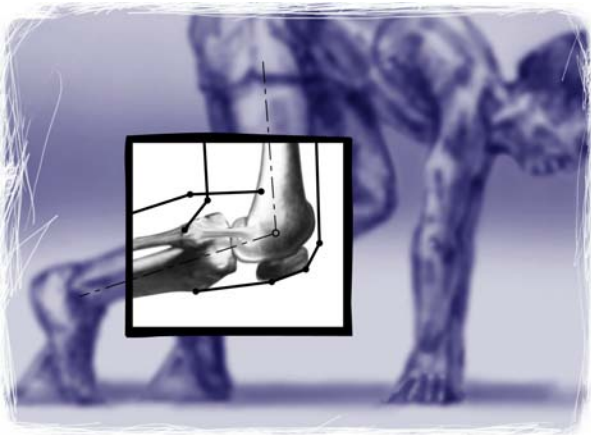


# **SIMM 4.0**

for Windows®

**Software for Interactive  
Musculoskeletal Modeling**

**User's Manual**



---

**SIMM 4.0**  
for Windows®

**User's Manual**

June 2004

Part # SPC-40001

**Important Notice**

The information in this manual is subject to change without notice and should not be construed as a commitment by MusculoGraphics, Inc.

The software described in this manual is furnished under a license agreement and may be used or copied only in accordance with the terms of such agreement.

Copyright © 1992-2004 MusculoGraphics, Inc.—a division of Motion Analysis Corporation.

All rights reserved. This manual may not be reproduced in whole or in part without the expressed permission of MusculoGraphics, Inc.

MusculoGraphics, Inc.

A division of Motion Analysis Corporation

3617 Westwind Blvd.

Santa Rosa, CA 95403

Phone: +1 (707) 579-6500

Fax: +1 (707) 526-0629

E-mail: [info@musculographics.com](mailto:info@musculographics.com)

<http://www.musculographics.com> and

<http://www.motionanalysis.com>

**Trademarks**

Adobe Postscript is a registered trademark of Adobe Systems Incorporated. Windows is a registered trademark of Microsoft Corp. QuickTime is a registered trademark of Apple Computer, Inc.

**Credits**

*Program Creation & Documentation:*

Peter Loan

Scott Delp

Kenny Smith

Krystyne Blaikie

*Cover Illustration:*

Idd Delp

**Technical Support**

+1 (773) 769-6152

[support@musculographics.com](mailto:support@musculographics.com)

**Acknowledgements**

Development of the initial version of SIMM was funded by the Rehabilitation Research & Development Center, Veterans' Affairs Medical Center, Palo Alto, CA.

We would also like to thank Frans C.T. van der Helm, Ph.D., of the Delft University of Technology, for his contribution of muscle-wrapping algorithms.

---

# 1

# Introduction

---

## 1.1 SIMM - What it is

SIMM (Software for Interactive Musculoskeletal Modeling) is a software system that enables you to create and analyze graphics-based models of the musculoskeletal system. In SIMM, a musculoskeletal model consists of a set of bones that are connected by joints. Muscle-tendon actuators and ligaments span the joints. The muscles and ligaments develop force, thus generating moments about the joints.

SIMM allows you to analyze and test a musculoskeletal model by calculating the moment arms and lengths of the muscles and ligaments. Given muscle activations, the forces and joint moments (muscle force multiplied by moment arm) that each muscle generates can be computed for any body position. By manipulating a model on the computer graphics system, you can quickly explore the effects of changing musculoskeletal geometry and other model parameters.

Since the software can be used to study many different musculoskeletal structures, it can enhance the productivity of investigators working on diverse problems in biomechanics. SIMM provides a framework that organizes the parameters of a model and allows people to work together on a modeling project. The moving, three-dimensional images of anatomical structures that you can create are extremely valuable when developing a model and when communicating the results of an analysis.

## 1.2 Applications

SIMM has a wide variety of applications. Here are a few examples.

**Biomechanics researchers** are using SIMM to create models of the human elbow, wrist, jaw, and other anatomical structures. These models can be altered according to particular surgical procedures to study how the surgical alterations affect muscle function. SIMM can also be used to analyze and display the mechanics of injuries.

**Neuroscientists** are using SIMM to study how the central nervous system controls movement. For example, muscle activation patterns determined from electromyographic recordings can be used to estimate muscle forces and joint moments generated during a task. The computed joint moments can then be compared to experimentally recorded moments.

**Medical students** and residents can use models created with SIMM to study musculoskeletal anatomy and function. In addition to visualizing anatomical structures, students gain an appreciation for the interplay of muscle architecture and joint geometry.

**Kinesiologists** who record and analyze the motion of persons with movement disabilities can use SIMM to create three-dimensional animations of a person's movement. Movements, such as walking, can be quantitatively compared to normal movement to gain insight into the causes of movement deformities. Motion can also be analyzed in the context of optimizing athletic performance.

**Human factors engineers** who need to account for muscle strengths when designing products or work stations can use SIMM to study how posture effects muscle

strength. Limits on joint ranges of motion can also be taken into account.

**Biologists** interested in animal movement can create models to quantify limb function. Investigating movement strategies in other species can provide insights needed to design machines that move.

**Computer scientists** who develop models of the human body for virtual environments can use SIMM to create the models and compare them with biomechanical data for verification.

**Animators** can use SIMM to develop realistic representations of human and animal movements. World objects can be added to provide a context for the animation.

## 1.3 Design Goals

We established four goals in designing and implementing the software. Specifically, it should:

- (i) be *general* enough so that a wide variety of musculoskeletal structures can be modeled,
- (ii) provide *realistic* models of muscle and tendon, and allow accurate specification of joint kinematics,
- (iii) provide an *interactive*, graphics-based environment so the model can be visualized, altered, analyzed, and tested efficiently, and
- (iv) be *extensible* so that new features, such as a more complex muscle-tendon model, can easily be added to the software.

## 1.4 The SIMM Environment

SIMM allows you to load one or more musculoskeletal models by reading sets of input files. Once loaded, a model can be acted upon by a number of editing and analysis tools (Figure 1-1). Each tool is contained within its own window and has a distinct function, such as controlling the viewing of the models (Model Viewer), altering the muscles (Muscle Editor), and making plots for analysis (Plot Maker).

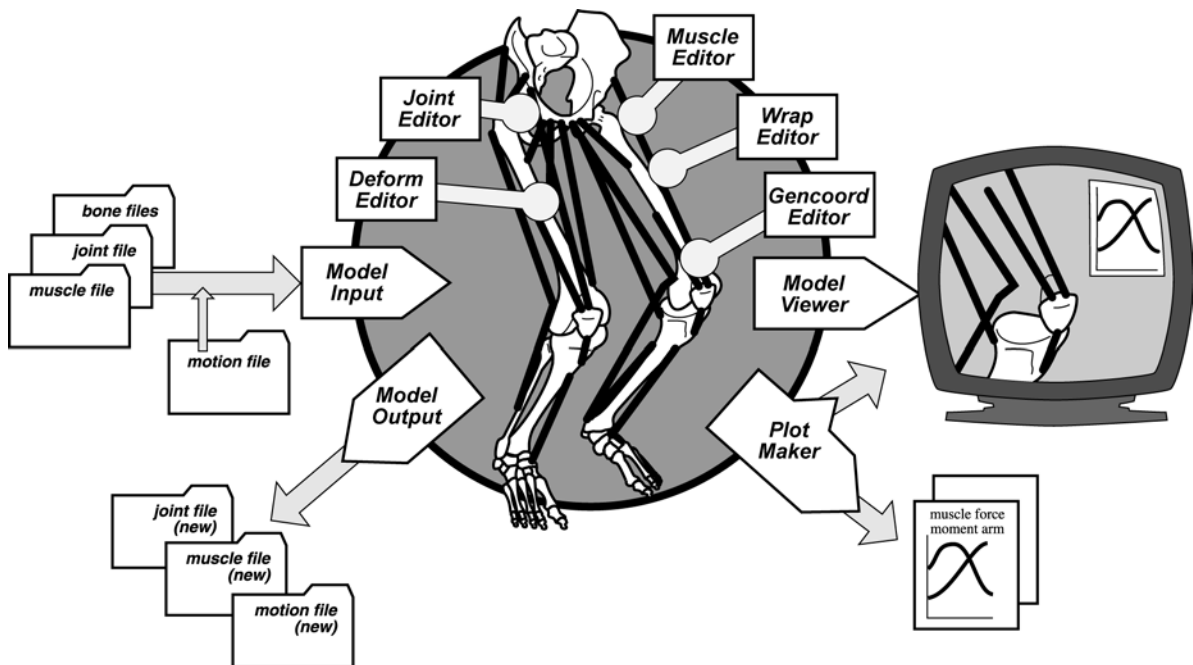


Figure 1-1. Structure of SIMM. Input files describing the bone surfaces (bone files), joint kinematics (joint file), and muscle-tendon parameters (muscle file) are read in to make a musculoskeletal model (motion files can also be read in to animate the model). A model can be altered using the Joint Editor, Muscle Editor, Gencoord Editor, Wrap Editor, Deform Editor, and Model Viewer. Information is extracted from the model by making plots, movies, or by exporting edited joint, muscle, and motion files.

A musculoskeletal model is specified with three types of input files. The *bone* files contain lists of the polygons representing the bone surfaces. The *joint* file specifies the kinematics of each joint. Finally, the *muscle* file contains a list of coordinates that describe the line of action of each muscle-tendon actuator, and the parameters needed to compute muscle force. Ligaments can also be defined in the muscle file. SIMM scans these input files and creates a data structure that represents the musculoskeletal model. Each of these types of files is described in detail in Chapter 3, Input Files.

## 1.5 About the Manual

This manual describes how to use SIMM to develop and analyze musculoskeletal models. Chapter 2, Tools, describes the user interface and the seven software tools. Chapter 3, Input Files, details how to define a musculoskeletal model. Chapter 4, Norm, describes the utility program that you use to preprocess bone files before loading them into SIMM. Appendix A contains some useful tips and caveats on using SIMM. Appendix B describes how to customize SIMM to better fit your needs and preferences. We recommended that you read both of the appendices before using SIMM to make your own musculoskeletal models.

## 1.6 SIMM Tutorials

SIMM comes with a sample leg model built into the software, and includes several tutorials that will introduce you to the basic capabilities of SIMM. The tutorials are located in the Help menu, accessible from the SIMM

menu bar. There are four tutorials: Viewing a Musculoskeletal Model, Plotting Muscle Properties, Joint Editing, and Gait Analysis.

A more complete version of the demo leg model is provided with the full version of SIMM. You can use this model for a musculoskeletal analysis of the lower extremity, or as a template for building your own model. A detailed description of the lower extremity model and its development is beyond the scope of this manual. For more details on the model and on using SIMM to create new musculoskeletal models, consult the following documents:

Delp, S. L., Loan, J. P., "A Computational Framework for Simulating and Analyzing Human and Animal Movement," *IEEE Computing in Science and Engineering*, vol. 2, 2000, pp. 46-55.

Delp, S. L., Loan, J. P., "A Graphics-Based Software System to Develop and Analyze Models of Musculoskeletal Structures," *Computers in Biology and Medicine*, vol. 25, No. 1, 1995, pp. 21-34.

Delp, S. L., "Surgery Simulation: A Computer Graphics System to Design and Analyze Musculoskeletal Reconstructions of the Lower Limb," Ph.D. Dissertation, Stanford University, 1990.



---

# 2 Tools

---

## 2.1 Introduction

SIMM has thirteen tools that enable you to create, modify, and analyze your musculoskeletal models. The tools are: Model Viewer, Plot Maker, Muscle Editor, Joint Editor, Gencoord Editor, Segment Editor, Bone Editor, Constraint Editor, Wrap Editor, Deform Editor, Motion Editor, Marker Editor, and Plot Viewer. Also, if you have the Dynamics Pipeline Module or FIT Module options to SIMM, there is a Dynamics Tool for interfacing SIMM to dynamic simulations. This chapter gives a detailed description of each tool.

The tools use *menus*, *forms*, *sliders*, and *toggle buttons* to give you access to a variety of information. Menus are used to make selections (*e.g.*, choosing a muscle). Forms are used to enter information (*e.g.*, entering a title). Sliders let you change the value of something (*e.g.*, a generalized coordinate) smoothly. Toggle buttons are used to control the state (yes/no or on/off) of a parameter.

### 2.1.1 Definitions

*menu* A *menu* is a vertical list of options or commands, each enclosed in a box. You can select a menu option by clicking the left mouse button in its box. When you select a menu item, it is highlighted, and the box appears to be pressed into the menu. Menu items such as **save muscles**

remain highlighted only as long as it takes to perform the necessary action. In other instances, a menu item is used to change modes within a tool, in order to perform a more complicated action. In this case, the menu item is highlighted until you complete the action by pressing a key or mouse button. For example, the Joint Editor has a menu item called **add point** which lets you add a control point to a joint function. When you select this option, the box is highlighted to indicate that you are in **add point** mode. The Joint Editor will remain in this mode until you add the desired point by clicking the left mouse button in the appropriate location. The method of cancelling this type of command can vary, depending on the particular command.

Some menu items display pop-up submenus when they are selected. Menu items that do this can be differentiated from other menu items by an angle bracket (greater-than sign) on the right side of the item box. For example, the model selector, which displays a pop-up menu of the models, is labeled **model**.

*muscle menu*

A *muscle menu* is a special type of menu. Muscle menus group muscles according to group names assigned in the muscle file (e.g., hip flexors, see Section 3.4, Muscle Files). When you click the left mouse button on one of the items in a muscle menu, the corresponding muscle is selected and the box to the left of the muscle name is highlighted. When you click the left mouse button on that item again, the muscle is unselected. Several of the tools display the muscle menus, but selecting a muscle in one tool's menu does not mean that it is selected in another tool's menu. For example, the Model Viewer and Plot Maker each display separate copies of the same muscle menus. When you select a muscle in the Model Viewer, it changes the model view, but does not select the muscle

within the Plot Maker. If you want to generate a plot curve for that muscle, you must also select it from the muscle menu in the Plot Maker.

To make it easier to select and unselect large groups of muscles, the title bar of each muscle menu also has a special function. If one or more muscles in a menu are unselected, then clicking the left mouse button in the title bar of that menu causes all of the muscles to be selected. If all of the muscles are already selected, then when you click in the title bar they become unselected. The last item in each muscle menu is labeled <<done>>. Selecting <<done>> with the left mouse button “puts away” the menu so that it is no longer displayed. If any of the muscles in the menu are selected when you put away the menu, *they remain selected even though you can no longer see the menu*. When you are finished selecting muscles from a particular muscle menu, you do not have to put the menu away by selecting <<done>>. Doing so is merely a convenience so that you can display other muscle menus without taking up too much space in the tool window.

*form* A *form* consists of a group of fields that can be edited. A field is a boxed region that contains a number or other text. You can position the cursor anywhere in a field by clicking the left mouse button at the desired position. A text cursor appears at this location to indicate that any text you type in will be inserted at this location. You can use the left and right arrow keys to move the text cursor one character at a time. You can highlight a section of the text field that you want to delete or change by “sweeping” the cursor over the region. To do this, press the left mouse button at one end of the region, move the cursor to the other end, then release the left mouse button. You can adjust the size of this highlighted region by holding down

the `Shift` key while pressing the left and right arrow keys to move one character at a time. If some text is highlighted when you type in a character, the highlighted text is deleted and the typed character is put in its place. To delete the character just before the text cursor, press the `BackSpace` or `Delete` key.

Some form fields can contain only numbers (e.g., the `gencoord` form in the `Model Viewer`). When editing a numerical field, you can enter only numbers, a decimal point, and a minus sign. You cannot enter exponents.

When you are done editing the text in a field, press the `Enter` key, or press the left mouse button while the cursor is outside the field (and not over any other menu item or form field). If the field is a numerical one, SIMM will check to make sure that the number you have entered is valid. If it is not, SIMM will restore the previous value. If the field is empty when you press return, SIMM will restore the text that was in the field before you began editing it. You can also press the `Tab` key when you are done editing a field. This will move the text cursor to the next field in the form and select all of the text within it.

*slider* *Sliders* in SIMM are just like sliders or scroll bars in other programs. Each has a thumb which you can click on with the left mouse button and drag along the shaft of the slider. You can also click the left mouse button on either of the two arrow buttons to change the value more slowly. If you click in the slider shaft away from the thumb, the thumb will jump to where you clicked.

*toggle button* *Toggle buttons* control the on/off (yes/no) state of various parameters. Clicking the left mouse button in the box toggles the state between the two values. When the state is on (yes), the toggle button is yellow. When the state is off (no), the button is gray.

*selector menu*

All tools have a selector menu at the top of the tool window. The text above this menu shows the current model (the model that the tool is currently working on), and the current plot (the plot that the tool is working on). Below this text are “selectors” for changing the current model or plot. You can change the current model with the model selector, which is the box labeled **model**. Clicking the left mouse button in this box pops up a menu of the models you have created. You can then choose the one you want by releasing the button while the cursor is over its name. *You only need to use the model selector if you have more than one model at a time loaded into SIMM.* You can select the current plot in a similar fashion using the plot selector (labeled **plot**). If a tool does not operate on specific models (*e.g.*, Plot Viewer), the model selector is not displayed in the header region. Likewise, if the tool does not operate on specific plots (*e.g.*, Model Viewer), the plot selector is not displayed. Also in this region of every tool is a help button at the far right. Click the left mouse button on it to open a window with some helpful text.

In general, the interface for each of the tools is confined to the tool window. That is, the tool window must be the active window for you to perform any of that tool’s commands. However, three of the tools (Model Viewer, Muscle Editor, and Plot Viewer) can recognize key and mouse button presses when they occur in the model or plot windows. For example, to select muscle points in the Muscle Editor, you press the `Space Bar` and click the left mouse button *in the model window*. This feature allows you to operate on the models and plots without having to open the tool window. In some cases the same command can be performed in the tool window as well, in other cases the command can only be performed in the model or plot window. See the relevant tool section in this chapter for more details.

## 2.2 Opening Files

To locate joint, muscle, motion, and plot files and load them into SIMM, you use the Windows<sup>®</sup> file browser. Select *File > Open...* to browse for files to load. By default the file browser displays all files with SIMM file name extensions (.jnt, .msl, .mot, and .plt). You can use the *Files of type* popup menu at the bottom of the browser to limit the display to files of a single type to make it easier to locate a specific file.

As a convenience it is possible to select and open multiple files at once using the file browser. This can be useful when working with multiple models, motions or plots, or when applying different muscle files to a single model.

### *joint and muscle files*

To create a model, the segments, joints, generalized coordinates, and kinematic functions must be defined in a joint file. If muscles are to be part of the model, they must be defined in separate muscle file (see Chapter 3, Input Files).

If you select a single joint file to open, SIMM will create a new model based on that file. If a muscle file name is specified within the joint file, that muscle file will be loaded as well. If no muscle file name is specified, the model will be loaded without any muscles, unless a muscle file was also selected in the file browser. If a muscle file was selected in addition to a joint file in the browser, SIMM will load the model from the joint file and add the muscles from the muscle file to it.

When selecting both joint and muscle files from the file browser, remember that any muscle file specified within the joint file will override a muscle file selected in the file browser.

If you select multiple joint files in the browser, SIMM will open each one and create a new model from it. However, when selecting more than one joint file you cannot select muscle files as well because SIMM cannot determine which muscle files correspond to which joint files. If you want to load multiple joint files at once, you must specify the corresponding muscle file names within each joint file.

*bone files*

You cannot open bone files directly in SIMM. Instead, bones are referenced from within a joint file and loaded automatically when the joint file is loaded. To locate bone files for bones referenced in a joint file, SIMM searches for a directory named *bones* in the same directory as the joint file. If a *bones* directory exists, SIMM first looks in it for each of the model's bone files. If the *bones* directory does not exist or a bone file cannot be found in the *bones* directory, then SIMM searches in the *bones* directory in the SIMM resources directory (see Appendix B, SIMM Resources, for more details).

*motion files*

Whenever you select a motion file in the file browser, SIMM will load the motion and attempt to apply it to every open model in SIMM (including any models you may have selected at the same time in the file browser). It does this by matching the parameter names in the motion file with the names of the generalized coordinates and muscles of each open model. SIMM will automatically link the motion to those models with matching names. Once a motion has been linked to a model, it behaves much like a true generalized coordinate. That is, you can animate the model display by pressing the keys specified in the motion file, or moving the motion slider in the Model Viewer (see Section 2.4, Model Viewer). You can also use the motion variable as the independent variable when making plots (see Section 2.5, Plot Maker).

*plot files* When you select a plot file in the file browser, the way that SIMM processes it depends on which SIMM window was topmost when you selected *File > Open...* If a plot window was topmost, SIMM will read the plot file and add the curves to the selected plot. If some other window was topmost, SIMM will create a new plot window and add the curves to it. In this way multiple plots from separate files can be combined into a single plot if desired.

## 2.3 Saving Files

When saving joint, muscle, motion, and plot files to disk, you use the Windows<sup>®</sup> file browser to specify the name and location of the file to write. Use the browser to navigate to the directory where you want the file to go, then either type the name of the file to create or select an existing file to be replaced.

When saving files to disk, SIMM does not preserve any comments that may have been present in the original input file. Be careful to avoid overwriting any existing SIMM files that contain useful comments.

*joint and muscle files* To save a model's joints or muscles, either the model's window must be the active (*i.e.*, topmost) window, or if a tool window is active, it must be set to that model. To save the model's segments, joints, generalized coordinates, and kinematic functions, select *File > Save Joints...* To save the model's muscles, select *File > Save Muscles...* Use the Windows<sup>®</sup> file browser to specify the name and location of the joint or muscle file to be written. If you do not specify a *.jnt* or *.msl* file name extension for the file, SIMM will add the appropriate one to the file name you specify.



*motion files* To create a motion file for a model, either the model's window must be the active (*i.e.* topmost) window, or if a tool window is active, it must be set to that model. Select *File > New Motion...* and use the Windows<sup>®</sup> file browser to specify the name and location of the motion file to be created. SIMM will automatically add a *.mot* extension to the file name for you if necessary. When you click “OK” in the file browser, the new motion file will be created with a motion header, but without any motion frames. To add motion frames to the new file you must repeatedly position the model by adjusting its generalized coordinate values, then select *File > Add Motion Frame*. Each time you select *Add Motion Frame* from the *File* menu, a new row of information containing the model's current generalized coordinates will be appended to the motion file.

When SIMM first creates a motion file, it gives the motion a default name and sets the `datarows` variable to 200 (see Section 3.5, Motion Files). After saving your motion to a file, you may want to change its name or the value of `datarows`. If the file contains fewer than 200 rows of data, SIMM will print a warning when it reads the file, but it will still load the motion correctly. Thus you can check the motion right after creating it by loading the file back into SIMM by selecting *File > Open...*

There is currently no way to edit frames of an existing motion and save the results to a file. The Motion Editor allows you to crop motions and save them to a file, but you cannot modify individual frames of data. To accomplish this task, you must use the Model Viewer to step through the motion frame by frame, altering the model configuration where desired, and writing each frame to a file with the *Add Motion Frame* command.

*plot files* To save the contents of a plot window to a file, either the plot window must be the active (*i.e.*, topmost) window, or

if a tool window is active it must be set to that plot. To save the plot, select *File > Save Plot...* and use the Windows<sup>®</sup> file browser to specify the name and location of the plot file to be written. SIMM can write plot files in two formats: tab-delimited columns of data, and Postscript. If the file name you specify has a *.ps* extension then SIMM will write the plot in Postscript format which can be sent directly to a Postscript printer. Otherwise SIMM will write the plot data in tab-delimited columns and add a *.plt* extension to the file name if necessary. Tab-delimited plot files are designed to be easy to read or import into word processing or spreadsheet programs.

Note: SIMM cannot read plot files in Postscript format. Plots that you intend to load back into SIMM later should be saved in the tab-delimited *.plt* format.

## 2.4 Model Viewer

The Model Viewer allows you to change the views of the musculoskeletal models. You can rotate, scale, and translate a model into any viewing perspective, as well as choose which muscles to display on the bones. There are also commands to display shadows of the body segments, and start or stop a continuous animation of a model (Figure 2-1). Some of the viewing commands that are available from within the Model Viewer window can also be performed by pressing keys while the model window is the active (*i.e.*, topmost) window. These key sequences serve as short cuts so that you can change the view of a model without having to open the Model Viewer window. These *model window viewing commands* are described in Section 2.4.6.

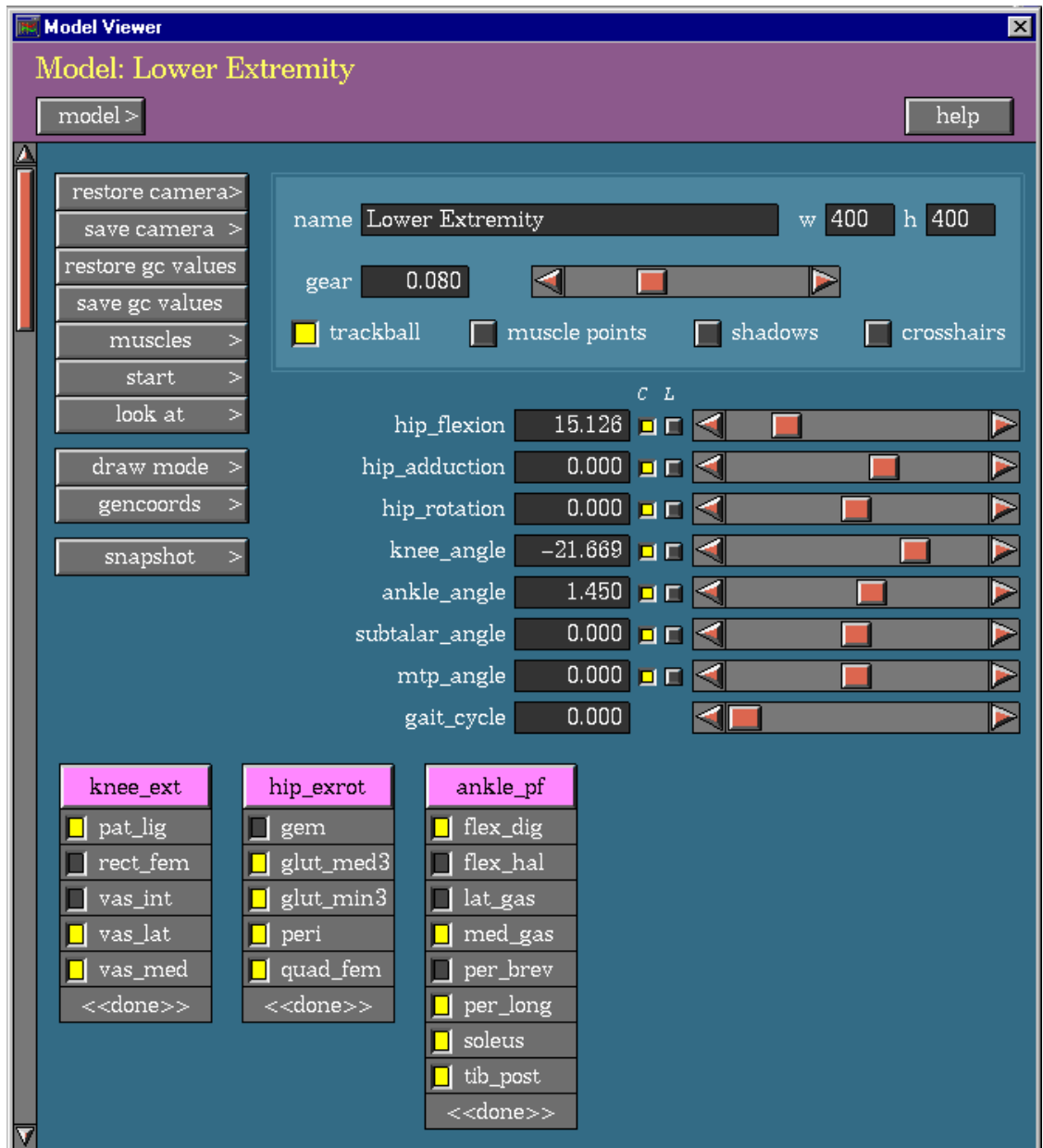


Figure 2-1. Model Viewer window

### 2.4.1 Selector Menu



The model selector lets you select which model you will change with the Model Viewer. The menus and forms in the Model Viewer window always correspond to the current model (setting the current model does not affect the use of model window viewing commands). If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Model Viewer. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.4.2 Command Menu



This command restores the camera view of the current model to one that you had previously saved. When you select this command a pop-up menu of the five buffers is displayed. Selecting one will restore the camera view to the one saved in that buffer. See the **save camera** command for directions on saving a camera view.



This command saves the current camera view of the model to one of five buffers. When you select this command a pop-up menu of the five view buffers is displayed. Selecting one will save the current model view to that buffer. See the **restore camera** command for directions on how to restore the camera once you have saved it.



This command restores the gencoords to their default values. If there are closed loops in the model and the inverse kinematics solver is on, SIMM will try to satisfy the loop constraints and close all the loops in the model. If the IK solver needed to change gencoord default values in order

to close the loops, a window will pop up with some options for handling this new default configuration.

**save gc values** >

This command saves the current gencoord values as their default values. Once they are saved, you can restore the model to this configuration at any time, using the **restore gc values** command.

**muscles** >

This command lets you choose which muscles are displayed on the current model. When you select this item, a pop-up menu of the muscle groups is displayed. When you choose a muscle group from this menu, it will appear at the bottom of the Model Viewer window. If the chosen muscle group is already displayed at the bottom of the window, then selecting it will turn it off, so that it is no longer displayed. As you select muscles from the menus at the bottom of the tool window, the model window is redrawn to show the new muscle selections.

**start** >

This command lets you continuously loop through motions without having to press keys or mouse buttons. When you select it, a pop-up menu is displayed containing a list of the motions that have been linked to the current model. When you select one of the motions, the model is continuously animated according to that motion data (and the button name is changed to **stop**). This will continue until you click on **stop** to stop it. Because this procedure repeatedly draws the model in the model window, it can slow down the computer, making it difficult to execute commands in other windows. So although you can start an animation and then open other tools and make plots, for example, this is not advised. You should also avoid changing the model view while the animation is proceeding. If you need to change the view of the model, stop the animation first, then restart it after positioning the model appropriately.

**look at**

This command lets you change the body segment that is fixed to ground, for display purposes only. The *origin* of this segment will not move as you flex the joints in your model, but it is allowed to rotate about its origin as the joints move.

Note to Dynamics Pipeline users: The **look at** command does not affect the ground segment used for calculating dynamics. Regardless of which segment is treated as ground for display purposes, the dynamics ground segment will always be the fixed segment in the SIMM model (see Section 3.3.2, Fixed Segment, for more details).

**drawmode**

This command gives you control over the display of the body segments and world objects. When you select this command, a pop-up menu of the body segments is displayed. After selecting one of them (or one of the four groups of objects at the bottom of the menu), roll-off to the right to select one of drawing styles. The six styles are: *gouraud shaded* (filled polygons, smooth shading), *flat shaded* (filled polygons, flat shading), *solid fill* (filled polygons, no shading), *outlined* (white polygons with highlighted edges), *wireframe* (hollow polygons), and *none* (no display). Furthermore, if body segment groups are defined in the model, then the **drawmode** pop-up menu will be organized by segment groups. This makes it easy to quickly change the draw mode of several body segments at a time. For example, if you are working with a multi-limbed model, you may want to group the body segments by limb. In doing so you could quickly change the drawmode of an entire limb without having to change the drawmode of each individual segment in the limb.

Note: There is another method for changing the draw-modes of bones and world objects. If you right-click on a bone or world object in the model window, SIMM will

display a pop-up menu. Using this menu, you can change the object's drawmode and material, as well as turn on and off the display of the segment axes and object's normal vectors. You can also set the camera to "look at" that segment, as if you had used the **look at** command.

**gencoords** >

This command allows you to control which gencoord sliders are visible in the Model Viewer window. If your model contains a large number of gencoords, it can be tedious to quickly locate an individual gencoord slider amongst all the others. When you select the **gencoord** command, a pop-up menu of the gencoords is displayed. Selecting items in this menu toggles the visibility of the gencoord's slider in the Model Viewer. Furthermore, if gencoord groups are defined in the model, then the gencoord menu will be organized by these groups. This makes it easy to quickly toggle the visibility of several gencoord sliders at a time.

Note: when a model is loaded, SIMM checks to see how many gencoords are in the model. If there are more than 50 gencoords in the model then SIMM hides all gencoord sliders by default to avoid filling the Model Viewer window with too much information.

**snapshot** >

The snapshot command allows you to easily capture still images and animations from the SIMM model window in TIFF image format. When you select the **snapshot** command, a pop-up menu is displayed. The menu contains the following items:

*choose snapshot file...*

This menu item displays a file browser dialog that allows you to specify the name and location of the next snapshot image to be created by SIMM. By default SIMM will create snapshots starting with the filename *snapshot\_0001.tif*. Each subsequent snapshot file will have the numeric portion of the filename incremented by

one (e.g., *snapshot\_0002.tif*, *snapshot\_0003.tif*, etc.). When you select *choose snapshot filename...* to specify a different snapshot filename, SIMM examines the filename you enter to see if it contains a number. If so, SIMM uses that number as its snapshot counter. Finally, SIMM will make sure that the name of each snapshot file it creates ends with *.tif*. This identifies the file to other programs as a TIFF image file.

*include alpha channel*

This menu item allows you to specify whether SIMM should include an alpha channel in the snapshot images it creates. By default SIMM creates a 24-bit RGB image without an alpha channel. If this menu item is selected, SIMM will create a 32-bit RGBA image that includes z-depth values as an alpha mask. By including an alpha channel, SIMM snapshot images can be combined with other images of 3D scenes using third-party compositing software.

*take snapshot*    *F12*

This menu item creates a single snapshot image of the current model window. If a file already exists with the current snapshot file name it is automatically overwritten (so be careful). The function key *F12* will also take a snapshot of the active model window.

*auto-snapshot*

This menu item puts SIMM in auto-snapshot mode. In this mode, SIMM will create a snapshot image each time the current model window is refreshed. One way to create an animation of a model spinning 360 degrees would be to enable auto-snapshot mode, disable trackball mode, select the model window, then press the *Control* key and click and hold the middle mouse button in the model window until the model has rotated a full 360 degrees. Once you are finished, be sure to select **auto-snapshot** again to disable auto-snapshot mode. SIMM helps you to remember by playing a snapshot sound each time a snapshot file is created.



In addition to these menu items, if the current model has any motions linked to it, then these motions will be listed at the bottom of the **snapshot** pop-up menu. Selecting one of these motions will cause the motion to be played back *starting at the current motion value* and using the current gear. A snapshot image will be created for each frame of motion until the end of the motion. To abort this process, you can click the **start/stop** command or select the motion's menu item in the **snapshot** pop-up menu again.

### ***Combining Multiple Snapshot Images Into An Animation***

SIMM does not automatically create digital movie files such as AVI, QuickTime, or MPEG. However there are several third-party software packages capable of combining multiple image files into a single animation. Inexpensive examples include QuickTime Pro from Apple, Inc. (available for MS-Windows<sup>®</sup> and Macintosh<sup>®</sup>), and *makemovie*, which comes free with all SGI computers running the IRIX<sup>®</sup> operating system.

### ***Specifying Snapshot Image Dimensions***

The height and width of snapshot images will match the dimensions of the model window exactly. Therefore, if you need to create an animation with a specific pixel resolution, you must first set the model window to the appropriate size. You can do this by entering values into the height and width text fields in the bottom-right corner of the Model Viewer.

### ***A Nifty Trick***

SIMM does not include the ability to take snapshot images of plot windows. However, on some computers it is possible to simply place a plot window on top of the model window to have the plot window included in the snapshot image. In this way it is possible to create snap-

shot images that include a 3D model with one or more 2D plots superimposed. Note that this technique is dependent on the organization of your computer's video memory and may not work on all systems.

### 2.4.3 Forms

#### **Generalized Coordinates Form**

*generalized coordinates*

This form contains a numerical field for each generalized coordinate in the current model. You can use it to position the model accurately by typing in exact values for the generalized coordinates. Just click the left mouse button in the field of the generalized coordinate you want to change, then enter a new value. If the gencoord is clamped, you will not be allowed to input a value outside its range. If the gencoord is not clamped, any values outside its range will be displayed in pink.

When you load a motion file, the motion variable (*e.g.*, % of gait cycle) appears in this form as if it were a generalized coordinate. Thus, you can also position the model at a specific time step in a motion sequence by selecting the motion variable field and entering a value.

The **gencoord** command can be used to toggle the visibility of generalized coordinate fields. For models with more than 50 gencoords, all gencoord fields will be hidden by default to avoid filling the Model Viewer window with too much information.

#### **Model Name Form**

*name*

This field contains the name of the current model. This name appears in the title bar of the model window, and on the model icon when the model window is iconified. The model name can be changed by editing this field.

- w, h (width, height)* These fields contain the width and height of the current model's window. If you need to create snapshot images of a particular resolution, you can enter the dimensions into these fields to set the model window to the same size as the desired snapshot image.
- gear* This field contains the current gear, which affects the speed of movement for the model viewing commands: *i*, *o*, *l*, *r*, *u*, and *d* (see Section 2.4.6, Model Window Viewing Commands) and the speed of the model animation controlled by the **start/stop** command. The gear can have a value between -10.0 and 20.0.

### 2.4.4 Sliders

Next to each field in the generalized coordinates form is a slider which can be used to change the value of the corresponding generalized coordinate. You can move the slider thumb to change the value quickly, or press the left and right arrow buttons to decrease or increase (respectively) the value a little at a time. The slider can not be used to set the *gencoord* value outside its range. If the *gencoord* value is outside its range, the slider thumb will display a red arrow to show this. If the current model has any motions linked to it, sliders for these motions are also located here.

Between each *gencoord* field and slider are two toggle buttons used to clamp (labeled 'C') and lock (labeled 'L') the *gencoord*. *Gencoords* can be clamped or locked from either the Model Viewer window or the *Gencoord* Editor window. If a *gencoord* is clamped, its value can never go outside its range. If, for example, a value outside the range is read in from a motion file, the *gencoord* value will be set to the closest limiting value (the range start or end). If a *gencoord* is unclamped, the value is allowed to go outside the range. If a *gencoord* is locked, its value cannot be changed by entering a new value in the form field or by

moving the slider. The value can only be changed if the gencoord is restored (using the **restore gencoord** command in the Gencoord Editor) or the model's gencoord values are restored (using the **restore gc values** command in the Model Viewer).

## 2.4.5 Toggle Buttons

*trackball* This button toggles the trackball method of interactively translating and rotating the 3D model window view. See Section 2.4.6, Model Window Viewing Commands, for a description of trackball and non-trackball view navigation.

*muscle points* This button toggles the display of unselected muscle points on the current model. Initially, muscle points are displayed only when you select them (with most points you can tell where they are and be able to select them without having SIMM explicitly draw them). If you wish to display all of the points on all of the muscles, however, use this command.

*shadows* This button toggles the display of the body segment shadows. Initially, shadows of the bones are not displayed in the model window. When you turn shadows on, they are drawn only for the body segments for which you defined a shadow direction and distance in the segment definition. See Section 3.3.1, Body Segments, for more details on specifying the shadow parameters for a body segment.

*crosshairs* This button toggles the display of cross hairs in the model window. Cross hairs are useful for locating the center of the model window, which is also the center of the perspective transformation that is applied to the model view.

## 2.4.6 Model Window Viewing Commands

All of the commands described in this section can be used to change the view of a model without opening the Model Viewer window. Just activate the window of the model you want to move, and press the appropriate keys.

There are two methods of changing the view of the model within the model window. The first method, which operates when the *trackball* toggle button in the Model Viewer window is turned on, works as follows. Pressing the `Control` key and the left mouse button will pan the view within the model window. As you move the cursor in any direction, the model will follow. Pressing the `Control` key and the middle mouse button lets you zoom in on and out of the model. Moving the cursor to the right will zoom in, and moving it left will zoom out. The zooming effect will be centered on the points under the cursor when you first press the two keys. Press the `Control` key and the right mouse button to rotate the model with a trackball method. Imagine a trackball superimposed on the model window, and you can move it using the cursor. Moving the cursor in some direction will rotate the model about an axis perpendicular to that direction.

The second method of changing the model view will function if the *trackball* toggle button in the Model Viewer window is turned off. In this case the `Control` key and mouse buttons function as follows. To rotate the model around the X-axis of the world frame, which always points horizontally to the right, press the `Control` key and the left mouse button. The direction and speed of rotation are determined by the position of the cursor in the model window. If the cursor is at the far right edge of the window, the model will rotate quickly in the positive direction around the X-axis. As you move the cursor left,

towards the center of the window, the rate of rotation slows down. If the cursor is in the middle of the window, the rotation rate is zero. As you move the cursor further left, towards the left edge of the window, the rotation rate increases, but in the negative direction. In a similar fashion, pressing the `Control` key and the middle mouse button rotates the entire model around the Y-axis, which always points straight up. Pressing the `Control` key and the right mouse button rotates the model around the Z-axis, which points out of the screen towards you.

In addition to the two methods described above, you can also translate the model by pressing certain keyboard keys. These keys function independently of the two viewing methods described above, so they will operate regardless of which of the two you have chosen. The speed with which they move the model depends on the current gear. These keys are described below.

- I** Pressing this key moves the model **in** towards you, making it larger.
- O** Pressing this key moves the model **out** away from you, making it smaller.
- L** Pressing this key moves the model to the **left** within the model window.
- R** Pressing this key moves the model to the **right** within the model window.
- U** Pressing this key moves the model **up** within the model window.
- D** Pressing this key moves the model **down** within the model window.

*moving the joints*

To move the joints of a model, you can press the keys that are listed in the definitions of the generalized coordinates (see Section 3.3.4, Generalized Coordinates). When you press these keys while the model window is the active window, the values of the generalized coordinates will change. For example, pressing the `k` key changes the value of `knee_angle` in the sample lower-extremity model and thus flexes and extends the knee. The speed and direction of the `gencoord` movements work the same way as the model rotations described above. If you keep the keys pressed to continue changing a `gencoord`, you will eventually reach the end of the `gencoord`'s range. When this happens, the value of the `gencoord` will stop changing, and the model will stop moving. However, when you move the cursor to the opposite side of the window, the `gencoord` will move in the other direction. If you specified that a `gencoord` can *wrap* around its range of motion, then you will not reach these stopping points. With wrapping, when a `gencoord` reaches the end of its range, it will wrap around to the other end of its range and continue to move.

*animating motions*

If you have loaded a motion file (see Section 3.5, Motion Files) into SIMM, you can move the model according to the motion sequence as if the motion parameter (*e.g.*, percent of gait cycle) were a `gencoord`. Just press the key[s] that you specified in the motion file and move the cursor horizontally to move the model through the motion sequence.

*changing display properties*

When you right-click on a bone or world object in the model window, a pop-up menu is shown with various display options. You can change the object's drawmode and material, as well as turn on and off the display of the segment axes and object's normal vectors. You can also set the camera to "look at" that segment, as if you had used the **look at** command described in Section 2.4.2.

## 2.5 Plot Maker

The Plot Maker allows you to plot various computed properties of the muscle-tendon actuators, including muscle force, moment, moment arm, and length. You can create these plot curves for any set of muscles, as a function of any generalized coordinate or motion variable. In addition to generating data curves that are calculated from the musculoskeletal model, there are also two ways you can read in and plot data that are stored in input files. The first method involves plot data files (see Section 3.6, Plot Files, for more information), and the second allows you to plot columns of data from motion files (see Section 3.5, Motion Files, for more information). The Plot Maker window is shown in Figure 2-2.

### 2.5.1 Selector Menu



The model selector lets you choose which model you want to use to make plots. Since some of the plotting options depend on the model, each model has its own set of menus and forms to set the plotting options. When you select a new model with the model selector, the Plot Maker window is redrawn with the new model's menu settings. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



The plot selector lets you choose the plot to which you will add curves. Before you have made any plots, the pop-up menu contains one item, called *new*. Initially, *new* is the current plot, and it indicates that if you generate any curves, a new plot window will be created to display the curves. When you create a new plot window by this



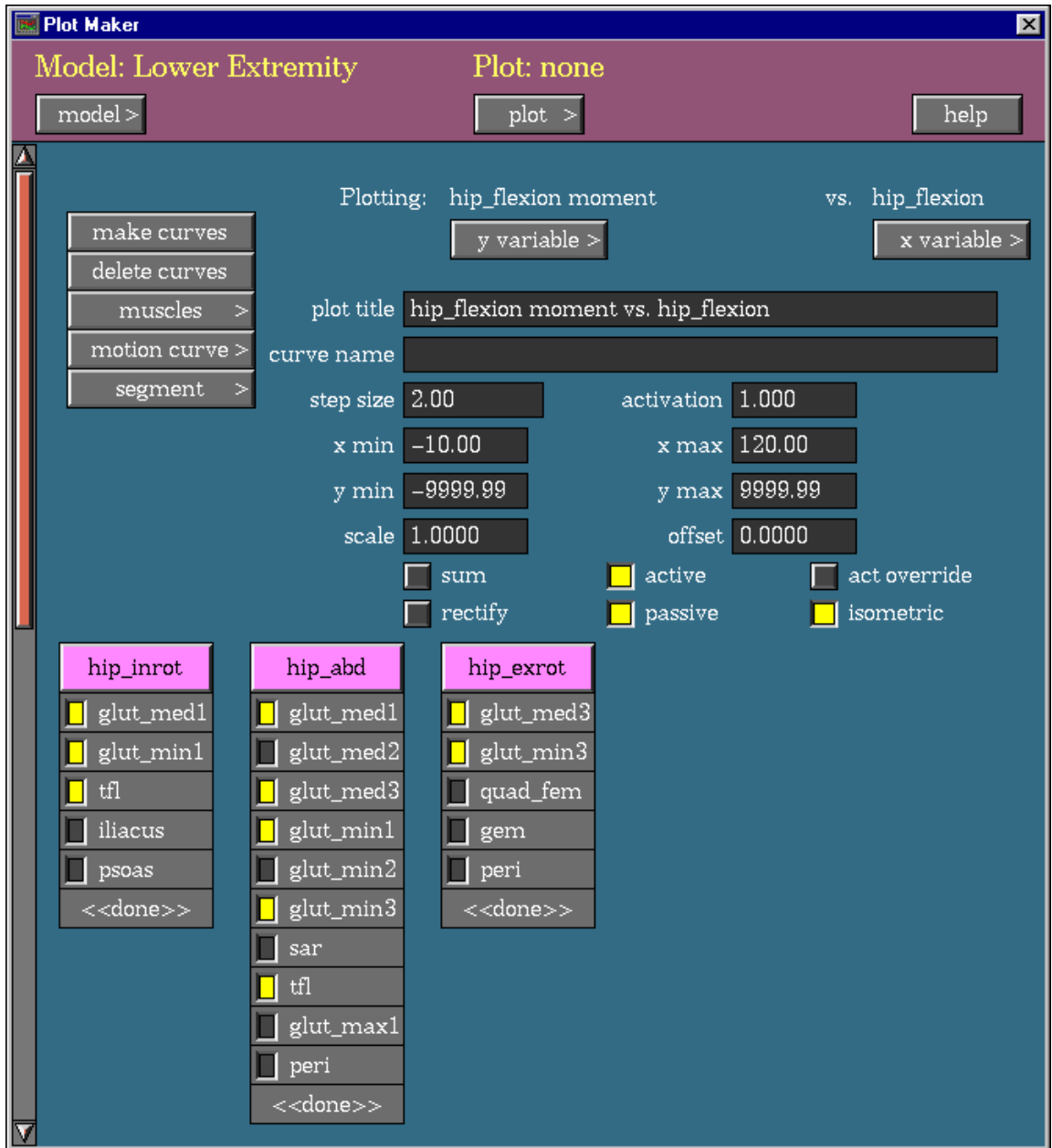


Figure 2-2. Plot Maker window

method, the window is given a title (see below), and the current plot is changed to this new one.

**help**

Clicking on the help button opens a window containing helpful text about the Plot Maker. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.5.2 Command Menu

**make curves**

Select this command to create data curves from a model, according to the plotting parameters you have chosen (see Section 2.5.3, Generating Curves From a Model, for more details). The number of curves added to a plot when **make curves** is selected depends on the *sum* option (see Section 2.5.8, Toggle Buttons). If the *sum* option is off, then one curve will be added to the plot for each selected muscle. If the *sum* option is on, then the individual muscle curves will be summed, and the result (one curve) will be added to the plot.

**delete curves**

Selecting this command deletes all of the selected curves in the current plot window. To select or unselect a plot curve, click the left mouse button on the curve name in the plot key. This toggles the curve between being selected and unselected. When the curve is selected, the region around the curve name is highlighted in white; when unselected the region is black. You can also delete selected curves in a plot by pressing `BackSpace` or `Delete` while the plot or plot key window is the active window.

**muscles** >

This command lets you choose the muscles for which you want to generate plot curves. When you select this item, a pop-up menu of the muscle groups is displayed. Choosing a muscle group from this menu causes it to appear at the bottom of the Plot Maker window. If the chosen muscle

---

group is already displayed at the bottom of the window, then selecting it again will turn it off.

**motion curve** >

This command lets you create plot curves from the data in motion files (see Section 3.5, Motion Files). When you select this command, a pop-up menu of motions is displayed. After choosing a motion, move the cursor to the right to display a menu of the data columns from that motion file. When you select one of these columns, the data curve is added to the current plot (see Section 2.5.5, Y-variables, for more details).

**segment** >

This command lets you choose the body segment to be used with the *muscle orientation* Y-variable. See the description of this Y-variable in Section 2.5.5, Y-variables, for more details.

### 2.5.3 Generating Curves from a Model

Use the following technique to plot computed properties (e.g., moment, force, length) of the muscles in a model. See Section 2.5.5, Y-variables, for a complete list of the musculotendon properties that you can plot.

- (1) If you have more than one model loaded into SIMM, use the model selector to make sure the current model is set correctly.
- (2) If there are already some plot windows on the screen, use the plot selector to choose which plot you want to add your new curves to. If you want to make a new plot window, choose the first item in the pop-up plot menu, labeled *new*. If there are no plot windows yet, the plot selector is already set to *new*, so you don't have to set it.

- (3) Select the Y-variable you want to plot (*e.g.*, force, moment) using the pop-up menu that is displayed when you click on the **y-variable** box.
- (4) Select the X-variable you want to plot against (a joint angle or motion variable) using the pop-up menu that is displayed when you click on the **x-variable** box.
- (5) Select the muscles whose properties you want to plot. Use the **muscles** command to display the appropriate muscle group menus, then choose the muscles from these group menus.
- (6) Set any other options, like *sum*, *active*, *passive*, and *curve name*.
- (7) Click on **make curves**.

Note: When plotting musculotendon properties that depend on activation (*e.g.* moment, force), make sure that the muscle activations are set correctly (they might have been changed by a motion variable). See the descriptions of **act override** in Section 2.5.8, Toggle Buttons, and **reset activations** in Section 2.7.2, Command Menu, for details.

## 2.5.4 Plotting Data from Motion Files

Use the following technique to plot columns of data from motion files. See Section 3.5, Motion Files, for information on creating these files.

- (1) Use the model selector to choose which model's motion curves you want to plot.
- (2) Use the plot selector to choose which plot the curve will be added to.
- (3) Click on the command labeled **motion curve**. A pop-up menu of the motions will appear. Choose

the motion you want, then move the cursor to the right to display the menu of motion curves. These menu items correspond to the columns of data in the motion file. When you release the left mouse button with a menu item highlighted, the corresponding column of data from the motion file will be plotted in the current plot window.

### 2.5.5 Y-variables

**moment** >

The moment of the selected muscles with respect to the chosen generalized coordinate. Moment is defined as force multiplied by the moment arm with respect to a generalized coordinate. The generalized coordinate is chosen from the submenu that appears when you move the cursor off the right side of the **moment** selection. You cannot compute the moment of a muscle with respect to a motion variable, so the **moment** submenu lists only true generalized coordinates. When selecting **moment** as the Y-variable, you must choose a generalized coordinate from the submenu.

**musculotendon length**

The origin-to-insertion length of the selected muscles. This length depends only on the locations of the muscle attachment points and the current configuration of the model's joints.

**moment arm** >

The moment arm of the selected muscles with respect to the chosen generalized coordinate. The generalized coordinate is chosen from the moment arm submenu, which appears when you move the cursor off the right side of the **moment arm** selection. You cannot compute the moment arm of a muscle with respect to a motion variable, so the submenu lists only true generalized coordinates. The moment arm depends only on the locations of the muscle

attachment points and the current configuration of the model's joints.

**fiber length**

The fiber length of the selected muscles. To compute the fiber length of a muscle, SIMM finds the static equilibrium force in the muscle-tendon actuator. Thus the fiber length depends on the settings of the *active* and *passive* toggle buttons (see Section 2.5.8, Toggle Buttons), and also on the current activation level if the *active* component is turned on.

**force**

The isometric force of the selected muscles. The force in a muscle depends on the settings of the *active* and *passive* toggle buttons (see Section 2.5.8, Toggle Buttons), and also on the current activation level if the *active* component is turned on.

**moment@max\_force >**

The moment of the selected muscles calculated using the muscles' peak isometric force at all muscle lengths. Since this selection involves a moment calculation, similar to the **moment** selection described above, a generalized coordinate must be chosen. The generalized coordinate is chosen from the submenu that appears when you move the cursor off the right side of the **moment@max\_force** selection. Since you cannot compute the moment of a muscle with respect to a motion variable, the submenu lists only true generalized coordinates.

**tendon strain**

The tendon strain of the selected muscles. Tendon strain is defined as the stretch of the tendon divided by the tendon slack length. To compute tendon strain, SIMM finds the static-equilibrium force in the muscle-tendon actuator. Thus the tendon strain depends on the settings of the *active* and *passive* toggle buttons (see Section 2.5.8, Toggle Buttons), and also on the current activation level if the *active* component is turned on.

**muscle orientation**

This variable does not generate a plot curve, but rather writes out an ASCII data file with muscle orientation information. For each selected muscle with an attachment point on the chosen body segment (the one selected using the **segment** command), the vector describing the muscle line as it attaches to that body segment is written to the data file. The point *pt1* is the attachment point on the segment; the point *pt2* is the next point on the muscle path, and *vec* is the normalized vector  $pt2 - pt1$ . This information is written to the file for each value of the corresponding X-variable. The name of the file is created using the names of the muscle[s], the name of the X-variable, and the suffix *.mo*.

**motion curve**

This option lets you plot one column of motion data versus another. The submenu contains a list of the motions linked to the current model, and each of those has submenus giving you access to all of the columns of motion data. To use this option you must select a motion curve as both the Y-variable and the X-variable. The selected curves do not have to be from the same motion, but they must contain the same number of data points. Once you have selected both, click on **make curves** to add the plot to the curve. The events for the motion will not be displayed in the plot since the X axis is not *time*.

## 2.5.6 X-variables

The possible independent (X) variables are the current model's generalized coordinates and the motion variables that are linked to the model. You can also select an individual column of motion data as the X-variable (see below). When you choose an X-variable, SIMM sets the *xmin* and *xmax* fields in the options form (see Section 2.5.7, Options Form) to the full range of motion for the generalized coordinate or motion variable. If you want to

plot computed muscle properties over a smaller range, you should edit these form fields before generating your plot curves.

**motion curve**

This item in the X-variable pop-up menu lets you plot one column of motion data versus another. The submenu contains a list of the motions linked to the current model, and each of those has submenus giving you access to all of the columns of motion data. To use this option you must select a motion curve as both the Y-variable and the X-variable. The selected curves do not have to be from the same motion, but they must contain the same number of data points. Once you have selected both, click on **make curves** to add the plot to the curve. The events for the motion will not be displayed in the plot since the X axis is not *time*.

## 2.5.7 Options Form

### *plot title*

This field contains the title of the current plot. If the current plot is *new*, then this field contains the title that will be used when a new plot window is created. The title of an existing plot can be changed by first making the plot the current one (using the plot selector), and then editing the plot title field.

When the current plot is *new*, the default-plot-title generator can be used to title the plot. If you have not yet edited the title of a *new* plot, the title defaults to a string containing the current Y-variable and the current X-variable. When you change the X and Y variables, the title changes to indicate the new variables. If you do not want to use the default title, you can edit the field. Once you enter your own title into the field, the default-plot-title generator is de-activated, so the title does not change when you choose different X and Y variables. If you want to turn the



default-plot-title generator back on, edit the title field and enter a title of zero length (simply delete the current title). The title will then change to a default one consisting of the X and Y variable names, and it will then change as you select new X and Y variables as described above. Note that the default-plot-title generator functions only when the current plot is *new*. Once a plot has been created, you cannot use the default-plot-title generator to automatically re-title it. As described above, however, you can enter your own title to an existing plot at any time.

*curve name* This field contains the text string that will be used to name the next data curve that is generated. If the field is empty when a curve is created, a default name will be used. The default name depends on what you are plotting and the settings of the plotting options as follows.

When plotting computed muscle properties from your model, the curve name depends on the setting of the *sum* option. If *sum* is turned off, the name of each curve generated defaults to the name of the muscle used to create it. If the *sum* option is on, then the curve name defaults to a concatenation of muscle names used to make the curve.

When plotting data from motion files, the default curve name is the name of the motion followed by the label of the column of data as it appears in the motion file.

When plotting data from plot files, the default curve name is the name of the file. You can change this default either by entering the name of the curve directly into the plot file (see Section 3.6, Plot Files, for details), or by editing the *curve name* field.

If you enter your own curve name and then try to generate multiple curves at once (by selecting several muscles and turning off the *sum* option), the curve name will be

deleted, and default names will be used for all the curves. When this happens, it usually indicates that you have forgotten to turn off the *sum* option or to select only one muscle for plotting.

*step size* This field contains the step size used to generate data points for the plot curves. It affects how many data points make up a curve, and thus affects the resolution of, and time to create, a curve. For example, if you are making a plot curve for a generalized coordinate that ranges from 0 to 90 degrees, a step size of 30 will mean that the curve will have data points at 0, 30, 60, and 90 degrees. If you enter a step size greater than the range of the generalized coordinate (*e.g.*, 100 in the above example) the step size will be changed to the magnitude of the generalized coordinate range (90 in this case). When this happens, any curves generated will contain only two data points.

*activation* This field contains a number between 0.0 and 1.0 representing the activation level of all of the muscles, as used for plotting purposes. For example, if this number is set to 1.0, then when you make a plot curve, all of the selected muscles will be assumed to be fully activated, regardless of their individual activation levels as listed in the muscle parameter menu (see Section 2.7, Muscle Editor). This “universal” activation level will be used only when *act override* is turned on (see Section 2.5.8, Toggle Buttons). Otherwise, the muscles’ individual activation levels will be used.

*x min, x max* These fields let you change the range of the generalized coordinate or motion variable which serves as the X-variable for making new data curves. When you choose an X-variable, the variable’s minimum and maximum values are copied into the *x min* and *x max* fields, respectively. When you generate data curves, these values serve as the default range for the X-variable. You can change the range

by editing these two fields. If you change the default range and then choose another X-variable, the values you entered are erased, and the new X-variable's default range is copied into the fields. If you then re-select the original X-variable, that variable's default range is copied into the fields, not the changed values you entered earlier.

- y min, y max* These fields let you limit the range of the Y-coordinates of the data points in the curves that you add to a plot. By default, *y min* is a large negative number and *y max* is a large positive number, meaning that all of the data points in the curves should fall within the range and thus not get clipped. By changing these values, you can clip out data points that do not lie within a desired range. Every time a curve is added to a plot, each of the data points is compared to *y min* and *y max*. If the point is less than *y min*, it is set equal to *y min*. If it is greater than *y max*, it is set equal to *y max*. If the *sum* option is being used to sum curves from more than one selected muscle, *the individual curves are compared to this clipping range*, not the final summed curve.
- scale* The scale field allows you to specify a scaling factor when generating plot curves using the **make curves** command. Whenever you generate a curve, each data point is multiplied by this scale factor before adding the curve to the plot window. This scaling is performed before the *offset* is applied.
- offset* The offset field allows you to specify an offset factor when generating plot curves using the **make curves** command. Whenever you generate a curve, this offset value is added to each data point before adding the curve to the plot window. This offset is performed after the *scale* is applied.

### 2.5.8 Toggle Buttons

- sum* This toggle button lets you choose whether or not you want to sum the curves of the selected muscles to generate a single curve. If *sum* is off, then each selected muscle creates a curve that is added to the plot. If *sum* is on, then the individual curves are summed and only one curve is added to the plot. The current setting of *sum* has no effect if you are plotting data from plot data files or columns of data from motion files.
- rectify* This toggle button controls the sign of the data points in the curves as they are added to a plot. If *rectify* is on, then all of the data points are made positive, in effect plotting the absolute value of the data. If it is turned off, then the data is not modified before plotting. Note that this option does not reverse the sign of the data points, but rather takes their absolute values, so be careful when using it while creating curves that have positive as well as negative data points. The current setting of *rectify* has no effect if you are plotting data from plot data files or columns of data from motion files.
- active* This toggle button lets you choose whether or not to plot the active component of muscle force. Turning it on means that the force from the active force-length curve of the muscle will be included when computing muscle-tendon force. The current setting of *active* has no effect if you are plotting data from plot data files or columns of data from motion files.
- passive* This toggle button lets you choose whether or not to plot the passive component of muscle force. Turning it on means that the force from the passive force-length curve of the muscle will be included when computing muscle-tendon force. The current setting of *passive* has no effect

if you are plotting data from plot data files or columns of data from motion files.

*act override* This toggle button controls the use of the “universal” activation level in the options form. When it is turned on, the activations of all of the muscles used for plotting will be set to the value shown in the *activation* field of the options form. This does not change their values in the muscle parameters form in the Muscle Editor; it merely sets the activations for plotting purposes only (to reset activations in the parameters form to their defaults, see **reset activations** in Section 2.7.2, Command Menu). When *act override* is turned off, the muscles’ individual activations as listed in the parameters form will be used. This toggle button is particularly useful when you have loaded a motion file which contains muscle activations (see Section 3.5, Motion Files). After animating the model according to the motion, the muscle activations will be changed to match the values in the motion file (so that their size and color will vary during the animation). However, if you want to plot the muscles’ maximum isometric forces during the motion, you can simply set the Plot Maker’s activation level to 1.0 and turn on *act override*, rather than individually changing each muscle’s activation in the Muscle Editor. There is no way to override just a subset of the muscle activations. The current setting of *act override* has no effect if you are plotting data from plot data files or columns of data from motion files.

*isometric* This toggle button gives you control over the use of the force-velocity characteristics of the muscle fibers when calculating muscle force during a motion. When on, fiber velocities are set to zero and isometric muscle forces are calculated. When off, fiber velocity is used to scale the muscle force according to the muscle’s force-velocity curve. See Section 3.4, Muscle Files, for details on how

fiber velocities are estimated and used to scale isometric muscle force.

Note: fiber velocities affect only force and moment calculations, and only when plotting against motion variables which contain gencoord velocity information. For all other plotting scenarios, this toggle button has no effect.

## 2.6 Joint Editor

The Joint Editor enables you to graphically manipulate the kinematics of the joints. Once you select a joint, the three rotations and three translations (the “DOFs”) that comprise the joint are displayed. If the rotations and translations are constants, you can change them by clicking in their fields and typing in new values. If they are functions of the generalized coordinates, they can be changed by moving, adding, and deleting the control points of the splines that define the kinematic functions (Figure 2-3). To display the kinematic function for a DOF, click on the field for that DOF. You can also change whether a given translation or rotation is a function or constant, thereby interactively increasing or decreasing the complexity of the joint model.

### 2.6.1 Selector Menu



The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Joint Editor. When you are done

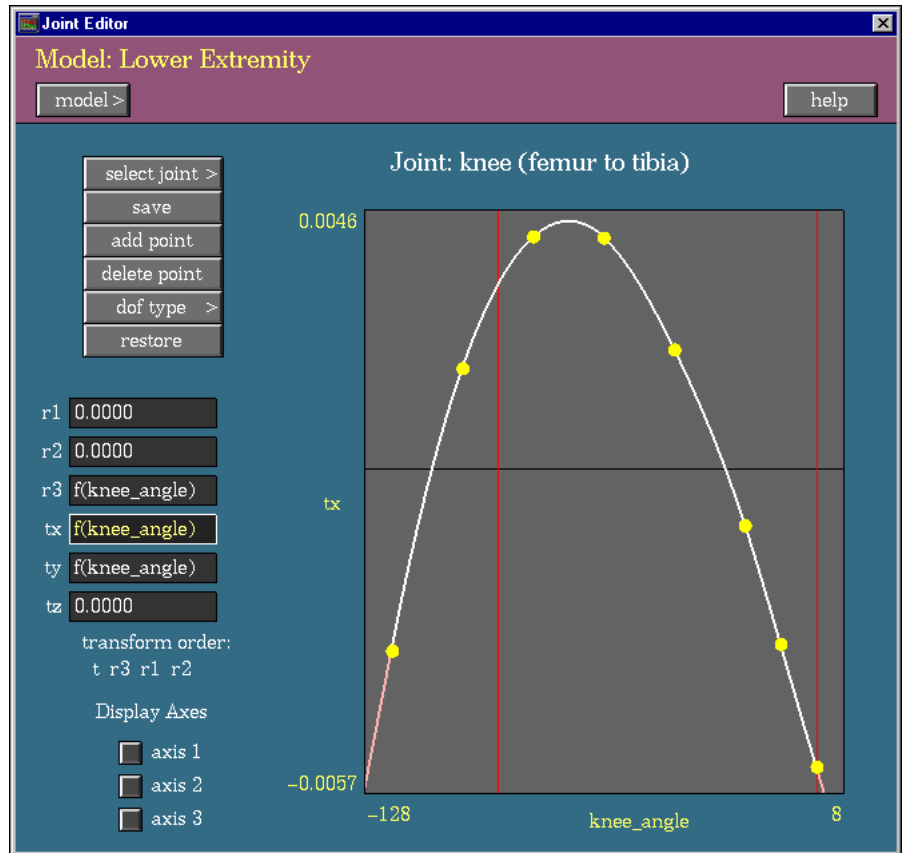


Figure 2-3. Joint Editor window

reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.6.2 Command Menu

**joint** >

This command lets you select the joint you want to edit. When a joint is selected, the six degrees of freedom (dofs) that comprise the joint are displayed in the dof form in the lower left corner of the Joint Editor window.

**save**

This command saves the current joint of the current model. It saves the joint to a buffer so that you can restore it if you make changes that you later want to undo. When you first load a model, SIMM saves copies of all the joints, so you can restore them back to their original form without having to save them first.

**add point**

This command lets you add a control point to the currently displayed function. After selecting this item, it remains highlighted until you click the left mouse button in the function plot area at the location where you want to add the new control point. SIMM figures out which two points the new point should go between, and adds it to the function. The point is added to the function as soon as you press the left mouse button, and if you keep the button pressed, you can move the point around within the plot area before releasing the button. This command does nothing if no function is currently displayed in the function plot area.

**delete point**

This command lets you delete a control point from the currently displayed function. After selecting it, this item remains highlighted until you select the point you want to delete by clicking the left mouse button on the point. You can cancel this command by clicking the left mouse button while the cursor is not on any point. This command does nothing if no function is currently displayed, or if the current function has only two points.

**dof type** >

This command lets you change the type (*constant* or *function*) of each of the dofs. Selecting it pops up a menu of the six dofs, each with a submenu containing two items, *constant* and *function*. If you choose *function*, then you must use the submenu to the right to choose which generalized coordinate to use in the function (the function's X-variable). If you change a dof from a constant to a function, SIMM creates a new function to use for the dof. This



new function is given two control points, whose X-values are -360.0 and 360.0, and whose Y-values are just below and just above the value of the dof when it was constant. If you change a dof from a function to a constant, its value is initially set to the average of the Y-coordinates of the function's control points. If you have closed loops in your model and the Inverse Kinematics solver is on, changing the complexity of a model will cause SIMM to re-evaluate the loop configurations. SIMM will try to find a configuration that closes all the loops. If no solution is found, an error message is displayed and the model is displayed with broken loops.

**restore**

This command restores a previously saved joint description. Selecting it restores the joint from a buffer that was saved last time you used the **save** command. When you first create a model, SIMM saves copies of all of the joint descriptions, so you can restore a joint back to its original form without having to save it first.

### 2.6.3 Dof Form

*r1 r2 r3*  
*tx ty tz*

The dof form in the lower left corner of the Joint Editor window gives you access to each of the six degrees of freedom (dofs) in the current joint. If the dof is a constant, you can change the value by selecting its field with the left mouse button and then typing in a new value. If the dof is a function, you can change the function in the plot area of the tool window. When you click the left mouse button in the dof's field, the function is displayed in the function plot area. See Section 3.3.3, Joints, for a description of the six dofs.

## 2.6.4 Function Plot Area

Once a function is displayed in the plot area, you can add, delete, and move its control points. To move a control point, press the left mouse button while the cursor is on the point. The region behind the point you select is shaded to indicate where you can move the point. You cannot move a control point so that it overlaps one of its neighbors.

As you move a control point, its coordinates are shown just above the plot area to help you position the point. When you release the left mouse button, the point remains at its new location, and the joint will now move according to this new function. To help you edit a joint function, you can press the `i`, `o`, `l`, `r`, `u`, and `d` keys to move the view of the function in, out, left, right, up, and down, respectively. You can also use the `x` and `y` keys to zoom in and out along just the X or Y axes (press the `Shift` key along with `x` or `y` to zoom out).

Note: If you create a kinematic function that has steep slopes or sharp corners, the display of the function may look choppy. That is, you may be able to notice the series of line segments that is used to approximate the curve in the plot area. This piece-wise linear approximation is for display purposes only. When SIMM calculates joint kinematics and needs a Y-value for the function (given an X-value, which is a generalized coordinate), it finds an exact solution using the algebraic form of the cubic spline.

The `gencoord` range limits are displayed in the plot area using red vertical lines. If the model has closed loops and the Inverse Kinematics solver is on, changing the kinematic function will change the model configuration. SIMM will try to find a configuration that closes all the

loops. If a solution cannot be found, an error message will appear and the model will be displayed with broken loops.

### 2.6.5 Joint Axis Toggle Buttons

You can turn on and off the display of the axes of rotation for each joint using these toggle buttons. The three buttons below the DOF form control the display of *axis1*, *axis2*, and *axis3* for the current joint.

## 2.7 Muscle Editor

The Muscle Editor gives you access to all of the parameters that describe the muscles. The paths of the muscles can be altered by selecting and moving the attachment points in the model window (see Sections 2.7.3 and 2.7.4). The Muscle Editor has several features to help you edit the muscle paths, such as an algorithm that moves a muscle point toward a specific vertex on a bone surface. The muscle-tendon parameters can also be changed with this tool (Figure 2-4). You cannot edit the force-length curves of the muscles or tendons with the Muscle Editor, but read Appendix A, Tips and Caveats, for details on how to edit these curves interactively in SIMM.

### 2.7.1 Selector Menu



The model selector lets you select which model's muscles you can change with the Muscle Editor. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Muscle Editor. When you are done

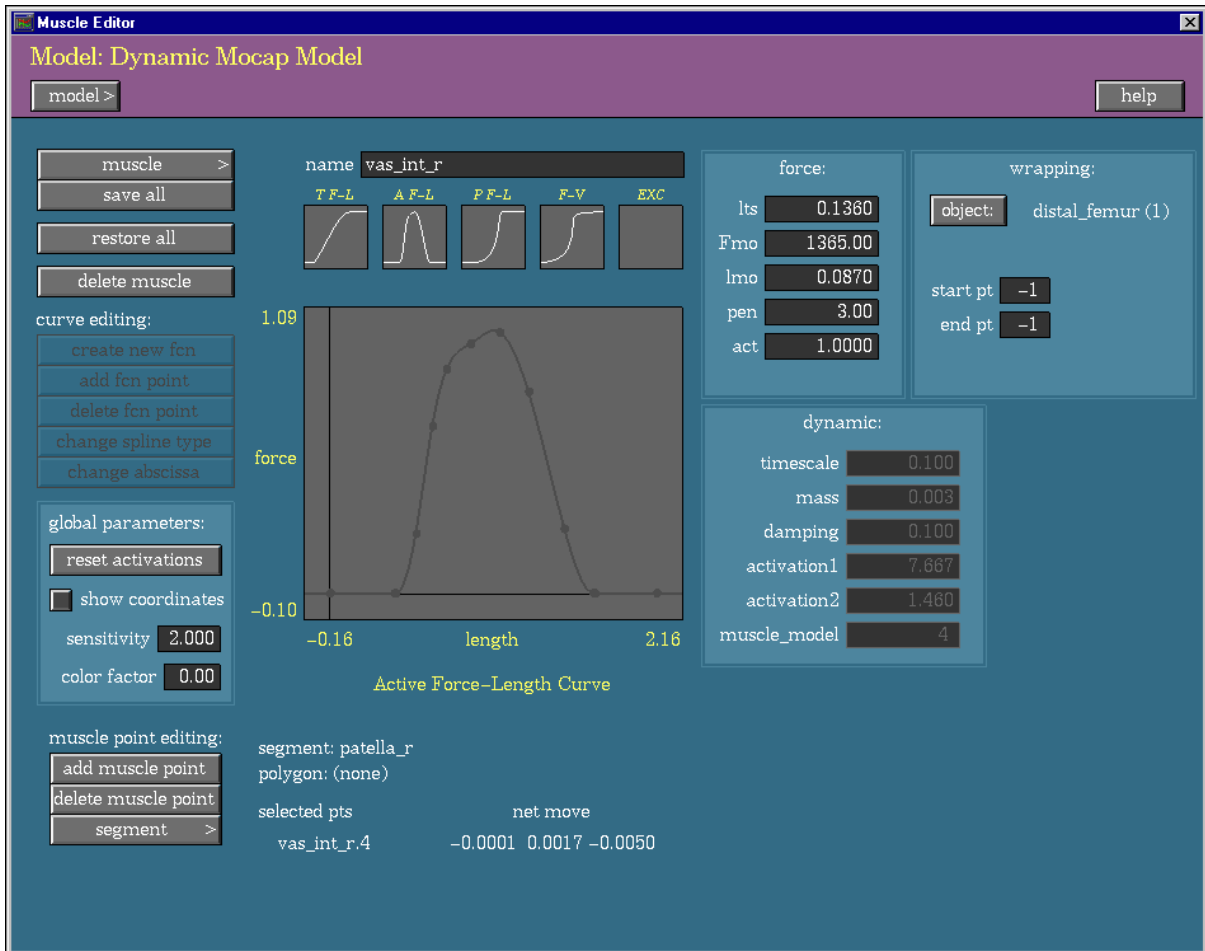


Figure 2-4. Muscle Editor window

reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.7.2 Command Menu



This command lets you create a new muscle or choose an existing muscle for editing. When you select this item, a

pop-up menu of existing muscles is displayed. The first item, *new muscle*, creates a new muscle named Muscle\_X, adds it to the current model, and makes it the current muscle for editing. The muscle is attached to the first segment in the model and is given two arbitrarily positioned attachment points. The muscle properties are inherited from the default muscle, if there one. If there is not, then selecting *new muscle* will create a default muscle. The remaining items in the pop-up menu allow you to select existing muscles for editing.

**save all**

This command saves the paths and muscle-tendon parameters of all of the muscles in the current model. It saves them to a buffer so that you can restore them if you make changes that you later want to undo. When you first load a model, SIMM saves copies of all of the muscles, so you can restore them back to their original form without having to save them first. There is no way to save a single muscle or subset of muscles.

**restore all**

This command restores the paths and muscle-tendon parameters of all the muscles in the current model. The muscles are restored from the buffer that was saved last time you used the **save** command. When you first load a model, SIMM saves copies of all of the muscles, so you can restore them back to their original form without saving them first. There is no way to restore a single muscle or subset of muscles.

**delete muscle**

This command deletes the currently selected muscle. The default muscle cannot be deleted.

### 2.7.3 Global Parameter Commands

These commands control parameters that not specific to a particular muscle.

**reset activations**

This command resets the activation levels for all of the muscles in the current model to their default values. If you specified activations in the muscle file, these values are used as the defaults. If you did not specify activations for some or all muscles, a value of 1.0 is used as the default for those muscles.

*show coordinates*

This toggle button turns on and off the display of the coordinates of selected muscle attachment points. The coordinates of each point are expressed in that point's reference frame, and are displayed next to the point in the model window.

*sensitivity*

This field contains the sensitivity factor, or gear, which affects the speed of movement for the commands  $x$ ,  $y$ ,  $z$ , and 1, 2, 3, 4, 5, 6, 7, 8, 9 (See Section 2.7.5, Model Window Keyboard Commands, for more details on these commands for moving selected muscle points).

*color factor*

This field contains the color factor, which is used to color all of the muscles according to their lengths. When the factor is 0.0, this technique is not used and the muscles are colored according to their material properties as specified in the muscle file. However, when the factor is greater than 0.0, the muscles are colored red when they are shorter than their resting lengths, blue when they are longer than their resting lengths, and white when they are near their resting lengths. By changing the value of the color factor, you can change the transitions between blue, white, and red. A higher factor means that a muscle's color is more sensitive to length change, so that it will turn red more quickly when lengthening, and blue more quickly when shortening.

## 2.7.4 Editing Force-Generating Functions

The spline functions which control the generation of force in the muscles can be edited using the function editing area. At the top of this region are thumbnails for the five functions that you can edit. They are: the force-length curve of tendon (T F-L), the active force-length curve of the muscle fibers (A F-L), the passive force-length curve of the fibers (P F-L), the force-velocity curve of the fibers (F-V) and the excitation pattern of the muscle (EXC). When you click on one of the thumbnails, the function is displayed in the larger editing window below. If the selected function is inherited from the default muscle, its display in the editing window will be grayed out, indicating that you cannot edit it. If you want to modify the function for this muscle only, then you must first right-click in the editing window and choose *make unique* from the pop-up menu. This will give the muscle its own copy of the function from the default muscle, which you can now edit by clicking on control points with the left mouse button and moving them to the desired locations. To remove the muscle's own copy of the function and instead have it inherit it from the default muscle, choose *inherit from default* from the pop-up menu. If you want to edit the default muscle's copy of the function, you must first select the default muscle using the **muscle** button.

The buttons to the left of the editing window can also be used to edit the selected muscle function. These buttons are described below.

**create new fcn**

This command creates a new muscle function for the current muscle. Active force-length curves, passive force-length curves, tendon force-length curves and force-velocity curves are defined as natural cubic splines. If an excitation curve is created it will be created using the default

excitation format. If no default excitation format is specified, a pop-up menu lets you choose what type of function (step or spline) to create. New step functions are created with 2 control points, spline functions with 3 control points. This command is not available if the function is already defined.

**add fcn point**

This command lets you add a control point to the muscle function displayed in the plot area. After selecting this item, it remains highlighted until you click the left mouse button in the function plot area at the location where you want to add the new control point. SIMM figures out which two points the new point should go between, and adds it to the function. The point is added to the function as soon as you press the left mouse button, and if you keep the button pressed, you can move the point around within the plot area before releasing the button. If you selected an existing control point, SIMM will add a node to the spline. A new points will be added slightly offset from the chosen point. Keeping the mouse pressed will not allow you to move this new point around. You will have to select and edit it individually. This command is not available if the function is not defined.

**delete fcn point**

This command lets you delete a control point from the currently displayed muscle function. After selecting it, this item remains highlighted until you select the point you want to delete by clicking the left mouse button on the point. You can cancel this command by clicking the left mouse button while the cursor is not on any point. This command does nothing if no function is currently displayed, if the current function has only two points, or if the function is inherited. This command is not available if the function is not defined.

**change spline type**

This command lets you change the spline type of excitation functions. The type can be a step function, a natural



cubic, or you can choose to inherit the type from the default muscle. This command is only available when excitation function is the currently selected function.

**change abscissa**

This command lets you change the abscissa of the excitation function. When selected, a pop-up menu will appear. The abscissa can be time, or any gencoord defined in the model. This command is only available when excitation function is the currently selected function.

## 2.7.5 Editing Force-Generating Parameters

When you choose a muscle using the **parameters** command, a form containing that muscle-tendon actuator's force-generating parameters is displayed. This form has five fields, described below. A detailed description of how these parameters are used to compute isometric muscle-tendon force is beyond the scope of this manual. For more details, consult the documents listed at the end of Section 1.6, SIMM Tutorials. To change the value of one of these parameters, click on its field and type in a new value. If the field is grayed out, it means that this parameter is inherited from the default muscle, and cannot be edited. If you want to modify the parameter for this muscle only, then you must first right-click in the field and choose *make unique* from the pop-up menu. This will give the muscle its own copy of the parameter from the default muscle, which you can now edit by left-clicking on the field. To remove the muscle's own copy of the parameter and instead have it inherit it from the default muscle, choose *inherit from default* from the pop-up menu. If you want to edit the default muscle's copy of the parameter, you must first select the default muscle using the **muscle** button.

- Its** *tendon slack length*, the maximum length at which the tendon develops no force. This parameter is used to find tendon strain, which scales the X-coordinates of the tendon's force-length curve.
- Fmo** *maximum isometric force* that the muscle fibers can produce (active component). This parameter scales the Y-coordinates of the tendon's force-length curve and of the muscle's active and passive force-length curves.
- lmo** *optimal fiber length*, the fiber length at which maximum active isometric force is generated. This parameter scales the X-coordinates of the muscle's active and passive force-length curves.
- pen** *pennation angle*, the angle at which the muscle fibers insert on the tendon or bone. As pennation angle increases, the amount of muscle force that is transmitted through the tendon to the bone is decreased.
- act** *level of activation*, which can range from 0.0 to 1.0. This parameter scales the active component of isometric muscle force.

### **Muscle Wrapping Form**

If the muscle has one or more wrap objects associated with it, SIMM will display the wrapping parameters in a region labeled *wrapping*. At the top of this region is a button labeled *object*. Clicking on it will display a pop-up menu of the wrap objects that are currently associated with the muscle. The order of the objects in the menu is the order in which they are processed when wrapping is calculated. If you want to change this order, you must use the Wrap Editor to unassign a wrap object from the muscle, and then reassign it [to the end of the list]. You can

use this pop-up menu to choose which object's parameters you want to display and/or change.

*start pt, end pt*

These two fields are used to specify the region of the muscle path that is allowed to wrap over the wrap object. *Start pt* defines the first muscle attachment point, and *end pt* defines the last attachment point in the region. Only muscle path segments that are between *start pt* and *end pt* will be checked for intersection with the wrap object. A value of -1 for *start pt* means to start at the first muscle point, and a value of -1 for *end pt* corresponds to the last point in the muscle. The muscle points are numbered starting at 1 and are in the order they are listed in the muscle definition, including all via points (points which are active only for certain gencoord ranges), whether or not the via points are active at the time wrapping is calculated.

*hybrid, midpoint, axial*

If the wrap object associated with the muscle is an ellipsoid, then you will also see a set of radio buttons, below the *start pt* and *end pt* fields, for specifying the desired ellipsoidal wrapping technique. The three techniques, hybrid, midpoint, and axial, are described in detail in Section 2.10.5, *Ellipsoidal Wrapping Methods*. The method you choose with these radio buttons will be used for this muscle only. When you choose the wrapping method in the Wrap Editor, it will set the method for all muscles which are associated with that wrap object.

### ***Dynamic Parameters Form***

If the muscle has dynamic parameters defined (for use with the muscle models in Dynamics Pipeline simulations), SIMM will display them in a form labeled *dynamic*. If you do not plan to use this model for dynamic simulations, then you can ignore this form. You cannot add or delete dynamic parameters in the Muscle Editor, but you can change their values for the selected muscle

via this form. To add or delete parameters, edit the muscle file as described in the Dynamics Pipeline manual. To change the value of one of these parameters, click on its field and type in a new value. If the field is grayed out, it means that this parameter is inherited from the default muscle, and cannot be edited. If you want to modify the parameter for this muscle only, then you must first right-click in the field and choose *make unique* from the pop-up menu. This will give the muscle its own copy of the parameter from the default muscle, which you can now edit by left-clicking on the field. To remove the muscle's own copy of the parameter and instead have it inherit it from the default muscle, choose *inherit from default* from the pop-up menu. If you want to edit the default muscle's copy of the parameter, you must first select the default muscle using the **muscle** button.

### 2.7.6 Editing Attachment Points

Use the following technique to select muscle attachment points. Once a muscle point is selected, you can move it using the two methods described in the following section.

- (1) Click on the model window to make it the active window.
- (2) Press and hold down the Space Bar.
- (3) Move the cursor onto each muscle point you want to select, and click the left mouse button.
- (4) Release the Space Bar when you are done selecting points.

The muscle points are highlighted in yellow when they are selected. As you select and unselect muscle points, the list of selected points is updated in the Muscle Editor window.

The points within each muscle are numbered according to their order in the muscle definition (in the muscle file).

### ***Moving Selected Attachment Points***

There are three ways to move selected muscle attachment points. The first involves pressing the `Shift` key and the left mouse button, then moving the cursor in the direction in which you want to move the points. With this method, the muscle points are constrained to move in the plane of the screen, so you can change the model view as needed in order to move the selected points in any desired direction.

The other methods involve pressing keys while the model window is the active window. These methods are described in detail in the next section. The first method moves the muscle points along the X, Y, and Z axes of the body segment to which each point is attached. The second method helps you position a single muscle point on the surface of a bone, using the *selected polygon*. You can select a polygon on the body segment of the muscle point by pressing the `F9` key and moving the cursor over the polygons of the bone. Each polygon is highlighted as you move over it. To select a polygon, click the left mouse button while it is highlighted. This polygon becomes the selected one, and you can then move the muscle point towards its vertices.

### ***Adding and Deleting Attachment Points***

You can add and delete attachment points, and change the body segment to which the points are attached, using the buttons in the lower left corner of the tool window. These buttons are described below.

**add muscle point**

This command lets you add an attachment point to a muscle path. It works by duplicating the selected muscle

point, then placing the copy a small distance away. The duplicated point becomes the new selected point, so you can then move it to the desired location. If there is not exactly one selected muscle point, this command does nothing.

**delete muscle point**

This command lets you delete an attachment point in a muscle path. It deletes the selected muscle point, leaving no currently selected points. If there is not exactly one selected muscle point, this command does nothing.

**segment** >

This command lets you change the body segment to which the selected muscle points are attached. When you change segments using the pop-up menu, the muscle points initially remain in the same positions on the screen, but if you move the joints you will see that they now move with the new segment.

### 2.7.7 Model Window Keyboard Commands

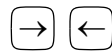
X Y Z  
Shift

These keys constitute the second method of moving selected muscle points. Pressing these keys while the model window is active moves the selected muscle points in the X, Y, and Z directions. To move the points in the positive direction along these axes, press the `Shift` key while pressing `x`, `y`, or `z`. You can press any number of these keys simultaneously to move the points in several directions at once (*e.g.*, to move the points in the positive X and Z directions, press the `x`, `z`, and `Shift` keys).

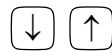
1 2 3  
4 5 6  
7 8 9

These keys constitute the third method of moving selected muscle points. Pressing the number keys while the model window is the active window moves a selected muscle point towards a vertex in the selected polygon. If the selected polygon has  $n$  vertices, then the keys 1 through  $n$  move the selected point. For example, if the polygon has

three vertices, pressing 3 moves the point toward the third vertex. Pressing 1 and 3 at the same time moves the muscle point toward the midpoint of the line connecting the first and third vertices. If you want to move the point directly onto the surface of the polygon, keep the appropriate number keys pressed until the “net move” coordinates displayed in the Muscle Editor window stop changing. These keys do nothing if more than one muscle point is selected or if no polygon has been selected.



The left and right arrow keys change the selected polygon. If there is no selected polygon, these keys do nothing. If there is one, then the right arrow key changes the selected polygon to the next polygon in the bone file. Similarly, the left arrow key changes the selected polygon to the previous polygon. The number of the selected polygon is displayed in the tool window.



The up and down arrow keys change the selected muscle point within a muscle. For example, if the second point in a muscle is selected, then pressing the up arrow key would select the third point (and unselect the second). Similarly, pressing the down arrow key would select the first point. The muscle points are numbered according to their order of appearance in the muscle file. If there is more than one muscle point selected, these keys do nothing.

## 2.8 Gencoord Editor

The Gencoord Editor lets you view and edit properties of the generalized coordinates, such as their ranges and default values. It also lets you create and modify restraint functions for the gencoords, which are used by the inverse kinematics solver as well as by the Dynamics Pipeline.

## 2.8.1 Selector Menu



The model selector lets you choose which model's gencoords to view and edit. When you choose a new model with the model selector, the Gencoord Editor window is redrawn with the settings of the new model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.

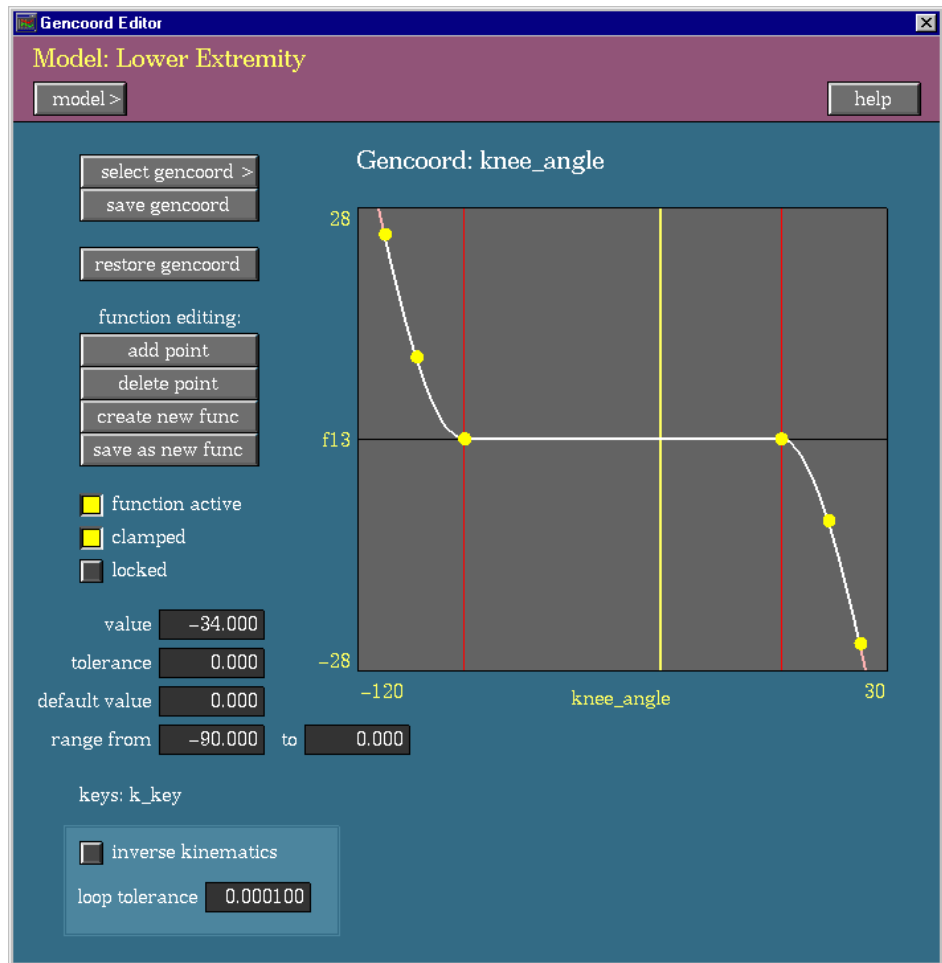


Figure 2-5. Gencoord Editor window



A rectangular button with a grey background and a black border, containing the text "help" in a bold, black, sans-serif font.

Clicking on the help button opens a window containing helpful text about the Gencoord Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.


## 2.8.2 Command Menu

A rectangular button with a grey background and a black border, containing the text "gencoord" in a bold, black, sans-serif font, followed by a right-pointing chevron symbol ">".

This command lets you select which gencoord's properties you want to edit. When a gencoord is selected, its properties can be viewed and edited in the toggle boxes and gencoord form in the lower left corner of the editor window. If the gencoord has a restraint function defined, the function is displayed in the restraint function plot area on the right side of the window.

A rectangular button with a grey background and a black border, containing the text "save gencoord" in a bold, black, sans-serif font.

This command saves the current gencoord of the current model. It saves all of the gencoord's properties to a buffer so that they can be restored if you make changes that you later want to undo. When you first load a model, SIMM saves copies of all the gencoords, so you can restore them back to their original form without having to save them first.

A rectangular button with a grey background and a black border, containing the text "restore gencoord" in a bold, black, sans-serif font.

This command restores a previously saved gencoord description. Selecting it restores the current gencoord's properties from a buffer that was saved last time you used the **save gencoord** command. When you first create a model, SIMM saves copies of all of the gencoords, so you can restore a gencoord back to its original form without having to save it first.

A rectangular button with a grey background and a black border, containing the text "add point" in a bold, black, sans-serif font.

This command lets you add a control point to the currently displayed restraint function. After selecting this item, it remains highlighted until you click the left mouse button in the function plot area at the location where you want to

add the new control point. SIMM figures out which two points the new point should go between, and adds it to the function. The point is added to the function as soon as you press the left mouse button, and if you keep the button pressed, you can move the point around within the plot area before releasing the button. This command does nothing if no function is currently displayed in the function plot area, or if the function is inactive (displayed in gray).

**delete point**

This command lets you delete a control point from the currently displayed restraint function. After selecting it, this item remains highlighted until you select the point you want to delete by clicking the left mouse button on the point. You can cancel this command by clicking the left mouse button while the cursor is not on any point. This command does nothing if no function is currently displayed, if the current function has only two points, or if the function is inactive (displayed in gray).

**create new func**

This command creates a new restraint function for the current gencoord. SIMM gives the new restraint function a new function number greater than the highest one already defined. The new function is created with 8 control points, including a pair of coincident points at the start and end of the gencoord's range. This command does nothing if the current gencoord already has a restraint function.

**save as new func**

This command saves the current restraint function to another function number. It is useful when you have used the same function for several gencoords in the joint file, but now you want to edit the function for just one of the gencoords that uses it. *You should use this command to save the function to a new number before editing it.* If the function is edited before being saved and the function is used by other gencoords, the changes will appear in the restraint function plots for the other gencoords as well. To

restore the original function for these gencoords, use the **restore gencoord** command.

### 2.8.3 Toggle Buttons

*function active*

This button allows you to toggle the restraint function from active to inactive. When the restraint function is active, it can be edited and it will be used by the inverse kinematics solver if the gencoord is not clamped. If the restraint function is inactive, it is displayed in gray in the function plot area and is not used to restrain gencoord values when using the inverse kinematics solver. When the function is inactive, it cannot be edited, and points cannot be moved, added, or deleted. If there is no restraint function for the gencoord, this button does nothing.

*clamped*

This toggle button allows you to clamp or unclamp the gencoord. If the gencoord is clamped, it is forced to remain within its specified range and cannot assume values outside its range. This toggle button has the same effect as the toggle button in the Model Viewer, and is updated if the button in the Model Viewer is used. If the gencoord is unclamped, its value can go beyond its range, and it will be displayed in pink if it does so. If the gencoord is unclamped and has an active restraint function, this function will be used by the inverse kinematics solver. If a gencoord is outside its range when clamping is turned on, its value will be set to the closest range limit.

*locked*

This toggle button allows you to lock and unlock the current gencoord. When a gencoord is locked, its value cannot be changed by moving its slider, typing in a new value, playing back a motion, or by the inverse kinematics loop solver. Its value can be changed only by restoring the gencoord using the **restore gencoord** command, or by

restoring a model's default configuration, using the **default config** command in the Model Viewer.

### 2.8.4 Gencoord Form

The gencoord form in the lower left corner of the Gencoord Editor window contains numerical fields for the gencoord properties.

*value* The current value of the gencoord is displayed in this field. If the gencoord is part of a loop, changing its value will cause the inverse kinematics solver to try to keep the loop closed by changing the other gencoords in the loop, while keeping the current gencoord constant. If it cannot find a solution at the current gencoord value, it will find the closest one. If the current gencoord is locked, its value cannot be changed by typing in a new value. If the gencoord is clamped, you cannot change its value to a number outside its specified range.

*tolerance* The tolerance of a gencoord is used when the gencoord is set to a new value. If the difference between the new value and the current value is less than the tolerance, the new value is ignored, and the gencoord remains unchanged. This tolerance feature can be useful when you are importing a motion which contains identical or similar gencoord values for some periods of time. By setting the tolerance of each gencoord to, say, 0.5 degrees, small changes in gencoord values will not result in the joint transformation matrices being recalculated, thus saving some computation time. This feature is especially useful when you are importing a motion in real-time (with the Motion Module), and want the SIMM animation to be as fast as possible. The default value for tolerance is 0.0.

*default value* The default value of a gencoord is the value that is reverts to when you select the **restore gc values** command in the Model Viewer. When you change the default value of a gencoord that is used in a loop, the inverse kinematics solver will re-evaluate the loop's default configuration, potentially changing one or more gencoord's default values so that the loop remains closed. If default values of other gencoords in the loop are changed, a message window will pop-up to alert you to the new default values. You can then choose to (1) accept and save the new default values, (2) view but not save the values as defaults, or (3) discard them.

*range from , to* The range of the gencoord is displayed in the *range from* and *to* fields. Both values can be changed by selecting the appropriate field with the left mouse button and then typing in a new value. Changing the range does not change the restraint function (if any), so you may need to edit the gencoord's restraint function after changing its range.

*keys* The *keys* display show you which keys control the current gencoord. See Section 2.4.6, Model Window Viewing Commands, for a description of how keyboard keys can be used to change a gencoord's value.

## 2.8.5 Inverse Kinematics

*inverse kinematics* The inverse kinematics (IK) toggle button allows you to turn loop solving on and off. When the IK solver is on, SIMM will try to find a set of gencoord values that keeps all of the loops of body segments closed, while still satisfying the joint constraints. When solving, SIMM tries to keep clamped gencoords within their ranges, applies restraint functions (if specified) to unclamped gencoords to keep them within limits, and keeps locked gencoords at their locked values. When a model is loaded, SIMM uses

the solver to assemble the system and close all the loops using the default values. If the solver is on when joint definitions are changed (changing dof types, editing functions), and when motions are applied to a model, the solver tries to close all loops given the current configuration. If individual gencoords are changed (by entering values, moving the slider, or using the gencoord keys in the model window), loops are solved keeping that gencoord value fixed (to the value you just set it to). Whenever the solver cannot find a solution with the exact gencoord value, it will try to find the solution with the closest gencoord value. In cases where no solution can be found, an error message will pop up and the model will be displayed with loops broken. If the restraint functions are poorly defined, the solver may be unable to find a solution because the errors are too large. An error message will be displayed alerting you of this. If there are no loops in your model, this button has no effect.

If your model has closed loops and constraints objects that involve the same gencoords, the IK solver will solve the loops and constraints simultaneously. That is, when a gencoord value is changed, or joint kinematics are modified, the IK solver will adjust all of the other gencoords in order to enforce all loops and constraint objects at the same time. In some cases, having loops and constraint objects that involve the same gencoords may make finding a solution more difficult because there are more constraints on the gencoords that can be adjusted. If this problem arises, you may want to “soften” the constraint objects by adjusting the *weight* of the constraint points, as described in Section 2.13.6, Constraint Point Attributes.

Note: if you turn off the IK solver using the *inverse kinematics* toggle button, SIMM will still enforce the constraint objects.

*loop\_tolerance* This parameter specifies the "acceptance level" for the solutions that the solver module reaches when implementing the loop constraints. It does not affect the speed or accuracy of the loop solving. Rather, it is a cutoff value specifying whether or not the solution is acceptable. If it is not acceptable, a dialog box may be presented indicating that the loop constraint is not satisfied (this notice appears only when loading or editing a model, not when changing gencoord values). This tolerance value corresponds to the distance (in model units) between the two segments of the loop joint (see Section 3.3.3, Joints, for an explanation of the loop joint). When the loop is properly enforced, the points on the two segments are coincident, and the distance is zero. When the loop is not properly enforced, the loop joint disarticulates and the two points are separated by a certain distance. This parameter specifies how large a distance is considered acceptable. Its value can also be specified in the joint file, as described in Section 3.3.

### 2.8.6 Restraint Function Plot Area

If the current gencoord has a restraint function, it is displayed in the function plot area. If the restraint function is inactive it will be shown in gray. For an active function, the control points are displayed in yellow, the curve in white, and the extrapolated parts of the curve in pink. The gencoord range and current value are also displayed in the plot area. Two red vertical lines show the start and end values of the gencoord range, and the yellow vertical line shows the gencoord's current value. When the IK solver is on, it uses the restraint functions to calculate the residuals for each gencoord (but only when the function is active and the gencoord is unclamped). The IK solver tries to find a solution in which all of the gencoord's residuals are zero. Tolerances showing the allowable range for the restraint function values (values which result in accept-

able residual errors) are shown by blue horizontal lines in the plot (these lines only appear when you zoom in close enough). The solver will never calculate a loop configuration with a gencoord value where the restraint function is outside this tolerance envelope. *When a gencoord is clamped, the IK solver does not use the restraint function to calculate residuals for the gencoord. Instead, it uses a different function that guarantees that the gencoord will not go outside its range of motion.*

When a function is displayed in the plot area, you can add, delete, and move its control points. To move a control point, press the left mouse button while the cursor is on the point. The region behind the point you select is shaded to indicate where you can move it. You cannot move a control point so that it overlaps one of its neighbors.

As you move a control point, its coordinates are shown just above the plot area to help you position the point. When you release the left mouse button, the point remains at its new location. To help you edit a restraint function, you can press the `i`, `o`, `l`, `r`, `u`, and `d` keys to move the view of the function in, out, left, right, up, and down, respectively. You can also use the `x` and `y` keys to zoom in and out along just the X or Y axes (press the `Shift` key along with `x` or `y` to zoom out).

Note: If you create a restraint function that has steep slopes or sharp corners, the display of the function may look choppy. That is, you may be able to notice the series of line segments that is used to approximate the curve in the plot area. This piece-wise linear approximation is for display purposes only. When SIMM needs a Y-value for the function (given an X-value, which is a generalized coordinate), it finds an exact solution using the algebraic form of the cubic spline.



## 2.9 Plot Viewer

The Plot Viewer lets you change the views of the plots. With this tool you can zoom in on a particular region of a plot curve, as well as display the coordinates of any point in a plot window. Like the Model Viewer (see Section 2.4, Model Viewer), some of the Plot Viewer commands are available from within the plot windows. These *plot window commands* are described in Section 2.9.3.

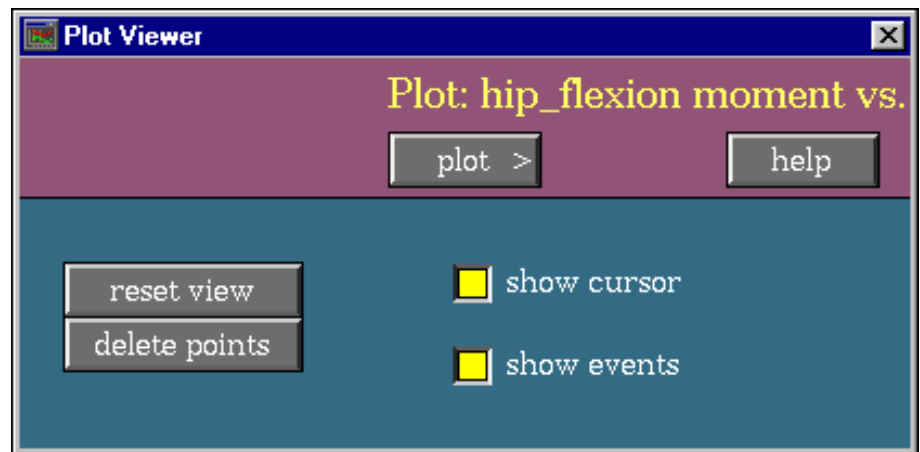


Figure 2-6. Plot Viewer window

### 2.9.1 Selector Menu



The plot selector lets you choose the current plot, which is the one affected by the **reset view** and **delete points** commands. All other viewing commands are performed in the plot windows, so the plot selector does not affect them.



Clicking on the help button opens a window containing helpful text about the Plot Viewer. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.9.2 Command Menu

**reset view**

This command resets the view of the current plot to its initial view. The plot curves are zoomed and/or translated so that they are all visible within the window.

**delete points**

This command deletes all of the cross-hair points from the current plot window. See Section 2.9.3 for directions on how to display cross-hair points.

## 2.9.3 Toggle Buttons

*show cursor*

This command toggles the display of the cursor in the current plot. The cursor is the vertical bar in the plot that shows the current value of the motion variable that was used in creating the plot. The cursor is visible only if the plot is a function of some motion, and if the motion file header contains the “cursor” parameter (see Section 3.5, Motion Files, for details on specifying a plot cursor).

*show events*

This command toggles the display of motion events. Motion events are vertical bars in plots that mark certain events in the motion that was used to create the plot. Motion events are visible only if the plot is a function of some motion, and if the motion file header contains one or more “event” parameters (see Section 3.5, Motion Files, for details on specifying a motion events).

## 2.9.4 Plot Window Commands

**I U L**  
**O D R**

These keys allow you to zoom and pan within the plot windows just like you can in the model windows. The **i** and **o** keys move the plot curves **in** and **out**, the **u** and **d** keys move the curves **up** and **down**, and the **l** and **r** keys move them **left** and **right**.



These keys allow you to zoom independently in the X and Y directions within the plot window. Pressing the `x` key will zoom in (in the X direction) on a part of the plot, and pressing the `x` key and the `Shift` key will zoom out (and similarly for the `y` key).

*left mouse button*

When you click the left mouse button in a plot window, a cross-hair is placed at the cursor location, and the coordinates of that point are displayed next to it. You can place up to 30 cross-hair points in each plot window. As you zoom in on a particular cross-hair using the `i` key, the precision of the displayed coordinates is increased so that you can more accurately note the point's location.

## 2.10 Wrap Editor

The Wrap Editor allows you to create and edit muscle wrapping primitives (wrap objects). Wrap objects provide automatic muscle wrapping without the need to explicitly specify intermediate muscle points (a.k.a. via points). Muscle via points are described in Section 3.4.1, Muscles.

Each wrap object is associated with (fixed to) a specific body segment in the model, and its position is specified relative to the reference frame of that segment.

A wrap object may have one or more muscles assigned to it. When muscles assigned to a wrap object make contact with the wrap object's surface, the muscle is automatically deflected to prevent it from penetrating the wrap object.

A muscle may have more than one wrap object assigned to it. Each time a wrap object is assigned to a muscle, the object is added to the end of the muscle's list of wrap

objects. When SIMM wraps a muscle over its assigned objects, the wrap objects are processed in the order in which they appear in this list. The resulting path of the muscle may depend on the order of processing, so care must be taken to define them in the appropriate order. To see the ordered list of wrap objects assigned to a given muscle, you must open the Muscle Editor tool and turn on the parameters for that muscle. The only way to change the order of the list is to unassign the wrap objects and then re-assign them in the desired order.

Wrap objects can take the form of spheres, cylinders, ellipsoids, or torii. Additionally, muscle wrapping can be constrained to a particular half of the wrapping primitive (*i.e.*, a hemisphere, hemicylinder, hemiellipsoid, or hemitorus).

### 2.10.1 Selector Menu



The model selector lets you choose which model's wrap objects you wish to edit. When you select a new model with the model selector, the Wrap Editor window is redrawn with the new model's settings. If the current model has no wrap objects in it, then the Wrap Editor window will be blank except for the **wrap object** button which will let you add new wrap objects to the model.



Clicking on the help button opens a window containing helpful text about the Wrap Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.10.2 Command Menu



Select this command to create a new wrap object or to select an existing wrap object for editing. When you select

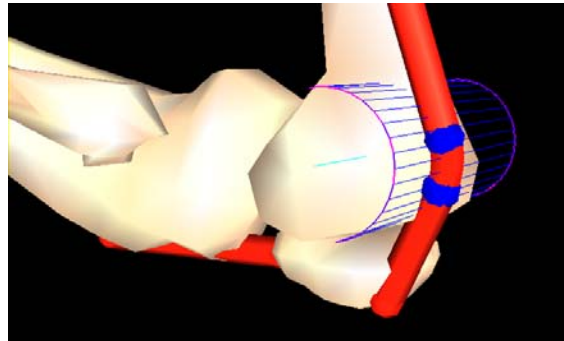


Figure 2-7. Cylindrical Muscle Wrap Example

this item a pop-up menu of existing wrap objects is displayed. The first item in the pop-up menu, *new wrap object*, creates a new wrap object named *Wrap\_1*, adds it to the current model, and makes it the current wrap object for editing. The remaining items in the pop-up menu allow you select existing wrap objects for editing.

**segment** >

Select this command to change the body segment to which the current wrap object is attached (defined as the *parent* segment). When you select this item a pop-up menu of the model's segments is displayed. Choosing a segment from this menu will reassign the current wrap object to the selected segment. The wrap object's overall position will not change when it is assigned to a new segment. However, when the model is animated, the wrap object will follow the movement of its new parent segment.

**muscles** >

This command lets you choose the set of muscles that wrap over the current wrap object. When you select this item, a pop-up menu of the muscle groups is displayed. Choosing a muscle group from this menu causes it to appear at the bottom of the Wrap Editor window. If the chosen muscle group is already displayed at the bottom of the window, then selecting it again will hide it. Selecting

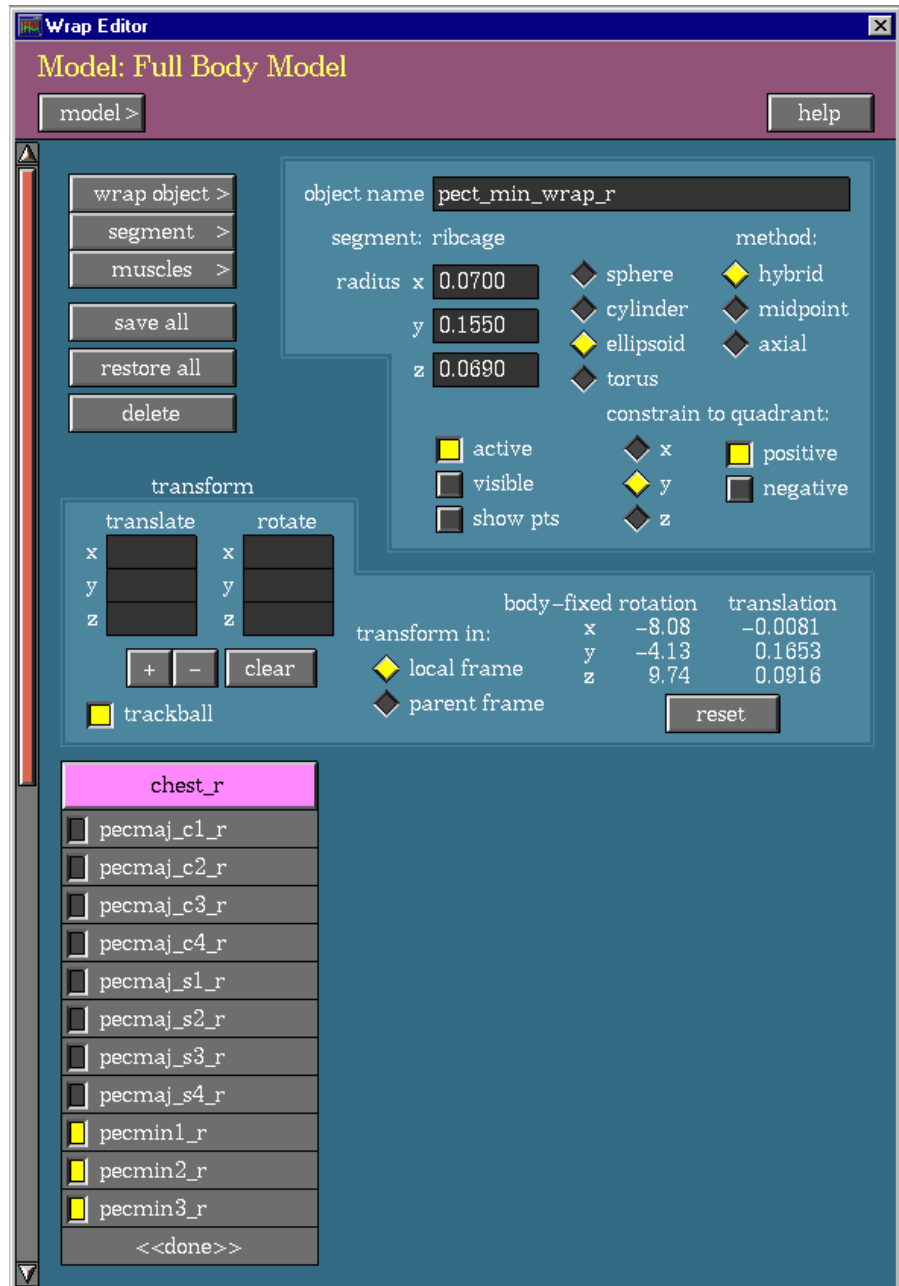


Figure 2-8. Wrap Editor window

individual muscles from these group menus will cause the muscles to wrap over the current wrap object. The muscles will continue to wrap even if you turn off the display of these muscle menus.

**save all**

This command saves the current state of *all* wrap objects in the model to a buffer. It can be used in combination with the **restore** button to help you undo changes made while editing wrap objects.

**restore all**

This command restores the most recently saved copy of *all* of the current model's wrap objects. It can be used in combination with the **save** button to help you undo changes made while editing wrap objects.

**delete**

This command deletes the current wrap object from the model.

### 2.10.3 Wrap Object Attributes

The panel to the right of the Wrap Editor's command menu can be used to edit the current wrap object's attributes. The previous section describes how to assign muscles to wrap objects and wrap objects to body segments. The controls described in this section allow you to edit the other parameters of the current wrap object.

*object name* This field contains the current wrap object's name. Click and type in this field to rename the wrap object.

*segment* This field displays the name of the parent segment of the current wrap object. To change the current wrap object's parent segment, click the **segment** button to the right of the field.

*radius, height* These fields let you specify the size of the current wrap object. For spherical wrap objects there is only one field

to specify: the sphere's radius. For cylinders there are two fields for the cylinder's radius and height. For ellipsoids there are three fields to specify the ellipsoid's X, Y, and Z radii. For torus objects there are two fields for the inner and outer radii.

*sphere, cylinder, ellipsoid, torus*

These radiobuttons let you choose the current wrap object's shape. Typically you will choose the shape that best resembles the surface you intend to wrap muscles over. Spheres and cylinders are simpler to define, and more straightforward for SIMM to wrap muscles over. Ellipsoids are more complicated, and there are three techniques available for wrapping muscles over them (see Section 2.10.5 for a description of these methods). Torii are very useful as pulleys, but they work best if the muscle path remains roughly perpendicular to the plane of the torus.

*hybrid, midpoint, axial*

These radiobuttons let you choose which method is used to compute the wrapping path around ellipsoidal wrap objects. If the current wrap object is not an ellipsoid, then these radio-buttons will not be visible in the Wrap Editor window. See Section 2.10.5 for more information about the hybrid, midpoint, and axial wrapping methods.

*active*

This toggle button lets you turn the current wrapping object on and off. When a wrap object is not active, it is drawn in red, and it has no effect on muscles that intersect it.

*visible*

This button lets you toggle the visibility of the current wrap object in the model window. Once you are done editing a wrap object, you will typically make it invisible so that it does not obscure the display of other features of your model. A wrap object can be invisible, but still active.



- show pts* This button toggles the visibility of muscle tangent points for muscles wrapping over the current wrap object. When checked, the X, Y, Z coordinates of the tangent points will be displayed in the model window.
- constrain to quadrant* These radiobuttons and toggle buttons are used to specify that wrapping be constrained to a particular section of the current wrap object. The *x*, *y*, and *z* radiobuttons let you specify the constraint axis, and the *positive* and *negative* toggle buttons let you specify the constraint direction. For example, if the *x* radiobutton is selected with the *positive* toggle button, then wrapping would be constrained to the half of the wrapping primitive in which all x-coordinates are greater than zero (in the wrap object's local reference frame). When using these buttons, you may need to rotate the wrap object within its local reference frame in order to properly orient the active section of the wrap object.

#### 2.10.4 Transforming Wrap Objects

The collection of controls labeled *transform* can be used to move and rotate the current wrap object within its parent segment's frame of reference. The current wrap object's transformation can be modified interactively in the model window, or it can be modified precisely by entering values in the *translate* or *rotate* fields of the Wrap Editor window.

##### **INTERACTIVE WRAP OBJECT TRANSFORMATION**



To interactively transform the current wrap object, hold down the *w* key then click and drag in the model window. The behavior of the left, middle, and right mouse buttons depends on the setting of the *trackball* toggle button at the bottom-left corner of the Wrap Editor window.

*trackball* If the *trackball* toggle button is checked, then the left mouse button will move the current wrap object left, right, up, and down in the plane of the computer screen. The middle mouse button will move the wrap object in and out, perpendicular to the screen. Finally, the right mouse button will rotate the wrap object as if it were attached to a virtual trackball.

If the *trackball* toggle button is not checked, then the left mouse button will rotate the current wrap object about the X-axis, the middle mouse button will rotate it about the Y-axis, and the right mouse button will rotate it about the Z-axis. Furthermore, if the *transform in local frame* radiobutton is selected then the wrap object will rotate about its local XYZ axes. If the *transform in parent frame* radiobutton is selected then the wrap object will rotate about its parent segment's XYZ axes.

#### **NUMERICAL WRAP OBJECT TRANSFORMATION**

*+*, *-*, *clear* To apply precise transformations to the current wrap object, first enter the distances and angles to transform by in the *translate* and *rotate* fields, then click the *+* button below the text fields one or more times to apply the transform. If the *transform in local frame* radiobutton is selected then the specified transform will be applied relative to the wrap object's local XYZ axes. Otherwise the transform will be applied relative to the wrap object's parent segment's XYZ axes. To apply the inverse transform, click the *-* button. The values you enter into these fields remain there even after they are applied. To clear all values from the *translate* and *rotate* fields, click the **clear** button.

**CURRENT TRANSFORM DISPLAY**

- transform in* The wrap object's current transform is displayed in the lower right corner of the Wrap Editor window as a sequence of XYZ rotations and an XYZ translation. The order of the rotation sequence that is displayed is X, followed by Y, followed by Z. This order cannot be changed. If the *transform in local frame* radiobutton is selected, then the rotations displayed are about the wrap object's local XYZ axes (a.k.a. body-fixed rotation). If the *transform in parent frame* radiobutton is selected, then the rotation sequence displayed is about the wrap object's parent segment's XYZ axes (a.k.a. space-fixed rotation).
- reset* Click the **reset** button below the current transform display to return the current wrap object to the origin of its parent segment's reference frame. The wrap object's transform is reset to the identity matrix.

### 2.10.5 Ellipsoid Wrapping Methods

To overcome numerical instabilities involved with computing the optimal path over the surface of an ellipsoid, SIMM allows you to choose between three different algorithms for calculating muscle paths over ellipsoidal wrap objects. The default method, *hybrid*, works well for most muscles. However, if you notice erratic muscle motion (e.g., skips or jumps when wrapping over an ellipsoidal wrap object) or your plots of muscle moment arms are not as smooth as you'd like, you should be able to achieve a smooth motion by experimenting with the *midpoint* or *axial* algorithms. In rare cases you may need to change the position or orientation of your wrap object to achieve smooth wrapping motion.

### **THE HYBRID WRAPPING METHOD**

The hybrid method chooses a wrapping path by computing a weighted average of the results of the other two 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 this situation you can use the wrap method radiobuttons in the Wrap Editor or Muscle Editor tools to restrict the choice of wrapping algorithm to either the midpoint or axial methods, thereby avoiding artifacts created as the hybrid method blends between the midpoint and axial methods. In most cases, however, the hybrid method will produce the best wrapping result.

### **THE 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.

### **THE AXIAL WRAPPING METHOD**

The axial method chooses one of the ellipsoid's principal axes, X, Y, or Z, 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.

### **2.10.6 Wrapping Tips & Techniques**



Use the *constrain to quadrant* controls to specify which side of a wrap object muscles will wrap over. If you do not constrain wrapping to one side of a wrap object, then muscles will automatically wrap over the side that permits the shortest muscle path. By constraining the wrap object, you

can force muscles to wrap over the desired side of the object even if it is not the shortest path around the object. However, when calculating a muscle path other than the true shortest path, SIMM may occasionally calculate an undesirable muscle path as it attempts to wrap over the active quadrants of the wrap object. If this occurs, you should rotate or translate the wrap object until the muscle paths behave as desired.



The radiobuttons labeled *local frame* and *parent frame* serve multiple purposes:

- (1) For transforms that are applied by numerical entry, they determine which reference frame the transform is relative to.
- (2) For transforms that are applied interactively when the *trackball* toggle button is unchecked, they determine which reference frame's major axes will be rotated about.
- (3) They control whether the wrap object's current transform is displayed as a sequence of body-fixed or space-fixed rotations.

## 2.11 Deform Editor

The Deform Editor allows you to twist, bend, and warp portions of a body segment that would normally be considered rigid. When a segment is deformed, all bone vertices, distal joint centers, muscle attachment points, and muscle wrapping objects in the deformed region are transformed accordingly. In this way the effect of a bone deformity can be examined and measured using SIMM's other tools.

### ***Modeling Bone Deformities in SIMM***

Bone deformities are simulated in SIMM by assigning one or more deform objects to a body segment. A deform object is defined by two regular six-sided boxes, an inner box and an outer box. These two boxes are attached to a portion of the segment to determine which region of the segment will be deformed. Once the boxes are attached to the segment, the inner box may be freely transformed while the outer box remains stationary. As the inner box is moved or rotated, all bone vertices, distal joint centers, muscle points, and wrap objects associated with the segment that lie within the inner box follow its movement exactly. Elements of the body segment that lie between the walls of the inner and outer boxes experience a gradually decreasing amount of deformation. Elements outside the outer deform box remain unaffected.

Complex deformities can be modeled by attaching multiple deform objects to a body segment. Each deform object is responsible for deforming the particular region of the segment that it surrounds. Taken together the deform objects act in a sequence, the output of the first deform becoming the input of the second deform, and so on. Interactive transformation of multiple deform objects can be coordinated by defining a deformity object in the model's joint file. A deformity is simply a description of how to manipulate multiple deform objects at once. Each deformity has an associated slider added to the bottom of the Deform Editor window. Dragging a deformity's slider side-to-side causes all deform objects associated with the deformity to move through their range of motion at the same time.

### 2.11.1 Selector Menu



The model selector lets you choose which model's deform objects you wish to edit. When you select a new model with the model selector, the Deform Editor window is redrawn with the new model's settings. If the current model has no deform objects in it, then the Deform Editor window will be blank except for the **segment** and **deform**

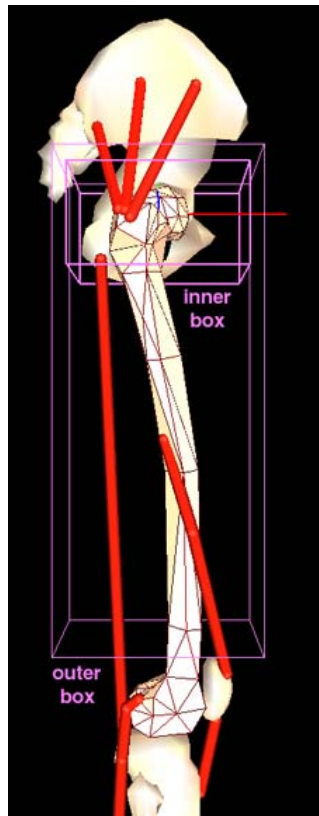


Figure 2-9. Attaching Deform Boxes

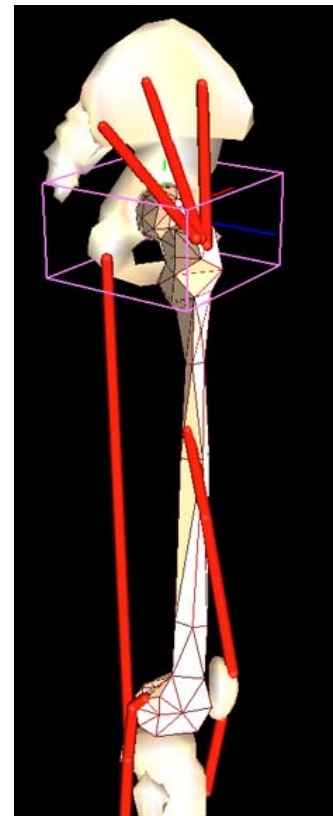


Figure 2-10. Applying Deformation



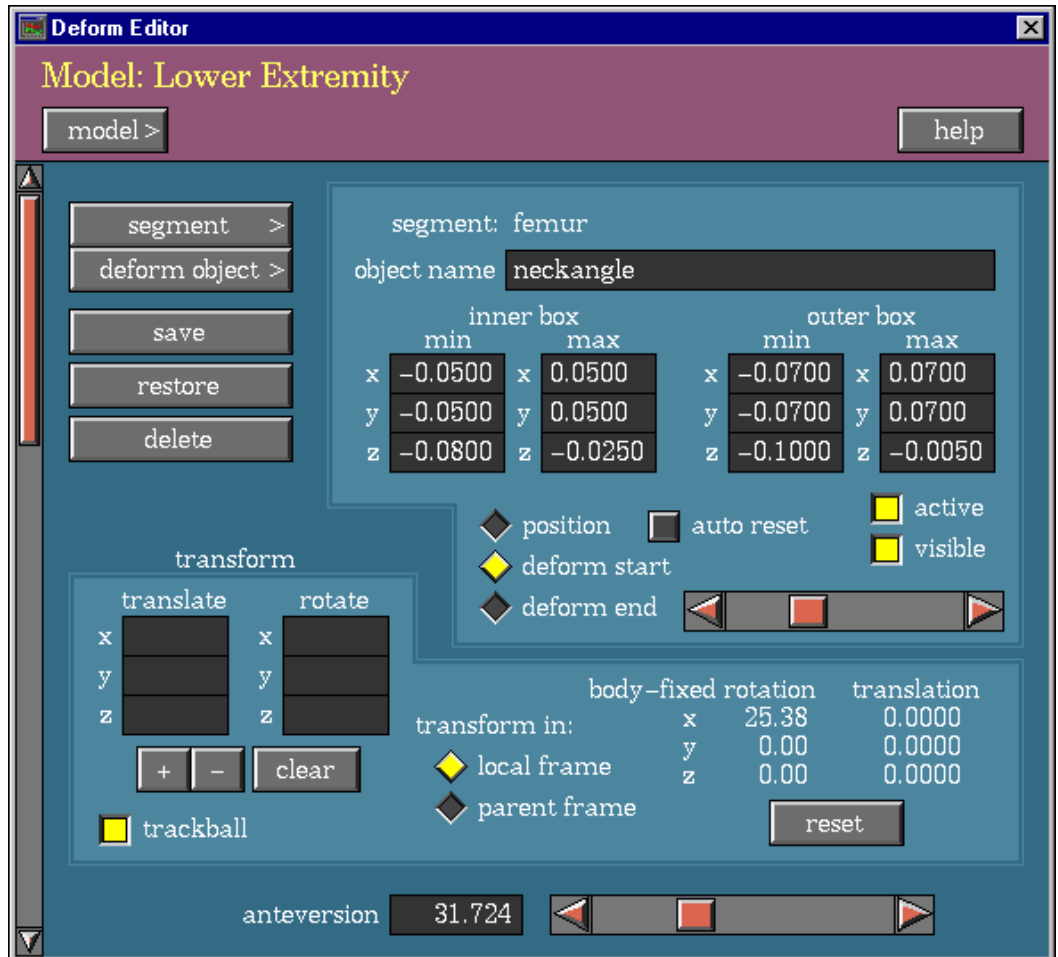


Figure 2-11. Deform Editor window

**object** buttons which will let you add new deform objects to the model.



Clicking on the help button opens a window containing helpful text about the Deform Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.11.2 Command Menu



Select this command to specify the current segment for the Deform Editor. Any newly created deform objects will be added to this segment. The name of the current segment is displayed to the right of the button.



Select this command to create a new deform object or to choose an existing deform object for editing. When you click this button a pop-up menu of all deform objects assigned to the current segment is displayed. The first item in the pop-up menu, *new deform object*, creates a new deform named *Untitled*, adds it to the current body segment, and makes it the current deform object for editing. The remaining items in the pop-up menu allow you select existing deform objects for editing.



This command saves a snapshot of the current state of all deform objects in the model. It can be used in combination with the **restore** button to help you undo changes made while editing deform objects.



This command restores the most recently saved snapshot of the current model's deform objects. It can be used in combination with the **save** button to help you undo changes made while editing deform objects.



This command deletes the current deform object from the model.

### 2.11.3 Deform Object Attributes

The panel to the right of the Deform Editor command menu can be used to edit the current deform object's attributes.

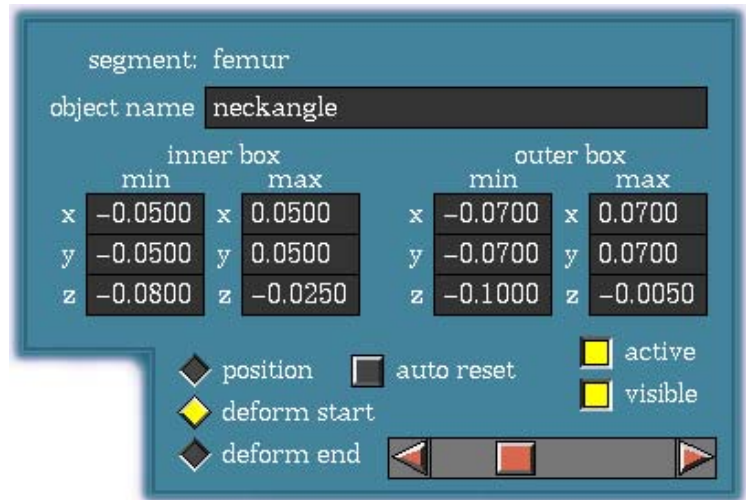


Figure 2-12. Deform Object Attributes Panel

- segment* This field displays the name of the Deform Editor's current body segment. Any new deform objects will be added to this segment. You can use the segment button to the left of this field to change the current segment.
- object name* This field contains the current deform object's name. Click and type in this field to rename the deform object.
- inner box* These six fields contain the minimum and maximum X, Y, and Z dimensions of the deform object's inner box. Change these values to modify the size and shape of the current deform's inner box.
- outer box* These six fields contain the minimum and maximum X, Y, and Z dimensions of the deform object's outer box. Change these values to modify the size and shape of the current deform's outer box.
- position*  
*deform start*  
*deform end* These radio buttons specify the Deform Editor's current mode. In *position* mode you can move and rotate both the

inner and outer deform boxes together to specify how they are attached to the current body segment. In *deform start* mode you can specify the inner box's starting transformation. In *deform end* mode you can specify the inner box's ending transformation.

The slider to the right of the *deform end* radiobutton allows you to smoothly move the inner deform box between its starting and ending positions.

- auto-reset* This toggle button allows you to override the current deform object's standard behavior by making it automatically restore the current segment's origin to its original, undeformed, position. For more information on auto-reset deforms, see Section 2.11.5, Combining Multiple Deform Objects.
- active* This toggle button lets you turn the current deform object on and off. When a deform object is deactivated it has no effect on bones, joints, muscle points, or wrap objects within it.
- visible* This button lets you toggle the visibility of the current deform object in the model window. When the Deform Editor is in *position* mode both the inner and outer boxes are displayed for all visible deforms in the current segment. When the Deform Editor is in *deform start* or *deform end* modes only the inner box is displayed for all visible deforms in the current segment. Typically once you are done editing a deform object you will make it invisible so that it does not obscure the other features of your model.

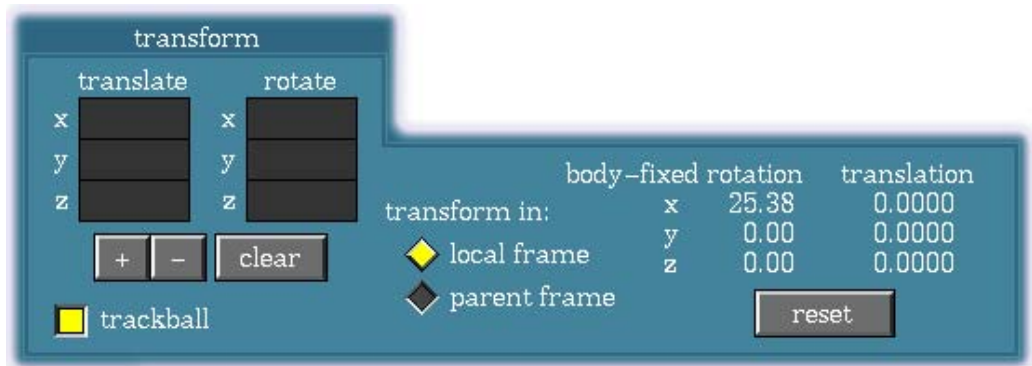


Figure 2-13. Deform Object Transform Panel

### 2.11.4 Transforming Deform Objects

When adding deform objects to a model, most of your time will be spent choosing the proper size and placement for each object. The inner box and outer box fields described in the preceding section allow you to specify the size and shape of the current deform object. In addition to specifying the size and shape of your deform objects, you will need to transform them into position, and then transform them again to actually apply a deformation to the model.

The collection of controls labeled *transform* can be used to position the current deform object within its parent segment's frame of reference (in *position* mode), and to apply a deform by moving and rotating the deform object's inner box (in *deform start* and *deform end* mode). The current deform object's transformation can be modified interactively in the model window, or it can be modified precisely by entering values in the *translate* or *rotate* fields of the Deform Editor window.

**Interactive Deform Object Transformation**

- E** To interactively transform the current deform object, hold down the  $\epsilon$  key then click and drag in the model window. The behavior of the left, middle, and right mouse buttons depends on the setting of the *trackball* toggle button at the bottom-left corner of the Deform Editor window, as described below.

*trackball* If the *trackball* toggle button is checked, then the left mouse button will move the current deform object left, right, up, and down in the plane of the computer screen. The middle mouse button will move the deform object in and out, perpendicular to the screen. Finally, the right mouse button will rotate the deform object as if it were attached to a virtual trackball.

If the *trackball* toggle button is not checked, then the left mouse button will rotate the current deform object about the X-axis, the middle mouse button will rotate the deform object about the Y-axis, and the right mouse button will rotate the deform object about the Z-axis. Furthermore, if the *transform in local frame* radiobutton is selected then the deform object will rotate about its local XYZ axes. If the *transform in parent frame* radiobutton is selected then the deform object will rotate about its parent segment's major axes.

**Numerical Deform Object Transformation**

*+*, *-*, *clear* To apply precise transformations to the current deform object, first enter the distances and angles to transform by in the *translate* and *rotate* fields, then click the *+* button below the text fields one or more times to apply the transform. If the *transform in local frame* radiobutton is selected then the specified transform will be applied relative to the deform object's local XYZ axes. Otherwise the transform will be applied relative to the deform object's

parent segment's XYZ axes. To apply the inverse transform, click the **-** button. To clear all values from the translate and rotate fields, click the **clear** button.

### **Current Transform Display**

*transform in*

The deform object's current transform is displayed in the lower right corner of the Deform Editor window as a sequence of XYZ rotations and an XYZ translation. The order of the rotation sequence that is displayed is X, followed by Y, followed by Z. This order cannot be changed. If the *transform in local frame* radiobutton is selected, then the rotations displayed are about the deform object's local XYZ axes (a.k.a. body-fixed rotation). If the *transform in parent frame* radiobutton is selected, then the rotation sequence displayed is about the deform object's parent segment's XYZ axes (a.k.a. space-fixed rotation).

*reset*

Click the **reset** button to restore the current deform object's transform to identity. In *position* mode this will realign the deform object's reference frame with its associated body segment's reference frame. In *deform start* or *deform end* modes this will realign the deform object's inner box with its outer box.

Note: If you hold down the **Alt** or **Ctrl** key while clicking the *reset* button, each of the deformity sliders at the bottom of the Deform Editor window will be reset to their default values.

## **2.11.5 Combining Multiple Deform Objects**

To simulate complex bone deformities you will typically need to apply more than one deform object to a body segment. Using multiple deform objects allows you to apply different types of deformation to different regions of the

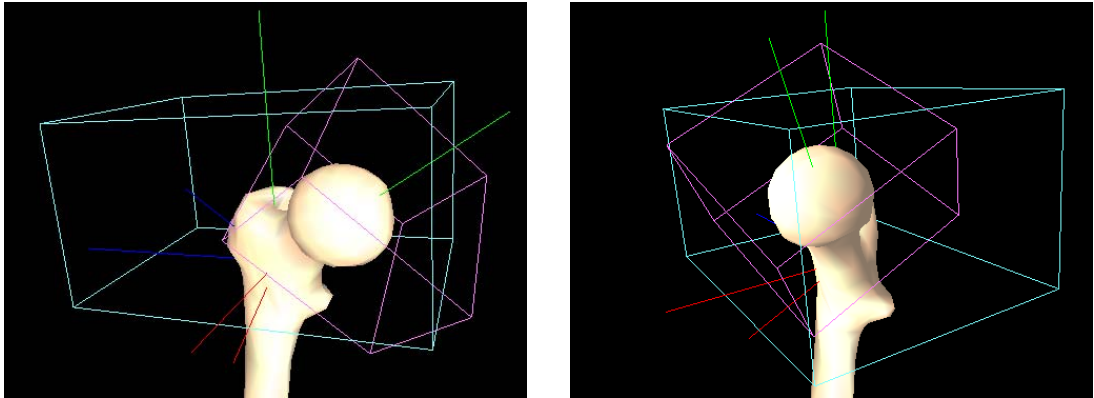


Figure 2-14. The left image shows the femur and deform boxes before any deformation has taken place. The right image shows the result of the first deform, represented by the larger box.

same segment. When working with multiple deform objects, it is important to keep in mind that the overall deformation applied to a body segment is the result of each of the individual deformations applied in sequence, starting with the first deform object in the segment's list. The order in which deform objects appear in their parent segment's list is significant. Rearranging a segment's list of deforms will change the aggregate deformation applied to the segment. Furthermore, the only way to rearrange a segment's list of deform objects is to edit the model's joint file, then reload the model. For these reasons it is helpful to have an understanding of how multiple deforms behave.

In Figure 2-14 above, the larger box in both images represents a deform object that twists the femur about its long axis. We call this deform object *Anteversión*. The smaller magenta box represents a deform object that bends neck of the femur up and down. We call this deform object *Neck-Shaft Angle*. In this example, *Anteversión* is the first deform in the list, followed by *Neck-Shaft Angle*. Notice



in the picture on the right how the smaller *Neck-Shaft Angle* box has followed the twisting rotation of the larger *Anteversio*n box. The reference frame of each deform box is itself deformed by any deform objects that precede it in the segment's list of deforms. This allows each deform box to “stick to” the region of bone that it was originally assigned to as deforms are applied to the segment. This also implies that the order in which deforms are applied contributes to the overall deformation of the body segment.

### **Auto-Reset Deform Objects**

A special type of deform object called an auto-reset deform is available to simplify the task of maintaining the spatial characteristics of a deformed segment's proximal joint. Auto-reset deforms were designed specifically to keep the ball-and-socket joint at the hip intact as the femur undergoes deformation. However auto-reset deforms are not limited to the femur, they may be used to preserve the proximal joint of any deformed segment.

In general terms, an auto-reset deform ignores its *deform start* and *deform end* transformations and automatically computes a deforming transformation that restores the segment's origin to its original, undeformed position. If the segment's reference frame is defined such that its origin coincides with the center of its proximal joint, then the result of an auto-reset deform will be to restore the elements within its inner box to their original, undeformed position. For an auto-reset deform to behave correctly the following conditions must be met:

- (1) the origin of the body segment must coincide with the center of the segment's proximal joint,
- (2) the inner box of the auto-reset deform must contain the origin of the body segment, and

- (3) the auto-reset deform must be the last deform object in the segment's list of deforms.

In the hip joint example, the femur segment would be modeled such that its reference frame and proximal joint were centered within the femoral head. An auto-reset deform whose inner box surrounded the femoral head would be added to the end of the femur's deform list. The auto-reset deform's outer box would surround all of the femur except the distal joint. With this setup, the auto-reset deform would move and rotate the femoral head back into the acetabulum with its deforming effect gradually decreasing along the length of the femur. The knee joint would remain unaffected by the auto-reset deform because it would lie outside of the auto-reset deform's outer box.

By default an auto-reset deform will apply both translation and rotation to restore the segment's origin to its original position and alignment. However, it is possible to specify that only translation be applied by including the `translationonly` keyword in the deform object's definition in the joint file. In this case the segment's origin is moved back to its original position, but the segment's reference frame may remain rotated from its original axes. There is no way to change the translation-only behavior of an auto-reset deform interactively within SIMM. See the Section 3.3.10, Deform Objects, for more information about how to specify the `translationonly` keyword in the joint file.

### ***Using Deformity Sliders***

Deformity sliders can be defined in a SIMM joint file as a way of coordinating the state of several deform objects with a single slider control. Each deformity in the joint file will appear as a slider at the bottom of the Deform Editor window.



Figure 2-15. Deformity Slider: Anteversion

The range and default value for a deformity are completely arbitrary and may be specified in the deformity's definition in the model's joint file. If the range is not specified it defaults to a minimum of 0.0 and a maximum of 100.0. If the default value is unspecified it defaults to 0.0. As you drag a deformity slider side-to-side, each of the deform objects associated with the deformity will be adjusted such that their individual deform percentages match the deformity slider's percentage. Holding down the `Alt` or `Ctrl` keys and clicking the *reset* button just above the deformity sliders will automatically reset all deformity sliders to their default values.

In addition to controlling multiple deform objects, you may find it useful to set up deformity sliders to control individual deforms. This allows you to specify a meaningful range of values for a specific deform, control the deform's default value when the model is first loaded, and adjust the deform object's percentage without having to make it the current deform object.

To create or modify deformity sliders you must edit the model's joint file manually, then reload the model. There is no way to add, remove, or modify deformity sliders interactively within SIMM.

### 2.11.6 Deform Tips, Techniques, and Caveats



The following list describes each element of a SIMM model that is deformed when it falls within the outer box of a deform object:

- (1) *bone surfaces*: Bone vertices for all bones in the segment are deformed. However, bone vertex normals are not deformed. Therefore you may notice irregular shading of deformed bone surfaces.
- (2) *distal joint centers*: The origin of each joint emanating from the segment is deformed. However, the joint's DOFs and kinematic functions are not deformed.
- (3) *muscle attachment points*: Each muscle point attached to the segment is deformed.
- (4) *muscle wrapping objects*: The origin of each wrap object associated with the segment is deformed. However, the surface of the wrap object is not deformed. In other words, wrap objects are moved and rotated to follow a deformation, but the wrap object surface retains its shape (*i.e.*, sphere, cylinder, or ellipsoid).



Keep in mind that deform objects are always associated with a specific body segment. A deform object will only affect elements that are also associated with that segment. Deformity sliders, on the other hand can control any combination of deform objects, and thus are not tied to a particular body segment.



When moving muscle points, be careful not to move muscle points that are currently being deformed. Although the muscle point may appear to follow the motion of the mouse correctly, the next time you make a change to any

deform object that is affecting the muscle point, the point will jump to a new location.

## 2.12 Marker Editor

The Marker Editor lets you edit the markers in your musculoskeletal model. Markers are used by the Motion Module to map motion capture data onto your model. If you do not use the Motion Module to import TRC or TRB motion files into SIMM, then you do not need to use markers or the Marker Editor. TRC and TRB files contain 3D coordinates for the markers that were placed on the subject while a motion was recorded. These files thus contain a series of “marker clouds,” each one representing the 3D coordinates of all the markers on the subject at a particular instant of time during the motion. The Motion Module reads these marker data and fits the SIMM model within the marker cloud for each time frame. By placing markers on the SIMM model that match the names and positions of the markers placed on the subject, the Motion Module is able to adjust the model’s gencoord values to determine a “best fit” of the model to the marker cloud. The result is a SIMM motion that matches the TRC or TRB motion.

### 2.12.1 Selector Menu



The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Marker Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

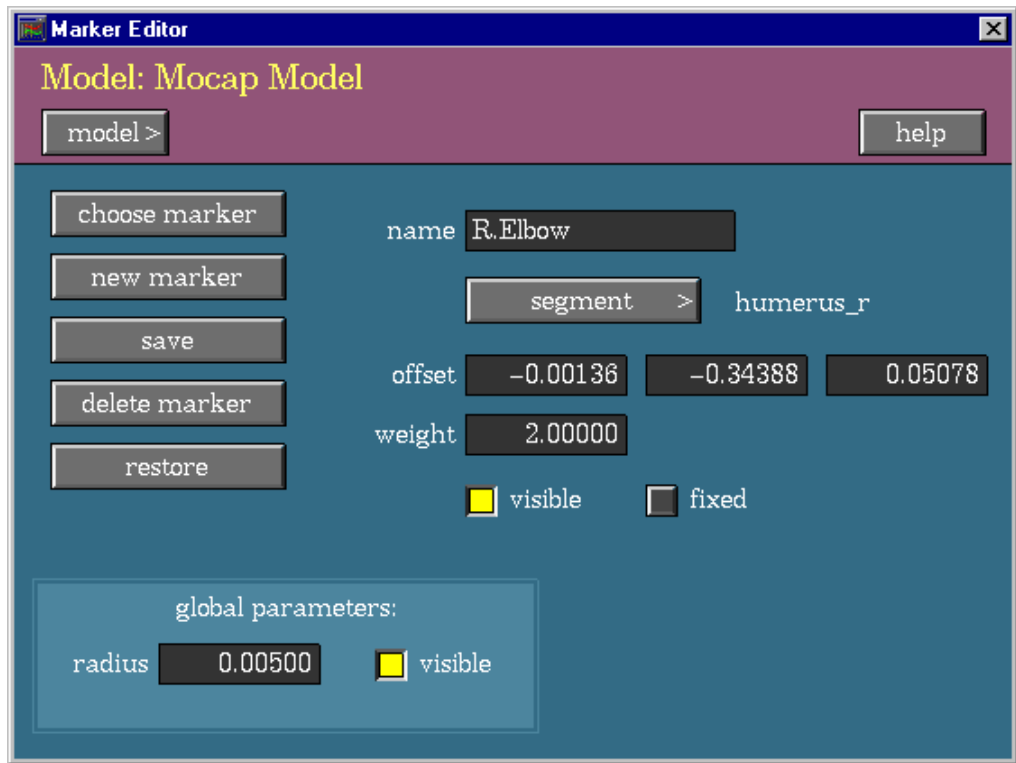


Figure 2-16. Marker Editor window

## 2.12.2 Command Menu

**choose marker**

This command lets you select the marker you want to edit. When you choose a marker from the pop-up menu, that marker is highlighted in the marker window. The parameters for the chosen marker are displayed in the Marker Editor window.

**new marker**

This command creates a new marker in the current model. It is given default values for all its parameters, which can then be changed as described in Section 2.12.3.

**save**

This command saves all of the markers in the current model. It saves them to a buffer so that you can restore them if you make changes that you later want to undo. When you first load a model, SIMM saves copies of all the markers, so you can restore them back to their original form without having to save them first.

**delete marker**

This command deletes the currently selected marker.

**restore**

This command restores all of the markers in the current model. The markers are restored from a buffer that was saved last time you used the **save** command. When you first create a model, SIMM saves copies of all of the markers, so you can restore them back to their original form without having to save them first.

### 2.12.3 Marker Parameters

*name*

The names of the markers in your model should exactly match the names of the markers in the tracked marker files that you load into the model (although case does not matter). For models that are used as *mocap models*, the names of the markers must match the marker names in the marker sets that OrthoTrak recognizes. See Chapter 5, Motion Module, for more information on *mocap models* and the OrthoTrak marker set.

*segment*

The segment button lets you change the body segment to which a marker is fixed. When you change a marker's segment, the marker remains in the same position with respect to ground. The link between the marker and the origin of the segment it is fixed to shows the change.

*offset*

These three fields contain the X, Y, and Z offsets from the origin of the segment to the marker. To change the offset,

simply type in new values. The units of the offsets are the implicit length units in your model (usually meters).

*weight* The weight of each marker determines how much emphasis is placed on it when the Motion Module fits the model to a frame of motion data. The higher the weight, the more the Motion Module tries to fit that marker to the data, at the expense of the other markers. If all the marker weights are the same (regardless of the actual weight value), the Motion Module treats all markers equally.

*visible* This checkbox controls whether or not the marker is visible in the model window. Even if the marker is not visible, it is still used by the Motion Module when importing motion files.

*fixed* This checkbox controls whether or not the marker is fixed. Fixed markers are optional markers whose X, Y, and Z offsets in the *mocap model* are not automatically calculated when the static trial is processed. Rather, the offsets in the marker definition in the joint file are used to position the marker on the model. These offsets are scaled with the body segment, however. Thus they are fixed to a certain region of the bone surface, even as the bone is scaled to fit the motion subject. The fixed / not fixed property of a marker is relevant only to the *mocap model*, since this property is used only when recalculating offsets during processing of the static trial.

## 2.12.4 Global Parameters

*radius* This field contains the radius of the markers in the model. It affects only the display of the markers in the model window; it does not affect any marker offset calculations that the Motion Module performs when loading a *mocap model*.



*visible* This checkbox controls the visibility of all of the markers in the model window. It is a convenience feature that allows you to turn all of the markers on or off without selecting them individually and changing their visibility. Even if the markers are not visible, they are still used by the Motion Module when importing tracked marker files.

## 2.13 Constraint Editor

The Constraint Editor allows you to create and edit constraints in the model. These constraints provide a way of constraining points on one body segment to remain on the surface of a geometric primitive (a.k.a. constraint object) on a second segment. This mechanism is useful for creating joint kinematics that are too complex to be modeled with a traditional SIMM joint. For example, in the shoulder, two (or more) points on the scapula can be constrained to remain in contact with the ribcage, which is represented as an ellipsoid. Once this constraint is activated, when you change the value of one of the generalized coordinates in the shoulder, SIMM will adjust the values of the other generalized coordinates in order to satisfy the constraints. This process is similar to the inverse kinematics algorithm that keeps loops together.

Each constraint is made up of a constraint object associated with a specific body segment in the model, and one or more constraint points on different body segments. A segment may have more than one constraint object attached to it. Constraint objects and points are defined within the reference frame of their associated segment and follow the movement of that segment. Constraint objects can take the form of planes, spheres, cylinders, or ellipsoids.

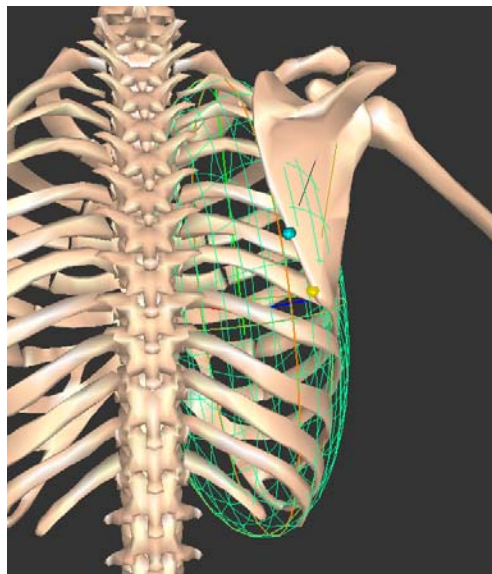


Figure 2-17. Sample Constraint. The constraint object is an ellipsoid, fixed to the ribcage. There are two constraint points, both fixed to the posterior edge of the scapula.

### 2.13.1 Selector Menu

model >

The model selector lets you choose which model's constraint objects you wish to edit. When you select a new model with the model selector, the Constraint Editor window is redrawn with the new model's settings. If the current model has no constraint objects in it, then the Constraint Editor window will be blank except for the **constraint object** button which will let you add new constraint objects to the model.

help

Clicking on the help button opens a window containing helpful text about the Constraint Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.13.2 Objects Command Menu

**constraint object >**

Select this command to create a new constraint object or to select an existing constraint object for editing. When you select this item a pop-up menu of existing constraint objects is displayed. The first item in the pop-up menu, new constraint object, creates a new constraint object, adds it to the current model, and makes it the current constraint object for editing. When a constraint object is created, a default point is created as well. The remaining items in the pop-up menu allow you select existing constraint objects for editing.

**segment >**

Select this command to change the body segment to which the current constraint object is attached (defined as the parent segment). When you select this item a pop-up menu of the model's body segments is displayed. Choosing a segment from this menu will reassign the current constraint object to the selected segment. The constraint object's overall position will not change when it is assigned to a new segment. However, when the model is put in motion the constraint object will follow the movement of the new segment to which it is attached. Constraint objects and points cannot both be assigned to the same body segment, so in the pop-up menu the segment[s] on which the points are defined will be grayed out.

**save all**

This command saves the current state of *all* constraint objects in the model to a buffer. It can be used in combination with the **restore all** button to help you undo changes made while editing constraint objects.

**restore all**

This command restores the most recently saved copy of *all* of the current model's constraint objects. It can be used in combination with the **save all** button to help you undo changes made while editing constraint objects.



Figure 2-18. Constraint Editor window

**delete**

This command deletes the current constraint object from the model.

### 2.13.3 Constraint Object Attributes

The panel to the right of the Constraint Editor's command menu can be used to edit the current constraint object's attributes.

*object name* This field contains the current constraint object's name. Click and type in this field to rename the constraint object.

*segment* This field displays the name of the parent segment of the current constraint object. To change the current constraint object's parent segment, click the **segment** button to the right of the field.

*radius, height* These fields let you specify the size of the current constraint object. For spherical constraint objects there will be only one field to specify: the sphere's radius. For cylinders there will be two fields, for the cylinder's radius and height. For ellipsoids there will be three fields to specify the ellipsoid's X, Y, and Z radii. For planes there will be two fields to specify, the size of the plane in the X and Y directions (this is for display only; the plane is infinite for the purpose of implementing the constraint).

*plane, sphere, cylinder, ellipsoid* These radiobuttons let you choose the current constraint object's shape. Typically you will choose the shape that most closely resembles the anatomical surface to which you want to constrain the points.

*active* This toggle button lets you turn the current constraint object on and off. When a constraint object is not active, it is drawn in red, and SIMM does not enforce its constraints.

- visible* This button lets you toggle the visibility of the current constraint object in the 3D model window. Typically once you are done editing a constraint object you will make it invisible so that it does not obscure the display of other features of your model. A constraint object can be invisible but still active.
- constrain to quadrant* These radiobuttons and toggle buttons are used to specify that constraint points be constrained to a particular half of the constraint object. The x, y, and z radiobuttons let you specify the constraint axis, and the positive and negative toggle buttons let you specify the constraint direction. For example, if the x radiobutton is selected with the positive toggle button, then the points would be constrained to the half of the constraint primitive in which all x-coordinates are greater than zero (in the constraint object's local reference frame).

### 2.13.4 Transforming Constraint Objects

The collection of controls labeled *transform* can be used to move and rotate the current constraint object within its parent segment's frame of reference. The current constraint object's transformation can be modified interactively in the model window, or it can be modified precisely by entering values in the *translate* or *rotate* fields of the Constraint Editor window.

#### **INTERACTIVE CONSTRAINT OBJECT TRANSFORMATION**

- C** To interactively transform the current constraint object, hold down the **c** key then click and drag in the model window. The behavior of the left, middle, and right mouse buttons depends on the setting of the *trackball* toggle button at the bottom-left corner of the Constraint Editor window.

*trackball* If the *trackball* toggle button is checked, then the left mouse button will move the current constraint object left, right, up, and down in the plane of the computer screen. The middle mouse button will move the constraint object in and out, perpendicular to the screen. Finally, the right mouse button will rotate the constraint object as if it were attached to a virtual trackball.

If the *trackball* toggle button is not checked, then the left mouse button will rotate the current constraint object about the X-axis, the middle mouse button will rotate it about the Y-axis, and the right mouse button will rotate it about the Z-axis. Furthermore, if the *transform in local frame* radiobutton is selected then the constraint object will rotate about its local XYZ axes. If the *transform in parent frame* radiobutton is selected then the constraint object will rotate about its parent segment's XYZ axes.

#### **NUMERICAL CONSTRAINT OBJECT TRANSFORMATION**

*+, -, clear* To apply precise transformations to the current constraint object, first enter the distances and angles to transform by in the *translate* and *rotate* fields, then click the + button below the text fields one or more times to apply the transform. If the *transform in local frame* radiobutton is selected then the specified transform will be applied relative to the constraint object's local XYZ axes. Otherwise the transform will be applied relative to the constraint object's parent segment's XYZ axes. To apply the inverse transform, click the – button. The values you enter into these fields remain there even after they are applied. To clear all values from the *translate* and *rotate* fields, click the **clear** button.

**CURRENT TRANSFORM DISPLAY**

- transform in* The constraint object's current transform is displayed in the lower right corner of the Constraint Editor window as a sequence of XYZ rotations and an XYZ translation. The order of the rotation sequence that is displayed is X, followed by Y, followed by Z. This order cannot be changed. If the *transform in local frame* radiobutton is selected, then the rotations displayed are about the constraint object's local XYZ axes (a.k.a. body-fixed rotation). If the *transform in parent frame* radiobutton is selected, then the rotation sequence displayed is about the constraint object's parent segment's XYZ axes (a.k.a. space-fixed rotation).
- reset* Click the **reset** button below the current transform display to return the current constraint object to the origin of its parent segment's reference frame. The constraint object's transform is reset to the identity matrix.

**2.13.5 Points Command Menu**A rectangular button with a light gray background and a dark gray border. The text "constraint point" is in a bold, black, sans-serif font, followed by a right-pointing chevron symbol ">".

When you select this item a pop-up menu of existing constraint points is displayed. The first item in the pop-up menu, new constraint point, creates a new constraint point, adds it to the current model, and makes it the current constraint point for editing. The remaining items in the pop-up menu allow you select existing constraint points for editing.

A rectangular button with a light gray background and a dark gray border. The text "segment" is in a bold, black, sans-serif font, followed by a right-pointing chevron symbol ">".

Select this command to change the segment to which the current constraint point is attached (defined as the parent segment). When you select this item a pop-up menu of the model's segments is displayed. Choosing a segment from this menu will reassign the current constraint point to the selected segment. The constraint point's overall position will not change when it is assigned to a new segment.



However, when the model is put in motion the constraint object will follow the movement of its new parent segment. Constraint objects and points cannot both be assigned to the same segment, so the parent segment of the associated constraint object will be grayed out.

#### save points

This command saves a snapshot of the current state of all constraint points for the current constraint object. It can be used in combination with the **restore points** button to help you undo changes made while editing constraint objects.

#### restore points

This command restores the most recently saved snapshot of the current constraint object's points. It can be used in combination with the **save points** button to help you undo changes made while editing constraint points.

#### delete point

This command deletes the current constraint point from the model.

### 2.13.6 Constraint Point Attributes

The panel to the right of the constraint point command menu can be used to edit the current constraint point's attributes.

- |                   |  |
|-------------------|--|
| <i>point name</i> | This field contains the current constraint point's name. Click and type in this field to rename the constraint point.  |
| <i>segment</i>    | This field displays the name of the body segment to which the constraint point is attached. To change the current constraint point's segment, click the <b>segment</b> button to the right of the field. |
| <i>offset</i>     | This field let you specify the location of the constraint point as an offset from the origin of its body segment.  |

*weight* The *weight* of a point is a non-negative floating point value which controls how much emphasis is placed on it when enforcing the constraints. The higher the weight, the more SIMM tries to position that point onto the object surface, at the expense of the other points. If the weights of all of the points are the same (regardless of the actual weight value), SIMM will treat them all equally (*e.g.*, you cannot try to enforce a constraint better by increasing all the weights from 1.0 to 2.0).

*tolerance* The *tolerance* of a point is the distance (in model units) that it is allowed to be from its corresponding object, and still be an acceptable solution. It does not affect SIMM's calculation of the solution (as does the weight of a point). Rather, it is a threshold value which is checked after the best constraint solution has been calculated. If the point is not within *tolerance* of its constraint object, it is colored orange to indicate this condition.

## 2.14 Segment Editor

The Segment Editor allows you to edit the properties of the body segments in your SIMM model. Many of these properties are used only for Dynamics Pipeline simulations. If you will not be using your model for dynamic simulations, then the only relevant items in the Segment Editor are the display parameters such as drawing mode and material. If you will be performing dynamics, then this tool gives you access to the mass properties (mass, mass center, inertia), force matte definitions, and collision detection parameters (contact objects and spring-floors). Consult the Dynamics Pipeline manual for more information on how to use these parameters.

### 2.14.1 Selector Menu

A rectangular button with a grey background and a black border. The text "model" is on the left and a right-pointing chevron ">" is on the right.

The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.

A rectangular button with a grey background and a black border. The text "help" is centered.

Clicking on the help button opens a window containing helpful text about the Segment Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.14.2 Command Menu

A rectangular button with a grey background and a black border. The text "segment" is on the left and a right-pointing chevron ">" is on the right.

This command lets you select the body segment you want to edit. When a segment is selected, the tool window is updated to show that segment's parameters.

A rectangular button with a grey background and a black border. The text "save segment" is centered.

This command saves the current state of the current segment in the model. It can be used in combination with the **restore all** button to help you undo changes made while editing segments.

A rectangular button with a grey background and a black border. The text "restore segment" is centered.

This command restores the most recently saved copy of the model's current segment. It can be used in combination with the **save all** button to help you undo changes made while editing segments.

A rectangular button with a grey background and a black border. The text "drawmode" is centered.

This command lets you change the drawing mode for all of the bones in the current segment. The six available modes are: *gouraud shaded* (filled polygons, smooth shading), *flat shaded* (filled polygons, flat shading), *solid fill* (filled polygons, no shading), *outlined* (white polygons with highlighted edges), *wireframe* (hollow polygons), and *none* (no display). To change the drawing mode of individual bones in a body segment, right-click on the bone in the model window.



Figure 2-19. Segment Editor window

**material**

This command lets you change the material that is used to display the bones in the current segment. The pop-up menu shows the materials that you defined in the joint file, plus a set of default materials that are defined within SIMM. To change the material of individual bones in a body segment, right-click on the bone in the model window.

### 2.14.3 Segment Parameters

#### *Inertial Parameters*

This panel can be used to modify the mass and inertia properties of the current segment. These properties are used only in dynamic simulations. The individual fields are described below.

*mass* This field lets you specify the mass of the current segment. You can specify the mass in whatever units you want, but they should match the units you use to specify inertias, muscle forces, and external forces and torques. The mass units in the models that come with SIMM are kilograms.

*mass center* These fields let you specify the location of the mass center of the current segment. The coordinates are expressed in the body segment's reference frame.

*inertia* These fields let you specify the inertia matrix of the current segment. These values are expressed in the reference frame of the body segment, and are relative to the mass center of the segment. You can specify inertia in whatever units you want, but they should match the units you use to specify mass, muscle forces, and external forces and torques. The inertia units in the models that come with SIMM are kilograms \* meters \* meters.

### **Force Matte**

Force mattes are used only by dynamic simulations. They transform external forces expressed in the ground reference frame into the reference frame of the body segment to which the force matte is attached. Their primary use is to apply forces measured by forceplates to the feet during a simulation. To learn more about how force mattes work, consult the Dynamics Pipeline manual.

A body segment in SIMM can contain only one force matte. To create a force matte for the current segment, click on the **new force matte** button. If the segment already has one defined, the button will be grayed out and the force matte's parameters will be displayed. The *name* field specifies the name of the force matte. The *filename* field specifies the name of the SIMM bone file which contains the polygon description[s] that comprise the force matte. Type a name into this field or click on the browse button to browse for a file. The *visible* checkbox controls whether or not the force matte is displayed in the model window.

### **Contact Object**

Contact objects are used only by dynamic simulations, and they are used to model rigid contact between polygonal surfaces. To learn more about how they work, consult the Dynamics Pipeline manual.

The interface in the Segment Editor does not let you define all of the properties necessary to set up rigid-body contact detection for dynamic simulation, but it does let you define and view the polygonal contact objects. Once you have defined and positioned these appropriately in your model, you will need to edit the joint file and enter

the contact pairs. Contact pairs specify which objects can contact which other objects.

To create a contact object for the current segment, click on the **contact object** button and choose *new* from the pop-up menu. The *name* field specifies the name of the contact object (which is used in the contact pair definitions). The *filename* field specifies the name of the SIMM bone file which contains the polyhedron that comprises the object. Type a name into this field or click on the browse button to browse for a file. The *visible* checkbox controls whether or not the contact object is displayed in the model window.

### **Spring Contact**

Spring contacts are used only by dynamic simulations, and constitute a form of penalty-based contact detection. A *spring floor* is attached to one body segment, and one or more *spring points* are attached to other body segments. When a point passes below the floor, a non-linear spring force is applied to the point's body segment to push it back above the floor. This technique is commonly used to model contact between the feet and the floor during a forward dynamic simulation.

Each body segment contain only one spring floor. To create a spring floor for the current segment, click on the **new spring floor** button. If the segment already has one defined, the button will be grayed out and the spring floor's parameters will be displayed. The *name* field specifies the name of the spring floor. The *filename* field specifies the name of the SIMM bone file which contains the polygon that comprises the floor (only the first polygon in the bone file is used). Type a name into this field or click on the browse button to browse for a file. The *visible*

checkbox controls whether or not the spring floor is displayed in the model window.

Once a spring floor is defined, you can define one or more spring points on other body segments. Note that although these points are defined on segments other than the currently selected segment, they are only accessible in the Segment Editor when the current segment is the one to which their corresponding *floor* is attached. The **spring point** button lets you select from the existing points or create a new one. The **segment** button lets you change the segment to which a point is attached. The *X*, *Y*, and *Z* fields specify the coordinates of the point in its associated segment's reference frame. The parameters *friction*, *a*, *b*, *c*, *d*, *e*, and *f* control the calculation of the spring force. Consult the Dynamics Pipeline manual for details on how these parameters are used.

#### **Display Parameters**

- |                          |  |
|--------------------------|--|
| <i>show mass center</i>  | This toggle button lets you turn on and off the display of the current segment's mass center location. If the mass center is not fully specified (the <i>X</i> , <i>Y</i> , and <i>Z</i> components must all be specified), the button is deactivated. |
| <i>show segment axes</i> | This button lets you toggle the visibility of the current segment's reference frame in the model window.   |
| <i>show normals</i>      | This button lets you toggle the visibility of the normal vectors in the model window for all the bones in the current segment.   |
| <i>show inertia</i>      | This toggle button lets you turn on and off the display of the inertia vectors of the current segment.   |



### **Global Parameters**

These display parameters are applied to all the segments in the model.

*show all mass centers*

This toggle button lets you turn on and off the display of the mass center locations of all the segments in the model.

*show inertia*

This toggle button lets you turn on and off the display of the inertia vectors of all the segments in the model.

## **2.15 Bone Editor**

The Bone Editor allows you to transform bones (translate rotate, and scale) and move individual vertices of a bone, as well as perform simple boolean operations between bones (addition, subtraction, intersection). This tool is especially useful when you are creating a new model and need to move or scale bones within their reference frames. It provides a graphical interface to all the functionality of the *norm* program.

### **2.15.1 Selector Menu**



The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Bone Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

## 2.15.2 Command Menu

Most of the operations in the command menu work on the bone specified by the **bone 1** button. This bone is called the current bone (which is not the same as the current segment). The boolean operations work on two bones specified by the **bone 1** and **bone 2** buttons.

### do boolean

This command executes the specified boolean operation between *bone 1* and *bone 2*. The output bone is given the name that is in the *new bone name* field, and is stored in the current segment, which can be changed using the **change segment** button.

### norm bone

This command runs *norm* on *bone 1*. Any transformations you have done to the bone are integrated into the bone file (the bone vertices are transformed), and the normals are recomputed. The translation and rotation fields are then reset to 0.0, and the scale fields are reset to 1.0.

### split polygon

This command splits the selected polygon (displayed in green) by adding a vertex at the center of it and forming triangles using that vertex and each edge of the original polygon. This command works only if there is one selected polygon.

### delete polygons

This command deletes the selected polygons. There is no way to undo this operation, except by reloading the bone file. However, in some cases you may be able to fill in holes created by deleting polygons by running *norm* on the bone with the *fill holes* option.

### scroll edges

This command lets you change the selected edge (of the selected polygon) by scrolling through them. It works only if there is one selected polygon.

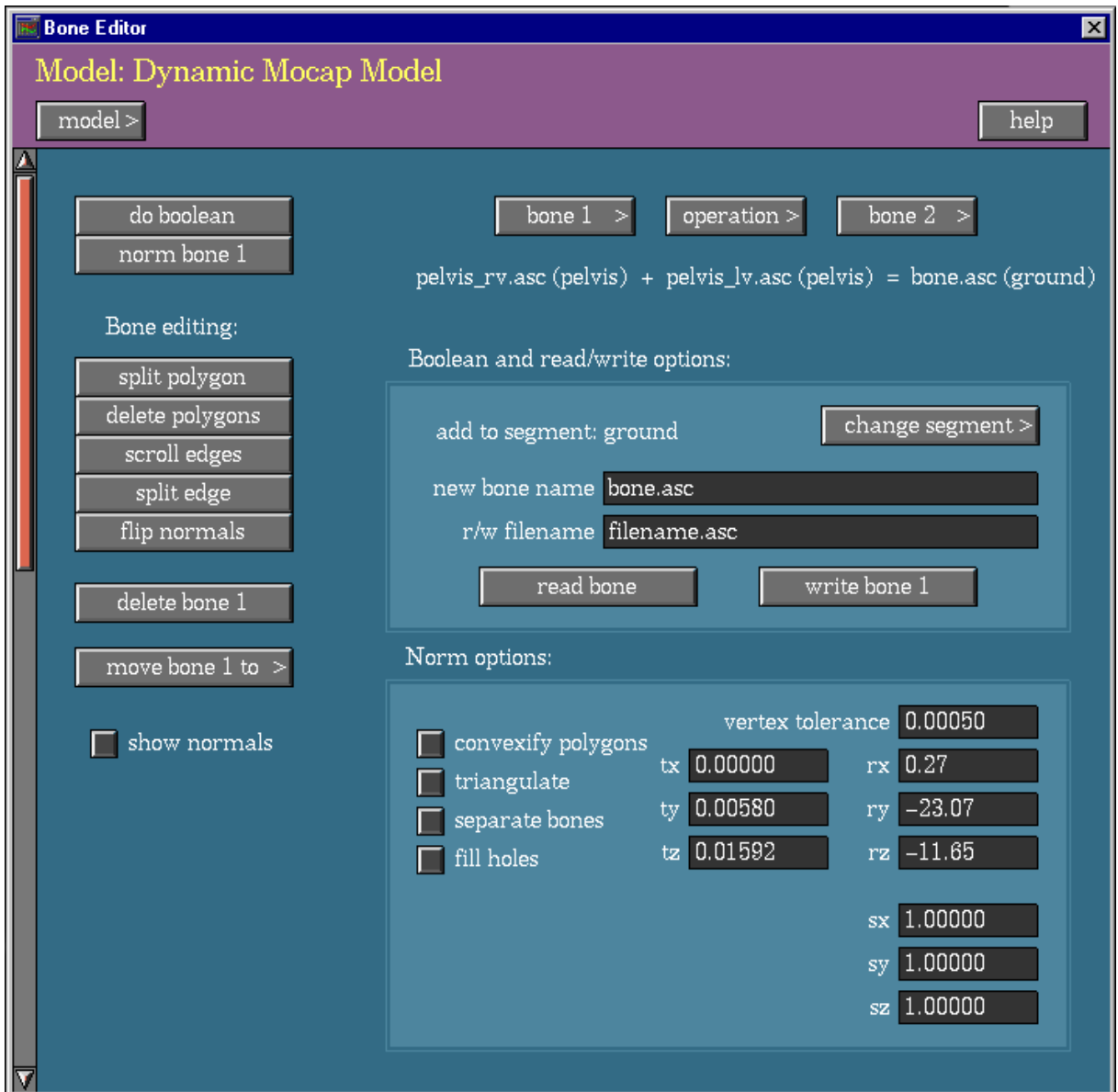


Figure 2-20. Bone Editor window

**split edge**

This command splits the selected edge in half, causing each of the two polygons which use it to be split into two pieces.

**flip normals**

This command reverses the directions of the normal vectors for the selected polygons. It then recomputes the vertex normals for the bone.

**delete bone**

This command deletes *bone 1*. This operation cannot be undone.

**move bone1 to >**

This command moves *bone 1* to the chosen body segment. It leaves the bone in its current position (relative to the rest of the model), but transforms the vertex coordinates so that they are expressed in the new segment's reference frame. It then runs *norm* on the bone using the current set of parameter values in the *norm options* region.

**bone 1 >**

This button lets you choose *bone 1*. *Bone 1* is the current bone, which can be transformed using the norm functions, edited with the polygon and vertex editing functions, and written to a file. *Bone 1* is also the first bone in the boolean operation, as shown in the boolean equation below this button.

**operation >**

This button lets you choose the boolean operation to perform on *bone 1* and *bone 2*. The available operations are: *add*, *intersect*, and *subtract*. The result of the operation will be one or more polygonal surfaces, depending on the input. If the *separate bones* checkbox is turned off, then all of these surfaces will be combined into a single output bone. If the checkbox is off, each surface will be output to a separate bone. The boolean operation is performed when you press the **do boolean** button.

**bone 2** >

This button lets you choose *bone 2*, which is used only in boolean operations, as shown in the boolean equation below this button.

### 2.15.3 Boolean and Read/Write Options

Performing a boolean operation is similar to reading a bone file into SIMM in the sense that they both create a new bone. Thus the options to specify the name of the new bone, and to which body segment it is added, are located in the *boolean and read/write options* region.

**change segment** >

This command changes the body segment to which the new bone will be added, when a boolean operation is performed or when a bone file is read. It does not move the currently selected *bone 1* to that segment.

*new bone name*

This field specifies the name of the bone that will be created, either from the result of a boolean operation or from reading in a bone file.

*r/w filename*

This field specifies the name of the bone file to use when reading a bone file (with the **read bone** command) or when writing a bone file (with the **write bone 1** command). The folder that this file is read from or written to is the folder containing the joint file for the current model.

**read bone**

This command reads the bone file specified in the *r/w filename* field and adds the bone to the body segment specified in the *add to segment* field.

**write bone 1**

This command writes *bone 1* to the bone file specified in the *r/w filename*.

### 2.15.4 Norm Options

*Norm* is used to process a bone when you select the **norm bone 1** button, and it is also used to postprocess the bone that is output from a boolean operation. The options to *norm* are used for both types of calls, unless otherwise specified.

When *norm* performs the specified transformations on a bone, it does them in the following order. First, it applies the rotations in a body-fixed manner (X, then Y, then Z). Then it applies the X, Y, Z scale factors and translations relative to the X, Y, and Z axes of the body segment's reference frame (not the rotated frame). After the transformations have been applied, the translation and rotation fields are reset to 0.0, and the scale fields are reset to 1.0. If you are unsure what the result will be of a series of different transformations, it is best to apply them one at a time, processing the bone with *norm* after each one.

When rotating, translating, or scaling a bone using either the keyboard commands (described in the next section) or typing values into the fields described below, remember to process the bone with *norm* (using the **norm bone 1** button) in order to apply the transformations to the bone vertices. Until you do this, the transformations are stored in a temporary buffer and the bone itself is unchanged (though you can see the effects of the transformation in the model window). When editing polygons and vertices of a bone, you should also run *norm* on it occasionally, to recompute the surface normals and update its shading. Lastly, you should frequently save the bone to a file as you make modifications, because there is no way to undo changes or restore the bone to a previously saved copy. If you make changes that you want to undo, you will need to delete the bone and read it back in from a previously saved file.

<i>convexify polygons</i>	When selected, <i>norm</i> will attempt to make every polygon in the bone convex by splitting concave polygons into convex pieces.
<i>triangulate</i>	When selected, <i>norm</i> will triangulate every polygon in the bone by splitting polygons (without adding vertices).
<i>separate bones</i>	When selected, <i>norm</i> will output a separate bone for each distinct polyhedral surface in the original object. This option is used only when postprocessing a bone created by a boolean operation.
<i>fill holes</i>	When selected, <i>norm</i> will attempt to fill in holes in the bone surface by adding triangles. If the bone contains large holes, <i>norm</i> may be unable to fill them.
<i>vertex tolerance</i>	The number in this field is the tolerance that <i>norm</i> uses to remove duplicate vertices. All vertices within <i>tolerance</i> of each other are merged into one vertex. A tolerance value of 0.0 will turn off vertex tolerance checking.
<i>tx, ty, tz</i>	These fields specify the X, Y, and Z translations of the bone within its segment's reference frame.
<i>rx, ry, rz</i>	These fields specify the X, Y, and Z rotations of the bone within its segment's reference frame. These rotations are applied in a body-fixed manner, in the order X, then Y, then Z (i.e., an Euler XYZ rotation). It is recommended that you apply one rotation at a time, so that the axis of each rotation remains parallel to one of the axes of the body segment's reference frame.
<i>sx, sy, sz</i>	These fields specify the X, Y, and Z scale factors of the bone, relative to its original size.

### 2.15.5 Keyboard Commands

There are several keyboard commands that you can use to transform *bone 1* or to move individual vertices on *bone 1*. To use them, click on the model window to make it the active window, and then press the appropriate key[s] described below.

*B key, left mouse button*

Pressing these two keys and moving the cursor will translate the current bone within the reference frame of its body segment. The translation is stored in a matrix; the actual bone vertex coordinates are transformed only when you run *norm* on the translated bone.

*B key, right mouse button*

Pressing these two keys and moving the cursor will rotate the current bone within the reference frame of its body segment. The rotation is stored in a matrix; the actual bone vertex coordinates are transformed only when you run *norm* on the rotated bone.

*C key*

Pressing this key will select the vertex on the current bone that is closest to the cursor.

*V key, left mouse button*

Pressing these two keys and moving the cursor will translate the selected vertex of the current bone. The vertex can be moved within the plane of the screen, and will track the cursor's movement.

*V key, X, Y, and/or Z keys*

Pressing these keys will translate the selected vertex along the negative X, Y, and/or Z axes, as appropriate. Press the `Shift` key as well to translate along the positive axes.



## 2.16 Dynamics Tool

The Dynamics Tool is a graphical interface for running dynamic simulations created by the Dynamics Pipeline module. These simulations are compiled as DLLs (dynamic link libraries), and can be loaded into SIMM for execution. The Dynamics Tool allows you to set various parameters of the simulation, such as the input motion (for inverse dynamics) and the desired output variables (*e.g.*, joint torques).

The Dynamics Pipeline performs forward and inverse dynamics analyses using the same numerical method. In both cases, the equations of motion are integrated forward in time using a variable-step integrator. Forces and torques are applied to the body segments (from gravity, muscles, ligaments, ground reactions, *etc.*), and the motion of the joints is calculated. If the motion of some or all of the joints is prescribed (*e.g.*, to follow motion capture data), then the dynamics code calculates and applies the necessary constraint torques in order to ensure that the joints move as prescribed. In a typical forward dynamics simulation, muscle and external forces are applied, and the motion of the joints is not prescribed. In a typical inverse simulation, ground-reaction forces are applied (but not muscle forces), and the motion of all of the joints is prescribed. In this case, the simulation will calculate the torques that are required at each joint to create the prescribed motion. However, because the same numerical method is used to perform both simulations, it is possible to mix forward and inverse dynamics. For example, you can prescribe the motion of all of the joints, but also specify the excitations of some of the muscles (*e.g.*, from EMG recordings). In this case, the simulation will apply the muscle forces to the body segments, and then calculate and apply any additional torques that are required to pre-

scribe the motion. This scenario is useful during an inverse dynamics analysis if you want to estimate the forces in the muscles whose EMG data you are collecting.

Dynamics Pipeline users can create their own DLLs of whatever musculoskeletal models they want, and thus can perform dynamic simulations on them in SIMM. FIT Module users cannot create their own DLLs, but can choose from several pre-built DLLs that come with the software. These pre-built DLLs include models of a full body (with simplified shoulder motion and lumped-mass hands), a lower-extremity model with lumped-mass upper body, and a right-arm model (with lumped-mass hand).

When a DLL is created, the following model properties are compiled into it, and thus you cannot change any of them in your SIMM model without creating a new DLL:

- the number of body segments
- the number of joints
- the number of degrees of freedom in any joint
- the order of rotations in a joint.

The following model properties are passed from SIMM to the DLL when the **initialize** button is pressed, and thus you can change them in SIMM without needing to reload or recreate a DLL:

- the number of muscles
- any property of any muscle
- the mass, mass center, or inertia of any body segment
- the clamped and locked states, and range of motion, of any gencoord
- the restraint torques at any gencoord
- the orientations of any joint's rotation axes

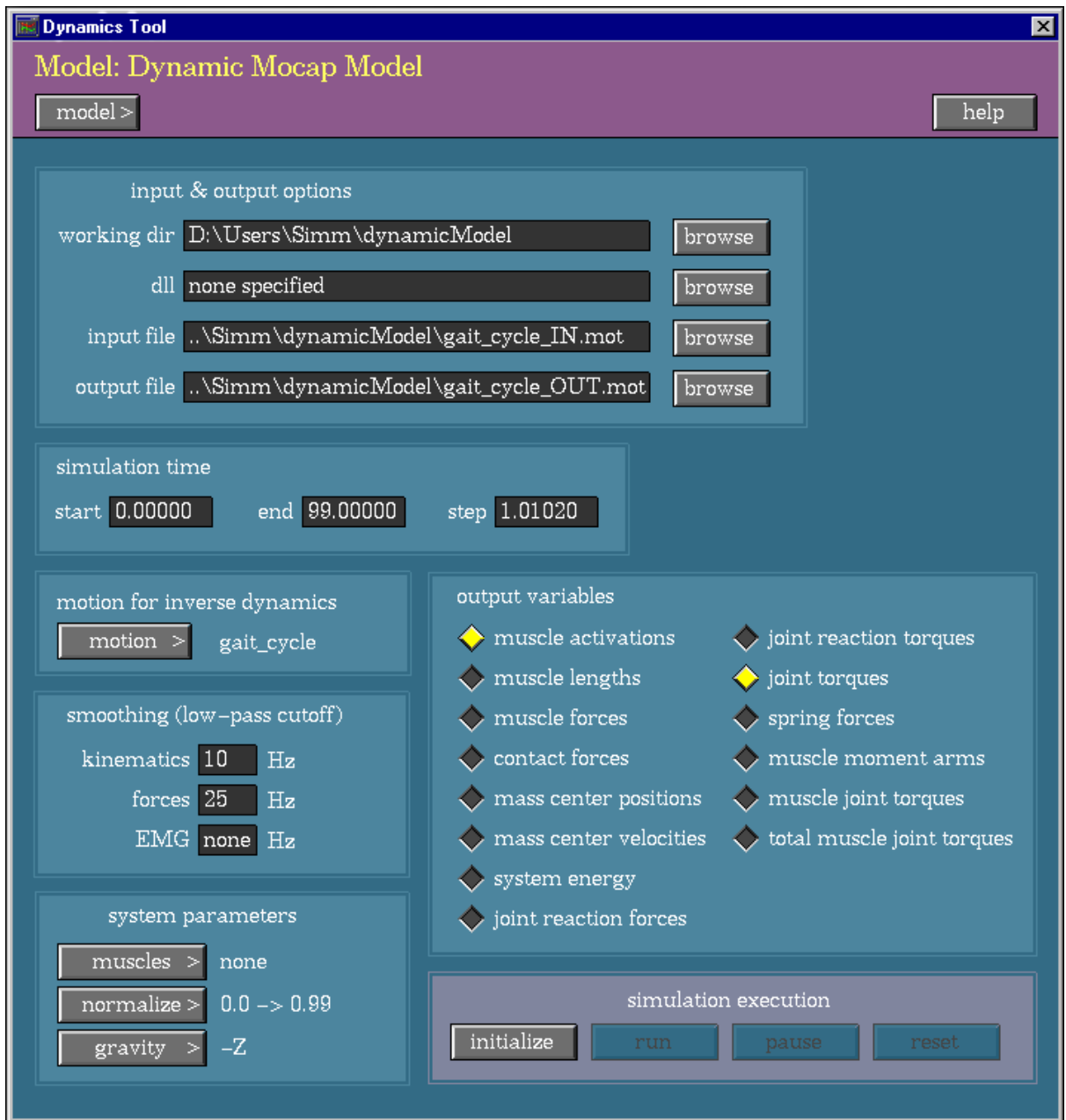


Figure 2-21. Dynamics Tool window

### 2.16.1 Selector Menu



The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Dynamics Tool. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.16.2 Input and Output Options

*working dir*

This specifies the directory to be used for the dynamic simulation. This directory is used as the default location for the input and output files. For inverse dynamics, you should set the working directory before choosing the motion so that the paths for the input and output files will be set accordingly (see Section 2.16.4, Motion For Inverse Dynamics, for more details).

*DLL*

This field specifies the DLL (dynamic link library) to be used for the dynamic simulation. This DLL can either be one that you create using the Dynamics Pipeline, or a pre-built one that comes with the FIT Module. Make sure to select the one that matches the model you have loaded into SIMM.

*input file*

This field specifies the input file to be used for the dynamic simulation. For forward dynamics, this file is optional. If used, it can contain external forces and muscle excitation patterns that you want to apply during the simulation. For most inverse dynamics simulations, you do not need to specify an input file. When you select a motion (see Section 2.16.4, Motion for Inverse Dynamics), SIMM will automatically fill in the *input file* field with the

name of a new file. SIMM will create this file with all the appropriate data columns and smoothing parameters when you initialize the simulation.

*output file* This field specifies the output file created by the dynamic simulation. As the simulation proceeds, it will write each time frame of output data to this file. When you select a motion for inverse dynamics (see Section 2.16.4, Motion for Inverse Dynamics), SIMM will automatically fill in the *output file* field with the name of a file based on the name of the input motion.

### 2.16.3 Simulation Time

The fields in this region control the time of the dynamic simulation, and how often the results are written to the output motion. For most inverse dynamics applications you do not need to set these parameters; they will be set automatically when you choose a motion (see Section 2.16.4, Motion for Inverse Dynamics).

*start* This specifies the starting time for the dynamic simulation. For forward dynamics, set this field to the desired starting time (usually 0.0). When you choose a motion for inverse dynamics, the starting time will automatically be filled in with the time of the first frame of data.

*end* This specifies the ending time for the dynamic simulation. For forward dynamics, set this field to the desired ending time. When you choose a motion for inverse dynamics, the ending time will automatically be filled in with the time of the last frame of data.

*step* This specifies the step size, or reporting interval, for the dynamic simulation. As the simulation proceeds, it will write data to the output file, and return the same data to

SIMM, every *step* seconds. When you choose a motion for inverse dynamics, the step size will automatically be filled in with the step size of the data in the motion.

### 2.16.4 Motion for Inverse Dynamics



This button lets you choose a motion to use for the dynamic simulation (primarily for inverse dynamics). When you press this button, a pop-up menu is displayed with an option called *no motion*, followed by the list of motions currently loaded for this model. For forward dynamics, you will usually want to choose the *no motion* option. However, if you would like to prescribe the motion of some of the gencoords during a forward simulation, then choose the appropriate "partial" motion. For inverse dynamics, choose a motion with data for all of the gencoords. When you choose a motion, SIMM will automatically fill in the *input file* field with the name of a motion file that SIMM will generate. This file will contain the same data as the chosen motion, but will also include the smoothing parameters you have chosen (see Section 2.16.5, Smoothing (Low-Pass Cutoff)). SIMM will generate this file when you press the **initialize** button. When you choose a motion, SIMM will also fill in the *output file* field with the name of a file in the working directory that will hold the results of the simulation. Lastly, when you choose a motion SIMM will fill in the three *simulation time* fields with time values taken from the motion data.

### 2.16.5 Smoothing (Low-Pass Cutoff)

The following smoothing parameters are used only for smoothing data that is contained in the motion chosen with the **motion** button. If you do not choose a motion, but instead specify an input file that you created for the

simulation, you can perform smoothing of that data only by entering the appropriate column labels in the file itself.

*kinematics* This specifies the low-pass cutoff frequency to use when smoothing the gencoord data. Enter a value of -1 for no smoothing.

*forces* This specifies the low-pass cutoff frequency to use when smoothing the external force data (*e.g.*, ground-reaction forces). Enter a value of -1 for no smoothing.

*EMG* This specifies the low-pass cutoff frequency to use when smoothing the EMG data. Enter a value of -1 for no smoothing.

### 2.16.6 System Parameters



This button lets you choose which muscles you want to include in the dynamic simulation. The following options are available:

*none* no muscles

*all* all the muscles in the SIMM model

*visible* all the muscles whose display is currently turned on in the Model Viewer

*with EMG data* all muscles that have EMG data in the currently selected motion

*with excitations* all muscles that have excitation patterns defined for them



This button lets you choose the normalization option for the output motion created by the dynamic simulation. Normalization of the output can be useful if you want to compare several dynamic analyses to each other. By default, normalization is off, and the output motion will

contain the appropriate number of data frames as specified by the *simulation time* parameters. The option *0.0 -> 0.99* will result in the output motion containing 100 frames of data. In addition, a "reference" number will be added to each frame of data, ranging from 0.0 to 0.99. The option *0.0 -> 99.0* will also result in 100 frames, but the reference number will range from 0.0 to 99.0. Normalization does not affect the accuracy or step size of the numerical integration during the simulation. It only affects the time interval between the frames of data stored in the output motion.



This button lets you specify the direction of the gravity vector. In general you should specify the gravity vector in your joint file, in which case that value is used to initialize the parameter in this tool. However, if you want to change it from the default, you can do so with this button.

### 2.16.7 Simulation Execution



This button initializes the dynamic simulation. A copy of the current model is sent to the simulation, which checks to make sure it matches the one built into the DLL. If the number and connectivity of the body segments, as well as the joint types, match between the two models, the simulation can proceed. The simulation model is updated with the mass properties and joint kinematics from SIMM, and an output motion is created to hold the data calculated by the simulation. This motion is added to the SIMM model, and a slider for it is added to the Model Viewer.



This button starts the dynamic simulation. As it proceeds, calculating each time step of data, SIMM stores the frames of data in the motion that was created during the *initialize* step (see above). SIMM also updates the display of the model and the plots with the new data. When the



simulation has completed, you can review the results using the motion's slider in the Model Viewer, and by plotting motion curves with the Plot Maker.

**pause / resume**

This button pauses a running simulation, or resumes a paused simulation. When paused, your only options for controlling the simulation are to resume it, or reset it (which ends the simulation, saving the partial results in the output motion).

**reset**

This button ends a simulation, and resets the dynamics to the uninitialized state. If the simulation has been initialized but is not yet running when the **reset** button is pressed, the motion created to hold the results is deleted from the model. If the simulation is running when **reset** is pressed, the results calculated so far are stored in the motion.

### 2.16.8 Output Variables

The checkboxes in this region specify which variables are calculated during the simulation, and saved in the output motion. When the **initialize** button is pressed, the output motion is created with the appropriate columns to hold all of the selected variables. These columns are initially filled in with zeros, and are overwritten with the simulation results as each time frame is calculated. Below is a list of the available output variables.

*muscle activations*

the dynamic activation levels of the muscles. This will not be the same as the excitation levels if the muscle model used in the dynamic simulation models non-linear activation dynamics (most models do).

*muscle lengths*

the musculotendon lengths of all of the muscles in the simulation.

<i>muscle forces</i>	the dynamic forces in all of the muscles in the simulation.
<i>contact forces</i>	the forces applied by the rigid-body contact detection algorithm. This includes the impulses applied during penetrating contact, and the forces applied during resting contact.
<i>mass center positions</i>	the X, Y, Z coordinates of the location of each segment's mass center. This is usually needed only when you create a new SIMM model and want to double check that it has been assembled correctly in the dynamics code.
<i>mass center velocities</i>	the X, Y, Z components of the velocity of each segment's mass center. This is usually needed only when you create a new SIMM model and want to double check that it has been assembled correctly in the dynamics code.
<i>system energy</i>	the total energy (potential plus kinetic) of the model during the simulation. If there are no muscle, external, or friction forces applied, system energy will be conserved during the simulation.
<i>joint reaction forces</i>	the reaction forces at each joint. These forces are those applied by the proximal segment to the distal segment. They are applied at the origin of the distal segment, and are expressed in the distal segment's reference frame.
<i>joint reaction torques</i>	the joint reaction torques at each joint. These are the torques applied by the proximal segment to the distal segment, expressed in the distal segment's reference frame. They include user-applied torques such as joint restraint torques.
<i>joint torques</i>	the torques applied to the gencoords by the simulation to generate the prescribed motion. These torques are calculated after all system forces ( <i>e.g.</i> , gravity) and user-applied forces ( <i>e.g.</i> , muscle, ground-reaction forces) are applied. In a typical inverse dynamics analysis, the genco-

ord motion is fully prescribed and no muscles are included. The joint torques are thus equal to the net torques required (by the subject's muscles) to produce the prescribed motion. If you include muscles in your inverse dynamics analysis, your joint torques will be different. If you include exactly the right set of muscles and excitations (as was used by the subject in the recorded motion), no additional torques will be needed to produce the prescribed motion, and the joint torques will be zero. In a simulation that has no prescribed motion (*e.g.*, forward dynamics), the joint torques will be zero.

*spring forces* the forces applied by the spring/floor objects. Each force is expressed in the reference frame of the corresponding spring point's body segment.

*muscle moment arms* the moment arms of all of the muscles in the simulation.

*muscle joint torques* the moment that each muscle applies to each gencoord.

*total muscle torques* the total (net) moment applied to each gencoord by all of the muscles.

## 2.17 Motion Editor

The Motion Editor allows you to modify the motions linked to each model. The currently selected motion can be renamed, deleted, cropped, written to a file, or have events added to it. If plots have been created using these motions before they were edited, these existing plots will not change to reflect the new motion characteristics; the changes will only appear in newly created plots. This tool can be useful if you want to compare several motion trials to each other because it allows you to normalize them according to certain events, such as heel strike and toe off.

### 2.17.1 Selector Menu



The model selector lets you select the current model. If you never have more than one model at a time loaded into SIMM, then you do not need to use the model selector.



Clicking on the help button opens a window containing helpful text about the Motion Editor. When you are done reading it, close the window using its close box. You will be able to re-open it at a later time by selecting **help** again.

### 2.17.2 Command Menu



When you select this item a pop-up menu is displayed of motions linked to the model. Selecting a motion makes it the current motion for editing. Its properties can then be viewed and edited using the forms and sliders in the tool window.



This command allows you to save the current motion to a file. When you select it, a file browser will let you choose the output file to create or overwrite.



This command deletes the current motion. The motion is unlinked from the model and it will no longer be able to be viewed or plotted. Any existing plots using the deleted motion will remain unchanged.

### 2.17.3 Cropping a Motion

You can change the range of an existing motion using this feature. *Start* and *end* frames can be entered into the form boxes, or you can use the slider to move the *start* and *end* frames up and down until you have chosen the range you would like (the model will move in the motion window

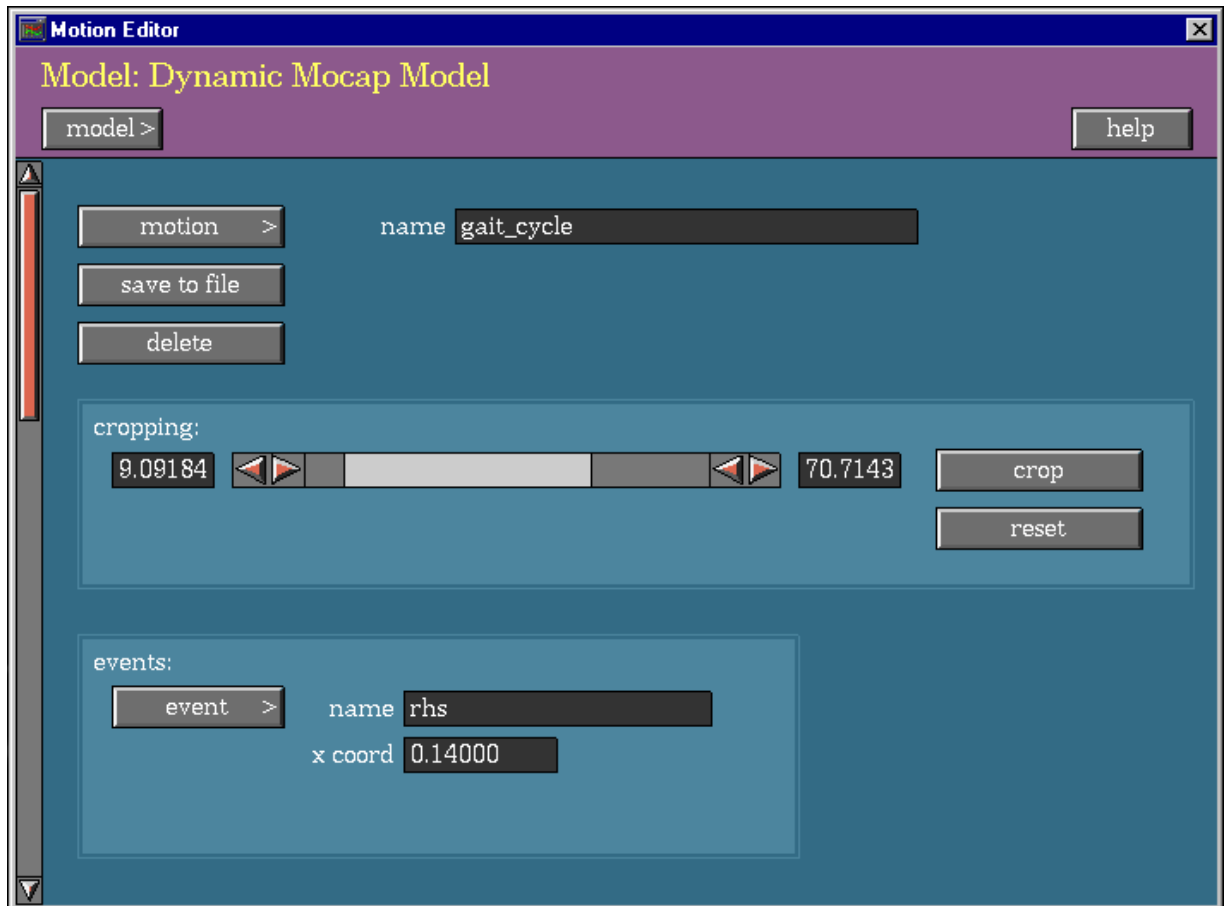


Figure 2-22. Motion Editor window

accordingly). Use the **crop** button to crop the motion when you are satisfied with the new range.



This command crops the motion between the two selected frames. This operation cannot be undone. Any parameter fields in other tools that are linked to the cropped motion will be updated accordingly (e.g., the *x min* and *x max* fields in the Plot Maker). Any existing plots of the motion will remain unchanged.

**reset**

Before you have cropped the motion you can use this command to reset the crop limits to their original values.

### 2.17.4 Motion Events

You can create and edit events in a motion using this feature. You can edit the event's name and the time at which it occurs in the motion. New events created using this feature will not appear in any existing plots, but will appear in any new plots created from the motion.

**events**



When you select this item a pop-up menu of existing events is displayed. The first item in the pop-up menu, *new event*, creates a new event named *event\_X*, adds it to the current motion at time 0.0, and makes it the current event for editing.

---

# Input Files

---

## 3.1 Introduction

A musculoskeletal model is specified with three types of input files. The bone files (Section 3.2) contain lists of polygons that define the bone surfaces. The joint file (Section 3.3) specifies the kinematics of the joints and the characteristics of the body segments. The muscle file (Section 3.4) contains lists of coordinates that describe the muscle lines of action, and the parameters needed to compute isometric muscle forces. SIMM scans these input files and creates a data structure that represents the musculoskeletal model. When SIMM scans the joint and muscle files, it ignores all text between the delimiters `/*` and `*/`, so you can include comments in the files to help you document your model. Bone files, which cannot contain comments, can be preprocessed with *norm* (see Chapter 4) before being loaded into SIMM.

This chapter describes how to create a musculoskeletal model using input files. Examples from the lower-extremity model are shown to demonstrate the format of each file. The input files that define this model are provided with SIMM. It is often helpful to edit these files and then reload them into SIMM when you are learning to create your own models. The files can also serve as templates when you make a new model.

This chapter also describes motion files (Section 3.5) and plot files (Section 3.6). Motion files are used to specify

the values of the generalized coordinates during a particular motion. They are used to animate the model and to plot data versus a motion parameter. For example, if you define a motion file that specifies the generalized coordinate values during pedaling, you can display the model as it moves through the pedaling cycle and plot independent variables vs. the crank angle. Plot data files are helpful for comparing data from outside sources to data computed with the model.

Note: SIMM does not impose specific units on your musculoskeletal model, nor do you need to define units explicitly. It assumes that the numbers you enter into data files for bone vertices, muscle parameters, and joint kinematics are internally consistent, and it uses them “as is” to calculate dependent variables such as muscle lengths and joint torques. The sample lower-limb model uses units of meters for length and newtons for force.

## 3.2 Bone Files

A bone is a mesh of planar polygons. It is usually a closed surface, but it does not need to be closed for SIMM to be able to load and display it (*i.e.*, it can have holes in it). Bone files list the 3D coordinates of the polygon vertices (in a vertex list) and define how the vertices are connected to form polygons (in a polygon list). The reference frame that is associated with each body segment serves as the local coordinate system for the polygon vertices. Each bone file can contain descriptions of any number of separate bone surfaces (*e.g.*, tibia and fibula), but all the bones in one file will move as a single object. An example bone file is shown in Figure 3-1.



```

NORM_ASCII
260 290
-0.20459 0.45827 -0.35921 0.27346 0.02846 0.68189
-0.062448 -0.071901 0.057002 0.195918 0.980604 0.005772
-0.062417 -0.062953 0.055540 -0.010377 0.946434 -0.322728
-0.045919 -0.070017 0.069408 -0.203832 0.677009 -0.707184
-0.057682 -0.053884 0.064535 0.232364 0.048436 -0.971422
.
.
.

3 0 1 2
4 2 3 6 5
3 2 1 3
4 4 7 12 10
.
.
.

```

Figure 3-1. Example bone file in ASCII format

Note: Bone files are not necessary for developing and analyzing musculoskeletal models in SIMM. Even without bones, you can define segments, joints, and muscles, and display the muscles in the model window. Bones are merely visual aids for moving the joints and positioning muscle attachment points.

Bone files can be specified in any of three formats. The first format, shown in Figure 3-1, is the recommended format for new musculoskeletal models. It is an ASCII format that contains the following information.

The first line in the bone file must be the string `NORM_ASCII` to indicate the file type. The second line contains the number of vertices (*num\_vertices*) followed by the number of polygons (*num\_polygons*). The third

line contains the bounding box of the bone. The six numbers are, respectively, X min, X max, Y min, Y max, Z min, and Z max. Then comes the vertex list, whose length must be *num\_vertices*. Each line in this list contains the X, Y, and Z coordinates of a vertex and the X, Y, Z coordinates of the vertex normal. The utility program *norm* can compute vertex normals for you (see Chapter 4, Norm, for more details). Lastly, there is the polygon list, whose length must be *num\_polygons*. The first number in each line is the number of vertices (*nv*) in the polygon, and the other numbers are the indices of the *nv* vertices. The indices start at 0, and correspond to the order in which the vertices are listed in the vertex list. In the example shown, the first polygon is a triangle, formed by the first, second, and third vertices in the vertex list. The second polygon is four-sided, using the third, fourth, seventh, and sixth vertices. When listing the indices for a polygon, you should order them so that they “walk” around the perimeter of the polygon. You can specify them in either clockwise or counter-clockwise fashion (and can switch directions from polygon to polygon), but you cannot list them randomly so that they criss-cross through the polygon. When you process the bone file with *norm*, the polygons will be re-ordered so that all of the vertices are specified in a counterclockwise fashion.

The second bone file format that is supported is an older ASCII format that is used on the SGI versions of SIMM and *norm*. This format is shown in Figure 3-2. It includes less information about the bone, so it is easier to create if you are making a bone file from scratch or converting a file from another software package. It is recommended that when creating new bone files you create them in the old ASCII format and then process them with *norm* to output them in the newer ASCII format. The old ASCII format is described below.

```

260   290

-0.062448  -0.071901   0.057002
-0.062417  -0.062953   0.055540
-0.045919  -0.070017   0.069408
-0.057682  -0.053884   0.064535
      •
      •
      •

 3     1     2     3
 4     3     4     7     8
 3     2     3     4
 4     4     5     13    12
      •
      •
      •

```

Figure 3-2. Example bone file in “older” ASCII format

The first line in the file must contain the number of vertices (*num\_vertices*) followed by the number of polygons (*num\_polygons*). Then comes the vertex list, whose length must be *num\_vertices*. Each line in this list contains the X, Y, and Z coordinates of a vertex. Lastly, there is the polygon list, whose length must be *num\_polygons*. The first number in each line is the number of vertices (*nv*) in the polygon, and the other numbers are the indices of the *nv* vertices. The indices start at 1, and correspond to the order in which the vertices are listed in the vertex list. In the example shown, the first polygon is a triangle, formed by the first, second, and third vertices in the vertex list. The second polygon is four-sided, using the third, fourth, seventh, and eighth vertices.

The third format supported by SIMM and *norm* is a binary format. Previous versions of SIMM required bones to be in this binary format, so *norm* would output binary bone files by default. If you have some of these binary files, you can load them into the current version of SIMM or you can use *norm* to convert them to ASCII (which is recommended if you plan to manipulate or transform them in any way).

*Norm* is located in the directory where the SIMM executable resides. In addition to converting between different file formats, *norm* processes bone files by calculating vertex normals, checking the vertex and polygon lists for inconsistencies, and, optionally, translating and rotating bones and filling in holes in their surfaces (see Chapter 4, Norm, for details).

## 3.3 Joint Files

Joint files define the body segments, joints, generalized coordinates, and kinematic functions used in musculoskeletal models. They can also include optional components such as world objects, muscle wrap objects, deform objects, constraint objects, and materials. Each model you load into SIMM must have one joint file defining all of the segments, joints, generalized coordinates, and kinematic functions that comprise the model. You may list these components in the file in any order, but the order in which you enter segment names can affect the display of the model by influencing the choice of the *fixed* segment (see Section 3.3.2, Fixed Segment, for more information).

The joint file also contains a number of parameters that specify various properties of the model, such as its name, some of the colors used in the display, and the units of

length and force in which the model is specified. All of these parameters are optional, and have reasonable default values if they are not specified in the joint file. These parameters generally go at the top of the file, although they can be placed anywhere. The rest of this section describes these optional parameters.

*model name* To give your model a name, use the keyword `name` followed by the desired name. SIMM assumes that all text following the keyword on the same line comprises the name, so you can put spaces in your model name. If you do not name your model in this way, SIMM will assign a default name to it, consisting of the word “model” followed by a number.

```
name Lower Extremity
```

*muscle and motion files* You can also specify the names of the muscle and motion files that you would like to load with the joint file. This is a convenient feature if you often load the same muscle and motion files with a certain joint file. To specify the muscle file, use the keyword `muscle_file` followed by the muscle file name. Then when you load a model by selecting the joint file with the file browser, SIMM will also open the specified muscle file. You can also specify motion files in this way (up to 100 per joint file) using the keyword `motion_file`. The corresponding motions will be added to the model when you load it. The top of your resulting joint file would look something like this:

```
muscle_file leg_model.msl  
motion_file walk.mot  
motion_file run.mot
```

*inverse kinematics* If there are loops in your model, there is an optional parameter to specify whether the inverse kinematics (IK) solver is on or off (it is on by default). Just put the keyword `inverse_kinematics_solver` and specify the

state: `on` or `off`. When SIMM loads the model, it will turn the IK solver on or off appropriately. If the solver is on, and there are loops in the model, SIMM will try to assemble the model so that all the loops are closed, using the default configuration as a starting point. See Section 2.8.5, Inverse Kinematics, for more information.

```
inverse_kinematics_solver on
```

### *colors*

Another optional component of a joint file is the specification of some model-specific colors. Shown below are the five colors that you can specify. `Background_color` refers to the background color of the model window. `Rotation_axes_color` refers to the color of the rotation axes of the joints. `Segment_axes_color` refers to the color of the axes of the body segment reference frames. `Vertex_label_color` refers to the color of the polygon outline and vertex numbers of the selected polygon. `Crosshairs_color` refers to the color of the crosshairs in the model window. All of the RGB values (ranging from 0.0 to 1.0) in the example below are the default values that SIMM uses if you do not specify your own values.

```
background_color 0.2 0.2 0.2
rotation_axes_color 1.0 1.0 0.0
segment_axes_color 1.0 1.0 1.0
vertex_label_color 1.0 0.0 0.0
crosshairs_color 1.0 1.0 1.0
```

### *model units*

You can also specify the length and force units for your model. When you specify units, they are used for two purposes. First, the Motion Module uses them to convert 3D marker coordinates and forceplate data in TRB/C, ANB/C, and C3D files into SIMM model coordinates. Second, the units are used for display purposes when creating plots. In this case, SIMM does not use the units to convert any data values, it merely adds the unit labels to the plot labels. `Length_units` specifies the units of all

lengths and translations in your model. `Force_units` specifies the units of muscle force.

```
length_units m
force_units N
```

*window\_position* This parameter lets you specify the position of the model window when the model is first loaded into SIMM. The X and Y coordinates are relative to the top left corner of the main SIMM window, with X going to the right, and Y going down.

```
window_position 800 20
```

*window\_size* This parameter lets you specify the size of the model window when the model is first loaded into SIMM. The numbers following the keyword are the width and height, respectively.

```
window_size 400 600
```

*gravity* This parameter specifies the direction of the gravity vector, which is used only during dynamic simulations. The default gravity vector is -Y. Available options are: -X, +X, -Y, +Y, -Z, +Z.

```
gravity -Z
```

*solver\_accuracy*  
*solver\_max\_iterations* There are two parameters you can specify to control the accuracy of the least-squares solver module within SIMM. This module is used by the Motion Module to solve frames of marker data, and by SIMM to implement closed loops and constraint objects. For both uses, the module adjusts the model's gencoords iteratively until the markers, loops, and/or constraints (collectively called "solver ordinates") are satisfied to within the specified accuracy, or until the specified maximum number of iterations has been reached. The accuracy value corresponds to the sum of the squares of the errors between the current

solver ordinates and the "ideal" ordinates. Decreasing this value will cause the solver module to iterate longer and return a more accurate solution. Shown below are the parameters with their default values. These parameters can only be specified in a joint file; they cannot be changed in SIMM once the model has been loaded.

```
solver_accuracy 0.0001  
solver_max_iterations 100
```

*loop\_tolerance*

This parameter specifies the "acceptance level" for the solutions that the solver module reaches when implementing the loop constraints. It does not affect the speed or accuracy of the solver module. Rather, it is a cutoff value specifying whether or not the solution is acceptable. If it is not acceptable, a dialog box may be presented indicating that the loop constraint is not satisfied (this notice appears only when loading or editing a model, not when changing gencoord values). This tolerance value corresponds to the distance (in model units) between the two segments of the loop joint (see Section 3.3.3, Joints, for an explanation of the loop joint). When the loop is properly enforced, the points on the two segments are coincident, and the distance is zero. When the loop is not properly enforced, the loop joint disarticulates and the two points are separated by a certain distance. The `loop_tolerance` parameter specifies how large a distance is acceptable. Shown below is the parameter with its default value. Once the model has been loaded into SIMM, this parameter can be changed via the Gencoord Editor.

```
loop_tolerance 0.0001
```

*marker\_visibility*  
*marker\_radius*

These parameters specify properties of all of the markers in the model. `Marker_radius` is the radius of the spheres used to represent the markers in the model window. It affects only their display; it does not affect any marker



offset calculations that the Motion Module performs when loading the model. `marker_visibility` controls the visibility of all of the markers in the model window. It is a convenience feature that allows you to turn all of the markers on or off without selecting them individually and changing their visibility. Even if the markers are not visible, they are still used by the Motion Module when importing tracked marker files. Both of these marker parameters can also be changed using the Marker Editor. Shown below are the parameters with their default values:

```
marker_visibility no
marker_radius 0.005
```

Note: Case is significant when entering text into joint files. All SIMM keywords (*e.g.*, `name`, `beginsegment`) must be in lower-case letters. You are free, however, to use upper- as well as lower-case letters for naming your model components (*e.g.*, names of joints and segments), and SIMM will display these names exactly as you type them into the joint file.

### 3.3.1 Body Segments

A body segment consists of one or more bones fixed in a reference frame. The only required component of a segment definition is a name. You will usually want to specify one or more bone files as well, however, so that you can see the body segment in the model window. Other optional components include the segment's material properties, `drawmode`, `shadow`, `shadow color`, associated deform objects, and display of the segment's reference frame. (Figure 3-3).

Bones are specified using the keyword `bone`, followed by the name of the bone file, and, optionally, the material and

```
/* BODY SEGMENT: right lower leg */
beginsegment right_shank      /* name of this body segment */
bone r_tibia.asc wireframe    /* this bone uses the default material */
bone r_fibula.asc flat_shading my_mat /* this bone overrides the default */
                                /* material and drawmode */

begingroups
right_leg lower_body          /* this segment belongs to two groups */
endgroups
drawmode flat_shading        /* override default and use flat shading */
material my_bone_mat         /* override default, which is def_bone */
shadow Y -0.9                /* draw the shadow directly below */
shadowcolor 40 40 40         /* use a dark gray for the shadows */
axes 0.2                      /* show reference frame, w/axes 0.2 long */
endsegment
```

Figure 3-3. Example body segment

drawmode for displaying the bone. The material is used to set the color and lighting of the bone. The drawmode gives you control over the shading of the bone's surface. These optional drawmode and material identifiers can be placed in any order after the file name. If you do not specify the material or drawmode for a bone, it will inherit the material or drawmode of the segment (as specified with the `drawmode` and `material` keywords). If no material is assigned to the body segment, the default material, `def_bone`, is used. See Section 3.3.7, *Materials*, for information on defining a material. If no drawmode is defined, the default is `gouraud_shading`. There are six drawmodes available: `wireframe`, `solid_fill`, `flat_shading`, `gouraud_shading`, `outlined`, and `none`. `wireframe` means draw just the outlines of the polygons in the bone. `solid_fill` means draw the bone as a collection of filled polygons. For both of these drawmodes, the color used to draw the bone is the ambient color specified in the bone's material (see Section 3.3.7, *Materials*).

`Flat_shading` means use the light sources in the model window to shade the bone polygons, but fill the polygons using only one color each. This drawmode will accentuate the individual faces on the bone surface. `Gouraud_shading` means to smooth-shade each polygon. This may be the slowest drawmode because it uses the light sources to calculate the color of each pixel on the bone surface so that it is rendered smoothly, hiding the polygon edges. `Outlined` means to draw the polygons in the bone's normal material, and highlight the edges in a darker color. `None` means to not display the body segment at all. You can also interactively change the drawmode of any bone or world object using the **drawmode** command in the Model Viewer, or by right-clicking on the bone or world object in the model window. If you interactively change the material or drawmode, the new values will be written to the joint file when you save the model.

The `begingroups` and `endgroups` keywords can be used to enclose the name(s) of one or more body segment groups to which the segment belongs. Organizing your body segments into groups can make it easier to navigate and display your model with the SIMM Model Viewer.

The `shadow` parameter is used to specify the direction and location of the body segment's shadow. In the example shown in Figure 3-3, the shadow of the femur would be displayed at a Y-value of -0.9 meters from the origin of the world reference frame, and would look as if the light casting the shadow were positioned on the Y-axis. The `shadowcolor` parameter can be used to specify the red, green, and blue components, respectively, of the shadow's color. Each of the components can range in value from 0 to 1.0, and the default color is 0.08, 0.08, 0.08. To display a body segment's shadow in the model window, use the **shadows** toggle button in the Model Viewer (see Section

2.4.5, Toggle Buttons). When you turn shadows on, only those body segments for which you have specified shadow properties will cast a shadow.

The `axes` parameter is used to display the coordinate system of the body segment in the model window. The number after the keyword `axes` is the length of each vector forming the reference frame.

### 3.3.2 Fixed Segment

The fixed segment is the body segment that does not move when you move the joints in your musculoskeletal model. That is, the origin of this segment is fixed, but the segment is allowed to rotate about its origin (if the joints in your model allow this). If you define a segment named “ground,” then this segment is automatically marked as the fixed segment. Otherwise, the segment whose name appears first in the joint file is marked as the fixed segment. Note that segment names are used in joint definitions as well as segment definitions, so the order in which you enter joints may also matter. For example, in the sample lower-extremity file supplied with SIMM, `pelvis` is the first body segment referenced in the joint file, so that segment is the fixed segment. When you flex, adduct, or rotate the hip, the femur rotates with respect to the pelvis, which remains fixed in the world reference frame. If the joint file were altered so that the femur were referenced first, then the pelvis would rotate with respect to the femur.

If you explicitly define a ground segment, you can define its characteristics just like you can with any other body segment. You can specify one or more bone files so that you can see the ground segment in the model window.

You can interactively change the fixed segment using the **look at** command in the Model Viewer (see Section 2.4.2, Command Menu, for details). When you select a body segment with this command, that segment becomes the new fixed segment. If you write the model to a file, however, this new fixed segment is not marked as such in the output file.

### 3.3.3 Joints

The body segments can be connected in any arrangement by defining joints. A joint specifies the transformations that relate the position and orientation of one body segment to another. The body-fixed transformations are performed in a user-defined order, and consist of three orthogonal translations and three rotations around user-defined axes. Each translation and rotation is either constant (*i.e.*, independent of body position) or a function of a generalized coordinate. A generalized coordinate defines a degree of freedom in the musculoskeletal model (*e.g.*, a joint angle). The functions that relate the translations and rotations to the generalized coordinates are called kinematic functions. A sample joint definition is shown in Figure 3-4.

The translations and rotations that comprise a joint are called *dofs*. The rotation dofs ( $r_1$ ,  $r_2$ ,  $r_3$ ) are rotations around `axis1`, `axis2`, and `axis3`, respectively. The translation dofs ( $t_x$ ,  $t_y$ ,  $t_z$ ) are translations along the X, Y, and Z axes (respectively) of the first body segment listed after the `segments` keyword. The second body segment listed is the one on the other side of the joint (the reference frame you end up after doing the ordered transformations). To make any of the dofs a constant, put the keyword `constant` after the name of the dof, then enter a value. This value should be in degrees for the rotations.

```

/* FEMORAL-TIBIAL JOINT */
beginjoint femoral-tibial      /* name of this joint */
segments femur tibia          /* joint from FEMUR frame to TIBIA frame */
order t r1 r2 r3              /* translate, then rotate about axis1, axis2,
                              and axis3 */

axis1 0.0 0.0 1.0
axis2 1.0 0.0 0.0              /* rotation axis definitions */
axis3 0.0 1.0 0.0

loopjoint                      /*
show axis1 0.2                 /* display rotation axis1, length = 0.2 */
tx function f1 (knee_angle)    /* The kinematic functions f1 and f2 */
ty function f2 (knee_angle)    /* specify the relations between the */
tz constant 0.0                /* generalized coordinate (knee_angle) */
r1 function f3 (knee_angle)    /* and tx and ty, the translations in */
r2 constant 0.0                /* the x and y directions, respectively. */
r3 constant 0.0                /* f3 specifies the relation between */
endjoint                       /* knee_angle and the rotation around axis1. */

```

Figure 3-4. Example joint definition

For translations you can use any units you like, but they must match the units used in the bone and muscle files (meters are used in the sample leg model). To make a dof a function of some generalized coordinate, put the keyword `function` after the name of the dof, then enter a function name, followed by the name of a generalized coordinate, in parentheses. The function name must start with the letter `f`, followed by a number.

In the example in Figure 3-4, the femoral-tibial joint definition (part of the knee joint) specifies the transformations from the `femur` segment to the `tibia` segment. The order of the transformations indicates that to move from the `femur` reference frame to the `tibia` reference frame, a translation is performed first, followed by rotations around `axis1`, `axis2`, and `axis3`, respectively. The translation is divided into three orthogonal components (`tx`, `ty`,

`tz`), where the X, Y, and Z directions are defined by the axes of the `femur` reference frame. The rotation axes are also defined relative to the `femur` reference frame, and their orientations can be changed by adjusting the axis definitions. Since all of the translations and rotations in this joint are either constants or functions of a single generalized coordinate (`knee_angle`), the joint has only one degree of freedom. It is not a simple revolute, however, because two of the translations (`tx`, `ty`) are functions of `knee_angle`.

If you want to display any of the axes of rotation in the model window, put the keyword `show` in the joint definition, followed by the name of the axis (*i.e.* `axis1`, `axis2`, or `axis3`) and the desired length of the axis. This can be helpful when verifying a new joint definition or when visualizing moment arm calculations.

The keyword `loopjoint` should no longer be needed in a joint definition, now that SIMM includes an inverse kinematics (IK) solver. If you have loops of body segments in your model, the IK solver will find `gencoord` values that satisfy the loop constraints and keep the loops closed. If you turn the IK solver off, or if it cannot find a configuration which keeps the loops closed, SIMM will display your model with one of the joints in the loop broken. This so-called “loop joint” will be ignored for display and plotting purposes, so you won’t be able to animate the joint or plot muscle properties as a function of the joint’s `gencoords`. To specify which joint should act as the loop joint, put the keyword `loopjoint` in the joint definition. In most cases, however, it is recommended that you do not specify loop joints, and instead allow the IK solver to control the loops in your model.

### 3.3.4 Generalized Coordinates

Generalized coordinates are degrees of freedom in the musculoskeletal model. The definition of the generalized coordinate `knee_angle`, from the femoral-tibial joint definition, is shown in Figure 3-5. A range of motion is defined that limits the values of the generalized coordinate and serves as the independent variable when making plots. If the generalized coordinate represents a rotational degree of freedom, the range should be specified in degrees. If the `gencoord` is translational, the units should match the units of the other lengths and translations that you specify in your model. Also specified is the keyboard key (`k_key`) that controls the value of the generalized coordinate when animating the model in the model window. You can specify either one or two keys to control the generalized coordinate, and they must be on the same line as the word `keys` in the joint file. The keyword `wrap` is an optional parameter that affects the control of the generalized coordinate in the Model Viewer. When you include `wrap` in the definition, you can repeatedly sweep through the range of motion for the generalized coordinate by

```
/* Generalized Coordinate: knee angle */
begingencoord knee_angle      /* name of gencoord */
begingroups
right_leg lower_body
endgroups
range 0.0 -120.0             /* range of motion for this gencoord */
keys k_key                   /* press this key to flex the knee */
default_value -30.0         /* starting, and default value of gencoord */
wrap                          /* lets you loop through range of motion */
endgencoord
```

Figure 3-5. Example generalized coordinate



resetting to the beginning of the range when you reach one end. This is described in more detail in Section 2.4, Model Viewer. To define a default value for a generalized coordinate, use the keyword `default_value` followed by some number in the range of motion. This will be the starting value of the generalized coordinate, and the value it will be reset to when you select *reset gc values* in the Model Viewer.

The keyword `clamped` is an optional parameter that lets you specify whether or not the gencoord is constrained to remain within its specified range. By default, gencoords are clamped and cannot assume values outside their range. The keyword `locked` is an optional parameter that lets you specify whether or not the gencoord is locked. By default, gencoords are unlocked. If a gencoord is locked, its value cannot change.

The keyword `active` is an optional parameter that can be used when a restraint function is associated with the gencoord. If the function is active, it will be used by the inverse kinematics solver when trying to close loops. If the function is inactive, it has no effect. This is useful if you want to use the function in the pipeline, but don't want to use it to solve loops. If you define a restraint function for a gencoord, it is active by default.

Restraint functions can be specified for each gencoord. These are used in the Pipeline to apply restraint torques to prevent the gencoords from moving beyond their ranges of motion. They are also used by the inverse kinematics solver to assign weights to the gencoords when finding model configurations to satisfy the loop constraints. The restraint functions are specified using the keyword `restraint` and a function name in the same format as that used for kinematic functions.

The `begingroups` and `endgroups` keywords can be used to enclose the name(s) of one or more `gencoord` groups to which the `gencoord` belongs. Organizing your generalized coordinates into groups can make it easier to navigate and display your model with the Model Viewer.

### 3.3.5 Kinematic Functions

Kinematic functions define translations and rotations between body segments as functions of the generalized coordinates. For example, the anterior-posterior translation ( $t_x$ ) between the femur and the tibia as a function of knee angle is specified by the kinematic function, `f1`, shown in Figure 3-6. The (`knee_angle`,  $t_x$ ) pairs are interpolated by a natural cubic spline. When SIMM needs the value of  $t_x$  to calculate a joint transformation, it evaluates the cubic spline function using the current value of `knee_angle`. When defining a kinematic function, you should make sure that you define it over a large enough X-range so that it is defined over the complete range of motion of every generalized coordinate that uses it. If you do not, SIMM will perform a linear extrapolation of the function to obtain any Y-values that it needs. The X-coordinates, like the range of motion of `gencoords`, should be specified in degrees for rotational `gencoords`. For translational functions, you should use units that are consistent with the other lengths and translations that you specify in your model.

Each pair of data points in the function should be entered between parentheses, with a comma separating the X and Y values. You can also put blank spaces between the values, commas, and parentheses.

```
/* KINEMATIC FUNCTION: Tx of the femoral-tibial joint */
beginfunction f1 /* defines x translation from femur to tibia */
/* knee_angle (degrees), tx (meters) */
(-120.0,-0.003)
(-100.0, 0.001)
( -80.0, 0.004)
( -60.0, 0.004)
( -40.0, 0.002)
( -20.0,-0.001)
( -10.0,-0.003)
(  0.0,-0.005)
endfunction
```

Figure 3-6. Example kinematic function

### 3.3.6 Restraint Functions

Restraint functions are used primarily for controlling gencoords during forward dynamic simulations performed with the Dynamics Pipeline. The functions define forces or torques that are applied to the gencoords in order to restrain them from going outside their ranges of motion. However, restraint functions are also used by SIMM's inverse kinematics (IK) solver when you have one or more closed loops of body segments. In this case, the restraint functions define error values that restrain the gencoords from going outside their ranges of motion while the IK solver is satisfying loop constraints (see Section 2.8.5, Inverse Kinematics, for more details on the IK solver).

If there are no loops of body segments in your model, the IK solver will be inactive, and the gencoords' restraint functions will not be used in SIMM. If your model does have loops, you may want to define restraint functions to

better control the IK solver. If you do define restraint functions for use with IK, you should define them so that they also have appropriate dynamic behavior, and thus can be used with the Dynamics Pipeline as well.

As with kinematic functions, natural cubic splines are fit to the control points that define restraint functions. Since a restraint function is defined for the entire range of motion of a gencoord, it is important to carefully define its values within this range. If you want the function to be exactly zero for a portion of the range (which is usually the case for dynamic simulations), you should define knots (coincident control points) in the appropriate places.

For example, suppose that a gencoord named *genc1* has a range of motion of -30 degrees to +60 degrees, and that if it goes beyond either end of its range, you want to apply a torque to drive it back into its range. Furthermore, suppose that you would like to start applying the restraint torque when the gencoord gets within 5 degrees of its limit, to make it less likely that it will go beyond its range. In this case, your restraint function would look something like the one in Figure 2-1. Note that when the gencoord goes below its range, a positive torque will be applied (to increase the gencoord value), and when it goes above its range a negative torque will be applied (to decrease its value).

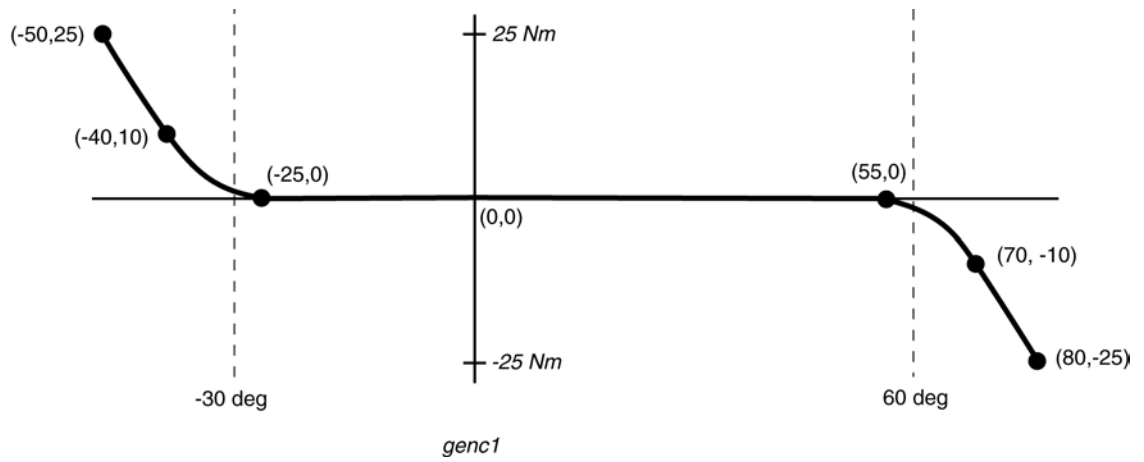


Figure 3-7. Example restraint function.

The gencoord and function definitions in this example would look like this:

```
beginencoord genc1
range -30 60
restraint f31
endencoord
```

```
beginfunction f31
(-50.0, 25.0)
(-40.0, 10.0)
(-25.0, 0.0)
(-25.0, 0.0)
(55.0, 0.0)
(55.0, 0.0)
(70.0, -10.0)
(80.0, -25.0)
endfunction
```

Note the coincident control points at -25 and 55 degrees. This is very important so that the restraint function has a value of exactly zero between -25 and 55 degrees.

### 3.3.7 World Objects

World objects are polygonal surfaces that are fixed to the world reference frame. They can be used to enhance the display of a model by providing a setting for the musculoskeletal structure. Typical examples include a floor, walls, stairs, and a bottom bracket and seat for a cycling model. A sample world object is shown in Figure 3-8. The file that contains the object's surface description must be in the same format as a bone file (see Section 3.2, Bone Files), and should be processed with *norm* (see Chapter 4, Norm) before being loaded into SIMM. You can position the object anywhere in the world reference frame using the `origin` parameter. Just specify the X, Y, and Z coordinates of the object's origin with respect to the origin of the world reference frame. To control the appearance of the object, assign a material to it, the same way you do for a body segment. If you do not assign a material, the default material, `def_bone`, will be used. The `drawmode` parameter gives you additional control over the shading of the object's surface. As with body segments, `flat_shading`, `gouraud_shading`, `wireframe`, `outlined`, `solid_fill`, and `none` are supported. The default drawmode is `solid_fill`.

```
/* WORLD OBJECT: floor */
beginworldobject floor      /* name of this world object */
filename floor.asc         /* norm-ed binary file describing surface */
origin 0.0 -0.9 0.0        /* vector from world ref frame to object */
material floor_material    /* use this material, defined in joint file */
drawmode flat_shading     /* override default and use flat shading */
endworldobject
```

Figure 3-8. Example world object

### 3.3.8 Materials

Materials define the color and surface reflectivity of objects. Once you have defined a material, you can assign it to any number of body segments, world objects, muscles, and motion objects. If you do not assign a material to a bone or world object, the default bone material, `def_bone`, is assigned to it. See Figure 3-10 for a complete list of the default materials and how they are used. A sample material definition is shown in Figure 3-9. You should consult the OpenGL Programming Guide for a complete description of material properties, but a brief summary follows. The keywords `ambient`, `diffuse`, `specular`, and `emission` should each be followed by the red, green, and blue color components of that material property. These values range from 0.0 to 1.0. `Ambient` refers to the background illumination which hits all sides of an object. `Diffuse` illumination is brightest where the light source strikes the object perpendicular to the surface. `Specular` lighting accounts for highlights or glare when

```
/* MATERIAL: default bone */
beginmaterial def_bone      /* def_bone is the name of this material */
ambient 0.7 0.7 0.7        /* red, green, and blue components */
diffuse 0.6 0.45 0.4       /* red, green, and blue components */
specular 0.7 0.55 0.4      /* red, green, and blue components */
emission 0.0 0.0 0.0       /* red, green, and blue components */
shininess 10.0             /* not very shiny, 10 out of 128 */
endmaterial
```

Figure 3-9. Example material

the light reflects off the surface directly towards the viewer. `Emission` is the property of giving off light, which happens equally at all locations on the object's

```
beginmaterial def_motion /* used for motion objects */
ambient 0.1 0.7 0.1
diffuse 0.2 0.3 0.2
specular 0.2 0.0 0.2
shininess 1.0
endmaterial

beginmaterial def_bone /* used for bones and world objects */
ambient 0.7 0.70 0.7
diffuse 0.6 0.45 0.4
specular 0.7 0.55 0.4
shininess 10.0
endmaterial

begin material def_sel_poly /* used for the selected polygon */
ambient 0.2 0.70 0.2
diffuse 0.1 0.45 0.1
specular 0.4 0.65 0.4
shininess 10.0
endmaterial

beginmaterial def_bone_outline /* used for outlined_polygons drawmode */
ambient 0.2 0.0 0.0
diffuse 0.2 0.0 0.0
specular 0.2 0.0 0.0
shininess 90.0
endmaterial

beginmaterial def_min_muscle /* used for muscles with an activation of 0.0 */
ambient 0.2 0.0 0.0
diffuse 0.4 0.0 0.0
specular 1.0 0.15 0.15
shininess 40.0
endmaterial

beginmaterial def_max_muscle /* used for muscles with an activation of 1.0 */
ambient 0.4 0.0 0.0
diffuse 0.7 0.0 0.0
specular 1.0 0.15 0.15
shininess 90.0
endmaterial

beginmaterial def_muscle_point /* used for muscle points when all are displayed */
ambient 0.0 0.0 0.0
diffuse 0.0 0.0 0.7
specular 0.15 0.15 1.0
shininess 90.0
endmaterial

beginmaterial def_sel_muscle_point /* used for selected muscle points */
ambient 0.0 0.0 0.0
diffuse 0.7 0.7 0.0
specular 1.0 1.0 0.2
shininess 90.0
endmaterial
```

Figure 3-10. SIMM's default materials



surface. Finally, the `shininess` value, which ranges from 0.0 to 128.0, specifies the shininess of the object's surface, and controls the size and brightness of the specular highlight.

Note: When an object (body segment or world object) is displayed in wireframe or `solid_fill` modes, the color used to draw it is the `ambient` color specified in the material assigned to that object.

SIMM has eight materials stored internally that it uses to draw various elements of a model. You can override these defaults by re-defining the materials in your joint file. These eight materials, with their default values, are shown in Figure 3-10.

### 3.3.9 Cameras

Cameras (formerly known as *model views*), define the position and orientation, or pose, of a camera relative to the model. Once you have loaded a model into SIMM, you can set the camera pose to any of the poses that you specified in the joints file, by using the **restore camera** command (see Section 2.4.2, Command Menu, for details). An example camera definition is shown in Figure 3-11. Each camera is specified by a name and a 4x4 transformation matrix. The 3x3 matrix in the upper left corner specifies the rotations of the model around the axes of the world reference frame. The first three numbers in the last row of the matrix specify the translation in the X, Y, and Z directions from the origin of the world frame (the Z translation should always be negative). You can enter cameras directly into a joint file if you know the exact coordinates, but it is usually easier to manipulate the camera interactively within SIMM, use the **save camera** command (see Section 2.4.2, Command Menu) to save the transforma-

```
/* Camera Pose */
begincamera posterior /* posterior is the name of this pose */
1.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0
0.2 0.3 -1.0 1.0
endcamera
```

Figure 3-11. Example camera

tion matrix to a buffer, then write the model to a file so that you have the precise 4x4 matrix saved to a file. You can enter up to five cameras in each joints file.

### 3.3.10 Wrap Objects

Wrap objects are geometric primitives that automatically deflect muscles to prevent them from penetrating the wrap object. Wrap objects are typically added to a model such that their surface approximates the surface of one or more of the model's bones. The only required component of a wrap object definition is its name. However you will usually want to specify its associated body segment and its placement within the segment's reference frame as well. Wrap objects may be created interactively within SIMM using the Wrap Editor tool. You may find it easier to create and modify wrap objects with the Wrap Editor rather than editing the joint file directly. See Section 2.9, Wrap Editor, for more information.

Wrap objects are specified in the model's joint file. However before a muscle can wrap over a wrap object, the muscle must be *associated* with that wrap object. Associations between muscles and wrap objects are specified in the muscle's definition in the muscle file (see Section

```

/* Example Wrap Object */
beginwrapobject hip_wrapper
wraptype ellipsoid
segment pelvis
xyz_body_rotation 20.0 0.0 0.0
translation -0.0826 -0.0286 0.0808
radius 0.05 0.02 0.05
active yes
visible no
quadrant -y
endwrapobject

```

Figure 3-12. Example wrap object

3.4.1, Muscles). Muscle wrap associations may also be specified within SIMM using the Wrap Editor tool. The fields in a wrap object definition, shown below, specify the size, position, and orientation of the geometric primitive, as well as method of wrapping.

#### **Wrap Object Fields**

wraptype	This parameter specifies the type of geometric primitive to use for the wrap object. The valid types are: <code>sphere</code> , <code>cylinder</code> , <code>ellipsoid</code> , and <code>torus</code> . If no wrap type is specified, the default type, <code>sphere</code> , is used.
segment	This parameter specifies the body segment to which the wrap object is attached.
xyz_body_rotation	This parameter specifies the orientation of the wrap object as an XYZ Euler rotation. The orientation is calculated by first applying the X rotation around its local X axis, then the Y rotation around its new local Y axis, and finally applying the Z rotation around its new local Z axis. If the rotation is not specified in the wrap object definition, the default values are <code>0.0</code> , <code>0.0</code> , <code>0.0</code> .

`translation` This parameter specifies the position of the wrap object's origin with respect to the origin of its body segment. It is expressed as an X, Y, Z translation vector. If no translation is specified, the default values are 0.0, 0.0, 0.0.

*object dimensions* The dimensions of the wrap object are specified with several keywords, depending on the type of the object, as described below.

*sphere*

The radius of the sphere is specified by the keyword `radius` followed by a single value.

```
radius 0.15
```

*cylinder*

The radius of the cylinder is specified by the keyword `radius` followed by a single value. The height of the cylinder is specified by the keyword `height` followed by a single value.

```
radius 0.10  
height 0.25
```

Note: The cylinder's height is used when calculating muscle wrapping. If *both* of the tangent points (the points of contact between the muscle and the wrap object) are beyond the visible portion of the cylinder, the muscle is considered to miss the wrap object and is not wrapped.

*ellipsoid*

The sizes of the ellipsoid in the X, Y, and Z directions are specified with the keyword `radius` followed by the X, Y, and Z sizes, respectively.

```
radius 0.1 0.2 0.1
```

*torus*

The inner and outer radii of the torus are specified by the keyword `radius` followed by the inner and outer radii, respectively. The outer radius is the distance from the center of the torus to the axis of the curved cylindrical section, and the inner radius is the radius of the cylindrical section itself (and thus should always be less than the outer radius).

```
radius 0.03 0.08
```

<code>active</code>	This parameter may be used to disable the wrap object. If its value is <code>no</code> or <code>false</code> , then the wrap object will have no effect on its associated muscles.
<code>visible</code>	This parameter may be used to disable the display of the wrap object's shape in the model window. If this parameter's value is <code>no</code> or <code>false</code> , then the wrap object will not be visible in the model window, but its associated muscles will still wrap over it.
<code>show_wrap_pts</code>	This parameter may be used to display the XYZ coordinates of muscle tangent points for each muscle that wraps over the wrap object. If this parameter's value is <code>yes</code> or <code>true</code> , then the tangent point coordinates will be displayed in the model window. The display of these points can be helpful when creating the wrap object or adjusting muscle attachment points to improve the wrapping behavior.
<code>quadrant</code>	This parameter may be used to limit wrapping to one half of the wrap object. The valid quadrants are: <code>x</code> , <code>-x</code> , <code>y</code> , <code>-y</code> , <code>z</code> , or <code>-z</code> for spherical and ellipsoidal wrap objects, and <code>x</code> , <code>-x</code> , <code>y</code> , or <code>-y</code> for cylindrical wrap objects.
<code>wrapmethod</code>	This parameter may be used to specify the algorithm to be used to compute muscle wrapping paths around ellipsoidal wrap objects. The valid wrap methods are: <code>hybrid</code> ,

midpoint, and axial. For more information, see Section 2.9.5, Ellipsoid Wrapping Methods.

### 3.3.11 Deform Objects

Deform objects allow you to twist, bend, and warp portions of a body segment that would normally be considered rigid. When a segment is deformed, all bone vertices, distal joint centers, muscle attachment points, and muscle wrapping objects in the deformed region are transformed accordingly. It is usually easier to create new deform objects using the SIMM Deform Editor rather than editing a joint file directly. See Section 2.10, Deform Editor, for information on how to do this. However, it can be useful in certain situations to create or modify deform objects manually by editing the model's joint file.

Deform objects are defined within a segment definition in the model's joint file, within a `begindeform ... enddeform` pair. All deform object fields have reasonable default values, so it is possible to specify a simple deform with just two lines as shown in Figure 3-13.

```
beginsegment
...
begindeform my_simple_deform
enddeform
...
endsegment
```

*Figure 3-13. Minimal deform object specification, located within a segment definition*

In most cases, however, you will also define the dimensions of the deform object's inner and outer boxes, the transformation necessary to position the deform within the

reference frame of its associated body segment, and the starting and ending transformations of the inner box that creates the object's deformation (see Figure 3-14).

```

deform neck_angle
innermin -0.05 -0.05 -0.080
innermax  0.05  0.05 -0.025
outermin -0.07 -0.07 -0.100
outermax  0.07  0.07 -0.005
xyz_body_rotation_POSITION 33.6900 -25.6600 16.1100
translation_POSITION -0.0153 -0.0279 0.0287
xyz_body_rotation_DEFORM_START -80 0 0
translation_DEFORM_START 0 0 0
xyz_body_rotation_DEFORM_END 80 0 0
translation_DEFORM_END 0 0 0
enddeform

```

Figure 3-14. Example deform object

If the deform object is an *auto-reset deform*, then it is not necessary to specify the starting and ending transformations since they will be computed automatically. Also remember that auto-reset deform objects should be the last deform object within a segment. Figure 3-15 shows a typical auto-reset deform object specification.

### **Deform Object Fields**

innermin, innermax

These fields specify the dimensions of the deform object's inner box. All bone vertices, distal joint centers, muscle attachment points, and wrap objects within the deform object's inner box will receive the full effect of the object's deformation. These two keywords must each be followed by three numbers that specify an XYZ coordinate in the deform object's reference frame. The default values for

`innermin` are -0.03, -0.03, -0.03, and the default values for `innermax` are 0.03, 0.03, 0.03.

`outermin`, `outermax`

These fields specify the dimensions of the deform object's outer box. The object's deforming effect is limited to the region enclosed by its outer box. These two keywords must each be followed by three numbers that specify an XYZ coordinate in the deform object's reference frame. The default values for `outermin` are -0.04, -0.04, -0.04, and the default values for `outermax` are 0.04, 0.04, 0.04.

`xyz_body_rotation_POSITION`

This field specifies the deform object's rotation relative to its associated body segment. This keyword must be followed by three numbers that specify body-fixed rotations that are applied in X, then Y, then Z order. The default values are 0.0, 0.0, 0.0.

`translation_POSITION`

This field specifies the deform object's translation relative to its associated body segment. This keyword must be followed by three numbers that specify the XYZ translation from the segment's origin. The default values are 0.0, 0.0, 0.0.

```
deform my_auto_reset_deform
autoreset yes
translationonly yes
innermin -0.05 -0.44 -0.08
innermax 0.05 0.05 0.08
outermin -0.07 -0.45 -0.1
outermax 0.07 0.1 0.1
xyz_body_rotation_POSITION 33.6900 -25.6600 16.1100
translation_POSITION      -0.0153 -0.0279 0.0287
enddeform
```

Figure 3-15. Example of an auto-reset deform



---

<code>xyz_body_rotation_DEFORM_START</code>	This field specifies a rotation of the deform object's inner box relative to the deform object's reference frame. This rotation specifies the deform object's starting deformation. The default values are 0.0, 0.0, 0.0.
<code>translation_DEFORM_START</code>	This field specifies a translation of the deform object's inner box relative to the deform object's reference frame. This translation specifies the deform object's starting deformation. . The default values are 0.0, 0.0, 0.0.
<code>xyz_body_rotation_DEFORM_END</code>	This field specifies a rotation of the deform object's inner box relative to the deform object's reference frame. This rotation specifies the deform object's ending deformation. The default values are 0.0, 0.0, 0.0.
<code>translation_DEFORM_END</code>	This field specifies a translation of the deform object's inner box relative to the deform object's reference frame. This translation specifies the deform object's ending deformation. The default values are 0.0, 0.0, 0.0.
<code>active</code>	This field specifies whether the deform object is enabled or not. When disabled, deform objects have no affect on their associated body segment. This keyword must be followed by the word <code>yes</code> , <code>no</code> , <code>true</code> , or <code>false</code> . The default value is <code>yes</code> .
<code>visible</code>	This field specifies whether the deform object is visible in the 3D model window or not. This keyword must be followed by the word <code>yes</code> , <code>no</code> , <code>true</code> , or <code>false</code> . The default value is <code>yes</code> .
<code>autoreset</code>	This field specifies whether the deform object is an auto-reset deform or not. If this keyword is followed by the word <code>yes</code> , or <code>true</code> , then the deform is an auto-reset deform and will behave as such. See Section 2.10.5, Combining Multiple Deform Objects, for more information about auto-reset deforms. The default value is <code>no</code> .

---

`translationonly` This field specifies whether an auto-reset deform should limit its deformation to translation only, instead of both translation and rotation. If this keyword is followed by the word `yes`, or `true`, and the deform is an auto-reset deform, then the auto-reset deform will translate the segment's origin back to its original, undeformed position but it will not apply a rotation to realign the segment's reference frame with its original axes. The default value is `no`.

### 3.3.12 Deformities

Deformity sliders are specified in the model's joint file, after all of the segment definitions. There is no way to create or modify deformity sliders within SIMM, so they must be added to the model's joint file manually. Each deformity defined in the model's joint file appears as a slider at the bottom of the SIMM Deform Editor window.

```

begindeformity anteversion
range -20.0 60.0
value 0.0
deform anteversion_twist
deform neck_angle
deform auto_reset
enddeformity

```

Figure 3-16. Example deformity

#### **Deformity Fields**

`range` This field specifies the minimum and maximum allowable values of the deformity slider. The default values for `range` are 0.0 to 100.0.

`value` This field specifies the default value of the deformity slider. The default value is 0.0.

`deform` This field specifies the name of a deform object that has been specified earlier in the same joint file. A deformity slider may contain one or more deform fields. Each deform object associated with the deformity will be controlled by the deformity's slider in the Deform Editor window.

### 3.3.13 Motion Objects

Motion objects are shapes that are added to the display of the model while a motion is being played back. They are useful for displaying ground-reaction forces, joint contact forces, contact points, and other values calculated by external software packages. The basic shape of the motion object is specified in a SIMM bone file. For each frame of data in a motion file, you can specify the position, orientation, size, and color of the motion object. See Section 3.5.3, Motion Objects, for a description of how to specify these properties.

Motion objects are automatically displayed when any of their parameters appear as columns of data in a motion file. However, before a motion object can be referenced in a motion file it must first be defined in the joint file, or be one of the motion objects built into SIMM. SIMM provides six built-in motion objects named `force`, `ball`, `contact`, `joint_force`, `joint_torque`, and `spring_force` that make it easy to visualize vectors and points in SIMM animations. The `ball` object is represented by a sphere defined in the SIMM bone file *unit\_sphere.asc*. The other five objects are each represented by an arrow defined in the file *arrow.asc*. You can override the parameters of these shapes, or define additional shapes by defining motion object in your joint file.

```
beginmotionobject box
filename unit_box.asc
position 0.0 0.0 -0.5
scale 0.1 0.3 0.05
xyzrotation 0.0 45.0 0.0
material my_red_material
drawmode flat_shading
vectoraxis z
endmotionobject
```

Figure 3-17. Example motion object

### **Motion Object Fields**

filename	This field specifies the name of the bone file containing the motion object's shape.
position	This field specifies the starting X, Y, and Z position of the motion object in its parent segment's frame of reference. The default value for this field is 0.0, 0.0, 0.0.
scale	This field specifies the starting X, Y, and Z scale factors of the motion object. The default value for this field is 1.0, 1.0, 1.0.
xyzrotation	This field specifies the starting X, Y, and Z rotations of the motion object (in degrees). The motion object's starting orientation is achieved by first applying the X rotation around its local X axis, then the Y rotation around its new local Y axis, and finally applying the specified Z rotation around its new local Z axis (this is known as an XYZ Euler rotation). The default value for this field is 0.0, 0.0, 0.0.
material	This field specifies the name of the material used to determine the color and shading of the motion object's surface. The material must be defined elsewhere in the joint file, or

be a default SIMM material. The default value for this field is `arrow_mat`. See Section 3.3.7, Materials, for information on defining materials, and for a list of default materials.

<code>drawmode</code>	This field specifies the drawing mode for the motion object. The six drawmodes are: <code>solid_fill</code> , <code>wireframe</code> , <code>flat_shading</code> , <code>gouraud_shading</code> , <code>outlined</code> , and <code>none</code> . The default value for this field is <code>gouraud_shading</code> .
<code>vectoraxis</code>	This field specifies the axis of the world object that is used when animating with <code>_vx</code> , <code>_vy</code> , <code>_vz</code> components in the motion file. The motion object will be rotated so that the specified axis coincides with the vector formed by the <code>_vx</code> , <code>_vy</code> , <code>_vz</code> components in the motion file. The motion object will also be scaled along this axis. The default value for this field is <code>y</code> .

### 3.3.14 Constraint Objects

Constraint objects provide a way of constraining points on one body segment to remain on the surface of a geometric primitive (the constraint object) on a second segment. This mechanism is useful for creating joint kinematics that are too complex to be modeled with a traditional SIMM joint. For example, in the shoulder, two (or more) points on the scapula can be constrained to remain in contact with the ribcage, which is represented as an ellipsoid. Once this constraint is activated, when you change the value of one of the generalized coordinates in the shoulder, SIMM will adjust the values of the other generalized coordinates in order to satisfy the constraints.

It is usually easier to create new constraint objects using the Constraint Editor rather than editing a joint file directly. See Section 2.13, Constraint Editor, for informa-

tion on how to do this. However, it can be useful in certain situations to create or modify constraint objects manually by editing the model's joint file.

```
/* Example constraint object */
beginconstraintobject r_scapula_con
constrainttype ellipsoid
segment ribcage
xyz_body_rotation -9.12 16.21 10.42
translation -0.0399 0.1799 0.0743
radius 0.0800 0.1800 0.0800
active yes
visible no
quadrant -x
beginpoints
/* name      X      Y      Z      weight  segment  tolerance */
  c1 -0.0900 -0.1800 -0.0520  1.0000  r_scapula  0.001
  c2 -0.0950 -0.1400 -0.0700  2.0000  r_scapula  0.001
  c3 -0.0550 -0.0900 -0.0550  1.0000  r_scapula  0.002
endpoints
endconstraintobject
```

Figure 3-18. Example constraint object

A constraint object is defined by a geometric primitive (e.g., sphere) fixed to one body segment, and one or more points fixed to other body segments. The points are constrained to remain on the surface of the geometric primitive at all times. The fields in a constraint object definition, shown below, specify the size, position, and orientation of the geometric primitive, as well as the locations of the points.

**Constraint Object Fields**

<code>constrainttype</code>	This field specifies the type of the geometric primitive. The available types are: <code>sphere</code> , <code>ellipsoid</code> , <code>cylinder</code> , and <code>plane</code> . If no type is specified, the default type, <code>sphere</code> , is used.
<code>segment</code>	This parameter specifies the body segment to which the geometric primitive (constraint object) is attached.
<code>xyz_body_rotation</code>	This parameter specifies the orientation of the constraint object as an XYZ Euler rotation. The orientation is calculated by first applying the X rotation around its local X axis, then the Y rotation around its new local Y axis, and finally applying the Z rotation around its new local Z axis. If the rotation is not specified in the constraint object definition, the default values are <code>0.0</code> , <code>0.0</code> , <code>0.0</code> .
<code>translation</code>	This parameter specifies the position of the constraint object's origin with respect to the origin of its body segment. It is expressed as an X, Y, Z translation vector. If no translation is specified, the default values are <code>0.0</code> , <code>0.0</code> , <code>0.0</code> .
<code>object dimensions</code>	The dimensions of the constraint object are specified with several keywords, depending on the type of the object, as described below.

*sphere*

The radius of the sphere is specified by the keyword `radius` followed by a single value.

```
radius 0.15
```

*cylinder*

The radius of the cylinder is specified by the keyword `radius` followed by a single value. The height of the cyl-

inder is specified by the keyword `height` followed by a single value.

```
radius 0.10  
height 0.25
```

Note: the height of the cylinder is used for enforcing the constraint. The constraint points are required to remain on the section of the cylinder defined by the height value.

### *plane*

The width (X dimension) and the length (Y dimension) of the plane are specified by the keyword `radius` followed by the width and length, respectively.

```
radius 0.2 0.3
```

Note: the width and length of the plane are used for enforcing the constraint. The constraint points are required to remain on the section of the plane defined by these two values.

### *ellipsoid*

The sizes of the ellipsoid in the X, Y, and Z directions are specified with the keyword `radius` followed by the X, Y, and Z sizes, respectively.

```
radius 0.2 0.3 0.2
```

`active` This parameter may be used to disable the constraint object. If its value is `no` or `false`, then the constraint object will have no effect on the model.

`visible` This parameter may be used to disable the display of the constraint object's shape in the model window. If this parameter's value is `no` or `false`, then the constraint object will not be visible in the model window, but it will still be enforced.



- `quadrant` This parameter may be used to limit the constraint object to one half of the geometric primitive. When the object is limited in this way, the constraint points are required to remain in contact with the selected half of the object. The valid quadrants are: `x`, `-x`, `y`, `-y`, `z`, or `-z` for spherical and ellipsoidal constraint objects, and `x`, `-x`, `y`, or `-y` for cylindrical constraint objects.
- `constraint points` Constraint points are specified between the keywords `beginpoints` and `endpoints`. You can define any number of points, attached to any body segments. However, the more points you define the more difficult it will be for SIMM to find a set of `gencoord` values which satisfy the constraint. In practice, two or three points is usually sufficient to create the desired constraint on the motion.
- Each point is defined by its name, its X, Y, Z position in the body segment's reference frame, its weight, the name of the body segment it is attached to, and its tolerance value. The `weight` of the point is a non-negative floating point value which controls how much emphasis is placed on it when enforcing the constraints. The higher the weight, the more SIMM tries to position that point onto the object surface, at the expense of the other points. The `tolerance` of a point is the distance (in model units) that it is allowed to be from its corresponding object, and still be an acceptable solution. It does not affect SIMM's calculation of the solution (as does the weight of a point). Rather, it is a threshold value which is checked after the best constraint solution has been calculated. If the point is not within `tolerance` of its constraint object, it is colored red to indicate this condition.

## 3.4 Muscle Files

Muscle files contain descriptions of muscle-tendon actuators. Each musculoskeletal model you load into SIMM must have all of its muscle-tendon actuators defined in one muscle file. If you do not have any muscle-tendon actuators in your model, then you do not need to create a muscle file. When you load a model, SIMM will first check the joint file to see if you specified the name of the muscle file to load with it (using the `muscle_file` keyword). If no name is specified, SIMM will read the muscle file that you selected with the file browser, if any.

Within muscle files you can define two types of objects: *muscles* and *muscle surfaces*. *Muscles* are biomechanical elements that have lines of action and that can generate forces and moments. They are described in Section 3.4.1. *Muscle surfaces* are display primitives designed to make musculoskeletal models look more realistic. They are described in Section 3.4.2.

### 3.4.1 Muscles

Each muscle-tendon actuator is defined by specifying its geometry and force-generating properties. The geometry is defined by a series of attachment points connected by line segments. Each point is fixed to one of the body segments, and its coordinates are expressed in that segment's reference frame. A minimum of two points is required to define the muscle path. In cases where a muscle wraps over bone or is constrained by retinacula, you can introduce intermediate “via” points to further constrain the path. You can also specify a restricted range of a generalized coordinate over which these via points constrain the path. Thus, a via point near the surface of a bone can be

used to constrain the muscle path only when the path would otherwise pass through the bone.

Instead of (or in addition to) via points, muscle wrapping objects may be added to body segments to provide a more automated form of muscle wrapping. For more information about wrap objects see Section 2.9, Wrap Editor, and Section 3.3.9, Wrap Objects.

The isometric force-generating properties of a specific muscle-tendon actuator are derived by scaling a generic, Hill-based model. To scale the generic model, you must supply four parameters and three curves. The four parameters are: peak isometric muscle force ( $F_0^M$ ), optimal muscle-fiber length ( $l_0^M$ ), pennation angle ( $\alpha$ ), and tendon slack length ( $l_S^T$ ). The three curves are the active and passive force-length relations of muscle, and the force-length relation of tendon. The active force-length curve of muscle is scaled linearly by muscle activation, which ranges from 0.0 (no activation) to 1.0 (full activation). All of the curves are defined by listing control points that are interpolated by natural cubic splines.

There are also some circumstances under which you can model the force-velocity characteristics of the muscle-tendon actuators. When calculating muscle force during a motion sequence, you have the option of estimating the velocities of the muscle fibers and using this information to scale the isometric force computations. This process works by first finding the velocities of the generalized coordinates (either taken from the motion file or calculated by differentiating the generalized coordinate position data. See Section 3.5, Motion Files, for details). Then, muscle-tendon velocity is calculated, and this value is distributed to the fibers and tendon based on their relative stiffnesses. The more compliant element is assigned more of the velocity.

To use these force-velocity features, you need to specify two additional parameters in the definition of each muscle. The first is `max_contraction_velocity`, which defines the maximum contraction velocity of the muscle fibers in `optimal_fiber_lengths` per second (a positive number). The second is the normalized force-velocity function for the muscle. This defines the scaling factor for the isometric force, as a function of the normalized fiber velocity (`optimal_fiber_lengths` per second divided by `max_contraction_velocity`). Thus the range of the function's abscissa should be -1.0 to 1.0, and its ordinate should be 1.0 for a velocity of 0.0. When a muscle contracts, it is defined to have a negative velocity.

Note: Case is significant when entering text into muscle files. All SIMM keywords (*e.g.*, `beginmuscle`, `endmuscle`) must be in lower-case letters. You are free, however, to use upper- as well as lower-case letters for naming the muscles, and SIMM will display these names exactly as you type them into the muscle file.

You can define a “default” muscle which contains the parameters that are usually identical for every muscle (*e.g.*, the normalized force-length curves). If a particular muscle definition does not contain a certain parameter or curve, the parameter or curve is inherited from the default muscle. This greatly reduces the amount of information that you have to specify in each muscle definition. A sample default muscle is shown in Figure 3-18.

A sample lower-limb muscle, rectus femoris, is shown in Figure 3-19. The muscle path is defined by three points, one in each of the `pelvis`, `femur`, and `patella` body segments. The point in the `femur` frame is included in the path only when the knee angle is between -120 and -80 degrees. This is done so that the muscle path wraps over the femur when the knee is flexed beyond -80 degrees.

The muscle is a member of two groups, `hip_flexion` and `knee_extension`, which means that the muscle name will appear in those two muscle-group menus in the Model Viewer and Plot Maker. The muscle definition also lists the values for the four muscle-tendon parameters ( $F_0^M$ ,  $\ell_0^M$ ,  $\ell_S^T$ ,  $\alpha$ ). Pennation angle must be specified in degrees. Since the curves that specify the force-length relations of

```

beginmuscle defaultmuscle /* must use this name for default muscle */
begin tendonforce lengthcurve /* force-length curve for tendon */
(0.00000, 0.000) (0.00131, 0.010)
(0.00581, 0.065) (0.00731, 0.091)
(0.00881, 0.123) (0.01030, 0.161)
(0.01230, 0.227) (3.00000, 98.000)
end tendonforce lengthcurve
begin passiveforce lengthcurve /* passive force-length curve for */
(0.00000, 0.000) (1.00000, 0.000) /* muscle fibers */
(1.10000, 0.035) (1.20000, 0.120)
(1.30000, 0.260) (1.40000, 0.550)
(1.50000, 1.170) (1.60000, 2.000)
end passiveforce lengthcurve
begin activeforce lengthcurve /* active force-length */
(0.000000, 0.000000) (0.403500, 0.000000) /* curve for muscle */
(0.527250, 0.226667) (0.718750, 0.856667) /* fibers */
(0.861250, 0.950000) (1.045000, 0.993333)
(1.217500, 0.770000) (1.438750, 0.246667)
(1.618750, 0.000000) (2.200000, 0.000000)
end activeforce lengthcurve
begin forcevelocitycurve /* force-velocity curve */
(-1.00000, 0.000000) (-0.50000, 0.166667) /* for muscle fibers */
(-0.20000, 0.444444) (-0.05000, 0.791000)
(0.000000, 1.000000) (0.050000, 1.482000)
(0.100000, 1.600000) (0.300000, 1.720700)
(0.500000, 1.700000) (1.000000, 1.775000)
end forcevelocitycurve
max_contraction_velocity 10.0
endmuscle

```

Figure 3-18. Example default muscle

```
/* Rectus Femoris muscle */
beginmuscle RectusFemoris
beginpoints
-0.029 -0.031 0.096 segment pelvis
 0.033 -0.403 0.001 segment femur ranges 1 knee_angle (-120, -80)
 0.012 0.043 -0.001 segment patella
endpoints
begingroups
hip_flexion
knee_extension
endgroups
wrapobject distal_femur_ell midpoint range 1 -1
max_force 780.0 /* Newtons */
optimal_fiber_length 0.0840 /* meters */
tendon_slack_length 0.3460 /* meters */
pennation_angle 5.0 /* degrees */
min_thickness 0.004
max_thickness 0.008
min_material muscle_min
max_material muscle_max
endmuscle
```

Figure 3-19. Example muscle definition

the muscle and tendon are not given in this example, they would be inherited from the default muscle.

When defining a muscle, you can optionally specify the activation level in addition to the other [required] muscle-specific parameters. Just put the keyword `activation` in the definition, followed by a number between 0.0 (no activation) and 1.0 (full activation). The thickness and color of the muscles in the model window depend on their activations, with the muscles getting thicker and brighter as activation increases. If you do not specify the activation level when defining a muscle, it will be initialized with a value of 1.0 (regardless of the default muscle). In most cases, this is the value that you will want to start with

because the Plot Maker uses the current activation level when calculating muscle forces (see Section 2.5, Plot Maker, for more details).

You can also optionally specify the minimum and maximum thicknesses and materials of the muscles. At an activation of 0.0, a muscle will be drawn with material `min_material` and a thickness of `min_thickness`. At an activation of 1.0, material `max_material` will be used and its thickness will be `max_thickness`. For activations between 0.0 and 1.0, a linear combination of the min and max values for the material and thickness will be used. If you do not specify values for these parameters, SIMM will calculate reasonable defaults.

If the muscle is to automatically wrap over a wrap object specified in the model's joint file, then an association between the muscle and the wrap object must be made. Include the `wrapobject` keyword in the muscle definition, followed by the name of the wrap object to be wrapped over. See Section 3.3.9, Wrap Objects, for more information about defining wrap objects in your joint file. If the muscle is assigned to wrap over an ellipsoid, then the name of the wrap method to be used may follow the wrap object name. The valid wrap method names are: `hybrid`, `midpoint`, and `axial`. If no wrap method is defined, then the `hybrid` method is used by default. For more information on wrapping methods see Section 2.9.5, Ellipsoid Wrapping Methods.

You may also specify the section of the muscle path that is allowed to wrap over the wrap object. To do this, put the keyword `range` after the wrap object name (or after the ellipsoid wrapping method, if specified), followed by the indices of the starting and ending muscle points. Only muscle path segments that are between these start and end points will be checked for intersection with the wrap

object. A value of -1 for the start point means to start at the first muscle point, and a value of -1 for the end point corresponds to the last point in the muscle. Note that these muscle point numbers refer to the points as they are listed in the definition of the muscle in the muscle file. If there are via points that are currently inactive when muscle wrapping is computed, they are still counted for the purposes of determining the range of muscle path segments to consider for wrapping. If you do not specify a range of points for the wrap object association, then the entire muscle path will be used.

### 3.4.2 Muscle Surfaces

Muscle surfaces are objects designed to make the display of your musculoskeletal models more realistic. They do not contain any biomechanical properties such as lines of action or optimal fiber lengths, but rather are hollow shells that represent the surfaces of the muscles. Like muscles, each section of the muscle surface is attached to a body segment, so the surface will change shape as you move the joints.

A muscle surface is a roughly cylindrical shape that is defined by a group of lines that run down the length of the cylinder. Each line is defined much like the series of attachment points of a normal muscle, between the keywords `beginpoints` and `endpoints` (you can even specify wrapping points). This group of lines is then put between the keywords `beginmusclesurface` and `endmusclesurface` to define the complete surface. The points within each muscle line should be ordered in the same direction (*e.g.*, proximal to distal). The sequence of muscle lines, when viewed from the outside, should be ordered left to right, assuming the points within each line are ordered top to bottom. If the points are ordered bottom



to top, then the sequence of lines should proceed right to left. The last line of points in the surface definition should be the same as the first, so that the surface is a complete cylinder (Figure 3-20).

To make the muscle surfaces change color with activation, you can specify their activation levels in a motion file. As

```

beginmusclesurface rf_surface /* Surface desc. of Rectus Femoris*/
beginpoints
-0.0337 -0.0326 0.0877 segment pelvis /* line 1: defines the */
0.0434 -0.0103 0.0059 segment femur /* anterior contour of */
0.0594 -0.3291 -0.0072 segment femur /* the muscle surface, */
0.0121 0.0439 -0.0090 segment patella /* proximal to distal */
endpoints
beginpoints
-0.0364 -0.0335 0.0930 segment pelvis /* line 2: defines the */
0.0392 -0.0108 0.0139 segment femur /* medial contour of */
0.0523 -0.3298 0.0060 segment femur /* the muscle surface, */
0.0064 0.0433 0.0016 segment patella /* proximal to distal */
endpoints
beginpoints
-0.0364 -0.0335 0.0930 segment pelvis /* line 3: defines the */
0.0392 -0.0108 0.0139 segment femur /* posterior contour of */
0.0523 -0.3298 0.0060 segment femur /* the muscle surface, */
0.0064 0.0433 0.0016 segment patella /* proximal to distal */
endpoints
beginpoints
-0.0309 -0.0320 0.0931 segment pelvis /* line 4: defines the */
0.0477 -0.0104 0.0139 segment femur /* lateral contour of */
0.0664 -0.3291 0.0060 segment femur /* the muscle surface, */
0.0177 0.0439 0.0016 segment patella /* proximal to distal */
endpoints
beginpoints
-0.0337 -0.0326 0.0877 segment pelvis /* line 5: same as line 1,*/
0.0434 -0.0103 0.0059 segment femur /* needed to close the */
0.0594 -0.3291 -0.0072 segment femur /* muscle surface */
0.0121 0.0439 -0.0090 segment patella
endpoints
endmusclesurface

```

Figure 3-20. Example muscle surface

with muscles, use the name of the muscle surface as a column label, and enter activations from 0.0 to 1.0. The materials *muscle\_mat0* through *muscle\_mat20*, specified in the joint file, are used to color the muscle surfaces. For example, *muscle\_mat0* is used when the activation is 0.0, *muscle\_mat10* is used for activations of 0.5, and *muscle\_mat20* is used for activations of 1.0.

You cannot interactively edit the points that define a muscle surface, so as you create them you should make each line an ordinary muscle so you can move the points using the Muscle Editor. Then write the muscles to a file and copy the attachment points to a muscle surface definition.

### 3.4.3 Ligaments

Ligaments are modeled in SIMM as muscles that have fibers of zero length. The muscle-tendon unit is thus only a tendon, with no active force properties. To define a ligament, create a muscle definition and specify an *optimal\_fiber\_length* of 0.0. This tells SIMM to treat the "muscle" as a tendon-only element, using the tendon force-length curve to calculate its force. By specifying a tendon force-length curve that models the mechanical behavior of ligamentous tissue, the "muscle" will thus behave as a ligament. As with muscles, ligaments can span any number of body segments, and can wrap over wrap objects. Because ligaments are created using a modified muscle definition, they will be listed as muscles in the SIMM graphical interface, but you can use muscle groups to differentiate them from normal muscles.

## 3.5 Motion Files

Motion files contain lists of joint angles that describe motions of musculoskeletal models. Once loaded into SIMM, motions can be linked to the models already loaded. The motion variable (e.g., *gait\_cycle*) is treated as a generalized coordinate. It can be used to animate the model, and can serve as the independent variable when making plots. You can also create plots of any column of motion data using the Plot Maker (see section 2.5.5, Y-variables). In addition to specifying joint angles during a motion, you can also specify muscle activation levels. Because the thickness and color of each muscle depends on its activation, you can use a motion file to not only playback the kinematics of an animation, but also to display the relative activity levels of the muscles used to create the motion. An example motion file is shown in Figure 3-21.

In addition to joint angles and muscle activation levels, you can also specify animation data for arbitrary objects to be added to the scene when the motion is played back. These are referred to as *motion objects*. SIMM provides two built-in motion objects, a green arrow and a green sphere. These two motion objects can be used to display force vectors and contact points respectively. However, you can add your own custom motion objects as well, and animate their position, scale, orientation, and color. See Section 3.5.3, Motion Objects, for more information about adding motion objects to an animation.

Columns of data in a motion file can correspond to a generalized coordinate (possibly with a suffix), a muscle (possibly with a suffix), a motion object component, or "other data."

If you always want to load a certain motion file, or files, with a musculoskeletal model, you can put the name of the motion file inside the joint file, preceded by the keyword `motion_file`. You can specify up to 100 motion files in this way. See the beginning of Section 3.3, Joint Files, for more details.

```

/* One-legged walking motion */

name gait_cycle          /* name of this motion */
datacolumns 6           /* 6 columns of data */
datarows 100           /* 100 rows of data, 1 for each motion time step */
range 0.0 100.0        /* 0 to 100% of gait cycle */
keys g_key              /* press this key to make the model walk */
wrap                    /* so you can "walk" the model continuously */
cursor 1.0 1.0 0.0     /* cursor in plot windows will be yellow */
event 40.0 heelstrike   /* heelstrike will be displayed in plot windows */
event_color 1.0 0.0 1.0 /* as a magenta vertical bar */
enforce_loops no        /* enforce loops when animating motion? */
enforce_constraints no  /* enforce constraint objs when animating motion? */
calc_derivatives 0.01  /* seconds between each frame of data, so SIMM
endheader               /* can calculate derivatives of motion curves */

hip_flexion  knee_angle  rf      semimem    knee_mom  hip_flexion_vel
17.3492      -21.7508  0.22  0.10     33.4217   0.0000
13.8255      -21.2797  0.25  0.10     35.8324   -1.8606
12.3931      -20.5520  0.25  0.00     38.0823   -3.6857
 9.8606      -19.2399  0.35  0.00     40.0734   -12.2093
 7.3476      -19.0344  0.35  0.00     42.9230   -11.5651
 6.0183      -19.0093  0.37  0.00     43.1642   -10.8911
 5.6743      -18.2846  0.37  0.00     45.3475   -9.7498
.
.
.

```

Figure 3-21. Example motion file

### 3.5.1 Motion File Header

A header in each motion file names the motion, defines the range of values for the motion (usually percent-of-activity), and lists the keys that are used to animate the model (as in a generalized coordinate definition). You can specify either one or two keys to control the motion, and they must be on the same line as the word `keys` in the motion file.

*events* To make it easier to correlate the animation of your model with plots of the data, you can specify some parameters in the motion file header that will show up in your plots of the motion data. If you put the keyword `cursor` in the motion file, followed by an RGB color value (see the example in Figure 3-21), a vertical bar will move along the plot data as you animate the model according to that motion. To place markers in the plots representing events in the motion (e.g., heelstrike), use the keyword `event`, followed by the time of occurrence (within the `range` that you specified) and the name of the event. The color of events can be specified with the `event_color` keyword and an RGB color value.

*derivatives* If you want SIMM to calculate the velocities of the gencoords, put the keyword `calc_derivatives` in the header of the file, followed by the time (in seconds) between rows of data. SIMM will differentiate each column of data in the file, and use these data to obtain the gencoord velocities that you did not specify explicitly in the file. To differentiate the data, SIMM assumes that the rows of motion data are equally spaced in time, with the spacing equal to the value entered after the `calc_derivatives` keyword. Natural cubic splines are then fit to the data, and derivatives are evaluated at each data point.

*enforce\_loops*  
*enforce\_constraints*

These two parameters control whether or not loops and constraints are enforced when playing back the motion in SIMM (*i.e.*, whether the solver module attempts to adjust the gencoords to satisfy the constraints). If the loops and constraints were enforced when the motion was created, you will usually want to set both of these parameters to *no* so that the solver module does not run unnecessarily (thus slowing down the animation). Motion files created by the Motion Module and the Dynamics Pipeline have these parameters set to *no* for this reason.

### 3.5.2 Gencoords and Muscles

The most common types of data found in motion files are generalized coordinate values and muscle activation levels. You can also specify velocities of one or more generalized coordinates in a motion file. These velocities can be used to estimate fiber velocities when calculating muscle forces during the motion. To specify the velocity of a gencoord, the column name must be the name of the gencoord appended by “\_vel”.

Since the thickness and color of the muscles in the model window depend on their activation levels, specifying muscle activations in a motion file can create more informative animations. As the body segments are animated through the motion sequence, the muscles will change their appearance to reflect their relative levels of activity. Muscle activation levels must be in the range 0.0 to 1.0. To specify a muscle’s activation in a motion file, the column name must be the name of the muscle.

If you want to put other muscle parameters in a motion file, such as their force (*e.g.*, as calculated by a dynamic simulation, to compare to SIMM’s force calculations), the column name should be the name of the muscle appended

by a suffix (e.g., "semimem\_force"). The actual suffix you choose does not matter in SIMM. Columns of muscle data that are labeled in this way are not used to set any muscle parameter values in the model. They only show up under the *motion curves -> muscles* menu when you use the **motion curves** command in the Plot Maker. These columns of muscle data do not count as "otherdata".

### 3.5.3 Motion Objects

SIMM allows you visualize phenomena such as ground reaction forces and joint contact points by animating arbitrary shapes when a motion is played back. By default SIMM provides an arrow for visualizing vectors and a sphere for visualizing points.

Animation data for motion objects are labeled in the motion file by combining the name of the body segment to which the object should be attached with the name of the motion object and the name of the component to be animated. The components that may be animated are:

- position: `_px, _py, _pz`
- scale: `_sx, _sy, _sz`
- vector: `_vx, _vy, _vz`
- color: `_cr, _cg, _cb`

For example, you can display force vectors attached to any body segment (e.g., ground) by specifying their point of application and size in a motion file. For each vector, enter six columns of data in the file, three for the X, Y, and Z coordinates of the point of application, and three for the X, Y, and Z coordinates of the vector itself. The column labels should be the name of the body segment to which the vector is attached, appended by `_force_px`, `_force_py`, and `_force_pz`, for the point of application,

and `_force_vx`, `_force_vy`, and `_force_vz` for the force vector.

The names of the motion objects built into SIMM are `force`, `ball`, `contact`, `joint_force`, `joint_torque`, and `spring_force`. You can override the properties of these six motion objects, or add additional motion objects in the joint file. See Section 3.3.13, Motion Objects, for more information about defining motion objects.

The bone file used to display the built-in force motion object is named `arrow.asc`. This shape is an arrow of unit length with the tail of the arrow at the origin and the head of the arrow pointing up the Y-axis. This produces force vectors that are displayed with their tail positioned at the point of application, and their arrow head pointing away from the point of application. If you prefer to have force vectors displayed with the arrow head pointing toward the point of application, you can use `norm` to translate the shape defined in “`arrow.asc`” from the positive Y axis to the negative Y axis with the following command:  
`norm arrow.asc arrow.asc -ty max.`

Lastly, the built-in force motion object includes XYZ scaling factors designed to produce force vectors of a reasonable size when forces are specified in newtons and your model length units are meters. You can override the default scaling factors by adding a motion object definition to your joint file like this:

```
beginmotionobject force
scale 0.5 0.002 0.5
endmotionobject
```

*Figure 3-22. Overriding built-in motion object parameters. The scale factors shown here are the SIMM defaults for the built-in force motion object.*



### 3.5.4 Other Data

Columns of motion data that do not correspond to generalized coordinates, muscles, or motion objects are called *other data*. These columns must contain `datarows` elements, like all data columns, but they do not need to correspond directly to anything in a model. For example, they can be used for plotting [previously-computed] joint moments or accelerations during gait. You can put columns of *other data* anywhere in the motion file (just make sure that the data lines up with the column names). When SIMM reads the motion file, it will automatically figure out which columns correspond to model components (e.g., generalized coordinates), and which ones are *other data*. It will print the *other data* column names to the message window so you can see which ones SIMM did not associate with a model component.

For any of the columns of data in a motion file, except muscle activations and motion object components, you can also specify standard deviations of the values for each time frame. This is useful if you averaged recorded data from several motions and want to display the standard deviations along with the averages. To specify them for a column of data, add a second column of data, containing the non-negative values for one standard deviation, and label the column with the original data label appended by `_std`. When you plot the data using the **motion curves** command in the Plot Maker, the standard deviations will be displayed underneath the data, in the same color.

When you load a motion file into SIMM, the new motion is linked only to the current model, so you should make the model window the topmost one in SIMM before loading a motion file to link to it.

Note: Case is significant when entering text into motion files. All SIMM keywords (*e.g.*, `datacolumns`, `datarows`) must be in lower-case letters. You are free, however, to use upper- as well as lower-case letters for labeling the columns of data, and SIMM will display these names exactly as you type them into the motion file. Spaces are not allowed within the names of the data columns.

## 3.6 Plot Files

Plot files contain sets of coordinates describing plot curves. They can be loaded into SIMM and added to plots in order to compare with other curves. They are designed to allow you to compare data derived from other sources (*e.g.*, experimental data, dynamic simulations) with the data that SIMM computes from your model. SIMM can also write out plot files containing curves that were computed from your musculoskeletal model. The format used for input and output is the same, so you can create a plot with the Plot Maker, save it to a file, and load it back into SIMM using the file browser (see Sections 2.3 and 2.4 for more details).

An example plot file is shown in Figure 3-23. Comments are indicated by a hash mark (#). All text on a line that follows a # is ignored by SIMM when reading the file. You can specify a title for the plot using the `title` keyword. All text on the line following `title` is interpreted as the title of the plot. You can specify plot curves individually, or in a table format (much like the columns of data in a motion file). In the example in Figure 3-23, three curves are specified, two in the first section, and one in the next. After the title, the X and Y variables and curve name are specified for the first set of curves. The name of the X variable is

```

# Ankle plantarflexion moment
# Sale, D.J., et al., Influence of joint position on
# ankle plantarflexion in humans, J. Appl. Physiol.:
# Resp. Environ, Exercise Physiol. 52(6): 1636-1642, 1982
# pos angles = dorsiflexion
# neg angles = plantarflexion

title   ankle_angle moment vs. ankle_angle

ankle_angle      Sale (ave 90) | ankle_angle moment
                  Sale (knee 90) | ankle_angle moment

-30.0  -35.0  -38.0
-25.0  -50.0  -40.0
-15.0  -79.0  -63.0
-5.0   -114.0 -87.0
 5.0   -144.0 -125.0
15.0   -170.0 -150.0
20.0   -168.0 -151.0

ankle_angle      Sale (knee ext) | ankle_angle moment
-30.0  -52.0
-25.0  -75.0
-20.0  -87.0
-15.0  -105.0
-10.0  -125.0
-5.0   -140.0
 0.0   -155.0
 5.0   -165.0
10.0   -170.0

```

Figure 3-23. Example plot file

ankle\_angle. If you plan to load this curve into an existing plot to compare to other curves, it is important to give the X and Y variables names that exactly match the names used in the other curves. Next comes the text describing each plot curve. Each curve's text should be preceded by a tab. The format for this text is: curve name followed by a vertical bar (|) followed by the name of the

Y variable. The name of the first curve is `Sale (ave 90)` and its Y variable is `ankle_angle_moment`. The name of the second curve is `Sale (knee 90)` and its Y variable is `ankle_angle_moment`. After specifying all of the curve names and Y variables (for the first set of curves), you then enter the columns of plot data. The first column contains the values of the X variable, and each of the rest of the columns contains one plot curve of data.

The second set of plot curves in the example file contains just one curve. This curve has the same X and Y variables as the other two curves, but it is defined separately because it uses a different range of the X variable `ankle_angle`.

---

# 4 Norm

---

*Norm* is a utility program that you can use to preprocess your bone and world object files (collectively called “object files”) before you can load them into SIMM. The main purpose of *norm* is to compute surface normals for the polyhedra in the files, but it can also perform other functions, such as removing duplicate vertices, making the polygons convex, and translating or rotating the polyhedra. These features can be useful if you are creating new object files from medical image data or 3D digitization and you want to load them into SIMM.

Most of the functionality of *norm* is accessible via the Bone Editor tool in SIMM. This graphical interface lets you view desired rotations, translations, and scales to a bone before they are applied with *norm*. The Bone Editor also has simple polygon and vertex editing functions, which *norm* does not. However, the stand-alone version of *norm* contains some file conversion and advanced normal calculation functions that are not accessible in the Bone Editor. Before you begin modifying your bone files, it is recommended that you learn about the Bone Editor and see if it can perform all the functions you will need. If not, then use the stand-alone version of *norm* described below.

*Norm* is able to process object files which contain more than one object (*e.g.*, a single input file containing all of the ribs of an upper-body model). After reading in the file, *norm* will separate out the individual polyhedra, compute surface normals for each of them, then write them all to

---

the specified output file. The polyhedra in an object file do not have to be closed surfaces (they can have holes in them). However, *norm* may have difficulty computing surface normals for such polyhedra. If *norm* reports that it was unable to compute normals for one or more polyhedra, it will still create an output file, but some of the surface normals will be incorrect. These polyhedra will not be shaded properly in SIMM, but you can still use them as object files in your model. Also, if there are concave polygons in the object file, *norm* may have difficulty computing surface normals for the object. However, *norm* has an option (described below) to subdivide polygons as it processes the file.

To process an object file, open a DOS window, change directories to where *norm* is installed (it is in the same directory as the SIMM executable) and type `norm` followed by one or more file names and the desired options, if any. *Norm* can read any of the three file formats described in Section 4.2, Bone Files, and by default writes out bones in the newer ASCII format. File names may appear anywhere on the command line. In other words, you can intersperse options and file names in any order. *Norm* identifies file names by their lack of an initial dash (-). Here are some examples of how you can specify file names, and the rules that *norm* follows when processing them:

```
norm file
```

*Norm* will process `file`, and save the output object in `file`. It will prompt you to confirm that the contents of `file` will be overwritten.

```
norm infile outfile
```

*Norm* will process `infile` and save the output object in `outfile`.

---

```
norm infile1 outfile1 infile2 outfile2 ...
```

*Norm* will process *infile1*, *infile2*, ... and save the output objects in *outfile1*, *outfile2*, ... respectively.

```
norm -files *.asc
```

All files in the current directory ending with *.asc* will be passed to *norm* as input files. For each file, *norm* will process it and then prompt you to make sure you want to overwrite the file with *norm*'s output. Use of the *-files* option may be combined with the use of input/output file name pairs on the command line.

*Norm* has several options that allow you to perform simple geometric transformations on your object files. They can be helpful when creating a musculoskeletal model from raw bone-surface data. Below is a complete list of *norm*'s options:

- overwrite* automatically overwrite existing files without prompting.
- outdir dirname* put all output files in the specified directory. If the directory doesn't already exist, *norm* will create it.
- files filter* process all files in the current directory that match the specified filter. These files will be overwritten with the output.
- sep* write out each polyhedron that is in the input file to a separate output file. The name of each output file will be based on the single output file name you specify (for each input file).
- tol num* this option sets the tolerance that *norm* uses when checking for duplicate vertices. All vertices that are within *num* units of each other will be merged into one vertex. A value of 0.0 (which is the default) stops *norm* from checking for duplicate vertices.

- 
- fill fill in holes in the polyhedra. As *norm* processes a polyhedron, it will find holes in it and attempt to fill them in with triangles. If the polyhedron contains large holes, *norm* may have difficulty filling them.
  - es *num* this option lets you specify the maximum allowable length for edges in the polyhedra. If *norm* finds any edges with a length greater than *num*, it will subdivide them until they are shorter than this length. *Norm* does not create additional polygons during this process; it just inserts vertices into existing edges.
  - con make all of the polygons in the file convex. *Norm* will split off pieces of concave polygons (making new triangles) until all of the polygons are convex.
  - tri triangulate all of the polygons in the file. *Norm* will divide all polygons with four or more edges into triangles.
  - ccw tells *norm* that all polygons are ordered in a counter-clockwise direction (i.e., the vertices within each polygon are specified counter-clockwise). If you use this option, *norm* will not reorder any polygons, but will calculate vertex normals assuming that all of the polygons are counter-clockwise.
  - cw tells *norm* that all polygons are ordered in a clockwise direction (i.e., the vertices within each polygon are specified clockwise). If you use this option, *norm* will reorder every polygon so that they are counter-clockwise, and will then calculate vertex normals.
  - rn *opt* this option is used to specify a reference normal which *norm* will use as a last resort when calculating vertex normals. *Norm* will attempt to determine the inside and outside of a polyhedron without using a reference normal. If the polyhedron is not a closed object, however, it may



---

have difficulty finding the outside. If *norm* does have difficulty, and you do not know if the polygons are all clockwise or all counter-clockwise, you should specify a reference normal. This vector corresponds to the major direction of the surface normal of the topmost polygon of the polyhedron. For *opt*, use one of: X, -X, Y, -Y, Z, -Z. The default value of the reference normal is Y.

- oldascii write the polyhedron to an old-style ASCII file. This style of ASCII was the required input format for early versions of *norm*, but has since been replaced with the new ASCII format. Use this option if you want to output a polyhedron without bounding cube information or vertex normals.
  
- binary write the polyhedron to a binary file. This option is provided so that you can create polyhedron files in the format which early versions of SIMM required, but it is recommended that you create ASCII files, and not use this option.
  
- tx *num* add *num* units to all vertex X coordinates.
- ty *num* add *num* units to all vertex Y coordinates.
- tz *num* add *num* units to all vertex Z coordinates.
- rx *num* rotate all vertex coordinates *num* degrees around the X axis.
- ry *num* rotate all vertex coordinates *num* degrees around the Y axis.
- rz *num* rotate all vertex coordinates *num* degrees around the Z axis.
- sx *num* scale all vertex X coordinates by a factor of *num*.
- sy *num* scale all vertex Y coordinates by a factor of *num*.

---

`-sz num` scale all vertex *Z* coordinates by a factor of *num*.

For the `tx`, `ty`, and `tz` options, you can specify certain keywords instead of entering an actual number. These are explained below. Shown are the options only for `tx`, but the options are the same for `ty` and `tz`:

`-tx min` translate the origin of the object along the *X* axis so that the smallest *X* coordinate is 0.0 (i.e., find the smallest *X* coordinate among all the vertices and subtract it from all the *X* coordinates).

`-tx max` translate the origin of the object along the *X* axis so that the largest *X* coordinate is 0.0 (i.e., find the largest *X* coordinate among all the vertices and subtract it from all the *X* coordinates).

`-tx mid` translate the origin of the object along the *X* axis to the midpoint between the smallest and largest *X* coordinates (i.e., average the smallest and largest *X* coordinates and subtract this value from all *X* coordinates).

The geometric operations are all implemented as space-fixed transformations, and are performed in the order in which they are entered on the command line (from left to right).

---

# Motion Module

---

## 5.1 Introduction

The Motion Module is an optional component to SIMM that allows you to easily import data recorded by a motion capture system. It reads files containing tracked marker data (3D positions of markers in global space) using the TRC or TRB file format developed by Motion Analysis Corporation. It can also read analog files in the ANB or ANC format with ground-reaction force and EMG data that was recorded in sync with the motion. The Motion Module can also read C3D files, which contain both tracked marker and analog data in the same file. Additionally, the real-time version of the Motion Module can connect to a Motion Analysis system and receive and display motion and analog data in real-time, as it is being recorded.

Files of tracked marker data contain a sequence of frames, each representing a snapshot of the subject's motion at a particular instant in time. Each frame contains the X, Y, and Z coordinates, expressed in a global coordinate system, of all the identified markers. A frame of marker data can thus be thought of as a "marker cloud" because the coordinates are not organized by body segment.

The Motion Module imports tracked marker data and fits a SIMM model within the marker cloud for each time frame. If the SIMM model contains markers whose names and positions match those of the markers placed on the

subject, the Motion Module can adjust the model's genco-ord values to determine a "best fit" of the model to the marker cloud. The quality of a fit is determined by how closely each of the model's markers is to its corresponding marker in the marker cloud. It then uses this best fit as the starting position for solving the next frame of data. The result is a SIMM motion that matches the tracked marker data. The model that is used to fit the data can either be one that you create or the pre-made model (the *mocap model*) that comes with the Motion Module.

The Motion Module has two primary components. The first component reads files containing tracked marker data (in the TRC, TRB, or C3D format) and creates SIMM motions from them, as described above. For more information on how this process works, and the various options for importing marker data, see Section 5.2, Opening Tracked Marker Files.

The second component of the Motion Module creates a musculoskeletal model of a given individual by scaling a generic full-body model (the *mocap model*) based on tracked marker data from a static pose. The algorithms that are used to scale the model are the same as those used in OrthoTrak, a full-body gait analysis package available from Motion Analysis. For more information on the mocap model and how it is created and used, see Section 5.3, Using the Mocap Model.

## 5.2 Opening Tracked Marker Files

SIMM can import tracked marker data that is stored in either a TRB or TRC data file. These file formats, described in the EVa and EVaRT manuals, contain X, Y, and Z coordinates for each identified marker for each time

frame. You can also import analog data files containing forceplate and EMG data recorded during the motion. These analog data files can be in either the ANB or ANC formats. The Motion Module can also read XLS files containing other motion-related data that you may want to view in SIMM, such the kinetic data contained in an OrthoTrak single trial spreadsheet. For more information on importing analog and XLS files, see Section 5.2.3, *Analog Data*.

The Motion Module can also read C3D data files. These files contain tracked marker and analog data in the same file, so you only need to load one file to import all of your motion data from a trial.

When you open a tracked marker file (along with any associated analog files), SIMM attempts to map the data onto the current musculoskeletal model, thus creating a SIMM motion that is linked to the model. Therefore, to open a tracked marker file, you must already have loaded into SIMM a model that contains the same marker set used in the marker file. For best results, you should make sure that every marker in the tracked marker file is also in the SIMM model, and that their locations in the SIMM model match where they were placed on the subject. The marker names should match exactly (except that they are case-insensitive). If the file contains markers that are not in the model, their data will be ignored by the Motion Module. Similarly, if the model contains markers that are not in the file, they will not be used to help fit the model to the motion data.

If you need to add, rename, or move markers in your SIMM model before loading a tracked marker file, you can use the Marker Editor to do so. See Section 2.12, *Marker Editor*, for more details.

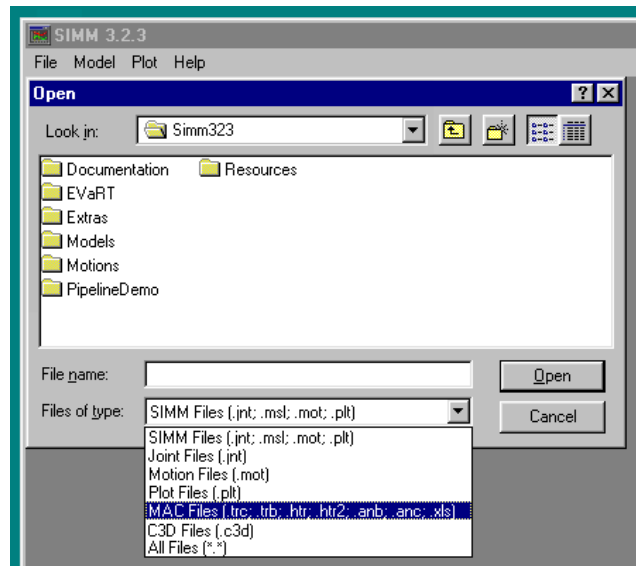


Figure 5-1. Windows File Browser

### 5.2.1 Selecting Tracked Marker Files

To import a tracked marker file into SIMM, first make sure that the model you want to apply it to is the current model (the topmost window in SIMM). Then select *File > Open* from the menu bar. When the Windows® file browser appears, change the *Files of type* popup menu at the bottom of the browser to *MAC Files* or to *C3D Files* then navigate to the folder containing the tracked marker file[s] you want to import.

Next, select the appropriate marker file[s] in the file browser. Click the *Open* button to import the file[s]. At this point SIMM will display a dialog box allowing you to specify several options for importing each data file into SIMM.

Note: If your analog data files have the same base name as your TRB/TRC file (e.g., *subject14.trc*, *subject14.anb*, and *subject14.xls*), then it is not necessary to select analog files in the file browser. SIMM will automatically open any analog or XLS files with the same base name and in the same folder as the tracked marker file (there is an option in the dialog box to turn off this feature). If you are loading C3D files, this is not an issue since all of the data for the motion are stored in the C3D file.

## 5.2.2 Tracked Marker Options Dialog

Once you have selected one or more tracked marker files using the process described in the previous section, SIMM displays a dialog box for each one (in sequence), allowing you to set some options for importing the marker data. In many cases, you will want to use the default settings for these options, so you can simply click the *OK* button to import the motion. The following list describes each option in the dialog box.

*import frames, to, increment*

These fields allow you to specify the range of frames to read from the marker file, as well as the increment. To use them, type into the first two fields the starting and ending frame numbers that you want to import. The third field specifies the increment to use when reading frames from the file. For example, to read every other frame from the file, enter an increment of 2. The starting frame number field and the increment field are initialized to 1. The ending frame number is initialized to the number of frames in the marker file.

*quick solve*

The Motion Module contains two optimization algorithms for fitting the musculoskeletal model to the marker data. The default method is fairly robust- it is designed to handle cases in which several markers are missing from a

frame or in which the markers move large amounts between frames. The other method, called *quick solve*, is less robust but works up to twice as fast as the default method. If speed is an issue, and you know that your marker data is well-behaved, you may want to turn this option on to use the faster optimization algorithm.

*crop ends*

Tracked marker data files often have frames at the beginning and end of a motion that are missing some markers (because the subject is outside the camera volume). To automatically detect and ignore these frames as the file is read, turn on this option (it is on by default). When the option is on, SIMM will start at the first frame and delete it if it is missing one or more markers. It will then continue to scan forward through the frames, deleting each one, until it encounters a frame containing all of the markers. It will then do the same procedure starting at the last frame and working backwards. SIMM will not remove

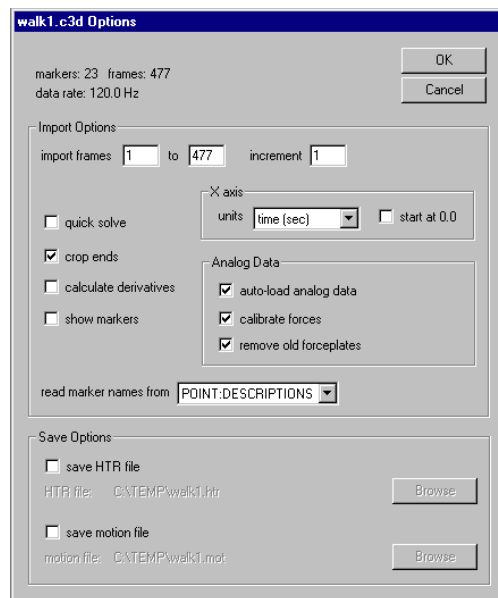


Figure 5-2. Tracked Marker Import Dialog Box



frames with missing markers that are in between full frames, so there may still be frames in the motion that are missing markers.

*calculate derivatives*

When loading a motion, SIMM has the capability of calculating derivatives of the motion variables. When this option is turned on, after SIMM has solved the marker data and created a SIMM motion, it will calculate first-order time derivatives of the generalized coordinate values (*i.e.*, joint velocities) during the motion. It will also calculate derivatives of any force or EMG data in the analog file (if present). These derivatives can then be plotted using the **motion curves** command in the Plot Maker (see Section 2.5.2 for more details).

*show markers*

This checkbox turns on the display of the global marker positions in each frame when playing back a motion. When it is on, SIMM will add spherical motion objects to the motion, representing the location of each marker, as recorded in the marker file. When you animate the model according to the motion, the blue spheres represent these actual, recorded marker locations. These are the marker locations that the Motion Module is trying to fit the model to, for each frame. It can be helpful to display them in the model window in order to visualize how good the fit is, and to help debug problems with the data.

*X axis units, start at zero*

These options give you control over the specification of the X axis of the motion that is created from the marker data. The units along the X axis can be either time (in seconds), or frame number. The starting X value of the motion will be 0.0 if the units are time, and 1 if the units are frame number, unless frames of data are cropped because of missing markers (see the *crop ends* option above). For example, if 12 frames of data are cropped from the beginning of the motion (and the data frequency is 60 Hz), the starting X value will be 0.2 seconds for units

of time, and 13 for units of frame number. If you want the X values to start at 0.0 (or 1 for frame number) even if frames are cropped, turn on the *start at zero* option.

*auto-load analog data*

When this box is checked, SIMM will look for and automatically load any analog (ANB, ANC) or XLS data files with the same base name as the TRB/TRC file. If SIMM did not detect the presence of any analog files when the TRB/TRC file was selected, this option is grayed out.

If you selected a C3D file with the file browser, then this box controls whether or not the analog data will be read from the C3D file.

*calibrate forces*

If an analog file is present, and the *auto-load analog data* box is checked (see above), then this box is active and gives you control over the calibration of the forceplate data. When this box is checked, SIMM determines the baseline of each forceplate channel and automatically subtracts these baseline values from the data, thus “zero-ing out” the force data.

*remove old forceplates*

In order to display forceplate data that is in the analog file, SIMM creates graphical objects in the model window representing the forceplates. Each time you load a tracked marker file with corresponding analog data, SIMM creates a new graphical object for each forceplate in the file. In most cases you will want to remove the existing forceplate objects from the model when loading a new file, so that the display is not cluttered with multiple (or redundant) sets of objects. Thus this option is turned on by default. If you load a series of marker and analog files that all have the same forceplate definitions, then you should leave this option turned on.

*read marker names from*

*For C3D import only:* This option allows you to choose from which parameter field in the C3D file to read the

names of the tracked markers. Because the `POINT:LABELS` field in a C3D file is limited to four characters, some software packages (e.g., EVaRT) store the full marker name in the `POINT:DESCRIPTIONS` field. Since the marker names in the tracked file must exactly match the names used in the *mocap model*, if your C3D file does not contain full marker names in the `POINT:DESCRIPTIONS` field, you may have to edit the *mocap model* so that the marker names match the four-character names stored in the `POINT:LABELS` field.

*save HTR file*

This option allows you to save an HTR file containing the motion that SIMM calculates from the marker data. This HTR file cannot be read back into SIMM, but is useful if you want to import the motion into another software package. If this box is checked, a *browse* button is enabled that allows you to specify the name and location of the HTR file.

*save motion file*

This option allows you to save a SIMM motion file containing the motion that SIMM calculates from the marker data. This file contains exactly the same data that is in the motion that SIMM loads onto the model. You can load this motion file into SIMM at a later time, rather than re-importing the marker file. If this box is checked, a *browse* button is enabled that allows you to specify the name and location of the motion file.

### 5.2.3 Analog Data

Analog data files contain forceplate and EMG data that was collected in sync with motion data. When loading C3D files, there are no separate analog files; all of the analog data is contained in the C3D file. When loading TRB/TRC files, you can load analog files only if they correspond to the chosen TRB/TRC files. If the analog file

has the same base name as the TRB/TRC file, then the Motion Module will load it automatically when you select the marker file. Otherwise, you should select the analog file as well in the file browser. The same holds for XLS files, which are not actually analog files, but are treated similarly. XLS files can contain other data corresponding to the recorded motion, such as kinetic data calculated by OrthoTrak and stored in a “single trial spreadsheet.”

SIMM can recognize three types of analog data: ground reaction forces, EMG activation levels, and “other” data (usually kinetic data from an XLS file). These data types, and how they are interpreted by SIMM, are described below:

*forceplate data*

SIMM displays forceplate data by drawing a vector in the model window at the appropriate point of application and with a size corresponding to the magnitude of the force. Forceplate data in an analog file are voltages measured by forceplate transducers. These voltages are converted into forces using a calibration file, *forcepla.cal*. This file is the same one used by EVa, EVaRT, and OrthoTrak. To use it with SIMM, you should put a copy of it in the same folder as your motion data, or in the folder *SIMM\Resources\mocap\misc*. If you have only one forceplate configuration for your motion capture system, it is preferable to put *forcepla.cal* in *SIMM\Resources\mocap\misc*, rather than copying it into every folder of motion data.

Note: C3D files that contain force plate data also contain the calibration information for the plates. Thus there is no separate calibration file that SIMM reads when importing C3D files.

SIMM also uses another configuration file, *importVariables.txt*, to map forceplate channels to SIMM variables. This file is located in *SIMM\Resources\mocap\misc*, and

contains mappings for typical channel names for up to six forceplates. You will only need to change this file if you use more than six forceplates, or use forceplates that have exotic channel configurations. This file is used when loading ANB/ANC files and when loading C3D files. See Section 5.4.1, *importVariables.txt*, for information on how to modify this file.

*EMG data* SIMM displays EMG data by varying the sizes and colors of the corresponding muscles in the SIMM model. EMG data in an analog file are voltages measured by the EMG system. SIMM rectifies and smooths these data, and then scales them based on an MVC value (maximum voluntary contraction), resulting in a smooth muscle excitation level that varies between 0.0 and 1.0. If MVC values are located in the configuration file *importVariables.txt*, SIMM will use them to scale the EMG data. If MVC values are not present, SIMM will use each muscle's maximum voltage in the analog file to scale that muscle's EMG data (thus each muscle's excitation will peak at 1.0 sometime during the motion). The file *importVariables.txt*, located in *SIMM\Resources\mocap\misc*, contains mappings between typical EMG channel names and the muscle names in the mocap model. It does not contain any MVC values. See Section 5.4.1, *importVariables.txt*, for information on how to add MVC values to this file. In most cases, however, it is sufficient to not specify them and use SIMM's default scaling method.

*other data* "Other" data is contained in XLS files, and can represent any motion variable that you choose to calculate and store in the file. It is usually reserved for kinetic data (*e.g.*, joint moments and powers) that OrthoTrak calculates and stores in its spreadsheet (XLS) format. It may also include motion events, such as toe-off and heelstrike, that are stored at the top of the XLS file. SIMM does not perform

any calculations on these data, but does import them so that you can create plots of them in the Plot Maker. SIMM will only import “other” data that are identified as such in *importVariables.txt*. This configuration file, located in *SIMM\Resources\mocap\misc*, contains mappings between OrthoTrak and the mocap model of all forces, moments, and powers for the hip, knee, and ankle joints. You will only need to edit this file if you want to import data other than these.

### 5.2.4 Real-time Import

In addition to importing tracked marker files, SIMM can import motion data that is sent over the network in real-time from EVaRT. SIMM is thus able to animate a musculoskeletal model and plots of joint angles and muscle lengths while the subject’s motion is being recorded. For this real-time connection, EVaRT solves tracked marker data using the mocap model. It then sends generalized coordinate values (as well as analog data) over the network to the SIMM computer. If the same mocap model is loaded into SIMM, these generalized coordinates will drive the animation of the model in real-time, with a small delay (whose length depends on the network speed and the graphics speed of the SIMM computer).

Follow these steps to use the real-time connection between EVaRT and SIMM:

#### First-time setup only:



Find the folder *SIMM\EVaRT* on your SIMM computer and look for *mocap.jnt* and *solver.dll* (*solver.dll* may be hidden in the folder view because it is a system file). Copy both files to the folder on the EVaRT machine where EVaRT resides (you’ll need to exit EVaRT first, if it is running). This will guarantee that EVaRT is using the same

mocap model and the same scaling algorithms as SIMM uses.

- ☞ Open the text file *SIMM\Resources\preferences* in a text editor such as Notepad or Wordpad. Locate the line that reads: `EVART_MACHINE <hostname>`, and change the hostname to the name of your EVaRT machine. Save and close the preferences file (make sure that the file is saved as ASCII text with the name *preferences* - Wordpad likes to surreptitiously append a *.txt* extension when it saves files that don't already have a filename extension).
- ☞ If your motion capture system includes forceplates, copy the file *forcepla.cal* from your EVaRT computer onto your SIMM computer and put it in the folder *SIMM\Resources\mocap\misc*.

### **Each motion capture session:**

- ☞ Copy the folder containing the motion data from the EVaRT computer to the SIMM computer (or make it shared). If there is a *personal.dat* file for this data, make sure it is in the folder too.
- ☞ Launch EVaRT. Load the appropriate project.
- ☞ Select *File... Load Tracks File* and select the tracked marker file corresponding to the static trial for the subject.
- ☞ Under *Setup - Misc*, click on the radio button for *SIMM OrthoTrak Solver*, located in the *Skeleton Options* area.
- ☞ Launch SIMM.
- ☞ Select *File... Open Mocap Model* and navigate to the motion folder. Choose the tracked marker file containing the static pose.
- ☞ Set the options as desired in the dialog box and click OK.

- ☞ Open the Model Viewer window.
- ☞ In the Model Viewer window, choose *start > realtime connection to <hostname>*. SIMM will display a dialog box allowing you to set some options for the connection. The *motion buffer size* options control how many seconds of motion data are saved in SIMM's buffer. The *time scale* options let you specify the minimum and maximum values, in seconds, for the time scale of the motion. If you want the scale to remain fixed between *minimum* and *maximum*, check the *sliding* checkbox, otherwise the scale will continue to increase as new data is received.
- ☞ SIMM will now wait to receive data from the EVaRT computer. Once the connection is established, SIMM will display “connected” in its message window, and the SIMM model will begin tracking the motion of the subject in real-time. You can pan, zoom, rotate, and change the draw mode of the SIMM model as it is tracking the motion. You can also create plots of kinematic variables and muscle properties and see the plots change in real-time.
- ☞ To disconnect SIMM from the real-time stream, click the *stop* button in the Model Viewer. You can play back the last N seconds of the motion.

Note: When analog data is imported into SIMM in real-time, it is processed slightly differently than when the data is post-processed in SIMM. This is because the real-time analog data is processed frame-by-frame, without the benefit of the full data set. This has the following implications:

- \* To set the baseline for the forceplates, the first frame of force data is used as the “zero” level. Thus when



you first connect SIMM to EVaRT, you should make sure that nothing is on the forceplates.

- \* If no MVC levels have been specified for some of the muscles, a running tally of each muscle's maximum level will be kept, and used to scale the EMG signals into the range 0.0 to 1.0. Thus if you want to accurately scale EMG levels throughout a real-time SIMM motion import, you should either specify MVC levels, or have the subject perform MVCs just after connecting SIMM to EVaRT.

## 5.3 Using the Mocap Model

As described in Section 5.2, SIMM has the ability to read tracked marker data and convert it into a motion by fitting a musculoskeletal model to it. For this to work well, the body segment lengths, marker names, and marker locations in the model must exactly match those for the subject whose motion is being recorded. Because it is time consuming to measure and scale the body segments, and measure and record the offsets of all of the markers, the Motion Module has the ability to automatically scale a pre-made model (the *mocap model*) to fit the subject.

To use the mocap model, select *Open Mocap Model* from the *File* menu. SIMM will display a Windows<sup>®</sup> file browser and ask you to select the name of a static pose file. This static pose (described in more detail in Section 5.3.2) is used to calculate joint center locations and segment lengths for the subject, using the same algorithms implemented in OrthoTrak. In other words, *the Motion Module recreates the OrthoTrak skeletal model from the static pose, and then maps this skeletal model onto the mocap model*. Thus to use the mocap model, you need to

use the same motion capture protocol as you would for OrthoTrak. You can use either the Helen Hayes or Cleveland Clinic marker sets (plus your own additional markers, if desired), as long as the marker names and locations match the protocol defined in the OrthoTrak manual. The Motion Module uses the tracked marker data from the OrthoTrak static pose, and also segment information from *personal.dat*, to scale the mocap model to the subject. The algorithms for calculating joint center locations and segment lengths have been designed to be as similar as possible to the OrthoTrak algorithms. This was done so that motion information in SIMM (*e.g.*, joint angles, EMG levels) would match the corresponding information in OrthoTrak, and also so that you would not have to change your OrthoTrak protocol in order to use SIMM. For more information on the markers built into the mocap model, and how to add your own, read the *Guide to Mocap Model Markers* document.

The mocap model and the algorithms used to scale it are described in the following sections.

### 5.3.1 The Mocap Model

SIMM comes with several musculoskeletal models that can serve as the *mocap model* in the Motion Module. The default model is a full-body model that has been customized for gait analysis, but can be used to import and display any type of full-body motion. The model has 41 body segments, 41 joints, 40 degrees of freedom, and 88 lower-extremity muscles. It represents an adult male, approximately 175 cm. tall, with a mass of 78 kg. The model is scaled to match the size of the motion capture subject using algorithms described in Section 5.3.4. The model's joints have been carefully constructed to represent normal

joint motion as closely as possible. The joint file for this model is *SIMM\Resources\mocap\mocap.jnt*.

There is also a *mocap model* that has been customized for use with the FIT Module and dynamic simulations. It is a simplified full-body model that includes the mass and inertia properties for all of the body segments. The joint file for this model is *SIMM\Models\dynamic-Model\dynamic.jnt*. There are also models of the right arm and left arm, for use in upper-extremity motion capture applications where full-body capture is not appropriate. These joint files are *rightArm.jnt* and *leftArm.jnt*, in the folder *SIMM\Resources\mocap*.

SIMM keeps track of which *mocap model* to use by storing a pointer to the appropriate joint file. When SIMM is first launched, it initializes this pointer by reading the `MOCAP_MODEL` variable in the preferences file (*SIMM\Resources\preferences*). The default setting for this variable is *SIMM\Resources\mocap\mocap.jnt*. If you want to change the *mocap model* while SIMM is running, click on *Options* in the menu bar and select *Choose Mocap Model*. You can then browse for the joint file of the desired model.

You may change any of the *mocap models* however you wish. For example, you can add or remove muscles from the model, or change the tendon and fiber parameters of existing muscles. You can also add degrees of freedom to the model, in order to more accurately represent a particular motion (*e.g.*, adding toe joints and gencoords to examine toe motion in greater detail). If you modify the *mocap model*, however, you should keep in mind two things.

First, the model has been set up to correspond to the skeletal model that OrthoTrak uses when processing gait data. The lower-extremity body segments and orientations of

the reference frames closely match those in the OrthoTrak model. Also, each body segment in the mocap model is scaled to fit the subject by relating its length to the length of an OrthoTrak segment. These relations are specified in the mocap model by defining scale segments and scale factors for each body segment. If you add, delete, or modify joints or body segments in the mocap model, you should make sure that each segment still properly relates to an OrthoTrak segment.

Second, *mocap.jnt* contains several macros that are used to properly define the orientation of the floor, and to automatically remove the upper body segments if there are no upper body markers. When SIMM reads a joint file, it performs these macros but does not save them internally. Thus when it writes out a joint file, all of the macros have been removed. If you make changes to the mocap model in SIMM and then save the new model to a file, do not replace *mocap.jnt* with the new file. Instead, copy the relevant portions of the new file into *mocap.jnt* using a text editor, thus preserving the macros and comments.

### 5.3.2 The Static Pose

When you open the mocap model, SIMM prompts you for the name of a tracked marker file containing a static pose of the subject. This static pose is the same one used by OrthoTrak, and for it you can use any of the six marker sets identified by that software package: *Cleveland Clinic Lower Body*, *Cleveland Clinic Full Body*, *Cleveland Clinic Full Body with Head*, *Helen Hayes Lower Body*, *Helen Hayes Full Body*, and *Helen Hayes Full Body with Head*. It is also strongly recommended that you include the medial knee and ankle markers in the static pose, for more accurate calculation of knee and ankle joint centers. You can also supplement the OrthoTrak marker set with your

own custom markers, as long as you do not move or remove any markers from the identified set. Lastly, the marker set used in the static trial must include all of the markers you plan to use for capturing motion. This is because the Motion Module calculates the locations of all markers in the mocap model based on their locations in the static trial. These are the steps you should follow when collecting the static trial:

- ☞ Choose which of the six OrthoTrak marker sets you would like to use for capturing motion.
- ☞ Add the medial knee and ankle markers, for better calculation of knee and ankle centers (not required, but highly recommended).
- ☞ Add any additional markers that you would like to use (e.g., extra markers on the feet, more markers on the arms). These markers must also be added to the mocap model, as described in Section 5.3.5, The Marker Set.
- ☞ Capture the static trial using the protocol outlined in the OrthoTrak manual. The subject should have their arms either down by their sides, or straight out from their body with their thumbs facing forward.
- ☞ Remove the medial knee and ankle markers, and any others that you do not want to use for capturing motion.

Note: If you use a marker set with no upper extremity markers, the Motion Module will remove the upper extremity from the mocap model and display only the pelvis and legs.

Once you have selected the static pose file to be used for opening the mocap model, SIMM displays a dialog box, allowing you to set some options for importing the static pose. In many cases, you will want to use the default set-

tings for these options, so you can simply click the *OK* button to import the motion. The following list describes each option in the dialog box:

*average frame from, to*

These fields allow you to specify the starting and ending numbers for the sequence of frames that are averaged together to determine the static pose. These fields are initialized to 1 and the number of frames in the file, meaning that all frames will be averaged. If frames in the chosen sequence are missing some markers, locations for markers that are present will still be used in the average.

*load personal.dat*

This option gives you control over the automatic loading of *personal.dat*. When SIMM loads the static marker file, it looks for a file called *personal.dat* in the same folder. This file is identical to the one created and used by OrthoTrak. If the file is present, SIMM will automatically load it and read model parameters from it, such as foot length and hip origin offsets. It will use these parameters to determine joint center locations and segment lengths, using the same algorithms that OrthoTrak does. If there is no *personal.dat* file present in the folder, this option will be grayed out. If it is checked and you do not want to load *personal.dat*, click the box to turn it off.

*read marker names from*

*For C3D import only:* This option allows you to choose from which parameter field in the C3D file to read the names of the tracked markers. Because the `POINT:LABELS` field in a C3D file is limited to four characters, some software packages (e.g., EVaRT) store the full marker name in the `POINT:DESCRIPTIONS` field. Since the marker names in the tracked file must exactly match the names used in the *mocap model*, if your C3D file does not contain full marker names in the `POINT:DESCRIPTIONS` field, you may have to edit the *mocap model* so that the marker names match the four-character names stored in the `POINT:LABELS` field.

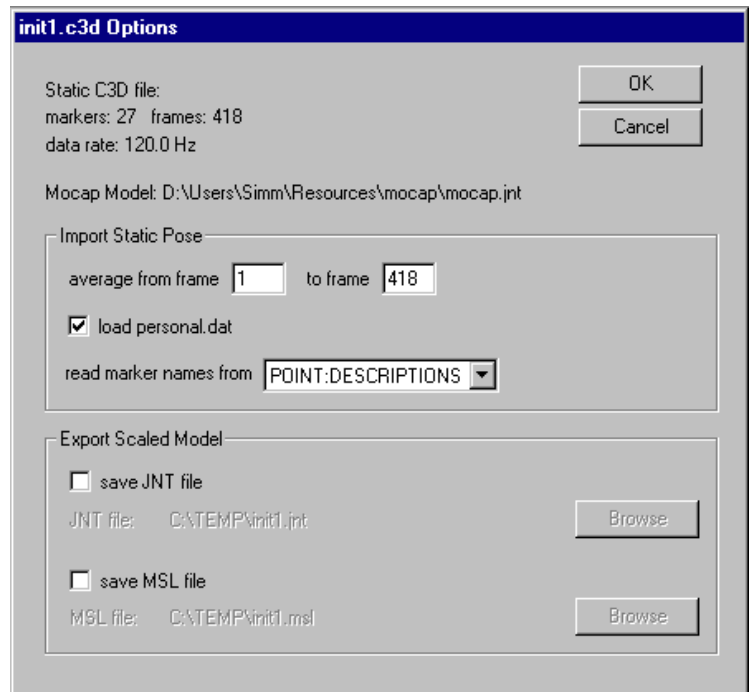


Figure 5-3. Static Trial Import Dialog Box

*save JNT file, save MSL file*

These options allow you to specify if SIMM will write out joint and muscle files containing the musculoskeletal model that is scaled to fit the subject. After SIMM has loaded the mocap model and scaled it based on the data in the static marker file and *personal.dat*, it will write out corresponding joint and muscle files, depending on the states of these check boxes. You may want to create these files so that you can make changes to them or to be able to re-load the model without going through the scaling process again.

### 5.3.3 Calculation of Joint Centers

Once the static pose has been loaded, the Motion Module recreates the OrthoTrak skeletal model from the marker cloud. The first step is determining the locations of the joint centers for all of the joints in the OrthoTrak model. The pelvis, hip, knee, and ankle centers are all found using the same procedure used by OrthoTrak.

The hip center is determined using percentage offsets from the pelvis markers. The Motion Module reads these offsets from *personal.dat*, as written by OrthoTrak. The default values for these offsets are taken from *Bell et al. Journal of Biomechanics, 23(6), 1990, pp. 617-21*:

```
posterior displacement: 22%  
lateral displacement: 32%  
inferior displacement: 34%
```

To change these values, edit the file *personal.dat*, as described in Appendix D of the OrthoTrak manual.

The knee and ankle centers are found using the medial and lateral markers. It is strongly recommended that you use medial markers for a more accurate calculation of joint centers. If you choose not to use them, you should enter knee and ankle diameter measurements into *personal.dat*. The Motion Module will use them to locate the knee and ankle centers if no medial markers are used.

The default method for determining shoulder, elbow, and wrist joint centers uses percentage offsets from the appropriate marker locations. If medial elbow and wrist markers are used in the static trial, their locations are averaged to get the joint centers, as is done with the knee and ankle. It is recommended that you use medial elbow and wrist markers in the static trial if you want an accurate representation of arm motion.



### 5.3.4 Scaling the Mocap Model

Once the locations of the OrthoTrak joint centers have been calculated from the static pose, the Motion Module determines the orientations of the OrthoTrak segment reference frames. It then can measure the lengths of the OrthoTrak segments and use them to scale the mocap model to match the size of the subject.

The reference frames for the foot, shank, thigh, pelvis, and torso are all determined using the procedure described in Appendix H of the OrthoTrak manual.

OrthoTrak does not create reference frames for the upper and lower arms, but the Motion Module does this using one of several methods. If medial elbow and wrist markers are used in the static trial, then the arm reference frames are found in the same way in which the thigh frames are found. If no medial markers are present, then the upper and lower arm reference frames are found using the line between the joint centers as the X axis, and using the same Y axis as the torso. The Z axis is then determined by crossing X and Y.

Once all of the segment reference frames have been determined, the length of each segment is calculated. For most segments, the length is simply the distance from one joint center to the next. For the foot, the Motion Module reads the length from *personal.dat*. If there are no foot length measurements in *personal.dat*, then the foot length is assumed to be 1.4 times the distance from the heel marker to the toe marker (the toe marker is actually placed on the top of the foot just posterior to the toes).

Each body segment in the mocap model contains scaling information that tells the Motion Module how to scale it based on an OrthoTrak skeletal segment. The scaling

information consists of the name of the OrthoTrak segment and X, Y, and Z reference numbers that correspond to the unscaled length of the SIMM segment. For example, the right femur in the mocap model contains the line:

```
gait_scale R_THIGH 0.3960 0.3960 0.3960
```

This tells the Motion Module that the unscaled femur is 0.3960 meters long. Once the length of the corresponding OrthoTrak segment is known (R\_THIGH), the femur can be scaled accordingly. If the R\_THIGH segment were 0.35 meters long, then the femur would be scaled by a factor of 0.35/0.396. In most cases the three reference values are the same number, indicating that the segment should be scaled uniformly in X, Y, and Z. The two exceptions are the TORSO and PELVIS, which are scaled differently in two dimensions. For SIMM segments that do not map directly to an OrthoTrak segment, their scaling information is copied from the most relevant segment. For example, the right hand in the mocap model copies the scaling information from the right lower arm, so that the hand is scaled the same amount as the lower arm.

### 5.3.5 The Marker Set

The marker set in the full-body *mocap model* that comes with SIMM includes every marker used in all six marker sets that OrthoTrak recognizes, plus the medial knee and ankle markers. In addition, many other markers have been added, such as medial elbow and wrist markers. For a complete list of the markers in the model, as well as information on when they should be used and where they should be placed on the subject, read the *Guide to Mocap Model Markers* document. The mocap model contains over 80 markers, which is more than the number used in most applications. When the static trial is loaded, any marker in the mocap model which is not in the static trial is removed from the model. Thus it is not a problem to

have extra markers in the mocap model. In fact, you should add to the model whatever extra markers you may need for any of your motion capture applications. Then for a particular application the mocap model will have all the necessary markers, and the unused ones will automatically be removed when the model is loaded into SIMM.

To add or change markers in the mocap model, use the Marker Editor, which is described in Section 2.12. As discussed in Section 5.3.1, you should be careful not to overwrite the original *mocap.jnt* file. Instead, after editing the marker set, save the model to a new file name, and copy the altered markers into *mocap.jnt*.

All of the markers in the mocap model have X, Y, and Z offsets that put them in realistic locations given the dimensions of the generic model. Thus if you load the unscaled mocap model into SIMM (using the *File... Open* command, not the *File... Open Mocap Model* command, which will scale it), the markers will appear in positions corresponding to where they are placed on the subject. *These offsets are purely decorative, to help you view the marker set. They are not used by the Motion Module to process any marker data.* To explain why this is so, we must first introduce the concept of *critical* markers and *non-critical* markers.

*Critical* markers are ones that *must* be present in the static trial in order for the Motion Module to load and scale the mocap model. For the lower body, these markers are: V.Sacral, R.ASIS, L.ASIS, R.Knee (or R.Knee.Lateral), R.Ankle (or R.Ankle.Lateral), R.Heel, R.Toe, L.Knee (or L.Knee.Lateral), L.Ankle (or L.Ankle.Lateral), L.Heel, and L.Toe. If any of these markers is missing from the static trial, the SIMM model of the lower body will not be loaded. For the upper body, the critical markers are: V.Sacral, R.ASIS, L.ASIS, R.Shoulder, R.Elbow, R.Wrist,

L.Shoulder, L.Elbow, L.Wrist. *Non-critical* markers are all other markers in the set. Note: starting with SIMM 4.0, the critical marker set has been modified slightly. For example, instead of using the sacral marker, V.Sacral, you may use the right and left PSIS markers, R.PSIS, and L.PSIS. Consult the *Guide to Mocap Model Markers* for complete details on which combinations of marker sets are required for all of the *mocap models*.

Once the Motion Module has determined the locations of the joint centers and the orientations of the segment reference frames from the static pose, it calculates the proper offsets for all of the critical markers (plus the static-only medial markers). For example, once the right thigh reference frame has been oriented within the static pose marker cloud, the exact positions of the critical markers attached to the right thigh can be measured directly from the static pose and entered into the mocap model, thus overwriting whatever offsets were in the model input file.

After the offsets of all the critical markers have been determined in this fashion, the mocap model is “fit” to the static pose marker cloud (as described in Section 5.1) using only the critical markers to find the best fit. This process orients the mocap model within the marker cloud, so that the offsets of the non-critical markers can be measured directly from the static pose. These offsets are then entered into the model, overwriting whatever values were in the model input file.

To summarize, the Motion Module uses a two-step process to calculate proper offsets for all of the markers in the mocap model. The first step determines the offsets of the critical markers, which the OrthoTrak algorithms can definitively locate without knowing anything about the mocap model. Then these critical markers are placed on the mocap model, and the model is fit to the static pose

marker cloud. Now the offsets of the other markers can be measured, because every body segment in the mocap model is now correctly placed in the static pose.

## 5.4 Analog Configuration Files

SIMM can include analog data such as ground reaction forces, EMG activation levels, and kinetic data when importing a motion. SIMM uses a configuration file named *importVariables.txt* to determine which analog variables to import from an analog file, and how the data for each variable should be interpreted. This configuration file is used for both TRB/TRC import (with corresponding ANB/ANC analog files) and for C3D import (where the analog data is contained in the C3D file itself). SIMM can interpret analog data as one of three types:

- forceplate data* These variables specify voltages representing force or moment components as measured by a forceplate transducer. Given the voltages generated by a forceplate (6 channels for an AMTI or Bertec forceplate, 8 channels for a Kistler forceplate) SIMM can calculate and display a force location and vector for the forceplate.
- EMG data* These variables define activation levels for one or more muscles in the SIMM full-body model. SIMM rectifies, smooths, and scales EMG data so that it can be plotted, and used to control the width and color of muscles during an animation.
- other data* Any variables that are not forceplate or EMG data are classified as other data. SIMM does not perform any calculations on these data variables, but they may be included in SIMM plots.

### 5.4.1 importVariables.txt

The *importVariables.txt* file, located in *SIMM\Resources\mocap\misc*, contains a list of variable names and attributes. When SIMM processes an analog data file or an OrthoTrak XLS file, it consults the *importVariables.txt* file to decide which variables to import and how to interpret them.

Each row in *importVariables.txt* defines a variable to be imported. The first column in a row specifies the name of the variable as it appears in the analog or XLS data file. Since certain analog files support variable names with spaces in them, the first column of the *importVariables.txt* file *must* be terminated by a tab character. SIMM considers all characters from the beginning of a row until the first tab character to be the name of the import variable. SIMM does a case-insensitive comparison when matching variable names defined in *importVariables.txt* with variable names in an analog data file. Therefore the name “Rt Tibialis” would be considered the same as “rt tibialis”.

The second column in a variable definition specifies the type of the variable. It must be one of the following keywords: `force_plate`, `muscles`, or `other_data`. These keywords must be lowercase. Following each keyword is information describing the variable:

`force_plate`

This keyword specifies a ground reaction force variable. It must be followed by three values:

1. The forceplate number (1, 2, 3, etc.), then
2. The keywords `force` or `moment`, then
3. The channel component (`x`, `y`, or `z` for AMTI or Bertec forceplates, or `x12`, `x34`, `y14`, `y23`, `z1`, `z2`, `z3`, `z4` for Kistler forceplates).

- `muscles` This keyword specifies an EMG variable. It must be followed by one or more SIMM muscle names. The keyword `mvc` may optionally appear after the last muscle name. If `mvc` appears, then it must be followed by an integer number that SIMM uses as the voltage for the maximum voluntary contraction when scaling that EMG channel. If no MVC value is specified, then the channel is scaled such that its maximum value is 1.0. EMG scaling is performed *after* the EMG channel's data has been smoothed and resampled to the motion's frequency.
- `other_data` This keyword specifies a data channel that exists simply to be included in SIMM plots. This keyword may be optionally followed by a single word that will be used to label this channel in SIMM plots. If no name follows the `other_data` keyword, then the name of the imported variable will be used.

### 5.4.2 **forcepla.cal**

When importing analog data from ANB/ANC files, SIMM uses the same calibration file as EVaRT and OrthoTrak for processing forceplate data. Therefore, you can simply copy the *forcepla.cal* file from your EVaRT folder into the *Resources\mocap\misc* folder. For users who need to create a *forcepla.cal* file to describe their forceplate(s), refer to Appendix C of the OrthoTrak manual.

Note: *forcepla.cal* is not used for C3D import since C3D files contain the necessary calibration information for the force plates.





# Appendix A: Tips and Caveats

---

## A.1 Tips on Using SIMM

The following is a list of tips on using SIMM. It is strongly recommended that you read through the entire list before using SIMM for any of your work.

- Use muscle groups. If you have more than about 15 muscles in your model, you will find it much easier to select the ones you want if they are divided into several groups. In the lower-extremity model provided with SIMM, the muscles are grouped by function. You could also group them by name, the number of joints they span, or any other criterion that you want.
- Use body segment groups and generalized-coordinate groups to make large models easier to navigate and display.
- Try to keep muscle names short. When SIMM reads in a muscle file, it scans for the longest muscle name, then makes every muscle group menu wide enough to hold that name. This is done so that it is easy to position many group menus at once in a window. To avoid large menus that run beyond the window borders, try to limit muscle names to 10 letters.

- In addition to selecting **delete** from the Plot Maker command menu, there is another way to delete plot curves. Pressing the `Delete` or the `Backspace` key in a plot or plot key window will delete all of the selected curves from the plot.
- To delete a model, click the close box in the upper right corner of the model's window, or press the `Delete` or `Backspace` keys in the model's window.
- If the display of your model is too slow (e.g., for smooth animation), try changing the drawing mode for some of the body segments. For small polygons, *flat shading* is often indistinguishable from *gouraud shading*, but is usually faster. For example, when viewing an entire human skeleton, you can probably use *flat shading* for the hand and foot bones without affecting the quality of the display a great deal.
- When displaying shadows of the body segments, you usually want the shadows to be cast on a world object, such as the floor or a wall. When setting the locations of the shadows and world objects to achieve this effect, you should position the shadow a small distance away from the world object so that you can see it. For example, if you position the floor with a Y-coordinate of -0.9, you should position the shadows at [about] -0.895 along the Y-axis. If you put them at exactly -0.9, they would be coincident with the floor, and you may or may not see them, depending on the viewing angle.
- When a body segment or world object is displayed in wireframe mode, the color that is used to draw the object is taken from the *ambient* color component of the material assigned to that object.

For the *def\_bone* material, this color is not white, but rather a light gray. If you want pure-white wireframe objects, you should define a new material with an ambient color of 1.0 1.0 1.0, and assign it to the object you want to display in wireframe mode.

- If you press a key or select a menu item and nothing seems to happen, you may have lost the input focus to that window. This means that your key presses are either “going” to another window, or getting lost entirely. This can happen when you press a lot of keys while moving the cursor into several different windows. To fix it, just release all the keys and mouse buttons, click the left mouse button on a different window, and then click it again over the window you were originally working in.
- Whenever SIMM pops up a confirm window, you can use keyboard keys as well as the right mouse button to confirm the action. Pressing *y* will select the *yes* box, and pressing *n* or *c* will select the *cancel* box.

## **A.2 Caveats and Limitations**

The following is a list of warnings and limitations of SIMM. It is strongly recommended that you read through the entire list before using SIMM for any of your work.

- It is sometimes difficult to determine which model or plot you are working on with a given tool. In most cases, the only indication is the information line at the top of the tool window. After you have made a plot, for example, it is impossible to tell which model the curves were generated from.
- Be careful when editing a function within the Joint Editor or Gencoord Editor. SIMM limits the point you are moving so that it stays between the two points it started between, but you can still create a bad function. If you move the point to a side of the bounding region so that it has nearly the same X coordinate as another point, the function may misbehave. If you want to create a knot (coincident control points) in a spline function, you should do it by typing the control point values into the function definition in the input file. It is very difficult to create knots interactively using SIMM's graphical function editor.
- When you select items from command menus, the corresponding action is performed as soon as you press down the left mouse button. You cannot abort the action by moving the cursor out of the box before releasing the button.
- You cannot interactively change the names of muscle groups, joints, generalized coordinates, world objects, or materials.
- You cannot interactively edit the generalized coordinate ranges for wrapping points. You can

move the point itself (by moving the joint so that the point is “active,” and then selecting it), but you cannot change the generalized coordinate values for which the point is active.



# Appendix B: SIMM Resources

---

The SIMM installation procedure creates a directory named *Resources* within the directory where SIMM is installed. For SIMM to execute properly the SIMM Resources directory must not be moved or renamed. SIMM needs to access several files within the Resources directory while it runs. Some of these files can be used to modify the way SIMM works. The following sections describe these files.

## B.1 Preferences

The *preferences* file is read by SIMM when it starts up. This file can be edited by the user to change SIMM's behavior. To change one of the options in the file, you either change its on/off state, or comment it out, as described below. Whenever you change the *preferences* file you must restart SIMM for the changes to take effect.

SET\_TOOLS\_TO\_NEW\_MODEL

This option, when present, causes all of the tools in SIMM to set themselves (via the model selector) to a new model as soon as you load it into SIMM. Thus you can use the tools to edit a new model without having to use each tool's model selector. This option is enabled by default. To disable it, comment it out by putting a # at the beginning of the line.

SET\_TOOLS\_TO\_NEW\_MODEL

This option causes the Plot Maker to change its x-variable and y-variable settings according to the current plot. If you have several plots on the screen which involve different x- and y-variables, then when you use the plot selector to set

the Plot Maker to a particular one, the x- and y-variable settings will change to the ones used in the plot you are choosing. Thus you can add additional curves (of the same type) to the plot without resetting these variables. This option is enabled by default. To disable it, comment it out by putting a # at the beginning of the line.

FASTER\_MUSCLE\_DRAWING

This option causes SIMM to display the muscles without their attachment points, and with a coarser cylinder representing their lines of action. SIMM performs this faster muscle drawing only when animating a motion continuously, using the **start** button in the Model Viewer. If you have many muscles in your model, this option can significantly increase the display of your model, making the animation playback much smoother. This option is on by default. To turn it off, change its value to "off."

SNAPSHOT\_FILE\_NAME

This option specifies the base file name to use when creating snapshot files of models in the Model Viewer. See the description of the **snapshot** command in Section 2.4.2, Command Menu, for more details. By default this option is set to "snapshot." To use a different file name, delete the default and type in a new name.

DISPLAY\_ANIMATION\_HZ

When this option is on, SIMM will display the frequency of the model display in the lower left corner of the model window, but only when animating a model using the **start** button in the Model Viewer. This frequency value, shown in frames per second, depends on the performance of your graphics card and the complexity of your model. This option is off by default. To turn it on, change its value to "on."



DISPLAY\_POLYGONS\_PER\_SECOND

When this option is on, SIMM will display the number of polygons per second it is drawing, in the lower left corner of the model window, but only when animating a model using the **start** button in the Model Viewer. This value depends on the performance of your graphics card and the complexity of your model. This option is off by default. To turn it on, change its value to "on."

## B.2 Colors

You can change any of the colors in SIMM by editing the color file, called *simmcolors*, in the SIMM Resources directory. You may want to make a backup of the file before you change it. To change colors in SIMM, find the name of the item whose color you want to change in the *simmcolors* file. After the name are listed the red, green, and blue components, respectively, of the color. The numbers can range from 0.0 to 1.0. The *simmcolors* file is loaded by SIMM each time it starts up, so to see the effects of your changes you will need to restart SIMM each time you make a change to *simmcolors*. Additionally, each time SIMM opens a joint file (.jnt), it looks for a file named *simmcolors* in the same directory as the joint file.

Note: model-specific colors and materials are specified in the joint file for that model, not in the *simmcolors* file. See Section 3.3, Joint Files, and Section 3.3.7, Materials, for more information on changing model-specific colors and materials.

## B.3 Password.txt

The first time you run SIMM on your computer you will be prompted to enter your SIMM password. At that point SIMM will automatically store your password in a file named *Password.txt*. This file is created by SIMM simply as a convenience to you so you will not need to reenter your password each time you run SIMM.

## B.4 SystemInfo.txt

Each time SIMM runs it creates a file named *SystemInfo.txt* containing information about your computer. This information can be useful to us when helping you resolve technical problems with SIMM. If you request technical support by sending email to [support@musculographics.com](mailto:support@musculographics.com), please include a copy of the SIMM SystemInfo.txt file with your message.

## B.5 Bones

The *bones* subdirectory in the SIMM resources directory contains the bone files used to display the SIMM demo model. The bones in this directory are in the "newer" ASCII format. If you want to use them in another model that you are developing, you should copy them to a *bones* directory located in the directory containing your joint and muscle files (see Section 3.2, Opening Files, for more details).



---

# Index

## A

- act override (toggle button) 43
- activation, muscle
  - defined 56
  - resetting 52
  - specifying in file 188
- active (toggle button) 42
- analog data files
  - opening 217
- animations, see snapshots

## B

- body segment groups
  - changing drawmode 20
  - defining in joint file 153
- body segments 151–154
  - changing display parameters 118
  - restoring 113
  - saving to buffer 113
  - setting inertial parameters 115
- Bone Editor 119–126
  - boolean operations 120
  - command menu 120
  - deleting polygons 120
  - flipping normals 122
  - information header region 119
  - running norm on bone 120
  - splitting polygons 120
- bone files 142–146
  - opening 13

- bones
  - deleting 122
  - moving to different body segment 122
  - reading from files 123
  - specifying in body segments 151
  - writing to files 123

## C

- C3D files
  - see tracked marker files
- cameras
  - restoring position of 18
  - saving position of 18
  - specifying in joints file 167
- capturing images, see snapshots
- color factor
  - using to display muscles 52
- colors
  - setting in simmcolors file B-3
- Constraint Editor 103–112
  - information header region 104
  - objects command menu 105
  - points command menu 110
- constraint objects
  - creating 105
  - defined 103
  - deleting 107
  - restoring 105
  - saving 105
  - specifying in joint file 179
  - transforming 108
- constraint points
  - creating 110
  - tolerance 112
- contact objects

---

- adding to body segments 116
- creating markers 100
- cropping motions 138

## D

- def\_bone 152, 164, 165

- default\_value

  - of generalized coordinates 67, 159

  - of materials 166

- Deform Editor 84–99

  - attributes panel 88

  - command menu 88

  - information header region 86

  - tips and techniques 98

  - transform panel 91

- deform objects

  - auto-reset deform objects 95

  - combining multiple 85, 93

  - creating 88

  - defined 85

  - deformity sliders 85, 96

  - deleting 88

  - editing attributes 88

  - inner and outer boxes 85

  - specifying in joint file 172

  - tips and techniques 98

  - transforming 91

- deformities

  - specifying in joint file 176

- deleting

  - bones 122

  - markers 101

- drawmode

  - flat\_shading 153

  - gouraud\_shading 153

  - none 153

  - of body segment groups 20

  - outlined 153

  - setting in Segment Editor 113

  - setting interactively 20

  - solid\_fill 152

  - specifying in joint file 152

  - wireframe 152

- dynamics

  - including muscles in 133

  - input and output options 130

  - normalizing output 133

  - output variables 135

  - pausing a simulation 135

  - running in SIMM 127

  - setting simulation time 131

  - smoothing input data 132

  - specifying gravity 134

- Dynamics Tool 127–137

  - information header region 130

## E

- ellipsoid wrapping methods 81–83

  - axial 83

  - hybrid 82

  - midpoint 82

  - specifying in Wrap Editor 78

- events

  - adding to motions 140

## F

- fiber length, defined 36

- fixed segment

---

- changing interactively 20
- fixed segment, defined 154
- flat\_shading, defined 153
- force mattes
  - adding to body segment 116
- force, defined 36
- form
  - defined 9
  - editing 9
- forward dynamics 127

## G

- Gencoord Editor 61–70
  - command menu 63
  - dof form 66
  - information header region 62
- generalized coordinate groups
  - defining in joint file 160
  - slider display 21
- generalized coordinate values
  - restoring 18
  - saving 19
- generalized coordinates 158–160
  - calculating velocities of 195
  - clamping 25, 65
  - default value 67, 159
  - defining restraint functions 159
  - locking 25, 65
  - restoring from buffer 63
  - saving to buffer 63
  - slider display 21
  - specifying velocities 196
  - tolerances for 66
  - typing in values 66
- gouraud\_shading, defined 153

- gravity
  - in dynamic simulations 134
  - specifying in joint file 149

## I

- image capture, see snapshots
- inertia matrix
  - specifying for body segments 115
- inertial parameters
  - setting 115
- Inverse Dynamics
  - specifying motion for 132
- inverse dynamics 127
- inverse kinematics
  - defined 67
  - specifying in joint file 147
  - turning on/off 67
- isometric (toggle button) 43

## J

- Joint Editor 44–49
  - command menu 45
  - dof form 47
  - function plot area 48, 69
  - information header region 44
- joint files 146–177
  - opening 12
  - saving 14
- joints 155–157
  - restoring from buffer 47
  - saving to buffer 46

---

## K

kinematic functions 160  
    adding control points 46  
    deleting control points 46

## L

ligaments 192  
look at (command) 20

## M

marker cloud  
    defined 209  
Marker Editor 99–103  
    command menu 100  
    information header region 99  
markers  
    creating 100  
    deleting 101  
    fixed 102  
    name 101  
    offset 101  
    radius 102  
    restoring from buffer 101  
    saving to buffer 101  
    segment of attachment 101  
    visible 102  
    weight 102  
mass  
    specifying for body segments 115  
mass center  
    specifying for body segments 115  
material properties  
    ambient 165

    diffuse 165  
    emission 165  
    specular 165  
materials  
    assigning to body segments 151  
    assigning to bones 151  
    assigning to world objects 164  
    default 165, 166  
    defined 165  
    properties, see material properties  
    setting in Segment Editor 115  
maximum isometric force, defined 56  
menu, defined 7  
mocap model 223–235  
    choosing 225  
    joint center calculations 230  
    marker set 232  
    scaling 231  
    static pose 226  
model display  
    animating motions 25, 29  
    moving joints 29  
    muscle attachment points 26  
    rotating 27  
    trackball method 26  
    zooming and panning 27  
model selector, defined 11  
model view, see camera  
Model Viewer 16–29  
    generalized coordinate form 24  
    information header region 18  
    model name form 24  
    model window viewing commands 27  
    moving joints 29  
    rotating the model 27  
    sliders 25



- 
- moment arm, defined 35
  - moment, defined 35
  - moment@maxforce, defined 36
  - Motion Editor 137–140
    - command menu 138
    - creating events 140
    - cropping motions 138
    - information header region 138
  - motion events
    - creating 140
  - motion files 193–200
    - opening 13
    - saving 15
    - specifying in joint file 147, 194
  - Motion Module 209–237
    - opening analog data files 217
    - opening C3D files 210
    - opening TRB/C files 210
    - opening XLS files 217
    - real-time import 220
  - motion objects 197–198
    - defined 193
    - specifying in joint file 177
    - specifying in muscle file 197
  - motions
    - adding events 140
    - animating 25, 29
    - cropping 138
    - displaying continuously 19
  - Muscle Editor 49–57
    - command menu 50
    - information header region 49
    - model window keyboard commands 60
    - moving muscle attachment points 59
    - muscle parameters form 55
    - selecting muscle attachment points 58
  - muscle files 184–192
    - opening 12
    - saving 14
    - specifying in joint file 147, 184
  - muscle menu, defined 8
  - muscle orientation, defined 37
  - muscle parameters
    - activation, defined 56
    - editing 55
    - maximum isometric force, defined 56
    - maximum thickness, defined 189
    - minimum material, defined 189
    - minimum thickness, defined 189
    - optimal fiber length, defined 56
    - pennation angle, defined 56
    - tendon slack length, defined 56
  - muscle wrapping, see wrap objects
  - muscles
    - adding attachment points 59
    - deleting attachment points 60
    - displaying attachment points 26
    - displaying coordinates 52
    - displaying in model window 19
    - including in dynamics 133
    - moving attachment points 59
    - parameters, see muscle parameters
    - restoring from buffer 51
    - saving to buffer 51
    - selecting attachment points 58
    - surfaces 190
  - musculotendon length, defined 35
- N**
- norm 203–208
-

---

options in Bone Editor 124  
normalizing dynamics output 133

## O

offsetting plot curves 41  
opening  
    bone files 13  
    motion files 13  
    muscle files 12  
    plot files 14  
    tracked marker files 210  
opening files 12  
optimal fiber length, defined 56  
other data  
    in motion files 199  
outlined drawmode, defined 153

## P

passive (toggle button) 42  
password.txt file B-4  
pennation angle, defined 56  
plot curves  
    clipping 40  
    creating, see Plot Maker  
    deleting 32  
    naming 39  
    rectifying 42  
plot files 200–202  
    opening 14  
    saving 15  
Plot Maker 30–44  
    command menu 32  
    generating curves from a model 33

information header region 30  
offset field 41  
options form 38  
plotting data from motion files 34  
scale field 41  
toggle buttons 42  
X-variables 37  
Y-variables 35  
plot points  
    creating 73  
    deleting 72  
plot selector, defined 11  
Plot Viewer 71–73  
    command menu 72  
    information header region 71  
    plot window commands 72  
    toggle buttons 72  
plots  
    creating, see Plot Maker  
    deleting 32  
    showing cursor 72  
    showing events 72

## Q

quick solve  
    defined 213

## R

real-time import  
    see Motion Module  
rectify (toggle button) 42  
restoring  
    body segments 113

---

- camera positions 18
- generalized coordinate values 18
- generalized coordinates 63
- joints 47
- markers 101
- muscles 51
- restraint functions 161–163
  - adding control points 63
  - creating 64
  - deleting control points 64
  - making active/inactive 65
  - saving as new function 64

## S

- saving
  - body segments 113
  - camera positions 18
  - generalized coordinate values 19
  - generalized coordinates 63
  - joint files 14
  - joints 46
  - markers 101
  - motion files 15
  - muscle files 14
  - muscles 51
  - plot files 15
- saving files 14
- scaling plot curves 41
- Segment Editor 112–119
  - command menu 113
  - information header region 113
  - setting inertial parameters 115
- selector menu 11
- shadows
  - defining 153

- displaying 26
  - specifying color of 153
- slider, defined 10
- snapshots 21–24
  - auto-snapshot mode 22
  - choosing snapshot file 21
  - combining multiple 23
  - including an alpha channel 22
  - motion snapshots 23
  - specifying dimensions 23
  - taking 22
- solid\_fill, defined 152
- spring floors
  - adding to body segments 117
- spring points
  - adding to body segments 117
- static pose
  - see Motion Module
- sum (toggle button) 42
- surface normals 203
- systemInfo.txt file B-4

## T

- tendon slack length, defined 56
- tendon strain, defined 36
- TIFF images, see snapshots
- toggle button, defined 10
- tracked marker files
  - cropping ends 214
  - opening 210
- TRB/TRC files
  - see tracked marker files

---

## U

units

specifying in input files 142

## V

velocities

of generalized coordinates 195

of muscle fibers 43, 185

specifying in motion file 196

## W

wireframe, defined 152

world objects 164

Wrap Editor 73–84

command menu 74

information header region 74

tips & techniques 83

wrap objects

creating 74

defined 73

specifying in joint file 168

transforming 79

wrapping methods, see ellipsoid wrapping

methods