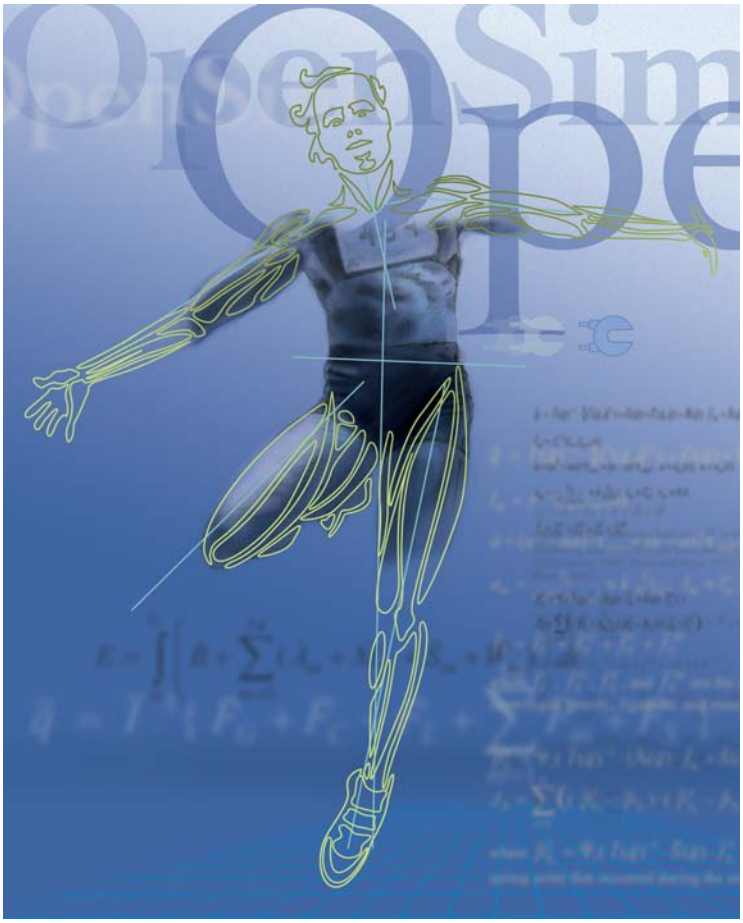
OpenSim



User's Guide

Release 1.0

August 1, 2008

Website: https://simtk.org/home/opensim

OpenSim User's Guide

Authors

Frank Anderson

Eran Guendelman

Ayman Habib

Samuel Hamner

Katherine Holzbaur

Chand John

Joy Ku

May Liu

Peter Loan

Jeff Reinbolt

Ajay Seth

Scott Delp

Trademarks and Copyright and Permission Notice

Acknowledgments

OpenSim was developed as a part of 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.

Table of Contents

List of Figures

chapter

1

Introducing OpenSim

1.1 What is OpenSim?

OpenSim is a freely available software package that enables you to build, exchange, and analyze computer models of the musculoskeletal system and dynamic simulations of movement. OpenSim version 1.0 was introduced at the American Society of Biomechanics Conference in 2007. Since then, hundreds 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, and biology and engineering education.

The software provides a platform on which the biomechanics community can build a library of simulations that can be exchanged, tested, analyzed, and improved through multi-institutional collaboration. The underlying software is written in C++, and the graphical user interface (GUI) is written in Java. OpenSim plug-in technology 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 low-level computational components such as integrators and optimizers to be updated as appropriate without extensive restructuring.

1.2 Capabilities

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

- Taking Pictures and Making Animated Movies (Chapter 8)
- Plotting Results (see Chapter 12)
- Scaling a Model (see Chapter 13)
- Performing Inverse Kinematics Analyses (see Chapter 14)
- Performing Inverse Dynamics Analyses (see Chapter 15)
- Generating Forward Dynamics Simulations (see Chapter 18)
- Analyzing Dynamic Simulations (see Chapter 19)

1.3 Model and Simulation Repository

You can create your own models of musculoskeletal structures and dynamic simulations of movement in OpenSim, as well as take advantage of computer models and dynamic simulations that other users have developed and shared. For example, you can use existing computer models of the human lower limb, upper limb, cervical spine, and whole body which have already been developed and posted at Simtk.org. You can also use dynamic simulations of walking and other activities that have been developed, tested and posted on Simtk.org. We encourage you to share your models and simulations 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.

1.4 Compatibility with SimTK

OpenSim is built using SimTK, an open-source simulation toolkit developed to create mathematical models of biological dynamics. SimTK is being developed by Simbios, an NIH National Center for Biomedical Computation based at Stanford University. The purpose of SimTK is to enable groundbreaking biomedical research by providing open access to high-quality tools for modeling and simulating biological structures. SimTK includes the LAPACK linear algebra library, IPOpt optimizer, and the Simbody multibody dynamics engine. SimTK software and documentation are available at http://simtk.org/home/simtkcore.

1.5 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.

1.6 Graphical User Interface

OpenSim provides a graphical user interface (Figure 1-1) 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.

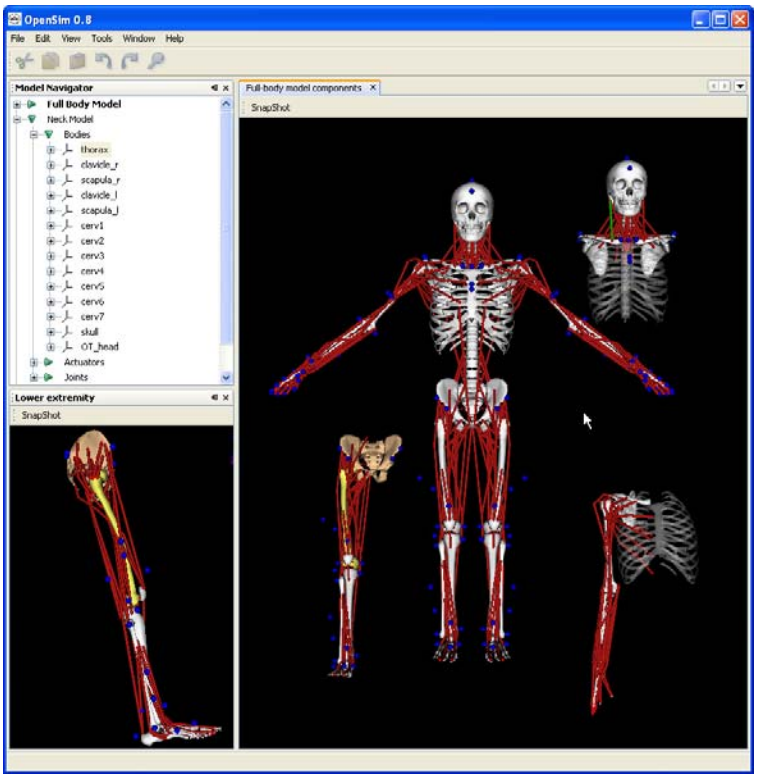


Figure 1-1: Screenshot from OpenSim. Models of many different musculoskeletal structures, including the lower extremity, upper extremity, and neck, can be loaded, viewed and analyzed. Muscles are shown as red lines; virtual markers are shown as blue spheres.

1.7 How to Get Started

It's easy to get started with OpenSim. The instructions in Chapter 2 explain how to download and install the software on your computer. The installation process generally takes less than 10 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.

1.8 What's Covered in this Manual

1.8.1 Organization of This Document

This manual is organized into four distinct sections. The first section (Chapter 2) covers installation of OpenSim. Chapters 3-8 describe how to use the basic components of the OpenSim graphical user interface (GUI). Chapters 9-20 cover the tools available within OpenSim, including plotting, scaling, inverse kinematics, inverse dynamics, forward dynamics, and computed muscle control. Chapter 21 describes how to prepare your data for use within OpenSim.

1.8.2 Conventions Used in This Document

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

1.9 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.

chapter

2

Installing and Running OpenSim

2.1 Overview

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

2.2 Getting OpenSim

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

2.2.1 Log In to Simtk.org

Connect your browser to the Simtk.org (https://simtk.org) home page (Figure 2-1) and log in. 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. 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.

Figure 2-1: Simtk.org login. Connect your browser to https://simtk.org and use links in the top border of the Simtk.org home page to register as a new member, log in, and view details about Simtk.org.

2.2.2 Download Installation Program

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

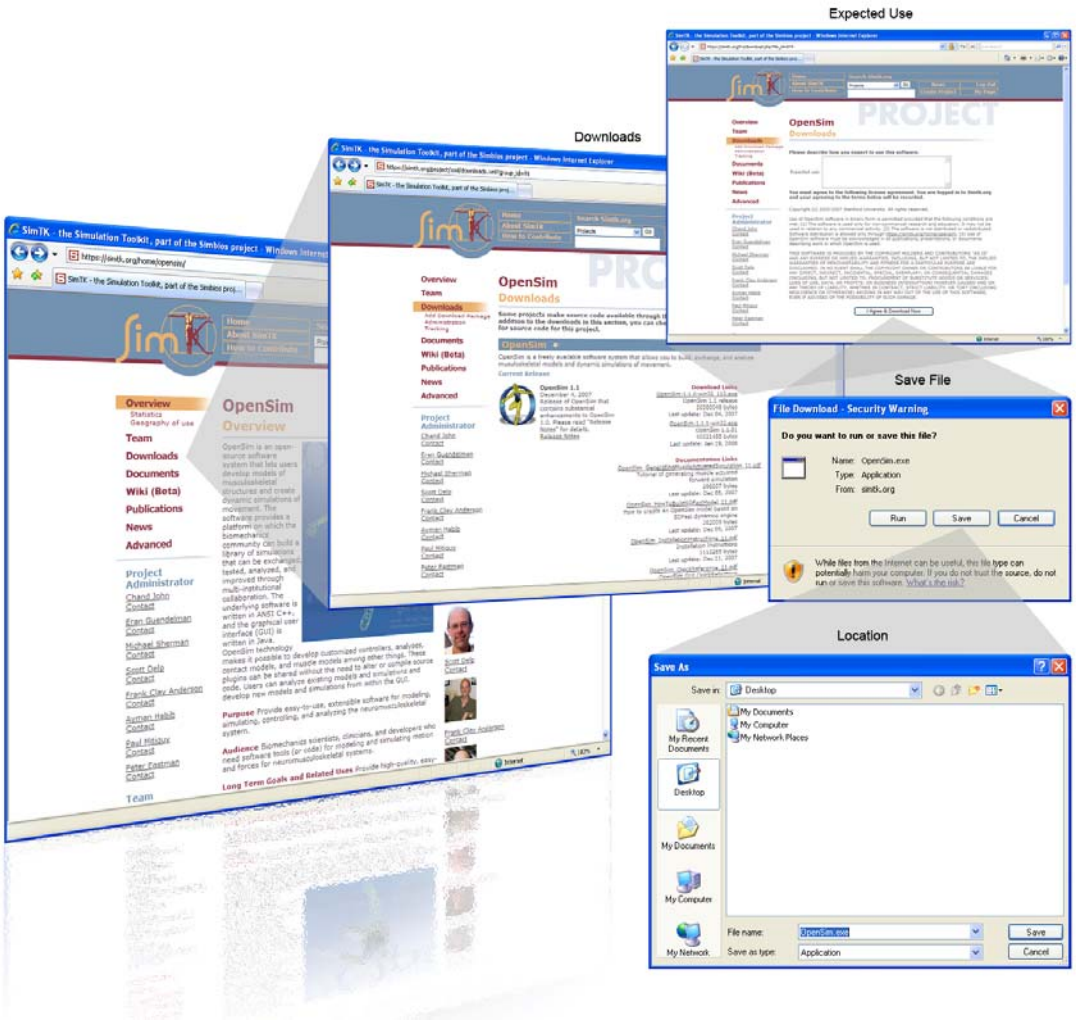


Figure 2-2: OpenSim download. Connect your browser to https://simtk.org/home/opensim and use the Downloads link in the left column of the OpenSim home page to begin the process for downloading the OpenSim installation program.

1. Connect your browser to the OpenSim project home page (https://simtk.org/home/opensim) (Figure 2-2).
2. Click on the **Downloads** link in the left-hand column of the OpenSim project home page. You will jump to the Downloads page, where you can view current and previous releases of OpenSim.
3. From the Downloads page, click on the link for the OpenSim executable, located in the Download Links section. This will be the latest release of OpenSim.

4. If you did not already log in (see Section 2.2.1), a new page will appear, requiring you to log in.
5. A page will then appear asking you to describe how you expect to use the software and presenting the license agreement for OpenSim. In the text field, describe your intended use of OpenSim and then click the button to accept the license agreement.
6. A dialog box will then appear, asking if you want to run or save the installation program. Click the **Save** button, choosing a convenient location (e.g., your Desktop) to save the installation program. If the dialog box does not appear, check your Browser security settings, which may be set to block pop-ups.

2.3 Installing OpenSim

2.3.1 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.

2.3.2 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 (Figure 2-3).
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.

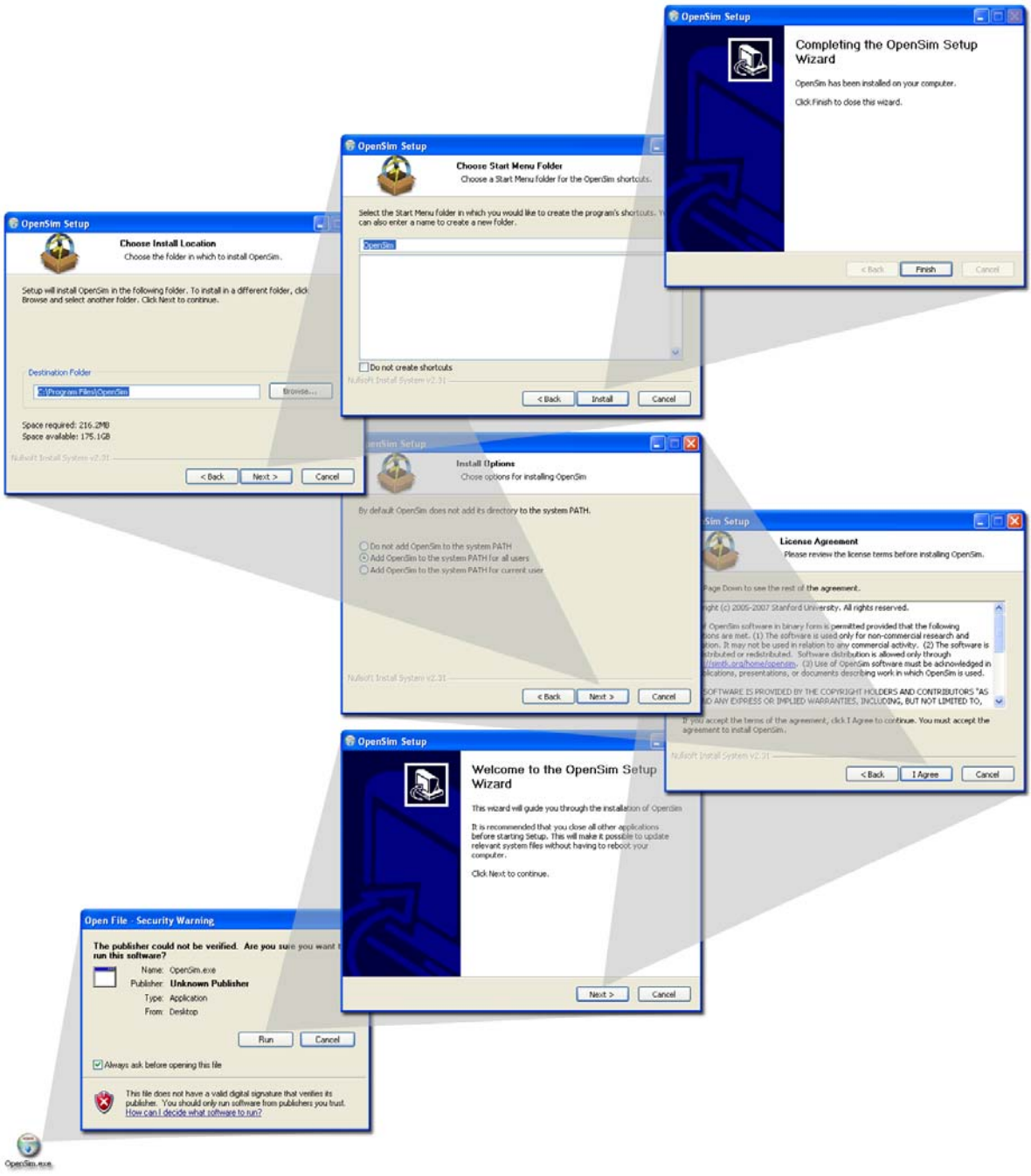


Figure 2-3: OpenSim installation. Run the installation program that you previously saved on your computer. Follow the Setup Wizard prompts to install the OpenSim program.

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.

2.4 Running OpenSim

You can begin using OpenSim immediately after installation. To launch OpenSim, go to the Start Menu. Navigate to the folder you chose for installing the OpenSim shortcuts (Figure 2-4). Click the OpenSim shortcut to start the program.

Figure 2-4: Starting OpenSim. From the Start Menu, go to the folder you chose for the OpenSim shortcuts (e.g., All Programs > OpenSim). Click on the icon for OpenSim.

chapter

3

Graphical User Interface

3.1 Overview

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

3.2 Menus

There are five drop-down menus available from the OpenSim main menu bar: File, Edit, Tools, Window, Help. Each of these is described in more detail in the sections below.

3.2.1 File Drop-Down Menu Options

The **File** drop-down menu (Figure 3-1) includes the following options which allow you to input and output information about models and motions:

- Open Model...
- Load Motion...
- Close Model

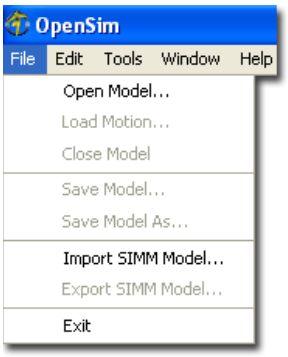


Figure 3-1: File Menu

- Save Model...
- Save Model As...
- Import SIMM Model...
- Export SIMM Model...
- Exit

These commands are discussed in detail in Chapter 4.

3.2.2 Edit Drop-Down Menu Options

The **Edit** drop-down menu (Figure 3-2) is used to modify the following:

- Muscles
- OpenSim Files (.osim, .xml)
- Preferences...

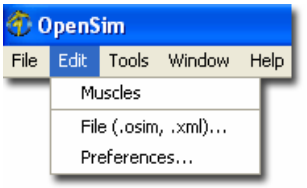


Figure 3-2: Edit Menu

See Sections 3.2.2.1 to 3.2.2.3 below for more information about each of these options.

3.2.2.1 Muscle Editor

The Muscle Editor allows you to view and modify all parameters defining muscles and other actuators in a model. See Chapter 9 for more details.

3.2.2.2 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. See Chapter 21 for more information about the model file formats. 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, click **Edit → File (.osim, .xml)...** In the window that appears, locate and select the file you wish to edit. 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. To save the edited file when you are finished, click the **Save** button in the Property Editor window (Figure 3-3).

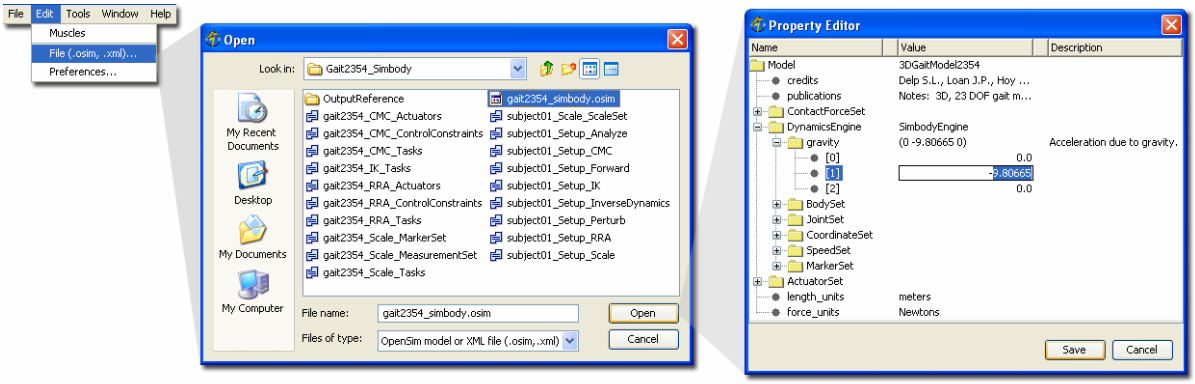


Figure 3-3: Changing the Gravity Property of a Model Using the File Editor

3.2.2.3 Preferences Editor

The Preferences Editor allows you to change user preferences for the 3D View window using the OpenSim GUI. These properties are described in detail in Section 5.6 and include the following options:

Background Color	Anti-Aliasing Frames
Marker Color	Marker Display Radius
Geometry File Path	Muscle Display Radius
Model Offset Direction	Non-current Model Opacity
Working Directory	

3.2.3 Tools

The Tools menu (Figure 3-4) enables you to easily perform the tasks needed to generate and analyze musculoskeletal simulations. The following tools are available in OpenSim:

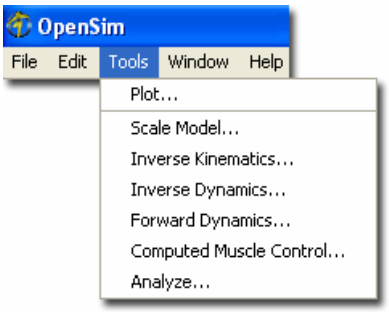


Figure 3-4: Tools Menu

- Plot... (see Chapter 12)
- Scale Model... (see Chapter 13)
- Inverse Kinematics... (see Chapter 14)
- Inverse Dynamics... (see Chapter 15)
- Forward Dynamics... (see Chapter 18)

- Computed Muscle Control... (see Chapter 17)
- Analyze... (see Chapter 19)

3.2.4 Window

The Window menu (Figure 3-5) controls what windows are displayed:

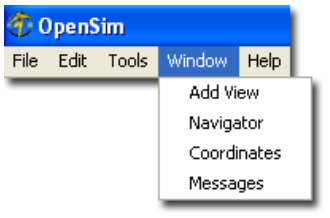


Figure 3-5: Window Menu

- Add View
- Navigator
- Coordinates
- Messages

The **Add View** command opens additional views of the model in the View window (see Section 3.3.1 and Chapter 5 for more information about the View window). Selecting **Navigator**, **Coordinates**, or **Messages** will open the corresponding window in the OpenSim GUI. More information about these windows is provided in Section 3.3.

3.2.5 Help

The Help menu (Figure 3-6) includes the following options:

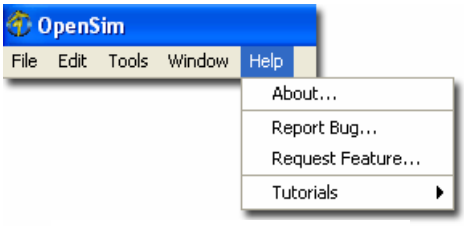


Figure 3-6: Help Menu

- About...
- Report Bug...
- Request Feature...
- Tutorials

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.

3.3 Windows

3.3.1 View Window

The View window provides 3D visualization and animation of musculoskeletal models and simulations in the OpenSim GUI (Figure 3-7). To add multiple views of the same model, go to the OpenSim main menu bar and click **Window → Add View**. The features of the 3D View window are discussed in Chapter 5.

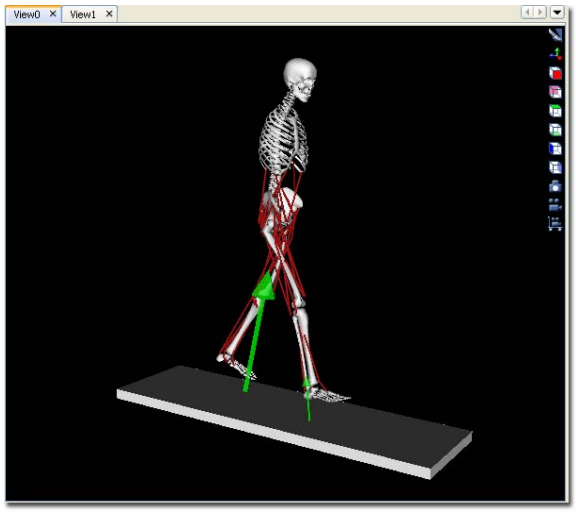


Figure 3-7: View Window

3.3.2 Navigator Window

The Navigator window provides information about a loaded model, such as bodies, actuators, joints, and associated motions, in a tree-style format (Figure 3-8). To see the Navigator window, 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. More details about the Navigator window are given in Chapter 6.

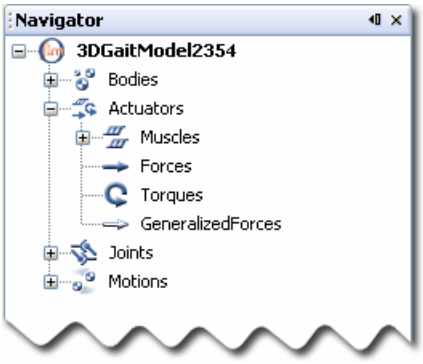


Figure 3-8: Navigator Window

3.3.3 Coordinates Window

The Coordinates window (Figure 3-9) allows a user to interactively modify the joint coordinates in the model. To see the Coordinates window, select **Window → Coordinates** from the OpenSim main menu bar. More details about the Coordinates window are given in Chapter 7.

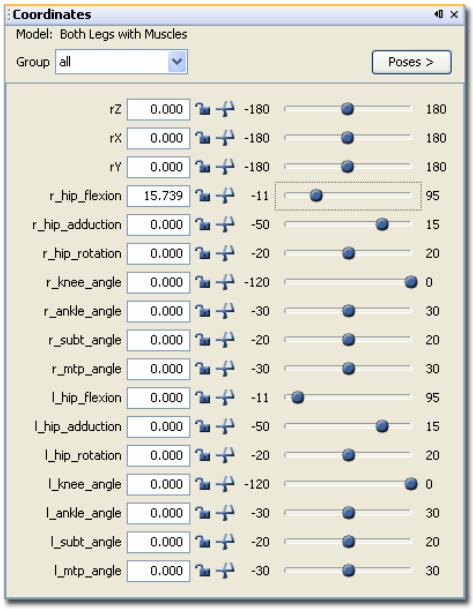


Figure 3-9: Coordinates Window

3.3.4 Messages Window

The Messages window prints details of commands and analyses performed within OpenSim. To see the Messages window, click **Window → Messages** from the OpenSim main menu bar. Figure 3-10 shows output in the Messages window from opening and scaling a model.

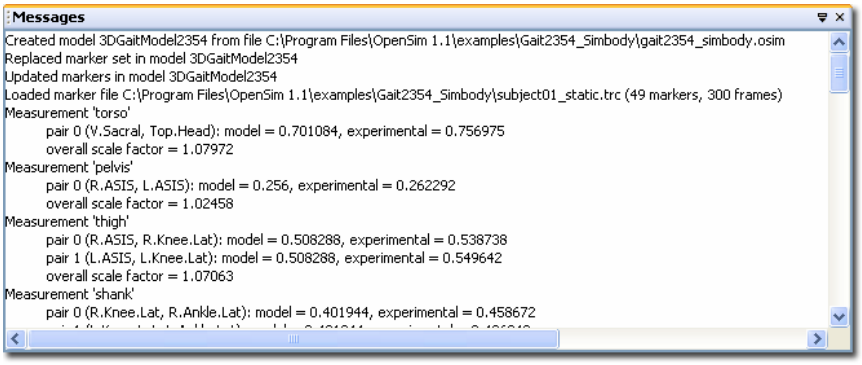


Figure 3-10: Output Printed to the Messages Window

3.4 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 (Figure 3-11).

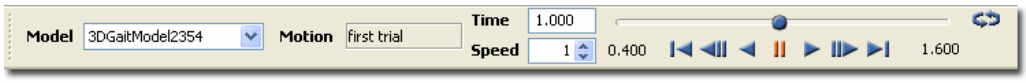


Figure 3-11: Toolbar

3.4.1 Model Drop Down Menu

The model drop down menu provides a list of all the models open in the GUI (Figure 3-12). Additionally, selecting a model from the drop down menu will make it the current model.

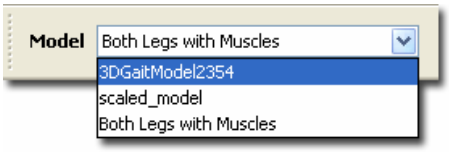


Figure 3-12: Model drop down menu

3.4.2 Motion Textbox

The motion textbox provides the name of the motion currently associated with a model (Figure 3-13).

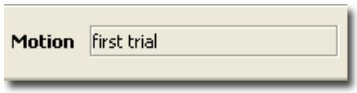


Figure 3-13: Motion textbox

3.4.3 Motion Slider / Video Controls

The motion slider and video controls are used for the animation of the model. The motion slider corresponds to the current time point in the motion file, which describes how the model moves. To animate the model, use the video control buttons (e.g., play, pause, and loop). Additionally, the value in the Speed textbox can be adjusted to change the playback speed of the animation (Figure 3-14).

Figure 3-14: Motion slider and video controls

3.5 Docking

The layout of the windows in the OpenSim GUI can be reconfigured by docking. To dock a window, 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. Figure 3-15 shows the Muscle Editor window being docked onto the Messages window.

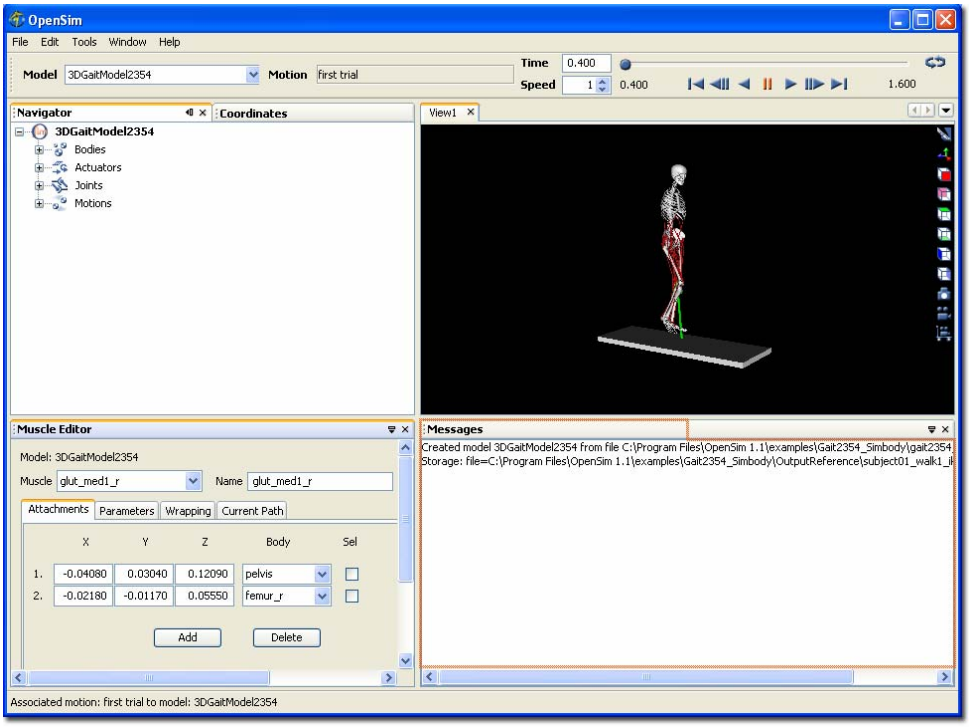


Figure 3-15: Docking the Muscle Editor Window

chapter

4

Loading and Saving Models and Motions

4.1 Overview

This chapter 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.

4.2 Opening a Model

To open a musculoskeletal model written in the OpenSim format (file type: ***.osim***), click **File → Open Model...** from the OpenSim main menu bar. In the window that appears, locate and select the desired model, and then click **Open** as shown in Figure 4-1. After loading a model, its name will appear in the model drop down menu, located in the toolbar, and in the Navigator window.

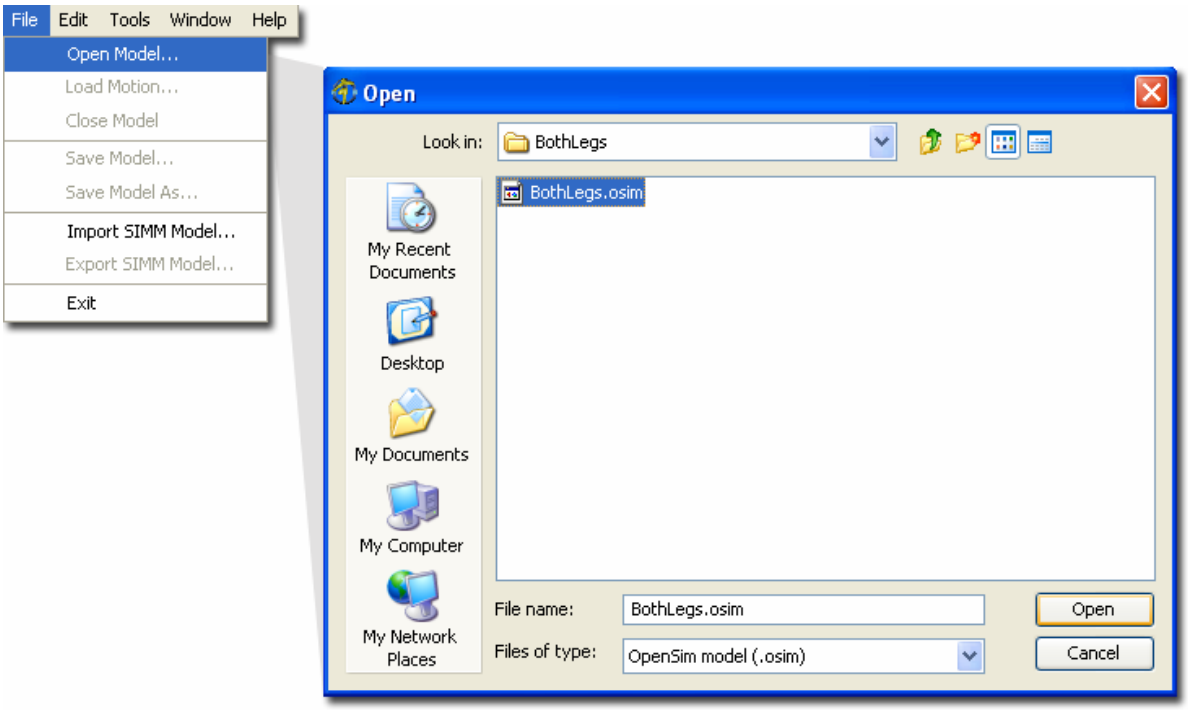


Figure 4-1: Opening an OpenSim Model

4.3 Loading a Motion

To animate a model, load an associated motion file (file type: ***.mot***) into OpenSim by selecting **File → Load Motion...** from the OpenSim main menu bar. In the window that appears, locate and select the desired motion file, and then click **Open** as shown in Figure 4-2. After loading a motion, its name will appear in the motion textbox, located on the toolbar. Additionally, a new branch will appear in the Navigator titled **Motions,** as shown in Figure 4-2. Expand the Motions branch by clicking the plus (+) icon next to it to see all motions loaded for a particular model.

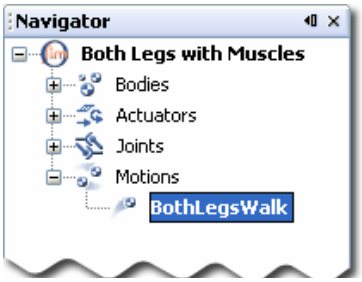


Figure 4-2: Navigator Window with a Motions Branch

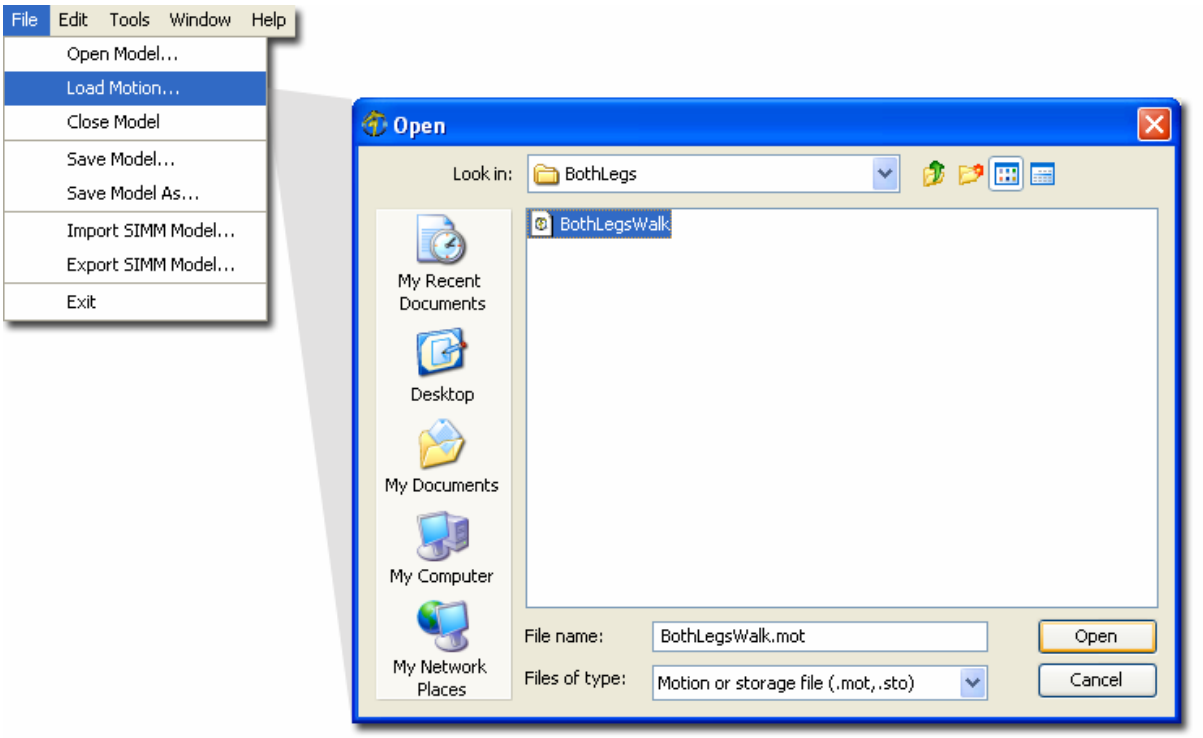


Figure 4-2: Loading a Motion

4.4 Closing a Motion

To close a motion, expand the **Motions** branch in the Navigator. **Right mouse click** on the desired motion and select **Close**, as shown in Figure 4-3.

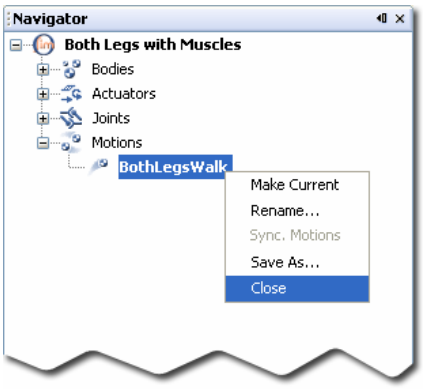


Figure 4-3: Closing a Motion

4.5 Saving a Model

To save a copy of a model, select **File → Save Model As...** from the OpenSim main menu bar. In the dialog box that appears, enter a name into the **File name** textbox, and then click **Save**, as shown in Figure 4-4. *Note: Be sure to include the file extension **.osim** when entering the file name.*

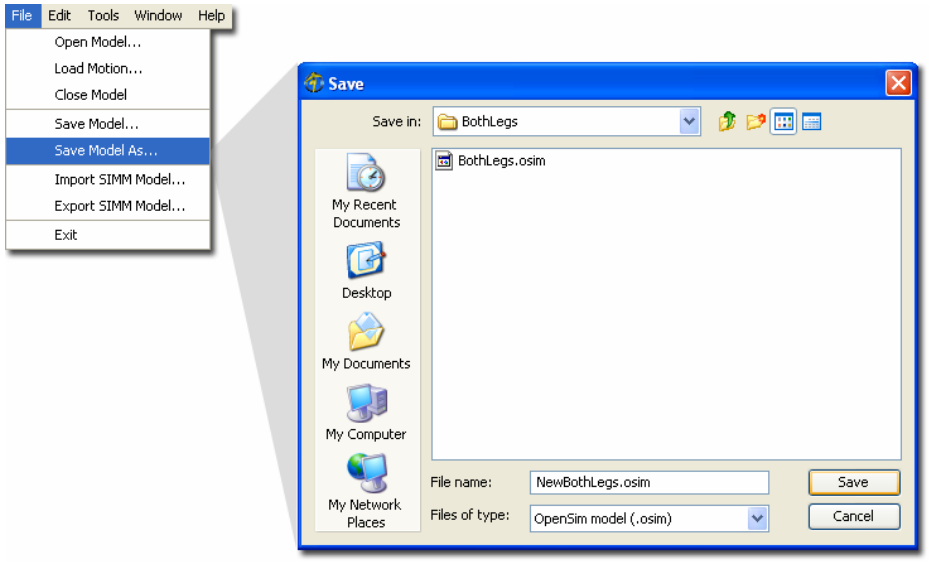


Figure 4-4: Saving a Model

4.6 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, select **File → Import SIMM Model...** from the OpenSim main menu bar. The Import SIMM Model window will appear and ask for the following information:

- The SIMM model to be imported: specify this by clicking the folder icons to select the joint file (.jnt) and muscle file (.msl).
- The new OpenSim model name: enter the name in the **OpenSim file (.osim)** textbox.

- The dynamics engine: in the **Dynamics engine** drop down menu, select either **SIMM Kinematics engine** (kinematics only) or **Simbody engine** (dynamics).
- 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.

Note: Be sure to include the extensions when entering file names.

After specifying the information in the Import SIMM Model window, click **OK,** as shown in Figure 4-5.

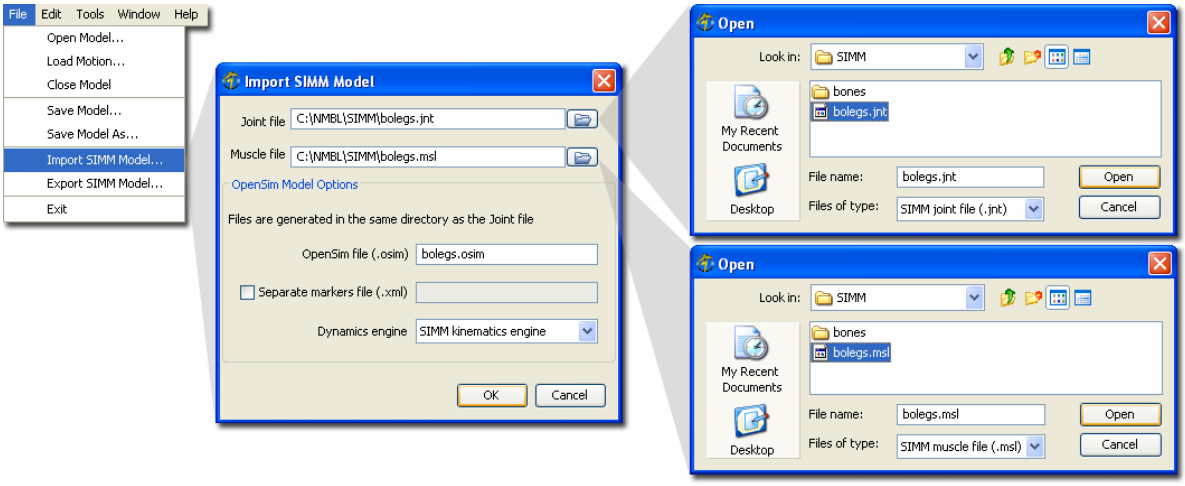


Figure 4-5: Importing a SIMM Model

4.7 Exporting a SIMM Model

To export an OpenSim model as a SIMM model, select **File → Export SIMM Model...** from the OpenSim main menu bar. 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.

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**.

The Save dialog box will close. Then click **OK** in the Export SIMM Model window, as shown in Figure 4-6.

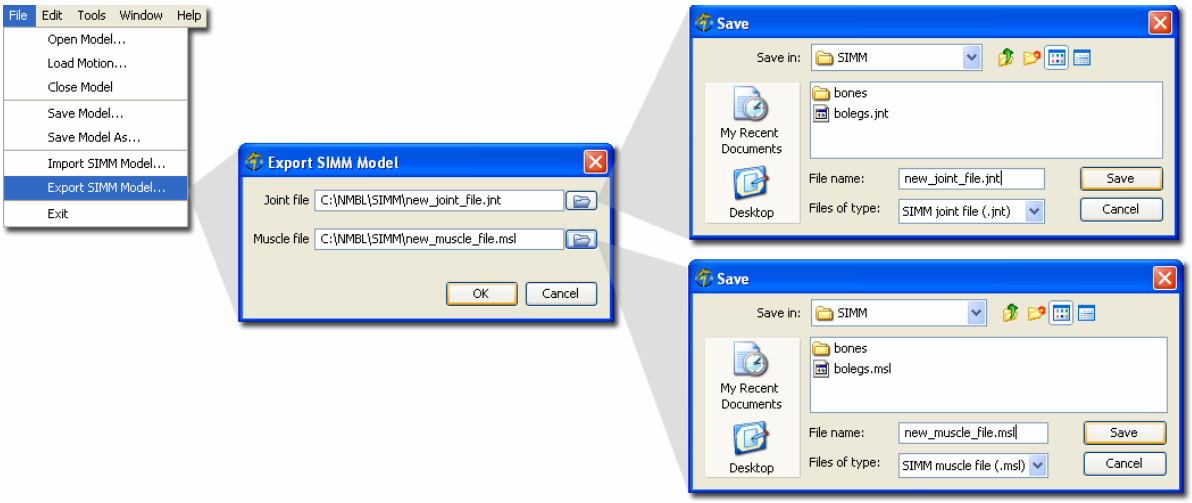


Figure 4-6: Exporting a SIMM Model

chapter

5

3D Views

5.1 Overview

The 3D View windows allow you to visualize your models within OpenSim to inspect your models' topologies or their kinematics during simulations. 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 following sections describe how to create and change parameters of the 3D View windows, as well as how to navigate within the windows.

5.2 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 (see Section 6.6.4.1).

To explicitly open a new 3D View window (without loading a model), click the **Window** menu from the main OpenSim menu bar and select the **Add View** option, as shown in Figure 5-1.

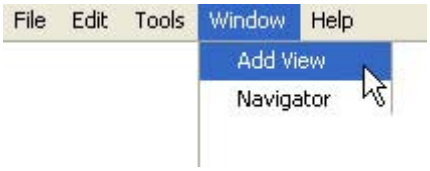


Figure 5-1: Creating a New 3D View

If you have a 3D View window already open, you can also create a new 3D View window. Go to the existing 3D View window. Right mouse click the tab label and select the **Add** option, as shown in Figure 5-2 below.

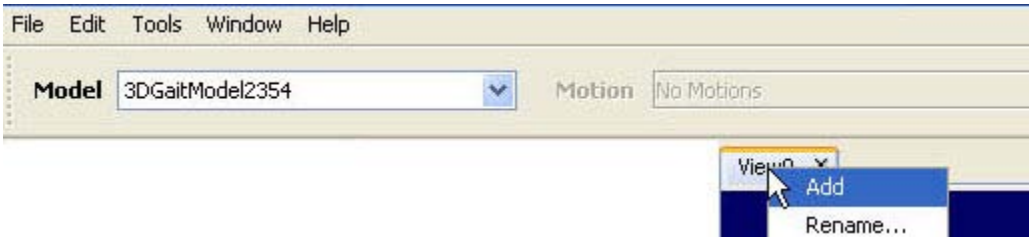


Figure 5-2: Creating a New 3D View Window from an Existing 3D View Window

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 (see Figure 5-3). 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.

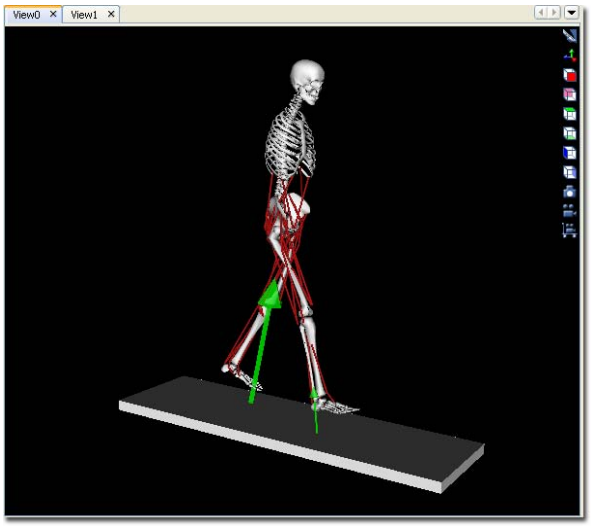


Figure 5-3: A 3D View Window with a Loaded Model. The two tabs in the upper-left indicate that there are two 3D View windows, one of which was created from the other.

5.2.1 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.

5.3 Closing 3D Views

3D View windows can be closed by clicking the "**x**" next to the tab label on the 3D View window (see Figure 5-4).

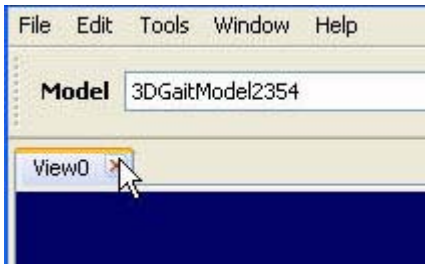


Figure 5-4: Closing 3D View Windows

5.4 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 by right mouse clicking on the tab label of the 3D View window and selecting **"Rename..."** (see Figure 5-5).

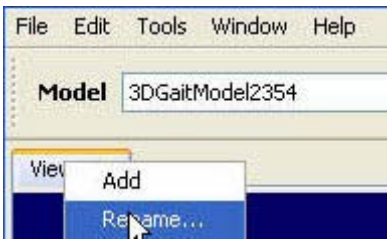


Figure 5-5: Renaming a View Window

This brings up the dialog box below (Figure 5-6), populated with the current 3D View name. Enter the new name in this dialog box and then click **OK**. Duplicate names are not allowed.

Figure 5-6: Dialog Box to Rename a 3D View Window

5.5 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, or by 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"**.

5.5.1 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 (see Section 5.6) 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 Section 8.3.2.

5.5.2 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.

5.5.3 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.

5.6 User Preferences

You can alter display preferences for the 3D View window by selecting **Edit → Preferences...** from the main OpenSim menu bar. 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** (Figure 5-7).

Details about the available preferences are given in Table 5-1. These preferences are saved and will be applied during the current session of OpenSim, as well as any future sessions of OpenSim.

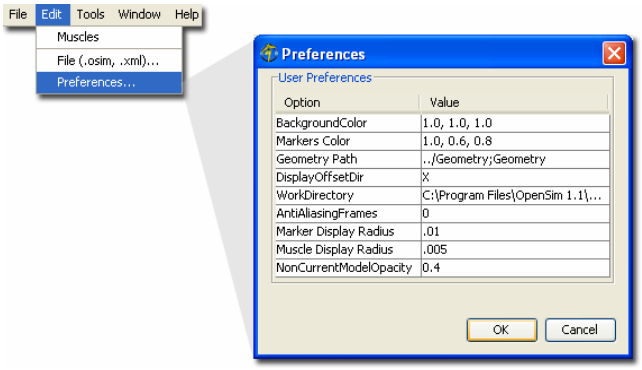


Figure 5-7: Editing User Preferences

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	Instantly, as well as to subsequently loaded models

Option Name	Default Value	Description	When Applied
		defined relative to the OpenSim installation directory.	
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

Table 5-1: 3D View Window Display Preferences

chapter

6

Navigator Window

6.1 Overview

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.

6.2 Opening and Closing

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.

To close the Navigator window, click on the "**x**" on the tab of the Navigator window. To open the Navigator window after it was closed, select **Window** → **Navigator** from the main OpenSim menu bar (Figure 6-1).

Similar to other windows opened by the OpenSim application, the Navigator window can be dragged and docked in various places on the screen.

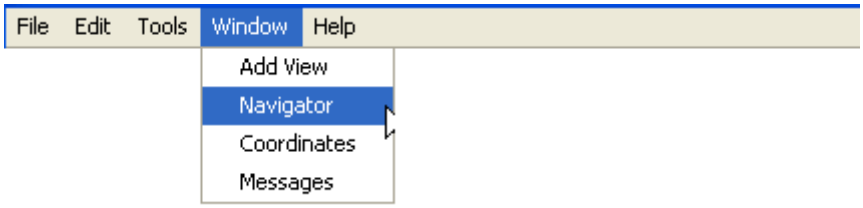


Figure 6-1: Opening Navigator Window

6.3 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. An example is shown below in Figure 6-2.

6.4 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**.

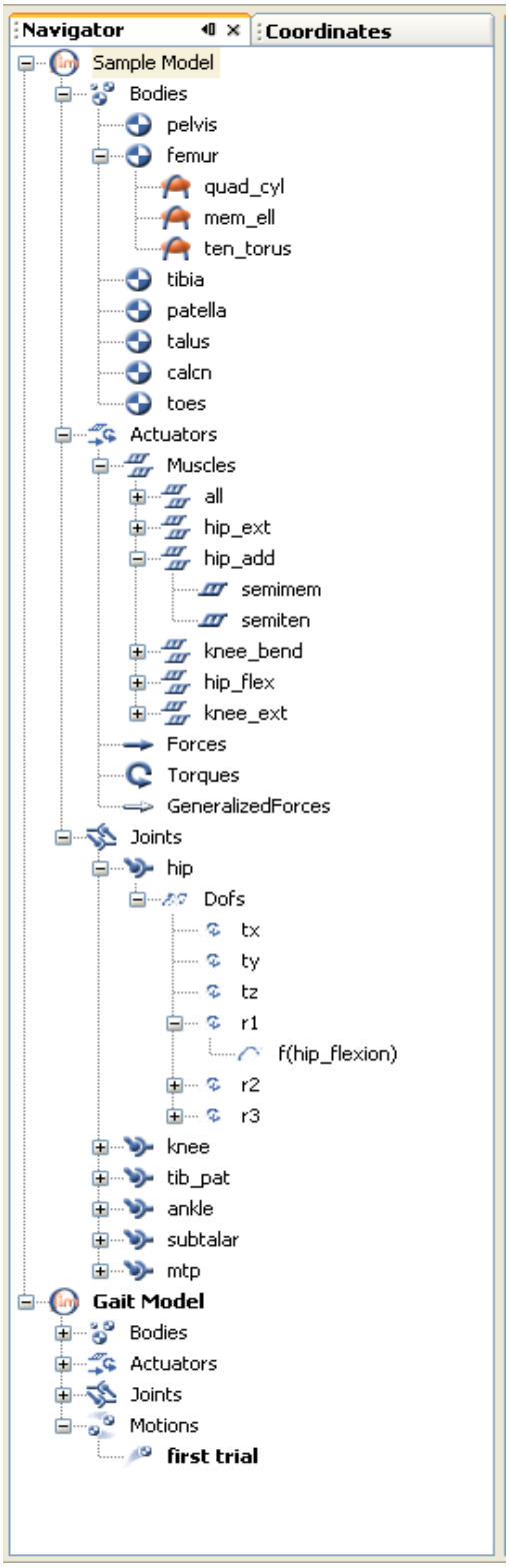


Figure 6-2: Example Navigator View with Two Models (Sample Model and Gait Model). Note: the current model is marked in bold, which is Gait Model in this case.

6.5 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. Figure 6-2 shows the navigator tree for an OpenSim model.

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
Actuators		All actuators in the model
- Muscles		All Muscles
- [Muscle Group Name]		A muscle group
- [Muscle Name]		A single muscle
- Forces		A force
- Torques		A torque
- GeneralizedForces		A Generalized 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
Motions		All motions associated with a model
- [Motion Name]		A specific motion

6.6 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.

Sections 6.6.1 to 6.6.3 describe commands common to most objects, while Section 6.6.4 details object-specific commands.

6.6.1 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 Section 6.6.4.

6.6.2 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 (see Chapter 9)
- Function Editor (see Chapter 10)
- Excitation Editor (see Chapter 11)

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

6.6.3 Property Viewer

This option brings up a window similar to the File Editor described in Section 3.2.2.2. However, it is not possible to change values in this window.

6.6.4 Object-Specific Commands

6.6.4.1 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 (see Section 3.4).

- **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 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 Section 6.6.1 above, 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.

  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 in Figure 6-3. 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.

  Note that this display offset is a display only property. The model itself is never affected by the changes to the display offset.

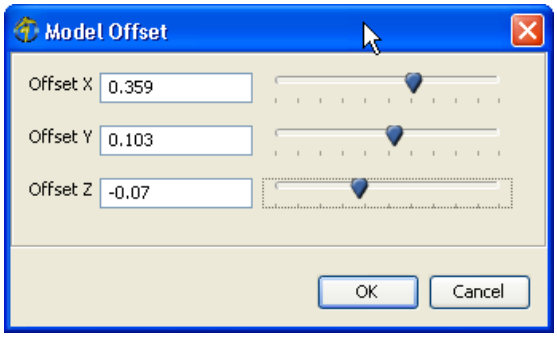


Figure 6-3: Model Offset Dialog Box

- **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).
  - *Dynamics Engine*: The engine used to solve multibody dynamics. Available options are SimbodyEngine, SimmKinematicsEngine, SDFastEngine.
  - *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 picking **File** → **Close** from 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).

6.6.4.2 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.

  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 and that they do not include spaces.

  Note that the name change is not permanent. The 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.

6.7 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.

chapter

7

Coordinates Window

7.1 Overview

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 (see Section 6.3).

7.2 Opening and Closing the Coordinates Window

When you launch OpenSim, the Coordinates window is opened in the same panel as the Navigator window. If you close it, you can open it again by selecting **Window** → **Coordinates** from the OpenSim main menu bar. 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.

7.3 Coordinate Groups

The Group drop down menu at the top of the Coordinates window allows you to select a subset of the model's coordinates for display, as shown below. Currently, the only ways to define coordinate groups are to enter them manually into the model file or to import a SIMM

model that contains gencoord groups. If no coordinate groups are defined in the model file, the drop down menu will contain just one group, named “all.”

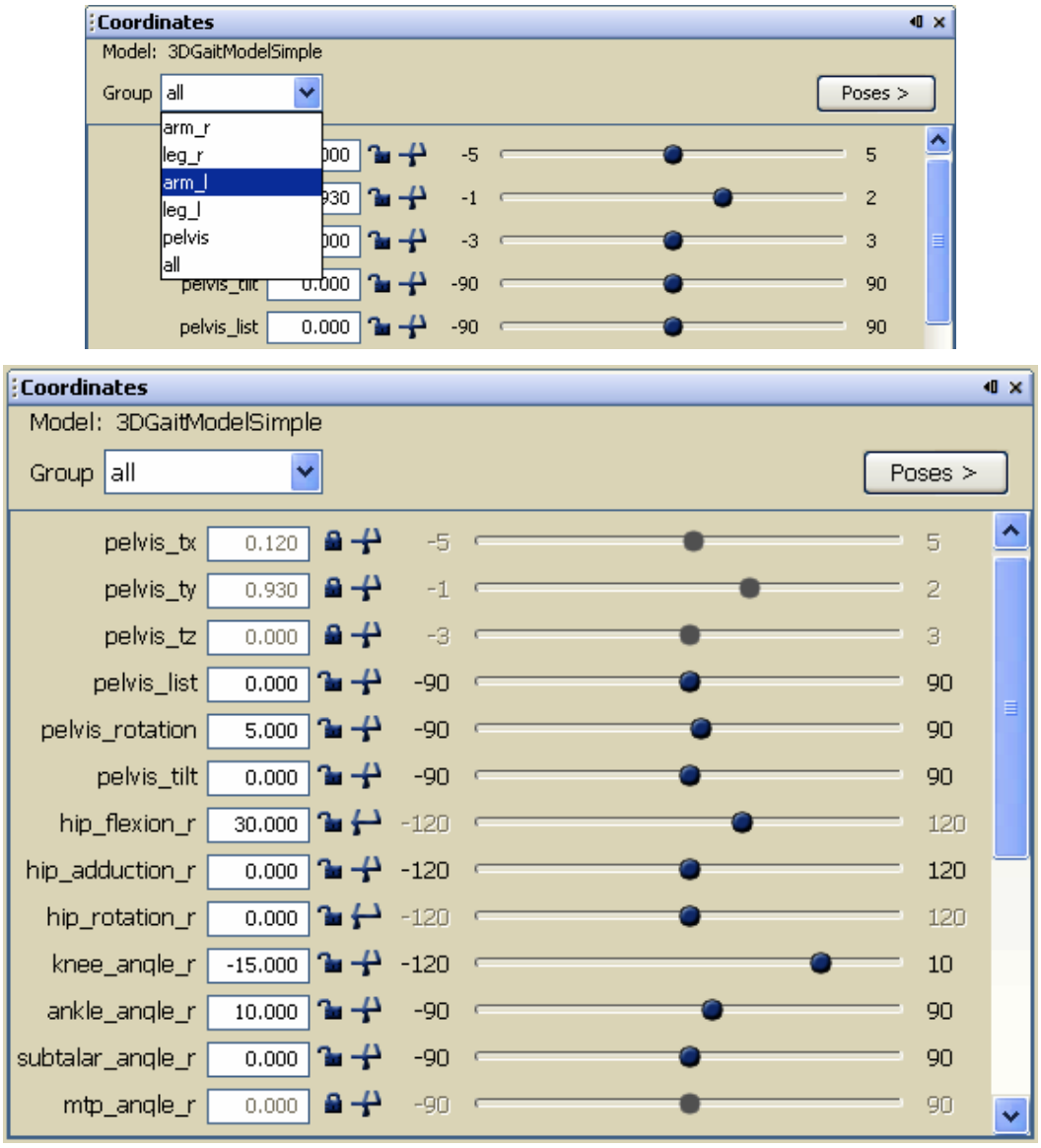


Figure 7-1: The Coordinates Window. The Coordinates window allows you to set the value of each coordinate in the current model, as well as to lock and clamp it. In this example, the pelvis_tx, pelvis_ty, pelvis_tz, and mtp_angle_r coordinates are locked, meaning that their current values cannot be changed. Thus, the number field and the slider bar are inactive. Hip_flexion_r and hip_rotation_r are unclamped, meaning that their values are allowed to exceed their ranges of motion. Thus, the numbers indicating the bounds of the range of motion are grayed out.

7.4 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 

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.

7.5 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.

7.5.1 Create a New Pose

To create a new pose, first set all of the coordinate values to put the model in the desired configuration. This can be done by typing in coordinate values, moving the sliders, or applying a particular frame of a motion (or a combination of all three). Then, click on the **Poses >** button and choose **New...** from the drop down menu, as shown in Figure 7-2. 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.

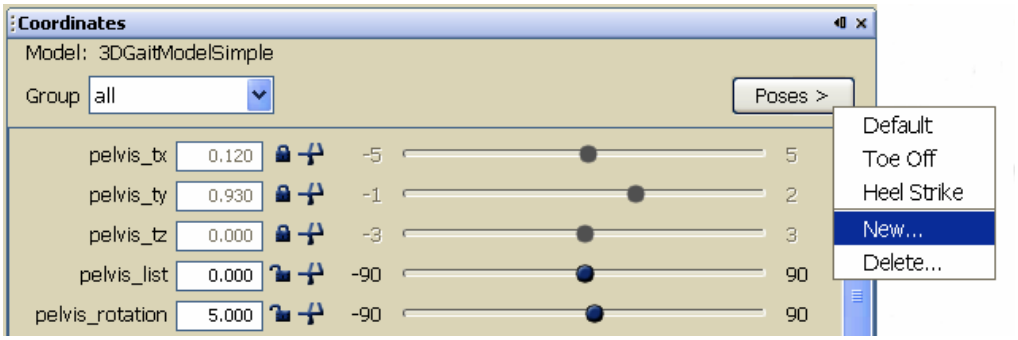


Figure 7-2: Drop Down Menu for Poses

7.5.2 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 (see Figure 7-2). "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.

7.5.3 Deleting a Pose

To delete a pose, click on the **Poses >** button and choose **Delete...** from the drop down menu. In the **Delete pose** dialog box that appears, select the pose to delete by clicking on its name, as shown in Figure 7-3. You can use **ctrl+left mouse** or **shift+left mouse** to select multiple poses. When you click **OK**, the poses will be deleted from the Poses drop down menu.

Figure 7-3: Dialog Box for Deleting a Pose

chapter

8

Snapshots and Movies

8.1 Overview

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.

8.2 Taking a Snapshot

To save the current view as an image, click the camera button  in the current 3D View Toolbar (Figure 8-1). In the Save dialog box that appears, enter a file name for the image and then click the **Save** button. The current view will be saved as a *.tiff* image.

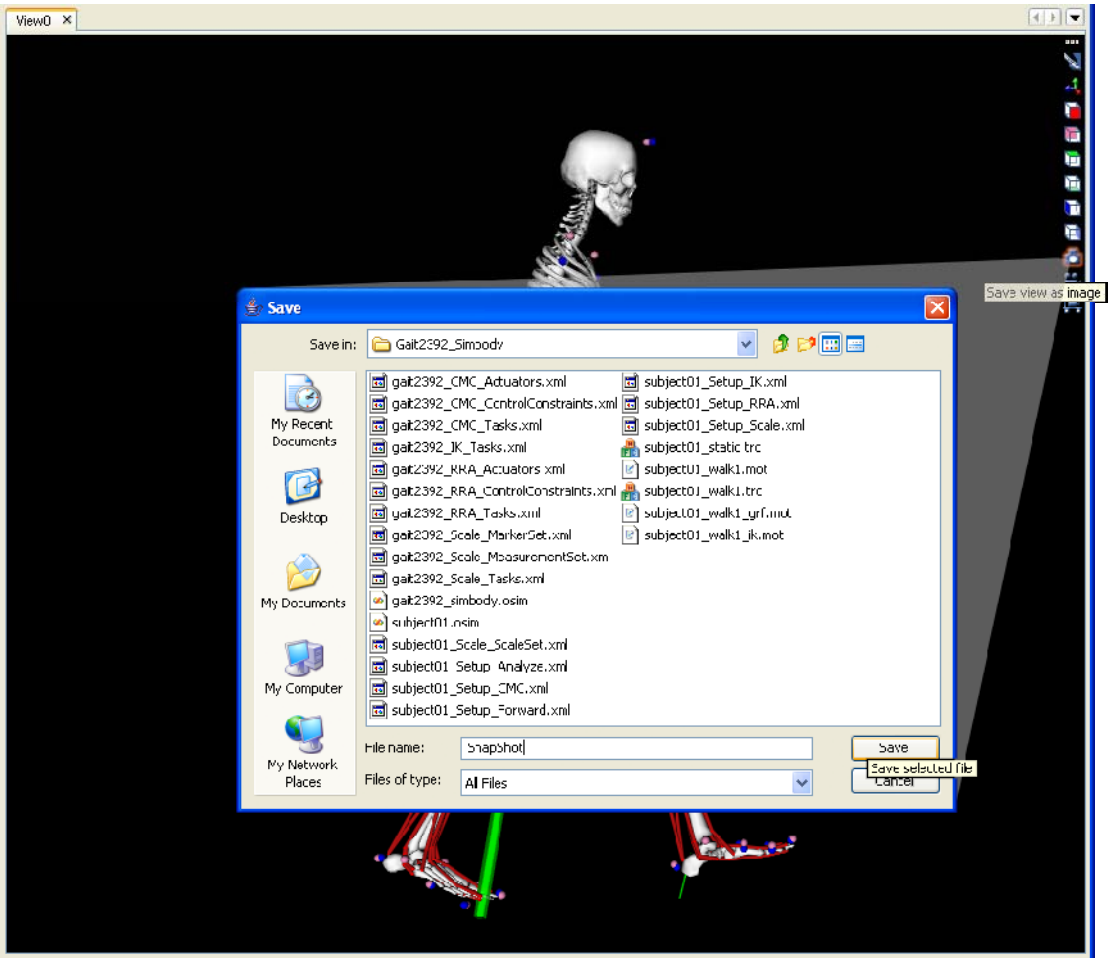


Figure 8-1: Taking a Snapshot

8.3 Making a Movie

OpenSim allows you to save the simulation playback in an AVI format movie file using the movie camera button (Section 8.3.1). 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 (Section 8.3.2). Posing the model with the coordinate viewer can also be recorded.

8.3.1 Recording a Movie

To record a movie, click the movie camera button in the 3D View Toolbar. A dialog box will appear, asking for the movie file name (Figure 8-2). Enter a name and then

click **Open,** which will cause recording to be toggled "on." 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. To stop recording and write out the movie file, press the movie camera button again.

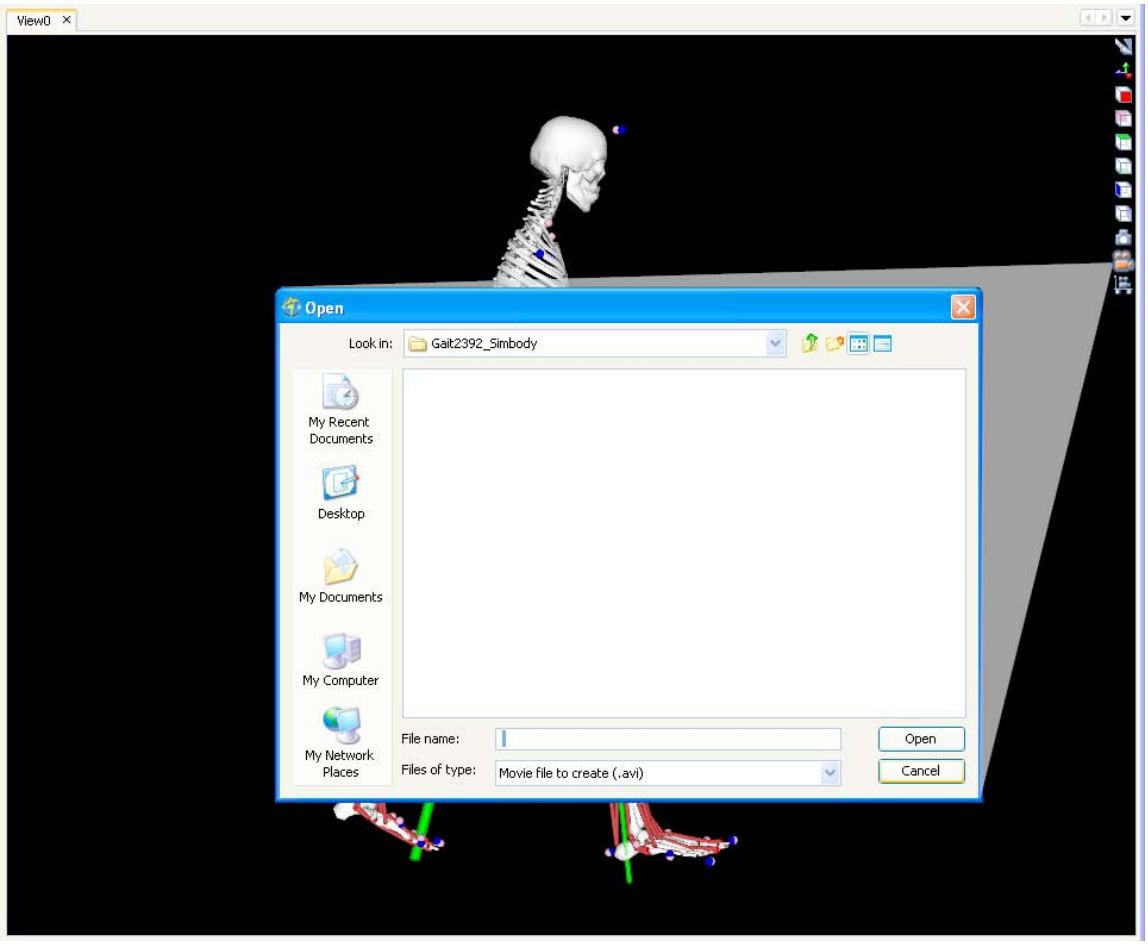


Figure 8-2: Recording Motion as a Movie

8.3.2 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.

8.3.2.1 Create/Edit a Pre-defined Camera

To create or edit a camera path, click on the camera dolly button  in the 3D View window and select the **Edit Cameras...** option from the drop down menu that appears (Figure 8-3).

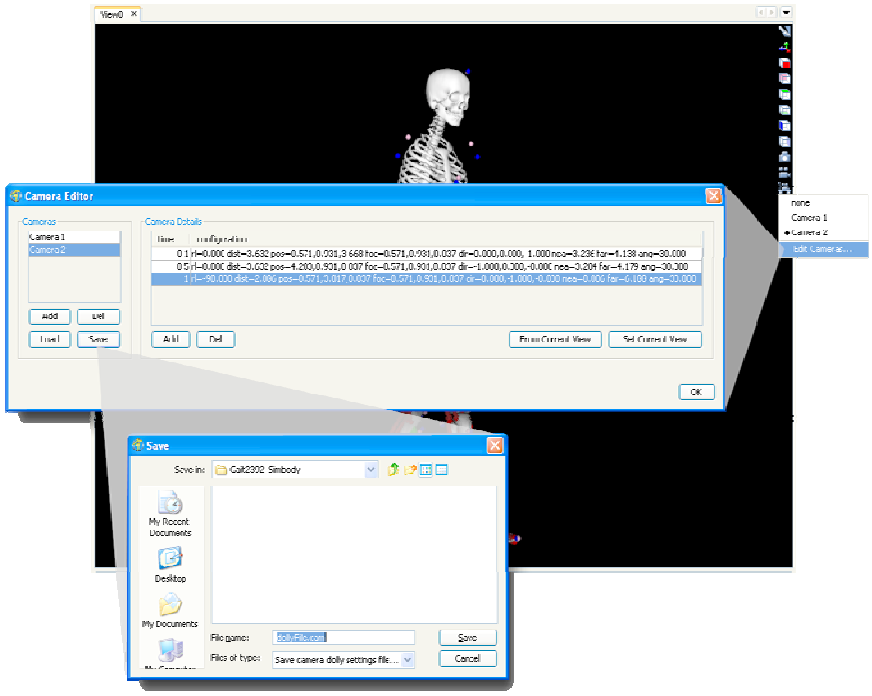


Figure 8-3: Camera Dolly Editing and Saving

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 (see Chapter 5 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°), 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.

8.3.2.2 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.

chapter

9

Muscle Editor

9.1 Overview

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.

9.2 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 (see Section 3.4.1), the Muscle Editor automatically switches to operate on this model, and displays the first muscle in the model.

9.3 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.

9.4 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 (see Section 3.4.1).

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 **Actuators**, 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. 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. The selected muscle is shown in the box. To switch to a different muscle, left click on the box and choose another muscle.

9.5 Panels

The properties of a muscle are organized into panels within the Muscle Editor window (Figure 9-1). The *Attachments* panel (Section 9.5.1) shows all of the attachment points (fixed, via, and moving) that define the muscle path.

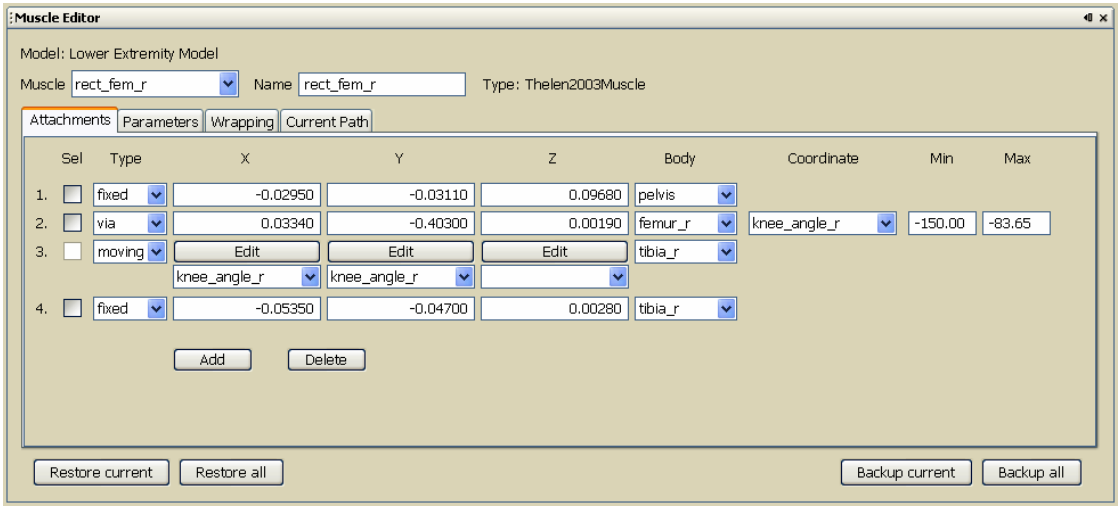


Figure 9-1: The Muscle Editor window. The properties of a muscle are organized into panels. The *Attachments* panel, shown in this figure, displays the fixed, via, and moving muscle points that define the muscle path.

The *Parameters* panel (Section 9.5.2) 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 (Section 9.5.3) 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 (Section 9.5.4) displays the list of wrap objects that are currently associated with the muscle. The *Current Path* panel (Section 9.5.5) shows the list of attachment points that currently define the muscle path.

9.5.1 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.

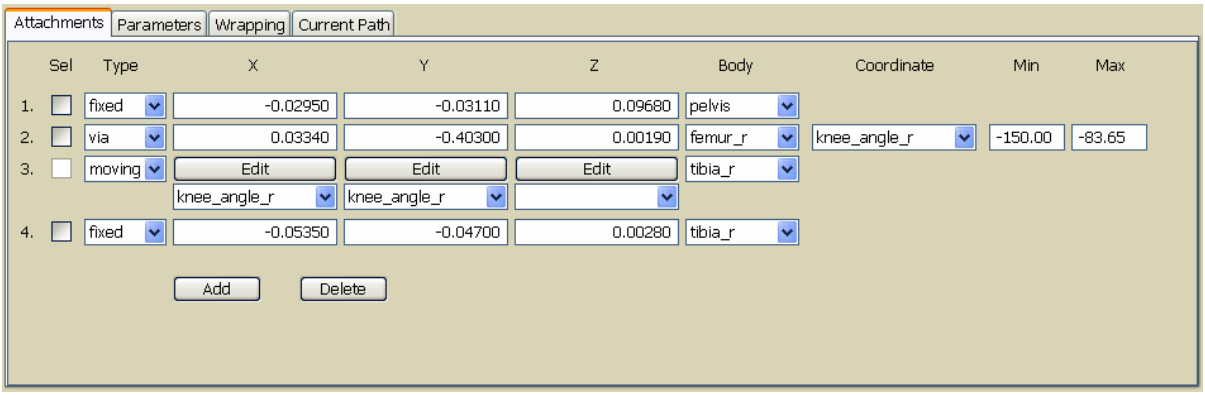


Figure 9-2: The Attachments Panel. This panel shows the fixed, via, and moving muscle points that define the muscle path. In this example, the path is defined by four attachment points. The first point is fixed to the pelvis and the second point is a via point on the femur that is active only when the 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.

9.5.2 Parameters

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.

9.5.3 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 (see Chapter 10). Muscles of type Thelen2003Muscle do not have any parameters that are functions, so the *Functions* panel is not shown.

9.5.4 Wrapping

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 in Section 9.5.4.1 below.

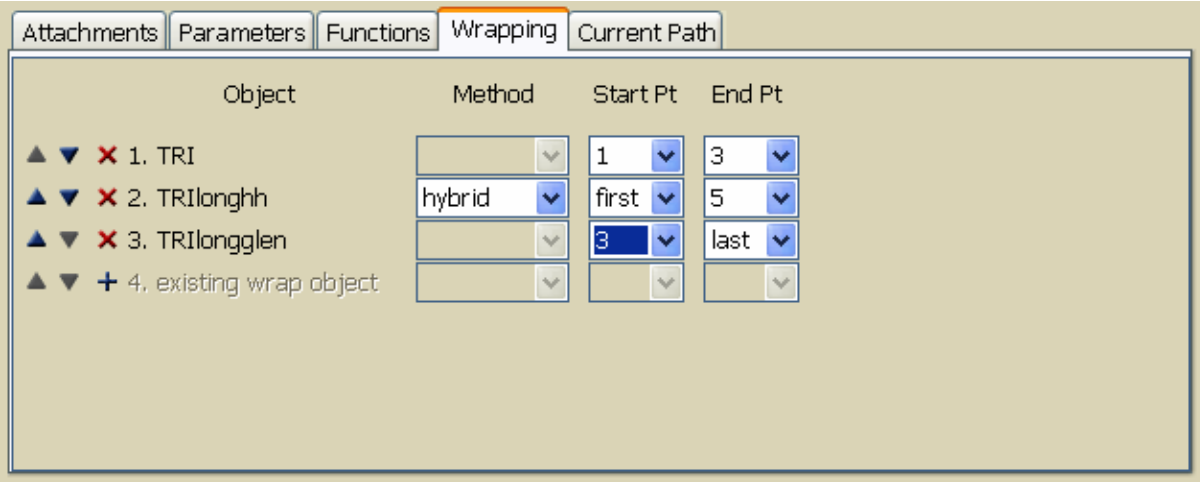


Figure 9-3: The Wrapping Panel. This panel shows all of the wrap objects that are associated with the muscle. You can add to, delete from, or reorder the list, as well as restrict the wrapping of each object to a subset of the muscle path. For ellipsoid wrap objects, you can also specify the algorithm 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.

9.5.4.1 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.

9.5.4.1.1 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.

9.5.4.1.2 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.

9.5.4.1.3 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.

9.5.5 Current Path

The *Current Path* panel (Figure 9-4) 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.

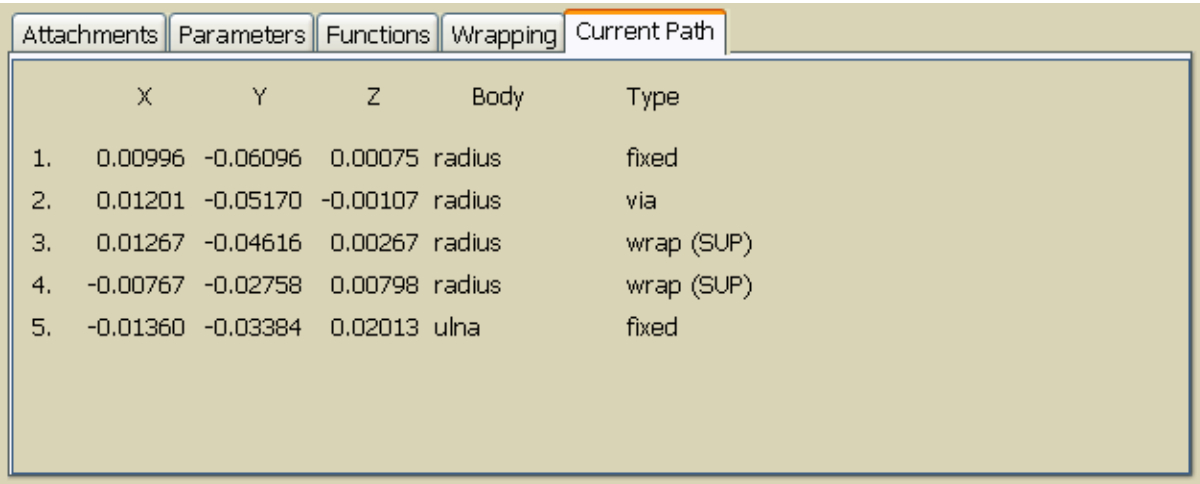


Figure 9-4: Current Path Panel. This panel shows the list of attachment points that currently define the muscle path.

9.6 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.

9.6.1 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 (see Section 9.4), 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.

9.6.2 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 (see Section 5.5.3). 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.

9.6.3 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.

9.7 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.

9.8 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.

chapter

10

Function Editor

10.1 Overview

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).

10.2 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.

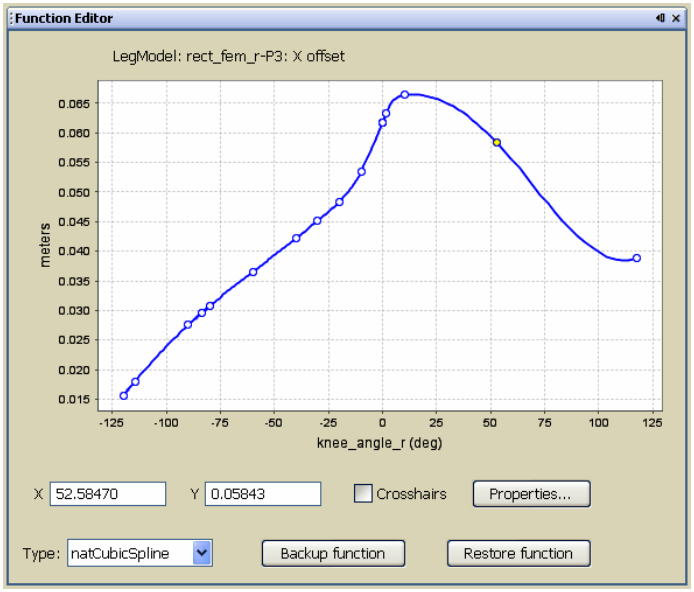


Figure 10-1: The Function Editor

10.2.1 Opening the Editor for a Joint Constraint Function

To edit a joint constraint function associated with a degree of freedom (DOF), locate the DOF in the Navigator window. Click on the **plus (+)** sign next to the name of the current model to display the model components. Click on the **plus (+)** sign next to **Joints**, and then click on the **plus (+)** sign next to the name of the appropriate joint. Then, click on the **plus (+)** sign next to **Dofs**.

Right-click on the name of the coordinate and choose **Edit...** from the drop down menu, as shown in Figure 10-2. OpenSim will open the Function Editor window and load the function so you can view and modify it.

Note: In version 1.5 of OpenSim, DOF functions can be edited only for models that use a SimmKinematicsEngine. Models using SdfastEngine or SimbodyEngine do not currently have editable joints.

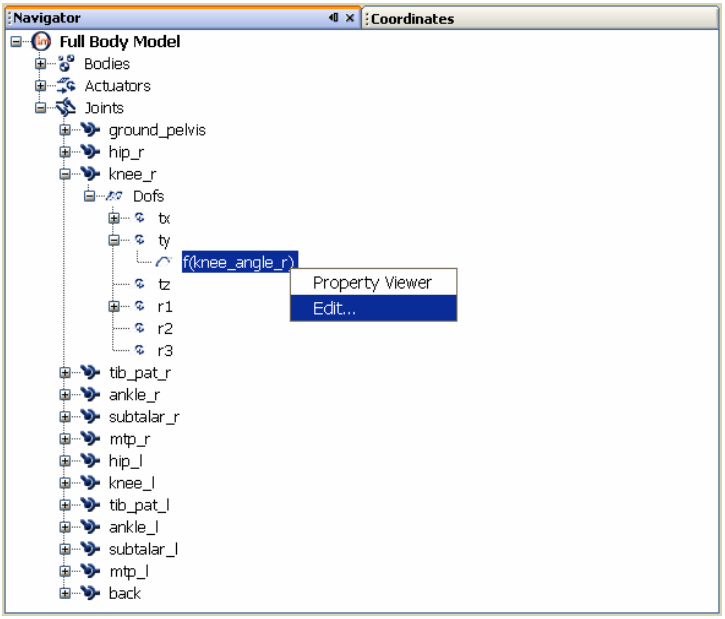


Figure 10-2: Selecting a DOF Function. In this example, the ty DOF in the knee_r joint is a function of the coordinate knee_angle_r. Right-clicking on the coordinate name brings up a drop down menu. Choosing Edit... allows you to load the function into the Function Editor.

10.2.2 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 (see Section 9.4 for details). 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.

10.2.3 Opening the Function 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 (see Section 9.4 for details). 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.

10.3 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).

10.4 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.

- **LinearFunction** – A linear function (also called piecewise linear) 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).

10.5 Zooming 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** (Figure 10-3) 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.

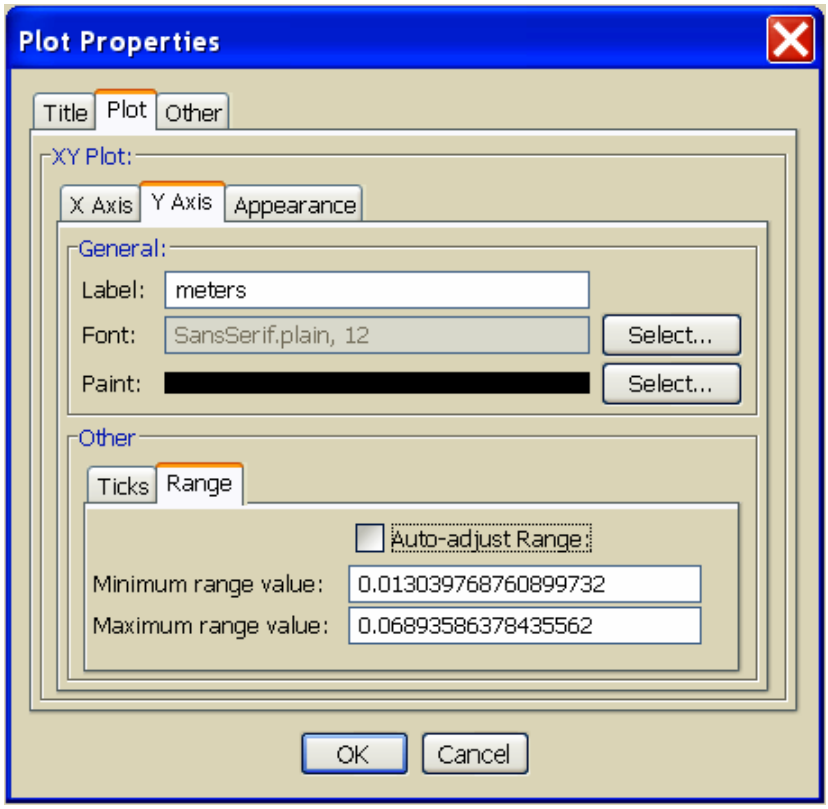


Figure 10-3: Plot Properties Dialog Box for the Function Editor. The Plot Properties dialog box gives you access to many properties for plotting the function. Shown here are the minimum and maximum values for the Y axis. To make the Function Editor auto-scale the Y axis to display all of the control 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.

10.6 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.

chapter

11

Excitation Editor

11.1 Overview

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 (Chapter 18) or when examining the outputs of a control algorithm that solves for muscle excitations, for example, the Computed Muscle Control (Chapter 17). 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.

11.2 Opening the Excitation Editor Window

To bring up the Excitation Editor (Figure 11-1), 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.

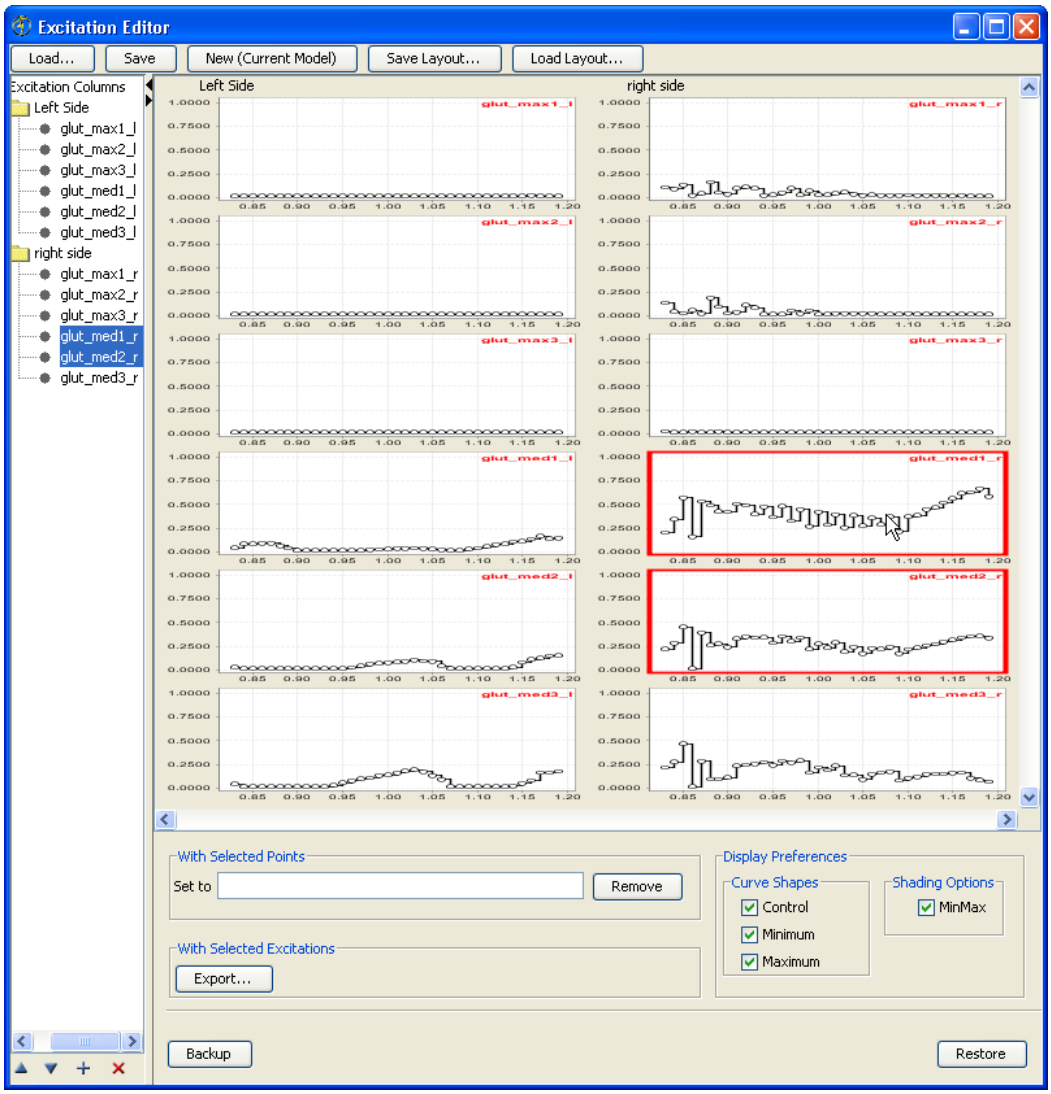


Figure 11-1: Excitation Editor. The Excitation Editor allows the display and editing of muscle excitations and other control waveforms. In this example, the muscle excitations are organized into two columns, named “left side” and “right side.” Two of the muscle excitations (glut_med1_r and glut_med2_r) are currently selected. They are highlighted in the *excitation tree* along the left-hand side of the Excitation Editor. Their plots also appear in the *excitation grid* panel with a red border.

11.3 Excitation Editor Layout

The Excitation Editor window is made up of the following areas:

- Command panel at the top of the window
- Excitation tree along the left-hand side of the window
- Excitation grid panel along the right-hand side of the window
- Control panel at the bottom-right of the window

The functions of these GUI items are explained in the following sections.

11.3.1 The Command Panel

The *command* panel in 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 (see Section 12.11) appears, 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**...).

11.3.2 Excitation Tree and Excitation Grid Panel

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 Section 11.3.2.1).

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. Figure 11-1 shows an example with two columns, one for the left glut muscles and one for the right glut muscles.

11.3.2.1 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 Section 12.11) 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 11-1).

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.

11.3.2.2 Excitation Panel

Each entry of the *Excitation Grid* is an *excitation panel* (Figure 11-2). 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, move the cursor to the desired location for the new point. Then, **right-click** to bring up a drop down (context) menu. Select **Add Control Point to Max** to add a point for the max curve; **Add Control Point to Min** to add a point for the min curve; **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.

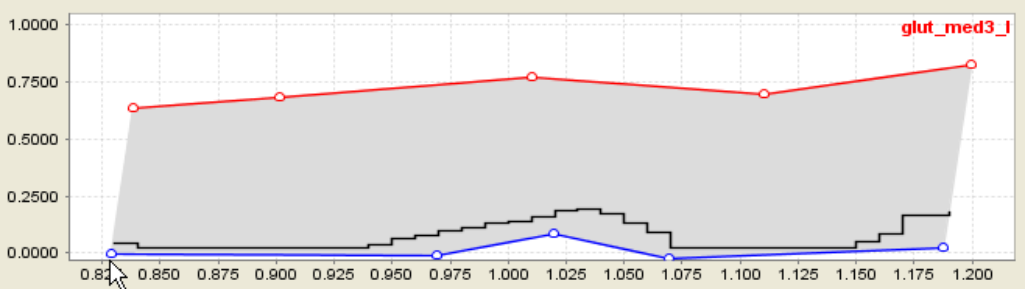


Figure 11-2: Excitation Panel. *Excitation panels*, such as the one in this figure, are organized in columns in the *Excitation Grid*. Each *excitation panel* shows a control curve (black). It may also show a maximum curve (red) and a minimum curve (blue). The control points for these curves appear as 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.

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.

11.3.3 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.

11.3.3.1 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 Section 10.3 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 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 yellow. 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.

11.3.3.2 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 in Figure 11-1.

11.3.3.3 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 then type in the new value in the text box labeled **Set to**.

You can also remove all selected points in all panels. Select the points and then press the **Remove...** button in the sub-panel labeled **With Selected Points**.

11.3.3.4 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.

11.3.4 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 (Figure 11-2). 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.

11.4 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.

Note: 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.

chapter

12

Plotting

12.1 Overview

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.

12.2 Opening and Closing the Plotter Window

To open the Plotter window, 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 in Figure 12-1. Multiple instances of the Plotter window can be opened simultaneously.

To close the window, select the standard **"x"** in the upper-right corner of the Plotter window.

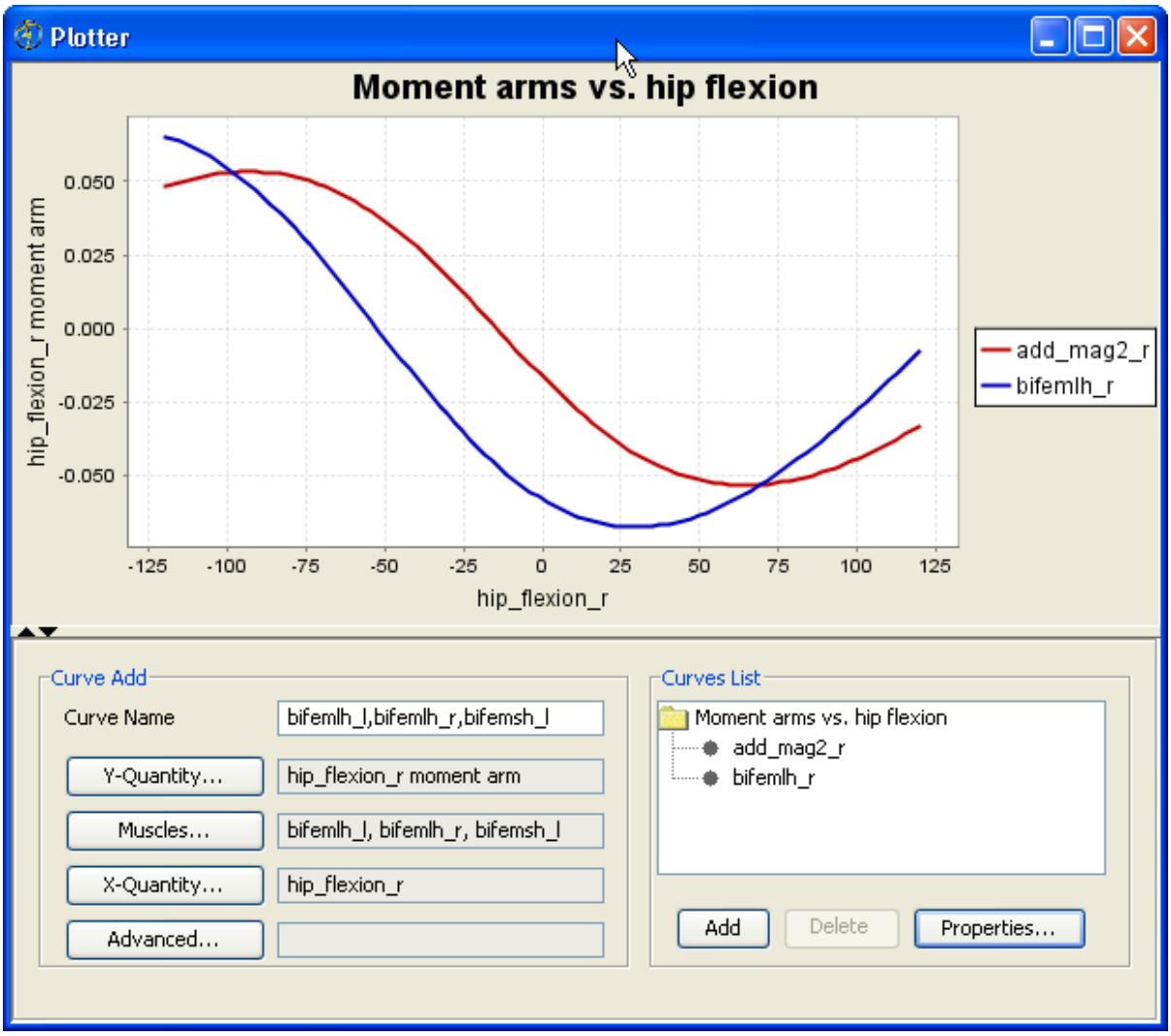


Figure 12-1: Plotter Window

12.3 The Plotter Window Layout

The Plotter window is made up of three panels as shown in Figure 12-1.

- The *plot* panel (Figure 12-2) at the top of the Plotter window displays curves and shows the plot title, x-label, x-axis, y-label, y-axis and legend.

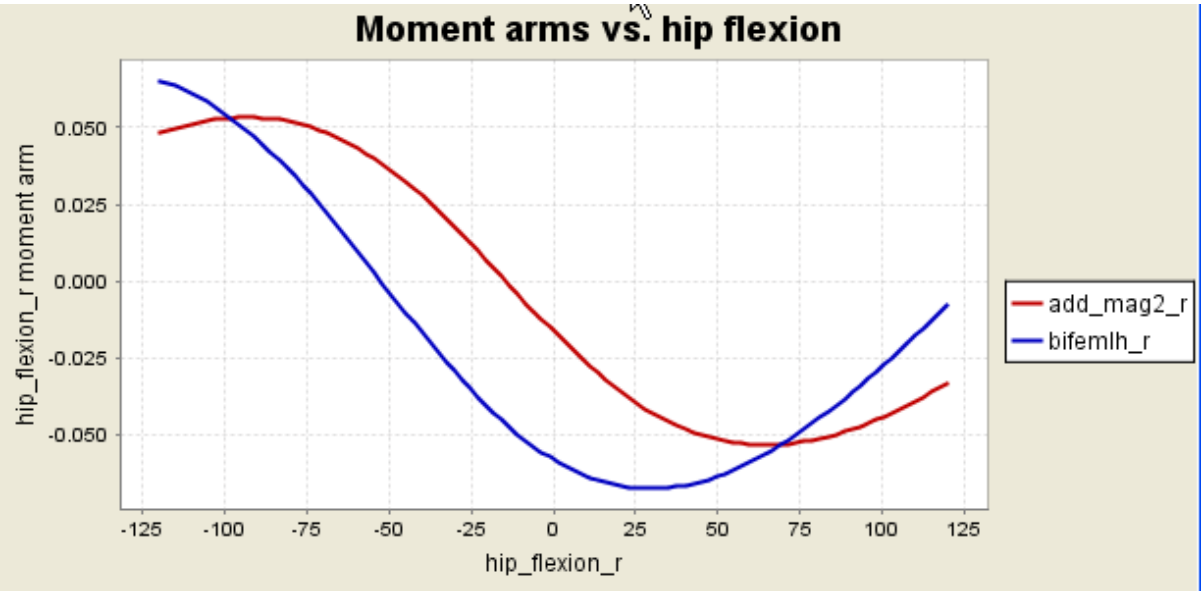


Figure 12-2: Plot Panel

- The *curve creation* panel (Figure 12-3) 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 Section 12.6 below.

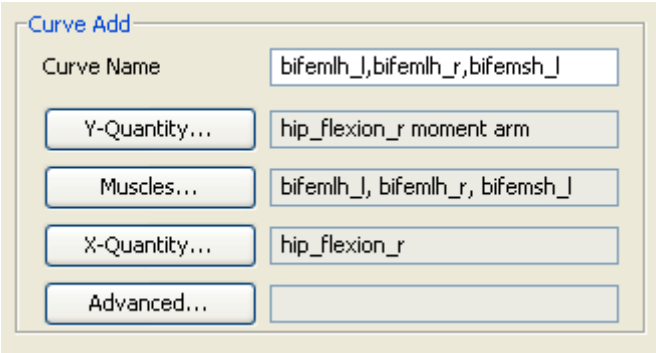


Figure 12-3: Curve Creation Panel

- 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.

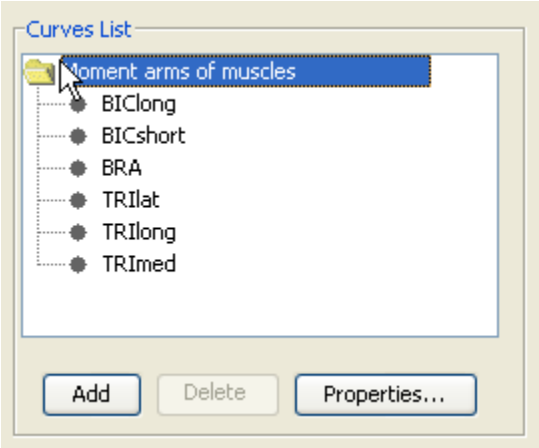


Figure 12-4: Plot Summary Panel

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.

12.4 Plot Panel Controls

The *plot* panel is the top part of the Plotter window and includes the drawing area, title, legend, and axis labels. 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 Figure 12-5. The options available through the context menu are discussed in the following sections.

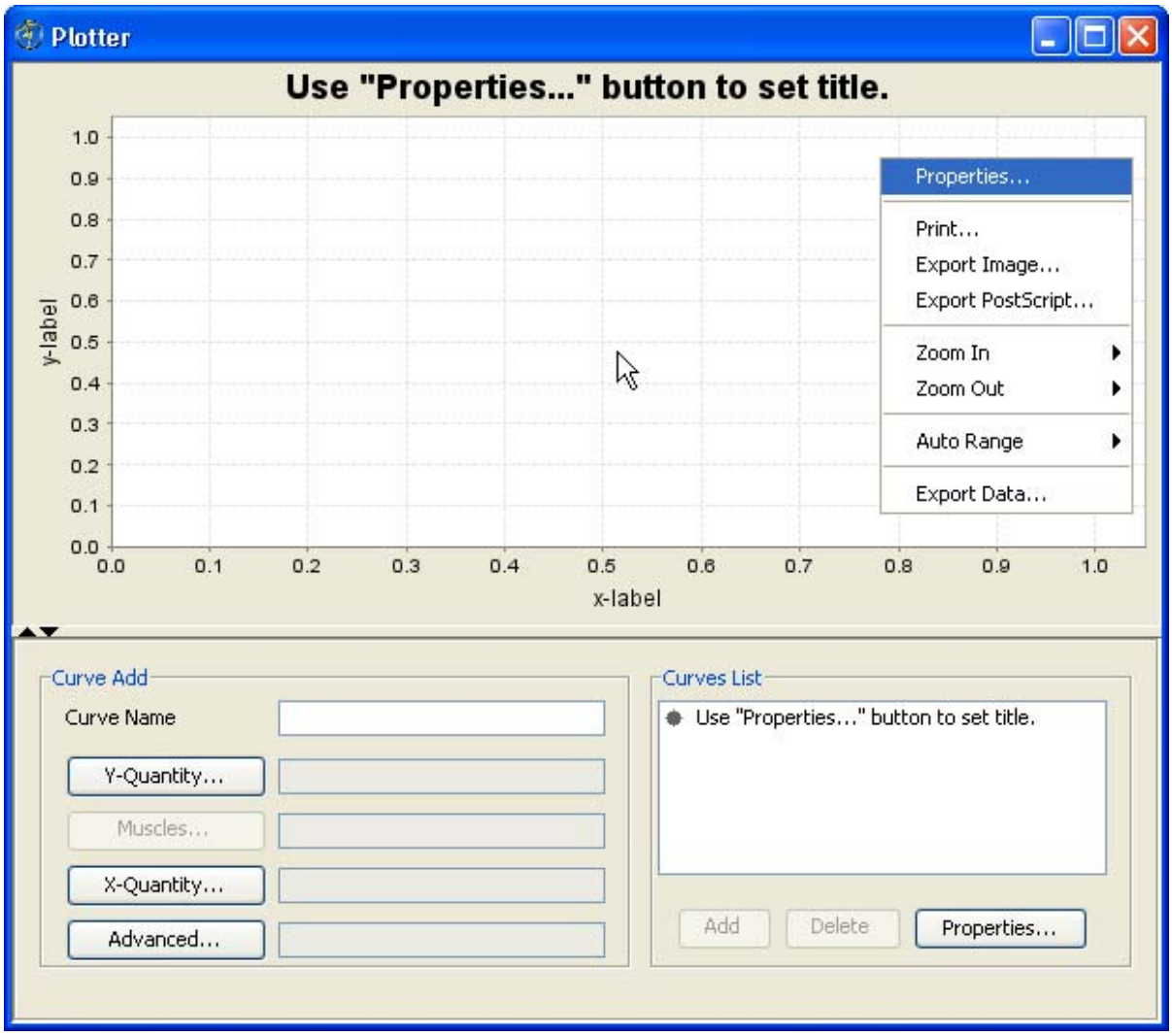


Figure 12-5: Plotter Context Menu

12.5 Plot 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 (see Section 12.6) 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 (Figure 12-6) will appear. This is the same window used in the Function Editor and described in Section 10.5.

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.

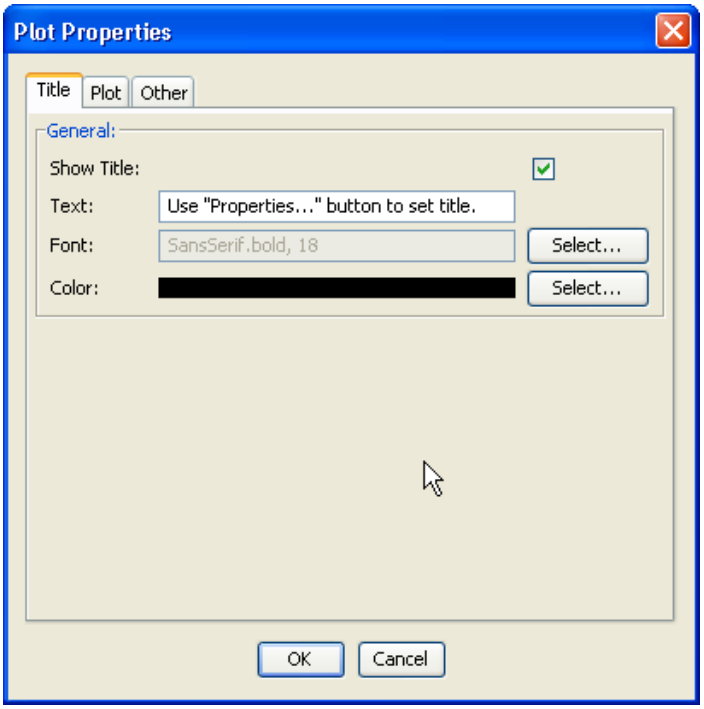


Figure 12-6: Plot Properties Window. This window is used to customize the *plot* panel.

In addition, there are drop down (context) menus associated with each curve (these appear in the *plot summary* panel next to a gray dot) and with each plot (these appear in the *plot summary* panel next to a folder icon). These menus are described in Sections 12.5.1 and 12.5.2 below.

12.5.1 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:

- Properties – see Section 10.5
- Export Data – see Section 12.7

12.5.2 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 shown in Figure 12-7, 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.

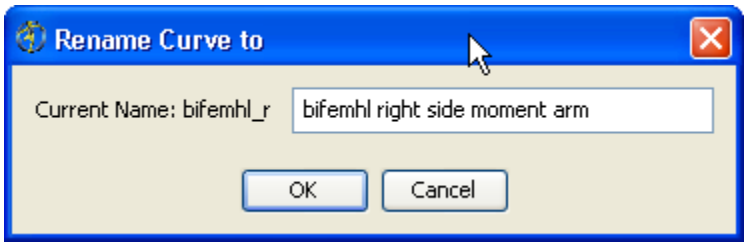


Figure 12-7: Curve Rename Dialog Box

- **Info** – Displays a dialog box, as shown in Figure 12-8, 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.

Figure 12-8: Curve Info Dialog Box

12.6 Curve Creation Panel

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 12-9). The options will vary, depending upon the data source (i.e., from an OpenSim analysis or an external source).

   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.

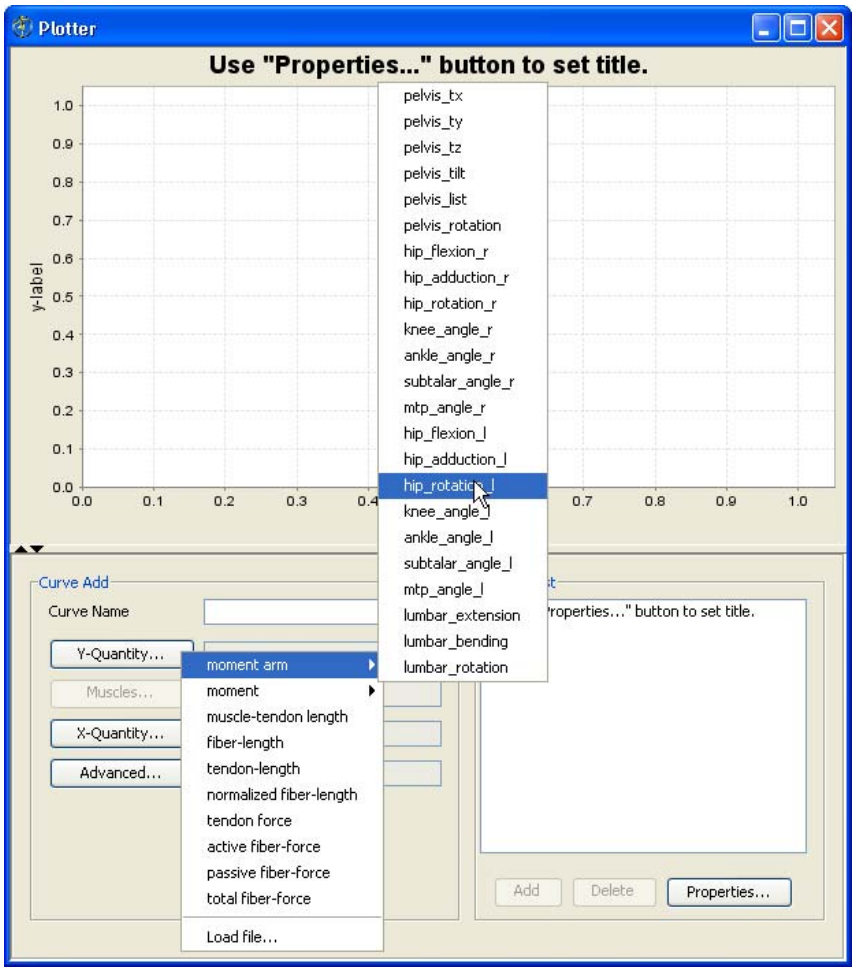


Figure 12-9: Y-Quantity Menu

2. 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.

   Pressing the **Muscles...** button brings up the filtering dialog box used to select muscles (see Section 12.11). 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.

3. 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.

4. 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 (see Section 12.5.2). This name will be used to identify the curve in the *plot summary* panel, as well as in the plot legend.

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

12.6.1 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.

12.6.2 Motion Curves

Instead of varying a generalized coordinate to compute the curves in Section 12.6.1, you can use a motion file to change the model configuration and compute the same quantities as described in Section 12.6.1 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.

12.6.3 External Curves

You can also plot data generated by other tools in OpenSim. For example, running the forward tool (Chapter 19) 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.

12.7 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.

12.8 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 Section 12.9), since advanced editing tools (e.g., Adobe Photoshop) can be used to modify the resulting files (e.g., changing line styles, labels and annotations).

12.9 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**.

12.10 Printing

You can print the plot to a printer by clicking the **right mouse** button in the *plot* panel and selecting **Print...** from the drop down menu that appears. The **Page Setup** dialog box shown in Figure 12-10 will appear. Use this to pick printing options (e.g., paper size, printing orientation, print margins). Then, press **OK**.

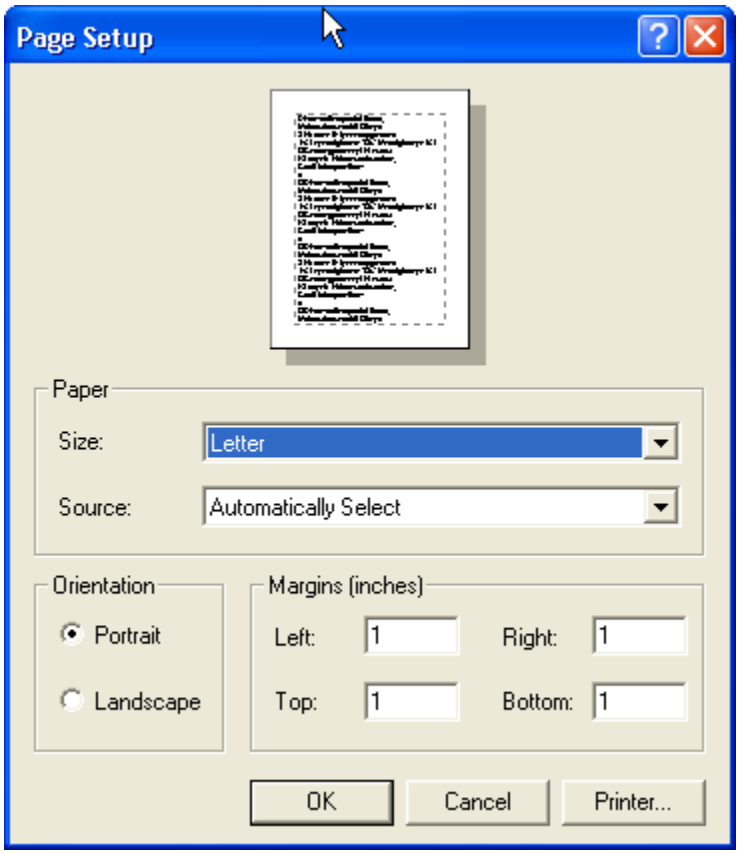


Figure 12-10: Printing from the Plotter window

12.11 Selection Filtering Window

When you press the **Muscles...** button from the *curve creation* panel, the dialog window shown below in Figure 12-11 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.

12.11.1 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

12.11.2 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.

12.11.3 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.

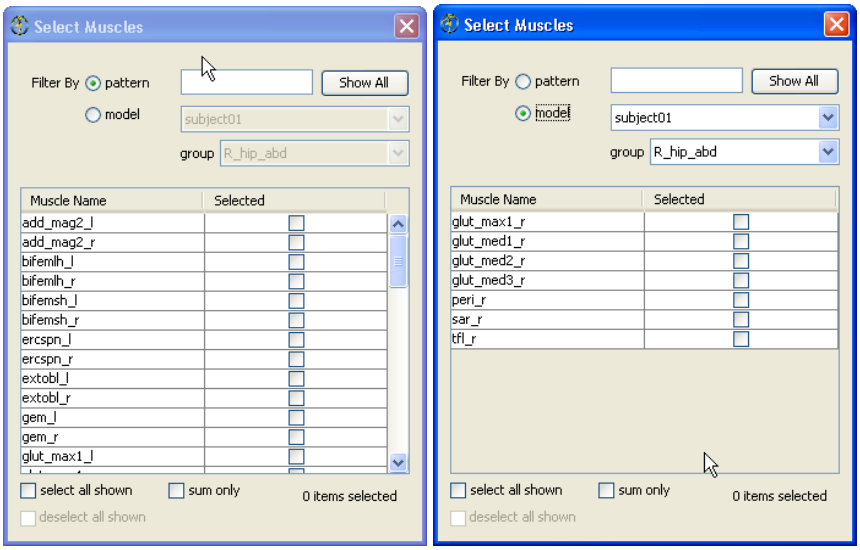


Figure 12-11: Filtering Options

12.12 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 (x) values

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

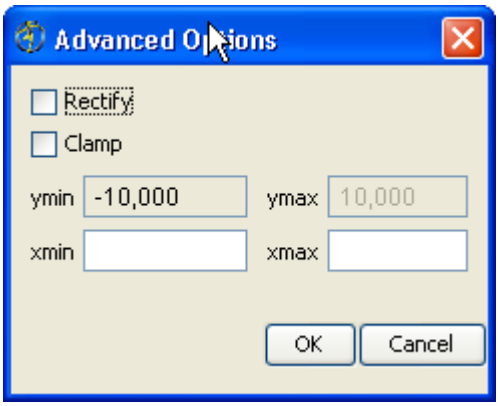


Figure 12-12: Advanced Option Dialog Box

chapter

13

Scaling

13.1 Overview

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.

This chapter covers how scaling and marker placement works, the input and output of the scale tool, how to use the GUI, and the settings used to configure the scale tool.

13.2 How It Works

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.

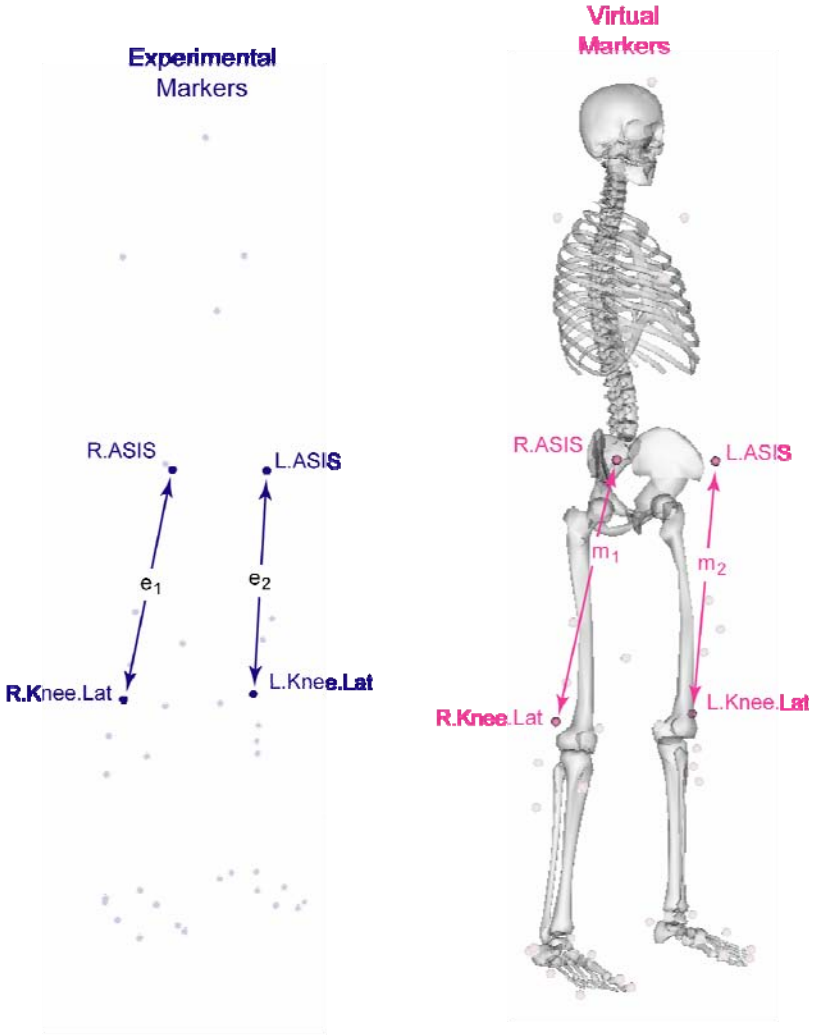


Figure 13-1: Experimental and Virtual Markers. Experimental marker positions are measured with motion capture equipment (dark blue). Virtual markers are placed on a model in anatomical correspondence (e.g., over the right anterior superior iliac spine (R.ASIS)). Distances between experimental markers (e_i) relative to the distances between corresponding virtual markers (m_i) are used to compute scale factors.

13.2.1 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.

13.2.1.1 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 Chapter 21 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 <SimmCoordinate> 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.

13.2.1.2 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.

13.2.1.3 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.

13.2.2 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 (see Chapter 14). 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. Refer to Chapter 14 for more background on how IK works. 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.

13.3 Inputs and Outputs

Figure 13-2 shows the required inputs and outputs for the Scale Tool. Each is described in more detail in the following sections.

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

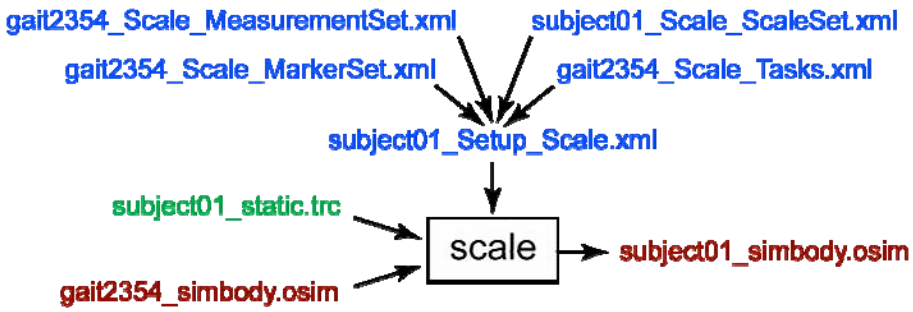


Figure 13-2: Inputs and Outputs of the Scale Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue.

13.3.1 Settings File(s)

The subject01_Setup_Scale.xml file is the setup file for the Scale Tool. It contains settings, described in detail in Section 13.6.1 , 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_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.

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.

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.

13.3.2 Input(s)

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.

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

13.3.3 Output(s)

The Scale Tool generates a single file:

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

13.4 How to Use the GUI for Scaling

The Scale Tool is accessed by selecting **Tools → Scaling...** from the OpenSim main menu bar. Like all tools, the operations performed by the Scale Tool apply to 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 Chapters 4 and 6 for information on opening models and making a particular model current.

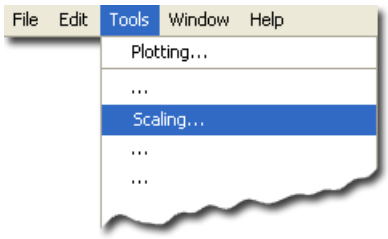


Figure 13-3: Tools Menu. The Scaling Tool is accessed via the Tools menu.

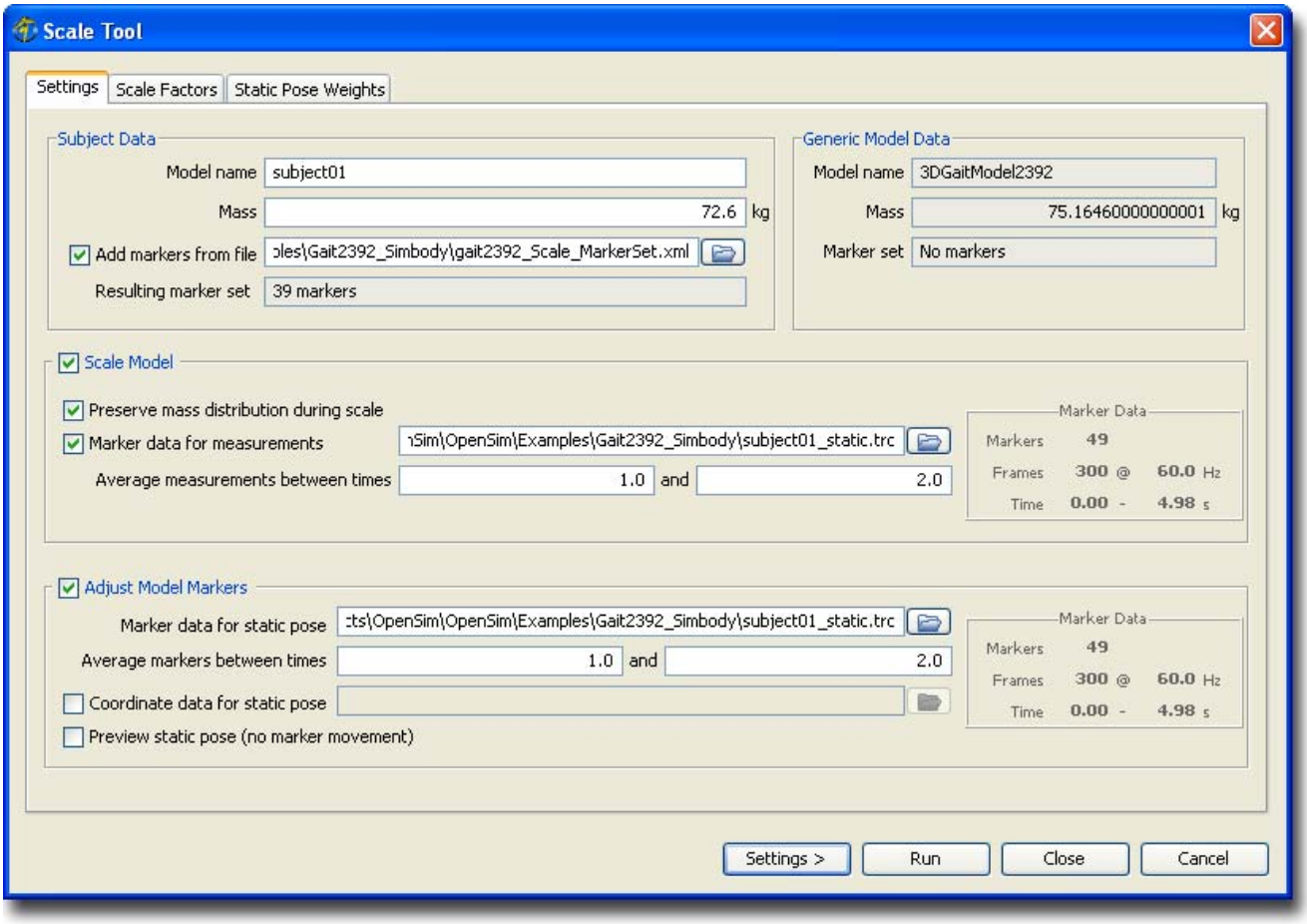


Figure 13-4: Window for the Scaling Tool

The Scale Tool is controlled by a window with three tabbed panes (Figure 13-4). 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 *Static Pose Weights* 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.

At the bottom of the window are four buttons. 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 (Figure 13-5). 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 Scale 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 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**.

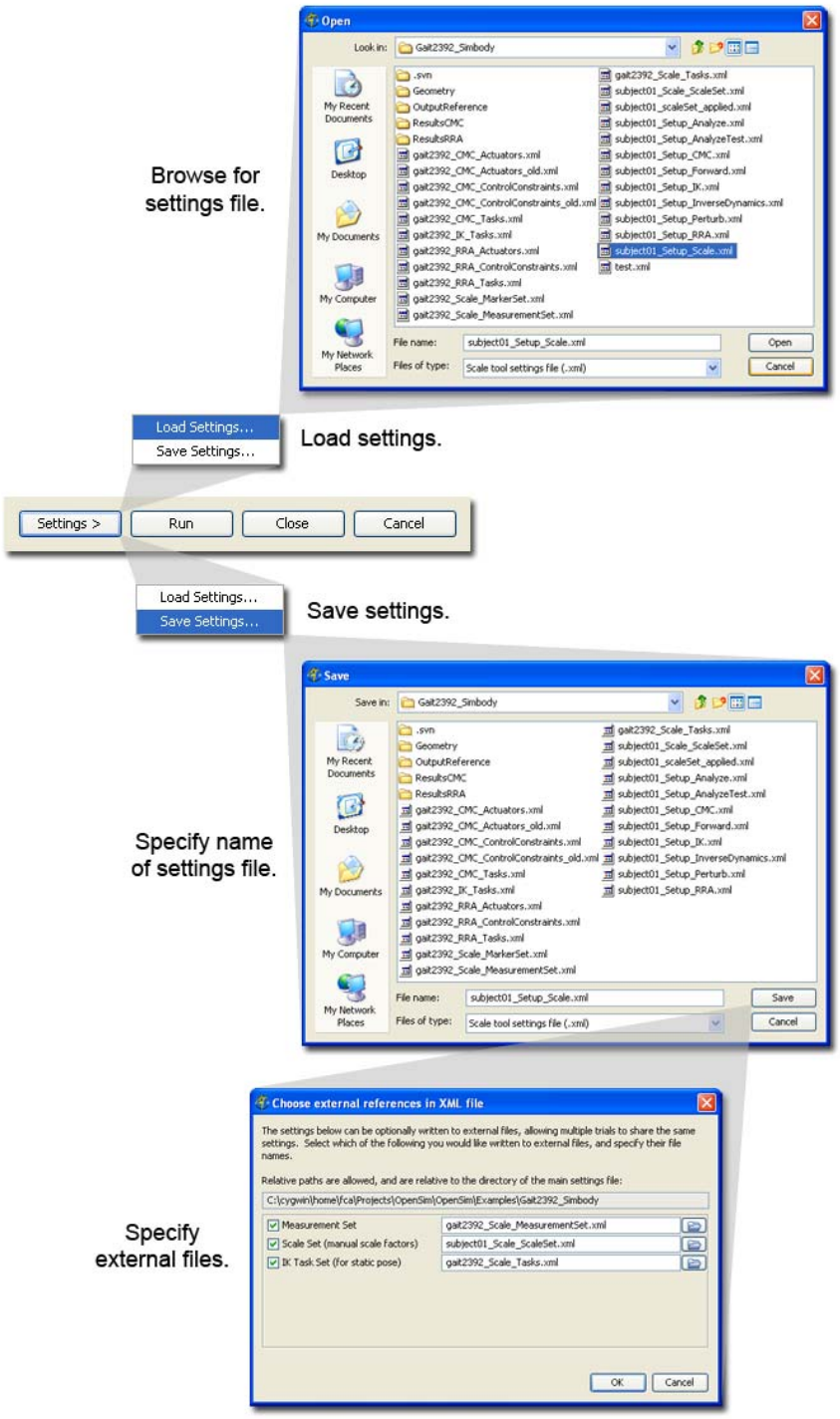


Figure 13-5: Saving and Loading Settings

13.4.1 Settings Pane

The *Settings* pane (Figure 13-4) 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 entitled *Generic Model Data*, *Subject Data*, *Scale Model*, and *Adjust Model Markers*.

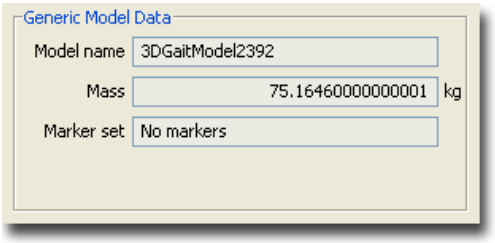


Figure 13-6: Generic Model Data Section

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 (Figure 13-6).

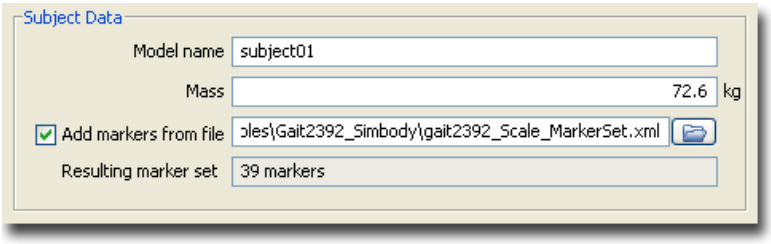


Figure 13-7: Subject Data Section

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.

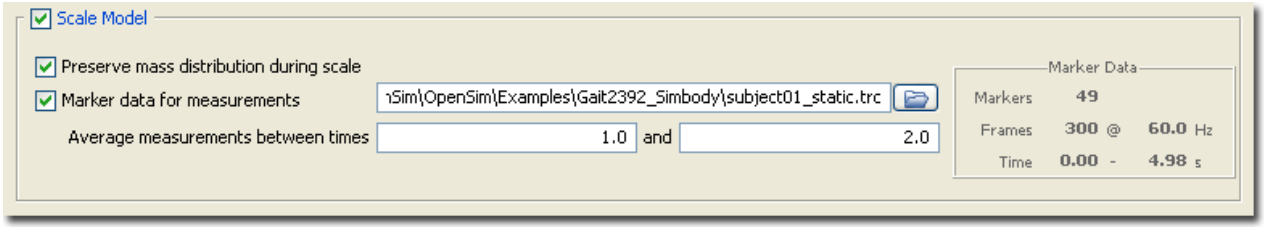


Figure 13-8: Controlling Model Scaling

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.

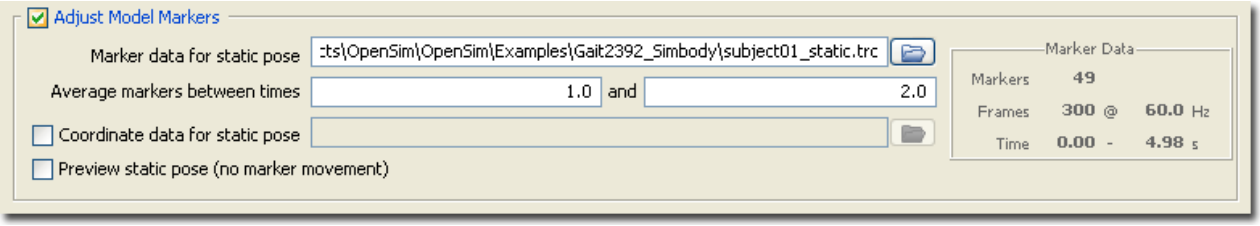


Figure 13-9: Adjusting Model Markers

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.

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.

13.4.2 Scale Factors Pane

The *Scale Factors* pane (Figure 13-10) 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. 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 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 (Figure 13-10), the settings for that segment become editable. You can select one of two radio buttons to **Use measurements** (Figure 13-11) or **Use manual scales** (Figure 13-12). 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.

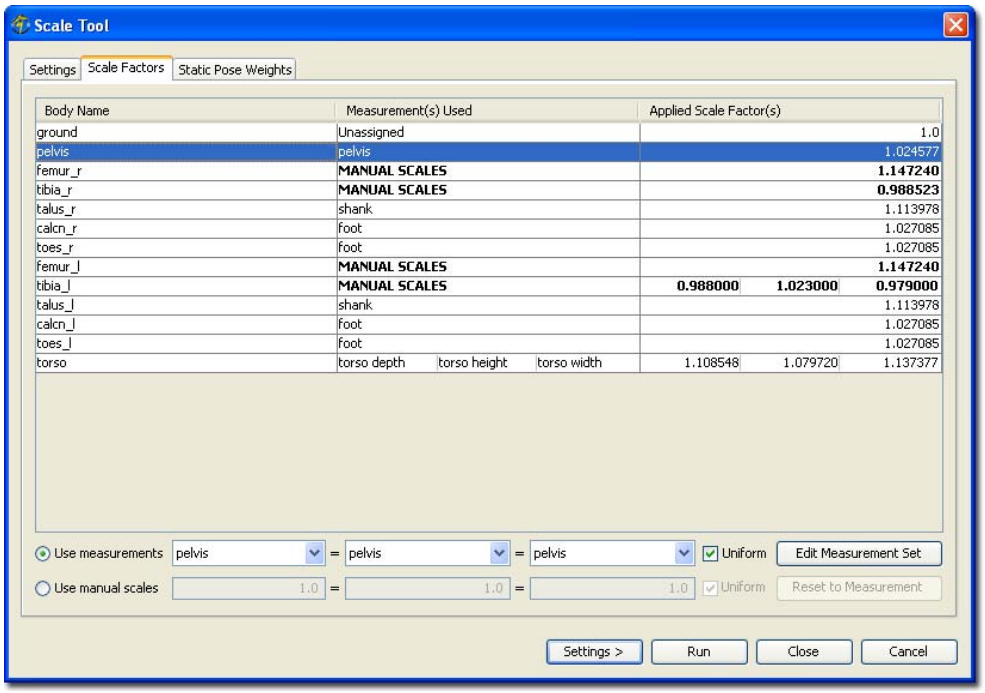


Figure 13-10: Scale Factors Pane. Scale factors for each segment can be specified manually (indicated by **MANUAL SCALES,** as shown for the femur_r segment) or based on the distances between a set of marker pairs. The measurement set is edited by clicking the **Edit Measurement Set** button (see Figure 13-13). 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).

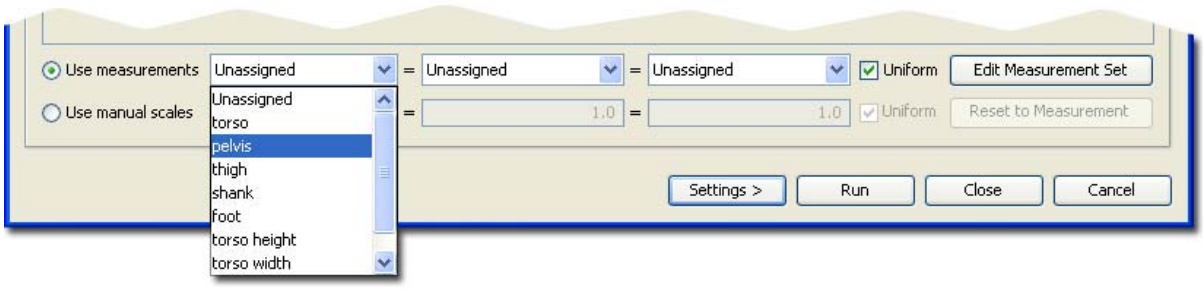


Figure 13-11: Using Measurement-based Scaling. To specify scale factors based on measurement, click the **Use measurements** radio button, click in any of the three scale-factor fields to the right, and select a measurement for the scale factors that will be applied along the X, Y, and Z axes of the selected body. 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. Use the **Edit Measurement Set** button to inspect and edit the measurements (Figure 13-13).

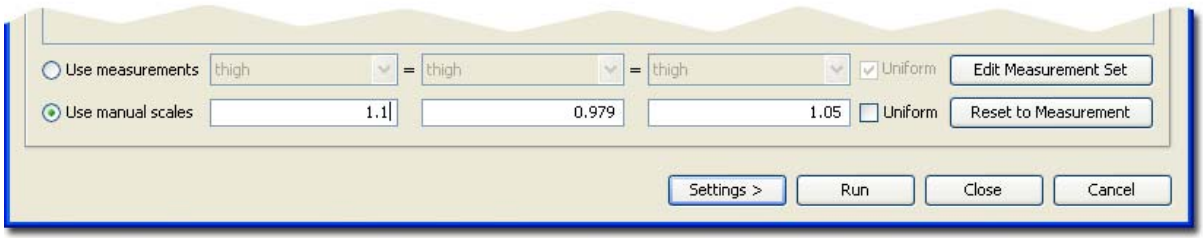


Figure 13-12: Using Manual Scale Factors. To specify scale factors manually, click the **Use manual scales** radio button and type in the three fields to the right to specify the factors that will be applied along the X, Y, and Z axes of the selected body. If the **Uniform** check box is unchecked, the 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 (Figure 13-11 and Figure 13-12). 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, visible in Figure 13-10 - Figure 13-12, but active only in Figure 13-12, 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.

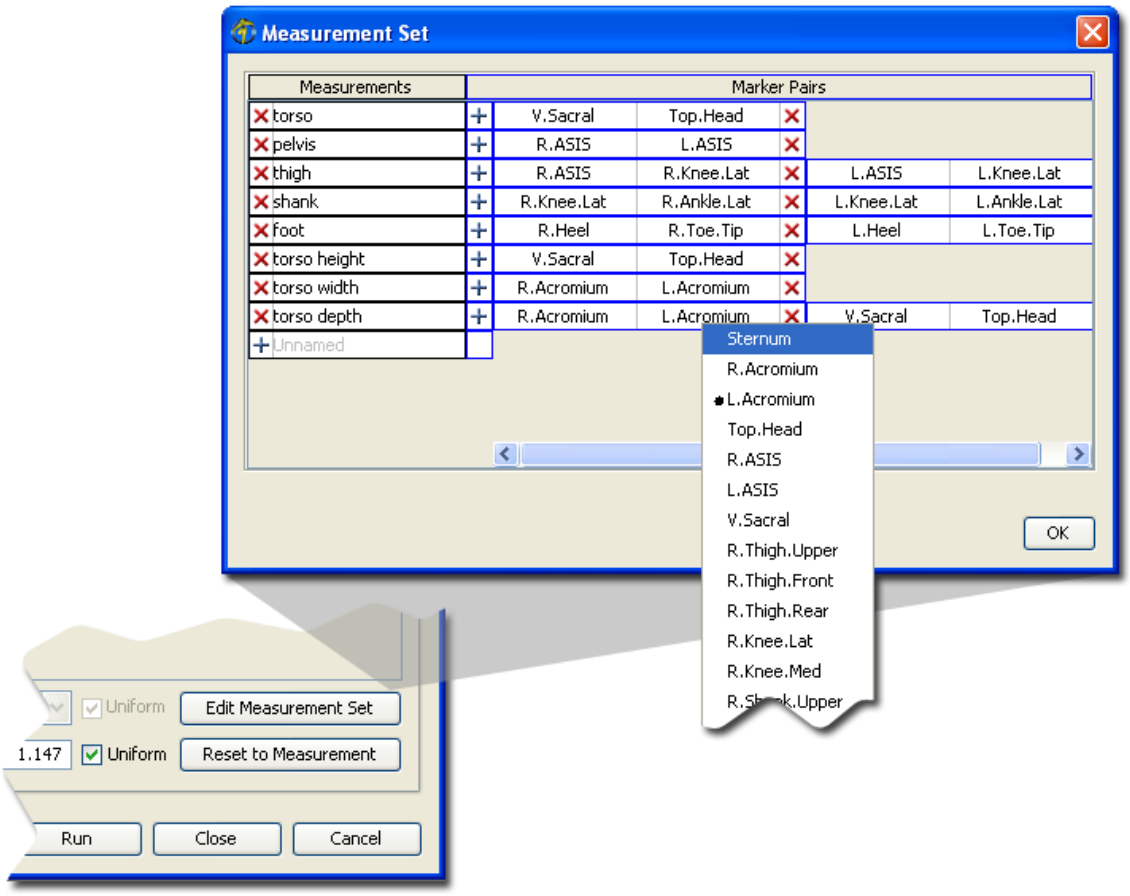


Figure 13-13: Editing the Measurement Set. Clicking the **Edit Measurement Set** button of the Scale Factors pane brings up an editor that allows you to inspect and edit the definitions of the measurements that can be used to compute scale factors for body segments. The name of each measurement is displayed 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 (Figure 13-13). 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 (see Section 13.2 for details). 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.

13.5 Command-line Execution

The Scaling Tool is run using the command **`scale -S <setup file name>`**, for example,

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

13.6 Settings Files and XML Tag Definitions

The settings files are XML files whose tags specify properties to be used by OpenSim for scaling models. The tags used for each type of settings file are defined in the following sections.

13.6.1 Scale Setup File

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 in Example 13-1.

Example 13-1: XML file for a scale setup file (e.g., subject01_Setup_Scale.xml)

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

<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>
```

13.6.1.1 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.

13.6.1.2 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.

13.6.1.3 <notes> Tag

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

13.6.1.4 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 13-1 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 Section 13.6.2 below.

13.6.1.5 Specifying Scaling Properties

Scaling properties are enclosed in the <ModelScaler> tag. As described in Section 13.2 (How It Works) above, 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.

13.6.1.5.1 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 Section 13.6.3 below for more details about the contents of this XML file.

13.6.1.5.2 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.

13.6.1.5.3 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 Section 13.2 above on How It Works).

13.6.1.5.4 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_joint_file> generates the scaled-only SIMM jnt file
- <output_muscle_file> generates the scaled-only SIMM msl file
- <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.

13.6.1.6 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 Section 13.2 (How It 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 (Chapter 14) 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 in Section 14.6.2 (Inverse Kinematics Tasks File) 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.

13.6.1.6.1 <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.

13.6.1.6.2 <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 Example 13-1, the coordinate values from time t=1 will be used as the "experimental coordinate values."

13.6.1.6.3 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 Section 13.6.2). The remaining markers are untouched.

13.6.1.6.4 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 for debugging and SIMM visualization purposes. These are:

- <output_joint_file> - the subject-specific SIMM jnt file
- <output_muscle_file> - the subject-specific SIMM msl file
- <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 does not appear to work.

13.6.2 Scale Marker File

The scale marker file contains a list of the virtual markers that are placed on the body segments of the model. In Figure 13-2, this is the file **gait2354_Scale_MarkerSet.xml**. An example of a marker file is shown in Example 13-2.

Example 13-2: XML file for a scale marker file (e.g., 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>
```

13.6.2.1 Specifying a List of Markers

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

13.6.2.2 Specifying a Marker for the Generic Model

Specifying a marker consists of specifying its <location>, 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 Example 13-2 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.

13.6.3 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. In Figure 13-2, the scale factors are specified in the file **gait2354_Scale_ScaleSet.xml**. An example of a manual scaling file is shown in Example 13-3.

13.6.3.1 <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 14-3 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.

13.6.3.2 Tag

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

13.6.3.3 <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 13-3: XML file for a manual scaling file (e.g., 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>
```

13.6.4 Measurement-Based Scaling File

The measurement-based scaling file contains pairs of experimental markers, the distance between which are used to scale the generic musculoskeletal model. In Figure 13-2, the experimental measurements used for scaling are specified in the file **gait2354_Scale_MeasurementSet.xml**.

Example 13-4: XML file for a measurement-based scaling file (e.g., 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>
```

An example of this type of file is shown in Example 13-4. 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.

13.6.4.1 <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.

13.6.4.2 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 13-4, two marker pairs are given. Recall from Section 13.2 (How It Works) that this means that the final scale factor will be computed by averaging the scale factors determined by each individual pair.

13.6.4.3 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.

13.6.5 Inverse Kinematics Tasks File

The inverse kinematics (IK) tasks file is an external XML file containing the <IKTaskSet> referred to from the main scaling Setup file above (Example 13-1). In Example 13-1, the IK tasks file is **gait2354_Scale_Tasks.xml**. Example 13-5 provides an example of this type of file.

The IK tasks file specifies properties associated with error terms. A brief description of the tags used to describe these properties is given below. Chapter 14 on Inverse Kinematics

provides more details.

13.6.5.1 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 (Chapter 14) for more details.

Example 13-5: XML file for an inverse kinematics tasks file (e.g., 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>
```

13.6.5.2 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 (Chapter 14) for more details. In Example 13-5, 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.

chapter

15

Inverse Kinematics (IK)

14.1 Overview

The purpose of the IK step is to find the set of generalized coordinates (joint angles and positions) for the model that best reproduce the experimental kinematics recorded for a particular subject. The experimental kinematics targeted by IK includes experimental marker positions as well as experimental generalized coordinate values.

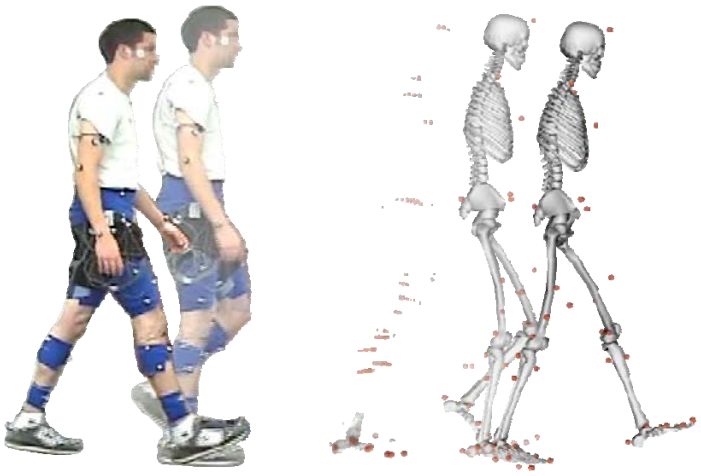


Figure 14-1: Inverse Kinematics (IK) Tool Overview. Experimental markers are matched by model markers throughout the motion by varying the joint angles (generalized coordinates) through time.

14.2 How It 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.

14.2.1 Marker Errors

A *marker error* is the distance between an experimental marker and the corresponding marker on the model (Figure 14-1) 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.

14.2.2 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.

14.2.3 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 (ω_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 the XML setup file section below (Section 14.6.2). 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.

14.2.4 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).

14.3 Inputs and Outputs

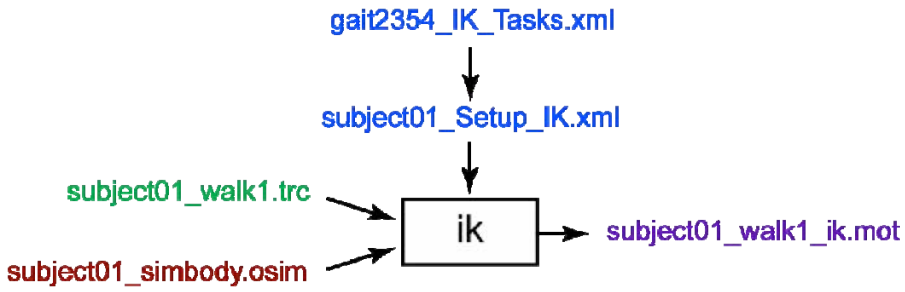


Figure 14-2: Inputs and Outputs of the IK Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue. *Note: These file names are examples that can be found in the examples/Gait2354_Simbody directory installed with the OpenSim distribution.*

The primary inputs to IK are the following files:

1. **Subject-specific OpenSim model** *[.osim file]* generated by scaling a generic model using the Scale Tool or by other means with an associated marker set.
2. **Experimental marker trajectories for each trial** *[.trc file]* obtained from the motion capture system.
3. **Experimental generalized coordinate values for each trial (optional)** *[.mot file]* obtained from alternative motion capture devices or other specialized algorithms.

For each trial processed by IK, the output is a motion file (.mot) containing the generalized coordinate trajectories computed by IK.

14.4 How to Use the GUI for Inverse Kinematics

To launch the IK Tool, select **Tools** → **Inverse Kinematics** from the OpenSim main menu bar (Figure 14-3). The **Inverse Kinematics Tool** window (Figure 14-4) will open. It has two tabbed panes. 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.

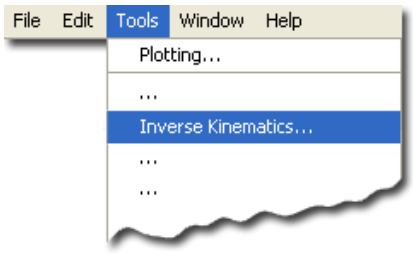


Figure 14-3: Tools Menu. The Inverse Kinematics Tool is accessed from the Tools menu.

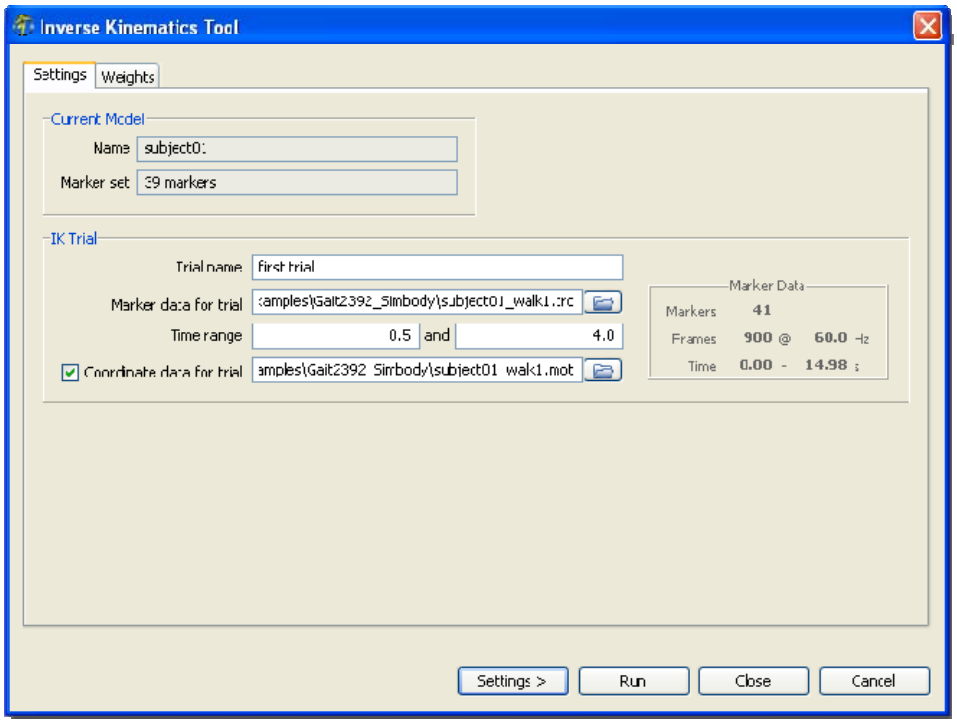


Figure 14-4: Inverse Kinematics Tool Window

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 Chapters 4 and 6 for information on opening models and making a particular model current.

14.4.1 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 (see Chapter 13 on Scaling for details).

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.

14.4.2 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 14-5). 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 Section 14.2 - How It 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.

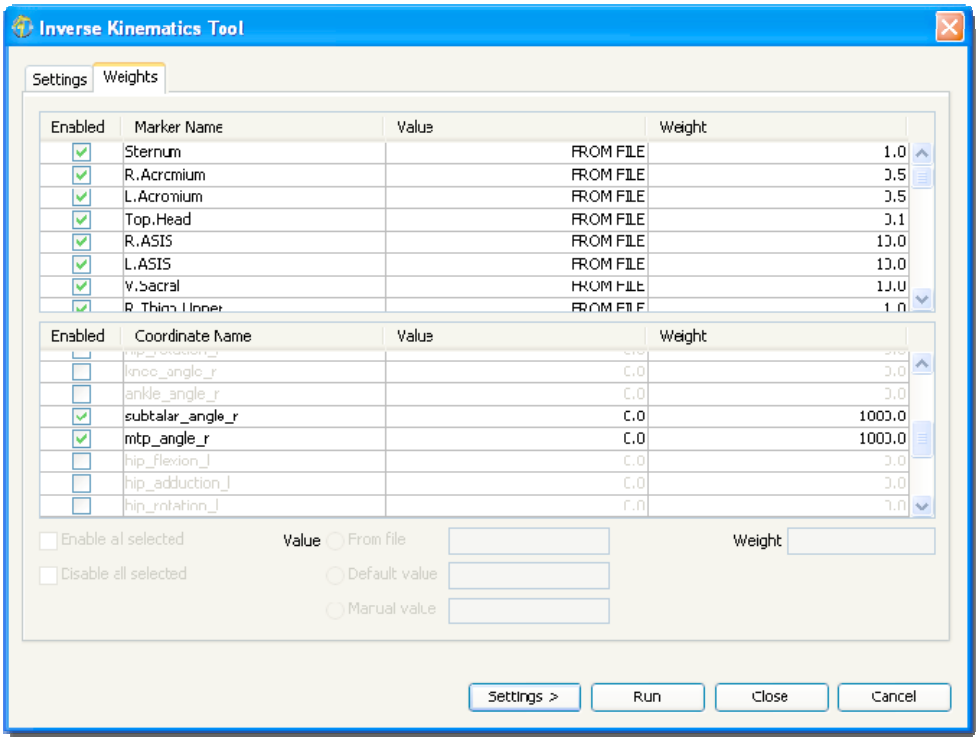


Figure 14-5: Specifying IK Tool Weightings

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 (Chapter 6). 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**.

14.5 Command-line Execution

The Inverse Kinematics Tool is run using the command **`ik -S <setup file name>.`**

14.6 Settings Files and XML Tag Definitions

The settings files are XML files whose tags specify properties to be used by OpenSim for performing the inverse kinematics. The tags used for each type of settings file are defined in the following sections.

14.6.1 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 Example 14-1.

Example 14-1: XML file for an inverse kinematics setup file (e.g., 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>
```

14.6.1.1 Specifying 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.

14.6.1.2 Specifying the 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.

14.6.1.3 Specifying the 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 Example 14-1, 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 in Section 14.6.2 below.

14.6.1.4 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 Example 14-1, 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.

14.6.1.4.1 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 Section 14.2 (How It 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 IK Task File, Section 14.6.2).

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.

14.6.1.4.2 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 Example 14-1 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.

14.6.1.4.3 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.

14.6.2 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 Example 14-1). An example IK tasks file is given in Example 14-2 below. Recall that this file is common to all IK trials.

Example 14-2: Annotated XML file for an inverse kinematics tasks file (e.g., gait2354_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>
</IKTaskSet>
```

If desired, the name attribute (e.g., name="gait2354_IK" as shown in Example 14-2) can be used to specify the execution name for IKTaskSet, though it is not used anywhere.

14.6.2.1 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 Section 14.2 (How It 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 IK setup file (Section 14.6.1.4).

14.6.2.2 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.

14.6.2.2.1 <weight> tag

The <weight> property specifies the weight multiplying the squared-error term for that coordinate. These weights are the ω_i's appearing in the formula given in Section 14.2 (How It 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 Section 14.2 (How It 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 (see Chapter 7) 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.

14.6.2.2.2 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 IK setup file (see Example 14-1).

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 Example 14-2 above, 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.

chapter

15

Inverse Dynamics

15.1 Overview

The inverse dynamics tool derives 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, $\boldsymbol{F} = m\boldsymbol{a}$, 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.

This chapter covers how inverse dynamics works, the input and output of the inverse dynamics tool, how to use the GUI, and the settings used to configure the inverse dynamics tool.

15.2 How it Works

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

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

where N is the number of degrees of freedom; $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \in \boldsymbol{R}^N$ are the vectors of generalized positions, velocities, and accelerations, respectively; $\boldsymbol{M}(\boldsymbol{q}) \in \boldsymbol{R}^{N\times N}$ is the system mass matrix; $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \boldsymbol{R}^N$ is the vector of Coriolis and centrifugal forces; $\boldsymbol{G}(\boldsymbol{q}) \in \boldsymbol{R}^N$ is the vector of gravitational forces; and $\boldsymbol{\tau} \in \boldsymbol{R}^N$ 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.

15.3 Inputs and Outputs

Figure 15-1 shows the required inputs and outputs for the Inverse Dynamics Tool. Each is described in more detail in the following sections.

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

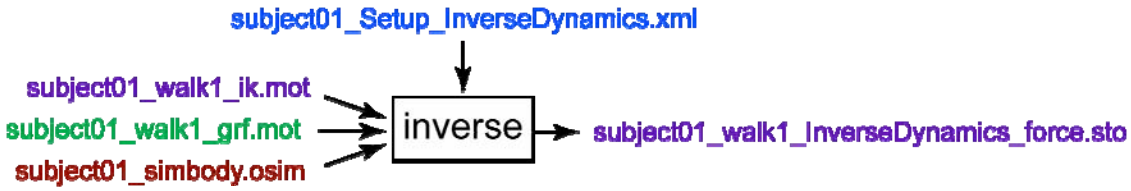


Figure 15-1: Inputs and Outputs of the Inverse Dynamics Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

15.3.1 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 Section 15.6.

15.3.2 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 was generated by the Inverse Kinematics Tool.

subject01_walk1_grf.mot: Motion file containing the measured ground reaction forces that should be applied to the model during simulation.

subject01_simbody.osim: OpenSim musculoskeletal model scaled to the dimensions of the subject. This file was generated by the Scale Tool.

15.3.3 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 forces and torques at each joint.

15.4 How to Use the GUI

The inverse dynamics tool is accessed by selecting **Tools → Inverse Dynamics...** from the OpenSim main menu bar (Figure 15-2). 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 chapters 4 and 6 for information on opening models and making a particular model current.

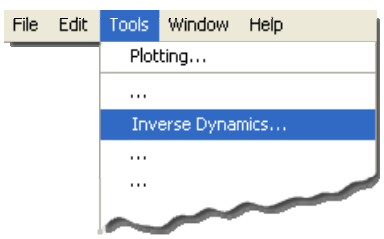


Figure 15-2: Tools Menu. The Inverse Dynamics Tool is accessed from the Tools menu.

The Inverse Dynamics Tool is controlled by a window with two tabbed panes (Figure 15-3). 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.

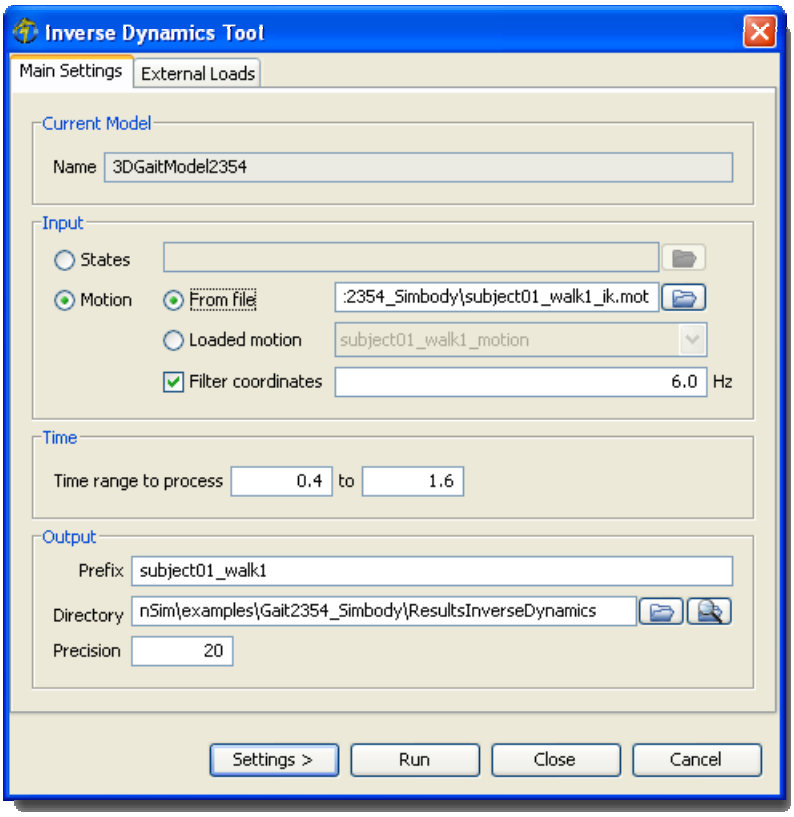


Figure 15-3: Window for the Inverse Dynamics Tool. The Inverse Dynamics window has two panes. The *Main Settings* pane is shown here.

At the bottom of the window are four buttons. 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.

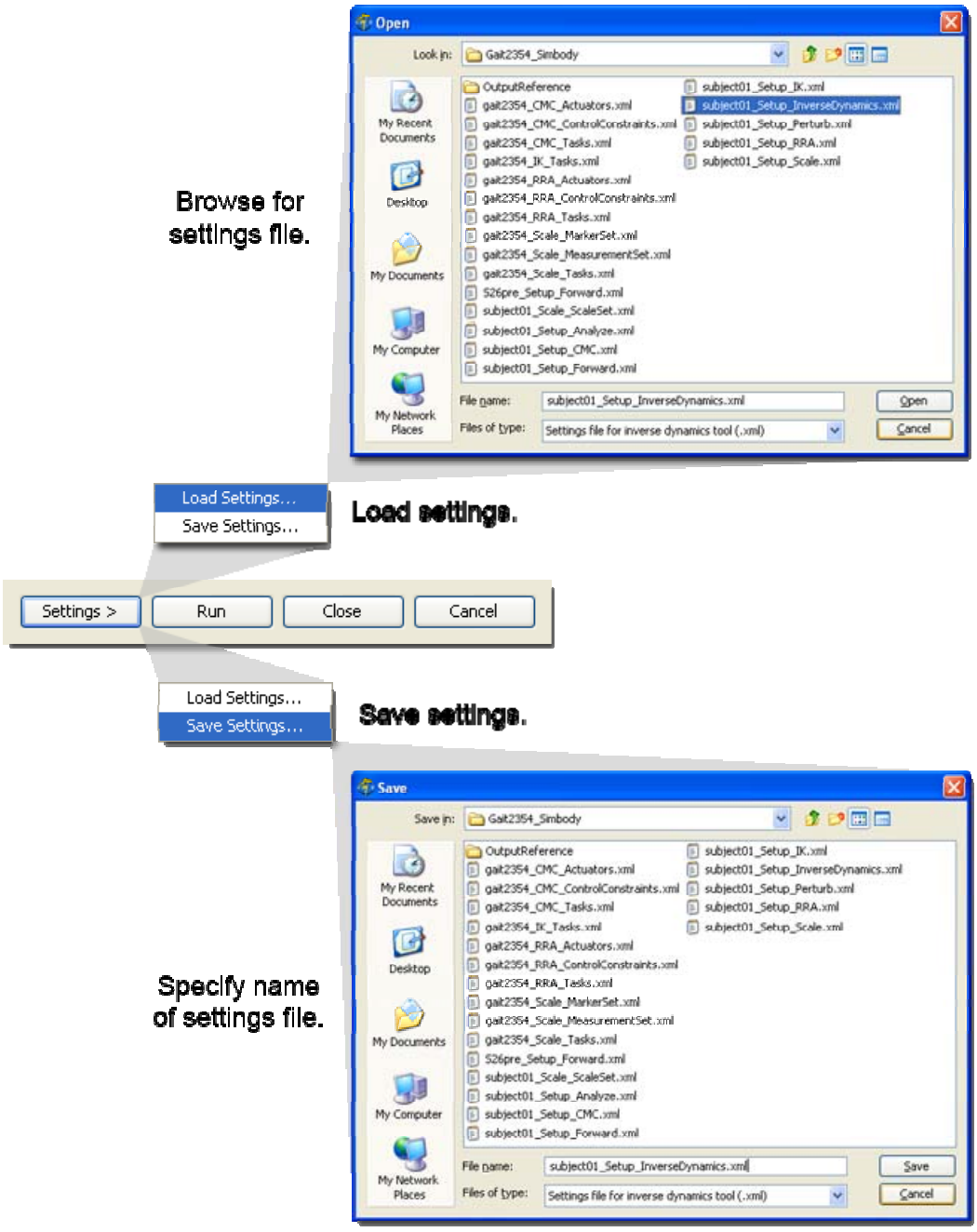


Figure 15-4: Saving and Loading Settings

When the **Settings >** button is clicked, you are presented with the choice of loading or saving settings for the tool (Figure 15-4). 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_Inverse_Scale.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.

15.4.1 Main Settings Pane

The *Main Settings* pane (Figure 15-3) 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*.

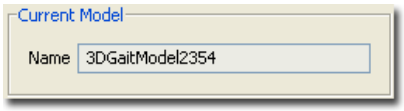


Figure 15-5: Current Model Section

The section for *Current Model* displays an uneditable name for the current model that is to be used for the inverse dynamics analysis (Figure 15-5).

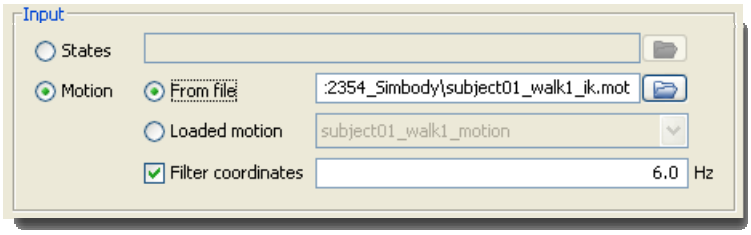


Figure 15-6: Input Section

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 (Figure 15-6). 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.

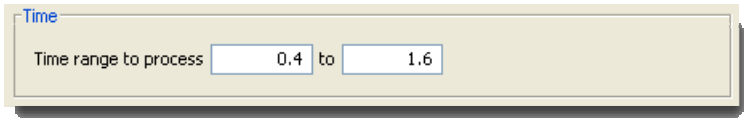


Figure 15-7: Time Section

The section for *Time* displays editable information that allows you to specify the start and end time for the inverse dynamics analysis (Figure 15-7).

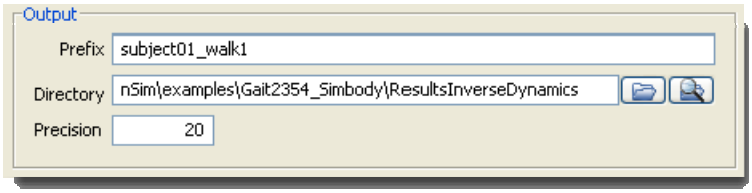


Figure 15-8: Output Section

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 (Figure 15-8). You may use the button to browse for and specify a directory in which to save the output files, and the button to open an Explorer window to the specified directory.

15.4.2 External Loads Pane

The *External Loads* pane (Figure 15-9) is used to specify parameters relating to the external loads applied to the model during the inverse dynamic analysis.

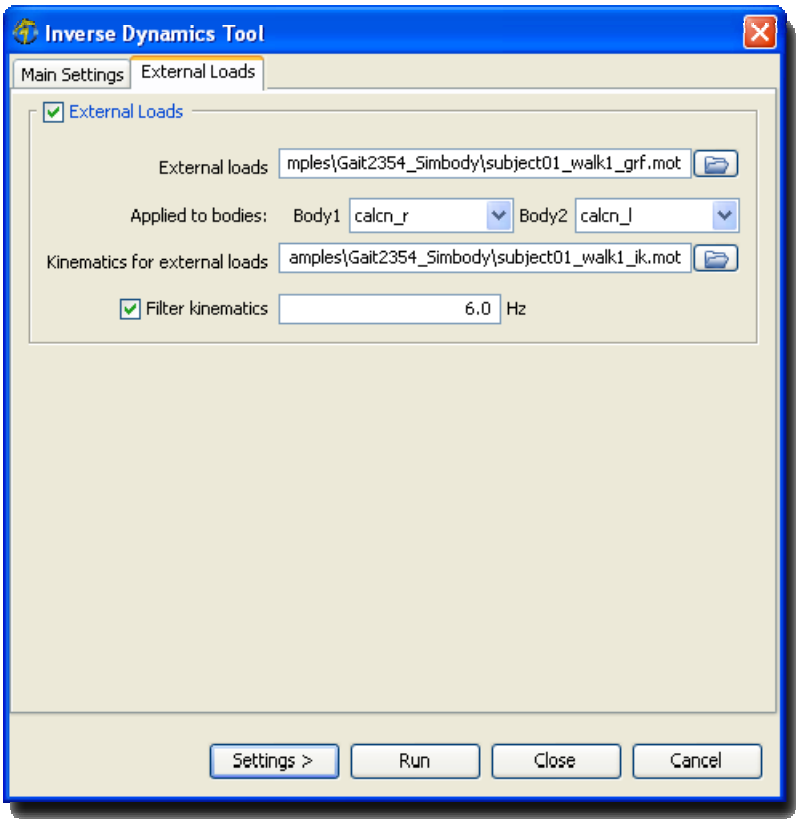


Figure 15-9: External Loads Pane

The section for *External Loads* is optional, and if checked, displays information that allows you to specify the external loads applied to the model, the bodies of the model to which the loads are applied, and the corresponding kinematics of the external loads. 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.

15.5 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
```

15.6 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 (Example 15-1) can be found in examples/Gait2354_Simbody.

Example 15-1: XML file for an inverse dynamics setup file (e.g., subject01_Setup_InverseDynamics.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
       <actuator_set_files>? If false, the actuator set is appended.-->
   <replace_actuator_set> false </replace_actuator_set>

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

   <!--Directory used for writing results.-->
   <results_directory> ResultsInverseDynamics </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>
         <InverseDynamics name="InverseDynamics">
            <on> true </on>
            <step_interval> 1 </step_interval>
            <use_model_actuator_set> false </use_model_actuator_set>
         </InverseDynamics>
      </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>

    <!--Motion file (.mot) or storage file (.sto) containing the external
        loads applied to the model.-->
    <external_loads_file> subject01_walk1_grf.mot </external_loads_file>

    <!--Motion file (.mot) or storage file (.sto) containing the model
        kinematics corresponding to the external loads.-->
    <external_loads_model_kinematics_file> subject01_walk1_ik.mot
    </external_loads_model_kinematics_file>

    <!--Name of the body to which the first set of external loads should be
        applied (e.g., the name of the right foot).-->
    <external_loads_body1> calcn_r </external_loads_body1>

    <!--Name of the body to which the second set of external loads should
        be applied (e.g., the name of the left foot).-->
    <external_loads_body2> calcn_l </external_loads_body2>

    <!--Low-pass cut-off frequency for filtering the model kinematics
        corresponding to the external loads. A negative value results in no
        filtering. The default value is -1.0, so no filtering.-->
    <lowpass_cutoff_frequency_for_load_kinematics> 6
    </lowpass_cutoff_frequency_for_load_kinematics>

</AnalyzeTool>
```

15.6.1 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.

15.6.2 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.

15.6.3 Specifying the Actuator Set

The <replace_actuator_set> allows the modification of the actuators defined in the model file. If the value of <replace_actuator_set> is **true**, then the actuators specified in any file listed under the property <actuator_set_files> will replace the corresponding model's actuators. If the value of <replace_actuator_set> is **false**, then the actuators specified in the files listed under <actuator_set_files> will be added to the model's existing actuator set.

15.6.4 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.

15.6.5 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.

15.6.6 Specifying an Inverse Dynamics 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 inverse dynamics analysis are enclosed inside the opening and closing tags <InverseDynamics> and </InverseDynamics>. The name attribute name="InverseDynamics" specifies the analysis name. The name of the results file generated will include this name. The <on> tag specifies whether or not the inverse dynamics analysis should be performed. The <step_interval> tag specifies how often to record results during the inverse dynamics analysis. A value of 1 means results are recorded every 1 frame of motion data. The <use_model_actuator_set> tag specifies whether or not the model's actuator set will be used in the inverse dynamics analysis. If not, generalized forces (e.g., net joint forces and torques) will be computed for all unconstrained degrees of freedom.

15.6.7 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.

15.6.8 Specifying External Loads

The <external_loads_file> tag specifies the file containing the external loads applied to the model during a inverse dynamics analysis. The loads and points of application are given in the global frame. The <external_loads_model_kinematics_file> tag specifies the associated model kinematics for the external loads. Using this kinematics information, the loads are transformed into the local frames of the body segments to which they are applied. Loads may be applied to up to two bodies. The <external_loads_body1> and <external_loads_body2> tags specify the first body and the second body, respectively. The <lowpass_cutoff_frequency_for_load_kinematics> tag specifies the low-pass cutoff frequency for filtering the model kinematics for the external loads. If the kinematics have already been filtered, specify a negative cutoff frequency to prevent filtering.

chapter

16

Residual Reduction

16.1 Overview

The residual reduction algorithm (RRA) consists of two passes: RRA1 and RRA2. The purpose of the RRA1 step is to alter the torso mass center of a dynamic subject-specific model so that excessive leaning of the torso in the left-right or fore-aft directions is corrected. The purpose of the RRA2 step is to alter the kinematics of the model to be more consistent with the ground reaction data.

16.2 How It Works

RRA consists of two passes: RRA1 and RRA2. In both passes, we perform a simulation to compute the joint torques needed to drive a dynamic subject-specific skeletal model to follow a prescribed motion (the motion computed by the inverse kinematics solver). In RRA, the model has no muscles to apply forces to the skeleton. Instead, the model has torque actuators at each joint to apply forces to the skeleton.

RRA is only 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 RRA as it works for a model consisting of ten rigid segments (bones). We will refer to this model as the gait23 model. 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 joint with its own torque actuator. Each of these 6 torque 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.

There are other reasons for needing actuators between the ground and the model. Typically, modeling assumptions (e.g., having a model with no arms) and numerical error from motion capture data lead to *dynamic inconsistency*; essentially, the forces and motion measured for a subject do not satisfy Newton's Second Law, $F = ma$. Roughly speaking, the 6 residuals amount to adding a new term to the equation that makes Newton's Second Law hold:

$$F + F_{residual} = ma$$

16.2.1 RRA Pass 1 (RRA1)

16.2.1.1 Simulation

RRA1 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) solver for the user-specified initial time (specified in the setup file as the <initial_time> property). Repeatedly, RRA1 takes small steps forward in time (with each time step less than or equal to the <cmc_time_window> property in the setup file) 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 23 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 Section 16.5.1.6 on Optimization Parameters below).

16.2.1.2 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 any "leaning" the model is doing due to inaccuracies in the representation of the torso geometry. A new model file containing the adjusted torso mass center (specified in the setup file under the <output_model_file> property) is created.

16.2.1.3 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 has the option of typing these changes into the OpenSim model file by hand. These mass changes are NOT applied to the model automatically.

16.2.2 RRA Pass 2 (RRA2)

16.2.2.1 Simulation

The same simulation process as in RRA1 is repeated in RRA2, but with three important changes:

- A new 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 during computed muscle control (CMC), the next stage of OpenSim, the forces exerted by muscles are in fact performing the functions that the muscles would perform in reality (instead of having the residuals exert forces that normally would be 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 they achieved in RRA1 to follow the kinematics with practically zero error. If the minimum and/or maximum allowed residual values are too restrictive, the motion will be altered so dramatically that the results of RRA2 cannot be used to generate a realistic simulation in CMC. If the residual minimum and/or maximum values are too lenient, then although RRA2 will be able to match the kinematics about as well as RRA1, the residuals will still be large enough to exert forces that would normally be exerted by muscles, and thus the results would lead to unrealistic muscle function in CMC.

16.3 Inputs and Outputs

The inputs into the RRA1 tool are:

- a dynamic subject-specific musculoskeletal model
- kinematics (generalized coordinates and generalized speeds of the model as functions of time) output by the inverse kinematics (IK) step
- ground reaction force, torque, and center of pressure (point of application of ground reaction force) for each foot

The main output of the RRA1 tool is a dynamic model with an adjusted torso mass center.

The inputs into RRA2 are the same as for RRA1, but with the new model output by RRA1 replacing the original dynamic model. The main output of RRA2 is a modified set of kinematics that is more consistent with the ground reaction data than the original kinematics computed by

the IK solver.

16.4 Command-line Execution

RRA1 and RRA2 are both run using the command **cmcgait -S <setup file name>**, for example,

```
cmcgait -S subject01_Setup_RRA1.xml
cmcgait -S subject01_Setup_RRA2.xml
```

16.5 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.

Two sets of files are required, one for each pass of the algorithm: RRA1 and RRA2. The two sets of files are very similar, so full explanations are provided for the RRA1 files and only differences between the RRA1 and RRA2 files are highlighted in the explanations.

16.5.1 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 16-1.

In the setup file, the property settings for RRA1, as well as RRA2, are enclosed in <CMCTool> tags since RRA and the computed muscle control (CMC) are versions of the same program. The types of properties listed in the XML setup files for RRA1 and RRA2 include model files, actuator and control information, integration parameters, kinematics and ground reaction data files, tracking information, optimization parameters, and output information.

16.5.1.1 Model Files

The <model_library> property specifies the model library to load. In Example 16-1, the value of this property was **gait23**, which means that OpenSim will load the scalable gait model library files.

Example 16-1: XML file for the setup file for residual reduction (e.g., subject01_Setup_RRA1.xml)

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

<CMCTool name="subject01_walk1_RRA1">

    <defaults/>
    <model_library> gait23 </model_library>
    <model_file> subject01_sdfast.osim </model_file>


    <initial_time> 0.75 </initial_time>
    <final_time> 1.25 </final_time>
    <cmc_time_window> 0.001 </cmc_time_window>
    <maximum_number_of_integrator_steps> 20000
           </maximum_number_of_integrator_steps>
    <maximum_integrator_step_size> 0.001
           </maximum_integrator_step_size>
    <integrator_error_tolerance> 0.001 </integrator_error_tolerance>
    <integrator_fine_tolerance> 1e-005 </integrator_fine_tolerance>



    <task_set_file> gait2354_RRA1_Tasks.xml </task_set_file>


    <replace_actuator_set> true </replace_actuator_set>
    <actuator_set_files> gait2354_RRA1_Actuators.xml
           </actuator_set_files>
    <constraints_file> gait2354_RRA1_ControlConstraints.xml
           </constraints_file>


    <desired_kinematics_file> subject01_walk1_ik.mot
           </desired_kinematics_file>
    <lowpass_cutoff_frequency> 6 </lowpass_cutoff_frequency>
    <external_loads_file> subject01_walk1_grf.mot
           </external_loads_file>
    <external_loads_model_kinematics_file> subject01_walk1_ik.mot
           </external_loads_model_kinematics_file>
    <external_loads_body1> calcn_r </external_loads_body1>
    <external_loads_body2> calcn_l </external_loads_body2>
    <lowpass_cutoff_frequency_for_load_kinematics> 6
           </lowpass_cutoff_frequency_for_load_kinematics>
```

```
    <use_fast_optimization_target> false
          </use_fast_optimization_target>
    <optimizer_derivative_dx> 0.0001 </optimizer_derivative_dx>
    <optimizer_convergence_criterion> 1e-006
          </optimizer_convergence_criterion>
    <optimizer_max_iterations> 2000 </optimizer_max_iterations>
    <optimizer_print_level> 0 </optimizer_print_level>


    <use_curvature_filter> false </use_curvature_filter>


    <results_directory> Results/ </results_directory>
    <output_precision> 8 </output_precision>
    <output_model_file> subject01_sdfast_adjusted.osim
          </output_model_file>

    <adjusted_com_body> torso </adjusted_com_body>
    <compute_average_residuals> true </compute_average_residuals>
    <adjust_com_to_reduce_residuals> true
          </adjust_com_to_reduce_residuals>

</CMCTool>
```

The <model_file> property specifies the name of the .osim file to load. In RRA1, the value of this property in the above example is **subject01_sdfast.osim**, the .osim file representing the dynamic subject-specific model. In RRA2, the value of this property would be the file for the model with an adjusted torso mass center, as computed by RRA1. So <model_file> for RRA2 is the same as the <output_model_file> for RRA1, which in Example 16-1 is **subject01_sdfast_adjusted.osim**.

16.5.1.2 Integration Parameters

The <maximum_number_of_integrator_steps> property indicates the maximum number of steps RRA1 or RRA2 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 RRA1 (Section 16.2.1.1).

16.5.1.3 Tracking Information

The coordinates that should be followed by the model during RRA1 or RRA2 are specified within a file, indicated by the <task_set_file> and </task_set_file> tags. In Example 16-1, that file is **gait2354_RRA1_Tasks.xml**. In Example 16-5, the setup file for RRA2, the task file is **gait2354_RRA2_Tasks.xsml**. See Section 16.5.4 for information about the property tags used within a task file.

16.5.1.4 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 16-1 has 54 muscles as its default set of actuators. These actuators can be modified using the tag <replace_actuator_set>. If the value of the property <replace_actuator_set> is **true**, then the actuators specified in any file listed under the property <actuator_set_files> will replace the corresponding model's actuators. If the value of <replace_actuator_set> is **false**, then the actuators specified in the files listed under <actuator_set_files> will be added to the model's existing actuator set. Details about the actuators file are given in Section 16.5.2 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 Section 16.5.3 for more information about the constraints file and the best method for altering an actuator's range.

16.5.1.5 Kinematics and Ground Reaction Data Files

RRA1 and RRA2 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 RRA1 or RRA2 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.

The <external_loads_file> tag is used to specify a .mot or .sto file containing the ground reaction data that will be applied to the model during simulation.

The <external_loads_model_kinematics_file> tag also specifies a .mot or .sto file. However, this file contains the model kinematics corresponding to the ground reaction data. Required columns in this file are:

- (ground reaction force vector for the right foot) **ground_force_vx**, **ground_force_vy**, **ground_force_vz**
- (center of pressure for the right foot) **ground_force_px**, **ground_force_py**, **ground_force_pz**
- (ground reaction force vector for the left foot) **ground_force_vx**, **ground_force_vy**, **ground_force_vz**
- (center of pressure for the left foot) **ground_force_px**, **ground_force_py**, **ground_force_pz**
- (ground reaction torque vector for the right foot) **ground_torque_x**, **ground_torque_y**, **ground_torque_z**
- (ground reaction torque vector for the left foot) **ground_torque_x**, **ground_torque_y**, **ground_torque_z**.

Typically the columns appear in the order presented above, although the only real requirement is that the first occurrence of any column name is for the right foot, while the second occurrence of a column name is for the left foot. See Chapter 21 on Preparing Your Data for more details about the file format.

The kinematics used to compute the location of each foot during simulation are low-pass filtered at a frequency specified by the <lowpass_cutoff_frequency_for_load_kinematics> tag. This frequency is assumed to be in Hertz (Hz). The ground reaction forces and torques are always applied in the reference frame of the appropriate foot so that even if the model's foot is not following the desired motion exactly, the ground reaction forces and torques are still applied to the actual location of each foot during simulation.

There are two sets of ground reaction data specified in the file identified by the <external_loads_file> tags. Typically, the first set corresponds to the ground reaction forces, torques, and center of pressure for the right foot, while the second set corresponds to the analogous data for the left foot.

16.5.1.6 Optimization Parameters

There are two possible objective functions to use during static optimization in RRA1 or RRA2: the fast target and the slow target.

The property <use_fast_optimization_target> indicates which of the two targets RRA1 or RRA2 should use. If the value of <use_fast_optimization_target> is set to **true**, then the fast target is used; if the value of <use_fast_optimization_target> is set to **false**, then the slow target is used.

The slow target consists of an objective function (J) that is a weighted sum of squared actuator controls plus the sum of desired acceleration errors:

$$J = \sum_{i=1}^{nx} x_i^2 + \sum_{j=1}^{nq} w_j (\ddot{q}_j * - \ddot{q}_j)$$

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 fast target is the sum of squared controls augmented by a set of equality constraints (C_j) that requires the desired accelerations to be achieved within the tolerance set for the optimizer:

$$J = \sum_{i=1}^{nx} x_i^2 \; ; \qquad C_j = \ddot{q}_j * - \ddot{q}_j, \quad for\ all\ \ j$$

The fast target is both faster and generally produces better tracking. However, if the constraints cannot be met, the fast target will fail and RRA1 or RRA2 will exit with an error message. Often the reason for the failure is that the dynamic subject-specific model is not strong enough.

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.

16.5.1.7 Curvature Filter

The <use_curvature_filter> property is only relevant for computed muscle control (CMC), the next step in OpenSim. See Chapter 17 for more details.

16.5.1.8 Output

Results printed by RRA1 or RRA2 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 all RRA1 or RRA2 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.

If the <compute_average_residuals> property is **true**, the average residuals computed during RRA will be printed out to a file in the <results_directory>.

16.5.2 RRA Actuators File

The RRA actuators file uses the <ActuatorSet> and </ActuatorSet>tags to describe actuators that replace the model's actuators. Example 16-2 below shows an actuator file for RRA1 (**gait2354_RRA1_Actuators.xml**). An example actuator file for RRA2 is provided in Section 16.5.5.

Example 16-2: XML file for the actuator set file (e.g., gait2354_RRA1_Actuators.xml)

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

<ActuatorSet name="gait2354_RRA1">

    <defaults>

        <Force name="default">
            <max_force> 10000.000 </max_force>
            <min_force> -10000.000 </min_force>
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 1.000 0.000 0.000 </direction_A>
            <body_B> </body_B>
            <point_B> 0.000 0.000 0.000 </point_B>
        </Force>

        <Torque name="default">
            <max_force> 10000.000 </max_force>
            <min_force> -10000.000 </min_force>
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> </body_A>
            <direction_A> 1.000 0.000 0.000 </direction_A>
            <body_B> </body_B>
        </Torque>

        <GeneralizedForce name="default">
            <max_force> 10000.000 </max_force>
            <min_force> -10000.000 </min_force>
            <coordinate> </coordinate>
        </GeneralizedForce>

    </defaults>

    <objects>

        <!-- Residuals -->

        <Force name="FX">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> -1.000 0.000 0.000 </direction_A>
```

```
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Force name="FY">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 0.000 -1.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Force name="FZ">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 0.000 0.000 -1.000 </direction_A>
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Torque name="MX">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> -1.000 0.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>

        <Torque name="MY">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> 0.000 -1.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>

        <Torque name="MZ">
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> 0.000 0.000 -1.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>


        <!-- Right Leg -->

        <GeneralizedForce name="hip_flexion_r">
            <coordinate> hip_flexion_r </coordinate>
            <optimal_force> 1000.0 </optimal_force>
        </GeneralizedForce>
```

```
  <!--..additional <GeneralizedForce> tags cut for brevity..-->
    </objects>
</ActuatorSet>
```

There are three types of actuators: <Force>, <Torque>, and <GeneralizedForce>. In Example 16-2 above, the residual forces and torques are specified as <Force> and <Torque> actuators respectively, while the joint torque actuators are <GeneralizedForce> actuators.

All three types of actuators contain an <optimal_force> property. Recall from Section 16.5.1.4 above on the RRA setup file that the maximum (or minimum) value for an actuator is the product of the <default_max> value (or the <default_min> value) and the <optimal_force> value. The <default_min> and the <default_max> values are specified in the RRA control constraints file (see Section 16.5.3). So for example, if the <default_min> value is **–20.0** and the <default_max> value is **20.0** for the residual actuator F_x, and the <optimal_force> is **1000.0** as specified in Example 16-2 above, then the actual values of F_x range from –2000.0 to 2000.0.

The <optimal_force> property for each actuator represents its weight in the objective function described Section 16.5.1.6. Recall that the objective function is used to distribute forces among all actuators during the simulation stage of RRA1 and RRA2. Thus, if you wish to adjust the minimum and maximum allowable values for an actuator, you should changee the <default_min> and <default_max> values in the control constraints file rather than changing the <optimal_force> property in the actuators file.

The <Force> and <Torque> actuators require the following properties: <body_A>, <direction_A>, and <body_B>. The <Force> actuator also requires the <point_A> and <point_B> properties. A <Force> actuator applies a force from <point_A> on <body_A> to <point_B> on <body_B> in the direction of the vector <direction_A>. This vector should be a unit vector, but if it is not a unit vector and is nonzero, then it will be normalized to become a unit vector.

Note: You must manually enter the mass center coordinates of any body segment (e.g., the pelvis) to which an actuator force (e.g., residual forces F_X, F_y, and F_z) applies into the <point_B> *property of each *_Actuators.xml file (including the actuators file for CMC). This is especially important after scaling a model.* The mass center coordinates can be found under the

"masscenter" field of the segment in the subject-specific model (*.osim) file. If you do not make this change, the residual forces will be applied to a point on the body segment other than its mass center. In general, the mass center is NOT the point (0, 0, 0) in the body frame.

Similarly, a <Torque> actuator applies a torque from <body_A> to <body_B> in the direction of the vector <direction_A>. The <GeneralizedForce> actuator requires a <coordinate> property specifying the name of the generalized coordinate to which the generalized force is applied.

16.5.3 RRA Control Constraints File

The constraints file uses the tags <ControlSet> and </ControlSet> to describe the controls in a simulation. This constraint file is referred to in the RRA setup file using the tags <constraints_file> and </constraints_file> (**gait2354_RRA1_ControlConstraints.xml** for RRA1 in Example 16-1 above and **gait2354_RRA2_ControlConstraints.xml** for RRA2 in Example 16-7 below).

Properties associated with controls are: <is_model_control> representing whether or not the control is associated with a model, <extrapolate> indicating whether or not the control should use extrapolation for times outside its time range, <default_min> representing the control's minimum allowed value, and <default_max> representing the control's maximum allowed value.

Recall from Section 16.5.1 above on the RRA setup file that the maximum (or minimum) value for an actuator is the product of the <default_max> value (or the <default_min> value), specified here in the constraints file, and the <optimal_force> value, specified in the actuators file above (see Section 16.5.2). Details about adjusting these values to set the range of values for an actuator are given in Section 16.5.2 above.

A <ControlLinear> control (the only type of control used in Example 16-3 below) also has other properties such as <use_steps>. If <use_steps> is **true**, the variation of control values with respect to time will be a step function (piecewise constant), whereas if <use_steps> is **false**, the control values will be a piecewise linear function of time.

Example 16-3: XML file for the control constraints file (e.g., gait2354_RRA1_ControlConstraints.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<ControlSet name="gait2354_RRA1">
    <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" />
        <ControlLinear name="FY.excitation" />
        <ControlLinear name="FZ.excitation" />
        <ControlLinear name="MX.excitation" />
        <ControlLinear name="MY.excitation" />
        <ControlLinear name="MZ.excitation" />

        <!-- 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>
```

16.5.4 RRA Task File

The task file uses the tag <rdCMC_TaskSet> to list the coordinates that should be followed by the model during RRA1 or RRA2. Each task involves some coordinate or set of coordinates to be followed by the model. Each generalized coordinate to be followed is specified under the <rdCMC_Joint> property. If any operational space point (i.e., a point on a body) needs to be followed, that task can be specified under the <rdCMC_Point> property.

Example 16-4: XML file for the tasks file (e.g., gait2354_RRA1_Tasks.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdCMC_TaskSet name="gait2354_RRA1">

    <defaults>
        <rdCMC_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>
            <r0> 0 0 0 </r0>
            <r1> 0 0 0 </r1>
            <r2> 0 0 0 </r2>
            <coordinate>  </coordinate>
            <limit> 0 </limit>
        </rdCMC_Joint>
    </defaults>



    <objects>

        <!-- Pelvis -->

        <rdCMC_Joint name="pelvis_tz">
            <wrt_body> -1 </wrt_body>
            <express_body> -1 </express_body>
            <on> true </on>
            <active> true false false </active>
```

```
            <weight> 1.0e5 </weight>
            <kp> 1600.0 </kp>
            <kv>  80.0 </kv>
            <coordinate> pelvis_tz </coordinate>
        </rdCMC_Joint>

        <rdCMC_Joint name="pelvis_tx">
            <wrt_body> -1 </wrt_body>
            <express_body> -1 </express_body>
            <on> true </on>
            <active> true false false </active>
            <weight> 1.0e5 </weight>
            <kp> 1600.0 </kp>
            <kv>  80.0 </kv>
            <coordinate> pelvis_tx </coordinate>
        </rdCMC_Joint>

        <rdCMC_Joint name="pelvis_ty">
            <wrt_body> -1 </wrt_body>
            <express_body> -1 </express_body>
            <on> true </on>
            <active> true false false </active>
            <weight> 1.0e5 </weight>
            <kp> 1600.0 </kp>
            <kv>  80.0 </kv>
            <coordinate> pelvis_ty </coordinate>
        </rdCMC_Joint>

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

    </objects>

</rdCMC_TaskSet>
```

In Example 16-4 above, only generalized coordinates are specialized for the model to follow. 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: <rdCMC_Joint> or <rdCMC_Point>. For a task that tracks a joint coordinate (like all tasks in Example 16-4 above), 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 Chapter 14). 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 16-4 above.

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.

16.5.5 RRA2 Example Files

Example 16-5: XML file for the setup file for RRA2 (e.g., subject01_Setup_RRA2.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMCTool name="subject01_walk1_RRA2">

    <defaults/>
    <model_library> gait23 </model_library>
```

```
    <model_file> subject01_sdfast_adjusted.osim </model_file>
    <replace_actuator_set> true </replace_actuator_set>
    <actuator_set_files> gait2354_RRA2_Actuators.xml
</actuator_set_files>
    <results_directory> Results/ </results_directory>
    <output_precision> 8 </output_precision>
    <initial_time> 0.75 </initial_time>
    <final_time> 1.25 </final_time>
    <maximum_number_of_integrator_steps> 20000
</maximum_number_of_integrator_steps>
    <maximum_integrator_step_size> 0.001
</maximum_integrator_step_size>
    <integrator_error_tolerance> 0.0001
</integrator_error_tolerance>
    <integrator_fine_tolerance> 1e-006 </integrator_fine_tolerance>

    <desired_kinematics_file> subject01_walk1_ik.mot
</desired_kinematics_file>
    <lowpass_cutoff_frequency> 6 </lowpass_cutoff_frequency>
    <task_set_file> gait2354_RRA2_Tasks.xml </task_set_file>
    <constraints_file> gait2354_RRA2_ControlConstraints.xml
</constraints_file>

    <external_loads_file> subject01_walk1_grf.mot
</external_loads_file>
    <external_loads_model_kinematics_file> subject01_walk1_ik.mot
</external_loads_model_kinematics_file>
    <external_loads_body1> calcn_r </external_loads_body1>
    <external_loads_body2> calcn_l </external_loads_body2>
    <lowpass_cutoff_frequency_for_load_kinematics> 6
</lowpass_cutoff_frequency_for_load_kinematics>

    <use_fast_optimization_target> false
</use_fast_optimization_target>
    <optimizer_derivative_dx> 0.0001 </optimizer_derivative_dx>
    <optimizer_convergence_criterion> 1e-006
</optimizer_convergence_criterion>
    <optimizer_max_iterations> 2000 </optimizer_max_iterations>
    <optimizer_print_level> 0 </optimizer_print_level>

    <cmc_time_window> 0.001 </cmc_time_window>
    <use_curvature_filter> false </use_curvature_filter>
    <compute_average_residuals> true </compute_average_residuals>
    <adjust_com_to_reduce_residuals> false
</adjust_com_to_reduce_residuals>

</CMCTool>
```

Example 16-6: XML file for the actuator set file for RRA2 (e.g., gait2354_RRA2_Actuators.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<ActuatorSet name="gait2354_RRA2">

    <defaults>

        <Force name="default">
            <max_force> 10000.000 </max_force>
            <min_force> -10000.000 </min_force>
            <optimal_force> 1000.00000000 </optimal_force>
            <body_A> </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 1.000 0.000 0.000 </direction_A>
            <body_B> </body_B>
            <point_B> 0.000 0.000 0.000 </point_B>
        </Force>

        <Torque name="default">
            <max_force> 1000.000 </max_force>
            <min_force> -1000.000 </min_force>
            <optimal_force> 300.00000000 </optimal_force>
            <body_A> </body_A>
            <direction_A> 1.000 0.000 0.000 </direction_A>
            <body_B> </body_B>
        </Torque>

        <GeneralizedForce name="default">
            <max_force> 1000.000 </max_force>
            <min_force> -1000.000 </min_force>
            <optimal_force> 300.00000000 </optimal_force>
            <coordinate> </coordinate>
        </GeneralizedForce>

    </defaults>

    <objects>

        <!-- Residuals -->

        <Force name="FX">
            <optimal_force> 4.0 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
```

```
            <direction_A> -1.000 0.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Force name="FY">
            <optimal_force> 8.0 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 0.000 -1.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Force name="FZ">
            <optimal_force> 4.0 </optimal_force>
            <body_A> ground </body_A>
            <point_A> 0.000 0.000 0.000 </point_A>
            <direction_A> 0.000 0.000 -1.000 </direction_A>
            <body_B> pelvis </body_B>
            <point_B> -0.07243756 0.00000000 0.00000000 </point_B>
        </Force>

        <Torque name="MX">
            <optimal_force> 2.0 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> -1.000 0.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>

        <Torque name="MY">
            <optimal_force> 2.0 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> 0.000 -1.000 0.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>

        <Torque name="MZ">
            <optimal_force> 2.0 </optimal_force>
            <body_A> ground </body_A>
            <direction_A> 0.000 0.000 -1.000 </direction_A>
            <body_B> pelvis </body_B>
        </Torque>


        <!-- Right Leg -->

        <GeneralizedForce name="hip_flexion_r">
            <coordinate> hip_flexion_r </coordinate>
            <optimal_force> 300.0 </optimal_force>
```

```
        </GeneralizedForce>

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

    </objects>

</ActuatorSet>
```

Example 16-7: XML file for the control constraints file for RRA2 (e.g., gait2354_RRA2_ControlConstraints.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<ControlSet name="gait2354_RRA2">
    <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 16-8: XML file for the tasks file for RRA2 (e.g., gait2354_RRA2_Tasks.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdCMC_TaskSet name="gait2354_RRA2">

    <defaults>
        <rdCMC_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>
            <r0> 0 0 0 </r0>
            <r1> 0 0 0 </r1>
            <r2> 0 0 0 </r2>
            <coordinate>  </coordinate>
            <limit> 0 </limit>
        </rdCMC_Joint>
```

```
    </defaults>

    <objects>

        <!-- Pelvis -->

        <rdCMC_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>
        </rdCMC_Joint>

        <rdCMC_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>
        </rdCMC_Joint>

        <rdCMC_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>
        </rdCMC_Joint>

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

    </objects>

</rdCMC_TaskSet>
```

chapter

17

Computed Muscle Control

17.1 Overview

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.

17.2 How It 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 17-1).

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.

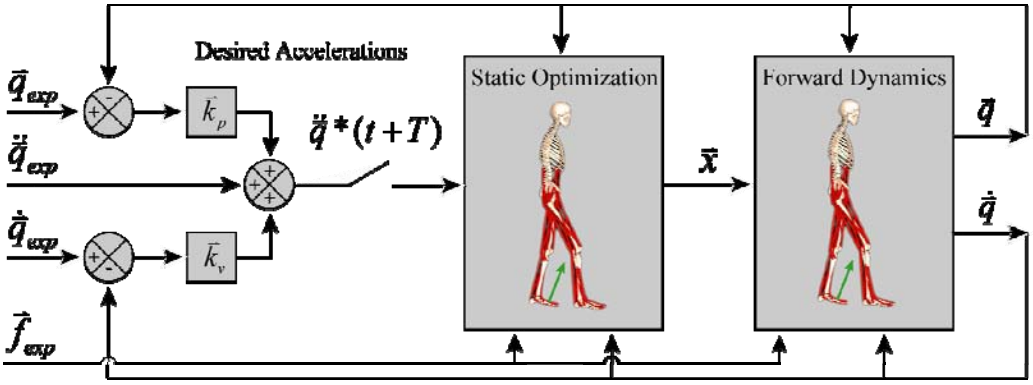


Figure 17-1: Schematic of the Computed Muscle Control Algorithm Applied to Gait [Thelen, D.G. and Anderson, F.C., "Using computed muscle control to generate forward dynamic simulations of human walking from experimental data, J. Biomech., 2006, 39(6):1107-1115]

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:

$$J = \sum_{i=1}^{nx} x_i^2 ; \qquad C_j = \ddot{q}_j^* - \ddot{q}_j, \quad for\ all\ j$$

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 $\vec{x}$ as a function of simulation time. The bounds on the controls $\vec{x}$ 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 (see Chapter 11).

17.3 Inputs and Outputs

The primary inputs to CMC consist of:

1. **OpenSim model** *[.osim and .dll files]* - this should be the SD/Fast model
2. **Desired kinematics** *[.mot or .sto file]* to be tracked
3. **External forces** *[.mot or .sto file]* applied to the model
4. **Tracking tasks** *[.xml file]* specifying which coordinates are to be tracked
5. **Control constraints** *[.xml file]* used to constrain the allowed values of the actuator controls

The subject-specific OpenSim model is often generated by running the Scale Tool (see Chapter 13) followed by the RRA tool (Chapter 16). The desired kinematics are typically obtained by running the IK and RRA tools (Chapters 14 and 16). External forces usually come directly

from force plate measurements, and it is often necessary to preprocess this data to put it in the expected format. Tracking tasks are specified in files that need to be created and edited manually, although existing task files can be duplicated and used as a starting point. Chapter 21 has information concerning motion (*.mot*) and storage (*.sto)* file formats.

The primary outputs of CMC consist of:

1. **Actuator controls** *[.xml file]* (e.g., muscle excitations) computed by CMC that will drive a forward dynamic simulation reproducing the desired kinematics
2. **Model states** *[.sto file]* containing the time histories of all model states that occurred during CMC execution. These states are used as input to a forward simulation to specify the initial conditions for that simulation.
3. **Actuator controls** *[.sto file]* computed by CMC in a format suitable for plotting

17.4 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
```

17.5 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.

17.5.1 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 17-1: XML file for the CMC setup file (e.g., subject01_Setup_CMC.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<CMCTool name="subject01_walk1">


   <!-- OpenSim Model -->
   <model_library> gait23 </model_library>
   <model_file> subject01_sdfast_adjusted.osim </model_file>
   <actuator_set_files> gait2354_CMC_Actuators.xml
           </actuator_set_files>
   <replace_actuator_set> false </replace_actuator_set>


   <!– Times over which to run CMC -->
   <initial_time> 0.80 </initial_time>
   <final_time> 1.20 </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>
    <integrator_fine_tolerance> 5e-008 </integrator_fine_tolerance>


   <!– Analyses and Results -->
   <results_directory> ./ResultsCMC </results_directory>
   <output_precision> 20 </output_precision>
   <AnalysisSet name="Analyses">
      <objects>
         <Kinematics name="Kinematics">
            <step_interval> 10 </step_interval>
         </Kinematics>
         <Actuation name="Actuation">
            <step_interval> 10 </step_interval>
         </Actuation>
      </objects>
   </AnalysisSet>


   <!– Desired Kinematics -->
   <desired_kinematics_file>
      Results/subject01_walk1_RRA2_Kinematics_q.sto
```

```
        </desired_kinematics_file>
   <lowpass_cutoff_frequency> -6 </lowpass_cutoff_frequency>



   <!– External Loads -->
   <external_loads_file> subject01_walk1_grf.mot
          </external_loads_file>
   <external_loads_model_kinematics_file>
      subject01_walk1_ik.mot </external_loads_model_kinematics_file>
   <external_loads_body1> calcn_r </external_loads_body1>
   <external_loads_body2> calcn_l </external_loads_body2>
   <lowpass_cutoff_frequency_for_load_kinematics>
      6 </lowpass_cutoff_frequency_for_load_kinematics>


   <!– Optimization -->
   <use_fast_optimization_target> true
          </use_fast_optimization_target>
   <optimizer_derivative_dx> 1.0e-4 </optimizer_derivative_dx>
   <optimizer_convergence_criterion> 1e-006
          </optimizer_convergence_criterion>
   <optimizer_max_iterations> 2000 </optimizer_max_iterations>
   <optimizer_print_level> 0 </optimizer_print_level>


   <!– CMC -->
   <cmc_time_window> 0.01 </cmc_time_window>
   <use_curvature_filter> true </use_curvature_filter>
   <adjust_com_to_reduce_residuals> false
         </adjust_com_to_reduce_residuals>


   <!– Tracking Tasks & Control Constraints -->
   <task_set_file> gait2354_CMC_Tasks.xml </task_set_file>
   <constraints_file> gait2354_CMC_ControlConstraints.xml
        </constraints_file>

</CMCTool>
```

17.5.1.1 Specifying the Model

Specifying which model to use requires four properties. <model_library> specifies the name of the compiled library (a .dll on Windows systems). <model_file> specifies the name of the .osim file used to construct and initialize the model. <actuator_set_files> is a list of actuators sets, 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_actuator_set>, if set to **true**, indicates that the actuator sets listed in <actuator_set_files> should replace any actuators that are already part of the model.

17.5.1.2 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.

17.5.1.3 Specifying Integrator Settings

The <maximum_number_of_integrator_steps> property indicates the maximum number of steps CMC 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.

17.5.1.4 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.

17.5.1.5 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.

17.5.1.6 Specifying External Loads

The file containing the external loads applied to the model during a simulation is specified using the property <external_loads_file>. The loads and points of application are given in the global frame. The property <external_loads_model_kinematics_file> specifies the appropriate accompanying model kinematics for the external loads. Using this information, the loads are transformed into the local frames of the body segments to which they are applied. If the global position of the model drifts during a simulation, the loads translate with their body segments. The direction (or orientation) of the loads do not change; they are always applied in the same direction with respect to the global frame. Loads may be applied to up to two bodies. <external_loads_body1> specifies the first body, and <external_loads_body2> specifies the second. As with the desired kinematics, the external loads model kinematics can be low-pass filtered using the <lowpass_cutoff_frequency_for_load_kinematics> tags. Again, a negative value is used to specify no filtering. *Note: Across all steps in the OpenSim workflow, it is important to use the same settings for specifying the external loads.*

17.5.1.7 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 (see Section 17.2). <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.

17.5.1.8 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 (see Section 17.2). <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 16). During CMC, this property should be set to **false**.

17.5.1.9 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 directly below.

17.5.2 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.

17.5.2.1 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>.

17.5.2.2 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 18-2: XML file for the CMC task set file (e.g., gait2354_CMC_tasks.xml)

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

<rdCMC_TaskSet name="gait2354_CMC">

   <objects>

      <rdCMC_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>
      </rdCMC_Joint>

      <rdCMC_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>
      </rdCMC_Joint>

   </objects>

</rdCMC_TaskSet>
```

17.5.2.3 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 It Works (Section 17.2) above 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).

17.5.3 Control Constraints File

17.5.3.1 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.

17.5.3.2 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.

17.5.3.3 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 17-3. <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 17-3: XML file for the CMC control constraints file (e.g., 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>
```

17.5.3.4 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 16), 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).

17.5.3.5 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 17-3).

17.5.3.6 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 17-3), it is only necessary to have a one-line entry that specifies the name of the control.

chapter

18

Forward Dynamics

18.1 Overview

Given the controls (e.g., muscle excitations) computed by the Computed Muscle Control (CMC), the Forward Dynamics Tool can drive a forward dynamic simulation. 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.

18.2 How it Works

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.

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 recommend 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.

18.3 Inputs and Outputs

Figure 18-1 shows the required inputs and outputs for the Forward Dynamics Tool. Each is described in more detail in the following sections.

Note: The following file names are examples that can be found in the examples/Gait2354 directory installed with the OpenSim distribution.

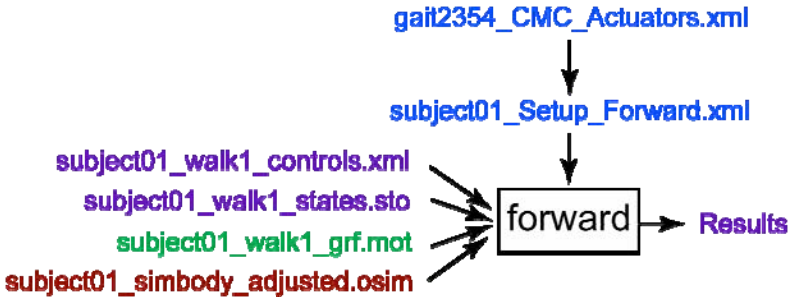


Figure 18-1: Inputs and Outputs of the Forward Dynamics Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

18.3.1 Settings File(s)

The **subject01_Setup_Forward.xml** file is the setup file for the Forward Dynamics Tool. It contains settings, as described in Section 18.7, 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 the CMC chapter (Chapter 17) for more details. These actuators must be included in the forward simulation so that the CMC solution can be reproduced.

18.3.2 Inputs

Four data files are required as input by the Forward Dynamics Tool:

subject01_walk1_controls.xml: Contains the time histories of muscle excitations computed by the CMC tool.

subject01_walk1_states.sto: Contains the time histories of model states, including joint angles, joint speeds, muscle activations, and more. These states were computed by the CMC tool and are used by the Forward Dynamics Tool to set the initial states of the model for forward integration.

subject01_walk1_grf.mot: Motion file containing the measured ground reaction forces that should be applied to the model during simulation

subject01_simbody_adjusted.osim: OpenSim musculoskeletal model scaled to the dimensions of the subject with a torso center of mass that has been adjusted to reduce residuals

18.3.3 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 Section 18.6.

18.4 How to Use the GUI

The Forward Dynamics Tool is accessed by selecting **Tools → Forward Dynamics...** from the OpenSim main menu bar (Figure 18-2). Like all tools, the operations performed by the Forward Dynamics Tool apply to the current model. The name of the current model is shown in bold in the Navigator. See chapters 4 and 6 for information on opening models and making a particular model current.

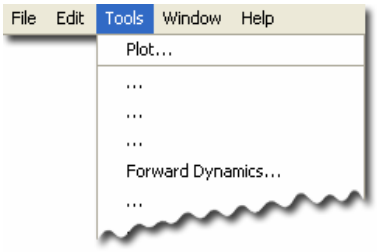


Figure 18-2: Tools Menu. The Forward Dynamics Tool can be accessed from the Tools menu.

The Forward Dynamics Tool is controlled by a window with three tabbed panes (Figure 18-3). The *Main Settings* pane specifies parameters relating to the controls and states that will be input into the model, the time range for the simulation, and the output of the results. The *Actuators and External Loads* pane specifies the actuator set and the

external loads applied to the model during the simulation. The *Integrator Settings* pane specifies integrator step sizes and tolerances used to solve the simulation. The controls and states from the CMC solution are used to drive the forward simulation of the recorded movement.

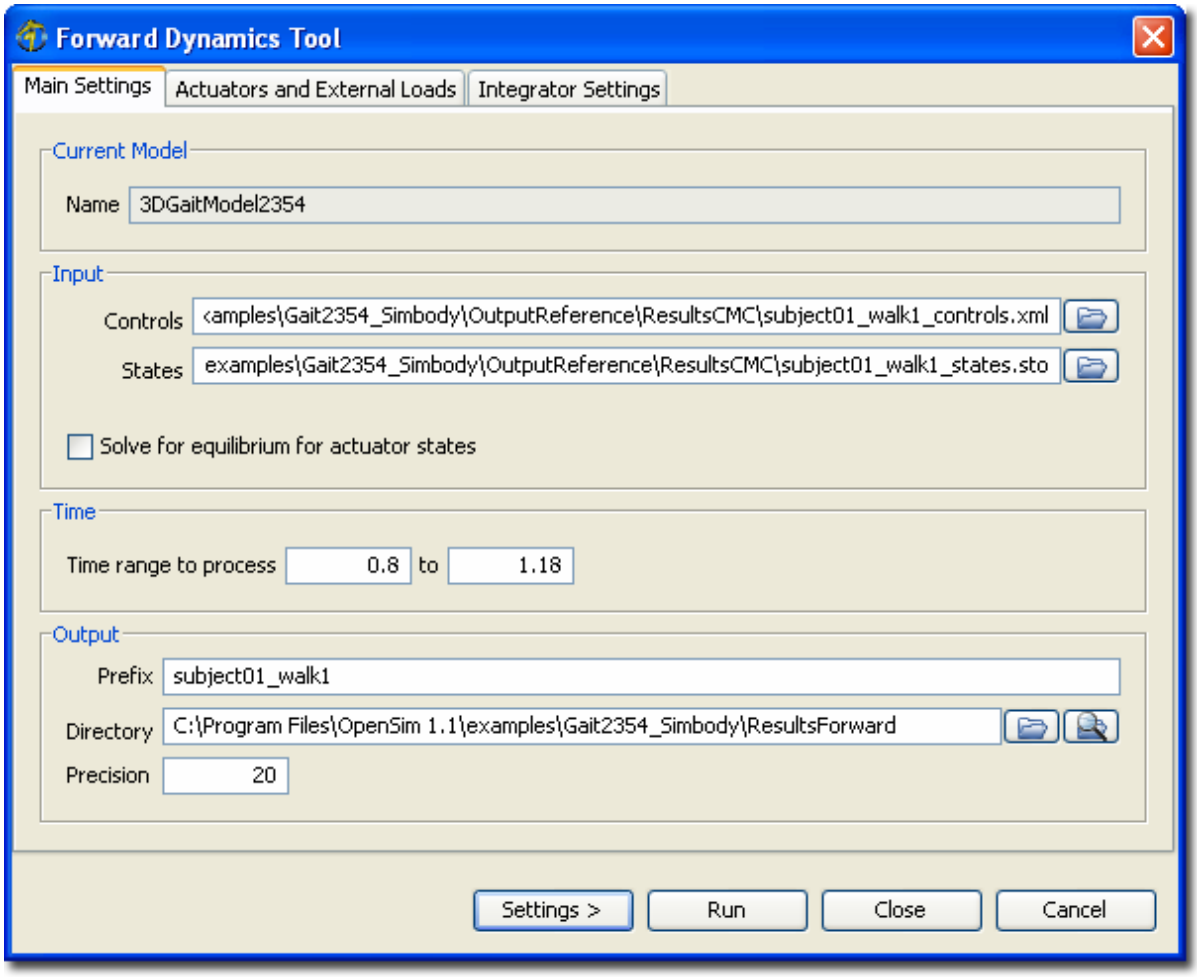


Figure 18-3: Window for the Forward Dynamics Tool

At the bottom of the window are four buttons. 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.

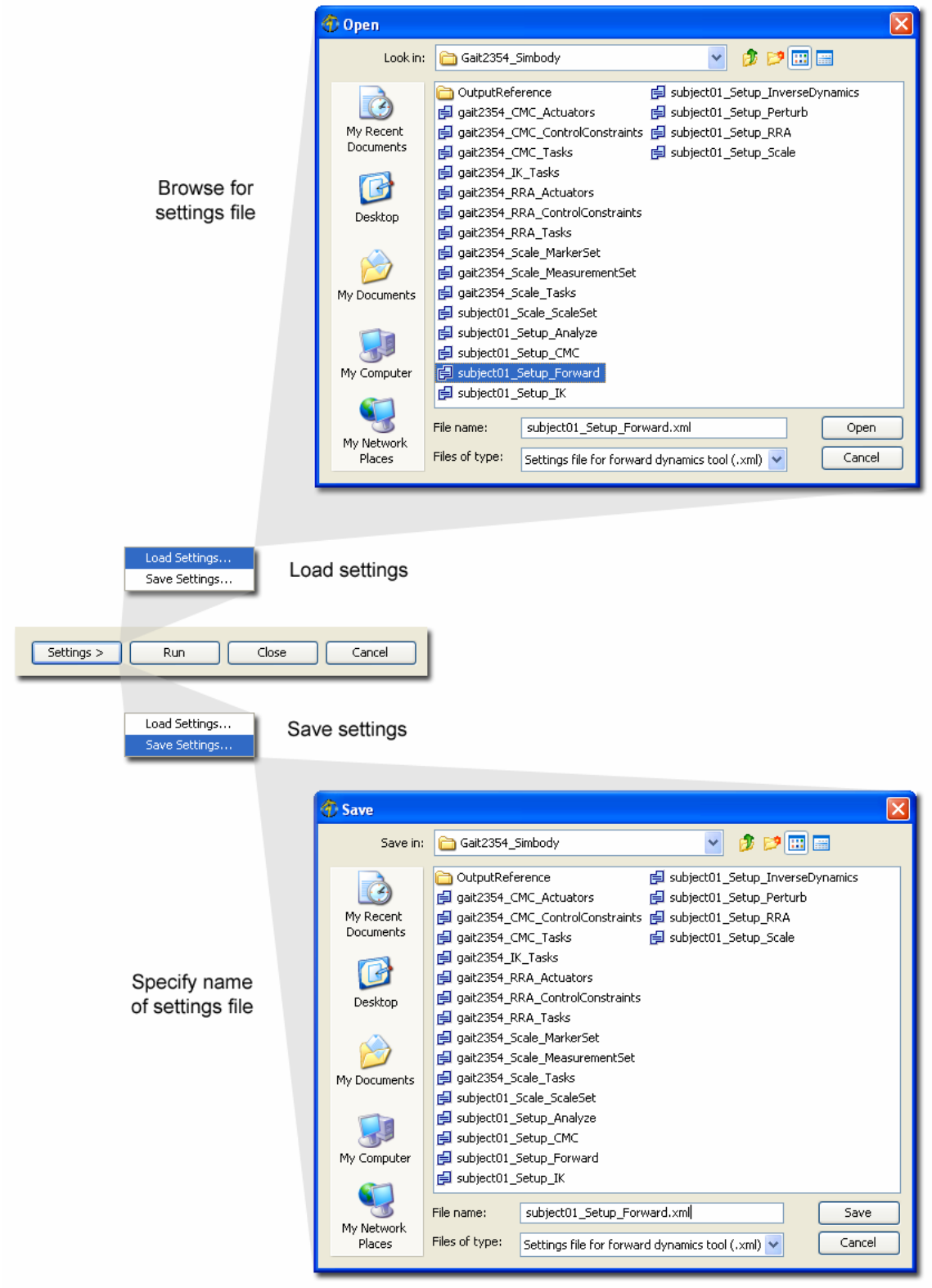


Figure 18-4: Saving and Loading Settings

When the **Settings >** button is clicked, you are presented with the choice of loading or saving settings for the tool (Figure 18-4). 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_Forward_Scale.xml**) and click **Open**. The Forward 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.

18.4.1 Main Settings Pane

The *Main Settings* pane (Figure 18-3) is used to specify parameters relating to the subject data, the generic model, and how the model is to be scaled. The pane is organized into four main sections entitled *Current Model, Input, Time,* and *Output*.

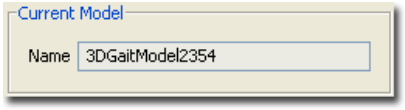


Figure 18-5: Current Model Section

The section for *Current Model* displays uneditable information about the current model that is to be used for analysis by the Forward Dynamics Tool (Figure 18-5).

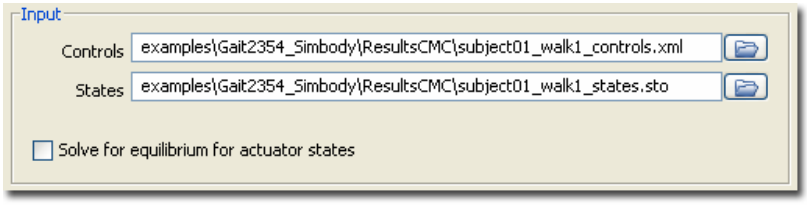


Figure 18-6: Input Section

The section for *Input* displays editable information that allows you to specify the controls and states to be used to run the forward simulation, as discussed in Section 18.3.2. You may use the button to browse for the controls and states files (Figure 18-6).

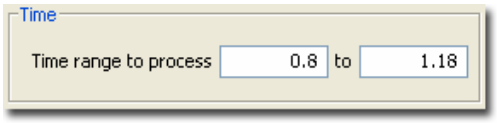


Figure 18-7: Time Section

The section for *Time* displays editable information that allows you to specify the start and end time for the forward simulation (Figure 18-7).

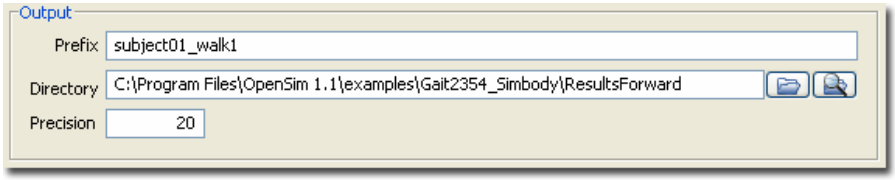


Figure 18-8: Output Section

The section for *Output* displays editable information that allows you to specify the prefix appended to all of the resulting output files, the directory to which the files are 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, and the button to open an Explorer window to the specified directory.

18.4.2 Actuators and External Loads Pane

The *Actuators and External Loads* pane (Figure 18-9) is used to specify parameters relating to the actuators appended to the model and the external loads applied to the model during the forward dynamic simulation. The pane is organized into two main sections entitled *Actuators* and *External Loads*.

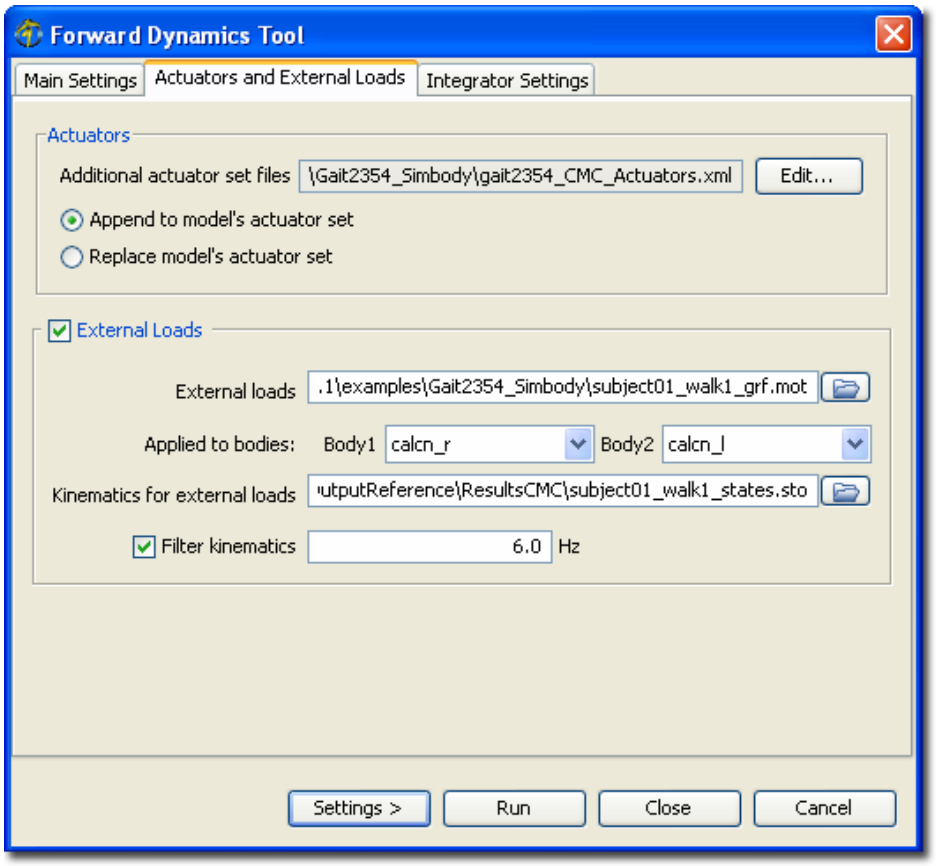


Figure 18-9: Actuators and External Loads Pane

The section for *Actuators* displays editable information that allows you to specify additional actuator set files that specify actuators to supplement the muscles of the model, as discussed in Section 18.3.1. You may use the Edit... button to edit the list of actuator set files describing the actuators to be appended to or replaced in the model (Figure 18-10).

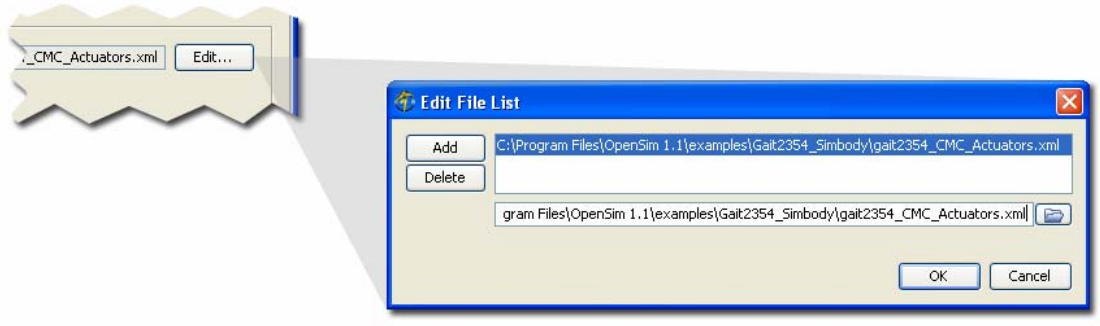


Figure 18-10: Editing the File List of Actuator Sets

The information in the *External Loads* section is optional. If checked, the section displays information so that you can specify the external loads applied to the model, the segments of the model to which the loads are applied, and the corresponding kinematics of the external loads. Additionally, there is an option to filter the kinematics for the external loads by selecting the checkbox next to **Filter kinematics** and entering the filter frequency (Figure 18-9).

18.4.3 Integrator Settings Pane

The *Integrator Settings* pane (Figure 18-11) is used to specify parameters of the numerical integrator used during the simulation. It is recommended that you use the same integration time steps in forward dynamics as were used in CMC (see Section 18.2). To do so, select the checkbox next to **Use time steps from states file**. You can also manually input the integrator settings. Deselect the checkbox next to **Use time steps from states file** and enter appropriate values in the textboxes for **Integrator steps** and **Integrator tolerances**.

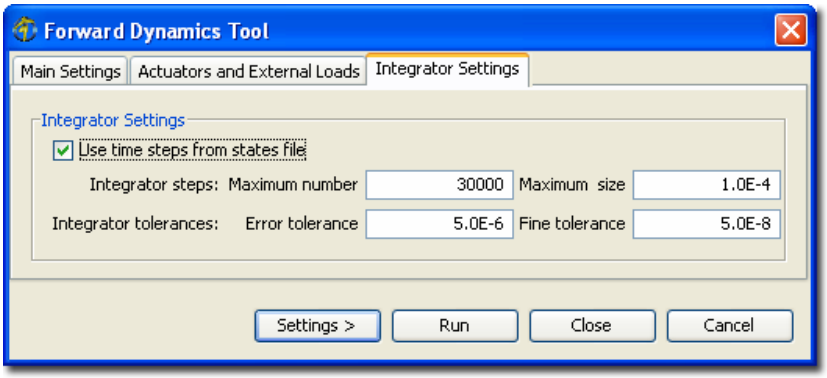


Figure 18-11: Integrator Settings Pane

18.5 Command-line Execution

The Forward Dynamics Tool is run using the command **`forward -S <setup file name>`**, for example,

```
forward -S subject01_Setup_Forward.xml
```

18.6 Analyses

The main purpose for the forward dynamics step, besides validating the CMC results in a basic open-loop forward dynamics system, is to record additional simulation data. Getting additional output data from the Forward Dynamics Tool is done through the use of analyses. These are not unique to the Forward Dyanamics Tool and in fact, analyses can be added to RRA and CMC as well. But since forward runs are significantly faster than CMC, it is more practical to tweak and re-run forward simulations to obtain new data rather than re-run CMC.

The basic analyses of interest are:

- **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.

18.7 Setup File and Properties

These sample XML files are from the examples/gait2354 directory and are part of the OpenSim distribution.

18.7.1 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.

Example 18-1: XML file for the setup file for forward dynamics (e.g., subject01_Setup_Forward.xml)

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

<ForwardTool name="subject01_walk1">

   <!-- OpenSim Model -->
   <model_library> gait23 </model_library>
   <model_file> subject01_sdfast_adjusted.osim </model_file>
   <actuator_set_files> gait2354_CMC_Actuators.xml
          </actuator_set_files>
   <replace_actuator_set> false </replace_actuator_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>
   <integrator_fine_tolerance> 5e-008 </integrator_fine_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 -->
   <initial_states_file> ResultsCMC/subject01_walk1_states.sto
          </initial_states_file>
   <controls_file> ResultsCMC/subject01_walk1_controls.xml
          </controls_file>


   <!-- External Loads -->
   <external_loads_file> subject01_walk1_grf.mot
          </external_loads_file>
   <external_loads_model_kinematics_file> subject01_walk1_ik.mot
          </external_loads_model_kinematics_file>
   <external_loads_body1> calcn_r </external_loads_body1>
   <external_loads_body2> calcn_l </external_loads_body2>
   <lowpass_cutoff_frequency_for_load_kinematics> 6
          </lowpass_cutoff_frequency_for_load_kinematics>

</ForwardTool>
```

18.7.2 Specifying the Model

The model specification for the Forward Dynamics Tool must match that used for CMC. See Chapter 17 on CMC for details on setting these properties.

18.7.3 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 Section 18.2, 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.

18.7.4 Specifying Integrator Settings

As explained in Section 18.2, 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 17 on CMC chapter for how these are used).

18.7.5 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.

18.7.6 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 the initial states and the control values used to drive the forward simulation are those computed during the CMC step. Therefore, 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).

18.7.7 Specifying External Loads

The file containing the external loads applied to the model during a simulation is specified using the property <external_loads_file>. The loads and points of application are given in the global frame. The property <external_loads_model_kinematics_file> specifies the appropriate accompanying model kinematics for the external loads. Using this information, the loads are transformed into the local frames of the body segments to which they are applied. If the global position of the model drifts during a simulation, the loads translate with their body segments. The direction (or orientation) of the loads do not change; they are always applied in the same direction with respect to the global frame. Loads may be applied to up to two bodies. <external_loads_body1> specifies the first body, and <external_loads_body2> specifies the second. As with the desired kinematics, the external loads model kinematics can be low-pass filtered using the <lowpass_cutoff_frequency_for_load_kinematics> tags. Again, a negative value is used to specify no filtering. *Note: Across all steps in the OpenSim workflow, it is important to use the same settings for specifying the external loads.*

chapter

19

Analyzing Simulations

19.1 Overview

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.

19.2 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.

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]

PSEUDOSTATES

Pseudostates are the variables of a model that are not governed by differential equations, and thus are not integrated, but nonetheless depend on the time history of a simulation. In other words, pseudostates cannot be computed from the states. Examples of pseudostates in

OpenSim are the coordinates used to specify locations of spring-and-damper units used to model foot ground contact. To model friction, the locations of the springs in the ground frame need to move as the foot slides on the ground.

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.

19.3 Inputs and Outputs

The Analyze Tool has the same inputs and outputs as the Forward Dynamics Tool. Refer to Section 18.3 for more information about the input and output files.

19.4 How to Use the GUI

The Analyze Tool is accessed by selecting **Tools** → **Analyze...** from the OpenSim main menu bar (Figure 19-1). Like all tools, the operations performed by the Analyze Tool apply to the current model. The name of the current model is shown in bold in the Navigator. See Chapters 4 and 6 for information on opening models and making a particular model current.

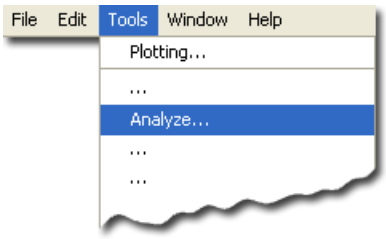


Figure 19-1: Accessing the Analyze Tool from the Tools Menu

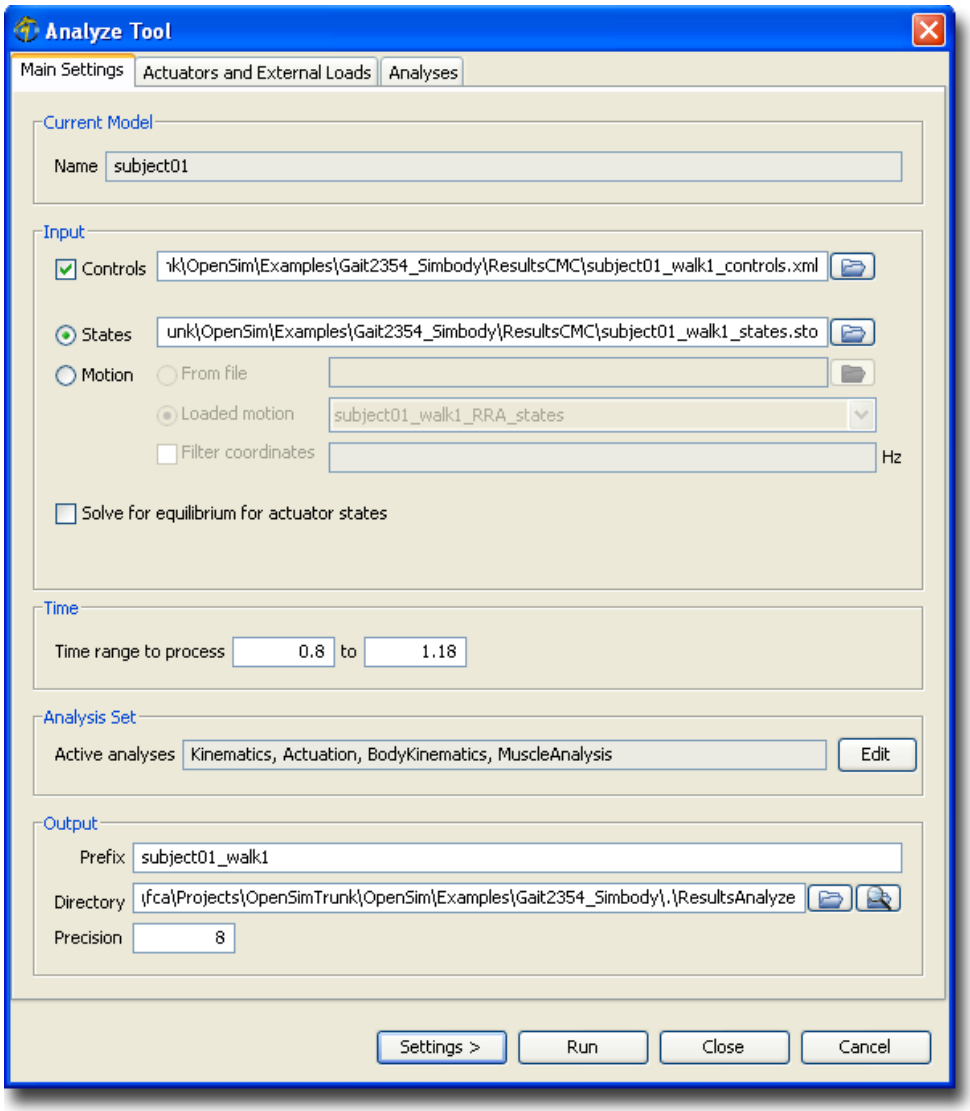


Figure 19-2: Window for the Analyze Tool. The Analyze Tool consists of three panes: *Main Settings*, *Actuators and External Loads*, and *Analyses*. Shown is the *Main Settings* pane.

The Analyze Tool is controlled by a window with two tabbed panes (Figure 19-2). The *Main Settings* pane is used to specify parameters relating to the input, the time range over which the analyses are to be run, which analyses are to be run, and the output. The *Actuators and External Loads* pane is used to specify parameters relating to the external loads applied to the model during analysis and whether or not an additional set of actuators should be added to the model. The *Analyses* pane provides a means of adding, removing, and editing the analyses that will be run.

At the bottom of the window are four buttons. 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.

19.4.1 Main Settings Pane

The *Main Settings* pane (Figure 19-2) is used to specify parameters relating to input, the time range over which analyses are to be performed, which analyses are to be performed, and the output. The pane is organized into five main sections entitled *Current Model*, *Input*, *Time*, *Analysis Set*, and *Output*.

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

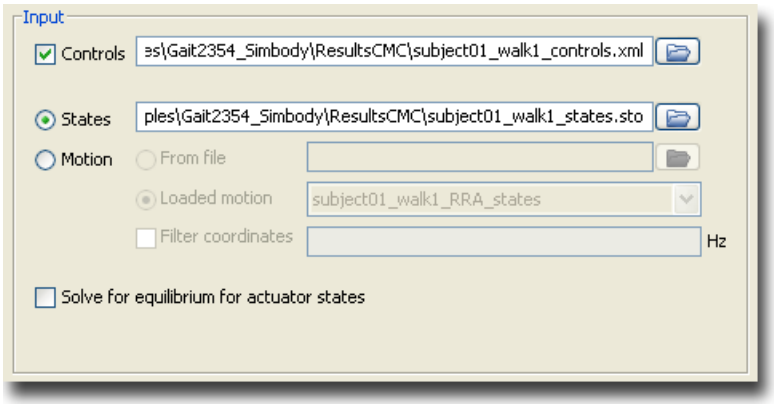


Figure 19-3: Specifying Input from a Simulation

The section for *Input* displays editable information that allows you to specify the controls, states or motion, and whether or not to solve for force equilibrium for the actuator forces (Figure 19-3). If you are running an analysis on an existing simulation, you will most often specify a set of controls and states output by that simulation. You may use the button to browse for the input files.

However, if you would like to run a set of analyses based on experimentally recorded motion, you may use the radio button to select **Motion** as the input type (Figure 19-4). If you choose to run the analyses from **Loaded motion**, you will need to choose a motion from the subject01_walk1_motion drop down list.

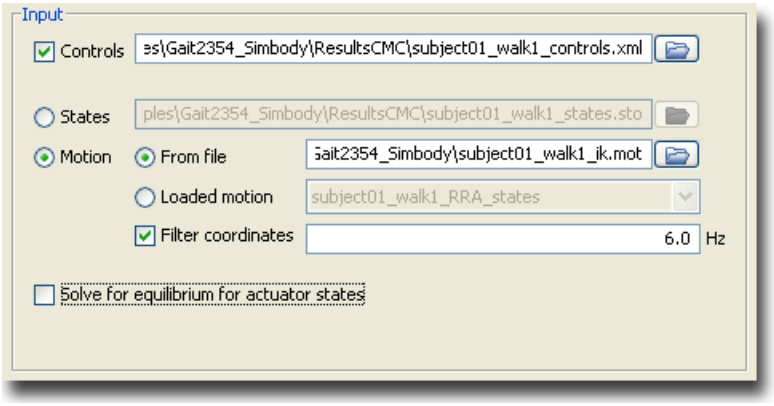


Figure 19-4: Specifying Input from an Experiment

The section for *Time* displays editable information that allows you to specify the start and end time for the analysis (Figure 19-5).

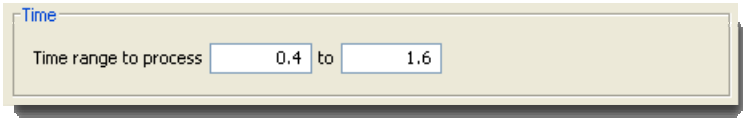


Figure 19-5: Time Section

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 (Figure 19-6). You may use the

button to browse for and specify a directory in which to save the output files, and the button to open an Explorer window to the specified directory.

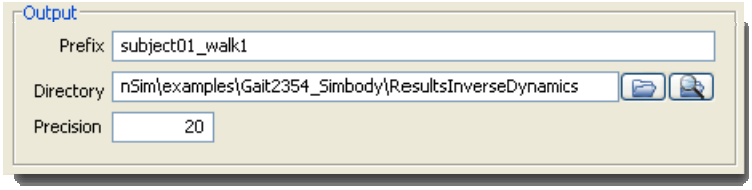


Figure 19-6: Output Section

19.4.2 Actuators and External Loads Pane

The *Actuators and External Loads* pane (Figure 19-7) is used to specify parameters relating to the actuators appended to the model and the external loads applied to the model during the analyses. The pane is organized into two main sections entitled *Actuators* and *External Loads*.

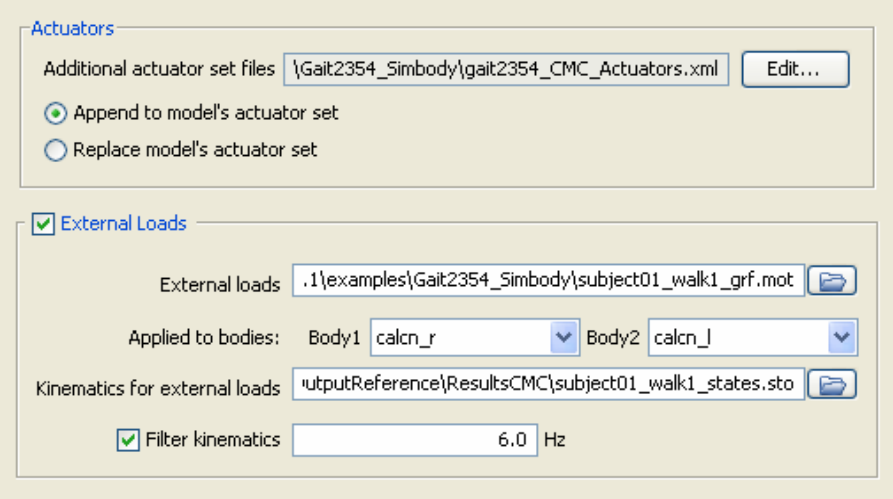


Figure 19-7: Actuators and External Loads Pane

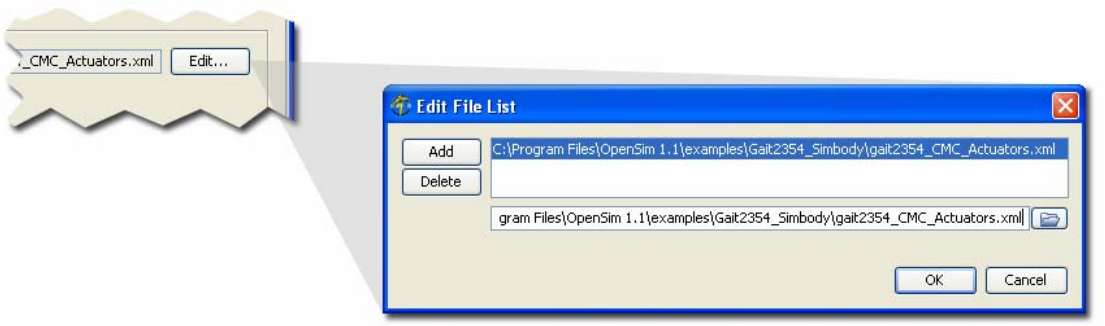


Figure 19-8: Editing the File List of Actuator Sets

The section for *Actuators* displays editable information that allows you to specify additional actuator set files that specify actuators to supplement the muscles of the model. You may use the Edit... button to edit the list of actuator set files describing the actuators to be appended to or replaced in the model (Figure 19-8).

The information in the *External Loads* section is optional. If checked, the section displays information so that you can specify the external loads applied to the model, the bodies of the model to which the loads are applied, and the corresponding kinematics of the external loads. Additionally, there is an option to filter the kinematics for the external loads by selecting the checkbox next to **Filter kinematics** and entering the filter frequency (Figure 19-8).

19.5 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.

chapter

20

Perturbation

20.1 Overview

The Perturbation Tool can be used to compute accelerations induced by the actuators of a model, i.e., the contribution of individual actuators to the measured acceleration. Typically, induced accelerations of generalized coordinates (e.g., knee angle) or body positions (e.g., hip center) may be desired, and the actuators consist of the muscles and any additional actuators (e.g., residual actuators, reserve actuators, etc.).

20.2 How it Works

[Adapted from Liu, *et al.*, "Muscles that support the body also modulate forward progression during walking," J. Biomech., 2006, 39(14):2623-2630]

Suppose you wanted to measure the contribution at current time t_i of a particular muscle m to the fore-aft acceleration of the body mass center, $\ddot{x}_m(t_i)$. The expression for this is

$$\ddot{x}_m(t_i) = \frac{\partial \ddot{x}}{\partial F_m}\vec{F}_m = \frac{\ddot{x}(F_m + \Delta F_m, t_i) - \ddot{x}(F_m, t_i)}{\Delta F_m} F_m$$

where F_m is the force generated by muscle m, ΔF_m is a constant perturbation to F_m and $\ddot{x}$ is the

acceleration of the center of mass in the fore-aft direction.

Acceleration depends linearly on force, so the forward difference expression for $\frac{\partial \ddot{x}}{\partial F_m}$ above is exact (as would be a backwards difference formula). However, aside from the contribution of muscles and other actuators to the body accelerations, there are system reaction forces (such as ground reaction forces) that also affect the kinematics of the body. The computation of the induced acceleration must take these into account as well. This requires some sort of a foot-ground contact model that can model the forces applied to the model in response to changing model accelerations.

The Perturbation Tool uses linear and torsional springs as a simple contact model. These work by applying forces in response to deviations in the foot's position from where it should be (see Section 20.2.1 for more details on the springs). Since these forces arise in response to position (and velocity) changes rather than instantaneous acceleration changes, instantaneous ground contact forces canot be computed. So instead, the Perturbation Tool works by perturbing a muscle's force, simulating forward in time by a short interval Δt, and observing the resulting change in position of the model's mass center.

Assuming that the acceleration induced by a muscle over this short interval is constant, the observed unperturbed ($x(F_m, t_i + \Delta t)$) and perturbed ($x(F_m + \Delta F_m, t_i + \Delta t)$) positions can be related to the accelerations using the following two equations:

$$x(F_m, t_i + \Delta t) \approx x(F_m, t_i) + \dot{x}(F_m, t_i)\Delta t + \tfrac{1}{2}\ddot{x}(F_m, t_i)\Delta t^2$$

$$x(F_m + \Delta F_m, t_i + \Delta t) \approx x(F_m + \Delta F_m, t_i) + \dot{x}(F_m + \Delta F_m, t_i)\Delta t + \tfrac{1}{2}\ddot{x}(F_m + \Delta F_m, t_i)\Delta t^2$$

Since positions and velocities cannot change instantaneously in response to a force perturbation, we see that

$$x(F_m + \Delta F_m, t_i) = x(F_m, t_i)$$
$$\dot{x}(F_m + \Delta F_m, t_i) = \dot{x}(F_m, t_i)$$

and therefore subtracting the earlier equations gives

$$\ddot{x}(F_m + \Delta F_m, t_i) - \ddot{x}(F_m, t_i) \approx 2 \cdot \frac{x(F_m + \Delta F_m, t_i + \Delta t) - x(F_m, t_i + \Delta t)}{\Delta t^2}$$

which substituted into our forward difference equation gives

$$\ddot{x}_m(t_i) \approx 2 \cdot \frac{x(F_m + \Delta F_m, t_i + \Delta t) - x(F_m, t_i + \Delta t)}{\Delta t^2 \Delta F_m} F_m$$

This is the formula we use to compute induced accelerations.

20.2.1 Springs

Each foot has a linear and torsional spring added to it which attempts to pull it towards its location and orientation in the unperturbed simulation.

The linear (translational) spring connects a point in the foot's frame to a point in world space. This point is the center of pressure. Its foot frame location is computed using the unperturbed trajectory. As the actuator forces are perturbed, the foot's trajectory will be perturbed, and this spring will try to pull the foot back to its unperturbed position.

The torsional spring tries to restore a foot's orientation to its orientation in the unperturbed simulation.

Since these springs are trying to model a rudimentary foot-contact model, they are not on at all times. The linear spring is primarily active between heel strike and toe off. The torsional spring is primarily active between foot flat and heel off. In order to make the transition between the springs smooth, a smooth falloff function is used to modulate their effects, rather than discontinuously turning them on/off at the right time. Due to this smoothing, the springs are technically on at all times, but outside of the intervals mentioned above, they are heavily attenuated so that their force is close to negligible. The falloff equation is defined as follows:

$$1.0 - \frac{1.0}{1.0 + e^{-(t-t0)/\tau}}$$

where t = time, t0 = location of the midpoint of the step, and τ = rise and fall time constant.

20.3 Superposition

When analyzing a simulation, it is important to test whether the sum of the accelerations induced by each muscle and by gravity is equal to 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).

20.4 Inputs and Outputs

The primary inputs to the Perturbation Tool consist of:

1. **OpenSim model** *[.osim and .dll files]* (same as input to CMC)

2. **External forces** *[.mot or .sto 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** *[.xml file]* output from CMC, used to set initial states
5. **Model kinematics** *[.sto files]* - these are the generalized positions and speeds computed in CMC (requires use of Kinematics analysis)

It is important for the inputs to the Perturbation Tool to match the inputs/outputs of CMC (e.g., using the same input model and external forces as CMC, and the controls, states, and kinematics output from CMC).

The output of the Perturbation Tool consists of the output(s) from the individual analyses you choose.

20.5 Command-line Execution

The Perturbation Tool is run using the command **`perturb -S <setup file name>`**, for example,

```
perturb -S subject01_Setup_Perturb.xml
```

20.6 Setup Files and Properties

20.6.1 Setup File

Example 20-1 shows the example of a setup file for perturbation.

20.6.1.1 Specifying an Execution Name

The properties for the Perturbation Tool are enclosed inside the opening and closing tags <PerturbationTool> and </PerturbationTool>. The name attribute name="subject01_walk1" specifies the execution name. The names of results files generated will be prefixed with this name.

Example 20-1: XML file for the setup file for perturbation (e.g., subject01_Setup_Perturb.xml)

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

<PerturbationTool name="subject01_walk1">

  <!-- Model specification -->
  <model_library> gait23 </model_library>
  <model_file> subject01_sdfast_adjusted.osim </model_file>
  <replace_actuator_set> false </replace_actuator_set>
  <actuator_set_files> gait2354_CMC_Actuators.xml
          </actuator_set_files>


  <!-- Time interval -->
  <initial_time> 0.80 </initial_time>
  <final_time> 1.18 </final_time>


  <!-- Perturbation parameters -->
  <perturbation_time_window> 0.03 </perturbation_time_window>
  <perturbation_time_increment> 0.01 </perturbation_time_increment>
  <perturbation_size> 1 </perturbation_size>


  <!-- Integrator settings -->
  <maximum_number_of_integrator_steps> 20000
          </maximum_number_of_integrator_steps>
  <maximum_integrator_step_size> 0.0001
          </maximum_integrator_step_size>
  <integrator_error_tolerance> 1e-007 </integrator_error_tolerance>
  <integrator_fine_tolerance> 1e-010 </integrator_fine_tolerance>


  <!-- Spring parameters -->
  <scaling_rise_time> 0.001 </scaling_rise_time>
  <corrective_spring_linear_stiffness> 5e+006 5e+006 5e+006
          </corrective_spring_linear_stiffness>
  <corrective_spring_linear_damping> 1500 1500 1500
        </corrective_spring_linear_damping>
  <corrective_spring_torsional_stiffness> 100000 100000 100000
        </corrective_spring_torsional_stiffness>
  <corrective_spring_torsional_damping> 1000 1000 1000
        </corrective_spring_torsional_damping>
```

```
<!-- Additional needed inputs -->
<controls_file> ResultsCMC/subject01_walk1_controls.xml
       </controls_file>
<states_file> ResultsCMC/subject01_walk1_states.sto </states_file>
<coordinates_file> ResultsCMC/subject01_walk1_Kinematics_q.sto
       </coordinates_file>
<speeds_file> ResultsCMC/subject01_walk1_Kinematics_u.sto
       </speeds_file>
<cop_file> subject01_walk1_grf.mot </cop_file>


<!-- Feet bodies -->
<right_foot_body> calcn_r </right_foot_body>
<left_foot_body> calcn_l </left_foot_body>


<!-- Event times for springs -->
<r_heel_strike> 0.63 </r_heel_strike>
<r_foot_flat> 0.75 </r_foot_flat>
<r_heel_off> 1.2 </r_heel_off>
<r_toe_off> 1.4 </r_toe_off>
<l_heel_strike> 1.28 </l_heel_strike>
<l_foot_flat> 1.38 </l_foot_flat>
<l_heel_off> 1.8 </l_heel_off>
<l_toe_off> 2 </l_toe_off>


<!-- External loads -->
<external_loads_file> subject01_walk1_grf.mot
        </external_loads_file>
<external_loads_model_kinematics_file> subject01_walk1_ik.mot
        </external_loads_model_kinematics_file>
<external_loads_body1> calcn_r </external_loads_body1>
<external_loads_body2> calcn_l </external_loads_body2>
<lowpass_cutoff_frequency_for_load_kinematics> 6
        </lowpass_cutoff_frequency_for_load_kinematics>


<!-- Output -->
<results_directory> ./ResultsPerturb </results_directory>
<output_precision> 12 </output_precision>
<AnalysisSet name="Analyses">
  <objects>
    <Kinematics name="Kinematics">
      <on> true </on>
      <step_interval> 10 </step_interval>
      <in_degrees> true </in_degrees>
      <coordinates> knee_angle_l knee_angle_r </coordinates>
    </Kinematics>
```

```
      <BodyKinematics name="BodyKinematics">
        <on> true </on>
        <step_interval> 10 </step_interval>
        <in_degrees> true </in_degrees>
        <bodies> center_of_mass </bodies>
      </BodyKinematics>
    </objects>
  </AnalysisSet>

</PerturbationTool>
```

20.6.1.2 Specifying the Model

The model specification for the Perturbation Tool must match that used for CMC. See the Chapter 17 on CMC for details on setting these properties.

20.6.1.3 Specifying Initial and Final Times

The properties <initial_time> and <final_time> specify the time interval over which the Perturbation Tool is to be run. The final time needs to be no greater than the last time available from CMC minus the perturbation time window (see Section 20.6.1.4).

20.6.1.4 Specifying Perturbation Parameters

The following properties are used by the Perturbation Tool: <perturbation_time_window>, <perturbation_time_increment>, <perturbation_size>.

<perturbation_time_window> specifies the Δt time interval value. As described in the How it Works section (Section 20.2), for each time t_i at which induced acceleration values are desired, the Perturbation Tool integrates forward from t_i to $t_i + \Delta t$ to compute the induced accelerations (by observing the changes in positions).

Induced accelerations are computed at discrete time values between the user-specified initial and final times. Starting at the initial time, the Perturbation Tool increments time by the value specified by <perturbation_time_increment> to get subsequent times at which to compute induced accelerations.

<perturbation_size> determines the amount by which an actuator's force should be perturbed (i.e., the ΔF from Section 20.2 on How it Works).

20.6.1.5 Specifying Integrator Settings

Just like in RRA, CMC, and forward dynamics, the integrator setting values are used to determine the adaptive time step sizes of the integrator (see Section 17.5.1.3).

20.6.1.6 Specifying Spring Parameters

Spings are used to model a rudimentary foot-contact model and can be described using the following parameters: <scaling_rise_time>, <corrective_spring_linear_stiffness>, <corrective_spring_linear_damping>, <corrective_spring_torsional_stiffness>, <corrective_spring_torsional_damping>.

<scaling_rise_time> is a parameter used in forming the smooth fall-off function for attenuating the spring's effects. It corresponds to the τ parameter shown in Section 20.2.1.

The remaining properties give the stiffness and damping for the linear and torsional springs. Each of these properties lists 3 values, allowing these properties to have different values along the X, Y, and Z axes. The currently chosen values have been carefully picked, and should typically be kept as they are.

20.6.1.7 Specifying Additional Necessary Inputs

The <controls_file> is the file name of the resulting controls from CMC. This is used to drive forward the unperturbed simulation, so the unperturbed forces can be determined. <states_file> is the name of the file containing the states output from CMC, and is used to set initial conditions before each perturbation time window integration is started.

<coordinates_file> and <speeds_file> are outputs from CMC. The <Kinematics> analysis must be added to CMC in order to get these _q and _u output files.

<cop_file> should be set to the file containing the ground reaction forces. In Example 20-1 above, a separate file is used, although the IK solution .mot file should also work since it contains the ground reaction data as well. This file is used to determine the center of pressure locations which are used to connect the linear springs on each foot.

20.6.1.8 Specifying Names of the Feet Bodies

<right_foot_body> and <left_foot_body> are properties that specify the names of the right and left feet in the model, respectively. The linear and torsional springs will be applied to these bodies. Each foot will get its own linear spring and its own torsional spring. **Calcn_r** and

calcn_l are the default values, so we could have omitted these properties from the file in Example 20-1.

20.6.1.9 Specifying Spring Event Times

As mentioned in Section 20.2 on How it Works, the linear springs are on between heel strike and toe off, and the torsional springs are on between foot flat and heel off. The following properties let you specify these event times:

```
<r_heel_strike>, <l_heel_strike>
<r_foot_flat>, <l_foot_flat>
<r_heel_off>, <l_heel_off>
<r_toe_off>, <l_toe_off>
```

For an individual foot, the events heel_strike, foot_flat, heel_off, and toe_off should be sequential events for a single occurrence of the foot hitting the floor. However, it does not matter whether the right contact or the left contact occurs first.

20.6.1.10 Specifying External Loads

The file containing the external loads applied to the model during a simulation is specified using the property <external_loads_file>. The loads and points of application are given in the global frame. The property <external_loads_model_kinematics_file> specifies the appropriate accompanying model kinematics for the external loads. Using this information, the loads are transformed into the local frames of the body segments to which they are applied. If the global position of the model drifts during a simulation, the loads translate with their body segments. The direction (or orientation) of the loads do not change; they are always applied in the same direction with respect to the global frame. Loads may be applied to up to two bodies. <external_loads_body1> specifies the first body, and <external_loads_body2> specifies the second. As with the desired kinematics, the external loads model kinematics can be low-pass filtered using the <lowpass_cutoff_frequency_for_load_kinematics> tags. Again, a negative value is used to specify no filtering. *Note: Across all steps in the OpenSim workflow, it is important to use the same settings for specifying the external loads.*

20.6.1.11 Specifying Induced Accelerations to Output

As usual, <results_directory> specifies the directory where results should be written.

<output_precision> specifies how many decimal places should be used when writing results.

You can use the <AnalysisSet> tags to specify which induced accelerations you wish to measure. Common analyses used in this case would be <Kinematics> and <BodyKinematics>. Note that Kinematics has a <coordinates> property, which allows you to select exactly which coordinates you care about. The keyword **all** can be used here to indicate all coordinates. This is also the default. BodyKinematics has the <bodies> property, which similarly allows specific bodies to be singled out for measurement. The keyword **all** indicates all bodies. The special body name **center_of_mass** is used to refer to the overall model's center of mass (combining all of the model's bodies).

Each measured coordinate or body will have three output files written. These are **dAdF, deltaA,** and **perturbed**.

dAdF records the induced acceleration per unit force, e.g.,

$$2 \cdot \frac{x(F_m + \Delta F_m, t_i + \Delta t) - x(F_m, t_i + \Delta t)}{\Delta t^2 \Delta F_m}$$

deltaA records the induced acceleration, e.g.,

$$2 \cdot \frac{x(F_m + \Delta F_m, t_i + \Delta t) - x(F_m, t_i + \Delta t)}{\Delta t^2 \Delta F_m} F_m$$

perturbed records the perturbed coordinate/body values, e.g.,

$$x(F_m + \Delta F_m, t_i + \Delta t)$$

Note that all file names are prefixed using the execution name (see Section 20.6.1.1) and the coordinate/body name, and all are suffixed with the Δt values and the ΔF values.

chapter

21

Preparing Your Data

21.1 Overview

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.

21.1.1 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 3×3 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:

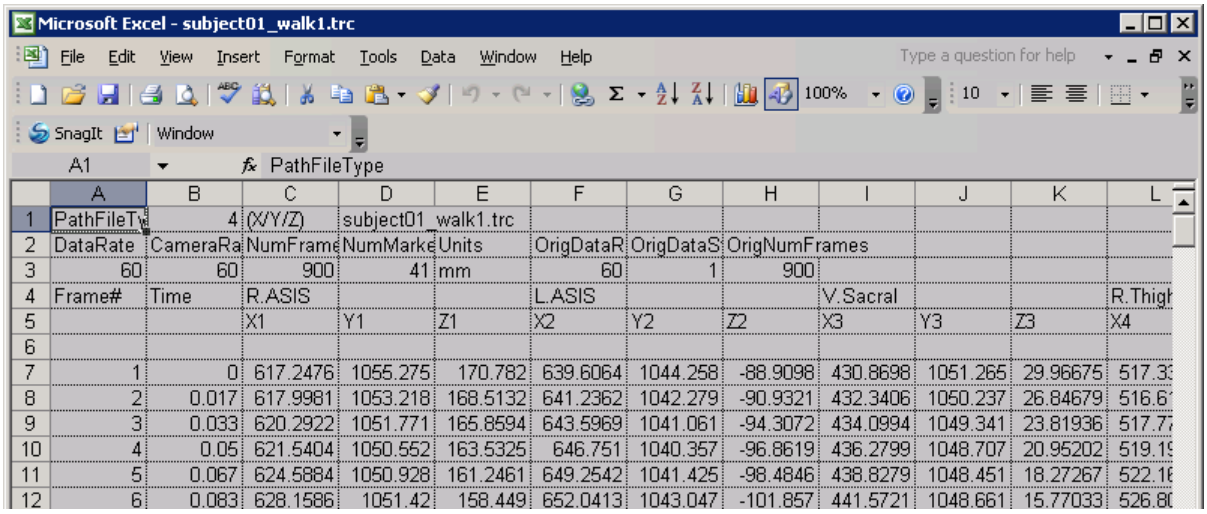
$$ {}^{model}P = {}^{model}_{lab}R * {}^{lab}P $$

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.

21.2 File Formats

21.2.1 Marker (.trc) Files

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 directory, which is part of the OpenSim distribution. A fragment of this file is shown in Figure 21-1 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 21-1: .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 in Figure 21-1 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.

21.2.2 Motion (.mot) Files

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 assumed to be **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. The names of these 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 directory, which is part of the OpenSim distribution.

21.2.3 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/OutputReference/ResultsRRA directory, which is part of the OpenSim distribution.

21.3 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 directory, which is part of the OpenSim distribution.

In addition, the chapter on scaling (Chapter 13) 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.

21.4 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.

21.5 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 directory, which is part of the OpenSim distribution.

The first row below the header in the .mot file must contain the following column headings, in this order:

time, ground_force_vx, ground_force_vy, ground_force_vz,
ground_force_px, ground_force_py, ground_force_pz,
ground_force_vx, ground_force_vy, ground_force_vz,
ground_force_px, ground_force_py, ground_force_pz,
ground_torque_x, ground_torque_y, ground_torque_z,
ground_torque_x, ground_torque_y, ground_torque_z

All rows below this line contain the corresponding data in each column. All data (except for the time column, column 1) must be specified in the model coordinate system. 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 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) in the model coordinate system. 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 in the model coordinate system. The last three columns represent the analogous quantities for the left foot.

21.6 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.