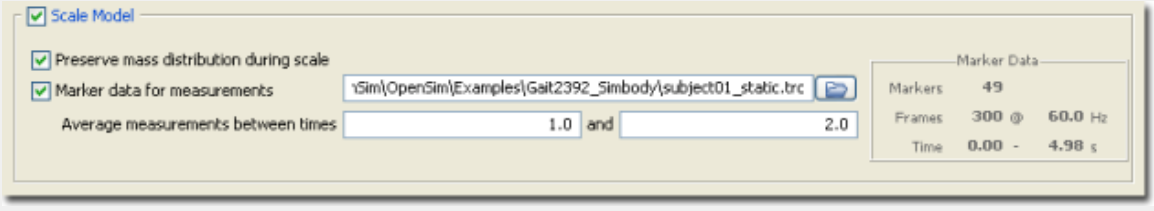
- The section for *Generic Model Data* displays uneditable information about the generic model that is to be scaled. It gives the model name, the model mass, and whether or not a marker set is included as part of the model).

Subject Data Section



- The section for *Subject Data* displays editable information that allows you to specify the name of the new model that will be generated, the subject's total mass, and whether or not to add markers to the model from an external marker set.
- If you select the check box next to **Add markers from file**, you will be required to specify a settings file that contains markers.
- You may use the button to browse for a marker set. Once a marker set is read in, the number of markers in the marker set will be displayed just below the file name.

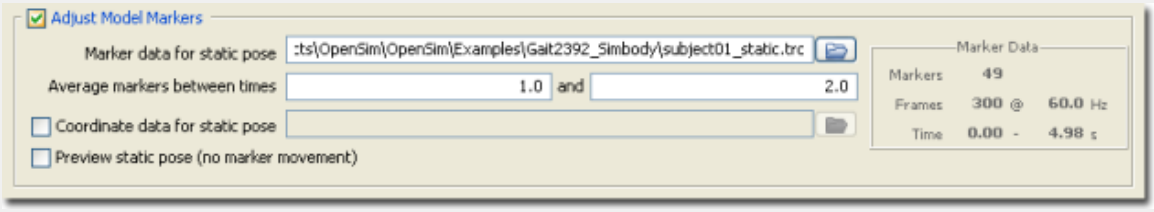
Controlling Model Scaling



- The *Scale Model* section provides access to basic settings for specifying how a model is scaled.
- If the **Scale Model box** in the upper left-hand corner is not checked, the model will not be scaled.
- If the box is checked, the model will be scaled and a number of options become available.
- When the box for **Preserve mass distribution during scale** is checked, the total mass of the generic model is scaled so that it equals the mass of the subject while preserving the relative masses of its body segments. So, for example, if in the generic model the mass of the thigh is twice that of the shank, the mass of the thigh in the scaled model will also be twice that of the scaled shank. If this check box is not checked, the segment masses are scaled based solely on the scale factors applied to each body segment.

To use measurement-based scaling, a .trc file that contains suitable marker data must be specified. A static trial (i.e., a trial in which the subject is stationary in some known position) is typically used for this purpose. To specify a file containing marker data, check the box labeled **Marker data for measurements**. When this is done, you will be able to specify a file in the text box to the right. Click the  button to browse for a file. Once the marker data is loaded, information about the trial will be displayed in a box on the far right entitled **Marker Data**. The information includes the total number of markers in the .trc file (the marker data is arranged in columns of x, y, and z coordinates), number of frames, frame rate, and the time interval over which the data is available. The distances between marker pairs used to compute the scale factors for the segments is based on averaging the marker positions over a time interval. To specify the time interval, enter the starting and final times in the text boxes to the right of the label **Average measurements between times**. The starting time should always be less than or equal to the final time. If you are computing scale factors based on a dynamic trial (i.e., a trial in which the subject was moving), use a small time interval or specify the same starting and final time to pick out a single frame of data.

Adjusting Model Markers



- The *Adjust Model Markers* section provides access to basic settings for specifying how markers on the model should be moved to match the experimental locations.
- If the **Adjust Model Markers box** is not check, the model markers will not be moved.
- If the box is checked, a number of options become available. The text box labeled **Marker data for static pose** is used to specify the .trc file containing the experimental marker locations. Typically, this file is the same file used to scale the model, but this is not necessary.
- The name of the file can be typed in the box, or you can click the  button to browse for the file.
- Once the marker data is loaded, information about the trial will be displayed in a box on the far right entitled **Marker Data**. The information includes the total number of markers in the .trc file (the marker data is arranged in columns of x, y, and z coordinates), number of frames, frame rate, and the time interval over which the data is available.

Before the virtual markers on the model can be moved, the model must be put into a position that closely matches the position of the subject. This includes placing the model in the correct location in the laboratory (three translations and three rotations) and also finding an appropriate set of joint angles to match the pose of the subject. To do this, the locations of all the experimental markers are averaged over a specified time interval, and an inverse kinematics problem is solved to find the joint angles that minimize the position error between the model markers and the corresponding average experimental markers. To specify the time interval over which the experimental marker locations are averaged, enter the starting and final times in the text boxes to the right of the label **Average markers between times**. The starting time should always be less than or equal to the final time. If you are using a dynamic trial (i.e., a trial in which the subject was moving), use a small time interval or specify the same starting and final time to pick out a single frame of data.

Unlike the settings controlling the scaling of the model, the settings for adjusting the model markers also allow you to specify coordinate data (e.g., angle joints in the model). This data can be used to control or influence the inverse kinematics solution. For example, if you already had joint angles for the static pose provided by the motion capture software that was used, you could specify a .mot file and use the joint angles contained in the motion file to place the model in an appropriate pose for adjusting the marker positions. To specify a coordinate data file, select the box labeled **Coordinate data for static pose**. Once this box is checked, you can type the name of a .mot file in the text box directly to the right or use the  button to browse for one.

Previewing Scale Solution and Static Pose

Frequently, it is helpful to preview the inverse kinematic solution for the static pose before adjusting the positions of the model markers. This can be useful for identifying markers that are poorly placed on the model that you may need to manually adjust or may decide not to include in the inverse kinematic problem. Checking the box labeled **Preview static pose (no marker movement)** will display the inverse kinematic solution in a 3D View without any adjustments made to the original model marker locations.

Once all the settings have been made on the **Settings** pane, use the **Scale Factors** pane to control exactly how each segment in the model is scaled.

Next: Scale Factors Pane

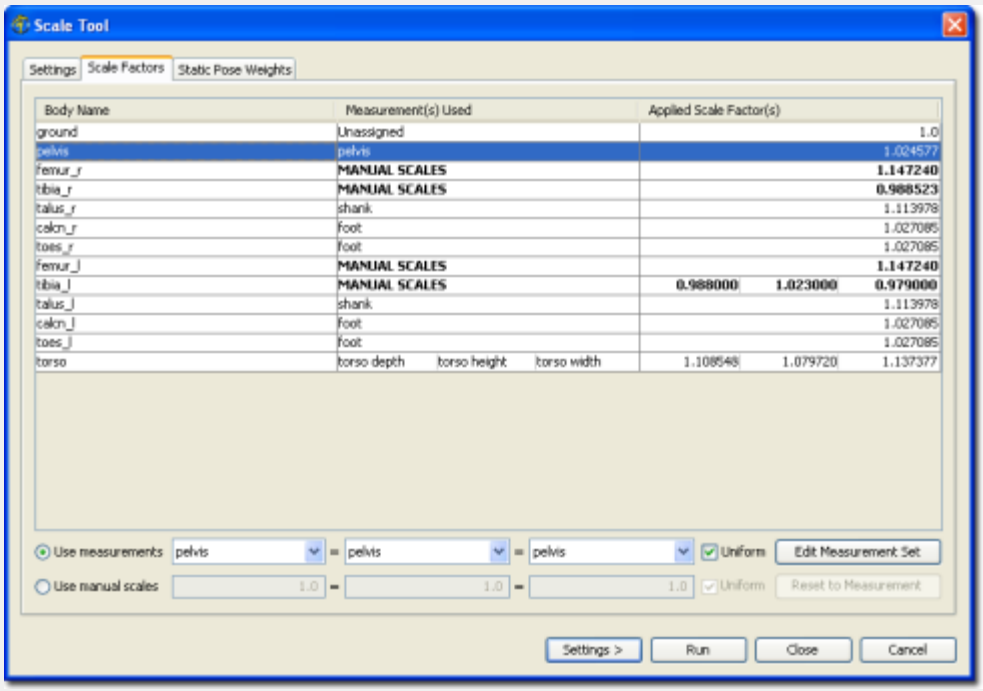
Previous: The Control Panel

Home: Scaling

Scale Factors Pane

The topics covered in this section include:

Overview

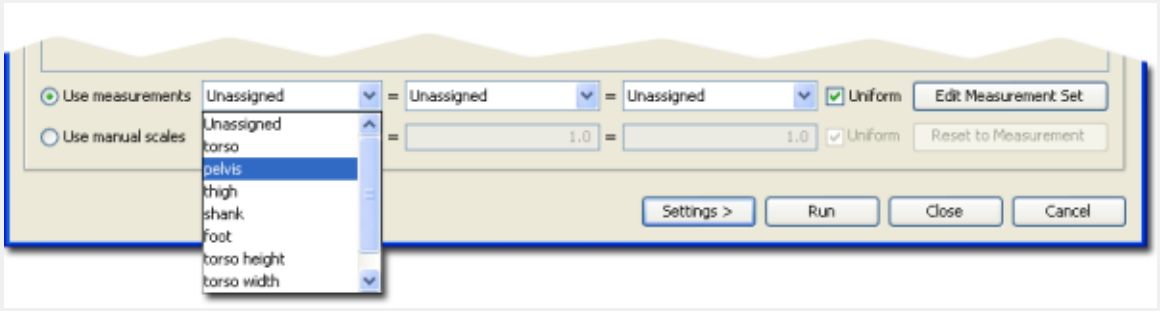


- The *Scale Factors* pane is used to specify the factors by which each segment in the model will be scaled.
- Settings for each body are listed in the table in the upper portion of the pane.
- Each row contains the name of the body, the measurement(s) used for that body, and the applied scale factor(s).
- The measurement(s) consist of either a label declaring MANUAL SCALES, indicating that the scale factors were specified manually, or a list of measurements (as shown for the femur_r segment).
- A measurement is a list of marker pairs; the distances between these markers are used to compute the scale factors applied to the segment in question. The measurement set is edited by clicking on the **Edit Measurement Set** button.
- Each segment can be scaled uniformly by a single scale factor (one scale factor is displayed) or non-uniformly by three independent scale factors along the X, Y, and Z axes of the body segment (three scale factors are displayed).

The controls at the bottom of the pane are used to specify the type of scaling and the scale factors for the selected segment. When a body is selected, such as the pelvis, the settings for that segment become editable. You can select one of two radio buttons to **Use measurements** or **Use manual scales**. To the right of each of the radio buttons are three text fields that are used to specify the scale factors that will be applied to the segment along the X, Y, and Z axes of the body. A typical choice for reference frames is X perpendicular to the frontal plane pointing forward, Y perpendicular to the transverse plane pointing up, and Z perpendicular to the sagittal plane pointing to the right. The choice of reference frames depends on how the model was constructed and may vary from model to model.

Using Measurement-Based Scaling

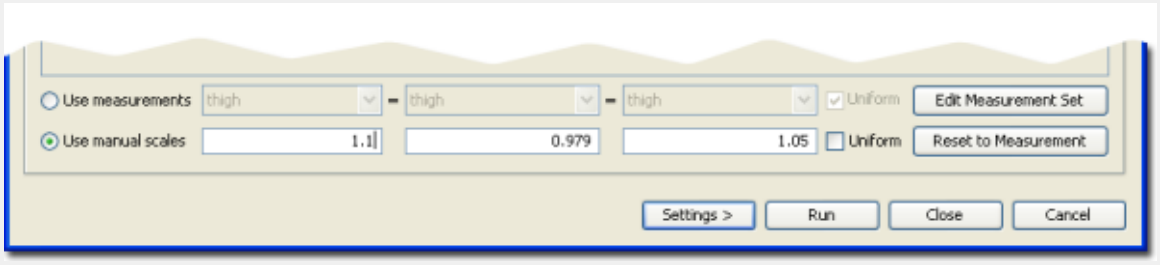
To specify scale factors based on measurement:



1. Click the **Use measurements** radio button.
2. Click in any of the three scale-factor fields to the right.
3. Select a measurement for the scale factors that will be applied along the X, Y, and Z axes of the selected body.
4. Clicking in any of these fields brings up a list of defined measurements (torso, pelvis, thigh, etc). If the **Uniform** check box is checked, equal signs appear between the scale-factor fields, and any entry made in one field will be set for the others
5. Use the **Edit Measurement Set** button to inspect and edit the measurements.

Using Manual Scale Factors

To specify scale factors manually:



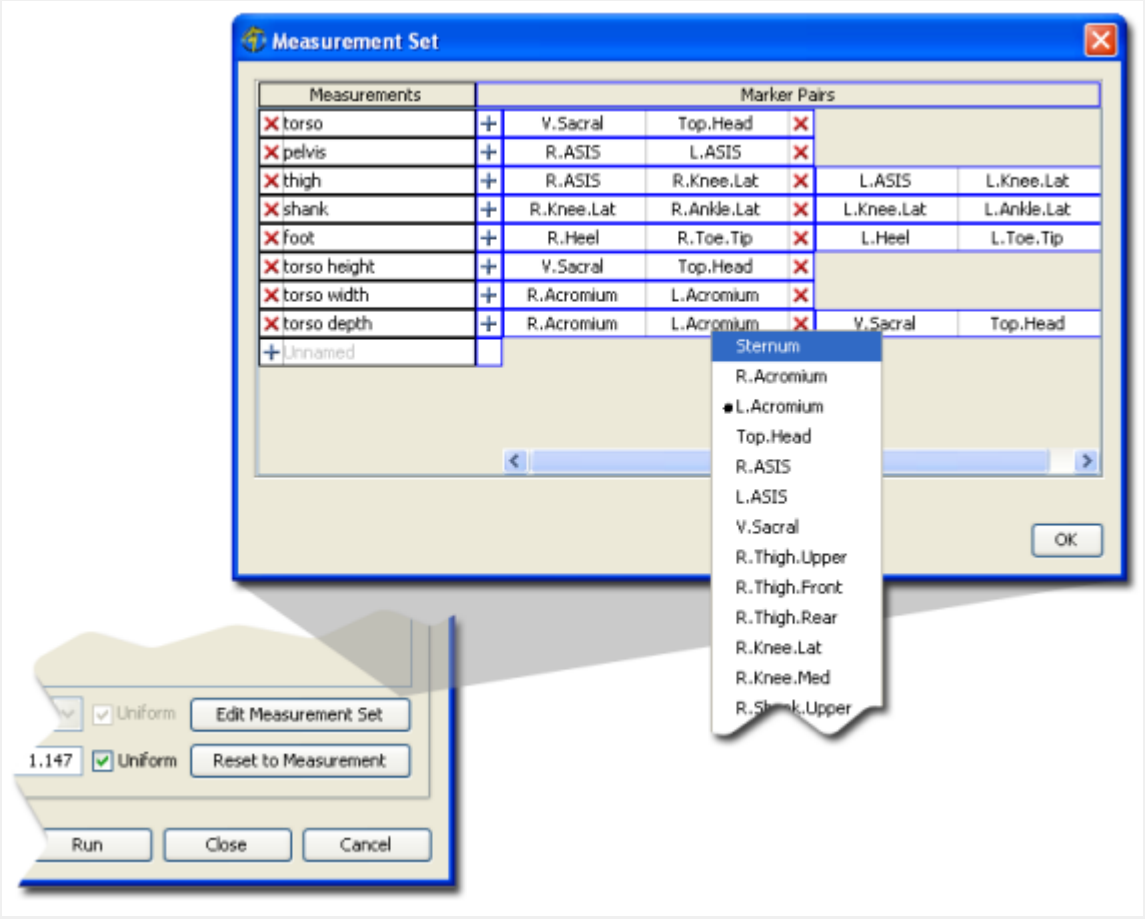
1. click the **Use manual scales** radio button
2. Type in the three fields to the right to specify the factors that will be applied along the X, Y, and Z axes of the selected body.
3. If the **Uniform** check box is unchecked, the three factors can be different from each other. Clicking the **Reset to Measurement** button will compute scale factors based on any measurement that was previously set for that segment.

For each segment, you may choose to apply one uniform scale factor, or three different scale factors along the X, Y, and Z axes of the segment. The checkboxes labeled **Uniform** to the right of the three scale-factor fields are used to choose between uniform and non-uniform scaling. When checked, equal signs appear between the three scale-factor fields, and any value specified in one field will be propagated to the other two. When a box is not checked, you can specify the measurement or scale factors for the X, Y, and Z directions independently.

The **Reset to Measurement** button is used in conjunction with manual scale factors. As a convenience, clicking the **Reset to Measurement** button will compute scale factors based on any measurements that were previously set for a segment, giving a starting place for modifying the manual scale factors.

Editing the Measurement Set

Clicking the **Edit Measurement Set** button of the Scale Factors pane brings up an editor that allows you to inspect and edit the definitions of the measurements that can be used to compute scale factors for body segments. The name of each measurement is displayed in the far left-hand column, and the marker pairs that make up a measurement are listed to the right.



- Definitions for the measurements used to scale the segments can be inspected and edited by clicking the **Edit Measurement Set** button . Doing so brings up another table that contains, in each row, the name of a measurement and the list of marker pairs that make up that measurement.
- The distances between the experimental marker pairs, relative to the distances between the same markers on the model, are used to compute the scale factors (How Scaling Works). To edit which markers make up a pair, click on one of the markers. A list of available markers will pop up, and you can then select from the list.
- Use the ✕ buttons to delete measurements or marker pairs, and the + buttons to add new measurements or marker pairs. Click the OK button in the lower right or the ✕ button in the upper right-hand of the window when you are finished inspecting or editing the measurements. Any changes you made will be saved; the only way to undo any changes you made is to undo them manually. Once a measurement has been added to the set, it will appear among the choices you have for computing the scale factors.

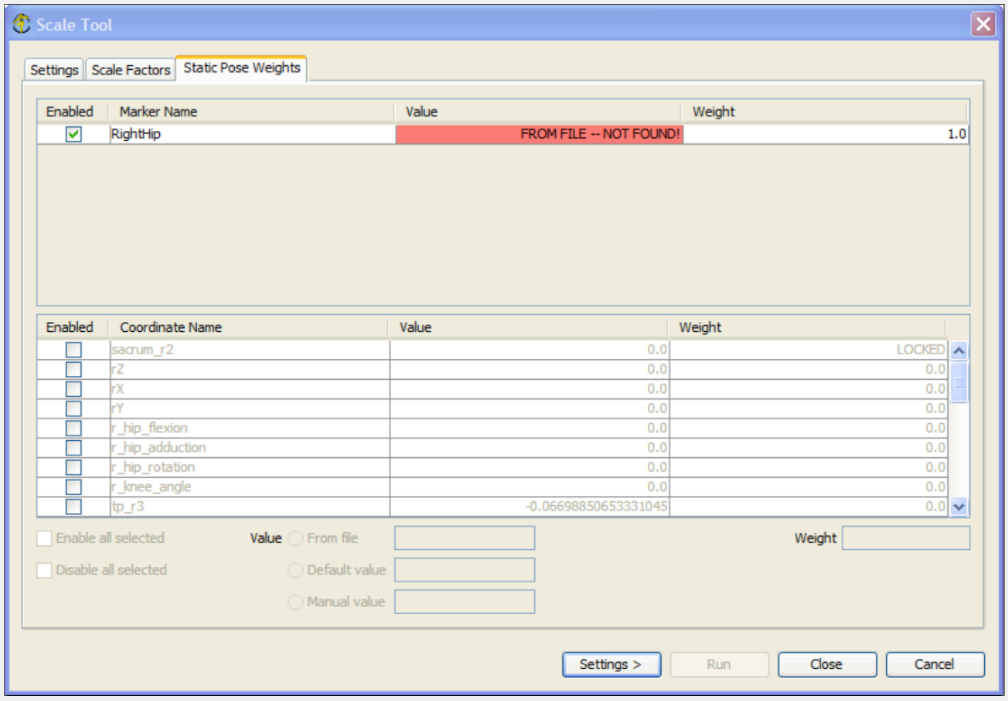
Next: Scale Static Pose Weights Panel

Previous: Settings Pane

Home: Scaling

Scale Static Pose Weights Panel

Once a marker and possibly a coordinate file have been specified, the specific behavior of the scaling tool can be described and modified using the *Weights* pane (Figure below). Each entry in the table represents a task in the least-squares matching for either a marker (top table) or a coordinate (lower table). By **left-clicking** on a row, you select it, making the entry fields below the tables editable so you can specify weights and values for the selected marker(s) or coordinate(s). You can select multiple rows to edit by using **ctrl + left-mouse-click** or **shift + left-mouse-click**.



- The weight value affects to what degree a match should be satisfied.
- Larger weightings penalize errors for that marker or coordinate more heavily and thus should match the experimental value more closely (see How Scaling Works).
- For coordinates, the coordinate value to be matched can come from a specified motion file, be set to its default value, or be set to a user-specified (*manual*) value.

Next: Scale Setup File

Previous: Scale Factors Pane

Home: Scaling

Scale Setup File

The topics covered in this section include:

There are 5 major sections to a scale setup file: execution parameters; subject-specific parameters (e.g., mass, height, age); parameters related to the generic model to scale; scaling properties; and marker placement properties. A sample scale setup file is provided below.

Example: XML file for a scale setup file

subject01_Setup_Scale.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSimDocument version="20000">
<ScaleTool name="subject01">

   <mass> 72.6 </mass>
   <height> 1803.4 </height>
   <age> 99 </age>
   <notes> This is an example setup file for scale.exe. </notes>

<GenericModelMaker name="">
   <model_file> gait2354.osim </model_file>
   <MarkerSet file="gait2354_Scale_MarkerSet.xml"/>
</GenericModelMaker>

<ModelScaler name="">
   <scaling_order> measurements manualScale </scaling_order>

   <ScaleSet file="subject01_Scale_ScaleSet.xml"/>

   <MeasurementSet file="gait2354_Scale_MeasurementSet.xml"/>
   <marker_file> subject01_static.trc </marker_file>
   <time_range> 1 2 </time_range>

   <preserve_mass_distribution> true </preserve_mass_distribution>

   <output_joint_file> subject01_scaledOnly.jnt </output_joint_file>
   <output_muscle_file> subject01_scaledOnly.msl </output_muscle_file>
   <output_model_file> subject01_scaledOnly.osim </output_model_file>
   <output_scale_file> subject01_scaleSet_applied.xml</output_scale_file>
</ModelScaler>

<MarkerPlacer name="">
   <IKTaskSet file="gait2354_Scale_Tasks.xml"/>
   <marker_file> subject01_static.trc </marker_file>
   <time_range> 1 2 </time_range>
   <coordinate_file> gait2354_zeros.mot </coordinate_file>
   <output_joint_file> subject01.jnt </output_joint_file>
   <output_muscle_file> subject01.msl </output_muscle_file>
   <output_model_file> subject01.osim </output_model_file>
   <output_motion_file> subject01_static_output.mot </output_motion_file>
</MarkerPlacer>

</ScaleTool>
</OpenSimDocument>
```

Specifying Execution Parameters

The properties for the scale operation are enclosed inside the opening and closing tags <ScaleTool> and </ScaleTool>. The name attribute name ="subject01" specifies the execution name, though it is not used anywhere.

Specifying Subject-specific Parameters

Subject specific measurements are specified in the <mass>, <height>, and <age> properties. These are in kg, mm, and years, respectively. Currently, <height> and <age> are specified for informational purposes only, and do not affect the scaling result. (In fact, they do not appear in the subject-specific model output by scale). The <mass> property, however, is used: the final model will have a total mass equal to this specified mass value.

<notes> Tag

The <notes> property is another purely informational (for the user) property that does not get propagated to any output files.

Specifying a Generic Model to Scale

The generic model that will be scaled is specified within the <GenericModelMaker> tag. In particular, <model_file> points to the OpenSim (.osim) model file that will be scaled. In Example 14-1 above, it is **gait2354.osim** (a gait model with 23 degrees of freedom and 54 muscles).

The <MarkerSet> property can be used to add or update markers in the generic model. In the example, **gait2354.osim** does not include any markers, so the <MarkerSet> given in the external file **gait2354_Scale_MarkerSet.xml** (referred to using the **file** attribute) adds the full marker set necessary for this example. This file is described in more detail in the <marker file> tag section below.

Specifying Scaling Properties

Scaling properties are enclosed in the <ModelScaler> tag. As described in How Scaling Works, a combination of measurement-based and manual scaling can be used. <scaling_order> specifies which of the two is used (using the keywords **measurements** and **manualScale**, respectively), or, if both are listed, the order in which they are applied.

Properties for manual scaling

The <ScaleSet> tag is used to specify the scale set supplying the manual scale factors. In the example above, this tag refers to the external file **subject01_Scale_ScaleSet.xml** (using the **file** attribute). See Scaling Settings Files and XML Tag Definitions for more details about the contents of this XML file.

Properties for measurement-based scaling

The relevant parameters for measurement-based scaling are <MeasurementSet>, <marker_file>, and <time_range>. <MeasurementSet> (in this case specified in the external file **gait2354_Scale_MeasurementSet.xml** using the **file** attribute) specifies which marker pairs are used to compute the scales and which bodies those scale factors are applied to. In addition, the experimental marker positions need to be supplied. The <marker_file> property points to a .trc file containing these marker positions, and <time_range> indicates which subset of the frames in the .trc file the marker-pair distances should be averaged across.

Mass scaling properties

The <preserve_mass_distribution> property specifies whether the scaled model's segment masses should be in the same proportion as they were in the generic model (see How Scaling Works).

Specifying output of scaling-only step (before marker placement)

The scaling step can (optionally) produce a number of output files. Assuming the marker placement step is executed, none of the intermediary output files from the scaling step are strictly necessary, but can be useful for debugging (e.g., to see what the model looks like after scaling but before markers are moved). The properties controlling the output file names are:

- <output_model_file> generates the scaled-only OpenSim model file
- <output_scale_file> generates a <ScaleSet> file consisting of the x-y-z scale factors used to scale each segment

If the tags specifying these files are omitted or the file names are blank, then no files will be written.

Specifying Marker Placement Properties

The marker placement properties are enclosed within the <MarkerPlacer> tag. Note that it is valid to omit this section from the XML file if only scaling (and no marker placement) is desired.

As mentioned in How Scaling Works, a "static pose" needs to be defined and the model needs to be placed in a configuration matching that pose as best as possible. This reduces to an inverse kinematics (IK) problem, and so many of the properties used in the marker placement step are similar to the properties in IK. The reader should therefore refer to the IK chapter (Inverse Kinematics) for additional details on some of these properties.

The <IKTaskSet> property is used to specify the marker and coordinate weights which IK will use in order to compute the static pose. In the example above, a **file** attribute is used to refer to an external XML file. More details of the file contents are given below.

The <marker_file> and <coordinate_file> properties are used to specify files containing experimental marker and coordinate values, respectively. These values are used by the IK tool to compute the static pose.

<marker_file> tag

Since the file specified by the <marker_file> tag will in general have multiple frames of marker data (i.e., marker trajectories recorded during a static trial), these are averaged within the time range specified by the <time_range> tags to give a single, fixed set of experimental marker positions. If the experimental marker locations in <marker_file> do not represent a static trial (where the subject is not moving or only slightly moving) then averaging marker positions across a large time window does not really make sense. In this case, the values for <time_range> should be reduced and/or the file specified by <marker_file> should be edited to only contain a few rows of nearly-stationary marker positions.

<coordinate_file> tag

The coordinate values specified by <coordinate_file> are not averaged the same way the marker data is. Rather, the values from the row corresponding to the initial time of <time_range> are used. In the example XML file for a scale setup file, the coordinate values from time t=1 will be used as the "experimental coordinate values."

Moving markers to match experimental locations

The IK solver will be invoked to try to find generalized coordinate values for the model which put it in a pose that closely matches the "static pose" experimental marker positions and coordinate values. After the model's static pose is computed, a selected subset of the model's markers are moved to match the averaged experimental marker locations (those that were used to define the static pose).

Only markers which are tagged as "not fixed" are moved to match the experimental markers. This is done within the file specified by <MarkerSet>. Each marker that is moveable should have its <fixed> property set to **false.** <fixed> is a property of <Marker> (see Scale Marker File). The remaining markers are untouched.

Specifying final scaling and marker placement output

After marker placement, the final subject-specific model is ready for output. The only essential file to be output in this step is the subject-specific OpenSim model, specified in <output_model_file>. Additional files can be output:

- <output_motion_file> - a motion file consisting of a single row of values representing the "static pose" in which model markers were moved to match experimental markers. This file can be visualized in SIMM. Note: since this motion file only contains a single frame, to view it in SIMM's Model Viewer, you need to use the "start >" button to start playing the motion; the motion slider in SIMM does not appear to work.

Next: Scale Marker File

Previous: Scale Static Pose Weights Panel

Home: Scaling

Scale Marker File

The topics covered in this section include:

- Specifying a List of Markers
- Specifying a Marker for the Generic Model

The scale marker file contains a list of the virtual markers that are placed on the body segments of the model. An example of a marker file is shown below.

Example: XML file for a scale marker file

gait2354_Scale_MarkerSet.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<MarkerSet name="gait2354_Scale">

   <objects>

      <Marker name="Sternum">
         <location> 0.07 0.3 0 </location>
         <body> torso </body>
         <fixed> false </fixed>
      </Marker>

      <Marker name="R.Acromium">
         <location> -0.03 0.44 0.15 </location>
         <body> torso </body>
         <fixed> false </fixed>
      </Marker>

      <Marker name="L.Acromium">
         <location> -0.03 0.44 -0.15 </location>
         <body> torso </body>
         <fixed> false </fixed>
      </Marker>

      <Marker name="Top.Head">
         <location>0.00084 0.657 0.0</location>
         <body> torso </body>
         <fixed> false </fixed>
      </Marker>

<!– . . additional <Marker> tags cut for brevity . . –>

   </objects>

</MarkerSet>
```

Specifying a List of Markers

The list of markers are enclosed inside the opening and closing tags <MarkerSet> and </MarkerSet>.

Specifying a Marker for the Generic Model

Specifying a marker consists of specifying its <location> in Cartesian coordinates, as well as the <body> to which the marker is attached (i.e., which body its location is measured with respect to). The marker name is given by the **name** attribute of the <Marker> tag (e.g., **Sternum** for the first marker in the example above).

The <fixed> property is used in the marker placement step and can be set to either **true** or **false**. If it is set to **false**, the marker will move during marker placement to match the position of its corresponding experimental marker. Otherwise it will stay fixed.

Next: Manual Scaling File

Previous: Scale Setup File

Home: Scaling

Manual Scaling File

The manual scale factors can be set using the <ScaleSet> and </ScaleSet> tags. <ScaleSet> is a set consisting of <Scale> tags, each of which gives manual scale factors as described below. An example of a manual scaling file is shown below:

- <scales> Tag
- Tag
- <apply> Tag

<scales> Tag

Each <Scale> tag specifies the x-y-z scale factors in its <scales> property. A scale factor of 1.0 means that no scaling occurs. In the example above, all of the scales represented uniform scaling (equal x, y, and z scale factors). Note: the scale factors in Example 1 3 were computed using a utility that computed experimental segment lengths using a functional joint center approach. This utility is not yet part of the standard OpenSim distribution.

Tag

The tag is used to specify which segment will be scaled using those factors.

<apply> Tag

The <apply> property can be used to disable certain scales. <apply> can be set to either **true** or **false**. By default <apply> is set to **true**, so it can generally be omitted.

Example: XML file for a manual scaling file

gait2354_Scale_ScaleSet.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<ScaleSet name="gait2354_Scale">

   <objects>

      <Scale name="">
         <scales> 1.14724 1.14724 1.14724 </scales>
         <segment> femur_r </segment>
         <apply> true </apply>
      </Scale>

      <Scale name="">
         <scales> 1.14724 1.14724 1.14724 </scales>
         <segment> femur_l </segment>
         <apply> true </apply>
      </Scale>


      <Scale name="">
         <scales> 0.988523 0.988523 0.988523 </scales>
         <segment> tibia_r </segment>
         <apply> true </apply>
      </Scale>

      <Scale name="">
         <scales> 0.988523 0.988523 0.988523 </scales>
         <segment> tibia_l </segment>
         <apply> true </apply>
      </Scale>

   </objects>

</ScaleSet>
```

Next: Measurement-Based Scaling File

Previous: Scale Marker File

Home: Scaling

Measurement-Based Scaling File

The topics covered in this section include:

- <apply> Tag
- Specifying Marker Pairs Used to Compute a Scale Factor
- Specifying Bodies to Be Scaled

The measurement-based scaling file contains pairs of experimental markers, the distance between which are used to scale the generic musculoskeletal model. The experimental measurements used for scaling are specified in the example file **gait2354_Scale_MeasurementSet.xml:**

Example: XML file for a measurement-based scaling file

gait2354_Scale_MeasurementSet.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<MeasurementSet name="gait2354">

  <objects>

    <!– . . additional <Measurement> tags cut for brevity . . –>

    <Measurement name="thigh">

      <apply> true </apply>

      <MarkerPairSet>
        <objects>
          <MarkerPair>
            <markers> R.ASIS R.Knee.Lat </markers>
          </MarkerPair>
          <MarkerPair>
            <markers> L.ASIS L.Knee.Lat </markers>
          </MarkerPair>
        </objects>
      </MarkerPairSet>

      <BodyScaleSet>
         <objects>
          <BodyScale name="femur_r">
            <axes> X Y Z </axes>
          </BodyScale>
          <BodyScale name="femur_l">
            <axes> X Y Z </axes>
          </BodyScale>
          <BodyScale name="patella_r">
            <axes> X Y Z </axes>
          </BodyScale>
          <BodyScale name="patella_l">
            <axes> X Y Z </axes>
          </BodyScale>
        </objects>
      </BodyScaleSet>

    </Measurement>

    <!– . . additional <Measurement> tags cut for brevity . . –>

  </objects>
</MeasurementSet>
```

The <MeasurementSet> tag specifies a set of <Measurement> objects. Each set of <Measurement> tags lists the marker pairs that go into computing the scale factor, and all the bodies to which this scale factor will be applied. When defining the bodies, the axes to scale along are also defined. Details about specifying the marker pairs and the bodies are given below.

<apply> Tag

The <apply> property of <Measurement> is used to specify whether a measurement is enabled or not, and takes on the values of either **true** or **false**. It is **true** by default and so can typically be omitted.

Specifying Marker Pairs Used to Compute a Scale Factor

The list of marker pairs is enclosed in a <MarkerPairSet> tag, each pair being given in a <MarkerPair> object that is defined by the <markers> tag. Each set of <markers> tags provides the names of two markers. In Example 1 4, two marker pairs are given. Recall from How Scaling Works that this means that the final scale factor will be computed by averaging the scale factors determined by each individual pair.

Specifying Bodies to Be Scaled

The list of bodies to be scaled using the scale factors determined by the marker-pair-measurements is given in a <BodyScaleSet> tag, with each <BodyScale> specifying the body through its **name** attribute.
The <axes> tag indicates the axis/axes along which to scale and can take on the values X, Y, and/or Z. In the above example, the femur and patella on both sides of the body (r = right, l = left) will be uniformly scaled in the X, Y, and Z directions using these measurement-based scale factors.

Next: Inverse Kinematics Tasks File

Previous: Manual Scaling File

Home: Scaling

Inverse Kinematics Tasks File

The topics covered in this section include:

- Marker Tasks
- Coordinate Tasks

The inverse kinematics (IK) tasks file is an external XML file containing the <IKTaskSet> referred to from the main Scale Setup File. In the example below, the IK tasks file is **gait2354_Scale_Tasks.xml**.

The IK tasks file specifies properties associated with error terms. A brief description of the tags used to describe these properties is given below. The section on Inverse Kinematics provides more details.

Marker Tasks

The <IKMarkerTask> tags are used to specify the weights associated with the marker error terms in the IK formulation. See the IK chapter (Inverse Kinematics) for more details.

Example: XML file for an inverse kinematics tasks file

gait2354_Scale_Tasks.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<IKTaskSet name="gait2354_Scale">

  <objects>
    <IKMarkerTask name="Sternum"> <weight>1</weight> </IKMarkerTask>
    <IKMarkerTask name="R.Acromium"> <weight>1</weight> </IKMarkerTask>
    <IKMarkerTask name="L.Acromium"> <weight>1</weight> </IKMarkerTask>
    <IKMarkerTask name="Top.Head"> <weight>1</weight> </IKMarkerTask>
    <!– . . additional <IKMarkerTask> tags cut for brevity . . –>

    <!– Coordinates –>
    <IKCoordinateTask name="subtalar_angle_r"> <value>0</value> </IKCoordinateTask>
    <IKCoordinateTask name="mtp_angle_r"> <value>0</value> </IKCoordinateTask>
    <IKCoordinateTask name="subtalar_angle_l"> <value>0</value> </IKCoordinateTask>
    <IKCoordinateTask name="mtp_angle_l"> <value>0</value> </IKCoordinateTask>
    <IKCoordinateTask name="lumbar_extension">
    <value>0</value>
    <weight>1000.0</weight>
    </IKCoordinateTask>

  </objects>
</IKTaskSet>
```

Coordinate Tasks

The <IKCoordinateTask> tags are optional and are used to specify properties associated with the coordinate error terms in the IK formulation. See the IK chapter (Inverse Kinematics) for more details. In the example above, the subtalar and mtp angles are locked, and are assigned to the constant value of 0 radians. Additionally, we have a task which attempts to keep the (unprescribed) lumbar_extension coordinate close to 0 by setting its desired value to 0 and giving it a large weight.

Next: Inverse Kinematics

Previous: Measurement-Based Scaling File

Home: Scaling

Inverse Kinematics

In this section, we provide a conceptual review of the inputs and outputs of the Inverse Kinematics (IK) tool, a set of troubleshooting tips and best practices, as well as how to use the IK tool in OpenSim:

- Getting Started with Inverse Kinematics
- How Inverse Kinematics Works
- How to Use the IK Tool
- IK Settings Files and XML Tag Definitions

Next: Getting Started with Inverse Kinematics

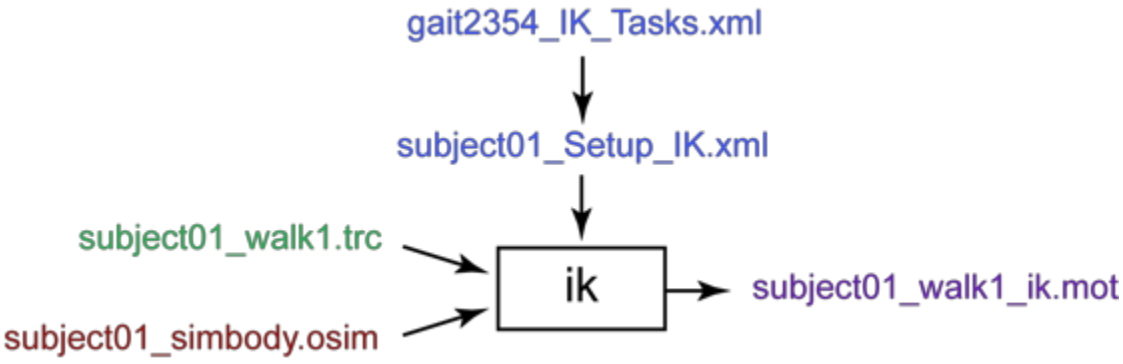
Getting Started with Inverse Kinematics

The Inverse Kinematics Tool steps through each time frame of experimental data and positions the model in a pose that "best matches" experimental marker and coordinate data for that time step. This "best match" is the pose that minimizes a sum of weighted squared errors of markers and/or coordinates. Getting accurate results from the IK tool is essential for using later tools like Static Optimization, Residual Reduction Algorithm, and Computed Muscle Control.

To launch the IK Tool, select **Tools Inverse Kinematics** from the OpenSim main menu bar.

Inverse Kinematics (IK) Tool Overview. Experimental markers are matched by model markers throughout the motion by varying the joint angles (generalized coordinates) through time.

Overview



Inputs and Outputs of the IK Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue.

The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Inputs

The primary inputs to IK are the following files:

1. **subject01_simbody.osim**: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers.
2. **subject01_walk1.trc**: Experimental marker trajectories for a trial obtained from a motion capture system, along with the time range of interest
3. **gait2354_IK_tasks.xml**: A file containing marker weightings. As in the scale tool, marker weights are relative and determine how "well" the virtual markers track experimental markers (i.e., a larger weight will mean less error between virtual and experimental marker positions).
4. **subject01_coords.mot** (optional): Experimental generalized coordinate values (joint angles) for a trial obtained from alternative motion capture devices or other specialized algorithms. You can optionally specify relative coordinate weights in the Tasks file, if joint angles are known a priori.

Outputs

1. **subject01_walk1_ik.mot**: A motion file containing the generalized coordinate trajectories (joint angles and/or translations) computed by IK.

Best Practices and Troubleshooting

Data Collection and Other Preparation

1. When collecting experimental data, place three non-collinear markers per body segment that you want to track. You need at least three markers to track the 6 DOF motion (position and orientation) of a body segment.
2. Place markers on anatomical locations with minimum skin/muscle motion.

Inverse Kinematics Settings

1. Weight "motion" segment markers, for example from a triad placed on the thigh segment, more heavily than anatomical markers affixed to landmarks like the greater trochanter and the acromion, which can be helpful for scaling, but are influenced by muscle and other soft tissue movements during motion.
2. Relative marker weightings are more important than their absolute values. Therefore, a weighting of 10 vs. 1 is 10 times more important whereas 20 vs. 10 is only twice as important. Markers are not necessarily tracked better because they both have higher weightings.
3. See How Inverse Kinematics Works and How to Use the IK Tool for more information about IK settings.

Evaluating your Results

1. Total RMS and max marker errors are reported in the messages window. Use these values to guide changes in weightings, or if necessary to redo marker placement and possibly scaling. Maximum marker error should generally be less than 2-4 cm and RMS under 2 cm is achievable. These guidelines will vary depending on the nature of the model and the motion being examined.
2. If using coordinates from a motion capture system make sure that the joint/coordinate definitions match otherwise you may cause more harm than good.
3. Compare your results to similar data reported in the literature. Your results from an unimpaired average adult should generally be within one standard deviation.
4. If you are unsatisfied with the results, recheck the results of Scale.

Next: How Inverse Kinematics Works

Home: Inverse Kinematics

How Inverse Kinematics Works

The IK tool goes through each time step (frame) of motion and computes generalized coordinate values which positions the model in a pose that "best matches" experimental marker and coordinate values for that time step. Mathematically, the "best match" is expressed as a weighted least squares problem, whose solution aims to minimize both marker and coordinate errors. The topics covered in this section include:

- Marker Errors
- Coordinate Errors
- Weighted Least Squares Equation
- Important Note about Units

Marker Errors

A *marker error* is the distance between an experimental marker and the corresponding marker on the model when it is positioned using the generalized coordinates computed by the IK solver. Each marker has a weight associated with it, specifying how strongly that marker's error term should be minimized.

Coordinate Errors

A *coordinate error* is the difference between an experimental coordinate value and the coordinate value computed by IK.

What are "experimental coordinate values?" These can be joint angles obtained directly from a motion capture system (i.e., built-in mocap inverse kinematics capabilities), or may be computed from experimental data by various specialized algorithms (e.g., defining anatomical coordinate frames and using them to specify joint frames that, in turn, describe joint angles) or by other measurement techniques that involve other measurement devices (e.g., a goniometer). A fixed desired value for a coordinate can also be specified (e.g., if you know a specific joint's angle should stay at 0). The inclusion of experimental coordinate values is optional; the IK tool can solve for the motion trajectories using marker matching alone.

A distinction should be made between *prescribed* and *unprescribed coordinates*. A prescribed coordinate (also referred to as a *locked coordinate*) is a generalized coordinate whose trajectory is known and which will not be computed using IK. It will get set to its exact trajectory value instead. This can be useful when you have enough confidence in some generalized coordinate value that you don't want the IK solver to change it.

An *unprescribed coordinate* is a coordinate which is not prescribed, and whose value is computed using IK.
Using these definitions, only *unprescribed coordinates* can vary and so only they appear in the least squares equation solved by IK. Each unprescribed coordinate being compared to an experimental coordinate must have a weight associated with it, specifying how strongly that coordinate's error should be minimized.

Weighted Least Squares Equation

The weighted least squares problem solved by IK is

$$\min_{\mathbf{q}} \left[\sum_{i \in \text{markers}} w_i \left\| \mathbf{x}_i^{\text{exp}} - \mathbf{x}_i(\mathbf{q}) \right\|^2 + \sum_{j \in \text{unprescribed coords}} \omega_j \left(q_j^{\text{exp}} - q_j \right)^2 \right]$$

$$q_j = q_j^{\text{exp}} \text{ for all prescribed coordinates } j$$

where **q** is the vector of generalized coordinates being solved for, $\mathbf{x}_i^{\text{exp}}$ is the experimental position of marker i, $\mathbf{x}_i(\mathbf{q})$ is the position of the corresponding marker on the model (which depends on the coordinate values), q_j^{exp} is the experimental value for coordinate j. Prescribed coordinates are set to their experimental values. For instance, in the gait2354 and gait2392 examples, the subtalar and metatarsophalangeal (mtp) joints are locked and during IK they are assigned the prescribed value of 0.

The marker weights (w_i's) and coordinate weights ($_j$'s) are specified in the <IKMarkerTask> and <IKCoordinateTask> tags, respectively. These are all specified within a single <IKTaskSet> tag, as will be outlined in How to Use the IK Tool. This least squares problem is solved using a general quadratic programming solver, with a convergence criterion of 0.0001 and a limit of 1000 iterations. These are currently fixed values that cannot be changed in the XML files.

Important Note about Units

The least squares solution is affected by the choice of length and angle units. The units used by IK are the model's units, which are **meters** for length and **radians** for angles. This is important, for example, if you wish to compare these results to a different IK solver that uses degrees for measuring coordinate errors. In order to get similar results using OpenSim's choice of radians as units, you should alter the weightings

accordingly: scale each coordinate's weight by $\left(\frac{180}{\pi}\right)^2$ (so a weight of 1 with a degrees-based IK solver becomes $\left(\frac{180}{\pi}\right)^2$ for the IKTool in OpenSim, which is radian-based).

Next: How to Use the IK Tool

Previous: Getting Started with Inverse Kinematics

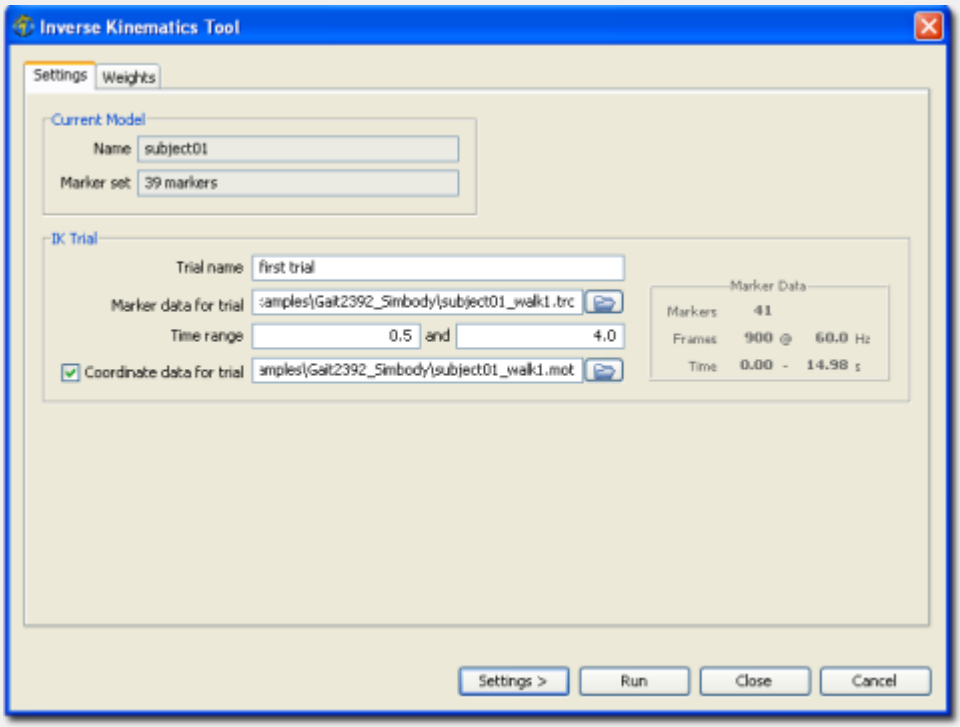
Home: Inverse Kinematics

How to Use the IK Tool

The topics covered in this section include:

How to Use the GUI

To launch the IK Tool:

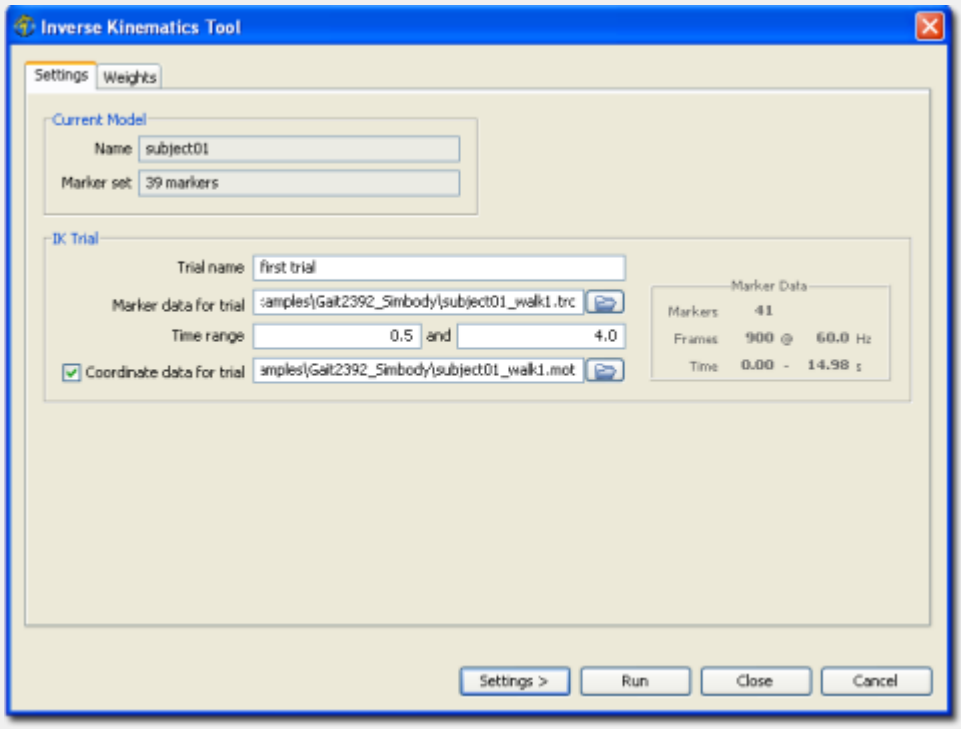


1. Select **Tools Inverse Kinematics** from the OpenSim main menu bar.
2. The *Settings* pane is used to specify parameters related to the experimental marker data. The *Weights* pane is used to specify the marker and coordinate weights used in the weighted least squares equation.

The **Inverse Kinematics Tool** window, like all other OpenSim tools, operates on the current model. The name of the current model is shown in bold in the Navigator window. Any model can be made the current model by **right-clicking** on its name and selecting **Make Current**. See Opening, Closing, and Using the Navigator Window for information on opening models and making a particular model current.

Settings Pane

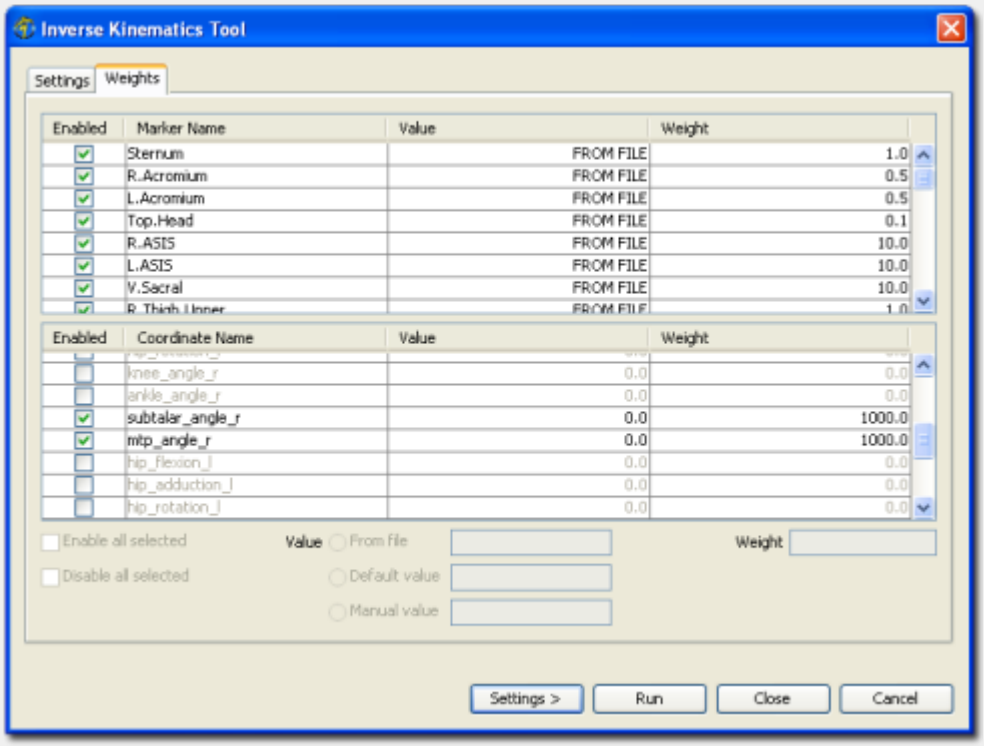
The *Settings* pane is used to specify parameters related to the experimental marker data. The pane is organized into two main sections: *Current Model* and *IK Trial*.



- The section for *Current Model* displays uneditable information about the current model. It gives the model name, as well as the number of markers associated with the model. Inverse kinematics requires that a marker set is associated with the model. This association with a subject-specific model is established at the time that a generic model is scaled to the subject markers visit Scaling for more information.
- The *IK Trial* section specifies the experimental marker data that the IK Tool will match with the current model. A **Trial name** can be associated with the trial to uniquely identify the resultant motion. The **Marker data for trial** field contains the path to the marker data (in .trc format). Information from the file, such as the number of markers, the number of frames and the sampling frequency, as well as the start and end times of the data set, are reported in the *Marker Data* box on the right-hand side. You can specify the time range for performing inverse kinematics in the **Time range** field. By default, the complete time range is used. If you are using experimental generalized coordinate values in the IK trial, check the **Coordinate data for trial** checkbox and then specify the motion (.mot) file with the coordinate values in the textbox to the right.

Weights Pane

Once a marker and possibly a coordinate file have been specified, the specific behavior of the inverse kinematics tool can be described and modified using the *Weights* pane (Figure below). Each entry in the table represents a task in the least-squares matching for either a marker (top table) or a coordinate (lower table). By **left-clicking** on a row, you select it, making the entry fields below the tables editable so you can specify weights and values for the selected marker(s) or coordinate(s). You can select multiple rows to edit by using **ctrl + left-mouse-click** or **shift + left-mouse-click**.



- The weight value affects to what degree a match should be satisfied.
- Larger weightings penalize errors for that marker or coordinate more heavily and thus should match the experimental value more closely (see How Inverse Kinematics Works).
- For coordinates, the coordinate value to be matched can come from a specified motion file, be set to its default value, or be set to a user-specified (*manual*) value.

When running the IK Tool using the GUI, the results are not automatically saved to a file. They are associated with the model under its **Motions** node in the Navigator window (Navigator Window). You can view multiple IK results before saving them to a file. To save the IK results to a file, **right click** on the motion in the Navigator window and select **Save as**.

The Control Panel

The *control panel* at the bottom of the dialog window is used to save the settings in the dialog to a file, read the settings from a file, and to run IK. It has the same behavior as described in Scale Setup File

Command Line Execution of IK

The Inverse Kinematics Tool is run using the command **ik -S <setup file name>**, for example:

```
ik -S subject01_Setup_IK.xml
```

Inverse Kinematics Setup File

There are three properties that need to be specified in an inverse kinematics setup file:

1. The model to which the IK solver is to be applied;
2. The marker and coordinate error weightings to be used; and
3. The specific trials to be used by the solver.

A sample inverse kinematics setup file is provided in the example beow:

subject01_Setup_IK.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<IKTool name="subject01">

  <model_file> subject01.osim </model_file>

  <IKTaskSet file="gait2354_IK_Tasks.xml"/>

  <IKTrialSet name="">
    <objects>

      <IKTrial name="first trial">
        <marker_file> subject01_walk1.trc </marker_file>
        <coordinate_file> subject01_walk1.mot </coordinate_file>

        <time_range> 0.4 2.0 </time_range>

        <output_motion_file> subject01_walk1_ik.mot  </output_motion_file>
      </IKTrial>

    </objects>
  </IKTrialSet>

</IKTool>
```

Example IK Setup File: XML file for an inverse kinematics setup file

IK Execution Parameters

The parameters/properties for running the IK tool are enclosed inside the opening and closing tags <IKTool> and </IKTool>. If desired, the name attribute name="subject01" can be used to specify the execution name, though it is not used anywhere.

Model

The <model_file> property specifies the file name of an OpenSim (.osim) model that will be used to match the experimental data and/or coordinate values. Typically this will be the subject-specific scaled model generated by the Scale Tool.

IK Tasks

The IK tasks are used to specify the weights on the marker and coordinate error terms used by the IK solver. The set of IK tasks is specified in the <IKTaskSet> property. In Settings Files and XML Tag Definitions for IK, it points to an external file (gait2354_IK_Tasks.xml) using the **file** attribute, but the IKTaskSet could be specified in the setup file itself. A separate IK tasks file is useful if several models (subjects) will share the same set of IK tasks. Details for specifying the IK tasks file is presented below.

Per-trial Properties

Multiple experimental trials for the same subject/model can be processed during a single execution of IK. The IK Tool will process each trial in sequence and output individual motion files for each set of trial (IKTrial) settings.

The settings for each trial are enclosed by the <IKTrial> and </IKTrial> tags, and the <IKTrialSet> and </IKTrialSet> tags enclose a set of such trials. The <objects> tag is needed because this is a set of data. Note that in the XML file shown in the Example IK Setup File above, only a single trial is computed.

For each trial, the following parameters are needed: the file name for the experimental marker trajectories, the file name for the experimental coordinate values, the time range for computing the inverse kinematics, and the output motion file name. Each of these parameters is explained in more detail in the following sections.

Experimental marker and coordinate values

The <marker_file> property specifies a .trc file containing marker trajectories for this trial. These trajectories give the experimental marker positions which IK will attempt to match as described in How Inverse Kinematics Works.

The <coordinate_file> property specifies a .mot file containing experimental coordinate values (e.g., joint angles computed using anatomical reference frames) for this trial. These values are read in from the file only if there is an IK Task for the coordinate that has a <from_file> tag set to

true (see <IKCoordinateTask> in the Example IK Task File below).

If none of the coordinates have <from_file> set to **true**, the <coordinate_file> property becomes optional. However, a coordinates (motion) file is typically specified even in this situation, because it may also contain other data, such as ground reaction forces and torques, which are copied into the output file generated by IK.

Time range

The <time_range> tag specifies the start and end times for the IK computation (e.g., t=0.4s to t=2.0s in the example above). The time values are in whatever units of time were used in the marker and coordinate files, which must share a range with one another. Note that IK does not require time be expressed in seconds. However, future dynamics analyses which compute accelerations must have time expressed in seconds.

A start time earlier than the first available marker/coordinate data is snapped to the initial time of the data. Similarly, the end time is snapped to the final time of the data, if necessary. Time points that do not coincide with discrete times in the experimental data set are snapped to the nearest time point before the specified time. For example, a specified end time of 0.410s will be snapped to 0.400s if the next time instant is 0.417s from a motion capture system sampling at 60Hz.

Note that the marker file is used to determine the actual time steps (frames) at which IK is computed. Furthermore, if a coordinate file is used, the time values of the rows of data in the coordinate file must match the time values of the data in the marker file. OpenSim will report an error if a corresponding time in the coordinate file cannot be found.

Output file

<output_motion_file> specifies the name of the motion file generated by IK. The columns of this file are:

- Unprescribed generalized coordinates (those solved for by IK), followed by
- Prescribed generalized coordinates (those with fixed values not solved for by IK), followed by
- Any user data (data found in the <coordinate_file> which is not directly used by IK, e.g., ground reaction forces)

Each row represents an instant for which IK determined coordinates corresponding to the times in the marker data file. As a motion (.mot) file, each column has an associated label identifying the joint coordinate, ground reaction, center of pressure, etc. values.

Inverse Kinematics Tasks File

The inverse kinematics tasks file is an external XML file containing the <IKTaskSet> tags used to specify properties associated with error terms during the IK computation. This file is referred to from the main IK setup file (see the Example IK Setup File above). An example IK tasks file is given in Example IK Task File below. Recall that this file is common to all IK trials.

gait_IK_tasks.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<IKTaskSet name="gait2354_IK">
  <objects>

    <!-- Markers -->
    <IKMarkerTask name="Sternum"> <weight>1</weight> </IKMarkerTask>

    <IKMarkerTask name="R.Acromium"> <weight>0.5</weight></IKMarkerTask>

    <IKMarkerTask name="L.Acromium"> <weight> 0.5 </weight></IKMarkerTask>

    <IKMarkerTask name="Top.Head"> <weight> 0.1 </weight> </IKMarkerTask>

    <!-- . . additional <IKMarkerTask> tags cut for brevity . . -->

    <!-- Coordinates -->
    <IKCoordinateTask name="subtalar_angle_r"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="mtp_angle_r"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="subtalar_angle_l"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="mtp_angle_l"> <value> 0 </value></IKCoordinateTask>

  </objects>
```

If desired, the name attribute (e.g., name="gait2354_IK" as shown above) can be used to specify the execution name for IKTaskSet, though it is not used anywhere.

Marker Tasks

The <IKMarkerTask> tags are used to specify the weights associated with the marker error terms in the IK formulation. A specific marker is identified using the **name** attribute of <IKMarkerTask>.

The <weight> property specifies the weight multiplying the squared-error term for that marker. These weights are the w_i's appearing in the formula given in How Inverse Kinematics Works. A larger weight means that the error term for that marker will be more significant in the least-squares equation, so the IK computation will try to find a closer match between that marker's position and its experimental position, as compared to a marker with a smaller weight. By default, the weight for a marker is assumed to be zero, indicating that the IK solver will not attempt to match that marker to its experimental position. The default weight is used if <weight> is not specified or if the <IKMarkerTask> is omitted for a marker.

Experimental marker positions are not specified here. They come from the .trc file specified by the <marker_file> property of the <IKTrial> specified in the Example IK Setup File.

Coordinate Tasks

The <IKCoordinateTask> tags are used to specify properties associated with the coordinate error terms in the IK formulation. A specific coordinate is identified using the **name** attribute of <IKCoordinateTask>.

Three properties can be specified in a <IKCoordinateTask>: <weight>, <from_file>, and <value>. These properties are not required.

<weight> tag

The <weight> property specifies the weight multiplying the squared-error term for that coordinate. These weights are the $_i$'s appearing in the formula given in How Inverse Kinematics Works. They tell the IK solver how closely it should try to match a given coordinate to its experimental value: the larger the weight, the closer the match should be.

Recall from How Inverse Kinematics Works that there are two types of coordinates: prescribed and unprescribed. If the OpenSim model specifies a coordinate to be locked (the <IKCoordinateTask> has <locked> set to **true**), then it is a prescribed coordinate. The lock symbol in the Coordinates Window will also be in the locked position for this coordinate. Prescribed coordinates will be assigned an exact value and will not be solved for by IK. Thus, the <weight> property has no effect on prescribed coordinates; it is only relevant for unprescribed coordinates.

For an unprescribed coordinate, if <weight> is not specified or if its <IKCoordinateTask> is omitted altogether, the weight for that coordinate is assumed to be zero, meaning that IK will not attempt to match that coordinate to any particular value.

Specify experimental values to be matched

For both prescribed coordinates and unprescribed coordinates with nonzero weights, the experimental value to be matched needs to be specified. One way to do this is to specify a .mot file using the <from_file> and </from_file> tags. If <from_file> is set to **true**, then the experimental coordinate values for that coordinate come from the .mot file specified by the <coordinate_file> property in the Example IK Setup File.

If <from_file> is set to **false** or is not specified, then a constant experimental value is used instead of the time-varying trajectory from a file. This constant value comes from:

- The <value> property of the <IKCoordinateTask>, or if that's not specified then
- The <value> property of the <SimmCoordinate> in the OpenSim .osim model, or if that's not specified then
- The <default_value> property of the <SimmCoordinate> in the OpenSim .osim model.

In the Example IK Task File, the subtalar and mtp angles are locked (prescribed) in the model file. This is indicated in the Coordinates Window (the lock symbol is in the locked position). So <value> is used to specify that they should be locked at 0 radians. A <weight> is not needed since these are prescribed coordinates and will be set to 0 exactly. Note, too, if the coordinate default was set to 0, you could have relied on the default behavior and omitted the <IKCoordinateTask> for the subtalar and mtp angles.

Next: IK Settings Files and XML Tag Definitions

Previous: How Inverse Kinematics Works

Home: Inverse Kinematics

IK Settings Files and XML Tag Definitions

The topics covered in this section include:

Inverse Kinematics Setup File

There are three properties that need to be specified in an inverse kinematics setup file:

1. The model to which the IK solver is to be applied;
2. The marker and coordinate error weightings to be used; and
3. The specific trials to be used by the solver.

A sample inverse kinematics setup file is provided in the example beow:
Example IK Setup File: XML file for an inverse kinematics setup file

subject01_Setup_IK.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<IKTool name="subject01">

  <model_file> subject01.osim </model_file>

  <IKTaskSet file="gait2354_IK_Tasks.xml"/>

  <IKTrialSet name="">
    <objects>

      <IKTrial name="first trial">
        <marker_file> subject01_walk1.trc </marker_file>
        <coordinate_file> subject01_walk1.mot </coordinate_file>

        <time_range> 0.4 2.0 </time_range>

        <output_motion_file> subject01_walk1_ik.mot  </output_motion_file>
      </IKTrial>

    </objects>
  </IKTrialSet>

</IKTool>
```

IK Execution Parameters

The parameters/properties for running the IK tool are enclosed inside the opening and closing tags <IKTool> and </IKTool>. If desired, the name attribute name="subject01" can be used to specify the execution name, though it is not used anywhere.

Model

The <model_file> property specifies the file name of an OpenSim (.osim) model that will be used to match the experimental data and/or coordinate values. Typically this will be the subject-specific scaled model generated by the Scale Tool.

IK Tasks

The IK tasks are used to specify the weights on the marker and coordinate error terms used by the IK solver. The set of IK tasks is specified in the <IKTaskSet> property. In Settings Files and XML Tag Definitions for IK, it points to an external file (gait2354_IK_Tasks.xml) using the **file** attribute, but the IKTaskSet could be specified in the setup file itself. A separate IK tasks file is useful if several models (subjects) will share the same set of IK tasks. Details for specifying the IK tasks file is presented below.

Per-trial Properties

Multiple experimental trials for the same subject/model can be processed during a single execution of IK. The IK Tool will process each trial in sequence and output individual motion files for each set of trial (IKTrial) settings.

The settings for each trial are enclosed by the <IKTrial> and </IKTrial> tags, and the <IKTrialSet> and </IKTrialSet> tags enclose a set of such trials. The <objects> tag is needed because this is a set of data. Note that in the XML file shown in the Example IK Setup File above, only a single trial is computed.

For each trial, the following parameters are needed: the file name for the experimental marker trajectories, the file name for the experimental coordinate values, the time range for computing the inverse kinematics, and the output motion file name. Each of these parameters is explained in more detail in the following sections.

Experimental marker and coordinate values

The <marker_file> property specifies a .trc file containing marker trajectories for this trial. These trajectories give the experimental marker positions which IK will attempt to match as described in How Inverse Kinematics Works.

The <coordinate_file> property specifies a .mot file containing experimental coordinate values (e.g., joint angles computed using anatomical reference frames) for this trial. These values are read in from the file only if there is an IK Task for the coordinate that has a <from_file> tag set to **true** (see <IKCoordinateTask> in the Example IK Task File below).

If none of the coordinates have <from_file> set to **true**, the <coordinate_file> property becomes optional. However, a coordinates (motion) file is typically specified even in this situation, because it may also contain other data, such as ground reaction forces and torques, which are copied into the output file generated by IK.

Time range

The <time_range> tag specifies the start and end times for the IK computation (e.g., t=0.4s to t=2.0s in the example above). The time values are in whatever units of time were used in the marker and coordinate files, which must share a range with one another. Note that IK does not require time be expressed in seconds. However, future dynamics analyses which compute accelerations must have time expressed in seconds.

A start time earlier than the first available marker/coordinate data is snapped to the initial time of the data. Similarly, the end time is snapped to the final time of the data, if necessary. Time points that do not coincide with discrete times in the experimental data set are snapped to the nearest time point before the specified time. For example, a specified end time of 0.410s will be snapped to 0.400s if the next time instant is 0.417s from a motion capture system sampling at 60Hz.

Note that the marker file is used to determine the actual time steps (frames) at which IK is computed. Furthermore, if a coordinate file is used, the time values of the rows of data in the coordinate file must match the time values of the data in the marker file. OpenSim will report an error if a corresponding time in the coordinate file cannot be found.

Output file

<output_motion_file> specifies the name of the motion file generated by IK. The columns of this file are:

- Unprescribed generalized coordinates (those solved for by IK), followed by
- Prescribed generalized coordinates (those with fixed values not solved for by IK), followed by
- Any user data (data found in the <coordinate_file> which is not directly used by IK, e.g., ground reaction forces)

Each row represents an instant for which IK determined coordinates corresponding to the times in the marker data file. As a motion (.mot) file, each column has an associated label identifying the joint coordinate, ground reaction, center of pressure, etc. values.

Inverse Kinematics Tasks File

The inverse kinematics tasks file is an external XML file containing the <IKTaskSet> tags used to specify properties associated with error terms during the IK computation. This file is referred to from the main IK setup file (see the Example IK Setup File above). An example IK tasks file is given in Example IK Task File below. Recall that this file is common to all IK trials.

gait_IK_tasks.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<IKTaskSet name="gait2354_IK">
  <objects>

    <!-- Markers -->
    <IKMarkerTask name="Sternum"> <weight>1</weight> </IKMarkerTask>

    <IKMarkerTask name="R.Acromium"> <weight>0.5</weight></IKMarkerTask>

    <IKMarkerTask name="L.Acromium"> <weight> 0.5 </weight></IKMarkerTask>

    <IKMarkerTask name="Top.Head"> <weight> 0.1 </weight> </IKMarkerTask>

    <!-- . . additional <IKMarkerTask> tags cut for brevity . . -->

    <!-- Coordinates -->
    <IKCoordinateTask name="subtalar_angle_r"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="mtp_angle_r"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="subtalar_angle_l"> <value> 0 </value></IKCoordinateTask>

    <IKCoordinateTask name="mtp_angle_l"> <value> 0 </value></IKCoordinateTask>

  </objects>
```

Example IK Task File: Annotated XML file for an inverse kinematics tasks file

If desired, the name attribute (e.g., name="gait2354_IK" as shown above) can be used to specify the execution name for IKTaskSet, though it is not used anywhere.

Marker Tasks

The <IKMarkerTask> tags are used to specify the weights associated with the marker error terms in the IK formulation. A specific marker is identified using the **name** attribute of <IKMarkerTask>.

The <weight> property specifies the weight multiplying the squared-error term for that marker. These weights are the w_i's appearing in the formula given in How Inverse Kinematics Works. A larger weight means that the error term for that marker will be more significant in the least-squares equation, so the IK computation will try to find a closer match between that marker's position and its experimental position, as compared to a marker with a smaller weight. By default, the weight for a marker is assumed to be zero, indicating that the IK solver will not attempt to match that marker to its experimental position. The default weight is used if <weight> is not specified or if the <IKMarkerTask> is omitted for a marker.

Experimental marker positions are not specified here. They come from the .trc file specified by the <marker_file> property of the <IKTrial> specified in the Example IK Setup File.

Coordinate Tasks

The <IKCoordinateTask> tags are used to specify properties associated with the coordinate error terms in the IK formulation. A specific coordinate is identified using the **name** attribute of <IKCoordinateTask>.

Three properties can be specified in a <IKCoordinateTask>: <weight>, <from_file>, and <value>. These properties are not required.

<weight> tag

The <weight> property specifies the weight multiplying the squared-error term for that coordinate. These weights are the $_i$'s appearing in the formula given in How Inverse Kinematics Works. They tell the IK solver how closely it should try to match a given coordinate to its experimental value: the larger the weight, the closer the match should be.

Recall from How Inverse Kinematics Works that there are two types of coordinates: prescribed and unprescribed. If the OpenSim model specifies a coordinate to be locked (the <IKCoordinateTask> has <locked> set to **true**), then it is a prescribed coordinate. The lock symbol in the Coordinates Window will also be in the locked position for this coordinate. Prescribed coordinates will be assigned an exact value and will not be solved for by IK. Thus, the <weight> property has no effect on prescribed coordinates; it is only relevant for unprescribed coordinates.

For an unprescribed coordinate, if <weight> is not specified or if its <IKCoordinateTask> is omitted altogether, the weight for that coordinate is assumed to be zero, meaning that IK will not attempt to match that coordinate to any particular value.

Specify experimental values to be matched

For both prescribed coordinates and unprescribed coordinates with nonzero weights, the experimental value to be matched needs to be specified. One way to do this is to specify a .mot file using the <from_file> and </from_file> tags. If <from_file> is set to **true**, then the experimental coordinate values for that coordinate come from the .mot file specified by the <coordinate_file> property in the Example IK Setup File.

If <from_file> is set to **false** or is not specified, then a constant experimental value is used instead of the time-varying trajectory from a file. This constant value comes from:

- The <value> property of the <IKCoordinateTask>, or if that's not specified then
- The <value> property of the <SimmCoordinate> in the OpenSim .osim model, or if that's not specified then
- The <default_value> property of the <SimmCoordinate> in the OpenSim .osim model.

In the Example IK Task File, the subtalar and mtp angles are locked (prescribed) in the model file. This is indicated in the Coordinates Window (the lock symbol is in the locked position). So <value> is used to specify that they should be locked at 0 radians. A <weight> is not needed since these are prescribed coordinates and will be set to 0 exactly. Note, too, if the coordinate default was set to 0, you could have relied on the default behavior and omitted the <IKCoordinateTask> for the subtalar and mtp angles.

Next: Inverse Dynamics

Previous: How to Use the IK Tool

Home: Inverse Kinematics

Inverse Dynamics

In this section we will cover:

- Getting Started with Inverse Dynamics
- How Inverse Dynamics Works
- How to Use the Inverse Dynamics Tool
- ID Settings Files and XML Tags

Next: Getting Started with Inverse Dynamics

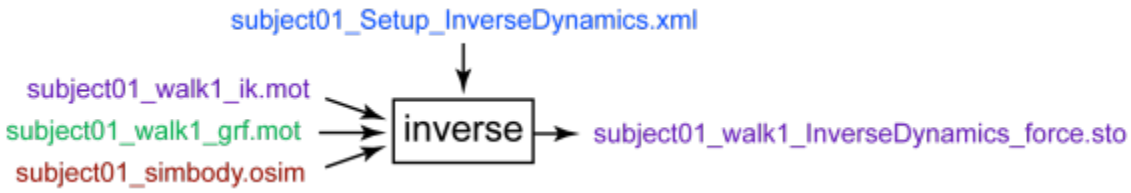
Getting Started with Inverse Dynamics

The inverse dynamics tool determines the generalized forces (e.g., net forces and torques) at each joint responsible for a given movement. Given the kinematics (e.g., states or motion) describing the movement of a model and perhaps a portion of the kinetics (e.g., external loads) applied to the model, the tool uses these data to perform an inverse dynamics analysis. Classical mechanics mathematically expresses the mass-dependent relationship between force and acceleration, F = ma , with equations of motion. The inverse dynamics tool solves these equations, in the inverse dynamics sense, to yield the net forces and torques at each joint which produce the movement.

To launch the ID Tool, select **Tools Inverse Dynamics** from the OpenSim main menu bar.

Overview

This figure shows the required inputs and outputs for the Inverse Dynamics Tool.



Inputs and Outputs of the Inverse Dynamics Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

> The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Settings File

The **subject01_Setup_InverseDynamics.xml** file is the setup file for the Inverse Dynamics Tool. It contains settings, as described in detail in How to Use the Inverse Dynamics Tool.

Inputs

Three data files are required as input by the inverse dynamics tool:

subject01_walk1_ik.mot: Motion file containing the time histories of generalized coordinates that describe the movement of the model. This file could be generated by the Inverse Kinematics Tool, or manually. The file does not need to contain values for all coordinates. The coordinates that were not specified are assumed to have default values by the Tool.

subject01_walk1_grf.xml: xternal load data (i.e., ground reaction forces, moments, and center of pressure location). Note that it is necessary to measure and apply or model all external forces acting on a subject during the motion to calculate accurate joint torques and forces. This file includes the name of the ground reaction force-data file (e.g. subject01_grf.mot) as well as the names of the bodies they are applied to. Options to specify the forces, point of application, and torques in a global or body local frame (relative to the body to which the force is being applied) are also defined here. Details are provided in How to Use the Inverse Dynamics Tool.

subject01_simbody.osim: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters. Note that forces like contact, ligaments, bushings, and even muscles will be applied to the model based on the kinematic state of the model and defaults for the muscle states, unless these forces are specifically excluded in the calculation.

Outputs

The Inverse Dynamics Tool generates a single file in a folder specified in the setup file:
subject01_walk1_InverseDynamics_force.sto: Storage file containing the time histories of the net joint torques and forces, acting along the coordinate axes that produce the accelerations estimated (via double differentiation) from your measured experimental motion and modeled and external forces applied.

Best Practices and Troubleshooting

1. Filter your raw coordinate data, since noise is amplified by differentiation. Without filtering, the calculated forces and torques will be very noisy.
2. Compare your results to data reported in the literature. Your results should be within one s.d. of reported values.
3. Inspect results from Inverse Dynamics to check if ground reaction forces were applied correctly or not. Are there large and unexpected forces at the pelvis? For gait, applying ground reaction forces should help reduce the forces computed by Inverse Dynamics at the pelvis.
4. See How Inverse Dynamics Works and How to Use the Inverse Dynamics Tool for more information about using the Inverse Dynamics Tool.

Next: How Inverse Dynamics Works

Home: Inverse Dynamics

How Inverse Dynamics Works

The classical equations of motion may be written in the following form:

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

where *N* is the number of degrees of freedom;

$q, \dot{q}, \ddot{q} \in R^N$ are the vectors of generalized positions, velocities, and accelerations, respectively;

$M(q) \in R^{N \times N}$ is the system mass matrix;

$C(q, \dot{q}) \in R^N$ is the vector of Coriolis and centrifugal forces;

$G(q) \in R^N$ is the vector of gravitational forces;

and $\tau \in R^N$ is the vector of generalized forces.

The motion of the model is completely defined by the generalized positions, velocities, and accelerations. Consequently, all of the terms on the left-hand side of the equations of motion are known. The remaining term on the right-hand side of the equations of motion is unknown. The inverse dynamics tool uses the known motion of the model to solve the equations of motion for the unknown generalized forces.

Next: How to Use the Inverse Dynamics Tool

Previous: Getting Started with Inverse Dynamics

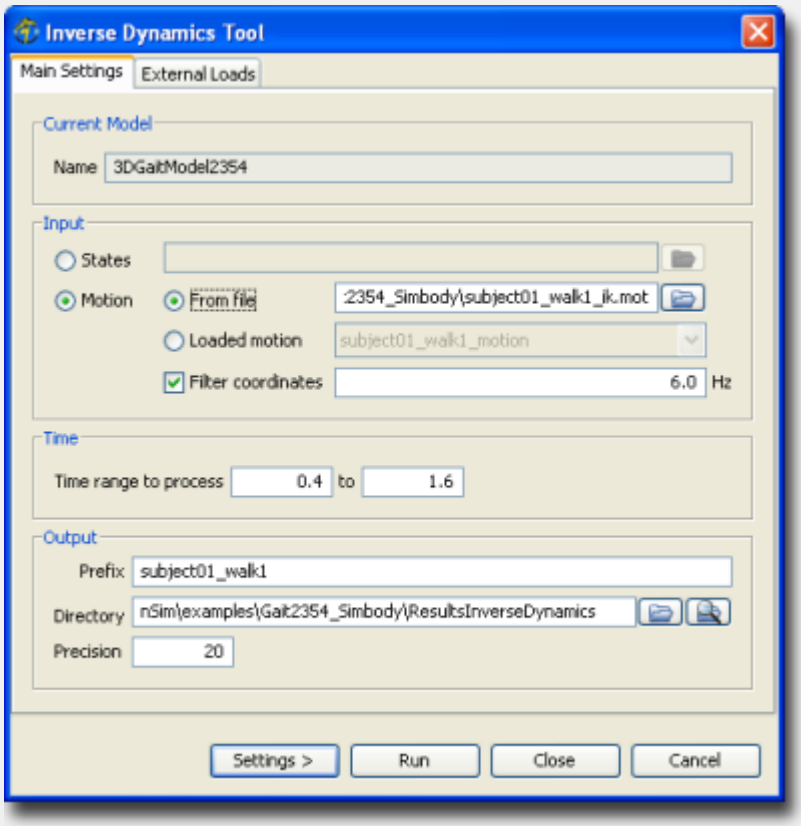
Home: Inverse Dynamics

How to Use the Inverse Dynamics Tool

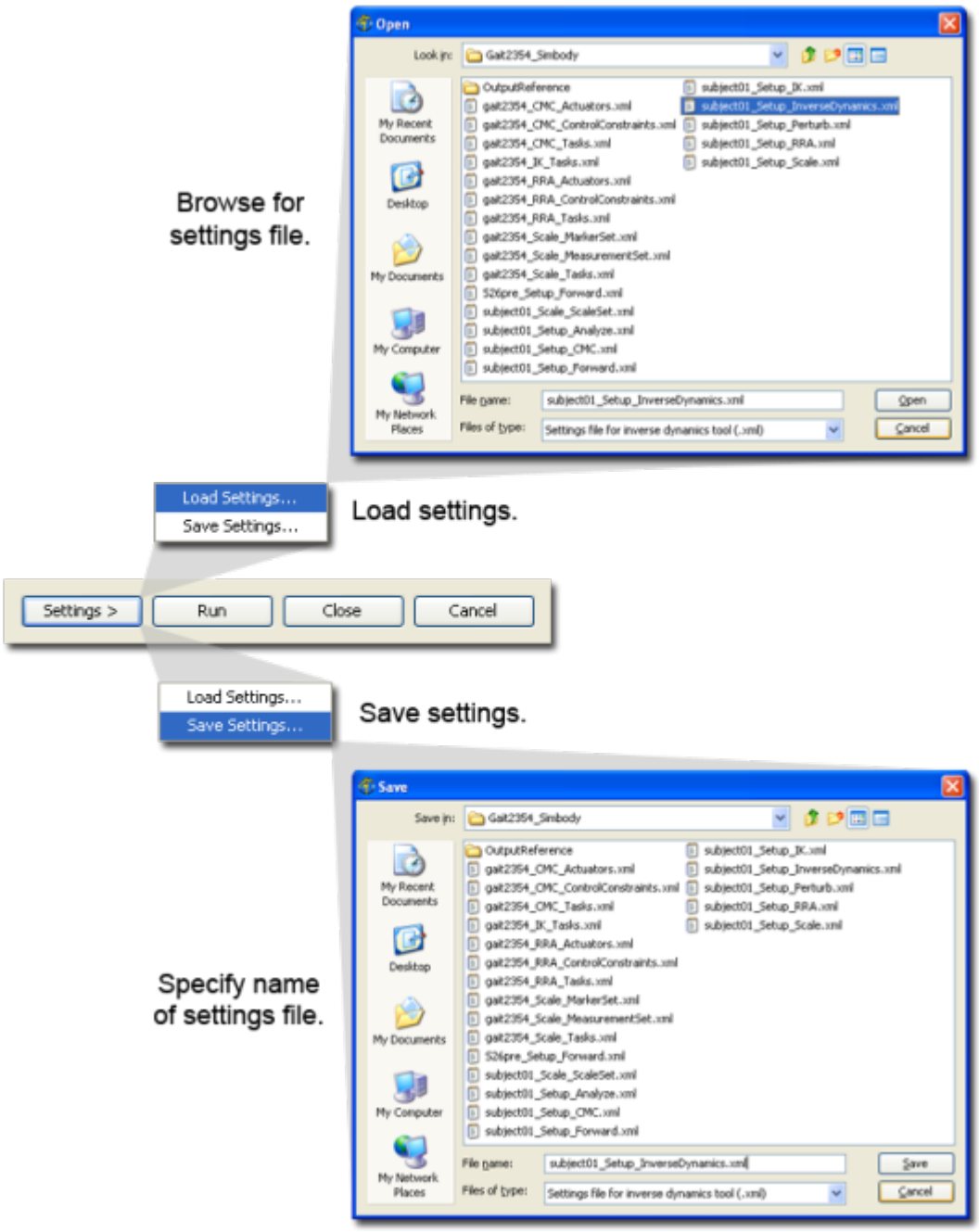
The topics covered in this section include:

How to Use the GUI

The inverse dynamics tool is accessed by selecting **Tools Inverse Dynamics…** from the OpenSim main menu bar. Like all tools, the operations performed by the inverse dynamics tool apply to the current model. The name of the current model is shown in bold in the Navigator. See Opening, Closing, and Using the Navigator Window for information on opening models and making a particular model current.



- The Inverse Dynamics Tool is controlled by a window with two tabbed panes.
- The *Main Settings* pane is used to specify parameters relating to the input kinematics of the current model, the time range for the analysis, and the output of the results.
- The *External Loads* pane is used to specify parameters relating to the external loads applied to the model during the analysis.
- At the bottom of the window are four buttons. (see below for more information)


Browse for settings file.
Open
Look in: Gait2354_Simbody
OutputReference
gait2354_CMC_Actuators.xml
gait2354_CMC_ControlConstraints.xml
gait2354_CMC_Tasks.xml
gait2354_IK_Tasks.xml
gait2354_RRA_Actuators.xml
gait2354_RRA_ControlConstraints.xml
gait2354_RRA_Tasks.xml
gait2354_Scale_MarkerSet.xml
gait2354_Scale_MeasurementSet.xml
gait2354_Scale_Tasks.xml
S26pre_Setup_Forward.xml
subject01_Scale_ScaleSet.xml
subject01_Setup_Analyze.xml
subject01_Setup_CMC.xml
subject01_Setup_Forward.xml
subject01_Setup_IK.xml
subject01_Setup_InverseDynamics.xml
subject01_Setup_Perturb.xml
subject01_Setup_RRA.xml
subject01_Setup_Scale.xml
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
File name: subject01_Setup_InverseDynamics.xml
Files of type: Settings file for inverse dynamics tool (.xml)
Open
Cancel
Load Settings...
Save Settings...
Load settings.
Settings >
Run
Close
Cancel
Load Settings...
Save Settings...
Save settings.
Save
Save in: Gait2354_Simbody
Specify name of settings file.

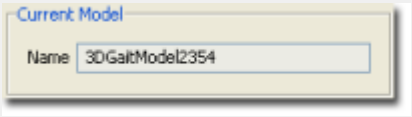
- The **Settings >** button is used to load or save settings for the tool.
- The **Run** button starts execution.
- The **Close** button closes the window.

Note that the **Close** button can be clicked immediately after execution has begun; the execution will complete even though the window has been closed. The **Cancel** button closes the window and cancels all operations that have not yet been completed.

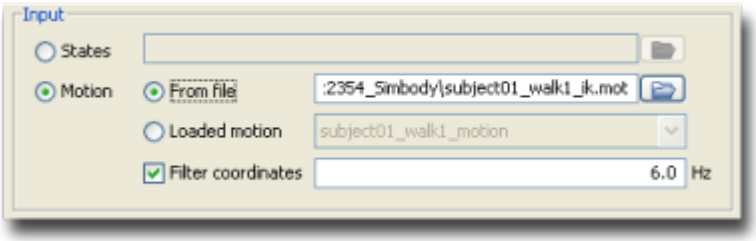
- When the **Settings >** button is clicked, you are presented with the choice of loading or saving settings for the tool, as shown in the figure on the left. This feature can be very useful as most tools require many parameters to be set before they can be run.
- If you click **Load Settings…**, you will be presented with a file browser that displays all files ending with the **.xml** suffix. You may browse for an appropriate settings file (e.g., subject01_setup_InverseDynamics.xml) and click **Open**. The Inverse Dynamics Tool will then be populated with the settings in that setup file.
- If you have manually entered or modified settings, you may save those settings to a file for future use. If you click **Save Settings…**, a Save dialog box will come up in which you can specify the name of the settings file. The name you specify for the file should have a suffix of **.xml**. Click **Save** to save the settings to file.

Main Settings Pane

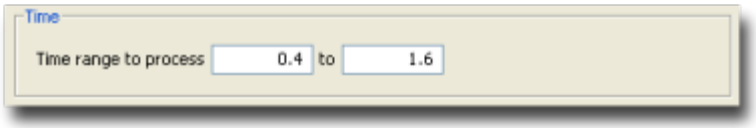
The *Main Settings* pane is used to specify parameters relating to the input kinematics of the current model, the time range for the analysis, and the output of the results. The pane is organized into four main sections entitled *Current Model, Input, Time,* and *Output*.



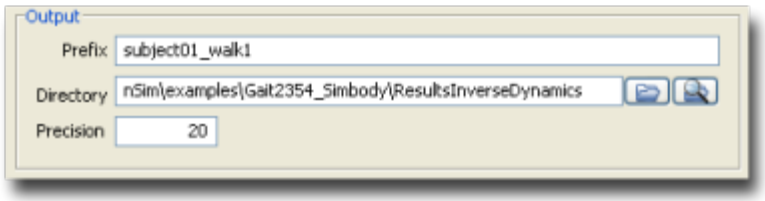
- The section for *Current Model* displays an uneditable name for the current model that is to be used for the inverse dynamics analysis



- The section for *Input* displays editable information that allows you to specify the kinematics (e.g., states or motion) describing the movement of a model.
- You may use the radio button to select either **States** or **Motion** as the input type.
- You may use the button to browse for the associated input file.
- If you select the radio button next to **Loaded motion**, you will need to choose a motion from the subject01_walk1_motion drop down list.



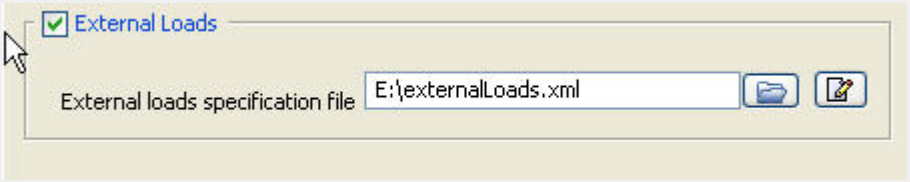
- The section for *Time* displays editable information that allows you to specify the start and end time for the inverse dynamics analysis



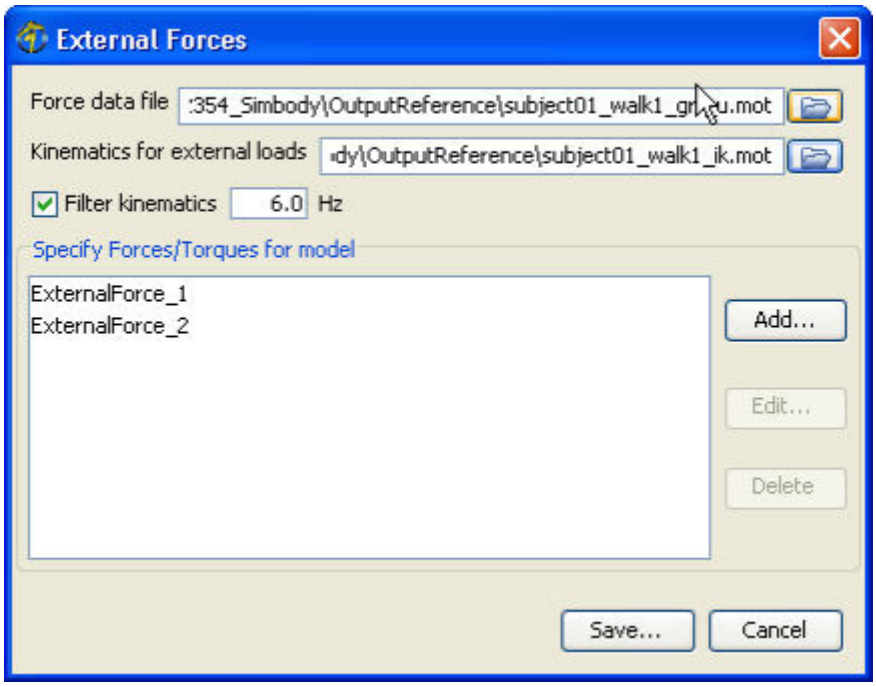
- The section for *Output* displays editable information that allows you to specify the prefix appended to the resulting output file, the directory to which the file is saved, and the precision of the decimal places used when writing results.
- You may use the button to browse for and specify a directory in which to save the output files.
- You may use the button to open an Explorer window to the specified directory.

External Loads Specification

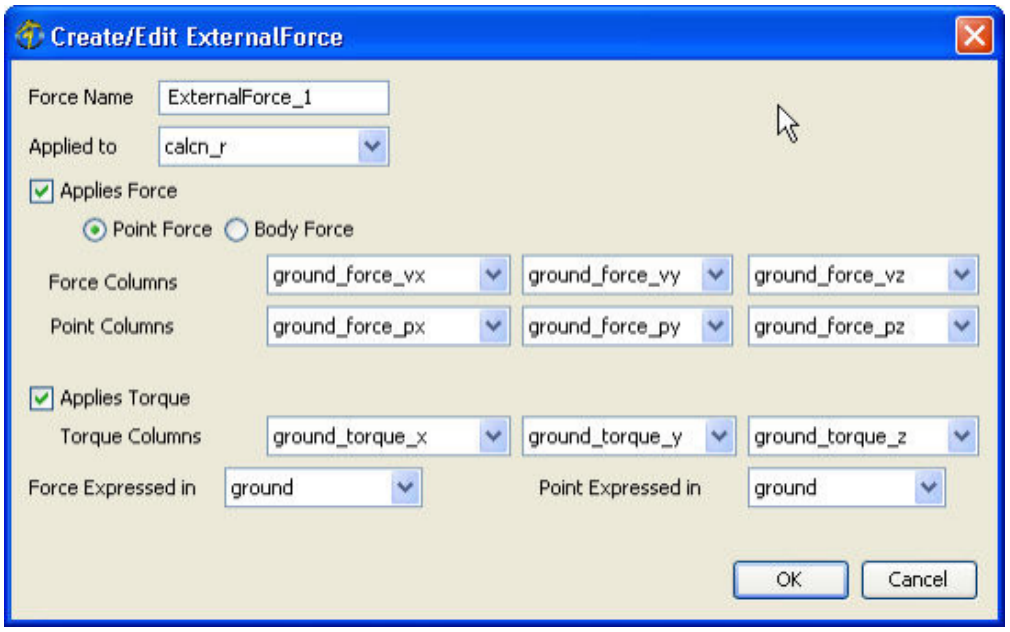
The *External Loads* pane (is used to specify parameters relating to the external loads applied to the model during inverse dynamic analysis or in other tools (e.g. Forward Dynamics, or Computed Muscle Control, AnalyzeTool etc.). The section for *External Loads* is optional, and if checked, displays information that allows you to specify the external loads applied to the model and the corresponding kinematics of the external loads if needed. Additionally, there is an option to filter the kinematics for the external loads by selecting the check box next to **Filter kinematics** and entering the filter frequency.



- The file specified under **External loads force specification file** is an XML file that contains an ExternalLoads object.
- The file can be created or edited by clicking the **editor button**  .



- The ExternalLoads editor dialog window shown on the left will appear when the **editor button**  is clicked.
- The **Force data file** specified at the top of this dialog window is a *.mot* or *.sto* file containing experimental data. The labels in this file have to be unique so that there's no ambiguity regarding which columns are used to define the forces.
- In case the column labels are not unique, the OpenSim GUI will detect that and offer the option to overwrite or write a new file with unique labels. Earlier versions of OpenSim allowed for duplicate column names but this behavior has been deprecated.
- Underneath you can specify the **Kinematics for external loads** file, filtering options for this file, and a list of **ExternalForce(**s) and/or torques that have been defined so far. In general, you should leave this blank. OpenSim will then use the specified kinematics that are being tracked by ID, RRA, and CMC (e.g. for ID for RRA it would be IK kinematics, and for CMC it would be RRA kinematics), which is generally more accurate.
- You must select **Save…** after you define your forces in order to write them to the XML file that will be used by the tool.



- You can select any of the listed forces and/or torques for editing or deletion by choosing it and then clicking on the **Edit…** or **Delete** buttons, respectively.
- Click on the **Add…** button to add a new force and/or torque to the list.
- Choosing the **Add…** button opens the dialog window (Create/Edit ExternalForce) shown in the figure on the left.

Within the Create/Edit ExternalForce window, you can define either a **Point Force** or a **Body Force**.

- A **point force** (most commonly used) is a force applied at a specified point and an optional torque. Using the drop down menu at the bottom of the window, you can specify the reference frames that the point and the force are expressed in.
- A **body force** is a force applied to the origin frame of the body (not necessarily the CoM) and a required torque, for a total of six components. By definition, a torque is a rotational "force" or moment so the point of application does not need to be specified. Thus for a body force, you should check applies force and applies torque. The selections in "Point Columns", "Force Expressed In", and "Point Expressed In" will be ignored.

The drop-down menus for **Force Columns**, **Point Columns**, and **Torque Columns** allow you to specify which of the columns in the **Force Data file** are to be used to define the ExternalForce. Columns in the Force data file corresponding to the same force are assumed to live in adjacent columns and to have a common prefix that's used as an identifier for the corresponding ExternalForce. This common prefix is stored in the xml file as a "**force_identifier**", similar rules apply for the "**point_identifier**" in case a point needs to be specified and "**torque_identifier**" for the case when a torque is also applied. Other attributes of the ExternalForce are:

- applied_to_body: specifies the name of the body that the ExternalForce is applied to.
- Force Expressed In: specifies the frame in which the force is expressed (e.g. Force plate data is usually measured in lab/ground frame)
- Point Expressed In: specifies the frame in which the point is expressed

Command-line Execution

The Inverse Dynamics Tool is run using the command **analyze -S <setup file name>,** for example,

```
analyze -S subject01_Setup_InverseDynamics.xml
```

Next: ID Settings Files and XML Tags

Previous: How Inverse Dynamics Works

Home: Inverse Dynamics

ID Settings Files and XML Tags

The topics covered in this section include:

Setup File and XML Tag Definitions

The settings file is an XML file whose tags specify properties to be used by OpenSim for the inverse dynamics analysis. The XML tags used are defined in the following sections.

Note: The following setup file example can be found in examples/Gait2354_Simbody.
Example: XML file for an inverse dynamics setup file

subject01_Setup_InverseDynamics.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSimDocument Version="20302">
   <InverseDynamicsTool name="subject01_walk1">

      <!--Directory used for writing results.-->
      <results_directory> ResultsInverseDynamics </results_directory>

      <!--Name of the .osim file used to construct a model.-->
      <model_file> subject01_simbody.osim </model_file>

      <!--Time range over which the inverse dynamics problem is solved.-->
      <time_range>       0.40000000       1.60000000 </time_range>

      <!--List of forces by individual or grouping name (e.g. All, actuators,
          muscles, ...) to be excluded when computing model dynamics.-->
      <forces_to_exclude> Muscles </forces_to_exclude>

      <!--XML file (.xml) containing the external loads applied to the model as
          a set of ExternalForce(s).-->
      <external_loads_file> subject01_walk1_grf.xml </external_loads_file>

      <!--The name of the file containing coordinate data. Can be a motion
          (.mot) or a states (.sto) file.-->
      <coordinates_file> subject01_walk1_ik.mot </coordinates_file>

      <!--Low-pass cut-off frequency for filtering the coordinates_file data
          (currently does not apply to states_file or speeds_file). A negative
          value results in no filtering. The default value is -1.0, so no
          filtering.-->
      <lowpass_cutoff_frequency_for_coordinates>       6.00000000
</lowpass_cutoff_frequency_for_coordinates>

      <!--Name of the storage file (.sto) to which the results should be
          written.-->
      <output_gen_force_file> inverse_dynamics.sto </output_gen_force_file>

   </InverseDynamicsTool>
</OpenSimDocument>
```

Execution Name

The properties for the analyze tool, which is responsible for performing the inverse dynamics analysis, are enclosed inside the opening and closing tags <InverseDynamicsTool> and </ InverseDynamicsTool >. The name attribute name="subject01_walk1" specifies the execution name. The name of the results file generated will be prefixed with this name.

Model

The <model_file> tag specifies the OpenSim .osim file used to construct a model. This file typically defines the OpenSim musculoskeletal model scaled to the dimensions of the subject by the Scale Tool.

Forces to Exclude/Include

The <forces_to_exclude> allows the modification of the forces defined in the model file for the purpose of InverseDynamics. Forces can be specified either individually or as groups, the default behavior is to exclude "Muscles" and leave all other forces (e.g. Bushing forces or contact) in effect.

Results Directory and Precision

The <results_directory> tag specifies the directory where results should be written.

Initial and Final Times

The <time_range> tag specifies the time interval over which the inverse dynamics analysis is to be performed. The initial and final times may be adjusted to the nearest frames of data available.

Coordinates and Filtering

The <coordinates_file> tag specifies the time histories of kinematics (e.g., generalized coordinates) that describe the movement of the model. For example, this file may be generated by the inverse kinematics tool. The <lowpass_cutoff_frequency_for_coordinates> tag specifies the low-pass cutoff frequency for filtering the kinematics. If the kinematics have already been filtered, specify a negative cutoff frequency to prevent filtering.

External Loads

The <external_loads_file> tag specifies the XML file containing the external loads applied to the model during an inverse dynamics analysis. Detailed description of the contents of this file are given above.

Next: Static Optimization

Previous: How to Use the Inverse Dynamics Tool

Home: Inverse Dynamics

Static Optimization

In this section we will cover:

- Getting Started with Static Optimization
- How Static Optimization Works
- How to Use the Static Optimization Tool
- Static Optimization Settings Files and XML Tags

Next: Getting Started with Static Optimization

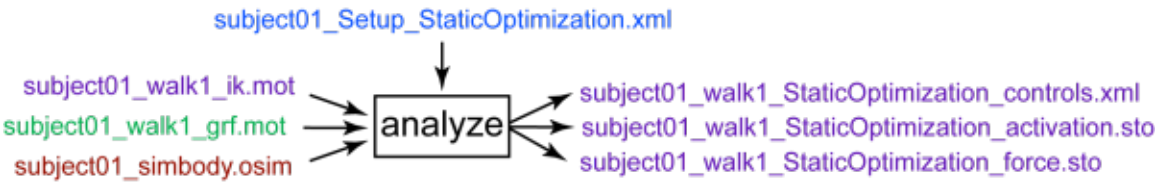
Getting Started with Static Optimization

Static optimization is an extension to inverse dynamics that further resolves the net joint moments into individual muscle forces at each instant in time. The muscle forces are resolved by minimizing the sum of squared (or other power) muscle activations.

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

Overview

The figure below shows the required inputs and outputs for the Static Optimization Tool. Each is described in more detail in the following sections:



Inputs and Outputs of the Static Optimization Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple. To run static optimization, you use the analyze command.

> The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Inputs

Three files are required as input by the Static Optimization Tool:

subject01_walk1_ik.mot: Motion file containing the time histories of generalized coordinates that describe the movement of the model. This can be kinematic data (i.e., joint angles) from IK or states (i.e., joint angles AND velocities) from RRA and the time range of interest.

subject01_walk1_grf.xml: External load data (i.e., ground reaction forces, moments, and center of pressure location). Note that you must measure or model all external forces acting on a subject during the motion to calculate accurate muscle forces. The xml file describes how to apply the measured ground reaction forces to the model during the analysis.

subject01_simbody.osim: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters (segment masses, etc.).

x: The exponent for the activation-based cost function, to be minimized (i.e., the criteria used to solve muscle force distribution problem).

Outputs

The Static Optimization Tool generates three files in a specified folder:

subject01_walk1_StaticOptimization_controls.xml: Contains the time histories of muscle activations. These controls were minimized by the Static Optimization Tool.

subject01_walk1_StaticOptimization_activation.sto: Storage file containing the time histories of muscle activations.

subject01_walk1_StaticOptimization_force.sto: Storage file containing the time histories of muscle forces.

Best Practices and Troubleshooting

Static Optimization Settings

1. You can use IK or RRA results as input kinematics. If using IK results, you usually need to filter them, either externally or using the OpenSim analyze/static optimization field. If using RRA results, you usually do not have to filter.
2. For gait and many other motions, you need to add (append) residual actuators to the first free joint in the model (typically the ground-pelvis joint).
    a. There should be one actuator for each degree of freedom (e.g. FX, FY, FZ, MX, MY, MZ).
    b. These residual actuators are required because there is dynamic inconsistency between the estimated model accelerations and the measured ground reaction forces. This inconsistency can result from marker measurement error, differences between the model and subject's geometry and intertial parameters.
    c. Running RRA will reduce, but not eliminate these residuals, thus appending actuators is still necessary.
3. See How Static Optimization Works and How to Use the Static Optimization Tool for more information.

Troubleshooting

1. If the residual actuators or the model's muscles are weak, the optimization will take a long time to converge or will never converge at all.
    a. If the residual actuators or weak, increase the maximum control value of a residual, while lowering its maximum force. This allows the optimizer to generate a large force (if necessary) to match accelerations but large control values are penalized more heavily. In static optimization, ideal actuator excitations are treated as activations in the cost function.
    b. If the muscles are weak, append Coordinate Actuators to the model at the joints in the model. This will allow you to see how much "reserve" actuation is required at a given joint and then strengthen the muscles in your model accordingly.
    c. If troubleshooting a weak model and each time, optimization is slow, try reducing the parameter that defines the max number of iterations.
2. StaticOptimization works internally by solving the InverseDynamics problem, then trying to solve the redundancy problem for actuators/muscles using the accelerations from the InverseDynamics solution as a constraint. If you a constraint violation is reported, this could be a sign that the optimizer couldn't solve for muscle forces while enforcing the InverseDynamics solution.
    a. This likely means that there is noise in the data or there is a sudden jump in accelerations in one frame.
    b. In this case you should examine the Inverse Dynamics solution to examine the problematic frame, and fix/interpolate the data during this portion of the motion.

Evaluating your Results

1. Are there any large or unexpected forces residual actuator forces?
2. Find EMG or muscle activation data for comparison with your simulated activations. Does the timing of muscle activation/deactivation match? Are the magnitudes and patterns in good agreement?

Next: How Static Optimization Works

Home: Static Optimization

How Static Optimization Works

As described in Inverse Dynamics, the motion of the model is completely defined by the generalized positions, velocities, and accelerations. 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 muscle activation-to-force conditions:

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

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.

Next: How to Use the Static Optimization Tool

Previous: Getting Started with Static Optimization

Home: Static Optimization

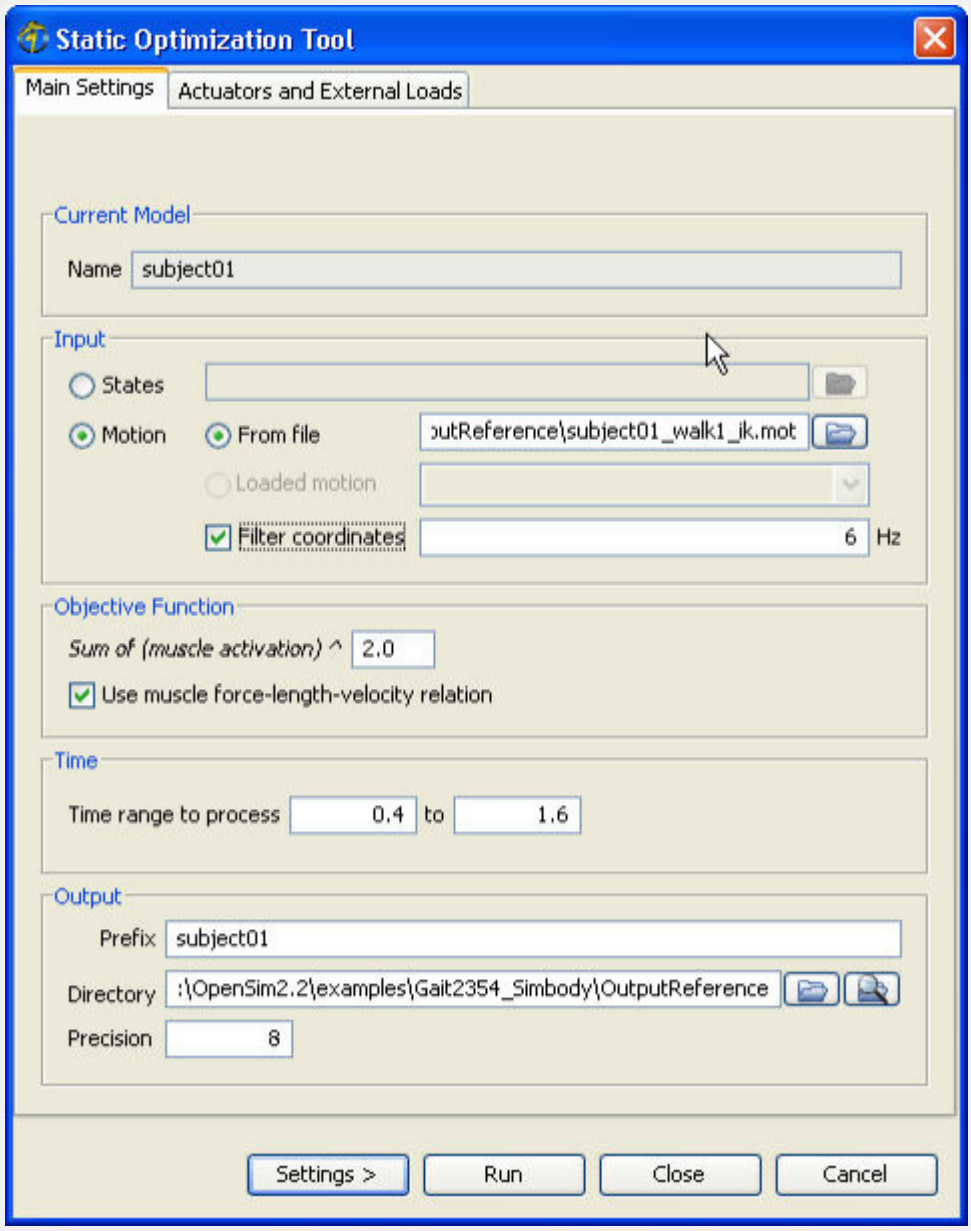
How to Use the Static Optimization Tool

The topics covered in this section include:

How to Use the GUI

To launch the Static Optimization Tool, select **Static Optimization…** from the **Tools** menu. The *Static Optimization Tool* dialog window (figure below), like all other OpenSim tools, operates on the current model open and selected in OpenSim (e.g., *subject01*). The Static Optimization Tool is controlled by a dialog box with two tabbed panes. The *Main Settings* pane specifies parameters related to the input kinematics of the current model, the time range for the analysis, and the output of the results. The *External Loads* pane specifies parameters related to the external loads applied to the model during the analysis. See Inverse Dynamics for additional details about the *External Loads* pane. The *Main Settings* pane is organized into five main sections:

Dialog Box for the Static Optimization Tool. The Main Settings pane.



- ***Current Model*:** Displays an uneditable name for the current model being used for the static optimization analysis.
- ***Input*:** Displays editable information specifying the kinematics (e.g., states or motion) describing the movement of a model.
- ***Objective Function***: Displays editable information specifying the power to which the muscle activations should be raised and whether or not to use the muscle force-length-velocity properties.
- ***Time***: Displays editable information specifying the start and end time for the Static Optimization analysis.
- ***Output***: Displays editable information specifying the prefix appended to the resulting output file, the directory to which the file is saved, and the precision (number of decimal places) used when writing results. You may use the button to browse for a directory to save the output files, and the button to open an Explorer window for the specified directory.

Command-line Execution

The Static Optimization Tool can also be run using the command **analyze -S <setup file name>,** for example,

```
analyze -S subject01_Setup_StaticOptimization.xml
```

Next: Static Optimization Settings Files and XML Tags

Previous: How Static Optimization Works

Home: Static Optimization

Static Optimization Settings Files and XML Tags

The topics covered in this section include:

Setup File and XML Tag Definitions

The settings file is an XML file whose tags specify properties to be used by OpenSim for the static optimization analysis. The XML tags used are defined in the following sections.

The following setup file example can be found in examples/Gait2354_Simbody.

Example: XML file for a static optimization setup file

subject01_Setup_StaticOptimization.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<AnalyzeTool name="subject01_walk1">

  <!--Name of the .osim file used to construct a model.-->
  <model_file> subject01_simbody.osim </model_file>

  <!--Replace the model's actuator set with sets specified in <force_set_files>? If false, the
actuator set is appended.-->
  <replace_force_set> false </replace_force_set>

  <!--List of xml files used to construct an actuator set.-->
  <force_set_files> </force_set_files>

  <!--Directory used for writing results.-->
  <results_directory> ResultsStaticOptimization </results_directory>

  <!--Output precision. It is 8 by default.-->
  <output_precision> 20 </output_precision>

  <!--Initial time for the simulation.-->
  <initial_time> 0.4 </initial_time>

  <!--Final time for the simulation.-->
  <final_time> 1.6 </final_time>

  <!--Set of analyses to be run during the investigation.-->
  <AnalysisSet name="Analyses">
    <objects>
      <StaticOptimization name="StaticOptimization">
        <on> true </on>
        <step_interval> 1 </step_interval>
        <use_model_force_set> false </use_model_force_set>
      </StaticOptimization>
    </objects>
  </AnalysisSet>

  <!--Motion file (.mot) containing the generalized coordinates for the model.-->
  <coordinates_file> subject01_walk1_ik.mot </coordinates_file>

  <!--Low-pass cut-off frequency for filtering the model generalized coordinates. A negative value
results in no filtering. The
      default value is 1.0, so no filtering.-->
  <lowpass_cutoff_frequency_for_coordinates> 6 </lowpass_cutoff_frequency_for_coordinates>

  <!--XML file (.xml) containing the forces applied to the model as ExternalLoads -->
  <external_loads_file> externalLoads.xml</external_loads_file>

</AnalyzeTool>
```

Specifying an Execution Name

The properties for the analyze tool, which is responsible for performing the inverse dynamics analysis, are enclosed inside the opening and closing tags <AnalyzeTool> and </AnalyzeTool>. The name attribute name="subject01_walk1" specifies the execution name. The name of the results file generated will be prefixed with this name.

Specifying the Model

The <model_file> tag specifies the OpenSim .osim file used to construct a model. This file typically defines the OpenSim musculoskeletal model scaled to the dimensions of the subject by the Scale Tool.

Specifying the Forces

The <replace_force_set> allows the modification of the actuators defined in the model file. If the value of <replace_force_set> is **true**, then the

actuators specified in any file listed under the property <force_set_files> will replace the corresponding model's actuators. If the value of <replace_force_set> is **false**, then the actuators specified in the files listed under <force_set_files> will be added to the model's existing actuator set.

Specifying the Results Directory and Precision

The <results_directory> tag specifies the directory where results should be written. The <output_precision> tag specifies how many decimal places should be used when writing results. A value of 20 is sufficient to avoid round-off error.

Specifying Initial and Final Times

The <initial_time> and <final_time> tags specify the time interval over which the inverse dynamics analysis is to be performed. The initial and final times may be adjusted to the nearest frames of data available.

Specifying a Static Optimization Analysis

The properties for the analysis set used by the analyze tool are enclosed inside the opening and closing tags <AnalysisSet> and </AnalysisSet>. The properties for the objects contained in the analysis set are enclosed inside the opening and closing tags <objects> and </objects >.

The properties for the static optimization analysis are enclosed inside the opening and closing tags <StaticOptimization> and </StaticOptimization >. The name attribute name="StaticOptimization" specifies the analysis name. The name of the results file generated will include this name. The <on> tag specifies whether or not the static optimization analysis should be performed. The <step_interval> tag specifies how often to record results during the static optimization analysis. A value of 1 means results are recorded every 1 frame of motion data. The <use_model_force_set> tag specifies whether or not the model's actuator set will be used in the static optimization analysis. If not, generalized forces (e.g., net joint forces and torques) will be computed for all unconstrained degrees of freedom.

Specifying Coordinates and Filtering

The <coordinates_file> tag specifies the time histories of kinematics (e.g., generalized coordinates) that describe the movement of the model. For example, this file may be generated by the inverse kinematics tool. The <lowpass_cutoff_frequency> tag specifies the low-pass cutoff frequency for filtering the kinematics. If the kinematics have already been filtered, specify a negative cutoff frequency to prevent filtering.

Specifying External Loads

The <external_loads_file> tag specifies the XML file containing the external loads applied to the model during a static optimization analysis. See Inverse Dynamics for additional details for applying any number of external loads as point forces and/or body torques.

Next: Forward Dynamics

Previous: How to Use the Static Optimization Tool

Home: Static Optimization

Forward Dynamics

In this section we will cover:

- Getting Started with Forward Dynamics
- How Forward Dynamics Works
- How to Use the Forward Dynamics Tool
- Forwards Dynamics Setup Files and XML Tags

Next: Getting Started with Forward Dynamics

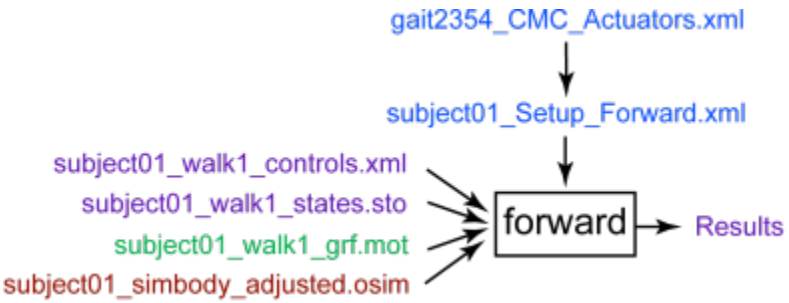
Getting Started with Forward Dynamics

Given the controls (e.g., muscle excitations) computed by the Computed Muscle Control (CMC) or another approach, the Forward Dynamics Tool can drive a forward dynamic simulation. A forward dynamics simulation is the solution (integration) of the differential equations that define the dynamics of a musculoskeletal model. By focusing on specific time intervals of interest, and by using different analyses, more detailed biomechanical data for the trial in question can be collected.

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.

Overview

The figure shows the required inputs and outputs for the Forward Dynamics Tool. Each is described in more detail in the following sections.



Inputs and Outputs of the Forward Dynamics Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

> The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Inputs

Four data files are required as input by the Forward Dynamics Tool:

subject01_walk1_controls.xml: Contains the time histories of the model controls (e.g., muscle excitations) to the muscles and/or joint torques. It is possible to specify the controls as .sto file instead with columns corresponding to desired excitations. This file may be generated by the user, Static Optimization Tool, or Computed Muscle Control Tool. If no controls are provided they are assumed to be zero for any actuators in the model.

subject01_walk1_states.sto: Contains the time histories of model states, including joint angles, joint speeds, muscle activations, muscle activations, muscle fiber lengths, and more. These states are used by the Forward Dynamics Tool to set the initial states of the model for forward integration. Alternately, the simulation can begin from the default pose of the model without providing initial states. Muscle states can be estimated by solving for muscle-fiber and tendon force equilibrium when the *Solve for equilibrium for actuator states* is checked.

subject01_walk1_grf.xml: An xml file describing the External Loads applied based (for example) on measured ground reaction forces that should be applied to the model during simulation

subject01_simbody_adjusted.osim: Subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters (segment masses, etc.).

The **subject01_Setup_Forward.xml** file is the setup file for the Forward Dynamics Tool. It contains settings, as described in How to Use the Forward Dynamics Tool, and refers to another settings file, **gait2354_CMC_Actuators.xml** which contains a set of actuators that supplement the muscles of the model. Refer to Computed Muscle Control for more details. These actuators must be included in the forward simulation so that the CMC solution can be reproduced.

Outputs

The Forward Dynamics tool generates results in a folder specified in the setup file:

Results: Additional data can be generated and written to files by adding analyses to the Forward Dynamics Tool. These analyses are specified in

the setup file (**subject01_Setup_Forward.xml**) and are discussed in the Analyses section.

Best Practices and Troubleshooting Tips

1. Forward dynamics simulations are sensitive to initial conditions and it is good practice to double check that they your ICs appropriate for the desired simulation.
2. If the Forward Tool fails gracefully (i.e., without crashing OpenSim) or the output of the Forward Tool drifts too much (i.e., the model goes crazy), shorten the interval over which the Forward Tool runs (i.e., make initial_time and final_time closer to each other in the Forward Tool setup dialog box or setup file). Open-loop forward dynamics tends to drift over time due to the accumulation of numerical errors during integration.

Next: How Forward Dynamics Works

Home: Forward Dynamics

How Forward Dynamics Works

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:

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

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

$$\tau_m = \left[R(q)\right] f(a,l,\dot{l})$$ Moments due to muscle forces

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

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

The net muscle moments, τ_m, in turn, are a result of the moment arms, $R(q)$, multiplied by muscle forces, f, which are a function of muscle activations, a, and muscle fiber lengths, l, and velocities, $\dot{l}$. 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, $\dot{a}$, of the muscles respond to input neural excitations, x, generally termed the model's controls. These form a set of differential equations that model *musculoskeletal dynamics*.

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 through time. 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.

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.

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 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.

The Forward Tool and CMC

The Forward Dynamics Tool uses the same model and actuator set used in CMC, together with the initial states and controls computed during the

CMC step, to run a muscle-driven forward dynamic simulation that aims to reproduce the same motion tracked by CMC. The input of initial states is optional. If not entered, the model assumes default values for the state variables. Similarly, if no controls are specified as input then the integration proceeds with zero value for the controls.

As in CMC and RRA, a 5^{th} order Runge-Kutta-Feldberg integrator is used. In contrast to CMC, which used PD controllers in a closed-loop system to ensure tracking of the desired trajectories, the Forward Dynamics Tool is an open-loop system. That is, it blindly applies the recorded actuator controls with no feedback or correction mechanism to help ensure accurate tracking. In theory, starting forward with the exact same conditions as CMC, and feeding it the exact same controls computed by CMC, it should reproduce the same trajectory computed during CMC. However, even tiny differences in values (due to truncation or round-off) or in the ways these values are used by the Forward Dynamics Tool as compared to CMC will cause the forward simulation to diverge from the expected trajectory. This is particularly a problem for longer simulations, in which small differences have more time to accumulate, and in which divergences can become more noticeable, eventually causing the simulation to become completely unstable.

The discrepancies between the Forward Dynamics Tool and CMC can be reduced in a number of ways, as was done with the OpenSim example subjects gait2354 and gait2392:

- During the CMC step, it is recommended that the output precision be set to 20 to ensure that the values read in and used by the Forward Dynamics Tool match the values from CMC with a high precision.
- The Forward Dynamics Tool sets the model's initial state to the CMC state values corresponding to the Forward Dynamics Tool's initial time. If you set the Forward Dynamics Tool to start at a time for which no CMC state values are available, the initial time is adjusted to the nearest time that does have state values. This is done automatically.
- Through a property in the Forward Dynamics Tool's setup file (<use_specified_dt>), you can also use the same integration time steps as were used in CMC. It is recommended that this property be set to **true** to minimize the divergence of long forward simulations. If set to **false**, the Forward Dynamics Tool's integrator adaptively computes its own time steps based on the desired set error tolerances. As these time steps tend to differ from those used in CMC, the results usually diverge faster.

Next: How to Use the Forward Dynamics Tool

Previous: Getting Started with Forward Dynamics

Home: Forward Dynamics

How to Use the Forward Dynamics Tool

Error: RuntimeException occurred while performing an XHTML storage transformation (null)

Forwards Dynamics Setup Files and XML Tags

The topics covered in this section include:

Setup File and XML Tag Definitions

These sample XML files are from the examples/Gait2354_Simbody directory and are part of the OpenSim distribution.

Example: XML file for the setup file for forward dynamics

subject01_Setup_Forward.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSimDocument Version="20302">
<ForwardTool name="subject01_walk1">

  <!-- OpenSim Model -->
  <model_file> subject01_adjusted.osim </model_file>
  <force_set_files> gait2354_CMC_Actuators.xml </force_set_files>
  <replace_force_set> false </replace_force_set>

  <!-- Times over which to run forward -->
  <initial_time> 0.80 </initial_time>
  <final_time> 1.18 </final_time>

  <!-- Integrator Settings -->
  <maximum_number_of_integrator_steps> 30000 </maximum_number_of_integrator_steps>
  <maximum_integrator_step_size> 0.00025 </maximum_integrator_step_size>
  <integrator_error_tolerance> 5e-006 </integrator_error_tolerance>
  <use_specified_dt> true </use_specified_dt>

  <!--Analyses and Results -->
  <results_directory> ./ResultsForward </results_directory>
  <output_precision> 20 </output_precision>
  <AnalysisSet name="Analyses">
    <objects>
      <Kinematics name="Kinematics">
         <on> true </on>
        <step_interval> 10 </step_interval>
        <in_degrees> true </in_degrees>
      </Kinematics>

      <Actuation name="Actuation">
        <on> true </on>
        <step_interval> 10 </step_interval>
        <in_degrees> true </in_degrees>
      </Actuation>

      <BodyKinematics name="BodyKinematics">
        <on> true </on>
        <step_interval> 10 </step_interval>
        <in_degrees> true </in_degrees>
      </BodyKinematics>
    </objects>
  </AnalysisSet>


  <!-- Initial states and controls -->
  <states_file> ResultsCMC/subject01_walk1_states.sto</states_file>
  <ControllerSet name="Controllers">
    <objects>
      <ControlSetController name="">
        <!--A list of actuators that this controller will control. The keyword ALL indicates the
controller will control all the acuators in the model-->
        <actuator_list> </actuator_list>
        <!--Flag (true or false) indicating whether or not the controller is enabled (ON) should-->

        <enable_controller> ALL </enable_controller>
        <!--XML file containing the controls for the controlSet.-->
        <controls_file> ResultsCMC/subject01_walk1_controls.xml </controls_file>
      </ControlSetController>
    </objects>
  </ControllerSet>

  <!-- External Loads -->
  <external_loads_file> subject01_walk1.xml</external_loads_file>

</ForwardTool>
</OpenSimDocument>
```

Specifying an Execution Name

The properties for the Forward Dynamics Tool are enclosed inside the opening and closing tags <ForwardTool> and </ForwardTool>. The name attribute name="subject01_walk1" specifies the execution name. The names of results files generated will be prefixed with this name.

Specifying the Model

The model specification for the Forward Dynamics Tool must match that used for CMC. Visit Computed Muscle Control for more information.

Specifying Initial and Final Times

The properties <initial_time> and <final_time> specify the time interval over which a forward simulation is to be run. As explained in How Forward Dynamics Works, the initial time may be adjusted to the nearest time for which state values are available, as these state values (typically from CMC) will be used to initialize the forward simulation.

Specifying Integrator Settings

As explained in How Forward Dynamics Works, setting <use_specified_dt> to **true** (which is recommended) ensures that the Forward Dynamics Tool uses the same integrator time steps as that for CMC. It does this by matching the time steps found in the <initial_states_file>. In this case, the values set for the remaining properties (<maximum_number_of_integrator_steps>, <maximum_integrator_step_size>, <integrator_error_tolerance>, and <integrator_fine_ tolerance>) are ignored. If <use_specified_dt> is set to **false** then the integrator properties are used to determine the adaptive time step sizes (see Chapter 4 on CMC chapter for how these are used).

Specifying Analyses and Results

Any number of available analyses can be added to a forward dynamics run. To get a listing of available analyses, run the command **forward -PropertyInfo**.
There are several properties associated with the analyses results. <results_directory> specifies the directory where results should be written. <output_precision> specifies how many decimal places should be used when writing results. A value of 20 is sufficient to avoid round-off error when reading results back in during other steps in the workflow. <step_interval> specifies how often to record results during numerical integration. A value of 10 means record results every 10 integration steps.

Specifying Initial States and Controls

Running a forward simulation requires that the states be set to initial values at the beginning of the simulation, and that the actuator control values (e.g., muscle excitations) are updated throughout the simulation. Both of these are optional with default behavior as explained in How Forward Dynamics Works. Often, the initial states and the control values used to drive the forward simulation are those computed during the CMC step, in these cases the <initial_states_file> property should be set to the states output from CMC (an .sto storage file), and <controls_file> should be set to the controls output from CMC (the .xml file, not the controls .sto file that is also written by CMC). The <initial_states_file> is used to (a) determine a valid initial time for the forward simulation (it has to coincide with a time for which state values are available), (b) supply the initial state values, and (c) if <use_specified_dt> is set to **true**, then it is also used to determine the integration time intervals used by CMC (in order to match CMC results as much as possible).

Specifying External Loads

The file containing the external loads applied to the model during a simulation is specified using the property <external_loads_file>. Please refer to section Forwards Dynamics Setup Files and XML Tags for details about the contents of this file.

Across all steps in the OpenSim workflow, it is important to use the same settings for specifying the external loads.

Next: Residual Reduction Algorithm

Previous: How to Use the Forward Dynamics Tool

Home: Forward Dynamics

Residual Reduction Algorithm

The topics covered in this section are:

Next: Getting Started with RRA

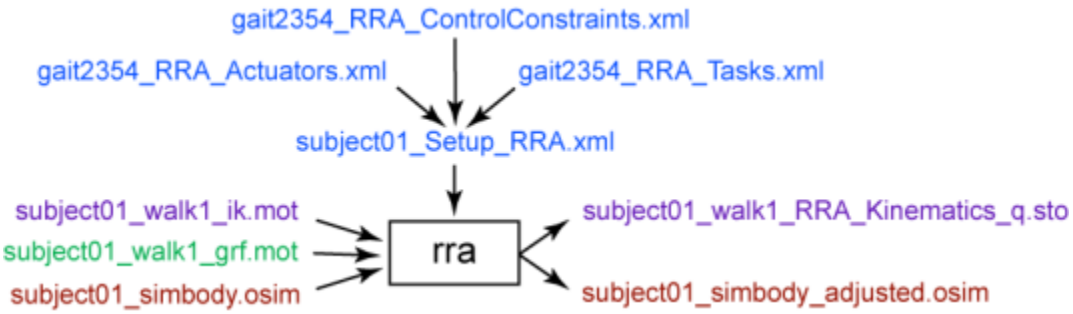
Getting Started with RRA

The purpose of Residual Reduction is to minimize the effects of modeling and marker data processing errors that aggregate and lead to large nonphysical compensatory forces called residuals. Specifically, residual reduction alters the torso mass center of a subject-specific model and permits the kinematics of the model from inverse kinematic to vary in order to be more dynamically consistent with the ground reaction force data.

The residual reduction algorithm tool is accessed by selecting **Tools Residual Reduction Algorithm…** from the OpenSim main menu bar. Like all tools, the operations performed by the computed muscle control tool apply to the current model.

Overview

The figure below shows the required inputs and outputs for performing the residual reduction algorithm. Each is described in more detail below.



Inputs and Outputs for performing residual reduction. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

Settings File

The **subject01_Setup_RRA.xml** file is a setup file for the RRATool, which specifies settings, inputs, and outputs that affect the behavior of the residual reduction algorithm, which can be defined using the GUI or by hand. Details of the settings are described in the section on using the Graphical User Interface.

The setup file identifies the actuators (i.e., the ideal residual and reserve joint actuators required by RRA) as well as the kinematic tracking tasks. Furthermore, control constraints on the actuators (to limit the maximum residual force) can be specified.

Inputs

Several files are required as input to the RRA Tool to perform residual reduction:

subject01_walk1_ik.mot: Contains the time histories of model kinematics including the joint angles and pelvis translations.

gait2345_RRA_Tasks.xml: A tracking tasks file specifying which coordinates to track and the corresponding tracking weight (weights are relative and determine how "well" a joint angle will track the specified joint angle from IK). A couple key considerations:

1. Selection of kp and kv are not arbitrary. They define the behavior of the error dynamics for each q as a second order linear system. We can write the kp and kv for the desired system behavior in terms of system poles. For a (stable) critically damped system (real negative poles) kp = lambda^2 and kv = -2*lambda.
2. This enables kinematics of joints (coordinates) for which we have high confidence (e.g. knee flexion, hip flexion) to be weighted more heavily compared to those of less confidence (e.g. hip internal rotation and ankle inversion).

gait2345_RRA_ControlConstraints.xml: Contains limits on the RRA actuators. The actuator constraints file specifying the maximum and minimum "excitation" (i.e., control signal) for each actuator. A few key considerations:

1. Note that the maximum/minimum force or torque generated by an ideal actuator is the product of the max/min force and max/min excitation.
2. Joint torques (and muscles) have a maximum magnitude of 1.
3. Residuals have bounds exceeding their anticipated force requirement. Weightings are implicit in this description. A high optimal_force means that large output force (torque) does not require a large control value (i.e. low cost). Conversely, residuals with low optimal force require high control values that incur higher costs.

subject01_walk1_grf.xml: ExternalLoads file specifying the measured ground reaction forces that should be applied to the model during simulation and how to apply them.

subject01_simbody.osim: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters.

gait2345_RRA_Actuators.xml: Ideal joint actuators used to replace muscles. The Actuator Set specifies the residual and reserve actuators to be applied and their parameters, like maximum/minimum force and body or joint, or location, depending on the actuator type. A few key considerations:

1. Each degree of freedom in the model should have an ideal torque or force (reserve) actuator. This includes the 6 DOFs of the model's base segment, which are called the residual actuators.
2. In most circumstances, these Ideal joint actuators used to replace the muscles in the model (by checking "Replace model actuators" in the Actuators tab.
3. Optimal forces are the maximum output of ideal actuators (torques, linear forces). Torque (force) applied is optimal_force x control_value
4. Residual at pelvis should be applied at scaled location of COM

Outputs

Residual reduction generates the following outputs:

subject01_RRA_states.sto: Adjusted kinematics (i.e., joint angles) and corresponding model states of the simulated motion (i.e., joint angles AND velocities).

subject01_adjusted.osim (optional): A model with adjusted mass properties.

subject01_RRA_forces.sto: Actuator forces and torques (i.e., joint torques corresponding to adjusted kinematics).

subjcet01_RRA_controls.xml: Actuator excitations (i.e., control signals needed to generate actuator forces and torques)

Best Practices and Troubleshooting

RRA Settings

1. You should replace the muscles in your model with residual actuators and ideal joint actuators. Residual reduction is a form of forward dynamics simulation that utilizes a tracking controller to follow model kinematics determined from the inverse kinematics. Computed muscle control (CMC) serves as the controller, but without muscles the skeleton of the model can be used to determine a mass distribution and joint kinematics that are more consistent with ground reaction forces.
2. Optimal forces for residuals should be low to prevent the optimizer from "wanting" to use residual actuators (an actuator with large optimal force and low excitation is "cheap" in the optimizer cost).
3. To help minimize residuals, make an initial pass with default inputs, then check residuals and coordinate errors. To reduce residuals further, decrease tracking weights on coordinates with low error. You can also try decreasing the maximum excitation on residuals or the actuator optimal force.
4. Typically, you should lock" the subtalar and mtp joints in *.osim file.
5. Make sure “use_fast_optimization_target” is false (unchecked). This allows the kinematics to be slightly adjusted to account for dynamic inconsistencies. This is the default in settings files distributed with OpenSim or created from the GUI. See How CMC Works for a comparison of the "slow" and "fast" targets.
6. The “cmc_time_window” in the settings file should be 0.001 s for RRA. This is the default in settings files distributed with OpenSim or created from the GUI.
7. See How RRA Works and How to Use the RRA Tool for more information about RRA settings.

Troubleshooting

1. Check the pelvis COM location in Actuator files.
2. If RRA is failing, try increasing the max excitation for residuals by 10x until the simulation runs. Then try working your way back down while also "relaxing" tracking weights on coordinates.
3. If residuals are very large (typically, this is greater than 2-3x BW, depending on the motion), there is probably something wrong with either i) the scaled model, (ii) the IK solution, or (iii) the applied GRFs. To double check that forces are being applied properly, visualize GRFs with IK data (you can use the Previewing Motion Capture (Mocap) Data function in the GUI).
4. If there is pelvis drift and/or FY is not centered around zero, check that the body mass and force calibration are correct.
5. When using the example RRA actuators XML file, you should note that residual forces are applied to the center-of-mass (COM) of the unscaled pelvis. However, if you scale the model, the COM of the pelvis can change. Although the effect may be small, you should change the location of the residual force actuators in the your RRA actuators file to correspond to the scaled pelvis COM.

Evaluating your Results

1. RMS difference in joint angle during the movement should be less than 2-5º (or less than 2 cm for translations).
2. Peak Residual Forces should typically be less than 10-20 N. Average residuals should typically be less than 5-10 N.
    a. The size of residuals will depend on the type of motion being studied. For example residuals for high speed activities, like sprinting, will typically be larger than walking.
    b. Residuals will also be larger if there are external forces that you have not accounted for, like a subject walking with a handrail.

3. Compare the residual moments from RRA to the moments from Inverse Dynamics. You should see a 30-50% reduction in peak residual moments.
4. Compare the joint torques/forces to established literature (if available). Try to find data with multiple subjects. Your results should be within one standard deviation of the literature

The table below shows an example of threshold values used to evaluate RRA results for full body simulations of walking and running:

Thresholds:	GOOD	OKAY	BAD
MAX Residual Force (N)	0-10 N	10-25N	> 25 N
RMS Residual Force (N)	0-5 N	5-10 N	> 10 N
MAX Residual Moment (Nm)	0-50 Nm	50-75 Nm	>75 Nm
RMS Residual Moment (Nm)	0-30 Nm	30-50 Nm	>50 Nm
MAX pErr (trans, cm)	0-2 cm	2-5 cm	>5 cm
RMS pErr (trans, cm)	0-2 cm	2-4 cm	>4 cm
MAX pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
RMS pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg

Next: How RRA Works

Home: Residual Reduction Algorithm

How RRA Works

The topics covered in this section include:

Overview

Residual reduction is a form of forward dynamics simulation that utilizes a tracking controller to follow model kinematics determined from the inverse kinematics. Computed muscle control (CMC) serves as the controller, but without muscles the skeleton of the model can be used to determine a mass distribution and joint kinematics that are more consistent with ground reaction forces.

Residual reduction is primarily intended for gait, i.e., movements like walking and running where the model is displaced relative to the ground while subject to ground reaction forces and torques. In this chapter, we describe an example gait model (gait2354_simbody.osim) consisting of ten rigid segments (bones) where 17 of the 23 generalized coordinates (degrees of freedom) of the model represent angles for the joints connecting the rigid segments together. Each of these 17 degrees of freedom is actuated by a single torque actuator.

The remaining 6 generalized coordinates represent the 6 degrees of freedom (3 translational, 3 rotational) between the model's pelvis and the ground. To simulate walking, we need some way of representing how the model propels itself forward relative to the ground. One way would be to use a foot-ground contact mechanism.

Instead, we present a simpler solution: represent the 6 degrees of freedom between the pelvis and the ground as a 6-degree-of-freedom joint between the pelvis and the ground, and actuate each degree of fredom with its own torque actuator. Each of these 6 actuators is called a *residual actuator*. Now our model has 23 degrees of freedom and 23 actuators, i.e., exactly one actuator per degree of freedom. The three residuals that actuate the 3 translational degrees of freedom between the pelvis and the ground are the *residual forces*, whose values we denote by F_x, F_y, and F_z. The 3 rotational degrees of freedom are actuated by the *residual torques* (or *moments*), whose values we denote by M_x, M_y, and M_z. F_x is the force applied along the X (forward) axis, F_y is the force applied along the Y (vertical) axis, M_x is the torque applied about the X (forward axis), and so on.

Typically, modeling assumptions (e.g., having a model with no arms), noise, and other errors from motion capture data lead to *dynamic inconsistency*; essentially, the ground reaction forces and acceleration estimated from measured marker kinematics for a subject do not satisfy Newton's Second Law, $F = ma$. Roughly speaking, the 6 residuals amount to adding a new force to the equation that accounts for inconsistency:

$$F + F_{residual} = ma$$

Residual Reduction Algorithm (RRA)

Tracking Simulation

RRA begins by placing the model in the starting configuration, i.e., by setting the values of the model's generalized coordinates to the values computed by the inverse kinematics (IK) tool for the user-specified initial time (specified in the setup file as the <initial_time> property). Repeatedly, RRA takes small steps forward in time (with each time step of .001) until the user-specified final time (specified under the <final_time> property of the setup file) is reached. In each step, force values are computed for all of the model's actuators to make the model move from its current configuration to the configuration (generalized coordinates) desired at the end of the step, which is computed from the IK output. The actuator forces are computed by choosing force and torque values that minimize an objective function (see Settings Files and XML Tag Definitions on Optimization Parameters).

Mass Center Adjustment

At the end of the simulation, the average value for each residual actuator is computed. The average values for M_x (the left-right residual torque) and M_z (the fore-aft residual torque) are used to adjust the torso mass center to correct excessive "leaning" of the model due to inaccuracies in the mass distribution and geometry of the torso in the model. A new model file containing the adjusted torso mass center (specified in the setup file under the <output_model_file> property) is created.

Mass Adjustment Recommendation

The average value of F_y is used to compute the recommended mass changes for all of the body segments. The desired mass change is:

F_y / g

where g = -9.80665 m/s^2. This mass change is then divided up proportionally among the body segments. The computed mass changes are recommended to the user, who can make these changes in the OpenSim model file by hand. The recommended mass changes are NOT applied to the model automatically.

Adjusted Kinematics

The same tracking simulation process is then repeated with three important differences:

- The model with the adjusted torso mass center is used
- The residuals are weighted more heavily to make the optimizer choose smaller values for the residuals when minimizing the objective function
- minimum and maximum limits are placed on the residual values

The goal of these restrictions on the residual values is to reduce the need for residuals to the absolute minimum that is necessary to closely follow the desired kinematics so that the motion is generated purely by internal joint moments. During computed muscle control (CMC), the next stage of OpenSim, the moments will be generated by forces exerted by muscles. This way, biomechanical results about muscle function concluded from CMC will be closer to reality than if we let the residuals be arbitrarily large.

With these restrictions placed on the residuals, the model's motion will likely be altered since the residuals may not be allowed to reach the magnitudes that would result from inverse dynamics while following the kinematics from IK exactly. If the minimum and/or maximum allowed residual values are too restrictive, the motion will be altered so dramatically that the results of RRA cannot be used to generate a realistic simulation with CMC. If the residual minimum and/or maximum values are too lenient, then the residuals will still be large enough to exert forces that might normally be exerted by muscles, and thus the results would lead to unrealistic muscle function from CMC.

Next: How to Use the RRA Tool

Previous: Getting Started with RRA

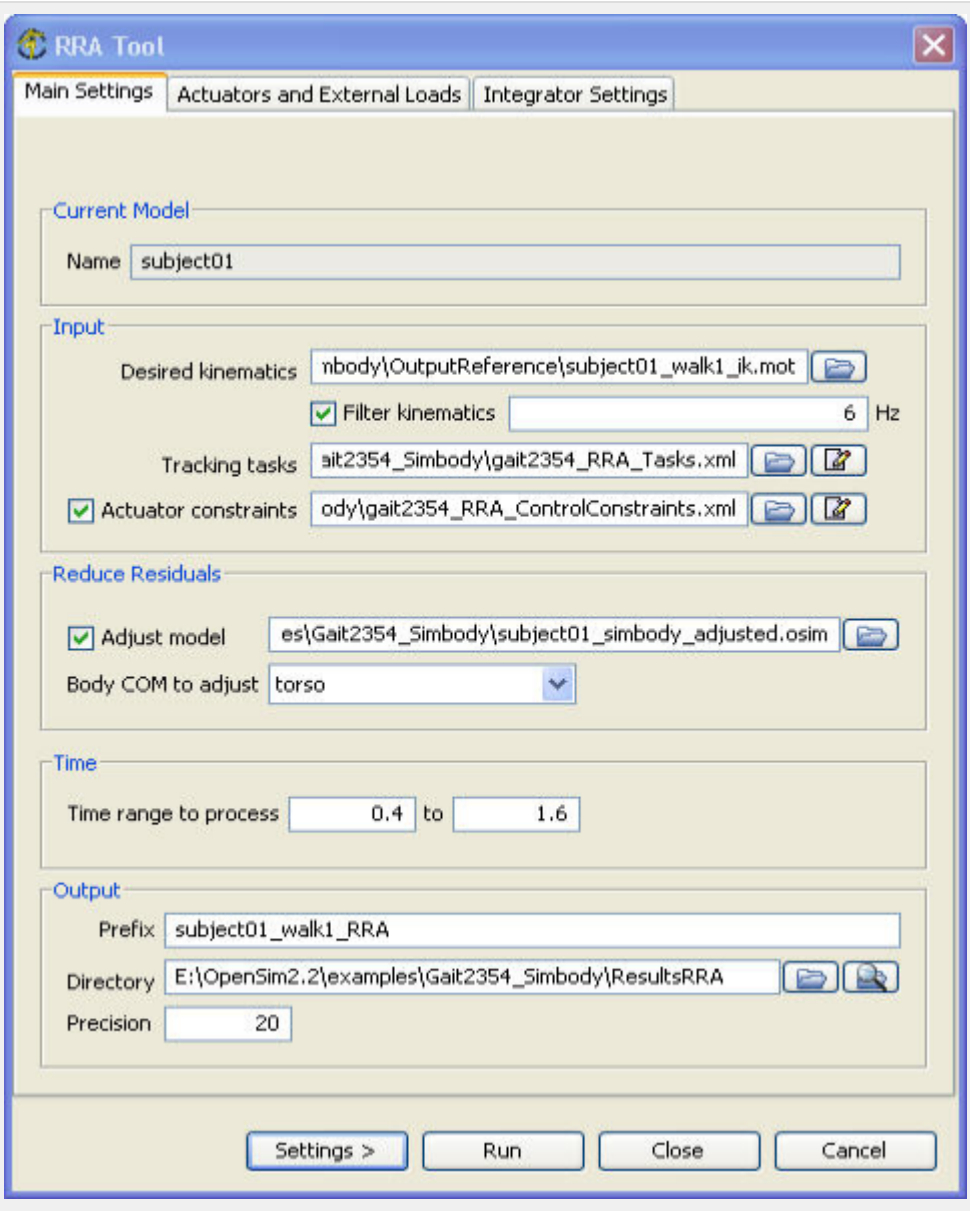
Home: Residual Reduction Algorithm

How to Use the RRA Tool

The topics covered in this section include:

How to Use the GUI

The computed muscle control tool is accessed by selecting **Tools Reduce Residuals ...** from the OpenSim menu. The RRA Tool is governed by three tabs:



- The *Main Settings* pane is used to specify the input kinematics, tracking tasks, and control constraints as well as the time range for the analysis, and the output of the results.
- The *External Loads* pane is used to specify parameters relating to the external loads applied to the model during the analysis.
- The *Integrator Settings* pane is described more in Settings Files and XML Tag Definitions.

Figure: Window for the RRA Tool. The residual reduction tool window has three panes. The *Main Settings* pane is shown here; there are other tabs for describing additional actuators (residual and reserve) to be used and integrator settings for forward dynamics simulation (as described for in How to Use the Forward Dynamics Tool).

Configuring the RRATool

To configure the RRATool, check **Adjust model** and specify a body to adjust. Specifically, RRA will compute the average residual forces that were necessary for tracking and use these values to change the center of mass location of the body specified and recommend mass adjustments.

Secondly, kinematics are tracked (according to the task weightings) and experimental accelerations are not imposed as constraints.

Replacing Model Actuators

Critical to RRA is the replacement of muscles with just one ideal actuator per coordinate. In the gait2354_simbody example, these correspond to residuals for the pelvis' six degrees of freedom and reserves for all other model internal coordinates (joint angles). Under the **Actuators and External Loads** tab, you would load the gait2354_RRA_Actuators.xml file and select **Replace model's force set**.

Command-line Execution

RRA uses the command **rra -S <setup file name>**, for example,

```
rra -S subject01_Setup_RRA.xml
```

Next: Settings Files and XML Tag Definitions

Previous: How RRA Works

Home: Residual Reduction Algorithm

Settings Files and XML Tag Definitions

The settings files are XML files whose tags specify properties to be used by OpenSim for performing the residual reduction. The tags used for each type of settings file are defined in the following sections:

RRA Setup File

A setup file provides the high-level information required for the first pass of the residual reduction algorithm. It references three other files: an actuators file, the constraints file, and a tasks file. All of these files are explained in detail below. An example of the setup file is given in Example 1 below.

In the setup file, the property settings for RRA are enclosed in <RRATool>. The types of properties listed in the XML setup files for RRA include model files, actuator and control information, integration parameters, kinematics and ground reaction data files, tracking information, optimization parameters, and output information.

Model Files

The <model_file> property specifies the name of the .osim file to load. In RRA, the value of this property in the above example is **subject01_simbody.osim**, the .osim file representing the dynamic subject-specific model. The output of RRA <output_model_file> in Example 1 is **subject01_simbody_adjusted.osim**.

Integration Parameters

The <maximum_number_of_integrator_steps> property indicates the maximum number of steps RRA can take before a particular integration terminates. During each integration, the maximum number of seconds that may elapse is specified by <maximum_integrator_step_size>. Increasing the <integrator_error_tolerance> will decrease the integrator step size, while decreasing the <integrator_fine_tolerance> will increase the integrator step size. Other simulation parameters like <initial_time>, <final_time>, and <cmc_time_window> are described earlier in the Simulation section for RRA (How RRA Works).

Tracking Information

The coordinates that should be followed by the model during RRA are specified within a file, indicated by the <task_set_file> and </task_set_file> tags. In Example 1, that file is **gait2354_RRA_Tasks.xml**. See RRA Tasks File for information about the property tags used within a task file.

Actuator and Control Information

A model has some set of actuators that can apply forces to its skeleton. For example, the dynamic subject-specific model in Example 1 has 54 muscles as its default set of actuators. These actuators can be modified using the tag <replace_force_set>. If the value of the property <replace_force_set> is **true**, then the actuators specified in any file listed under the property <force_set_files> will replace the corresponding model's actuators. If the value of <replace_force_set> is **false**, then the actuators specified in the files listed under <force_set_files> will be added to the model's existing actuator set. Details about the actuators file are given in RRA Actuators File below.

The <constraints_file> property specifies the name of an XML file containing minimum and maximum values for the control values for the model's actuators. An actuator's actual range of values is equal to the range from the minimum to maximum control values times its optimal force. The maximum and minimum control values, as well as the optimal force, are specified in the constraints file. See Example 3 below for more information about the constraints file and the best method for altering an actuator's range.

Kinematics and Ground Reaction Data Files

RRA will attempt to make the model follow the kinematics (generalized coordinates as functions of time) specified in the <desired_kinematics_file>, which should be a motion (.mot) or a storage (.sto) file.
Prior to simulation, the kinematics to be tracked by RRA are low-pass filtered at a frequency specified by the tag <lowpass_cutoff_frequency>. The value of this frequency is assumed to be in Hertz (Hz). A negative value for this property leads to no filtering. The default value is **–1.0**, i.e., no filtering.

There can be multiple sets of ground reaction forces for different locations on the body. These are specified in the file identified by the <external_loads_file> tags. The external loads .xml file specifies the source file for input forces and to which location and body the forces are to be applied. See Inverse Dynamics for additional details for applying any number of external loads as point forces and/or body torques.

Optimization Parameters

The target consists of an objective function (*J*) that is a weighted (*w*) sum of squared actuator controls, *x*, plus the sum of desired acceleration ($\ddot{q}_j^*$) errors:

$$J = \sum_{i=1}^{nx} x_i^2 + \sum_{j=1}^{nq} w_j (\ddot{q}_j^* - \ddot{q}_j)$$

The first summation minimizes and distributes loads across actuators and the second drives the model accelerations ($\ddot{q}_j^*$) toward the desired accelerations ($\ddot{q}_j$).

The behavior of the optimizer is controlled using three tags: <optimizer_derivative_dx>, <optimizer_convergence_criterion>, <optimizer_max_iterations>.

The <optimizer_derivative_dx> tag determines the perturbation size used by the optimizer to compute numerical derivatives. Valid values for <optimizer_derivative_dx> range from **1.0e-4** to **1.0e-8**.

The <optimizer_convergence_criterion> tag specifies the depth of optimization required for convergence. The smaller the value of this property, the better the solution is. But decreasing this value also can increase computation time.

The <optimizer_max_iterations> property limits the number of iterations the optimizer can take in searching for an optimal solution.

The optimizer can be set to print out details of what it is doing by using the <optimizer_print_level> tag. Valid values for this property are **0**, **1**, **2**, or **3**, where a value of **0** means no printing, a value of **3** means detailed printing, and **1** and **2** represent levels in-between.

Output

Results printed by RRA will be output into the directory specified by the tag <results_directory>. The name of the file that is created is determined by the <output_model_file> tag. By default, the name of the <output_model_file> is **adjusted_model.osim**. The precision of RRA output is specified in the property <output_precision>, which is **8** by default.

If the <adjust_com_to_reduce_residuals> property is **true**, the file that is output contains data where the mass center of the body is specified by the tag <adjusted_com_body>. The <adjusted_com_body> should normally be the heaviest segment in the model. For the gait model, **torso** is usually the best choice. The body name must correspond to the body name in the navigator (or model file). The average residuals computed during RRA will be printed out to a file in the <results_directory>.

RRA Actuators File

The RRA actuators file uses the <ForceSet> and </ForceSet>tags to describe actuators that replace the model's actuators. Example 2 below shows an actuator file for RRA (**gait2354_RRA_Actuators.xml**).

RRA Tasks File

Each generalized coordinate task contains the following properties: <on>, <wrt_body>, <express_body>, <active>, <weight>, <kp>, <kv>, <ka>, <r0>, <r1>, <r2>, <coordinate>, and <limit>.
<on> indicates whether or not the coordinate(s) specified in the task should be followed or not. The body to which the task is applied is specified using the tags <wrt_body> and </wrt_body>. The reference frame which is used to specify the coordinates to follow is determined by the property <express_body>, which refers to a specific body. For example, if a point on body 2 is to be followed, and the point's coordinates are expressed in the frame of body 1, then <wrt_body> would be **2** and <express_body> would be **1**.

The <active> property is an array of three flags, each flag indicating whether a component of a task is active. For example, the trajectory of a point in space could have three components (x, y, z). This allows the tracking of each coordinate of the point to be made active (**true**) or inactive (**false**). The definition of a flag depends on the type of coordinate: <CMC_Joint> or <CMC_Point>. For a task that tracks a joint coordinate (like all tasks in Example 34 below), only the first of the three flags is valid.

The tags <kp>, <kv>, < ka> are parameters in the proportional-derivative (PD) control law used to compute desired accelerations for tracking the experimental kinematics computed by the inverse kinematics (IK) solver (see Inverse Kinematics). This control law contains a position error feedback gain (stiffness) and a velocity error feedback gain (damping). The stiffness for each tracked coordinate is specified within the <kp> property, while the damping is specified within the <kv> property. An acceleration feed-forward gain is also allowed, but in the above example, this gain is set to **1**, i.e., there is no acceleration gain. The acceleration gain is specified within the <ka> property.

The <r0>, <r1>, and <r2> properties indicate direction vectors for the three components of a task, respectively. These properties are not used for tasks representing tracking of a single joint coordinate, such as the tasks in Example 4 below.

The name of the coordinate to be tracked is specified within the property <coordinate>. The error limit on the tracking accuracy for a coordinate is specified within the <limit> property. If the tracking errors approach this limit during simulation, the weighting for this coordinate is increased.

RRA Example Files

Example 1: XML file for the setup file for RRA

subject01_Setup_RRA.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSimDocument Version="20300">

<RRATool name="subject01_walk1_RRA">
  <!--Name of the .osim file used to construct a model.-->
  <model_file> subject01_simbody.osim </model_file>

   <!--Replace the model's force set with sets specified in <force_set_files>? If false, the force
set is appended to.-->
  <replace_force_set> true </replace_force_set>

  <!--List of xml files used to construct an force set for the model.-->
  <force_set_files> gait2354_RRA_Actuators.xml </force_set_files>

  <!--Directory used for writing results.-->
  <results_directory> ResultsRRA/ </results_directory>

  <!--Output precision. It is 20 by default.-->
  <output_precision> 8 </output_precision>

  <!--Initial time for the simulation.-->
  <initial_time> 0.75000000 </initial_time>

  <!--Final time for the simulation.-->
  <final_time> 0.78000000 </final_time>

  <!--Flag indicating whether or not to compute equilibrium values for states other than the
coordinates or speeds.
      For example, equilibrium muscle fiber lengths or muscle forces.-->
  <solve_for_equilibrium_for_auxiliary_states> false </solve_for_equilibrium_for_auxiliary_states>

  <!--Maximum number of integrator steps.-->
  <maximum_number_of_integrator_steps> 20000 </maximum_number_of_integrator_steps>

  <!--Maximum integration step size.-->
  <maximum_integrator_step_size> 0.00100000 </maximum_integrator_step_size>

  <!--Minimum integration step size.-->
  <minimum_integrator_step_size> 0.00000001 </minimum_integrator_step_size>

  <!--Integrator error tolerance. When the error is greater, the integrator step size is
decreased.-->
  <integrator_error_tolerance> 0.00010000 </integrator_error_tolerance>

  <!--Set of analyses to be run during the investigation.-->
  <AnalysisSet name="Analyses">
    <objects/>
    <groups/>
  </AnalysisSet>

  <!--Controller objects in the model.-->
  <ControllerSet name="Controllers">
    <objects/>
    <groups/>
  </ControllerSet>

  <!--XML file (.xml) containing the external loads applied to the model as a set of
```

```
PrescribedForce(s).-->
  <external_loads_file> subject01_externalForces.xml </external_loads_file>

  <!--Motion (.mot) or storage (.sto) file containing the desired point trajectories.-->
  <desired_points_file> </desired_points_file>

  <!--Motion (.mot) or storage (.sto) file containing the desired kinematic trajectories.-->
  <desired_kinematics_file> subject01_walk1_ik.mot </desired_kinematics_file>

  <!--File containing the tracking tasks. Which coordinates are tracked and with what weights are
specified here.-->
  <task_set_file> gait2354_RRA_Tasks.xml </task_set_file>

  <!--File containing the constraints on the controls.-->
  <constraints_file> gait2354_RRA_ControlConstraints.xml </constraints_file>

  <!--File containing the controls output by RRA. These can be used to place constraints on the
residuals during CMC.-->
  <rra_controls_file> </rra_controls_file>

  <!--Low-pass cut-off frequency for filtering the desired kinematics. A negative value results in
no filtering.
      The default value is -1.0, so no filtering.-->
  <lowpass_cutoff_frequency> -1.00000000 </lowpass_cutoff_frequency>

  <!--Preferred optimizer algorithm (currently support "ipopt" or "cfsqp", the latter requiring the
osimFSQP library.-->
  <optimizer_algorithm> ipopt </optimizer_algorithm>

  <!--Perturbation size used by the optimizer to compute numerical derivatives. A value between
1.0e-4 and 1.0e-8
      is usually appropriate.-->
  <optimizer_derivative_dx> 0.00010000 </optimizer_derivative_dx>

  <!--Convergence criterion for the optimizer. The smaller this value, the deeper the convergence.
Decreasing this number
      can improve a solution, but will also likely increase computation time.-->
  <optimizer_convergence_criterion> 0.00000100 </optimizer_convergence_criterion>

  <!--Flag (true or false) indicating whether or not to make an adjustment in the center of mass of
a body to reduced DC offsets
      in MX and MZ. If true, a new model is writen out that has altered anthropometry.-->
  <adjust_com_to_reduce_residuals> true </adjust_com_to_reduce_residuals>

  <!--Initial time used when computing average residuals in order to adjust the body's center of
mass. If both initial and final
      time are set to -1 (their default value) then the main initial and final time settings will
be used.-->
  <initial_time_for_com_adjustment> -1.00000000 </initial_time_for_com_adjustment>

  <!--Final time used when computing average residuals in order to adjust the body's center of
mass.-->
  <final_time_for_com_adjustment> -1.00000000 </final_time_for_com_adjustment>

  <!--Name of the body whose center of mass is adjusted. The heaviest segment in the model should
normally be chosen. For a gait model, the
      torso segment is usually the best choice.-->
  <adjusted_com_body> torso </adjusted_com_body>

  <!--Name of the output model file (.osim) containing adjustments to anthropometry made to reduce
average residuals. This file is written
      if the property adjust_com_to_reduce_residuals is set to true. If a name is not specified,
the model is written out to a
      file called adjusted_model.osim.-->
  <output_model_file> subject01_RRA_adjusted.osim </output_model_file>

  <!--True-false flag indicating whether or not to turn on verbose printing for cmc.-->
  <use_verbose_printing> false </use_verbose_printing>
```

```
 </RRATool>
</OpenSimDocument>
```

Example 2: XML file for the actuator set file for RRA

gait2354_RRA_Actuators.xml

```
<?xml version="1.0" encoding="UTF-8"?>
 <ForceSet name="gait2354_RRA">

 <defaults>

   <PointActuator name="default">
     <max_force> 10000.000 </max_force>
     <min_force> -10000.000 </min_force>
     <optimal_force> 1000.00000000 </optimal_force>
     <body> </body_A>
     <point> 0.000 0.000 0.000 </point>
     <direction> 1.000 0.000 0.000 </direction>
   </PointActuator >

   <TorqueActuator name="default">
     <max_force> 1000.000 </max_force>
     <min_force> -1000.000 </min_force>
     <optimal_force> 300.00000000 </optimal_force>
     <body_A> </body_A>
     <axis> 1.000 0.000 0.000 </axis>
     <body_B> </body_B>
   </TorqueActuator>

   <CoordinateActuator name="default">
     <max_force> 1000.000 </max_force>
     <min_force> -1000.000 </min_force>
     <optimal_force> 300.00000000 </optimal_force>
     <coordinate> </coordinate>
   </CoordinateActuatorCoordinateActuator>

 </defaults>

 <objects>

   <!-- Residuals -->

   <PointActuator name="FX">
     <optimal_force> 4.00000000 </optimal_force>
     <body> pelvis </body>
     <point> -0.0724376 0.00000000 0.00000000 </point>
     <point_is_global> false </point_is_global>
     <direction> 1 0 0 </direction>
     <force_is_global> true </force_is_global>
   </PointActuator>

   <PointActuator name="FY">
     <optimal_force> 8.00000000 </optimal_force>
     <body> pelvis </body>
     <point> -0.0724376 0.00000000 0.00000000 </point>
     <point_is_global> false </point_is_global>
     <direction> 0 1 0 </direction>
     <force_is_global> true </force_is_global>
   </PointActuator>

   <PointActuator name="FZ">
     <optimal_force> 4.00000000 </optimal_force>
```

```
  <body> pelvis </body>
  <point> -0.0724376 0.00000000 0.00000000 </point>
  <point_is_global> false </point_is_global>
  <direction> 0 0 1 </direction>
  <force_is_global> true </force_is_global>
</PointActuator>

<TorqueActuator name="MX">
  <optimal_force> 2.0 </optimal_force>
  <body_A> pelvis </body_A>
  <axis> 1.000 0.000 0.000 </axis>
  <body_B> ground </body_B>
</TorqueActuator>

<TorqueActuator name="MY">
  <optimal_force> 2.0 </optimal_force>
  <body_A> pelvis </body_A>
  <axis> 0.000 1.000 0.000 </axis>
  <body_B> ground </body_B>
</TorqueActuator>

<TorqueActuator name="MZ">
  <optimal_force> 2.0 </optimal_force>
  <body_A> pelvis </body_A>
  <axis> 0.000 0.000 1.000 </axis>
  <body_B> ground </body_B>
</TorqueActuator>

<!-- Right Leg -->

<CoordinateActuator name="hip_flexion_r">
  <coordinate> hip_flexion_r </coordinate>
  <optimal_force> 300.0 </optimal_force>
  </CoordinateActuator>

<!-..additional <CoordinateActuator> tags cut for brevity..->
```

```
    </objects>

    </ForceSet>
```

Example 3: XML file for the control constraints file for RRA

gait2354_RRA_ControlConstraints.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<ControlSet name="gait2354_RRA">

  <defaults>
    <ControlLinear name="default">
      <is_model_control> true </is_model_control>
      <extrapolate> true </extrapolate>
      <default_min> -1.0 </default_min>
      <default_max> 1.0 </default_max>
      <use_steps> false </use_steps>
    </ControlLinear>
  </defaults>

  <objects>
    <ControlLinear name="FX.excitation">
      <default_min> -20.0 </default_min>
      <default_max> 20.0 </default_max>
    </ControlLinear>

    <ControlLinear name="FY.excitation">
      <default_min> -20.0 </default_min>
      <default_max> 20.0 </default_max>
    </ControlLinear>

    <ControlLinear name="FZ.excitation">
      <default_min> -20.0 </default_min>
      <default_max> 20.0 </default_max>
    </ControlLinear>

    <ControlLinear name="MX.excitation">
      <default_min> -50.0 </default_min>
      <default_max> 50.0 </default_max>
    </ControlLinear>

    <ControlLinear name="MY.excitation">
      <default_min> -50.0 </default_min>
      <default_max> 50.0 </default_max>
    </ControlLinear>

    <ControlLinear name="MZ.excitation">
      <default_min> -50.0 </default_min>
      <default_max> 50.0 </default_max>
    </ControlLinear>

    <!-- Right Leg -->
    <ControlLinear name="hip_flexion_r.excitation" />
    <ControlLinear name="hip_adduction_r.excitation" />
    <ControlLinear name="hip_rotation_r.excitation" />
    <ControlLinear name="knee_angle_r.excitation" />
    <ControlLinear name="ankle_angle_r.excitation" />

    <!-- Left Leg -->
    <ControlLinear name="hip_flexion_l.excitation" />
    <ControlLinear name="hip_adduction_l.excitation" />
    <ControlLinear name="hip_rotation_l.excitation" />
    <ControlLinear name="knee_angle_l.excitation" />
    <ControlLinear name="ankle_angle_l.excitation" />

    <!-- Back -->
    <ControlLinear name="lumbar_extension.excitation" />
    <ControlLinear name="lumbar_bending.excitation" />
    <ControlLinear name="lumbar_rotation.excitation" />

  </objects>

</ControlSet>
```

Example 4: XML file for the tasks file for RRA

gait2354_RRA_Tasks.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<CMC_TaskSet name="gait2354_RRA">

  <defaults>
    <CMC_Joint name="default">
      <on> false </on>
      <wrt_body> -1 </wrt_body>
      <express_body> -1 </express_body>
      <active> false false false </active>
      <weight> 1 1 1 </weight>
      <kp> 1 1 1 </kp>
      <kv> 1 1 1 </kv>
      <ka> 1 1 1 </ka>
      <coordinate> </coordinate>
      <limit> 0 </limit>
    </CMC_Joint>
  </defaults>

  <objects>

    <!-- Pelvis -->

    <CMC_Joint name="pelvis_tz">
      <wrt_body> -1 </wrt_body>
      <express_body> -1 </express_body>
      <on> true </on>
      <active> true false false </active>
      <weight> 5.0e0 </weight>
      <kp> 100.0 </kp>
      <kv> 20.0 </kv>
      <coordinate> pelvis_tz </coordinate>
    </CMC_Joint>

    <CMC_Joint name="pelvis_tx">
      <wrt_body> -1 </wrt_body>
      <express_body> -1 </express_body>
      <on> true </on>
      <active> true false false </active>
      <weight> 5.0e0 </weight>
      <kp> 100.0 </kp>
      <kv> 20.0 </kv>
      <coordinate> pelvis_tx </coordinate>
    </CMC_Joint>

    <CMC_Joint name="pelvis_ty">
      <wrt_body> -1 </wrt_body>
      <express_body> -1 </express_body>
      <on> true </on>
      <active> true false false </active>
      <weight> 5.0e0 </weight>
      <kp> 100.0 </kp>
      <kv> 20.0 </kv>
      <coordinate> pelvis_ty </coordinate>
    </CMC_Joint>

  <!-- . . additional <CMC_Joint> tags cut for brevity . . -->

  </objects>

</CMC_TaskSet>
```

Next: Computed Muscle Control

Previous: How to Use the RRA Tool

Home: Residual Reduction Algorithm

Computed Muscle Control

In this section we will cover:

- Getting Started with CMC
- How CMC Works
- How to Use the CMC Tool'
- CMC Settings Files and XML Tag Definitions

Next: Getting Started with CMC

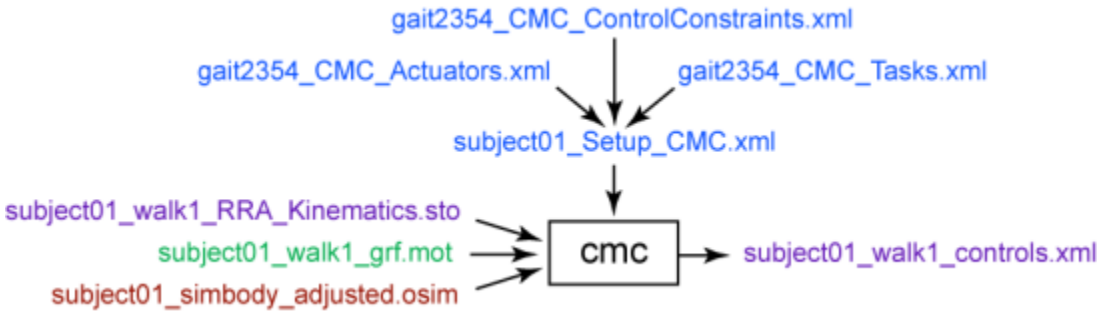
Getting Started with CMC

The purpose of Computed Muscle Control (CMC) is to compute a set of muscle excitations (or more generally actuator controls) that will drive a dynamic musculoskeletal model to track a set of desired kinematics in the presence of applied external forces (if applicable).

The Computed Muscle Control tool is accessed by selecting **Tools Computed Muscle Control…** from the OpenSim main menu bar. Like all tools, the operations performed by the Computed Muscle Control tool apply to the current model.

Overview

The figure below shows the required inputs and outputs for the Computed Muscle Control Tool. Each is described in more detail in the following sections.



Inputs and Outputs of the Computed Muscle Control Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown inblue; files generated by the workflow are shown in purple.

> The file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.

Settings File

The **subject01_Setup_CMC.xml** file is a setup file for the CMC Tool, which specifies settings, inputs, and outputs that affect the behavior of the tracking controller to determine actuator (including muscles) controls. These can be defined using the GUI or by hand. Details of the settings are described in the section on the Graphical User Interface.

The setup file identifies the actuators (i.e., the residual actuators, such as required for dynamic consistency) as well as the kinematic tracking tasks. Furthermore, control constraints on the actuators (to limit the maximum residual force) can be specified.

Inputs

Several data files are required as input by the Computed Muscle Control Tool:

subject01_walk1_RRA_Kinematics_q.sto: Contains the time histories of model kinematics including the joint angles and pelvis translations from RRA.

gait2345_CMC_Tasks.xml: The tracking tasks file specifying which coordinates to track and the corresponding tracking weight (weights are relative and determine how "well" a joint angle will track the specified joint angle from RRA).

gait2345_CMC_ControlConstraints.xml: Contains limits on model actuators, which include muscles, reserve and residual actuators. The control constraints file specifies the maximum and minimum "excitation" (i.e., control signal) for each actuator. Control constraints can also be used to enforce when certain actuators are "on" or "off" and the range in which they can operate.

subject01_walk1_grf.xml: External load data (i.e., ground reaction forces, moments, and center of pressure location). See Inverse Dynamics for more details.

subject01_simbody_adjusted .osim: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters. The model should have an adjusted torso center of mass to reduce residuals

gait2345_CMC_Actuators.xml: Contains the residual and reserve actuators, as in RRA.

Outputs

The Computed Muscle Control Tool tool primarily reports the necessary controls:

subject01_walk1_controls.xml: Contains the excitations to individual muscles as well as controls for any residual and/or reserve actuators.

subject01_CMC_forces.sto: Muscle forces and reserve/residual forces and torques.

For completeness, CMC outputs the state trajectories (these are joint coordinate values and their speeds, as well as muscle states such activation and fiber length) :

subject01_walk1_states.sto: Model states and muscle states of the simulated motion (i.e., joint angles AND velocities, muscle fiber lengths AND activations).

Best Practices and Troubleshooting Tips

CMC Settings

1. The reserve actuators are torques that are added about each joint to augment the actuator's force in order to enable the simulation to run (reserves turn on when an actuator cannot produce the needed at a given time point). To help minimize reserve torques, make an initial pass with default inputs, and then check reserves, residuals, and joint angle errors. To reduce reserves further, decrease tracking weights on coordinates with low error.
2. Optimal forces for reserves should be low to prevent optimizer from "wanting" to use reserve actuators (an actuator with large optimal force and low excitation is "cheap" in the optimizer cost). If larger forces are needed for a successful simulation, increase the maximum control value of residuals. The residuals will then be able to generate sufficient force, but will be penalized for doing so.
3. If you are still getting high reserves for a particular degree-of-freedom during a particular time range in the simulation, it may be useful to examine more closely the muscles which span the degree-of-freedom during that time region (see TipsForDebuggingMuscleActuatedSimulations.pdf). In particular:
    a. Check passive muscle force (i.e., quadriceps). Large passive forces (from large knee flexion angles) may induce active forces in the antagonistic muscles (i.e. hamstrings, gastrocnemius) which may not be desired. Passive forces cannot be controlled in CMC - they are purely a function of the whole body kinematics of the motion. Although tempting, do not increase the maximum isometric force excessively unless you know its consequences in the muscle model. Because the passive muscle force is modeled as a function of maximum isometric force, if you increase the maximum isometric force in the hope of making your active muscles stronger, you will also be increasing the passive forces in the muscles as well, thereby not helping the situation. To decrease passive muscle forces, you want to reduce the passive muscle stiffness property of muscle (or more specifically, increase the FmaxMuscleStrain parameter in the Thelen2003Muscle).
    b. Check normalized fiber length during the motion (plotter tool). Is the muscle acting at sub-optimal fiber lengths (i.e. less than 0.8 or greater than 1.2) at the time where the reserves are being generated? If so, then you may consider modifying either the tendon slack length or the optimal fiber length of the muscle so that it is operating more optimally (and thus capable of generating a greater force) during this time in the simulation. Although many cadaver studies report the optimal fiber length of a muscle, the tendon slack length is almost never reported, even though it is especially sensitive to the operating region on the force-length curve. Small adjustments to the tendon slack lengths can therefore be justified in reducing the reserves on the basis that we don't have as much confidence in this value to begin with.
4. You should almost always use the "fast" target for CMC. This requires the joint accelerations at each time step are matched to the RRA results. Fast target should work for normal subjects, slow target may be needed for subjects with pathologies
5. Start CMC at least 0.03 seconds before point where you want to start analyzing your data, as CMC requires a 0.03 sec time for initialization.
6. See How CMC Works and How to Use the CMC Tool for more information.

CMC Troubleshooting

1. If CMC is failing, try increasing the max excitation for reserves and residuals by 10x until the simulation runs. Then try working your way back down while also "relaxing" tracking weights on coordinates.

Evaluating your Results

1. Peak reserve actuators torques should typically be less than 10% of the peak joint torque.
2. Peak residual forces should typically be less than 10-20 N and peak residual moments less than 75Nm, depending on the type of motion.
3. Double check your kinematics, in comparison to RRA. Generally, they should match well as long as you are using the "fast" target.
4. If performing an Induced Acceleration Analysis, you should verify that reserves and residuals contribute less than 5% to the net acceleration of interest.
5. Compare the simulated activations to experimental EMG data (either recorded from your subject or from the literature). Activations should exhibit similar timing and magnitude to EMG data. You can also compare your muscle activations and/or forces to other simulations from SimTK or the literature.

The table below shows an example of threshold values used to evaluate CMC results for full body simulations of walking and running:

Thresholds:	GOOD	OKAY	BAD
MAX Residual Force (N)	0-10 N	10-25N	> 25 N
RMS Residual Force (N)	0-10 N	10-25 N	> 25 N
MAX Residual Moment (Nm)	0-50 Nm	50-75 Nm	>75 Nm
RMS Residual Moment (Nm)	0-30 Nm	30-50 Nm	>50 Nm
MAX pErr (trans, cm)	0-1 cm	1-2 cm	>2 cm
RMS pErr (trans, cm)	0-1 cm	1-2 cm	>2 cm
MAX pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
RMS pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
MAX Reserve (Nm)	0-25 Nm	25-50 Nm	> 50 Nm
RMS Reserve (Nm)	0-10 Nm	10-25 Nm	> 25 Nm

Next: How CMC Works

Home: Computed Muscle Control

How CMC Works

At user-specified time intervals during a simulation, the CMC tool computes muscle excitation levels that will drive the generalized coordinates (e.g., joint angles) of a dynamic musculoskeletal model towards a desired kinematic trajectory. CMC does this by using a combination of proportional-derivative (PD) control and static optimization (figure below).

Before starting the CMC algorithm, initial states for the model are computed. The states comprise the generalized coordinates (joint angles), generalized speeds (joint angular velocities), plus any muscle states (e.g., muscle activation levels and fiber lengths). While the initial values of the generalized coordinates and speeds can be taken from the desired kinematics that you specify, the initial values of the muscle states are generally unknown. To compute viable starting muscle states, CMC is applied to the first 0.030 seconds of the desired movement. Because the muscle states are generally out of equilibrium and muscle forces can change dramatically during this initial time interval, the simulation results during this interval are generally not valid. Therefore, you should make sure to start CMC at least 0.030 seconds prior to the interval of interest.

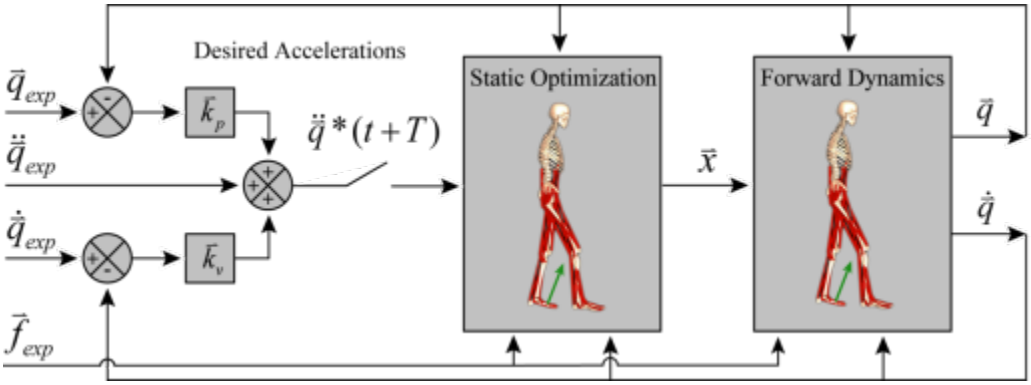


Figure: Schematic of the Computed Muscle Control Algorithm Applied to Gait [Thelen, D.G. and Anderson, F.C., "Using computed muscle control to generate forward dynamic simulations of human walking from experimental data, J. Biomech., 2006, 39(6):1107-1115]

The first step in the CMC algorithm is to compute a set of desired accelerations, $\ddot{\vec{q}}^*$, which when achieved will drive the model coordinates, $\vec{q}$, toward the experimentally-derived coordinates, $\vec{q}_{exp}$. The desired accelerations are computed using the following PD control law:'

$$\ddot{\vec{q}}^*(t+T) = \ddot{\vec{q}}_{exp}(t+T) + \vec{k}_v \cdot [\dot{\vec{q}}_{exp}(t) - \dot{\vec{q}}(t)] + \vec{k}_p \cdot [\vec{q}_{exp}(t) - \vec{q}(t)]$$

where $\vec{k}_v$ and $\vec{k}_p$ are the feedback gains on the velocity and position errors, respectively. Because the forces that muscles apply to the body cannot change instantaneously, the desired accelerations are computed for some small time *T* in the future. For musculoskeletal models, *T* is typically chosen to be about 0.010 seconds. This time interval is short enough to allow adequate control, but long enough to allow muscle forces to change.

If these desired accelerations are achieved, errors between the model coordinates and experimentally-derived coordinates will be driven to zero. To drive these errors to zero in a critically damped fashion (i.e., without over-shooting or over-damping), the velocity gains can be chosen using the following relation:

$$\vec{k}_v = 2 \cdot \sqrt{\vec{k}_p}$$

For musculoskeletal models, it works well if the error gains are chosen to drive any errors to zero slowly. The error gains $\vec{k}_v = 20$ and $\vec{k}_p = 100$ will cut down tracking errors.

The next step in CMC is to compute the actuator controls, $\vec{x}$, that will achieve the desired accelerations, $\ddot{\vec{q}}^*(t+T)$. Most of the time the controls are predominantly comprised of muscle excitations, but this is not required. Any kind of actuator can be used with CMC (e.g., idealized joint moments). Static optimization is used to distribute the load across synergistic actuators. It is called "static" optimization because the performance criterion (i.e., the cost index) is confined to quantities that can be computed at any instant in time during a simulation. Using criteria like jump height or total metabolic energy over a gait cycle, for example, are not possible because these require simulating until the body leaves the ground or until the end of the gait cycle is reached.

Two formulations of the static optimization problem are currently available in CMC. The first formulation, called the slow target, consists of a

performance criterion (*J*) that is a weighted sum of squared actuator controls plus the sum of desired acceleration errors:

$$J = \sum_{i=1}^{nx} x_i^2 + \sum_{j=1}^{nq} w_j (\ddot{q}_j * - \ddot{q}_j)^2$$

The first summation minimizes and distributes loads across actuators and the second drives the model accelerations ($\ddot{q}_j$) toward the desired accelerations ($\ddot{q}_j *$).

The second formulation, called the fast target, is the sum of squared controls augmented by a set of equality constraints ($C_j = 0$) that requires the desired accelerations to be achieved within the tolerance set for the optimizer:

$$J = \sum_{i=1}^{nx} x_i^2 \quad ; \; C_j = \ddot{q}_j * - \ddot{q}_j, \; for \; all \; j$$

The fast target is both faster and generally produces better tracking. However, if the constraints cannot be met, the fast target will fail and CMC will exit with an error message. Often the reason for the failure is that the musculoskeletal model is not strong enough.

To prevent the fast target from failing, it is possible to add a number of *reserve actuators* to a model that are able to make up for strength deficiencies in muscles should any be encountered. The reserve actuators have very low strength (or optimal force) and so require very high excitations to apply very much load to the model. As a result, use of the reserve actuators is highly penalized in both the fast and slow formulations. When the forces (or moments) applied by the reserve actuators are written to file and plotted, their values are a good indication of which joints in the musculoskeletal model are not strong enough. If the model is strong enough, the reserve actuator forces/moments should generally be very small relative to the forces/moments applied by the primary actuators.

The final step in the CMC algorithm is to use the computed controls to conduct a standard forward dynamic simulation, advancing forward in time by *T* . These steps—computing the desired accelerations, static optimization, and forward dynamic simulation—are repeated until time is advanced to the end of the desired movement interval.

Once CMC finishes execution, you will typically want to compare the computed muscle excitation patterns with prototypical or measured electromyographical measurements. If desired, constraints can be placed on the upper and lower bounds of the controls $\bar{x}$ as a function of simulation time. The bounds on the controls $\bar{x}$ are specified in an XML input file. For muscle excitations, the default upper bound is typically 1.0 (full excitation), and the default lower bound is typically a small number just above 0.0 (no excitation), such as 0.01 or 0.02. The lower bound is not set at precisely 0.0 because mathematical models of muscle are often not as well behaved when excitation goes all the way to 0.0. The format of the control constraints is shown in the following section.

You can view the excitations produced by CMC using the Excitation Editor.

Next: How to Use the CMC Tool

Previous: Getting Started with CMC

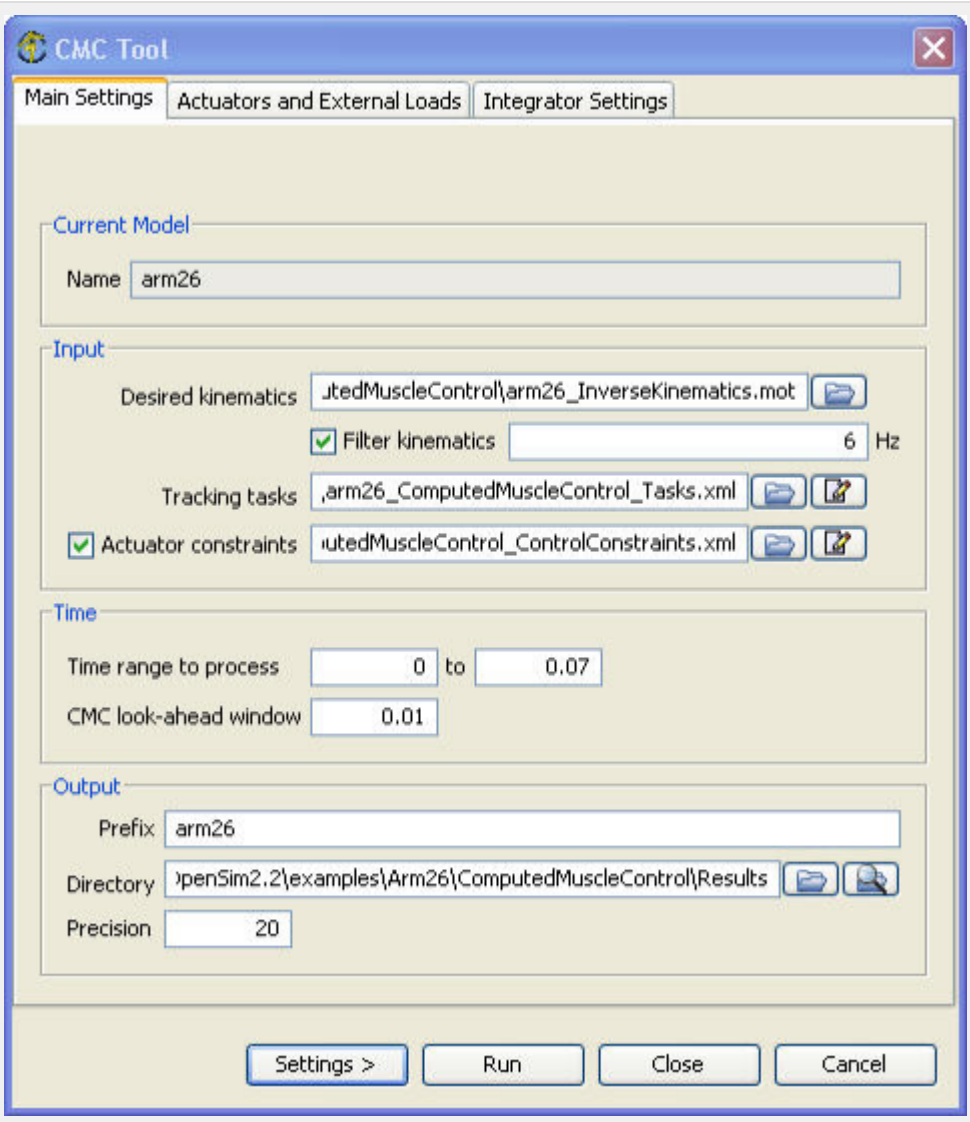
Home: Computed Muscle Control

How to Use the CMC Tool

How to Use the GUI

The computed muscle control tool is accessed by selecting **Tools Computed Muscle Control…** from the OpenSim main menu bar (figure below). Like all tools, the operations performed by the computed muscle control tool apply to the current model. The name of the current model is shown in bold in the Navigator. See chapters Opening, Closing, and Using the Navigator Window for information on opening models and making a particular model current.

The Computed Muscle Control (CMC) Tool is controlled by a window with three tabbed panes (figure below). This window is the same as that used for RRA (see Residual Reduction Algorithm).



- The *Main Settings* pane is used to specify parameters related to the input kinematics of the current model, the time range for the analysis, and the output of the results.
- The *Actuators and External Loads* pane is used to specify additional actuators, such as residual and reserve actuators.
- The external loads and integrator settings are specified in the same manner as in Forward Dynamics.

Command-line Execution

CMC is run using the command **cmcgait -S <setup file>** where <setup file> is the name of the setup file, for example,

```
cmcgait -S subject01_Setup_CMC.xml
```

Next: CMC Settings Files and XML Tag Definitions

Previous: How CMC Works

Home: Computed Muscle Control

CMC Settings Files and XML Tag Definitions

This section of the chapter covers how to control CMC execution. The properties governing execution are contained in XML files. The topics covered in this section include:

CMC Setup File

Execution of the CMC tool is controlled by properties specified in the CMC setup file. Some properties, such as <initial_time> and <final_time>, are specified directly in the CMC setup file. Other properties, such as <task_set_file>, refer to additional files that contain additional settings that affect CMC execution. These are discussed in the sections that follow.

The properties for CMC are enclosed inside the opening and closing tags <CMCTool> and </CMCTool>. The name attribute name ="subject01_walk1" specifies the execution name. The names of results files generated will be prefixed with this name.

Example 1: XML file for the CMC setup file

```
<?xml version="1.0" encoding="UTF-8"?>

<OpenSimDocument Version="20302">

  <CMCTool name="subject01_walk1">
    <!--Name of the .osim file used to construct a model.-->
    <model_file> subject01_simbody_adjusted.osim </model_file>

    <!--Replace the model's force set with sets specified in
        <force_set_files>? If false, the force set is appended to.-->
    <replace_force_set> false </replace_force_set>

    <!--List of xml files used to construct an force set for the model.-->
    <force_set_files> gait2354_CMC_Actuators.xml </force_set_files>

    <!--Directory used for writing results.-->
    <results_directory> ./subject01_ResultsCMC </results_directory>

    <!--Output precision. It is 8 by default.-->
    <output_precision> 20 </output_precision>

    <!--Initial time for the simulation.-->
    <initial_time> 0.76000000 </initial_time>

    <!--Final time for the simulation.-->
    <final_time> 1.24000000 </final_time>

    <!--Flag indicating whether or not to compute equilibrium values for
```

```
        states other than the coordinates or speeds. For example, equilibrium
        muscle fiber lengths or muscle forces.-->
    <solve_for_equilibrium_for_auxiliary_states> true </solve_for_equilibrium_for_auxiliary_states>

    <!--Maximum number of integrator steps.-->
    <maximum_number_of_integrator_steps> 30000 </maximum_number_of_integrator_steps>

    <!--Maximum integration step size.-->
    <maximum_integrator_step_size> 1.00000000 </maximum_integrator_step_size>

    <!--Minimum integration step size.-->
    <minimum_integrator_step_size> 0.00000000 </minimum_integrator_step_size>

    <!--Integrator error tolerance. When the error is greater, the integrator
        step size is decreased.-->
    <integrator_error_tolerance> 0.00000100 </integrator_error_tolerance>

    <!--Set of analyses to be run during the investigation.-->
    <AnalysisSet name="Analyses">
      <objects/>
      <groups/>
    </AnalysisSet>

    <!--Controller objects in the model.-->
    <ControllerSet name="Controllers">
      <objects/>
      <groups/>
    </ControllerSet>

    <!--XML file (.xml) containing the forces applied to the model as
        ExternalLoads.-->
    <external_loads_file> subject01_externalForces.xml </external_loads_file>

    <!--Motion (.mot) or storage (.sto) file containing the desired point
        trajectories.-->
    <desired_points_file> </desired_points_file>

    <!--Motion (.mot) or storage (.sto) file containing the desired kinematic
        trajectories.-->
    <desired_kinematics_file> subject01_walk1_RRA_Kinematics_initial_q.sto
</desired_kinematics_file>

    <!--File containing the tracking tasks. Which coordinates are tracked and
        with what weights are specified here.-->
    <task_set_file> gait2354_CMC_Tasks.xml </task_set_file>

    <!--File containing the constraints on the controls.-->
    <constraints_file> gait2354_CMC_ControlConstraints.xml </constraints_file>

    <!--File containing the controls output by RRA. These can be used to place
        constraints on the residuals during CMC.-->
    <rra_controls_file> </rra_controls_file>

    <!--Low-pass cut-off frequency for filtering the desired kinematics. A
        negative value results in no filtering. The default value is -1.0, so
        no filtering.-->
    <lowpass_cutoff_frequency> -10.00000000 </lowpass_cutoff_frequency>

    <!--Time window over which the desired actuator forces are achieved.
        Muscles forces cannot change instantaneously, so a finite time window
        must be allowed. The recommended time window for RRA is about 0.001
        sec, and for CMC is about 0.010 sec.-->
    <cmc_time_window> 0.01000000 </cmc_time_window>

    <!--Flag (true or false) indicating whether or not to use the curvature
        filter. Setting this flag to true can reduce oscillations in the
        computed muscle excitations.-->
    <use_curvature_filter> false </use_curvature_filter>

    <!--Flag (true or false) indicating whether to use the fast CMC
```

```
    optimization target. The fast target requires the desired
    accelerations to be met. The optimizer fails if the acclerations
    constraints cannot be met, so the fast target can be less robust. The
    regular target does not require the acceleration constraints to be
    met; it meets them as well as it can, but it is slower and less
    accurate.-->
<use_fast_optimization_target> true </use_fast_optimization_target>

<!--Preferred optimizer algorithm (currently support "ipopt" or "cfsqp",
    the latter requiring the osimFSQP library.-->
<optimizer_algorithm> ipopt </optimizer_algorithm>

<!--Perturbation size used by the optimizer to compute numerical
    derivatives. A value between 1.0e-4 and 1.0e-8 is usually
    approprieate.-->
<optimizer_derivative_dx> 1.00000000 </optimizer_derivative_dx>

<!--Convergence criterion for the optimizer. The smaller this value, the
    deeper the convergence. Decreasing this number can improve a solution,
    but will also likely increase computation time.-->
<optimizer_convergence_criterion> 0.00010000 </optimizer_convergence_criterion>

<!--Maximum number of iterations for the optimizer.-->
<optimizer_max_iterations> 200 </optimizer_max_iterations>

<!--Print level for the optimizer, 0 - 3. 0=no printing, 3=detailed
    printing, 2=in between-->
<optimizer_print_level> 0 </optimizer_print_level>

<!--True-false flag indicating whether or not to turn on verbose printing
    for cmc.-->
<use_verbose_printing> false </use_verbose_printing>
```

```
</CMCTool>

</OpenSimDocument>
```

Specifying the Model

The <model_file> specifies the name of the .osim file used to construct and initialize the model. The <force_set_files> are a list of actuators set file names, each of which may be included as actuators of the model. These actuators are in addition to any actuators specified in the model file that are already part of the model. <replace_force_set>, if set to true, indicates that the actuator sets listed in <force_set_files> should replace any actuators that are already part of the model otherwise actuators are appended.

Specifying Initial and Final Times

The properties <initial_time> and <final_time> specify the time interval over which CMC is to be run. Be aware that CMC uses 0.030 seconds starting from the specified initial time to initialize any muscle states. In addition, to avoid undesirable effects having to do with the processing of the desired kinematics and external loads, it is prudent not to start CMC at the very beginning of the desired kinematics but to allow a buffer. One or two percent of a gait cycle, for example, is usually sufficient.

Specifying Integrator Settings

The <maximum_number_of_integrator_steps> property indicates the maximum number of steps CMC can take before a particular integration terminates. Making the <integrator_accuracy> finer (smaller) will decrease the integrator step size.

Specifying Analyses and Results

Any number of available analyses can be added to a CMC run. To get a listing of available analyses, run the command **cmcgait -PropertyInfo**.

There are several properties associated with the analyses results. <results_directory> specifies the directory where results should be written. <output_precision> specifies how many decimal places should be used when writing results. A value of 20 is sufficient to avoid round-off error when reading results back in during other steps in the workflow. <step_interval> specifies how often to record results during numerical integration. A value of 10 means record results every 10 integration steps.

Specifying Desired Kinematics

The file containing the desired kinematics is specified using the property <desired_kinematics_file>. A low-pass cutoff frequency for filtering the kinematics can be specified using the property <lowpass_cutoff_frequency>. An order 50 Finite Impulse Response Filter (FIR) is currently used. If the desired kinematics have already been filtered, as is normally the case by the time the CMC tool is used, specify a negative cutoff frequency to prevent any filtering at all.

Specifying External Loads

There can be multiple sets of ground reaction forces for different locations on the body. These are specified in the file identified by the <external_loads_file> tags. The external loads .xml file specifies the source file for input forces and to which location and body the forces are to be applied. See Inverse Dynamics for additional details for applying any number of external loads as point forces and/or body torques.

Note: Across all steps in the OpenSim workflow, it is important to use the same settings for specifying the external loads.

Controlling the Optimizer

A number of properties are available for controlling the optimizer. <use_fast_ optimization_target> specifies whether the fast or slow optimization target should be used (How RRA Works). <optimizer_derivative_dx> specifies the perturbation size in the controls for computing numerical derivatives. <optimizer_convergence_criterion> specifies a convergence criterion; the smaller this value, the deeper the convergence. <optimizer_max_iterations> specifies an iteration limit. <optimizer_print_level> allows diagnostic information to be printed. The larger the number (up to a maximum of 3), the more information that is printed.

Settings for the CMC algorithm

A number of properties are available for controlling the CMC algorithm proper. <cmc_time_window> specifies the time allowed for time-dependent actuators like muscles to change force. If the window is too small, actuator forces will not have enough freedom to generate the desired accelerations. If the window is too large, the control will be too coarse to allow for good tracking (How RRA Works). <use_curvature_filter> specifies whether or not to apply a curvature filter to computed excitations. If control values are oscillating, the curvature filter can used to attenuate these oscillations. To apply the curvature filter, set <use_curvature_filter> to **true**. <adjust_com_to_reduce_residuals> is a property used for residual reduction (Chapter 3). During CMC, this property should be set to **false**.

Specifying Tracking Tasks and Control Constraints

Which coordinates should be tracked and how those coordinates should be tracked are specified in a separate XML file specified by the property <task_set_file>. Constraints on the controls (e.g., muscle excitations) are also specified in a separate XML file using the property <constraints_file>. These files are covered below.

Task Set File

A task set is used to specify which coordinates (e.g., joint angles) in a model should follow a desired kinematic trajectory. It also specifies the relative weighting between coordinates.

Task Sets

Tasks, the class objects used to specify a tracking goal, are kept in sets. Sets are lists of objects. The individual tasks are found between opening and closing xml tags <objects> and </objects>.

Task Types

Although there will be additional task types in the future, such as orientation and point tracking, currently only tracking of generalized coordinates (or joints) is supported. The properties for each joint task are enclosed between opening and closing XML tags <rdCMC_Joint> and </rdCMC_Joint>.

Example 2: XML file for the CMC task set file

gait2354_CMC_tasks.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<CMC_TaskSet name="gait2354_CMC">

  <objects>

    <CMC_Joint name="pelvis_tz">
      <on> true </on>
      <active> true false false </active>
      <weight> 1.0e0 </weight>
      <kp> 100.0 </kp>
      <kv> 20.0 </kv>
      <coordinate> pelvis_tz </coordinate>
    </CMC_Joint>

    <CMC_Joint name="hip_flexion_r">
      <on> true </on>
      <active> true false false </active>
      <weight> 1.0e2 </weight>
      <kp> 100.0 </kp>
      <kv> 20.0 </kv>
      <coordinate> hip_flexion_r </coordinate>
    </CMC_Joint>

  </objects>

</CMC_TaskSet>
```

Task Properties

The property <on> is used to turn a task on or off. The <active> property specifies which component of task is active. While other task types might have three components, joint tasks only have one, so the typical setting is **true false false**. Properties <kp> and <kv> specify the position and velocity error gains, respectively. See How RRA Works for an explanation of error gains. The <coordinate> property specifies to which model coordinate a task applies. The value of the property must agree with the name of the model coordinate exactly. Joints can be translational (e.g., pelvis_tz) or rotational (e.g., hip_flexion_r).

Control Constraints File

Control Sets

The properties governing the controls in a simulation are contained in a <ControlSet>. Controls are functions that vary as a function of time. Currently, two types of controls are supported: <ControlConstant> and <ControlLinear>. <ControlConstant> objects are constant for any value of time. <ControlLinear> objects consist of an array of paired time and control values or nodes (i.e., [t1,x1], [t2,x2], … [tn,xn]). These values are linearly interpolated to determine the value of the control at a particular time. <ControlLinear> is the class most commonly used during CMC and is reviewed further below.

Default Objects

Many control objects within a file may have the same values for a number of properties. At the beginning of a control set, there is a defaults section, demarcated by the opening and closing tags <defaults> and </defaults>, that can be used to specify default property values. If a control has the same value as the default control, that property need not be repeated.

Properties of <ControlLinear> Control Type

A number of properties are used to describe the <ControlLinear> control type.

- <is_model_control> indicates whether or not a control belongs to the model. Muscle excitations and actuator controls in general are examples of such controls. For specifying control constraints for CMC, this property should always be **true**.
- <extrapolate> specifies that the value of the control should be extrapolated when the time value is outside the valid time range of a control (e.g., before t1 or after tn).
- <default_min> and <default_max> specify the default minimum and default maximum values, respectively, of a control. For muscles, these are frequently **0.02** and **1.00** as in Example 3, below
- <use_steps> specifies whether or not the value of a control should be linearly interpolated between nodes. If set to **true**, a control is treated as a step function in which the value of a step is determined by the value of the nearest node to the right or, equivalently, later in time. For CMC, it generally works better for muscle excitations to use steps.

Example 3: XML file for the CMC control constraints file

gait2354_CMC_ControlConstraints.xml

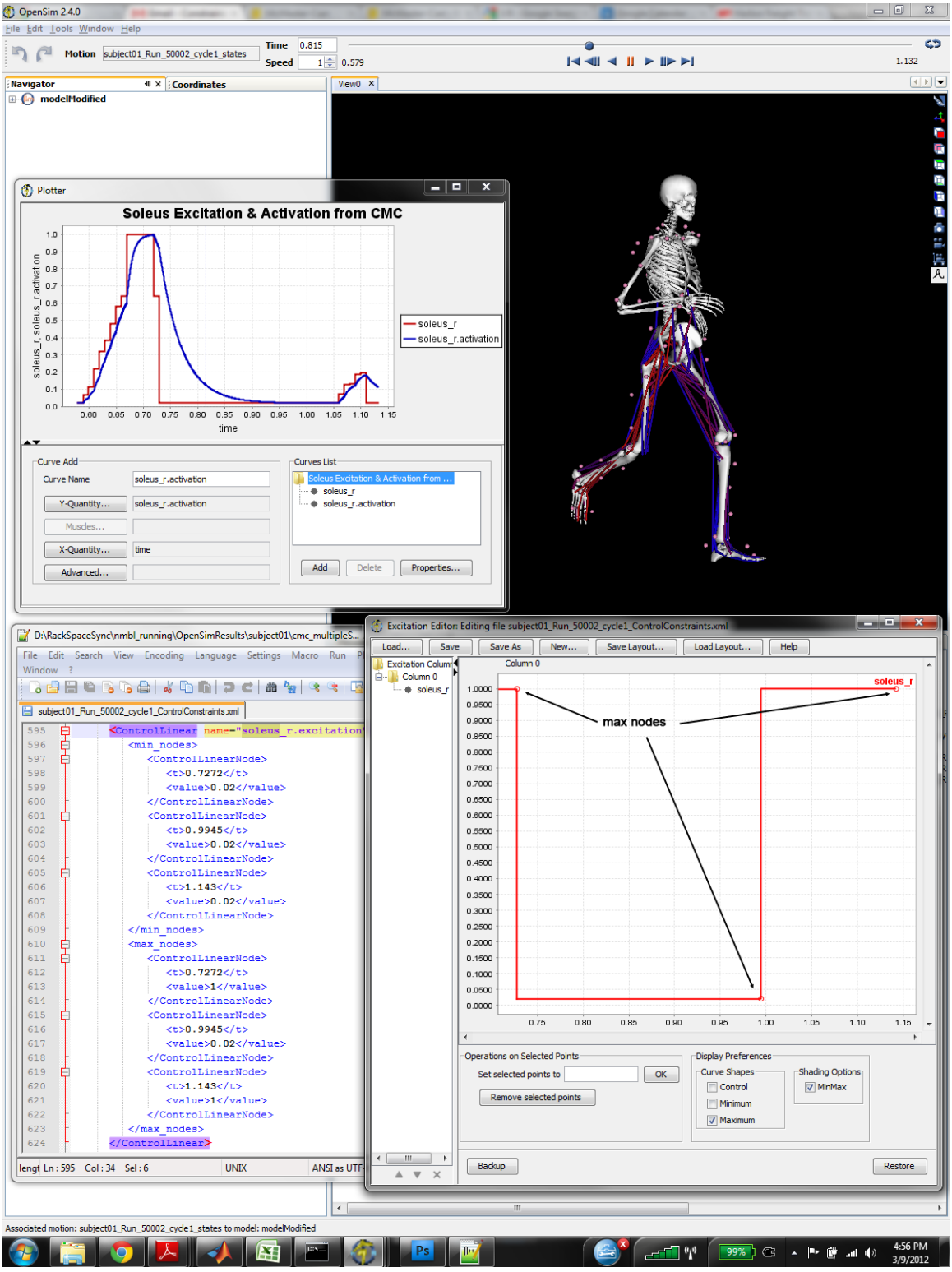
```
<?xml version="1.0" encoding="UTF-8"?>

  <ControlSet name="gait2354_CMC">

    <defaults>

      <ControlLinear name="default">
        <is_model_control> true </is_model_control>
        <extrapolate> true </extrapolate>
        <default_min> 0.02 </default_min>
        <default_max> 1.00 </default_max>
        <use_steps> true </use_steps>
      </ControlLinear>

    </defaults>

    <objects>

      <!--Residual actuators -->
      <ControlLinear name="FX.excitation">
        <default_min> -1.0 </default_min>
        <default_max> 1.0 </default_max>
        <use_steps> false </use_steps>
      </ControlLinear>

      <!-- Reserve actuators -->
      <ControlLinear name="hip_flexion_r_reserve.excitation">
        <default_min> -1000 </default_min>
        <default_max> 1000 </default_max>
      </ControlLinear>

      <!-- Muscles -->
      <ControlLinear name="glut_med1_r.excitation" />
      <ControlLinear name="glut_med2_r.excitation" />
      <ControlLinear name="glut_med3_r.excitation" />
      <ControlLinear name="bifemlh_r.excitation" />
      <ControlLinear name="bifemsh_r.excitation" />

  </objects>

</ControlSet>
```

Example 4: Using control constraints file to constrain CMC excitations to match EMG data

When using CMC, sometimes the optimizer will choose to activate a muscle during a part of the movement when it should not be activated. For example, based on EMG recordings during running, we know that soleus should not have much activity during swing. But sometimes CMC will choose to activate the muscle since it is not informed about the EMG data. To get a better match with experimental EMG, we can constrain the activation CMC chooses for soleus during a specific time period of the movement.

Below is an example of a constraint, where the right soleus is specified to be off between 0.7272 and 0.9945s.

The control constraints are displayed in the Excitation Editor GUI. Lines have been added to illustrate how the nodes are connected. The value associated with a node is applied **before** the specified time. In other words, the value is applied from the previous node until the specified node (Note: This is **not** consistent with how nodes are specified in controls file used by Forward Dynamics.)


OpenSim 2.4.0
File Edit Tools Window Help
Motion subject01_Run_50002_cycle1_states
Time 0.815
Speed 1 0.579
1.132
Navigator
Coordinates
modelModified
View0
Plotter
Soleus Excitation & Activation from CMC
soleus_r, soleus_r.activation
time
soleus_r
soleus_r.activation
Curve Add
Curve Name soleus_r.activation
Y-Quantity... soleus_r.activation
Muscles...
X-Quantity... time
Advanced...
Curves List
Soleus Excitation & Activation from ...
soleus_r
soleus_r.activation
Add
Delete
Properties...
D:\RackSpaceSync\nmbl_running\OpenSimResults\subject01\cmc_multipleS...
File Edit Search View Encoding Language Settings Macro Run
Window ?
subject01_Run_50002_cycle1_ControlConstraints.xml
595 <ControlLinear name="soleus_r.excitation"
596 <min_nodes>
597 <ControlLinearNode>
598 <t>0.7272</t>
599 <value>0.02</value>
600 </ControlLinearNode>
601 <ControlLinearNode>
602 <t>0.9945</t>
603 <value>0.02</value>
604 </ControlLinearNode>
605 <ControlLinearNode>
606 <t>1.143</t>
607 <value>0.02</value>
608 </ControlLinearNode>
609 </min_nodes>
610 <max_nodes>
611 <ControlLinearNode>
612 <t>0.7272</t>
613 <value>1</value>
614 </ControlLinearNode>
615 <ControlLinearNode>
616 <t>0.9945</t>
617 <value>0.02</value>
618 </ControlLinearNode>
619 <ControlLinearNode>
620 <t>1.143</t>
621 <value>1</value>
622 </ControlLinearNode>
623 </max_nodes>
624 </ControlLinear>
lengt Ln : 595 Col : 34 Sel : 6
UNIX
ANSI as UTF
Excitation Editor: Editing file subject01_Run_50002_cycle1_ControlConstraints.xml
Load... Save Save As New... Save Layout... Load Layout... Help
Excitation Columi
Column 0
soleus_r
Column 0
soleus_r
max nodes
Operations on Selected Points
Set selected points to
OK
Remove selected points
Display Preferences
Curve Shapes
Control
Minimum
Maximum
Shading Options
MinMax
Backup
Restore
Associated motion: subject01_Run_50002_cycle1_states to model: modelModified
99%
4:56 PM
3/9/2012

Control Constraints XML to Constraint Soleus

```
<ControlLinear name="soleus_r.excitation">
     <min_nodes>
          <ControlLinearNode>
               <t>0.7272</t>
               <value>0.02</value>
          </ControlLinearNode>
          <ControlLinearNode>
               <t>0.9945</t>
               <value>0.02</value>
          </ControlLinearNode>
          <ControlLinearNode>
               <t>1.143</t>
               <value>0.02</value>
          </ControlLinearNode>
     </min_nodes>
     <max_nodes>
          <ControlLinearNode>
               <t>0.7272</t>
               <value>1</value>
          </ControlLinearNode>
          <ControlLinearNode>
               <t>0.9945</t>
               <value>0.02</value>
          </ControlLinearNode>
          <ControlLinearNode>
               <t>1.143</t>
               <value>1</value>
          </ControlLinearNode>
     </max_nodes>
</ControlLinear>
```

Residual Actuators

Residual actuators are actuators that act directly between the model and the ground. These are used to control the global position and orientation of a model. If the residuals for a simulation have been reduced (Chapter 3), the loads applied by these actuators should be relatively small. However, unlike muscles, the residual actuators can apply negative as well as positive forces. The minimum and maximum control values here, therefore, differ from the default values set at the top of the example file (0.02 and 1.0) and must be specified explicitly (-1.0 and 1.0).

Reserve Actuators

Reserve actuators are actuators that can make up for insufficient muscle strength during a simulation. However, they should only apply significant loads (e.g., forces above 1 N or 1 Nm) when necessary. To penalize the use of reserve actuators, the minimum and maximum control values are allowed to be very large (-1000 and 1000 in Example 20-3).

Muscles

When a control has the same property values as that specified in the defaults section at the top of the control constraints file (Example 20-3), it is only necessary to have a one-line entry that specifies the name of the control.

Next: Analyses

Previous: How to Use the CMC Tool

Home: Computed Muscle Control

Analyses

The Analyze Tool enables you to analyze a model or simulation based on a number of inputs that can include time histories of model states, controls, and external loads applied to the model. A typical use case is to analyze an existing simulation, which may have been computed using computed muscle control, without having to rerun the simulation. This not only saves compute time, but more importantly allows a simulation to be analyzed exactly as it occurred, avoiding the numerical drift that frequently occurs when rerunning a forward integration, particularly if the duration of the simulation is long. This chapter describes how you can use the Analyze Tool:

Next: How Analysis Works

How Analysis Works

The topics covered in this section include:

How it Works

The Analyze Tool steps in time through a set of input data specifying the state of a model; at each time step, the tool runs a set of analyses on the model. Available analyses include:

- **Kinematics:** Records the generalized coordinates (q's), generalized speeds (u's), and the accelerations (i.e., derivatives of the generalized speeds: du/dt)
- **BodyKinematics:** Records the configuration (center of mass position and orientation) of each body, as well as their velocities (linear and angular) and accelerations (linear and angular). Additionally, it records the overall center of mass of the model, as well as the velocity and acceleration of this center of mass.
- **Actuation:** Records the generalized force, speed, and power developed by each actuator of the model. The generalized force can either be a force (with units N) or a torque (with units Nm). The actuator speed is the rate at which the actuator shortens. Depending on the actuator, a speed can be either a translational speed (m/s) or an angular speed (deg/s). An actuator power (Watts) is the rate at which an actuator does work. Positive work means that the actuator is delivering energy to the model; negative power means that the actuator is absorbing energy from the model.
- **JointReaction:** Reports the joint reaction loads from a model. For a given joint, the reaction load is calculated as the forces and moments required to constrain the body motions to satisfy the joint as if the joint did not exist. The reaction load acts at the joint center (mobilizer frame) of both the parent and child bodies and either force can be reported and expressed in either the child, parent or ground frames. The default behavior is to express the force on the child in the ground frame.

The analyses can be added to the list to run via the GUI or by editing a setup file. The input data typically are loaded from files, and may come from experiments or simulations. The results are collected and written to file, usually storage (.sto) files, which you can open with other programs, such as Matlab or Microsoft Excel, for further analysis or plotting. Any analysis available in OpenSim can be included in the list of analyses to run.

Depending on the analyses you would like to run, there are four types of input that may be needed to analyze a model. These are described below.

states States are the variables of a model that are governed by differential equations and thus are integrated during a simulation. The most common examples of states are the generalized coordinates (q; e.g., joint angles) and speeds (u; e.g., joint angular velocities) that specify the configuration of a model. Every model has them. The states are not limited to the coordinates and speeds, however. Muscles frequently have states. Muscle activation and fiber length are common examples of muscle states. [1]

controls Controls are independent variables used to control the behavior of a model. Muscle excitations are an example. They are not governed by differential equations, but typically are free to take on any value between zero (no excitation) and one (full excitation). The controls of a model are frequently the variables used as the control parameters in optimization problems.

external loads External loads are forces or torques applied between the ground (or world) and the bodies of a model. For example, you might have force plate data that were recorded during a gait experiment. The recorded forces and moments can be applied to the right and left feet of the model by inputting the data as external loads. In OpenSim 1.5, external loads can be applied to at most two bodies.

Inputs and Outputs

The Analyze Tool has the same inputs and outputs as the Forward Dynamics Tool. Refer to How to Use the Forward Dynamics Tool for more information about the input and output files.

References

[1] Schutte, L.M., Rodgers, M.M., Zajac, F.E., "Improving the efficacy of electrical stimulation-induced leg cycle ergometry: an analysis based on a dynamic musculoskeletal model," IEEE Transactions on Rehabilitation Engineering, 1993, 1: 109-125.

Next: How to Use the Analysis Tool

Home: Analyses

How to Use the Analysis Tool

Error: RuntimeException occurred while performing an XHTML storage transformation (null)

Induced Acceleration Analysis

The Induced Acceleration Analysis is used to compute accelerations caused or "induced" by individual forces acting on a model, for example, the contribution of individual muscle forces to the mass center acceleration. Typically, induced accelerations of generalized coordinates (e.g., knee angle) or body positions (e.g., model's center of mass) are desired, and the forces consist of muscles, gravity, and any additional forces (e.g., residual actuators, reserve actuators, etc.). The topics covered in this section are:

Next: Inputs and Outputs IAA

Inputs and Outputs IAA

Inputs and Outputs

The inputs for setting up the Analyze Tool consists of (see Analyses):

1. **OpenSim model** *[.osim file]* (same as input to CMC)
2. **External loads** *[.xml file]* applied to the model (same as input to CMC)
3. **Actuator controls** *[.xml file]* output from CMC, used to drive muscles and other actuators
4. **Model states** *[.sto file]* output from CMC, used to define actuator and kinematic state of the model for analysis

The Induced Acceleration Analysis, also requires several additional inputs:

1. **Coordinate names** of the coordinate induced accelerations one wishes to obtain. Key word 'all' reports all coordinate accelerations.
2. **Body names** of individual bodies whos accelearations one wishes to obtain. Key word 'all' reports all body accelerations. Keyword 'center_of_mass' reports on the induced acceleration of the mass center.
3. **ConstraintSet** is a list of Constraints used to replace the ExternalForces in the set of ExternaLoads. The user can specify the constraint type from PointConstraint, WeldConstraint and RollingOnSurfaceConstraint. The Induced Acceleration analysis will replace the ExternalForce with the corresponding Constraint when the*force_threshold* is exceeded.
4. **force_threshold** is the minimum external force magnitude required for constraints to be considered on. For example if the force threshold is set to 10N, if the ground reaction force is below 10N, no constraint is applied and if the ground reaction force is above 10N the specified constraint (Weld, Point, or RollingOnSurface) is applied.
5. **compute_potentials_only** if true provides just the potential (acceleration per unit force) of each contributor. Since a muscle's potential is not dependent on the activation level of the muscle a states file is not required to calculate the potentials. .
6. **report_constraint_reactions** if true reports the induced reaction forces for each constraint applied. This option provides the means of comparing the superposition of reaction forces to the experimentally measured reaction forces.

The outputs from an Induced Acceleration Analysis consists of:

1. The induced accelerations generated by each force contributor (muscles, actuators, gravity and velocity related forces) listed for each coordinate or body of interest listed (above).
2. Note, if compute_potentials_only is true, then the induced acceleration results correspond to 1N of applied force for any of the muscles and other actuators or 1Nm for residual and reserve torque actuators.
3. Optionally, if report_constraint_reactions is true, the contribution of each force contributor to the constraint reaction forces and moments are reported on the bodies connected by the constraint (typically the ground and the foot).

Next: How IAA Works

Home: Induced Acceleration Analysis

How IAA Works

The topics covered in this section include:

- Equations of Motion
- Constraints to model contact
- Superposition

Equations of Motion

The equations of motion (EOMs) of an OpenSim model are given by:

$$[M]\ddot{q} = G(q) + V(q,\dot{q}) + S(q,\dot{q}) + [R]f$$

where *M* is the mass matrix, *q*, are the generalized coordinates, *G*, is the generalized force due gravity, *V*, is the force due to Coriolis and centrifugal effects, *S*, is the resulting generalized force due to contact elements, and *f*, is a muscle force or any other applied force transformed to a generalized force via a force transmission matrix, *R*, which for a muscle contains its moment-arms. The induced acceleration of any force contributor is simply:

$$\ddot{q}_i = [M]^{-1}\{F_i\}$$

where, F_i, is the contribution of any force (gravitiation, Coriolois, contact or muscle) to the acceleration. For example, calculating the acceleration would require solving:

$$\ddot{q}_m = [M]^{-1}[R]f_m$$

An induced acceleration analysis with state dependent contact forces represented by, *S*, (including externally applied forces as function of time) poses unique challenges because these forces are not independent of the other forces acting on the model. In most cases, *S* represents the reaction forces due to the model's interaction with its environment. For example during walking, the ground reaction force is the force applied to the model by the ground, which includes the contributions of gravity, velocity and muscle forces to the reaction measured by a force-plate. Since internal forces, such as muscles, do not directly accelerate the center of mass – it is the reaction force with the environment that causes the mass center acceleration – then it is also necessary to determine how an internal force contributes to an external reaction force. Therefore, in computing the acceleration of the system due to individual force contributors, it is essential to decompose the external reaction force in order to include only the partial (induced) contribution of each internal force to the external reaction force and to the acceleration of the system. In this case the acceleration induced by a single force contributor is,

$$\ddot{q}_i = [M]^{-1}\{F_i + S_i\}$$

where, F_i, is the force contributor we are interested in analyzing, whether it be gravity or muscle forces or ideal torque actuators. In this case both the induced acceleration and the partial (induced) contact force, S_i, is unknown.

There are two fundamental approaches for solving this problem. One approach is to slightyly augment (or perturb) the force of interest, F_i, (by 1N) and integrate the equations of motion a small interval forward in time to evaluate a change in the reaction force. This method is commonly referred to as a "Perturbation Analysis." This yields a sensitivity of the contact force due a unit change in force, and multiplying this change by the applied force yields its induced reaction force. There are several issues with this approach that include having to use very stiff contact springs in order for the contact force sensitivity to remain linear over a small integration interval, which makes forward integration of the system dynamics slow. Second, results are sensitive to the selection of contact stiffness and time interval over which to evaluate the change in contact force. Applying stiff 3-dimensional linear and torsional springs approximates rigid constraints imposed by a weld constraint effectively eliminationg the 6 degrees of freedom at the point of contact.

The other approach, which is the current implementation of the Induced Acceleration Analysis, replaces the contribution of contact with an appropriate kinematic constraint. Kinematic constraint reaction forces are resolved simultaneously with the constrained equations of motion,

$$[M]\ddot{q} = G(q) + V(q,\dot{q}) + [R]f + [C]^T\lambda$$

$$[C]\ddot{q} = B(t,q,\dot{q})$$

where the constraint matrix, *C*, maps from components of constraint reaction forces, , to a system generalized force (replacing the applied external force or compliant contact model, *S*). The kinematic constraints on positions and velocities are differentiated so that they are expressed in terms of conditions on the coordinate accelerations. The left hand side contains the accelerations and the right hand side defines the conditions on accelerations described as a function, *B*, of time, system position and/or velocity.

Using this approach with kinematic constraints does not require forward integration to estimate the response of contact as in the perturbation

method, and results in a computationally efficient and precise analysis.

Constraints to model contact

OpenSim provides several constraint types that are supported with an Induced Acceleration Analysis. These include Point, Weld and RollingOnSurface constraints as summarized in the figure below. A Point constraint forces two points on separate bodies to remain coincident, but allows free relative rotation about that point, similar to a ball-and-socket joint. A Weld is similar but also constrains the orientation of the two bodies to remain fixed to one another. A RollingOnSurface describes constraint on a rolling body that is in contact with a plane defined on another body (Hamner *et al.*, 2010). Translations of the point of contact normal to the plane are constrained only if the point of contact penetrates into the surface of the plane, otherwise the point is free to lift-off. When the point is in contact with the plane its velocity relative to the plane is constrained to be zero in the plane. Finally, when in contact with the plane, the rolling body is constrained not to rotate about the normal of the surface (no-twisting) at the point of contact. These four individual constraints form define a kinematic behavior known as pure rolling. The point at which the constraint is applied is obtained from the point of application of a corresponding external force being applied during a forward simulation. In the case of the Weld, the orientationof the bodies are set according to current values of the generalized coordinates.

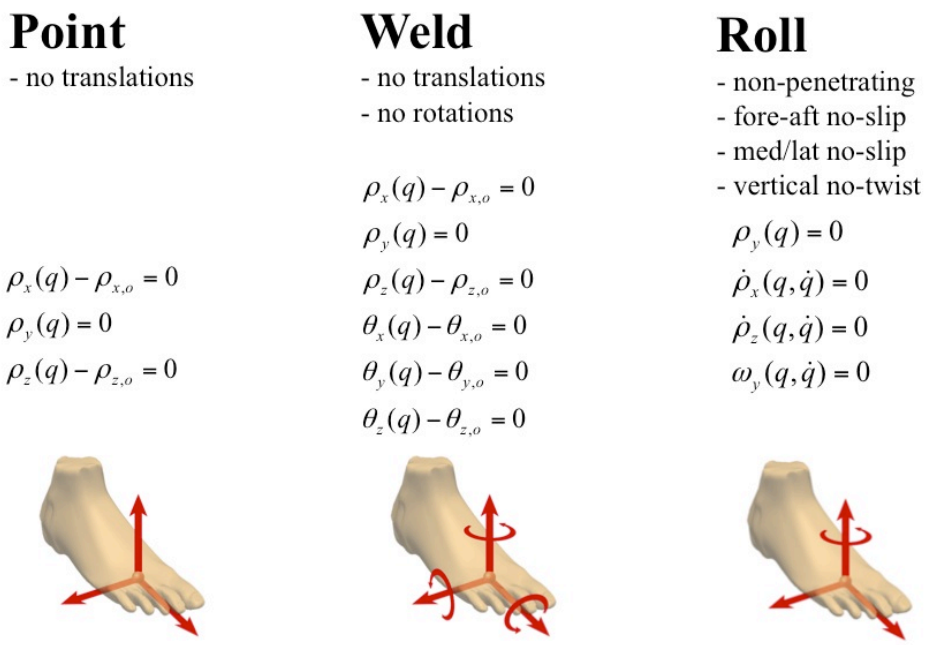


Figure: Available constraint types for an Induced Acceleration Analysis

Superposition

When analyzing the results of an induced acceleration analysis, it is important to test whether the sum of the accelerations induced by each muscle and by gravity is equal to the total acceleration. Since the induced acceleration analysis determines the acceleration of each actuator to the total acceleration, summing the induced acceleration of all the actuators should give the total acceleration. For example, you may want to compute the sum of the contributions of each muscle force and other forces to mass center acceleration and compare that to the net acceleration of the mass center (see Liu *et al*., 2006, and Hamner *et al*., 2010). The output file for each coordinate or body acceleration contains the "total" or summed induced accelerations from all of the actuators. This total acceleration can be compared to the original accelerations of joints or the mass center which can be obtained by running the "Kinematics" analysis or "Body Kinematics" analysis in OpenSim. OpenSim also can report constraint reaction forces, such that the induced reaction forces can be summed and compared to the total simulated and/or measured reaction forces to test the accuracy of the constraint in representing the contact conditions.

Next: How to Use IAA Tool

Previous: Inputs and Outputs IAA

Home: Induced Acceleration Analysis

How to Use IAA Tool

The topics covered in this section include:

Command-line Execution

The Induced Acceleration Analysis is run using the command **analyze -S <setup file name>,** for example,
analyze -S subject01_Setup_IAA.xml

Setup Files and XML Tag Definitions

Induced Acceleration Analysis Properties

The Induced Acceleration Analysis is performed using the AnalyzeTool (inputs defined in the section on How Analysis Works) and is added to its set of Analyses. The specific XML properties for an Induced Accelerations analyis are presented above, Setup Files and XML Tags.

Example: Sample properties for the setup of an Induced Acceleration Analysis

```
<InducedAccelerations name="InducedAccelerations">

<!--Flag (true or false) specifying whether whether on. -->
<on> true </on>

<!-- Starting time for the Analysis.-->
<start_time> 0.0 </start_time>

<!--End time for the Analysis.-->
<end_time> 1.5 </end_time>

<!--Specifies how often to store results during a simulation.-->
<step_interval> 10 </step_interval>

<!--Flag (true or false) indicating whether output angles are in degrees.-->
<in_degrees> true </in_degrees>

<!--Names of the coordinates for which to compute induced accelerations. The key word 'All'
indicates that the analysis should be performed for all coordinates.-->
<coordinate_names> knee_angle_r </coordinate_names>

 <!--Names of the bodies for which to compute induced accelerations. The key word 'All' indicates
that the analysis should be performed for all bodies. Use 'center_of_mass' for the induced
accelerations of the system mass center -->
<body_names> center_of_mass </body_names>

 <!-- ConstraintSet for replacing of ExternalForces applied to the model -->
<ConstraintSet>
  <objects>
  <RollingOnSurfaceConstraint name="right_foot_contact">

    <!-- Specify to disable constraint by default -->
    <isDisabled> true </isDisabled>

    <!-- Specify the rolling body for this constraint -->
    <rolling_body> calcn_r </rolling_body>

    <!-- Specify the body containing the surface (plane) that the rolling body rolls on. -->
    <surface_body> ground </surface_body>

    <!-- Surface's normal direction defined in the surface body -->
    <surface_normal> 0 1 0 </surface_normal>

    <!-- Surface height in the direction of the normal in the surface body: -->
```

```
    <surface_height> 0 </surface_height>

    <!-- Coulomb friction coefficient for rolling on the surface-->
    <friction_coefficient> 0.75 </friction_coefficient>

    <!-- A guess at the area of contact approximated by a circle-->
    <contact_radius> 0.01 </contact_radius>
  </RollingOnSurfaceConstraint>

  <RollingOnSurfaceConstraint name="left_foot_contact">
    <!-- Specify to disable constraint by default -->
    <isDisabled> true </isDisabled>

    <!-- Specify the rolling body for this constraint -->
     <rolling_body> calcn_l </rolling_body>

    <!-- Specify the body containing the surface (plane) that the rolling body rolls on. -->
    <surface_body> ground </surface_body>

    <!-- Surface's normal direction defined in the surface body -->
    <surface_normal> 0 1 0 </surface_normal>

    <!-- Surface height in the direction of the normal in the surface body -->
    <surface_height> 0 </surface_height>

    <!-- Coulomb friction coefficient for rolling on the surface: -->
    <friction_coefficient> 0.75 </friction_coefficient>

    <!-- A guess at the area of contact approximated by a circle -->
    <contact_radius> 0.01 </contact_radius>
  </RollingOnSurfaceConstraint>
  </objects>
</ConstraintSet>

 <!—Minimum magnitude of force necessary to turn on constraint.-->
<force_threshold> 6 </force_threshold>

<!— Only compute the potential (acceleration/N) of an actuator to accelerate the model..-->
<compute_potentials_only> false </compute_potentials_only>

<!— Report individual (induced) force contributions to constraint reactions forces in addition to
resulting accelerations->
```

```
<report_constraint_reactions> true </report_constraint_reactions>

</InducedAccelerations>
```

Next: Joint Reactions Analysis

Previous: How IAA Works

Home: Induced Acceleration Analysis

Joint Reactions Analysis

The topics covered in this section include:

- Input
- Output
- Best Practices and Troubleshooting Tips

JointReaction is an OpenSim Analysis for calculating the joint forces and moments transferred between consecutive bodies in a model. These forces and moments correspond to the internal loads carried by the joint structure. These loads represent the contributions of all un-modeled joint structures that would produce the desired joint kinematics, such as cartilage contact and any omitted ligaments. The reaction load acts at the joint center (mobilizer frame) of both the parent and child bodies. The loads can be reported and expressed in either the child, parent, or ground frames. The default behavior is to express the force on the child in the ground frame. For more detailed description of the method implementation see *Tibiofemoral Contact Force during Crouch Gait* (Steele et al, in review).

In this section, we provide a conceptual review of the inputs and outputs for a JointReaction Analysis along with a set of troubleshooting tips and best practices. Running an Analysis like JointReaction is covered in detail in the How to Use the Analysis Tool section.

Input

Inputs Specific to JointReaction:

1. joint_names: List of the names of the joints of interest. JointReaction reports loads for only listed joints that exist in the model. Joint names may be repeated any number of times to allow reporting on different bodies or with respect to different reference frames. Using the keyword 'all' reports the loads for all joints in the model. Default is 'all'.
2. apply_on_bodies: List of the body (parent or child) on which the corresponding reaction occurred. If the array has only one entry, that selection is applied to all joints specified in joint_names. The default is 'child'.
3. express_in_frame: List of the frames (ground, parent, or child) in which the corresponding reaction is expressed. If the array has only one entry, that selection is applied to all joints specified in joint_names.
4. forces_file: The name of a file containing forces storage. If a file name is provided, the applied forces for all actuators will be constructed from the forces_file instead of from the states. This option should be used to calculate joint loads from static optimization results.

Output

JointReaction prints results to one storage file with the suffix "_ReactionLoads.sto". This file contains rows of time-stamped data containing the 3 force and 3 moment vector components of the reaction load at each joint specified. Each data column label includes all information about how the load is applied and expressed. Specifically, the form is
< joint name >*on*<body>*in*<frame>_<component>.

For example, the column containing the z-component of hip force occurring on the femur and expressed in the femur frame would be labeled "hip_on_femur_in_femur_FX" while the y-component of the knee moment occurring on the tibia and expressed in the ground frame would have the label "knee_on_tibia_in_ground_MY".

Best Practices and Troubleshooting Tips

1. If JointReaction doesn't recognize a specified joint, body, or frame name, it will perform its default action. The overall default action is to report the loads on all joints as applied to the joint's child body and expressed in the ground reference frame.
2. *Consistency of modeling inputs to the Analyze tool:* The validity of the joint loads depends on modeling assumptions and correct modeling practices. JointReaction is very sensitive to the consistency of all inputs that define a dynamic trial, including the following inputs to the Analyze Tool: <model_file>, <replace_force_set>, <force_set_files>, <states_file>, <coordinates_file>, <speeds_file>, <lowpass_cutoff_frequency_for_coordinates>,<external_loads_file>, <external_loads_model_kinematics_file>, <lowpass_cuttoff_frequency_for_load_kinematics>
3. If any of these files are not associated with the same dynamic trial, the system of accelerations and forces will not be consistent. Therefore, JointReaction will calculate incorrect joint loads.
4. *Special Types of Forces:* Certain types of forces and actuators, called SpringGeneralizedForce, CoordinateLimitForce, and CoordinateActuator, are associated with specific degrees of freedom and treated as part of the joint structure. This means that any contribution from these components will be treated as part of the resultant loads reported by JointReaction instead of body forces. As an example, consider a reserve actuator on the hip joint of a gait model. If this reserve actuator is defined as a CoordinateActuator, its contribution is treated as a residual force provided by the joint and therefore will be added to the resultant load at the hip. However, if the hip reserve is defined as a TorqueActuator, its torque will be treated as a motor external to the joint and attached between the pelvis and femur. Therefore, its torque will not be added to the reported resultant load at the hip.

Next: Preparing Your Data

Preparing Your Data

This chapter describes the formats for data files that can be imported into OpenSim. Generally, you must input the following types of data into OpenSim to generate simulations:

1. Marker trajectories
2. Ground reaction forces and moments and centers of pressure

You may also import joint angles to provide additional kinematic data. Marker trajectories must be specified in .trc files, and ground reaction and center of pressure data must be specified in .sto or .mot files. Joint angles must be specified in .sto or .mot files. The .sto file format, which is similar to the .mot file format, is described below. EMG data may also be imported using .sto or .mot files, for example, to compare experimental EMG data to muscle excitations obtained from a simulation. This section covers:

Next: Coordinates and Utilities

Coordinates and Utilities

The topics covered in this section include:

- Laboratory Coordinates
- Best Practices
- OpenSim Utilities

Laboratory Coordinates

Every set of (x, y, z) coordinates obtained from a motion capture system is given relative to some coordinate system. Typically, this coordinate system is called the *laboratory coordinate system*, or simply *laboratory coordinates*. The laboratory coordinate system is generally an inertial frame fixed to the Earth. Before inputting any coordinates from motion capture into OpenSim, it is your responsibility to ensure that all (x, y, z) coordinates have been transformed from the laboratory coordinate system to the model coordinate system used in OpenSim. Although you can define an arbitrary model coordinate system, the standard convention used in OpenSim is as follows: Assume that the model is a full-body musculoskeletal model of the human body, standing in an upright position on the ground. The origin of the model coordinate system is halfway between its feet. The x-axis of the model coordinate system points forward from the model, the y-axis points upward, and the z-axis points to the right of the model.

If all positions and distances are converted to meters, then all (x, y, z) coordinates can be mapped from the laboratory coordinate system to the model coordinate system by an orthonormal transformation. An orthonormal transformation can be represented by a 3 X 3 rotation matrix whose rows (and columns) are a set of orthogonal vectors of length one,. This matrix represents the orientation of the laboratory coordinate frame in the model coordinate frame. So, to transform the coordinates of a point ${}^{lab}P = (x, y, z)$ given in the laboratory coordinate frame to its coordinates ${}^{model}P = (x', y', z')$ in the model coordinate frame, you would employ the following transformation, where ${}^{model}_{lab}R$ is the matrix whose columns are the vectors of the laboratory coordinate frame specified in the model coordinate frame:

$${}^{model}P = {}^{model}_{lab}R * {}^{lab}P$$

External forces and moments are usually given in the coordinate system of a particular force sensor, such as a force plate, which may be different than the laboratory coordinate system. In this case, the force and moment data must be transformed from the appropriate force sensor's coordinate system to the model coordinate system.

Best Practices

The first step for most researchers is collecting experimental data in the form of marker trajectories, ground reaction forces, and EMG. There are several key steps you can take to ensure that your motion capture data is as accurate as possible and represented correctly in OpenSim:

1. Develop and document lab protocols and standards for your marker sets, camera positions, and force plate/camera system coordinate frames.
2. Calibrate center-of-pressure measurements with marker positions using a calibration "T" to pinpoint where on the force-plate the point load (lowest tip of the "T") is being applied. If the center-of-pressure calculated from the force-plate does not match the "T" location from markers (within marker resolution), you need align the force-plate and marker mo-cap reference frames.
3. Digital cameras and camcorders are cheap! Take lots of photos/video during experiments so that you can verify marker placement and other factors for the data you collect.

OpenSim Utilities

Example Matlab scripts for converting certain specific types of motion capture data into a form recognized by OpenSim are provided in the OpenSim Utilities project on Simtk.org (https://simtk.org/home/opensim-utils). We provide these utilities as examples that have been applied successfully to data sets from the Center for Gait and Motion Analysis at Gillette Children's Specialty Healthcare in St. Paul, MN, USA, and the Human Performance Laboratory at Stanford University in Stanford, CA, USA. However, it is difficult to anticipate lab-specific formats and conventions, so it is your responsibility to adapt these examples to the needs of your individual laboratories and motion capture systems.

Next: Marker (.trc) Files

Home: Preparing Your Data

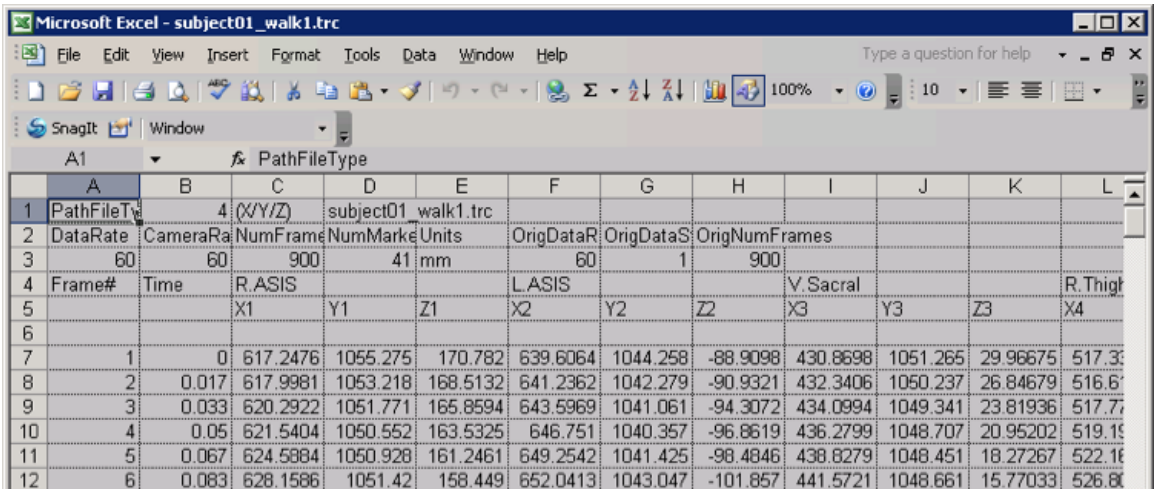
Marker (.trc) Files

The topics covered in this section include:

- Overview
- Representing Marker Data in a .trc File

Overview

The *.trc* (Track Row Column) file format was created by Motion Analysis Corporation to specify the positions of markers placed on a subject at different times during a motion capture trial. An example .trc file (subject01_walk1.trc) is provided in the examples/Gait2354_Simbody directory, which is part of the OpenSim distribution. A fragment of this file is shown in the figure below.



	A	B	C	D	E	F	G	H	I	J	K	L
1	PathFileTy	4	(X/Y/Z)	subject01_walk1.trc								
2	DataRate	CameraRa	NumFrame	NumMarke	Units	OrigDataR	OrigDataS	OrigNumFrames				
3	60	60	900	41	mm	60	1	900				
4	Frame#	Time	R.ASIS			L.ASIS			V.Sacral			R.Thigh
5			X1	Y1	Z1	X2	Y2	Z2	X3	Y3	Z3	X4
6												
7	1	0	617.2476	1055.275	170.782	639.6064	1044.258	-88.9098	430.8698	1051.265	29.96675	517.3
8	2	0.017	617.9981	1053.218	168.5132	641.2362	1042.279	-90.9321	432.3406	1050.237	26.84679	516.6
9	3	0.033	620.2922	1051.771	165.8594	643.5969	1041.061	-94.3072	434.0994	1049.341	23.81936	517.7
10	4	0.05	621.5404	1050.552	163.5325	646.751	1040.357	-96.8619	436.2799	1048.707	20.95202	519.1
11	5	0.067	624.5884	1050.928	161.2461	649.2542	1041.425	-98.4846	438.8279	1048.451	18.27267	522.1
12	6	0.083	628.1586	1051.42	158.449	652.0413	1043.047	-101.857	441.5721	1048.661	15.77033	526.8

Figure: .trc File. The first few lines of the file are shown here.

The first three lines of the .trc file is a header, followed by two rows of column labels, followed by a blank row, followed by the rows of data. Each row of data contains a frame number followed by a time value followed by the (*x*, *y*, *z*) coordinates of each marker. As a plain text file, a .trc file is commonly tab-delimited. So, for example, the fourth line in the .trc file figure above would look like this in plain text: Frame#<tab>Time<tab>R.ASIS<tab><tab><tab>L.ASIS<tab><tab><tab>V.Sacral… where <tab> indicates each tab character that would be present in the file.

DataRate, 60 in this case, indicates the sampling rate of the data in this .trc file in Hertz. So, for example, from frame 1 to frame 2, the time increases by 1/60 second. **NumFrames**, 900 in this case, indicates the number of frames (rows) of data in the whole .trc file. **OrigDataStartFrame**, 1 in this case, indicates the frame number of the first frame (row) of data in the .trc file. Thus, the time *t* at which frame *f* was captured is *t* = (*f* – OrigDataStartFrame) / DataRate.

Note that the fourth row, which contains column labels, contains the name of each marker in the first column where the marker's coordinates appear. In the fifth row, the individual coordinates of each marker are actually labeled as **X1**, **Y1**, **Z1**, **X2**, **Y2**, **Z2**, etc. If this format for the fourth and fifth rows is not followed by a .trc file, OpenSim may fail to read the file's data correctly.

The manual for Motion Analysis' EVaRT (EVa Real-Time Software) contains a description of the .trc file format. For example, in the EVaRT 4.2 manual, the .trc file format is described in Appendix G on pages G-4 and G-5.

Representing Marker Data in a .trc File

As mentioned earlier, marker data must be imported into OpenSim in a .trc file with all marker coordinates transformed to the model coordinate system. An example .trc file (subject01_walk1.trc) is provided in the examples/Gait2354_Simbody directory, which is part of the OpenSim distribution.

In addition, the chapter on Scaling has details on how to represent the marker set in an XML file. It is important that the names of the markers given in the XML file describing the marker set be exactly the same as the names of the markers in the fourth line of the .trc file.

Next: Motion (.mot) Files

Previous: Coordinates and Utilities

Home: Preparing Your Data

Motion (.mot) Files

The topics covered in this section include:

Overview

The *.mot* (motion) file format was created by the developers of SIMM (Software for Interactive Musculoskeletal Modeling). The .mot file format is compatible with both SIMM and OpenSim. A .mot file consists of two parts: the motion header and the data. The motion header can come in two forms: (1) SIMM header only or (2) OpenSim and SIMM header.

Form (1) of a motion header looks like this:

```
name subject01_walk1_grf.mot
datacolumns 19
datarows 9009
range 0.000000 15.013300
endheader
```

The first line must start with **name** followed by a space and the name of the .mot file. The next line should contain **datacolumns**, a space, and then the total number of columns of data in the .mot file (including the time column). The next line should contain **datarows**, a space, and then the total number of rows of data in the .mot file. A later line should contain **range**, a space, the first time value in the time column, a space, and then the last time value in the time column. Optionally, other comments could be included in subsequent lines. The final line **endheader** indicates the end of the header.

Form (2) of a motion header looks like this:

```
Coordinates
nRows=500
nColumns=24
# SIMM Motion File Header:
name Coordinates
datacolumns 24
datarows 500
otherdata 1
range 0.750000 1.249000
Units are S.I. units (second, meters, Newtons, ...)
Angles are in degrees.
endheader
```

The first line is the name, which is **Coordinates** in this case, to be used to represent this .mot file when it is loaded into OpenSim. This does not have to be the name of the .mot file. The second line contains **nRows=** followed by the number of rows of data in the .mot file. The third line contains **nColumns=** followed by the number of columns of data (including the time column) in the .mot file. The fourth line is empty. The fifth line has a comment indicating that the SIMM motion file header is beginning, and then the following lines should have the same format as form (1) of the .mot motion header.

Note that extra lines containing newline characters or comments can be included before the **endheader** line in the SIMM header section of both forms (1) and (2) of the .mot motion header.

Immediately after the **endheader** line, the data section of the .mot file begins. The first line after the **endheader** line should contain tab-delimited labels for each column of (tab-delimited) data in the .mot file. The first column is usually **time**, followed by values that vary with time such as generalized coordinates, marker coordinates, ground reaction forces and moments, centers of pressure, muscle activations, or muscle lengths. In old SIMM .mot files, there is no time column, in which case OpenSim uses the **range** specified in the SIMM header and assumes a fixed time step to specify the time interval for the motion. The names of the column labels should match the names used in the model with which the .mot file is intended to be used. The rows below this line of column labels must be the corresponding values of each of these quantities at the time represented by the first number in each row.

The time values in the time column of a .mot file must be uniformly spaced. An example .mot file (subject01_walk1_grf.mot) is provided in the examples/Gait2354_Simbody directory, which is part of the OpenSim distribution.

Representing Joint Angles in a .mot File

Optionally, if joint angles have been computed previously using other software, they may be imported into OpenSim in addition to marker data. The joint angles must be provided in a .sto or .mot file. The .mot file must contain a header. Below the header, there must be a row of column labels, and the corresponding columns of data below that. Time must be the first column and the generalized coordinates of the model must be the subsequent columns. Angles in .mot files are assumed to be in degrees.

Representing Ground Reaction Data in a .mot File

You need to represent your ground reaction data in a .mot file for input into OpenSim. An example file (subject01_walk1_grf.mot) is given in the examples/Gait2354_Simbody directory, which is part of the OpenSim distribution.

The first row below the header in the .mot file may contain the following column headings. As of OpenSim version 2.3.2 the column headers need to be unique (the program will ask you to save the file in this format first if using a file that doesn't conform to this format). Also each group of three force, point or torque columns should have a common prefix.:

time, ground_force_vx, ground_force_vy, ground_force_vz,

ground_force_px, ground_force_py, ground_force_pz,

l_ground_force_vx, l_ground_force_vy, l_ground_force_vz,

l_ground_force_px, l_ground_force_py, l_ground_force_pz,

ground_torque_x, ground_torque_y, ground_torque_z,

l_ground_torque_x, l_ground_torque_y, l_ground_torque_z

All rows below this line contain the corresponding data in each column. Columns 2-4 (ground_force_vx, ground_force_vy, ground_force_vz) represent the *x*, *y*, and *z* components of the right foot's ground reaction force vector in a specific body coordinate system (ground for the model coordinate system). Columns 5-7 represent the *x*, *y*, and *z* components of the right foot's center of pressure (i.e., the point at which the ground reaction force is applied to the right foot). Similarly, columns 8-13 represent the ground reaction force vector and center of pressure for the left foot. Columns 14-16 represent the *x*, *y*, and *z* components of the right foot's ground reaction moment vector (typically in the model coordinate system). The last three columns represent the analogous quantities for the left foot.

Next: Storage (.sto) Files

Previous: Marker (.trc) Files

Home: Preparing Your Data

Storage (.sto) Files

The *.sto* file format was created by the developers of OpenSim. It is very similar to the .mot file format, with two main differences:

- The time values in the time column of a .sto file do not have to be uniformly spaced
- The first column of a .sto file *must* contain time, whereas a .mot file can contain other quantities in the first column

There is only one format for the header of a .sto file and it is very simple, as shown below:

```
Coordinates
nRows=153
nColumns=24
endheader
```

The first line contains the name with which the .sto file will be referred to when it is loaded into OpenSim. The second line is **nRows=** followed by the number of rows of data in the .sto file. The third line is **nColumns=** followed by the number of columns of data in the .sto file (including the time column). The last line is **endheader**. Immediately following the **endheader** line is the data section of the .sto file, which is identical to the data section of a .mot file, except that the time column is allowed to have non-uniform spacing.

Example .sto files, such as subject01_walk1_RRA_Actuation_force.sto, are provided in the examples/Gait2354_Simbody/OutputReference/ResultsRRA directory, which is part of the OpenSim distribution. As of version 2.3.2, the sto files have a header that includes:

- Version number (version=1)
- Whether angular data specified in the file are in radians or degrees (inDegrees=yes/no). OpenSim would assume that old sto files are in radians, if this is not the case you can manually change this flag in the file or use the Help->Convert Files option to set it.

Next: Previewing Motion Capture (Mocap) Data

Previous: Motion (.mot) Files

Home: Preparing Your Data

Previewing Motion Capture (Mocap) Data

Motion capture data can be previewed in OpenSim to verify that preprocessing was done correctly and that data is in agreement with the intended model. If you have multiple files representing different pieces of data, this tool allows you to verify that the data was transformed consistently. Since the goal is to ensure that the data is consistent with an OpenSim model for further analysis, this module handles data files in the format accepted by OpenSim tools. The two types of data currently handled are marker trajectories (contained in *.trc* files) and measured forces (contained in *.mot* files), described earlier. The topics covered in this section include:

Using the Data Previewer

To choose the motion capture data file to be visualized, select **Preview Motion Data…** from the **File** menu. Once selected, OpenSim performs the following actions:

- Adds a new model to the Navigator with the default name of "ExperimentalData" and a unique number so that multiple data files can be uniquely identified. The model has no display-offset in the GUI to be faithful to the data.
- Makes the loaded motion capture data file the *current* motion.

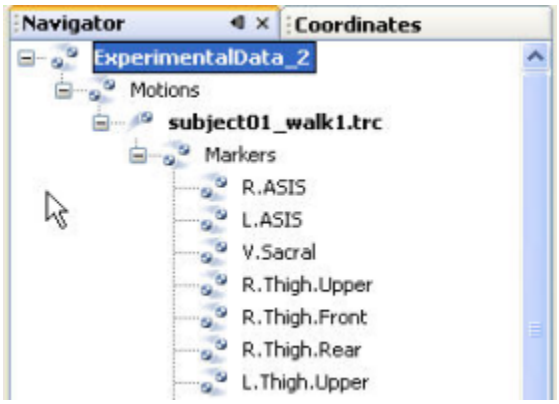
This enables you to perform the following tasks:

- Use the motion slider in the toolbar to go through different frames
- Synchronize the motions from different motion files. These other motions could be either motion capture files or results from OpenSim tools. For example, marker trajectories and ground reaction forces can be synchronized and superimposed on the result of a forward simulation.

Visualizing Marker Trajectories in OpenSim

Marker trajectories are expected to be contained within a *.trc* file. The ExperimentalData model will contain the loaded motion file (e.g., subject01_walk1.trc) with marker data. The **Markers** node will list each individual marker found in the file (figure below). Nodes corresponding to individual markers have the following options, accessible by right-clicking the mouse:

- **Show**: Enabled only if a marker is hidden
- **Show Only**: Hides all other markers except for those selected
- **Toggle Trail Display**: Toggles the display of a line representation of the trajectory of selected marker(s)



Navigator View of a Trajectory Data File

Visualizing External Forces in OpenSim

External forces (such as ground reaction forces) in OpenSim are contained in a *.mot* file. The header of the selected force (*.mot*) file is the same as that expected by OpenSim tools (Motion (.mot) Files).

For the ground reaction forces used by the gait model, the column labels are shown below (for 2 external forces force1, force2, torque1, torque2):

ground_force1_vx, ground_force1_vy, ground_force1_vz,

ground_force1_px, ground_force1_py, ground_force1_pz,

ground_force2_vx, ground_force2_vy, ground_force2_vz,

ground_force2_px, ground_force2_py, ground_force2_pz,

ground_torque1_x, ground_torque1_y, ground_torque1_z,

ground_torque2_x, ground_torque2_y, ground_torque2_z

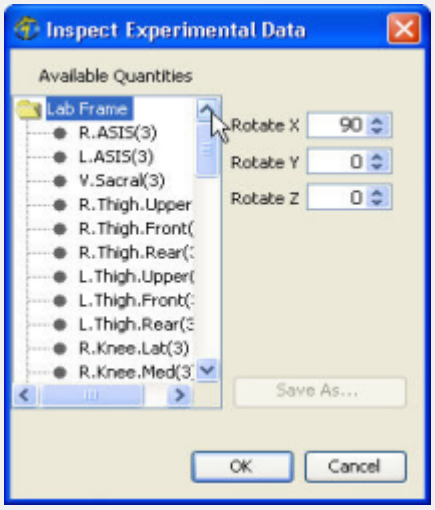
The data previewer expects groups of 6 columns for a force of the form “body”_”ForceName”_{vx, vy, vz, px, py, pz}, where *vi* corresponds to each component of the force and *pi* corresponds to each component of the location of the force (e.g., center of pressure) and 3 columns for a torque of the form “body”_”TorqueName”_{x, y, z}, corresponding to each component of the applied torque. Note that this naming convention is only necessary for previewing purposes. OpenSim tools employ a new user interface enabling you to specifiy any number of forces, along with points of application, or torques to a model during any simulation or analysis.

Previewing Transformed Data

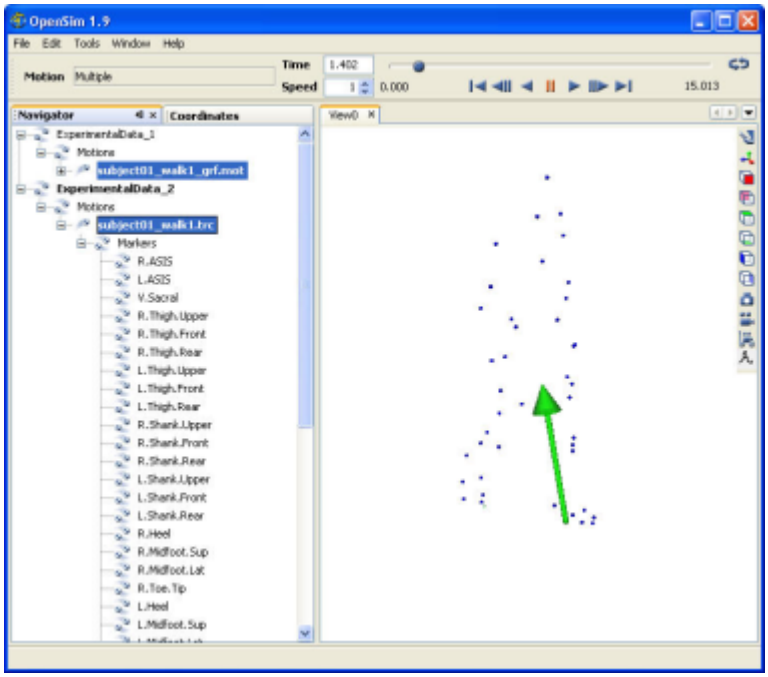
For any kind of previewed data (e.g., marker trajectories or ground reaction forces), you can visualize the effect of a rigid-body-transform applied to the data and save the data as a new file.

> ⚠ **OpenSim does not modify the data contained in the file**.

However, the visualization is transformed to enable you to modify the preprocessing tools, as needed. To access the visual transformation dialog window:



- **Right-mouse click** on the navigator node corresponding to the motion file viewed.



- Use the arrow buttons to change the angle for **Rotate X**, **Rotate Y**, and **Rotate Z**; this will transform the data in the viewer.
- Angles are always positive (0-360 degrees).
- Translations can be previewed by modifying the display offset.

The data file itself is NOT modified by OpenSim; the data previewer just displays the transformed data, so that the appropriate transformation(s) may be determined and applied during preprocessing.

Transform Data Previewer Window, Preview of Motion Capture Data in OpenSim

Next: OpenSim Models

Previous: Storage (.sto) Files

Home: Preparing Your Data

OpenSim Models

UNDER CONSTRUCTION

The topics covered in this section include:

You can also find more information in the Example - Model Editing.

Overview

An OpenSim model represents the dynamics of a system of rigid bodies and joints that are acted upon by forces to produce motion. The OpenSim model file is made up of components corresponding to parts of the physical system. These parts are: **bodies, joints, forces, markers, constraints, contact geometry, and controllers.**

In OpenSim, the skeletal part of a model is represented by rigid bodies interconnected by joints. Joints define how a body (e.g., bone segment) can move with respect to its parent body. In OpenSim, all bodies have a parent and are connected to its parent via a joint, except for ground. Constraints can also be applied to limit the motion of bodies.

Muscles are modeled as specialized force elements that act at muscle points (e.g., insertion and origin points) connected to rigid bodies. The force of a muscle is typically dependent on the path through muscle points comprised of muscle fiber and tendon lengths, the rate of change of the fiber lengths, and the level of muscle activation. OpenSim also has a variety of other forces, which represent externally applied forces (e.g. ground reaction forces), passive spring-dampers (e.g., ligaments), and controlled linear and torsional actuators.

The figure below shows a conceptual schematic of an OpenSim model. In the remainder of the chapter, we will discuss the OpenSim model file format used to describe these models.

OpenSim Model File Format

An OpenSim model is described by a file that utilizes the XML code structure to organize its contents. XML uses tags to identify and manage information, such as:

```
<body> r_humerus </body>
```

where **<body>** signifies the opening of the tag, **r_humerus** is the name of a body in the model, and **</body>** signifies the end of the tag. The name of the tag identifies the type of information between described. When you edit an OpenSim model file, there are tags representing each part

of the model, as shown in the figure below.

> To view and edit XML files with colored tags, you can use **Notepad++** on PC or **EditiX** on Mac, both of which are freely available.

An OpenSim Model File (*arm26.osim*). The file below was opened in the XML editor Notepad++, which provides the color coding. The sections have been collapsed to highlight the model components. Clicking on the + icon to the left of a section would expand it, displaying the relevant tags for that section.

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <OpenSimDocument Version="10905">
3        <Model name="arm26">
4            <defaults>
276          <credits> Model authors names.. </credits>
277          <publications> List publications... </publications>
278          <length_units> meters </length_units>
279          <force_units> N </force_units>
280          <!--Acceleration due to gravity.-->
281          <gravity> 0.0 -9.80665 0.0 </gravity>
282          <!--Bodies in the model.-->
283          <BodySet name="">
608          <!-- Forces in the model -->
609          <ForceSet name="">
1432         <!--Constraints in the model.-->
1433         <ConstraintSet name="">
1437         <!--Markers in the model.-->
1438         <MarkerSet name="">
1506         <!--ContactGeometry objects in the model.-->
1507         <ContactGeometrySet name="">
1511     </Model>
1512 </OpenSimDocument>
```

Bodies

In formulating the equations-of-motion (i.e., the system dynamics), OpenSim employs Simbody which is an open-source multibody dynamics solver. In Simbody and OpenSim, the body is the primary building block of the model. Each body in turn owns a joint and that joint defines the coordinates and kinematic transforms that govern the motion of that body with respect to its parent body. Within the model all bodies are contained in a BodySet.

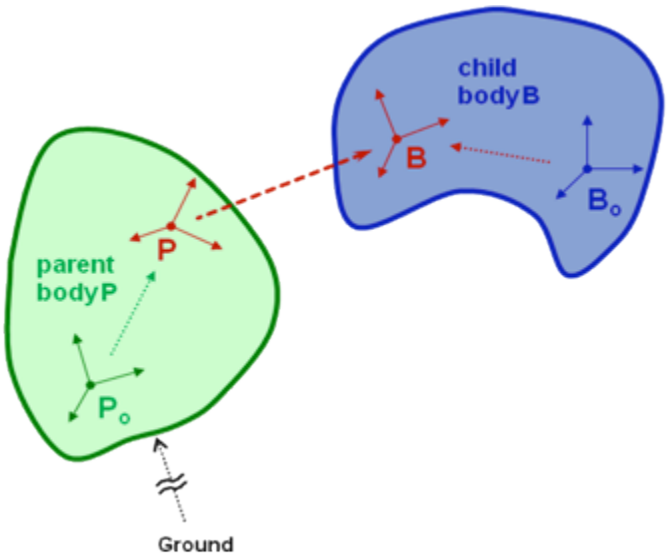
Thus, to start our model, we need to define a set of rigid bodies that represent our system. In the **<BodySet>** section, we define this group of bodies, with the name, mass properties, and visible objects associated with each body. The figure below shows an example of the **r_humerus** body in the *Arm26* model. Note the key tags, such as **<mass>, <mass_center>, <geometry_files>,** and **<inertia_xx> (**and similarly named tags for inertia in other directions).

Example XML Code from Model *Arm26* to Represent a *Body*

```
<Body name="r_humerus">
    <!-- Mass Properties -->
    <mass>        1.86457200 </mass>
    <mass_center>        0.00000000       -0.18049600        0.00000000 </mass_center>
    <inertia_xx>        0.01481000 </inertia_xx>
    <inertia_yy>        0.00455100 </inertia_yy>
    <inertia_zz>        0.01319300 </inertia_zz>
    <inertia_xy>        0.00000000 </inertia_xy>
    <inertia_xz>        0.00000000 </inertia_xz>
    <inertia_yz>        0.00000000 </inertia_yz>
    <!-- Set visable objects (i.e., bone files) associated with this body -->
    <VisibleObject name="">
        <geometry_files> arm_r_humerus.vtp </geometry_files>
        <VisibleProperties name="">
            <display_preference> 4 </display_preference>
            <show_normals> false </show_normals>
            <show_axes> false </show_axes>
            <material_name> DEFAULT </material_name>
        </VisibleProperties>
        <scale_factors>        1.00000000        1.00000000        1.00000000 </scale_factors>
    </VisibleObject>
    <!--Joint that connects this body with the parent body.-->
    <Joint>
    <!-- Wrapping objects associated with this body-->
    <WrapObjectSet name="">
</Body>
```

Joints

In addition to the set of rigid bodies, we also need to define the relationship between those bodies (i.e., joint definitions). In the figure below, a joint (in red) defines the kinematic relationship between two frames (B and P) each affixed to a rigid-body (the parent, P_o, and the body being added, B_o) parameterized by joint coordinates



A body is a moving reference frame (B_o) in which its center-of-mass and inertia are defined, and the location of a joint frame (B) fixed to the body can be specified. Similarly, the joint frame (P) in the parent body frame (P_o) can also be specified. Flexibility in specifying the joint is achieved by permitting joint frames that are not coincident with the body frame.

As an example, the *body* **r_humerus** contains the *joint* **r_shoulder**. The figure below shows an example from *Arm26* defining the **r_shoulder** joint. Note the key tags, such as **<parent_body>, <location_in_parent>, <orientation_in_parent>, <location>, <orientation>,** and **<Coordinate>.**

Example XML Code from Model *Arm26* to Represent a *Joint:*

```
<!--Joint that connects this body with the parent body.-->
<Joint>
    <CustomJoint name="r_shoulder">
        <parent_body> ground </parent_body>
        <location_in_parent>      -0.01754500      -0.00700000       0.17000000 </location_in_parent>
        <orientation_in_parent>       0.00000000       0.00000000       0.00000000 </orientation_in_parent>
        <location>       0.00000000       0.00000000       0.00000000 </location>
        <orientation>       0.00000000       0.00000000       0.00000000 </orientation>
        <!--Generalized coordinates parameterizing this joint.-->
        <CoordinateSet name="">
            <objects>
                <Coordinate name="r_shoulder_elev">
                    <!--Cooridnate can describe rotational, translational, or coupled values.
                    Defaults to rotational.-->
                    <motion_type> rotational </motion_type>
                    <default_value>       0.00000000 </default_value>
                    <default_speed_value>       0.00000000 </default_speed_value>
                    <initial_value>       0.00000000 </initial_value>
                    <range>      -1.57079633       3.14159265 </range>
                    <clamped> false </clamped>
                    <locked> false </locked>
                    <prescribed_function/>
                </Coordinate>
            </objects>
            <groups/>
        </CoordinateSet>
        <reverse> false </reverse>
        <SpatialTransform name="">
    </CustomJoint>
</Joint>
```

Available Joint Types

1. WeldJoint: introduces no coordinates (degrees of freedom) and fuses bodies together
2. PinJoint: one coordinate about the common Z-axis of parent and child joint frames
3. SliderJoint: one coordinate along common X-axis of parent and child joint frames
4. BallJoint: three rotational coordinates that are about X, Y, Z of B in P
5. EllipsoidJoint: three rotational coordinates that are about X, Y, Z of B in P with coupled translations such that B traces and ellipsoid centered at P
6. FreeJoint: six coordinates with 3 rotational (like the ball) and 3 translations of B in P
7. CustomJoint: user specified 1-6 coordinates and user defined spatial transform to locate B with respect to P

The CustomJoint Transform

Most joints in an OpenSim model are custom joints since this is the most generic joint representation, which can be used to model both conventional (pins, slider, universal, etc…) as well as more complex biomechanical joints. The user must define the transform (rotation and translation) of the child in the parent (B and P, in the joint definition figure above) as a function of the generalized coordinates listed in the Joint's CoordinateSet. Consider the spatial transform ${}^{P}\mathbf{X}^{B}$:

$$ {}^{P}\mathbf{X}(q)^{B} = \begin{bmatrix} {}^{P}\mathbf{R}^{B}(x_1, x_2, x_3) & \begin{matrix} x_4 \\ x_5 \\ x_6 \end{matrix} \end{bmatrix} $$

where

$$ x(q) = \begin{Bmatrix} f_1(q_1, q_2, \ldots, q_m) \\ f_2(q_1, q_2, \ldots, q_m) \\ \vdots \\ f_6(q_1, q_2, \ldots, q_m) \end{Bmatrix}, $$

q are the joint coordinates, and x are the spatial coordinates for the rotations (x_1, x_2, x_3) and translations (x_4, x_5, x_6) along user-defined axes that specify a spatial transform (**X**) according to functions f_i. The behavior of a CustomJoint is specified by its SpatialTransform. A SpatialTransform is

comprised of 6 TransformAxes (3 rotations and 3 translations) that define the spatial position of B in P as a function of coordinates. Each transform axis enables a function of joint coordinates to operate about or along its axis. The function of *q* is used to determine the displacement for that axis. The order of the spatial transform is fixed with rotations first followed by translations. Subsequently, coupled motion (i.e., describing motion of two degrees of freedom as a function of one coordinate) is easily handled. The example below (from the gait2354.osim model) describes coupled motion of the knee, with both tibial translation and knee flexion described as a function of knee angle:

Spatial transform of a custom joint that implements a translating knee joint:

```
<SpatialTransform name="">
  <!--3 Axes for rotations are listed first.-->
  <TransformAxis name="rotation1">
    <function>
      <LinearFunction name="">
        <coefficients> 1.00000000 0.0000000 </coefficients>
      </LinearFunction>
    </function>
    <coordinates> knee_angle_r </coordinates>
    <axis> 0.00000000 0.00000000 1.00000000 </axis>
  <TransformAxis>

  <TransformAxis name="rotation2">
    <function>
      <Constant name="">
        <value> 0.00000000 </value>
      </Constant>
    </function>
    <coordinates> </coordinates>
    <axis> 0.00000000 1.00000000 0.00000000 </axis>
  </TransformAxis>

  <TransformAxis name="rotation3"> ...

  <!--3 Axes for translations are listed next.-->
  <TransformAxis name="translation1">
    <function>
      <NaturalCubicSpline name=""> ...
    </function>
    <coordinates> knee_angle_r </coordinates>
    <axis> 1.00000000 0.00000000 0.00000000 </axis>
   </TransformAxis>

  <TransformAxis name="translation2">
    <function>
      <NaturalCubicSpline name=""> ...
    </function>
    <coordinates> knee_angle_r </coordinates>
    <axis> 0.00000000 1.00000000 0.00000000 </axis>
  </TransformAxis>

  <TransformAxis name="translation3"> ...
</SpatialTransform>
</CustomJoint>
</Joint>
```

Kinematic Constraints in OpenSim

OpenSim currently supports three types of built-in constraints: PointConstraint WeldConstraint and CoordinateCouplerConstraint. A point constraint fixes a point defined with respect to two bodies (i.e., no relative translations). A weld constraint fixes the relative location and orientation of two bodies (i.e., no translations or rotations). A coordinate coupler relates the generalized coordinate of a given joint (the dependent coordinate) to any other coordinates in the model (independent coordinates). The user must supply a function that returns a dependent value based on independent values. The following example implements coordinate coupler constraint for the motion of the patella as a function of the knee ankle and also welds the foot to ground.

Example of constraints in OpenSim:

```
<!--Constraints in the model.-->
<ConstraintSet name="">
  <objects>
  <CoordinateCouplerConstraint name="pat_tx_r">
    <isDisabled> false </isDisabled>
    <coupled_coordinates_function>
      <natCubicSpline name="">...
    </coupled_coordinates_function>
    <independent_coordinate_names> knee_angle_r </independent_coordinate_names>
    <dependent_coordinate_name> pat_tx_r </dependent_coordinate_name>
  </CoordinateCouplerConstraint>
  <CoordinateCouplerConstraint name="pat_ty_r">...
  <CoordinateCouplerConstraint name="pat_angle_r">...
  <WeldConstraint name="">
    <isDisabled> false </isDisabled>
    <body_1> ground </body_1>
    <body_2> calcn_r </body_2>
    <location_body_1>       0.0000000000       0.0000000000       0.0840000000
    <orientation_body_1>       0.0000000000       0.0000000000       0.00000000
    <location_body_2>       0.0000000000       0.0000000000       0.0000000000
    <orientation_body_2>       0.0000000000       0.0000000000       0.00000000
  </WeldConstraint>
  </objects>
  <groups/>
</ConstraintSet>
```

Forces

In order to actuate our model, we need to define the *forces* that will be applied to the model. Just like bodies are defined within the **<BodySet>** section, forces are defined in the **<ForceSet>** section of the model file. Forces come in two varieties: *passive* forces like springs, dampers, and contact and *active* forces like springs, idealized linear or torque actuators, and muscles. Active forces that require input (controls) supplied by the user or by a controller are called Actuators and are a subset of the ForceSet.

Available Forces

OpenSim has several built-in forces that include: PrescribedForce, SpringGeneralizedForce, BushingForce, as well as HuntCrossleyForce and ElasticFoundationForce to model forces due to contact (Note: contact forces also require defining contact geometry). Below is an example of a bushing force used to model passive structures surrounding a single lumbar joint that connects a torso body to a pelvis body.

Example of a passive BushingForce:

```
<!-- Generate a force proportional to the separation of two frames in terms of both relative
rotational and translational displacement (stiffness) and velocity and velocity (damping) -->
<BushingForce name="BackJointBushing">
 <body_1> pelvis </body_1>
 <body_2> torso </body_2>
 <location_body_1> -0.1007 0.0815 0.0000 </location_body_1>
 <orientation_body_1> 0.0000 0.0000 0.0000 </orientation_body_1>
 <location_body_2> 0.0000 0.0000 0.0000 </location_body_2>
 <orientation_body_2> 0.0000 0.0000 0.0000 </orientation_body_2>
 <rotational_stiffness> 10.000 10.000 10.000 </rotational_stiffness>
 <translational_stiffness> 0.0000 0.000 0.000 </translational_stiffness>
 <rotational_damping> 0.0000 0.0000 0.0000 </rotational_damping>
 <translational_damping> 0.0000 0.000 0.000 </translational_damping>
</BushingForce>
```

Common Actuators

OpenSim also includes “ideal” actuators which apply pure forces or torques that are directly proportional to the input control (i.e., excitation) via its optimal force (i.e., a gain). Forces and torques are applied between bodies, while generalized forces are applied along the axis of a generalized coordinate (i.e., a joint axis).

Sample of linear and torque actuators in a model's ForceSet:

```
<!--Apply an equal and opposite force at points on two bodies along the line that connects the two
points -->
<PointActuator name="FY_residual">
 <optimal_force> 8.0 </optimal_force>
 <body> pelvis </body>
 <point> -0.0724376 0.00000000 0.00000000 </point>
 <direction> 0.0 1.0 0.0 </direction>
</PointActuator>

<!--Apply an equal and opposite torque on two bodies about the axis defined in the the first body
-->
<TorqueActuator name="MZ_residual">
 <optimal_force> 1000.0 </optimal_force>
 <bodyA> ground </bodyA>
 <axis> 0.000 0.000 -1.000 </axis>
 <bodyB> pelvis </bodyB>
</TorqueActuator >

<!--Apply a generalized force along (force) or about (torque) the axis of a generalized coordinate.
Positive force increases the coordinate -->
<CoordinateActuator name="knee_reserve">
 <optimal_force> 300.0 </optimal_force>
 <coordinate> knee_angle_r </coordinate>
</CoordinateActuator>
```

The Muscle Actuator

There are several muscle models in OpenSim. All muscles include a set of muscle points where the muscle is connected to bones (bodies) and provide utilities for calculating muscle-actuator lengths and velocities. Internally muscle models may differ in the number and type of parameters. Muscles typically include muscle activation and contraction dynamics and their own states (for example activation and muscle fiber length). The control values are typically bounded excitations (ranging from 0 to 1) which lead to a change in activation and then force. Below is an example of a muscle model, as described by Thelen (2003), from an OpenSim model.

In addition to the muscle properties, we need to define its geometry. In this example, a *geometry path* is defined for the *muscle* using a set of *path points*.

Sample muscle actuator from a model's ForceSet:

```
<Thelen2003Muscle name="soleus_r">
 <GeometryPath name="">
   <!—- points on bodies that define the path of the muscle -->
   <PathPointSet name="">
     <objects>
       <PathPoint name="soleus_r-P1">
         <location> -0.00240000 -0.15330000 0.00710000 </location>
         <body> tibia_r </body>
       </PathPoint>

       <PathPoint name="soleus_r-P2">
         <location> 0.00000000 0.03100000 -0.00530000 </location>
         <body> calcn_r </body>
       </PathPoint>
     </objects>
   </PathPointSet>
   <PathWrapSet name=""> ...
 </GeometryPath>

 <!--maximum isometric force of the muscle fibers-->
 <max_isometric_force> 4000.00000000 </max_isometric_force>
 <!--optimal length of the muscle fibers-->
 <optimal_fiber_length> 0.08000000 </optimal_fiber_length>
 <!--resting length of the tendon-->
 <tendon_slack_length> 0.22000000 </tendon_slack_length>
 <!--angle between tendon and fibers at optimal fiber length-->
 <pennation_angle> 0.43633231 </pennation_angle>
 <!--time constant for ramping up of muscle activation-->
 <activation_time_constant> 0.01000000 </activation_time_constant>
 <!--time constant for ramping down of muscle activation-->
 <deactivation_time_constant> 0.04000000 </deactivation_time_constant>
 <!--maximum contraction velocity at full activation (fiber length/s)-->
 <Vmax> 10.00000000 </Vmax>
 <!--maximum contraction velocity at low activation (fiber lengths/s)-->
 <Vmax0> 5.00000000 </Vmax0>
 <!--tendon strain due to maximum isometric muscle force-->
 <FmaxTendonStrain> 0.03300000 </FmaxTendonStrain>
 <!--passive muscle strain due to maximum isometric muscle force-->
 <FmaxMuscleStrain> 0.60000000 </FmaxMuscleStrain>
 <!--shape factor for Gaussian active muscle force-length relationship-->
 <KshapeActive> 0.50000000 </KshapeActive>
 <!--exponential shape factor for passive force-length relationship-->
 <KshapePassive> 4.00000000 </KshapePassive>
 <!--passive damping in the force-velocity relationship-->
 <damping> 0.05000000 </damping>
 <!--force-velocity shape factor-->
 <Af> 0.30000000 </Af>
 <!--maximum normalized lengthening force-->
 <Flen> 1.80000000 </Flen>
</Thelen2003Muscle>
```

Markers

In order to perform Inverse Kinematics, you will need to define a virtual *marker set* that matches the experimental marker set used to collect motion capture data. Markers are defined in a **<MarkerSet>**. The figure below (Example XML Marker) shows an example from *Arm26* defining a **<Marker>**. Note tags that define the marker, such as **<body>** and **<location>**. Additionally, the marker **name** is important, as it must match the name of the corresponding experimental marker.

Example XML Code from Model *Arm26* to Represent a *Marker*

```
<!--Markers in the model.-->
<MarkerSet name="">
    <objects>
        <Marker name="r_acromion">
            <!--Body segment in the model on which the marker resides.-->
            <body> ground </body>
            <!--Location of a marker on the body segment.-->
            <location>       -0.01256000        0.04000000        0.17000000 </location>
            <!--Flag (true or false) specifying whether or not a marker should be kept
              fixed in the marker placement step.  i.e. If false, the marker is
              allowed to move.-->
            <fixed> false </fixed>
            <!--Used for displaying a marker in the visuals.-->
            <VisibleObject name="">
        </Marker>
        <Marker name="r_humerus_epicondyle">
        <Marker name="r_radius_styloid">
    </objects>
    <groups/>
</MarkerSet>
```

Contact Geometry

A model may have some specific contact geometry that is associated with a model. In OpenSim, contact geometry can be an analytical shape, such as a half-place, sphere, or cube, or a user-defined shape represented by an object (***.obj*)** file**.** The figure below shows an example defining contact for the ground (half-space) and a user-defined block from the *tugOfWar* model. Note tags that define the contact object, such as ***<body_name>**, **<location>**, **<orientation>**, and **<filename>**.

Example XML Code from Model *tugOfWar* to Represent *Contact Geometry*

```
<!--ContactGeometry objects in the model.-->
<ContactGeometrySet name="">
    <objects>
        <ContactHalfSpace name="ground">
            <body_name> ground </body_name>
            <location>       0.00000000        0.00000000        0.00000000 </location>
            <orientation>        0.00000000        0.00000000       -1.57079633 </orientation>
        </ContactHalfSpace>
        <ContactMesh name="block">
            <body_name> block </body_name>
            <location>       0.00000000        0.00000000        0.00000000 </location>
            <orientation>        0.00000000        0.00000000        0.00000000 </orientation>
            <filename> blockRemesh192.obj </filename>
        </ContactMesh>
    </objects>
    <groups/>
</ContactGeometrySet>
```

Next: Extending OpenSim's Capabilities

Extending OpenSim's Capabilities

Overview

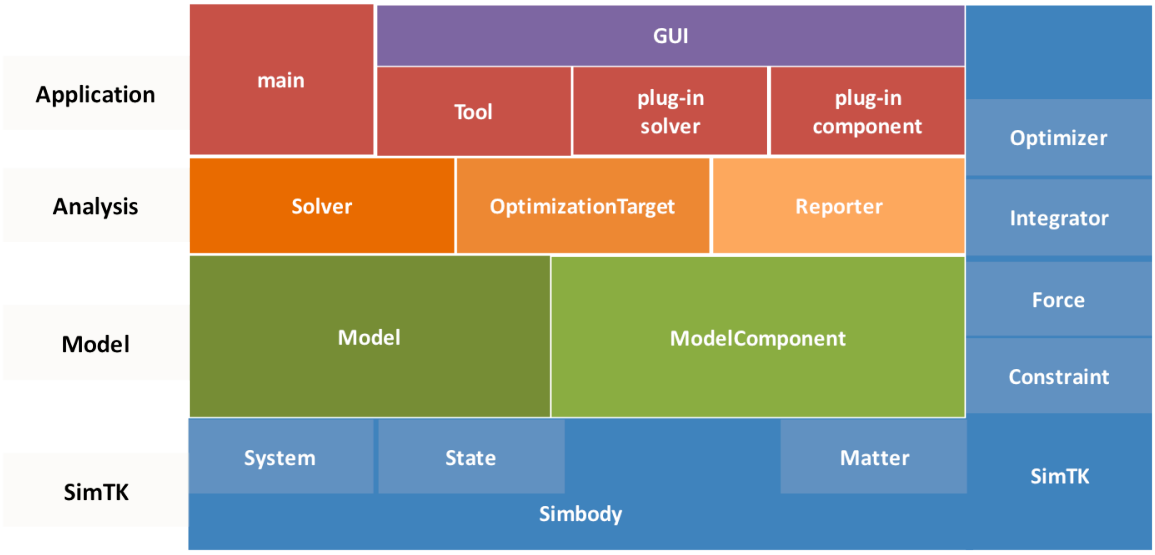
OpenSim provides several mechanisms for extending its existing capabilities either by adding new model elements, computing new quantities, or computing existing quantities in a new way. For example, you may want to model the drag acting on bodies moving through a fluid, which OpenSim does not provide. Another example is being able to extract the linear and angular momentum of the model during a simulation. In order to extend to OpenSim, it is important to know what functionality exists and to have a sense of where to add new functionality.

Organization of OpenSim

OpenSim is built on the computational and simulation core provided by SimTK. This includes low-level, efficient math and matrix algebra libraries such as LAPACK as well as the infrastructure for defining a dynamical system and its state. One can think of the system as the set of differential equations and the state comprised of its variables.

Empowering the computational layer is Simbody^{TM}, an efficient multibody dynamics solver, which provides an extensible multibody system and state. The OpenSim modeling layer maps biomechanical structures (bones, muscles, tendons, etc.) into bodies and forces so that the dynamics of the system can be computed by Simbody.

The three interface layers of OpenSim built on SimTK:

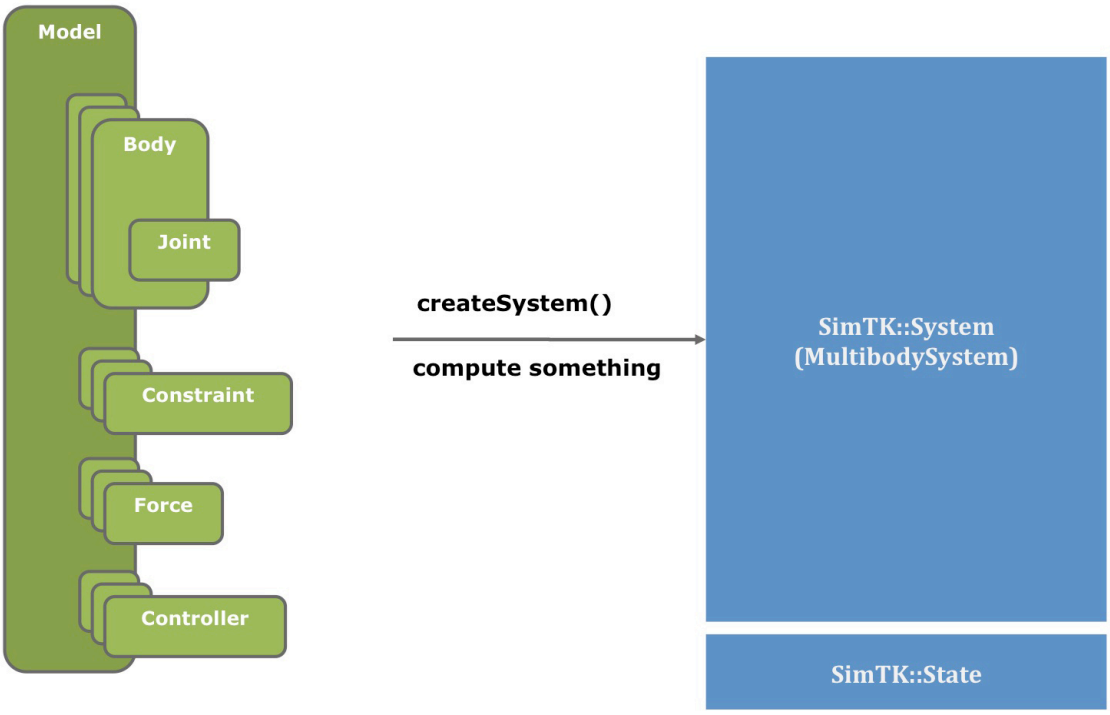


OpenSim is essentially a set of modeling libraries for building complex actuators (e.g. muscles) and other forces (e.g. contact) and enabling the motion (kinematics) of highly articulated bodies (bones). Actuators can then be controlled by model controllers (e.g. Computed Muscle Control) to estimate the neural control and muscle forces required to reproduce human movement. An analysis layer is equipped with solvers and optimization resources for performing calculations with the model and to report results. At the highest level these blocks are assembled into specialized applications (ik.exe, forward.exe, analyze.exe) to simulate and analyze model movement and internal dynamics. The OpenSim application is a Java based program that calls Tools, Models, and underlying computations in SimTK to provide an interactive graphical user-interface (GUI).

OpenSim Model and ModelComponents

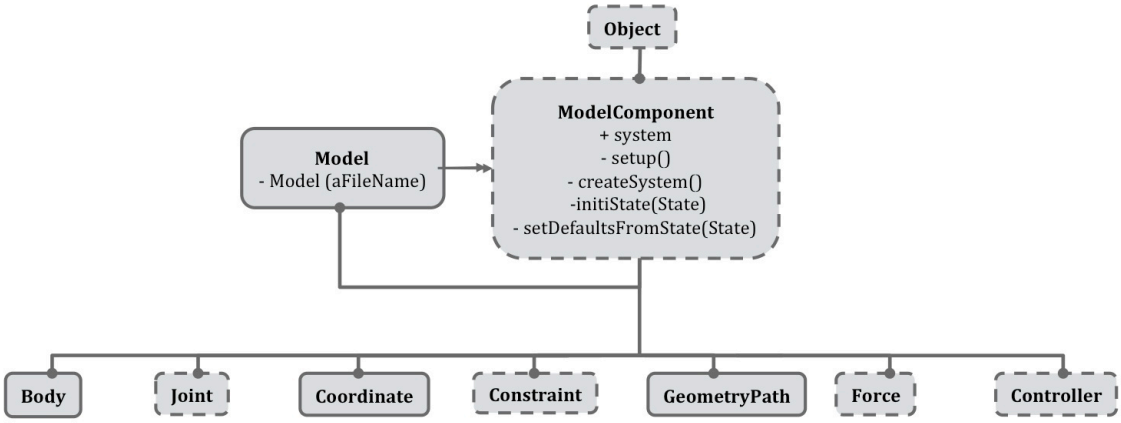
The job of an OpenSim::Model is to organize (hierarchically) the pieces (components) of a musculoskeletal system and to create a representative computational (mathematical) system that can be solved accurately and efficiently using Simbody and the flat SimTK::System framework.

Organizational Context vs. Computation



By separating the contextual organization of a model from its computational representation, OpenSim can exploit the conceptual benefits of hierarchically organized models and software without sacrificing computational efficiency. One can then think of the system as the set of system equations while the state is a coherent set of system variable values that satisfies the system equations. Model components know about the parts they add to the multibody system (for example, another rigid body, a force, or a constraint) and are free to mix and match. For example, a Coordinate component knows how to access its underlying degree-of-freedom value, velocity and even its acceleration, given the system has been "solved" for accelerations. A Coordinate also adds different constraints to the underlying system, in the case that Coordinate is locked or if its motion is prescribed. It provides context to organize locking constraints with the Coordinates being locked, but computationally it is just another constraint equation. The Coordinate therefore acts to manage the bookkeeping (which DOF, constraint, etc.) and provide an interface that has context.

OpenSim Model and its ModelComponents



All model components in OpenSim have a similar responsibility to create their underlying system representation (createSystem()). A setup() method ensures that a model is appropriately defined (for example, a Body is being connected to a parent that exists) before creating the system. Two additional methods allow the ModelComponent to initialize the state of the system (from default properties) and also to hold the existing state in the ModelComponent's defaults. For example, a Coordinate's default may indicate that it should be locked, in which case its initState would set the state of its underlying constraint as "enabled". Similarly, after performing an analysis to find the coordinates to satisfy a static pose, calling setDefaultsFromState(state) will update the Coordinate's default values for the coordinate value from the desired state. Next time the model is initialized, it will be in the desired pose.

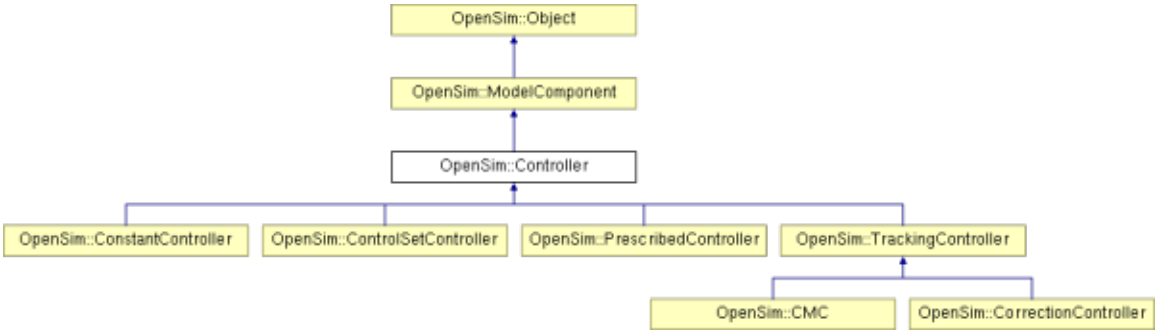
OpenSim Application Programming Interface (API)

In order to build custom components, it is necessary to have a general understanding of which objects (classes) are responsible for what actions/behaviors. The functions (methods) that OpenSim's public classes provide (that other applications/programs can call) define its Application Programming Interface or API.

We have already seen four methods that a model component must implement to behave as a ModelComponent in OpenSim. This defines the ModelComponent interface. Each type of ModelComponent, in turn, specifies additional methods in order to satisfy that type of component. For example, a Force in OpenSim must implement a computeForce() method (in addition to the ModelComponent methods), a Controller must implement computeControls(), etc. The set of all Classes and their interfaces defines the OpenSim API.

The OpenSim API is undergoing rapid development and improvement. We therefore rely on Doxygen to automatically generate html documentation of the latest source, which describe the classes that are available and the accessible methods. The Doxygen pages can be viewed using a web browser and are available with your OpenSim installation in: <OpenSim_Install_Dir>/sdk/doc/index.html. This provides the latest organization of the available classes where one can see the list of available controllers, for example.

Example Doxygen documentation for available controllers:



What is an OpenSim plug-in?

When creating a new component (like a force, controller) or a new analysis, you may want to include it in an existing model, run it with existing tools, and/or share your contribution with colleagues. An OpenSim plug-in is a way of packaging of your code in a dynamically linked library so that an existing OpenSim application can recognize it, load it, and make your code "runnable". For an example of creating an analysis as a plug-in please see <OpenSim_Install_Dir>/sdk/examples/plugin.

What is an OpenSim "main" program?

A main program in C/C++ results in a standalone executable that you can run from a command prompt or by double clicking in Windows. All C/C++ programs have a main() function, which can be as simple as printing "Hello World" or it can invoke several libraries to produce complex applications, like Word and Excel. By including the OpenSim libraries, your main program can call the OpenSim API, and you may also include any other (C++) libraries that provide additional computational and/or visualization resources. Main programs are extremely flexible, but they are particularly useful for streamlining/automating processes independent of the GUI. For example, ik.exe, id.exe, and cmc.exe (available with the OpenSim distribution) are main programs that take setup files and perform tasks related to the OpenSim workflow. Alternatively, users have created their own main programs to systematically scale strengths of all muscles in a model, run forward simulations with their own controllers, perform design optimizations, etc. An advantage of a main program (compared to a plug-in) is that any classes you define in the project are immediately useable by your program. This can make prototyping and testing of your new component or analysis faster and easier without having to wrap, load, and call your plug-in from the GUI.

OpenSim Developer's Guide

The developer's guide provides a step-by-step example of calling the OpenSim API to build a model, including muscles and contact forces, and to perform a simulation in a main program. Please refer to the OpenSim Developer's Guide for more details.

Command Line Utilities

All of the OpenSim Tools are available as command-line utilities that take as input the same setup (or settings) file loaded into or saved from the OpenSim GUI application. For example, to perform Inverse Kinematics from the command line (the Command Prompt in Windows) one can execute the following command:

```
ik –S arm26_Setup_InverseKinematics.xml
```

Similarly, this command line arguments work for CMC or any other tool, with the complete set of command line executables available in <OpenSim_Install_Dir>/bin. In addition to the *–**S**etup* option, there are *–**H**elp* , *-**P**rint**S**etup* and *–**P**roperty**I**nfo* options. Help provides this list of options. Print Setup prints a default setup file for that Tool with all available properties (XML tags) for Tool settings.

The –***P***roperty***I***nfo option can be a very handy resource to obtain information about existing settings for Tools and ModelComponents including the XML tags needed in the model and/or setup file. This is the same information listed in the "Available Objects…" panel under the Help menu in the OpenSim GUI. Executing ik –PI lists all the available classes (components, analyses, utilities and tools) available in OpenSim. For more information about a particular object, such as adding a point constraint to the model, executing

```
ik –PI PointConstraint
```

yields:

```
PROPERTIES FOR PointConstraint (5)
1. isDisabled
2. body_1
3. body_2
4. location_body_1
5. location_body_2
```

The information returned lists the properties for defining a point constraint in OpenSim.

MATLAB Utilities for Data Import

There are several MATLAB scripts for reading .trc, .c3d, .mot, and .sto files into MATLAB and writing out the data file formats required by OpenSim. Scripts provided by the Neuromuscular Biomechanics Lab at Stanford are available on the OpenSim Utilities project on SimTK.org: https://simtk.org/home/opensim-utils. Additional utilities by OpenSim users are posted on SimTK.org and can be found using the search tool on SimTK.org

Video Gallery

Below are video tutorials on the following sections:

Introduction

video pending

Graphical User Interface

video pending

Scaling

- Set up: http://www.youtube.com/watch?v=V5f7OMkfRBs&list=PLFECE3229007D54F6&feature=plcp&context=C3deccf5FDOEgsToPDskJ9Z_urB
- Evaluating your results: http://www.youtube.com/watch?v=7STRMJefzpI&feature=BFa&list=PLFECE3229007D54F6&lf=plcp&context=C3deccf5FDOEgsToPDskJ

Authors

Authors

The following people have contributed to the information on this wiki by helping to develop the user and developer's guide, examples, and other materials:

- Frank Anderson
- Scott Delp
- Matt DeMers
- Eran Guendelman
- Ayman Habib
- Samuel Hamner
- Jennifer Hicks
- Jill Higginson
- Katherine Holzbaur
- Chand John
- Marjolein van der Krogt
- Joy Ku
- May Liu
- Katie Lund
- Peter Loan
- Jeff Reinbolt
- Ajay Seth
- Michael Sherman

Acknowledgements

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.

The OpenSim project is also supported by the National Center for Simulation in Rehabilitation Research (NCSRR) is a National Center for Medical Rehabilitation Research supported by research infrastructure grant R24 HD065690 from the National Institute of Child Health & Human Development and the National Institute of Biomedical Imaging and Bioengineering. See our People page for a list of the many people who have contributed to the OpenSim project over the years.

Trademarks and Copyright

SimTK and Simbios are trademarks of Stanford University.

The documentation for OpenSim is freely available and distributable under the MIT License.