SimTK Documents

OpenSim Advanced User & Developer Workshop

March 19-21, 2012, Stanford University

Websites: SimTK.org/home/opensim and opensim.stanford.edu

OpenSim Workshop Agenda

Day One – Monday, March 19, 2012
Li Ka Shing Center, Room 005, Stanford University

8:30 – 9:00am	Welcome, Workshop Goals, and Meet the OpenSim Team *Scott Delp and Jen Hicks*
9:00 – 10:15am	Participant Introduction and Goals *You: Each presenter will be limited to 2 min + 1 min Q&A*
10:15 – 10:30am	BREAK
10:30 – 12:00pm	Generating Forward Simulations with OpenSim: Theory, Best Practices, and Examples *Ajay Seth, Sam Hamner, and Tim Dorn*
12:00 – 1:00pm	LUNCH
1:00 – 1:45pm	Components of an OpenSim Model with a Hands-On Example *Matt DeMers*
1:45 – 2:00pm	BREAK
2:00 – 2:15pm	Solidify Project Plans
2:15 – 5:00pm	Work on Projects
6:00pm	Informal Social Outing in Downtown Palo Alto

Day Two – Tuesday, March 20, 2012
Li Ka Shing Center, Room 005, Stanford University

8:30 – 9:00am	Work on Projects
9:00 – 10:00am	Breakout Session: Hands-On Introduction to the OpenSim API *Ajay Seth*
9:00 – 12:00pm	Work on Projects
12:00 – 1:00pm	LUNCH
1:00 – 1:30pm	Preview of OpenSim 3.0 *Jen Hicks and Ayman Habib*
1:30 – 5:00pm	Work on Projects

Day Three – Wednesday, March 21, 2012
Li Ka Shing Center, Room 005, Stanford University

8:30 – 12:00pm	Work on Projects
12:00 – 1:00pm	LUNCH
1:00 – 2:00pm	Prepare Final Presentations
2:00 – 3:45pm	Presentation of Progress, Hurdles, and Future Plans *You*
3:45 – 4:00pm	Closing Remarks *Scott Delp and Jen Hicks*
4:00 – 5:00pm	RECEPTION

Table of Contents

1 **Introduction** **4**
1.1 **Getting the Most Out of an OpenSim Workshop** **4**
1.2 **Where to Find Additional Resources and Support** **4**
1.2.1 The OpenSim GUI 4
1.2.2 Online Documentation 5
1.2.3 Model and Simulation Repository 5
1.2.4 Publications 5
2 **Overview of the OpenSim Workflow** **6**
2.1 **The OpenSim Model** **6**
2.2 **Importing Experimental Data** **6**
2.3 **Scaling** **7**
2.4 **The Inverse Problem** **7**
2.4.1 Inverse Kinematics 7
2.4.2 Inverse Dynamics 8
2.4.3 Static Optimization 8
2.5 **The Forward Problem** **8**
2.6 **Analyzing Simulations** **9**
3 **Scaling** **11**
3.1 **Overview** **11**
3.2 **Settings Files** **11**
3.3 **Inputs** **12**
3.4 **Outputs** **12**
3.5 **Best Practices and Troubleshooting** **13**
3.5.1 Data Collection and Other Preparation: 13
3.5.2 Scale Settings: 13
3.5.3 Evaluating your Results: 13
3.5.4 Troubleshooting Tips: 14
4 **Inverse Kinematics** **15**
4.1 **Overview** **15**
4.2 **Inputs** **15**
4.3 **Outputs** **16**
4.4 **Best Practices and Troubleshooting** **16**
4.4.1 Data Collection and Other Preparation 16

4.4.2 Inverse Kinematics Settings 16
4.4.3 Evaluating your Results 16

5 **Inverse Dynamics 18**
5.1 **Overview 18**
5.2 **Settings File 18**
5.3 **Inputs 18**
5.4 **Outputs 19**
5.5 **Best Practices and Troubleshooting 19**

6 **Static Optimization 20**
6.1 **Overview 20**
6.2 **Inputs 20**
6.3 **Outputs 21**
6.4 **Best Practices and Troubleshooting 21**
6.4.1 Static Optimization Settings 21
6.4.2 Troubleshooting 21
6.4.3 Evaluating your Results 22

7 **Residual Reduction Algorithm 23**
7.1 **Overview 23**
7.2 **Settings File 23**
7.3 **Inputs 23**
7.4 **Outputs 25**
7.5 **Best Practices and Troubleshooting 25**
7.5.1 RRA Settings 25
7.5.2 Troubleshooting 26
7.5.3 Evaluating your Results 26

8 **Computed Muscle Control 28**
8.1 **Overview 28**
8.2 **Settings File 28**
8.3 **Inputs 29**
8.4 **Outputs 29**
8.5 **Best Practices and Troubleshooting Tips 29**
8.5.1 CMC Settings 30
8.5.2 CMC Troubleshooting 31
8.5.3 Evaluating your Results 31

9 **Forward Dynamics 32**
9.1 **Overview 32**

9.2 **Inputs** **32**
9.3 **Outputs** **33**
9.4 **Best Practices and Troubleshooting Tips** **33**

10 **Checklist - Evaluating your Simulation** **34**
10.1 **Scaling** **34**
10.2 **Inverse Kinematics** **34**
10.3 **Inverse Dynamics** **34**
10.4 **Static Optimization** **34**
10.5 **RRA** **35**
10.6 **CMC** **35**

11 **Extending the Capabilities of OpenSim** **36**
11.1 **Overview** **36**
11.2 **Organization of OpenSim** **36**
11.3 **OpenSim Model and ModelComponents** **37**
11.4 **OpenSim Application Programming Interface (API)** **38**
11.5 **What is an OpenSim plug-in?** **39**
11.6 **What is an OpenSim "main" program?** **39**
11.7 **OpenSim Developer's Guide** **39**
11.8 **Command Line Utilities** **40**
11.9 **MATLAB Utilities for Data Import** **40**

12 **Example: Forward Simulation with RRA and CMC** **41**

13 **Example: Model Editing** **45**
13.1 **Purpose** **45**
13.2 **Connecting an Additional Segment to the Model** **45**
13.3 **Adding an Additional Actuator** **49**

1 Introduction

1.1 Getting the Most Out of an OpenSim Workshop

The OpenSim team at Stanford puts many hours into preparing for workshops and developing the software, documentation, and examples. As participants, you've put many hours into collecting and analyzing your data and now have traveled from around the world to spend three days working on your projects. There are several guidelines we can follow to ensure that everyone gets a maximum benefit from the workshop:

- Use the didactic lectures, handouts, and online OpenSim resources we've provided as the first step for resolving problems.
- Work together! The participant sitting next to you might be able to answer your question just as well or better than a member of the Stanford team. Included with your handout materials is a list of all of the workshop attendees and their project topics.
- There will be many Stanford graduate students and staff researchers available to help answer questions during the workshop. Everyone has different areas of expertise. Please see the see the workshop leaders to find where best to direct your questions.
- Have fun and take breaks. We've purposely included breaks and time for social interaction and ask that you follow this part of the schedule. Taking the time to rest and recharge is essential for everyone.
- Set a clear and manageable project goal for the workshop. This is the purpose of preparing goals slides and the related pre-workshop interaction.
- Share your results. Create a project on SimTK.org to share your models and simulation results at the end of the workshop, if you haven't done so already. Documenting your work will allow other researchers to build on your findings and give you credit for the discoveries you've made in your research.
- Teach others. We hope you will share what you learn at the workshop with your students and colleagues Please contact us if you are interested in starting an OpenSim user group or leading a workshop at your local institution.
- Fill out our online survey to give us feedback and help us improve OpenSim and future workshops.

1.2 Where to Find Additional Resources and Support

There are many resources available to help with troubleshooting, access models and simulation data, and interact with the rest of the OpenSim community.

1.2.1 The OpenSim GUI

The OpenSim Help menu provides the following resources:

- Direct links for filing a bug or requesting a new feature.
- Direct links to three tutorials for becoming familiar with the OpenSim GUI.

- A “Convert Files…” utility for converting older OpenSim model and setup file formats to the latest version.
- An “Available Objects…” option that opens a panel that lists all the model components, analyses, and tools that are available in OpenSim and lists their setup properties that specify the behavior of these objects.

1.2.2 Online Documentation

We are preparing to launch a new online portal to OpenSim resources, and we are giving you a preview at the workshop. You can find the main support page at:
http://opensim.stanford.edu/support/support_index_test.html

This page has links to all of our online support resources including the User’s and Developer’s Guide, User Forums, Examples and Tutorials, Frequently Asked Questions, a Best Practices and Troubleshooting Guide, Videos, and more.

At the top of the main support page, you will also find a custom Google search box that will let you search all of the available resources for topics and keywords of interest.

In the rest of the document, blue highlighted text indicates a page on confluence. You can find the page using the search box.

1.2.3 Model and Simulation Repository

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

1.2.4 Publications

You can find additional information in the following article:

Delp, S.L., Anderson, F.C., Arnold, A. S., Loan, P., Habib, A., John, C., Guendelman, E.G., Thelen, D.G., OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940-1950, 2007. Please cite this work in any of your own publications that use OpenSim.

2 Overview of the OpenSim Workflow

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

2.1 The OpenSim Model

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

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

2.2 Importing Experimental Data

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

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

See Preparing Your Data in the User's Guide for detailed information about preparing and importing your experimental data.

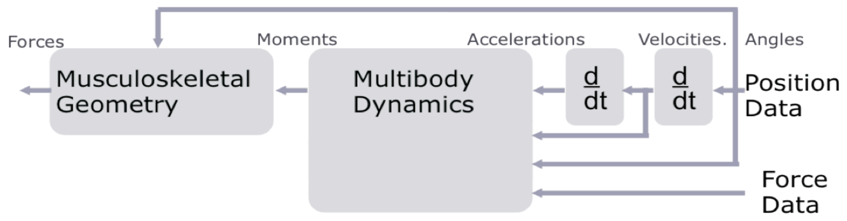
2.3 Scaling

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

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

2.4 The Inverse Problem

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



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

2.4.1 Inverse Kinematics

The Inverse Kinematics (IK) Tool in OpenSim finds the set of generalized coordinates (joint angles and positions) for the model that best match the experimental kinematics recorded for a particular subject (figure below). The experimental kinematics targeted by IK can include experimental marker positions, as well as experimental generalized coordinate values (joint angles). The IK tool goes through each time step of motion and computes generalized coordinate values which positions the model in a pose that "best matches" experimental marker and coordinate values for that time step. Mathematically, the "best match" is expressed as a weighted least squares problem, whose solution aims to minimize both marker and coordinate errors.

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

2.4.2 Inverse Dynamics

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

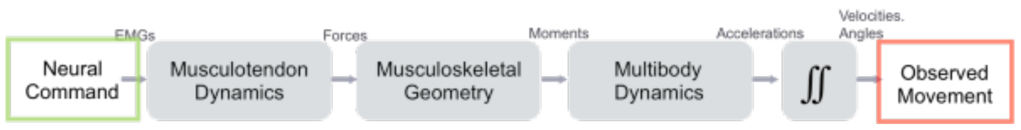
See Inverse Dynamics in the User's Guide for full documentation for running ID in OpenSim. Tutorial 3 - Scaling, Inverse Kinematics, and Inverse Dynamics walks through an example of using ID for human walking.

2.4.3 Static Optimization

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

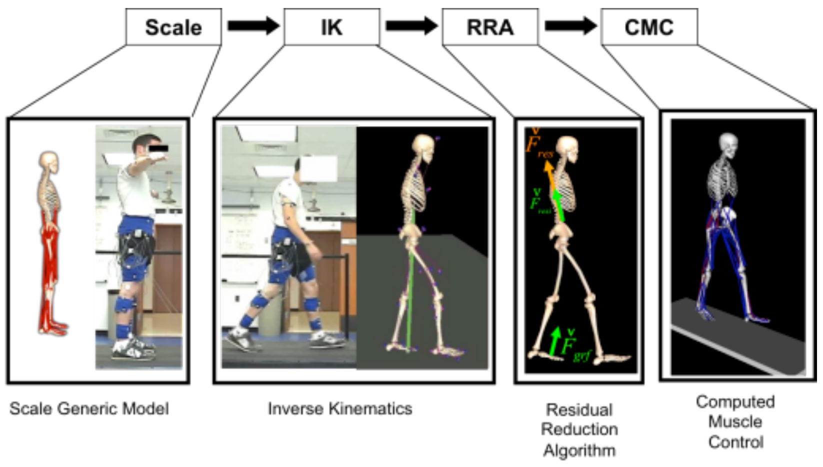
2.5 The Forward Problem

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



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

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



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

2.6 Analyzing Simulations

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

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

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

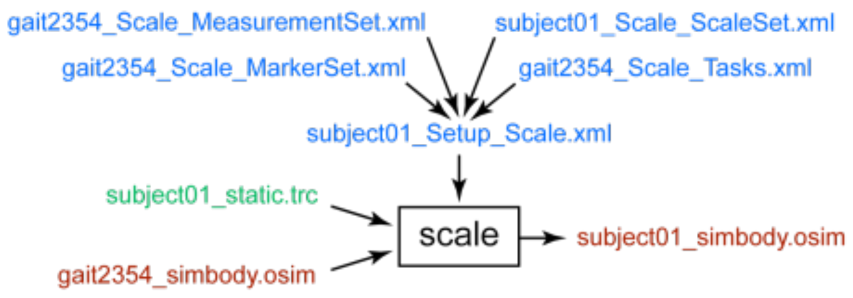
3 Scaling

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

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

3.1 Overview

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


gait2354_Scale_MeasurementSet.xml
subject01_Scale_ScaleSet.xml
gait2354_Scale_MarkerSet.xml
gait2354_Scale_Tasks.xml
subject01_Setup_Scale.xml
subject01_static.trc
scale
subject01_simbody.osim
gait2354_simbody.osim

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

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

3.2 Settings Files

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

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

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

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

AND/OR

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

3.3 Inputs

Two data files are required by the Scale Tool:

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

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

You can also provide an additional, optional file:

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

3.4 Outputs

The Scale Tool generates a single file:

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

3.5 Best Practices and Troubleshooting

3.5.1 Data Collection and Other Preparation:

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

3.5.2 Scale Settings:

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

3.5.3 Evaluating your Results:

1. Scaling a model is an iterative process. Use the "preview static pose" option in the GUI. See the section on "Previewing Scale" in the Settings Pane section for more information. After running preview, perform steps 2 to 5 described below.
2. Check the messages window, which has information about the results of scaling, including the overal RMS marker error and the maximum marker error.
    1. In general, maximum marker errors for bony landmarks should be <2 cm.
    2. RMS error should typically be less than 1 cm.
    3. Pay close attention to errors in the bony landmark and FJC markers when assessing the quality of your scaling results.
3. Visualize the scaled model's anatomical marker positions relative to the corresponding experimental markers to see how well the model "fits" the data. Use the pictures you took to assess the results, comparing the joint angles in the "Coordinates" window to the angles you observe in the pictures.
    1. Do the hip, knee, and ankle angles from scale match what you observe in the picture?

    2. Are there any large mismatches between experimental and model markers? Can these mismatches be explained by examining the pictures you took?
    3. If pictures aren't available, use what you know about a typical static pose capture. For example, the ankle angle is generally less than 5° and hip flexion angle is less than 10°.
    4. Again, pay close attention to errors in the landmark and FJC markers when assessing the quality of your scaling results.
4. After examining the messages window and performing a visual comparison, adjust the virtual markers and marker weightings to improve your results:
    1. Again, avoid adjusting the positions of the landmark and FJC virtual markers to match the experimental markers.
5. Once you've adjusted the virtual marker positions and the scale settings, preview the new static pose. Re-assess your results using steps 2 to 4 above. Once you are happy with your results, hit "Run" to generate a scaled model and adjust the virtual markers on the model to match all of the experimental markers.

3.5.4 Troubleshooting Tips:

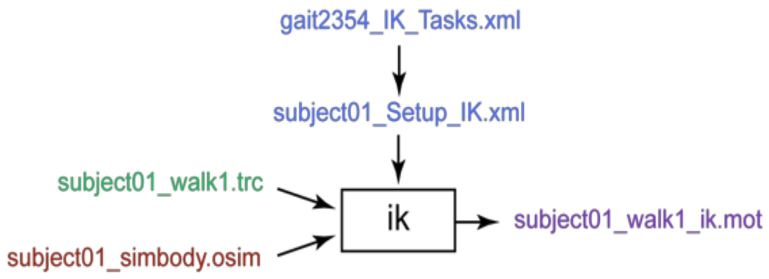
1. It is common to iterate through Scale and Inverse Kinematics to fine-tune segment dimensions and marker positions that yield low marker errors for the task of interest.
2. Use coordinate tasks (Static Pose Weights) to set joint angles for troublesome joints that are very sensitive to how the markers are placed (commonly the ankle joint and lumbar joint). For example if it is known that the foot is flat, an ankle angle can be provided and then the markers adjusted in order to match the known pose.
3. If using coordinates from a motion capture system make sure that the joint/coordinate definitions match otherwise you may cause more harm than good.
4. The model has a built in assumption that the global Y axis is up. If your data doesn't fit this, then consider transforming it. You can use Previewing Motion Capture (Mocap) Data to determine the proper transform to apply.

4 Inverse Kinematics

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

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

4.1 Overview



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

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

4.2 Inputs

The primary inputs to IK are the following files:

1. **subject01_simbody.osim**: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers.
2. **subject01_walk1.trc**: Experimental marker trajectories for a trial obtained from a motion capture system, along with the time range of interest
3. **gait2354_IK_tasks.xml**: A file containing marker weightings. As in the scale tool, marker weights are relative and determine how "well" the virtual markers track experimental markers (i.e., a larger weight will mean less error between virtual and experimental marker positions).

4. **subject01_coords.mot** (optional): Experimental generalized coordinate values (joint angles) for a trial obtained from alternative motion capture devices or other specialized algorithms. You can optionally specify relative coordinate weights in the Tasks file, if joint angles are known a priori.

4.3 Outputs

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

4.4 Best Practices and Troubleshooting

4.4.1 Data Collection and Other Preparation

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

4.4.2 Inverse Kinematics Settings

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

4.4.3 Evaluating your Results

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

4. If you are unsatisfied with the results, recheck the results of Scale.

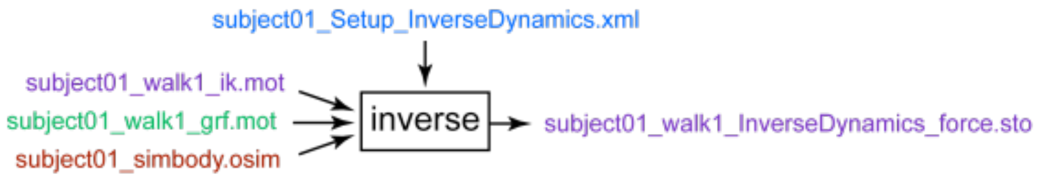
5 Inverse Dynamics

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

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

5.1 Overview

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


subject01_Setup_InverseDynamics.xml
subject01_walk1_ik.mot
subject01_walk1_grf.mot
subject01_simbody.osim
inverse
subject01_walk1_InverseDynamics_force.sto

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

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

5.2 Settings File

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

5.3 Inputs

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

subject01_walk1_ik.mot: Motion file containing the time histories of generalized coordinates that describe the movement of the model. This file could be generated by the

Inverse Kinematics Tool, or manually. The file does not need to contain values for all coordinates. The coordinates that were not specified are assumed to have default values by the Tool.

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

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

5.4 Outputs

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

5.5 Best Practices and Troubleshooting

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

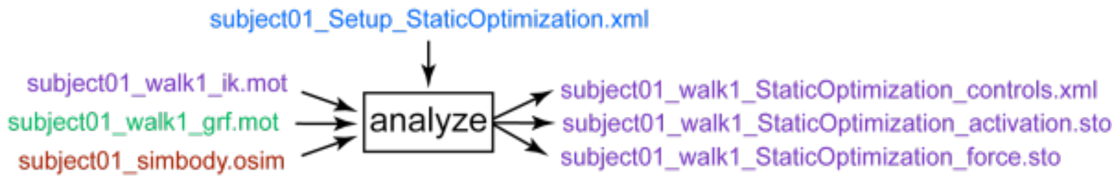
6 Static Optimization

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

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

6.1 Overview

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


subject01_Setup_StaticOptimization.xml
subject01_walk1_ik.mot
subject01_walk1_grf.mot
subject01_simbody.osim
analyze
subject01_walk1_StaticOptimization_controls.xml
subject01_walk1_StaticOptimization_activation.sto
subject01_walk1_StaticOptimization_force.sto

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

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

6.2 Inputs

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

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

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

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

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

6.3 Outputs

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

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

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

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

6.4 Best Practices and Troubleshooting

6.4.1 Static Optimization Settings

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

6.4.2 Troubleshooting

1. If the residual actuators or the model's muscles are weak, the optimization will take a long time to converge or will never converge at all.

    1. If the residual actuators or weak, increase the maximum control value of a residual, while lowering its maximum force. This allows the optimizer to generate a large force (if necessary) to match accelerations but large control values are penalized more heavily. In static optimization, ideal actuator excitations are treated as activations in the cost function.
    2. If the muscles are weak, append Coordinate Actuators to the model at the joints in the model. This will allow you to see how much "reserve" actuation is required at a given joint and then strengthen the muscles in your model accordingly.
    3. If troubleshooting a weak model and each time, optimization is slow, try reducing the parameter that defines the max number of iterations.
2. StaticOptimization works internally by solving the InverseDynamics problem, then trying to solve the redundancy problem for actuators/muscles using the accelerations from the InverseDynamics solution as a constraint. If a constraint violation is reported, this could be a sign that the optimizer couldn't solve for muscle forces while enforcing the InverseDynamics solution.
    1. This likely means that there is noise in the data or there is a sudden jump in accelerations in one frame.
    2. In this case you should examine the Inverse Dynamics solution to examine the problematic frame, and fix/interpolate the data during this portion of the motion.

6.4.3 Evaluating your Results

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

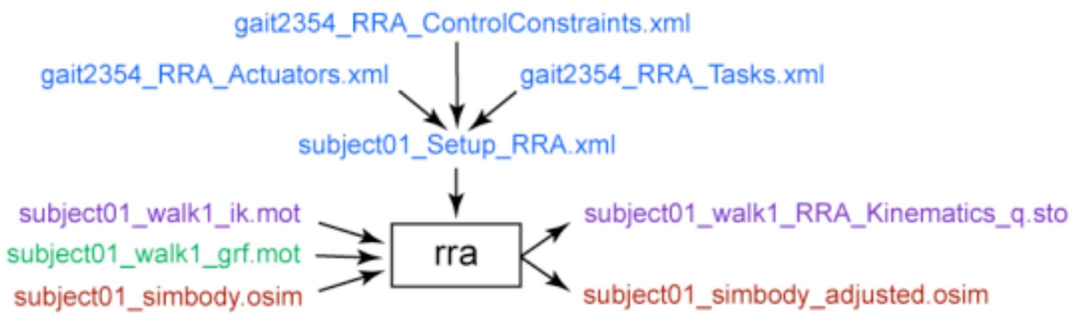
7 Residual Reduction Algorithm

The purpose of Residual Reduction is to minimize the effects of modeling and marker data processing errors that aggregate and lead to large nonphysical compensatory forces called residuals. Specifically, residual reduction alters the torso mass center of a subject-specific model and permits the kinematics of the model from inverse kinematic to vary in order to be more dynamically consistent with the ground reaction force data.

The residual reduction algorithm tool is accessed by selecting **Tools** → **Residual Reduction Algorithm…** from the OpenSim main menu bar. Like all tools, the operations performed by the computed muscle control tool apply to the current model.

7.1 Overview

The figure below shows the required inputs and outputs for performing the residual reduction algorithm. Each is described in more detail below.


gait2354_RRA_ControlConstraints.xml
gait2354_RRA_Actuators.xml
gait2354_RRA_Tasks.xml
subject01_Setup_RRA.xml
subject01_walk1_ik.mot
subject01_walk1_grf.mot
subject01_simbody.osim
rra
subject01_walk1_RRA_Kinematics_q.sto
subject01_simbody_adjusted.osim

Inputs and Outputs for performing residual reduction. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown in blue; files generated by the workflow are shown in purple.

7.2 Settings File

The **subject01_Setup_RRA.xml** file is a setup file for the RRATool, which specifies settings, inputs, and outputs that affect the behavior of the residual reduction algorithm, which can be defined using the GUI or by hand. Details of the settings are described in the section on using the Graphical User Interface.

The setup file identifies the actuators (i.e., the ideal residual and reserve joint actuators required by RRA) as well as the kinematic tracking tasks. Furthermore, control constraints on the actuators (to limit the maximum residual force) can be specified.

7.3 Inputs

Several files are required as input to the RRA Tool to perform residual reduction:

subject01_walk1_ik.mot: Contains the time histories of model kinematics including the joint angles and pelvis translations.

gait2345_RRA_Tasks.xml: A tracking tasks file specifying which coordinates to track and the corresponding tracking weight (weights are relative and determine how "well" a joint angle will track the specified joint angle from IK). A couple key considerations:

1. Selection of kp and kv are not arbitrary. They define the behavior of the error dynamics for each q as a second order linear system. We can write the kp and kv for the desired system behavior in terms of system poles. For a (stable) critically damped system (real negative poles) kp = lambda^2 and kv = -2*lambda.
2. This enables kinematics of joints (coordinates) for which we have high confidence (e.g. knee flexion, hip flexion) to be weighted more heavily compared to those of less confidence (e.g. hip internal rotation and ankle inversion).

gait2345_RRA_ControlConstraints.xml: Contains limits on the RRA actuators. The actuator constraints file specifying the maximum and minimum "excitation" (i.e., control signal) for each actuator. A few key considerations:

1. Note that the maximum/minimum force or torque generated by an ideal actuator is the product of the max/min force and max/min excitation.
2. Joint torques (and muscles) have a maximum magnitude of 1.
3. Residuals have bounds exceeding their anticipated force requirement. Weightings are implicit in this description. A high optimal_force means that large output force (torque) does not require a large control value (i.e. low cost). Conversely, residuals with low optimal force require high control values that incur higher costs.

subject01_walk1_grf.xml: ExternalLoads file specifying the measured ground reaction forces that should be applied to the model during simulation and how to apply them.

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

gait2345_RRA_Actuators.xml: Ideal joint actuators used to replace muscles. The Actuator Set specifies the residual and reserve actuators to be applied and their parameters, like maximum/minimum force and body or joint, or location, depending on the actuator type. A few key considerations:

1. Each degree of freedom in the model should have an ideal torque or force (reserve) actuator. This includes the 6 DOFs of the model's base segment, which are called the residual actuators.

2. In most circumstances, these Ideal joint actuators used to replace the muscles in the model (by checking "Replace model actuators" in the Actuators tab.
3. Optimal forces are the maximum output of ideal actuators (torques, linear forces). Torque (force) applied is optimal_force x control_value
4. Residual at pelvis should be applied at scaled location of COM

7.4 Outputs

Residual reduction generates the following outputs:

subject01_RRA_states.sto: Adjusted kinematics (i.e., joint angles) and corresponding model states of the simulated motion (i.e., joint angles AND velocities).

subject01_adjusted.osim (optional): A model with adjusted mass properties.

subject01_RRA_forces.sto: Actuator forces and torques (i.e., joint torques corresponding to adjusted kinematics).

subjcet01_RRA_controls.xml: Actuator excitations (i.e., control signals needed to generate actuator forces and torques)

7.5 Best Practices and Troubleshooting

7.5.1 RRA Settings

1. You should replace the muscles in your model with residual actuators and ideal joint actuators. Residual reduction is a form of forward dynamics simulation that utilizes a tracking controller to follow model kinematics determined from the inverse kinematics. Computed muscle control (CMC) serves as the controller, but without muscles the skeleton of the model can be used to determine a mass distribution and joint kinematics that are more consistent with ground reaction forces.
2. Optimal forces for residuals should be low to prevent the optimizer from "wanting" to use residual actuators (an actuator with large optimal force and low excitation is "cheap" in the optimizer cost).
3. To help minimize residuals, make an initial pass with default inputs, then check residuals and coordinate errors. To reduce residuals further, decrease tracking weights on coordinates with low error. You can also try decreasing the maximum excitation on residuals or the actuator optimal force.
4. Typically, you should lock" the subtalar and mtp joints in *.osim file.
5. Make sure "use_fast_optimization_target" is false (unchecked). This allows the kinematics to be slightly adjusted to account for dynamic inconsistencies. This is the default in settings files distributed with OpenSim or created from the GUI. See How CMC Works for a comparison of the "slow" and "fast" targets.

6. The “cmc_time_window” in the settings file should be 0.001 s for RRA. This is the default in settings files distributed with OpenSim or created from the GUI.
7. See How RRA Works and How to Use the RRA Tool for more information about RRA settings.

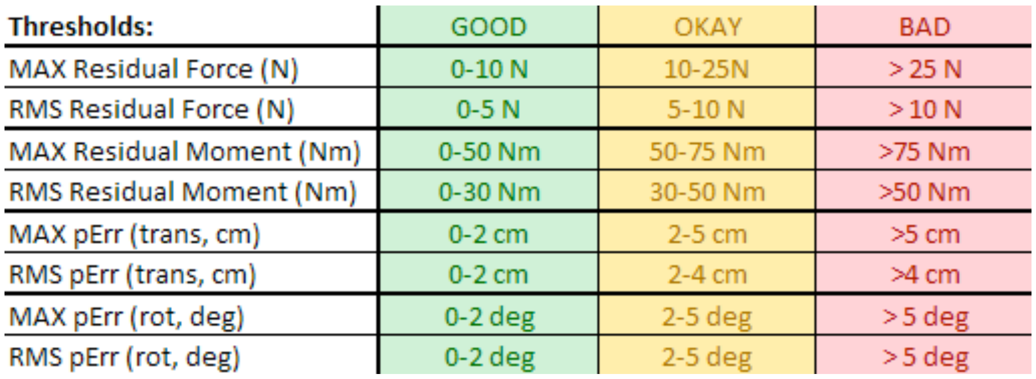
7.5.2 Troubleshooting

1. Check the pelvis COM location in Actuator files.
2. If RRA is failing, try increasing the max excitation for residuals by 10x until the simulation runs. Then try working your way back down while also "relaxing" tracking weights on coordinates.
3. If residuals are very large (typically, this is greater than 2-3x BW, depending on the motion), there is probably something wrong with either i) the scaled model, (ii) the IK solution, or (iii) the applied GRFs. To double check that forces are being applied properly, visualize GRFs with IK data (you can use the Previewing Motion Capture (Mocap) Data function in the GUI).
4. If there is pelvis drift and/or FY is not centered around zero, check that the body mass and force calibration are correct.
5. When using the example RRA actuators XML file, you should note that residual forces are applied to the center-of-mass (COM) of the unscaled pelvis. However, if you scale the model, the COM of the pelvis can change. Although the effect may be small, you should change the location of the residual force actuators in the your RRA actuators file to correspond to the scaled pelvis COM.

7.5.3 Evaluating your Results

1. RMS difference in joint angle during the movement should be less than 2-5^o (or less than 2 cm for translations).
2. Peak Residual Forces should typically be less than 10-20 N. Average residuals should typically be less than 5-10 N.
    1. The size of residuals will depend on the type of motion being studied. For example residuals for high speed activities, like sprinting, will typically be larger than walking.
    2. Residuals will also be larger if there are external forces that you have not accounted for, like a subject walking with a handrail.
3. Compare the residual moments from RRA to the moments from Inverse Dynamics. You should see a 30-50% reduction in peak residual moments.
4. Compare the joint torques/forces to established literature (if available). Try to find data with multiple subjects. Your results should be within one standard deviation of the literature

The table below shows an example of threshold values used to evaluate RRA results for full body simulations of walking and running:



Thresholds:	GOOD	OKAY	BAD
MAX Residual Force (N)	0-10 N	10-25N	> 25 N
RMS Residual Force (N)	0-5 N	5-10 N	> 10 N
MAX Residual Moment (Nm)	0-50 Nm	50-75 Nm	>75 Nm
RMS Residual Moment (Nm)	0-30 Nm	30-50 Nm	>50 Nm
MAX pErr (trans, cm)	0-2 cm	2-5 cm	>5 cm
RMS pErr (trans, cm)	0-2 cm	2-4 cm	>4 cm
MAX pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
RMS pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg

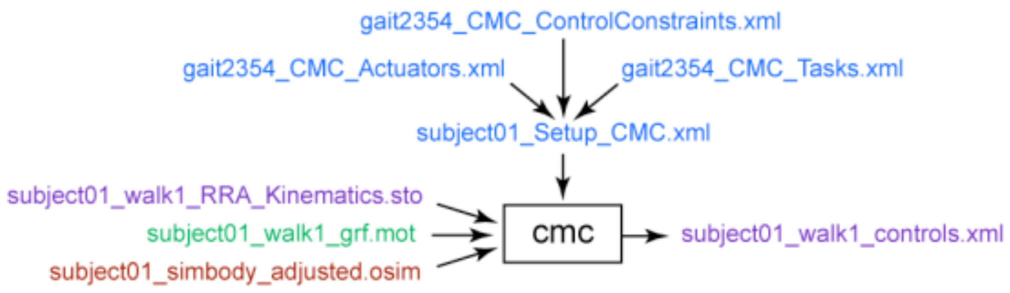
8 Computed Muscle Control

The purpose of Computed Muscle Control (CMC) is to compute a set of muscle excitations (or more generally actuator controls) that will drive a dynamic musculoskeletal model to track a set of desired kinematics in the presence of applied external forces (if applicable).

The Computed Muscle Control tool is accessed by selecting **Tools → Computed Muscle Control...** from the OpenSim main menu bar. Like all tools, the operations performed by the Computed Muscle Control tool apply to the current model.

8.1 Overview

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


gait2354_CMC_ControlConstraints.xml
gait2354_CMC_Actuators.xml
gait2354_CMC_Tasks.xml
subject01_Setup_CMC.xml
subject01_walk1_RRA_Kinematics.sto
subject01_walk1_grf.mot
subject01_simbody_adjusted.osim
cmc
subject01_walk1_controls.xml

Inputs and Outputs of the Computed Muscle Control Tool. Experimental data are shown in green; OpenSim files (.osim) are shown in red; settings files are shown inblue; files generated by the workflow are shown in purple.

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

8.2 Settings File

The **subject01_Setup_CMC.xml** file is a setup file for the CMC Tool, which specifies settings, inputs, and outputs that affect the behavior of the tracking controller to determine actuator (including muscles) controls. These can be defined using the GUI or by hand. Details of the settings are described in the section on the Graphical User Interface.

The setup file identifies the actuators (i.e., the residual actuators, such as required for dynamic consistency) as well as the kinematic tracking tasks. Furthermore, control constraints on the actuators (to limit the maximum residual force) can be specified.

8.3 Inputs

Several data files are required as input by the Computed Muscle Control Tool:

subject01_walk1_RRA_Kinematics_q.sto: Contains the time histories of model kinematics including the joint angles and pelvis translations from RRA.

gait2345_CMC_Tasks.xml: The tracking tasks file specifying which coordinates to track and the corresponding tracking weight (weights are relative and determine how "well" a joint angle will track the specified joint angle from RRA).

gait2345_CMC_ControlConstraints.xml: Contains limits on model actuators, which include muscles, reserve and residual actuators. The control constraints file specifies the maximum and minimum "excitation" (i.e., control signal) for each actuator. Control constraints can also be used to enforce when certain actuators are "on" or "off" and the range in which they can operate.

subject01_walk1_grf.xml: External load data (i.e., ground reaction forces, moments, and center of pressure location). See Inverse Dynamics for more details.

subject01_simbody_adjusted .osim: A subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters. The model should have an adjusted torso center of mass to reduce residuals

gait2345_CMC_Actuators.xml: Contains the residual and reserve actuators, as in RRA.

8.4 Outputs

The Computed Muscle Control Tool tool primarily reports the necessary controls:

subject01_walk1_controls.xml: Contains the excitations to individual muscles as well as controls for any residual and/or reserve actuators.

subject01_CMC_forces.sto: Muscle forces and reserve/residual forces and torques.

For completeness, CMC outputs the state trajectories (these are joint coordinate values and their speeds, as well as muscle states such activation and fiber length) :

subject01_walk1_states.sto: Model states and muscle states of the simulated motion (i.e., joint angles AND velocities, muscle fiber lengths AND activations).

8.5 Best Practices and Troubleshooting Tips

8.5.1 CMC Settings

1. The reserve actuators are torques that are added about each joint to augment the actuator's force in order to enable the simulation to run (reserves turn on when an actuator cannot produce the needed at a given time point). To help minimize reserve torques, make an initial pass with default inputs, and then check reserves, residuals, and joint angle errors. To reduce reserves further, decrease tracking weights on coordinates with low error.
2. Optimal forces for reserves should be low to prevent the optimizer from "wanting" to use reserve actuators (an actuator with large optimal force and low excitation is "cheap" in the optimizer cost). If larger forces are needed for a successful simulation, increase the maximum control value of residuals. The residuals will then be able to generate sufficient force, but will be penalized for doing so.
3. If you are still getting high reserves for a particular degree-of-freedom during a particular time range in the simulation, it may be useful to examine more closely the muscles which span the degree-of-freedom during that time region (see TipsForDebuggingMuscleActuatedSimulations.pdf). In particular:
    1. Check passive muscle force (i.e., quadriceps). Large passive forces (from large knee flexion angles) may induce active forces in the antagonistic muscles (i.e. hamstrings, gastrocnemius) which may not be desired. Passive forces cannot be controlled in CMC - they are purely a function of the whole body kinematics of the motion. Although tempting, do not increase the maximum isometric force excessively unless you know its consequences in the muscle model. Because the passive muscle force is modeled as a function of maximum isometric force, if you increase the maximum isometric force in the hope of making your active muscles stronger, you will be increasing the passive forces in the muscles as well. To decrease passive muscle forces, you should reduce the passive muscle stiffness property of muscle (or more specifically, increase the FmaxMuscleStrain parameter in the Thelen2003Muscle).
    2. Check normalized fiber length during the motion (plotter tool). Is the muscle acting at sub-optimal fiber lengths (i.e. less than 0.8 or greater than 1.2) at the time where the reserves are being generated? If so, then you may consider modifying either the tendon slack length or the optimal fiber length of the muscle so that it is operating more optimally (and thus capable of generating a greater force) during this time in the simulation. Although many cadaver studies report the optimal fiber length of a muscle, the tendon slack length is almost never reported, even though it is especially sensitive to the operating region on the force-length curve. Small adjustments to the tendon slack lengths can therefore be justified in reducing the reserves on the basis that we don't have as much confidence in this value to begin with.
4. You should almost always use the "fast" target for CMC. This requires the joint accelerations at each time step are matched to the RRA results. Fast target should work for normal subjects, slow target may be needed for subjects with pathologies
5. Start CMC at least 0.03 seconds before point where you want to start analyzing your data, as CMC requires a 0.03 sec time for initialization.

6. See How CMC Works and How to Use the CMC Tool for more information.

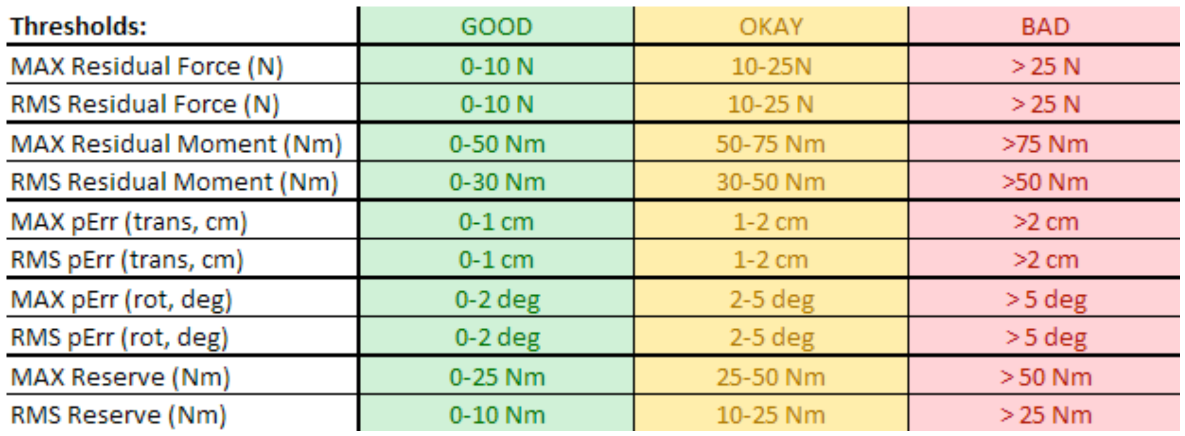
8.5.2 CMC Troubleshooting

1. If CMC is failing, try increasing the max excitation for reserves and residuals by 10x until the simulation runs. Then try working your way back down while also "relaxing" tracking weights on coordinates.

8.5.3 Evaluating your Results

1. Peak reserve actuators torques should typically be less than 10% of the peak joint torque.
2. Peak residual forces should typically be less than 10-20 N and peak residual moments less than 75Nm, depending on the type of motion.
3. Double check your kinematics, in comparison to RRA. Generally, they should match well as long as you are using the "fast" target.
4. If performing an Induced Acceleration Analysis, you should verify that reserves and residuals contribute less than 5% to the net acceleration of interest.
5. Compare the simulated activations to experimental EMG data (either recorded from your subject or from the literature). Activations should exhibit similar timing and magnitude to EMG data. You can also compare your muscle activations and/or forces to other simulations from SimTK or the literature.

The table below shows an example of threshold values used to evaluate CMC results for full body simulations of walking and running:



Thresholds:	GOOD	OKAY	BAD
MAX Residual Force (N)	0-10 N	10-25N	> 25 N
RMS Residual Force (N)	0-10 N	10-25 N	> 25 N
MAX Residual Moment (Nm)	0-50 Nm	50-75 Nm	>75 Nm
RMS Residual Moment (Nm)	0-30 Nm	30-50 Nm	>50 Nm
MAX pErr (trans, cm)	0-1 cm	1-2 cm	>2 cm
RMS pErr (trans, cm)	0-1 cm	1-2 cm	>2 cm
MAX pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
RMS pErr (rot, deg)	0-2 deg	2-5 deg	> 5 deg
MAX Reserve (Nm)	0-25 Nm	25-50 Nm	> 50 Nm
RMS Reserve (Nm)	0-10 Nm	10-25 Nm	> 25 Nm

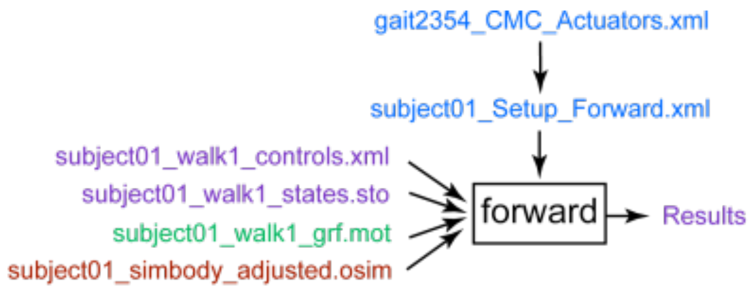
9 Forward Dynamics

Given the controls (e.g., muscle excitations) computed by the Computed Muscle Control (CMC) or another approach, the Forward Dynamics Tool can drive a forward dynamic simulation. A forward dynamics simulation is the solution (integration) of the differential equations that define the dynamics of a musculoskeletal model. By focusing on specific time intervals of interest, and by using different analyses, more detailed biomechanical data for the trial in question can be collected.

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

9.1 Overview

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


gait2354_CMC_Actuators.xml
subject01_Setup_Forward.xml
subject01_walk1_controls.xml
subject01_walk1_states.sto
subject01_walk1_grf.mot
subject01_simbody_adjusted.osim
forward
Results

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

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

9.2 Inputs

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

subject01_walk1_controls.xml: Contains the time histories of the model controls (e.g., muscle excitations) to the muscles and/or joint torques. It is possible to specify the controls as .sto file instead with columns corresponding to desired excitations. This file may be generated by the user, Static Optimization Tool, or Computed Muscle Control Tool. If no controls are provided they are assumed to be zero for any actuators in the model.

subject01_walk1_states.sto: Contains the time histories of model states, including joint angles, joint speeds, muscle activations, muscle activations, muscle fiber lengths, and more. These states are used by the Forward Dynamics Tool to set the initial states of the model for forward integration. Alternately, the simulation can begin from the default pose of the model without providing initial states. Muscle states can be estimated by solving for muscle-fiber and tendon force equilibrium when the *Solve for equilibrium for actuator states* is checked.

subject01_walk1_grf.xml: An xml file describing the External Loads applied based (for example) on measured ground reaction forces that should be applied to the model during simulation

subject01_simbody_adjusted.osim: Subject-specific OpenSim model generated by scaling a generic model with the Scale Tool or by other means, along with an associated marker set containing adjusted virtual markers. The model must include inertial parameters (segment masses, etc.).

The **subject01_Setup_Forward.xml** file is the setup file for the Forward Dynamics Tool. It contains settings, as described in How to Use the Forward Dynamics Tool, and refers to another settings file, **gait2354_CMC_Actuators.xml** which contains a set of actuators that supplement the muscles of the model. Refer to Computed Muscle Control for more details. These actuators must be included in the forward simulation so that the CMC solution can be reproduced.

9.3 Outputs

The Forward Dynamics tool generates results in a folder specified in the setup file:

Results: Additional data can be generated and written to files by adding analyses to the Forward Dynamics Tool. These analyses are specified in the setup file (**subject01_Setup_Forward.xml**) and are discussed in the Analyses section.

9.4 Best Practices and Troubleshooting Tips

1. Forward dynamics simulations are sensitive to initial conditions and it is good practice to double check that they your ICs appropriate for the desired simulation.
2. If the Forward Tool fails gracefully (i.e., without crashing OpenSim) or the output of the Forward Tool drifts too much (i.e., the model goes crazy), shorten the interval over which the Forward Tool runs (i.e., make initial_time and final_time closer to each other in the Forward Tool setup dialog box or setup file). Open-loop forward dynamics tends to drift over time due to the accumulation of numerical errors during integration.

10 Checklist - Evaluating your Simulation

The following are a set of necessary, but not sufficient, questions for evaluating your simulation. You may not be able to answer "yes" to all of the questions. But if the answer is "no", you should be able to provide a plausible explanation to convince yourself (and reviewers).

10.1 Scaling

1. If you are using any coordinates from a mo-cap system, do the definitions match your model?
2. Is maximum marker error for bony landmarks and functional joint centers less than 2 cm?
3. Is the RMS error less than ~1 cm?
4. Do the joint coordinates in the static pose match your knowledge about experimental data collection (comparison to photos, etc.)?

10.2 Inverse Kinematics

1. If you are using any coordinates from a mo-cap system, do the definitions match your model?
2. Is maximum marker error less than 4 cm?
3. Is the RMS error less than ~2 cm?
4. Is there data for similar motions in the literature or other past studies? Are your results within 1 SD?

10.3 Inverse Dynamics

1. Are there any large or unexpected forces at the pelvis (how large)?
2. Is there data for similar motions in the literature or from other past studies? Are your results within 1 SD?

10.4 Static Optimization

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

10.5 RRA

1. Is the RMS difference between experimental and simulated joint angles less than 2-5^{o} (or less than 2 cm for translations)?
2. Are the peak residual forces less than 10-25N? Are the RMS residual forces less than 5-15N?
3. Are the peak residual moments less than 75 Nm? Are the RMS residual moments less than 50 Nm?
4. Are the residual moments reduced 30-50% compared to inverse dynamics?
5. Is there joint torque data for similar motions in the literature or from other past studies? Are your results within 1 SD?

10.6 CMC

1. Are the peak reserve actuators torques less than 10% of the corresponding peak joint torques?
2. Is the RMS difference between experimental and simulated joint angles less than 2-5^{o} (or less than 2 cm for translations)?
3. Are the peak residual forces less than 10-20N? Are the average residual forces less than 5-10N?
4. Are the peak residual moments less than 75 Nm? Are the average residual moments less than 50 Nm?
5. Find EMG or muscle activation data for comparison with your simulated activations. Does the timing of muscle activation/deactivation match? Are the magnitudes and patterns in good agreement?

11 Extending the Capabilities of OpenSim

11.1 Overview

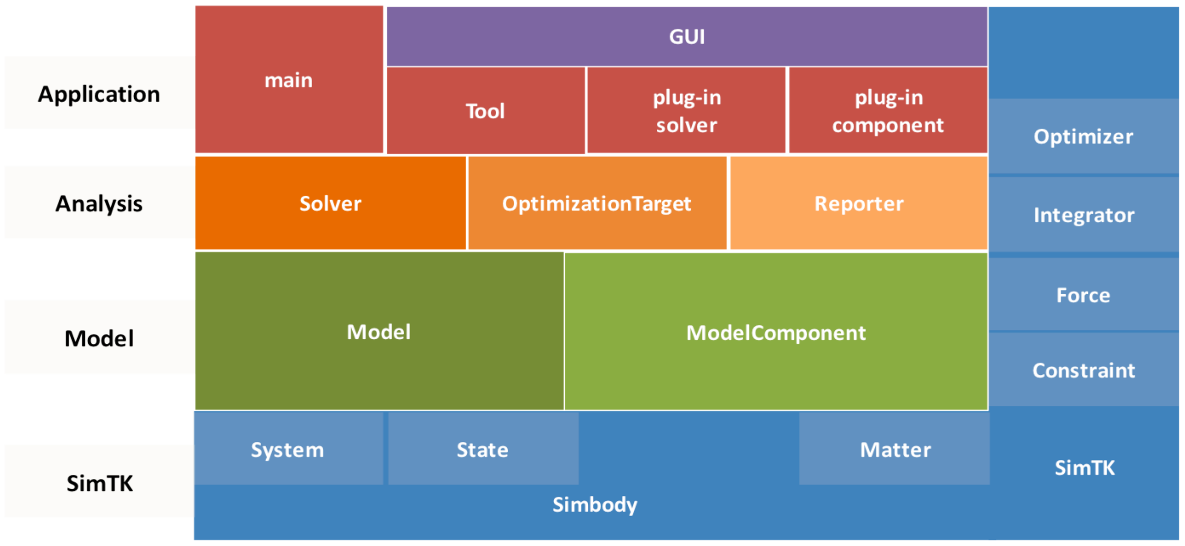
OpenSim provides several mechanisms for extending its existing capabilities either by adding new model elements, computing new quantities, or computing existing quantities in a new way. For example, you may want to model the drag acting on bodies moving through a fluid, which OpenSim does not provide. Another example is being able to extract the linear and angular momentum of the model during a simulation. In order to extend to OpenSim, it is important to know what functionality exists and to have a sense of where to add new functionality.

11.2 Organization of OpenSim

OpenSim is built on the computational and simulation core provided by SimTK. This includes low-level, efficient math and matrix algebra libraries such as LAPACK as well as the infrastructure for defining a dynamical system and its state. One can think of the system as the set of differential equations and the state comprised of its variables.

Empowering the computational layer is Simbody™, an efficient multibody dynamics solver, which provides an extensible multibody system and state. The OpenSim modeling layer maps biomechanical structures (bones, muscles, tendons, etc.) into bodies and forces so that the dynamics of the system can be computed by Simbody.

The three interface layers of OpenSim built on SimTK:

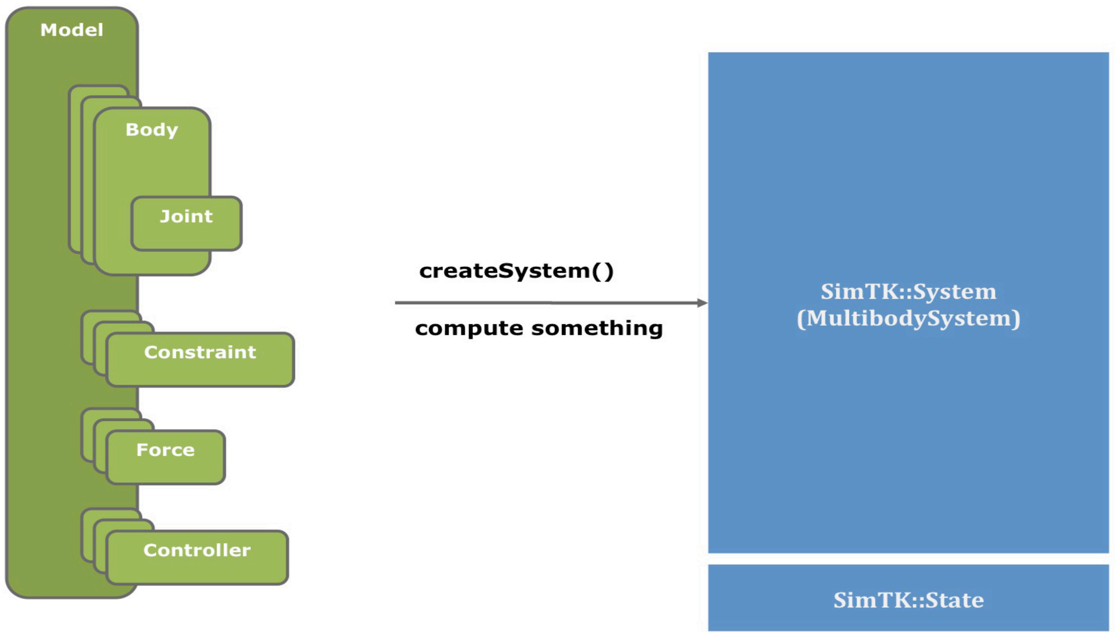


OpenSim is essentially a set of modeling libraries for building complex actuators (e.g. muscles) and other forces (e.g. contact) and enabling the motion (kinematics) of highly articulated bodies (bones). Actuators can then be controlled by model controllers (e.g. Computed Muscle Control) to estimate the neural control and muscle forces required to reproduce human movement. An analysis layer is equipped with solvers and optimization resources for performing calculations with the model and to report results. At the highest level these blocks are assembled into specialized applications (ik.exe, forward.exe, analyze.exe) to simulate and analyze model movement and internal dynamics. The OpenSim application is a Java based program that calls Tools, Models, and underlying computations in SimTK to provide an interactive graphical user-interface (GUI).

11.3 OpenSim Model and ModelComponents

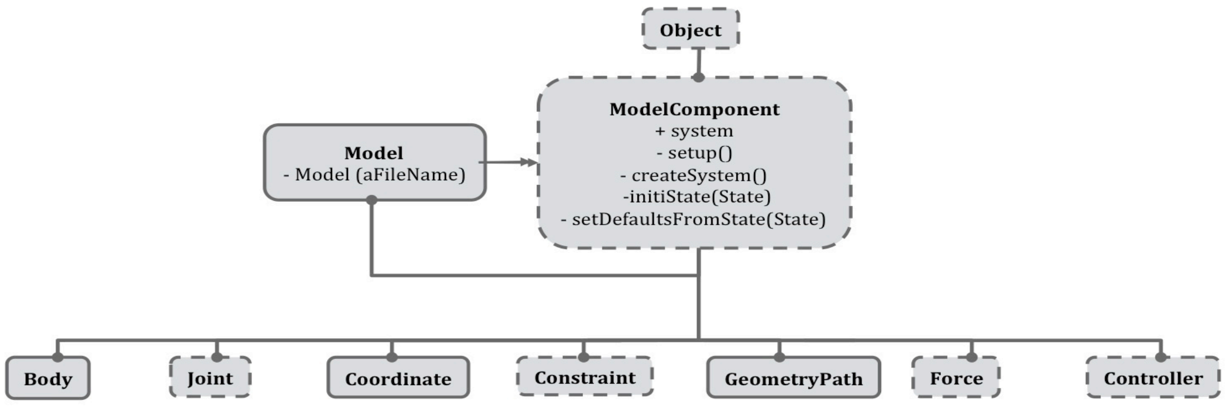
The job of an OpenSim::Model is to organize (hierarchically) the pieces (components) of a musculoskeletal system and to create a representative computational (mathematical) system that can be solved accurately and efficiently using Simbody and the flat SimTK::System framework.

Organizational Context vs. Computation



By separating the contextual organization of a model from its computational representation, OpenSim can exploit the conceptual benefits of hierarchically organized models and software without sacrificing computational efficiency. One can then think of the system as the set of system equations while the state is a coherent set of system variable values that satisfies the system equations. Model components know about the parts they add to the multibody system (for example, another rigid body, a force, or a constraint) and are free to mix and match. For example, a Coordinate component knows how to access its underlying degree-of-freedom value, velocity and even its acceleration, given the system has been "solved" for accelerations. A Coordinate also adds different constraints to the underlying system, in the case that Coordinate is locked or if its motion is prescribed. It provides context to organize locking constraints with the Coordinates being locked, but computationally it is just another constraint equation. The Coordinate therefore acts to manage the bookkeeping (which DOF, constraint, etc.) and provide an interface that has context.

OpenSim Model and its ModelComponents



All model components in OpenSim have a similar responsibility to create their underlying system representation (createSystem()). A setup() method ensures that a model is appropriately defined (for example, a Body is being connected to a parent that exists) before creating the system. Two additional methods allow the ModelComponent to initialize the state of the system (from default properties) and also to hold the existing state in the ModelComponent's defaults. For example, a Coordinate's default may indicate that it should be locked, in which case its initState would set the state of its underlying constraint as "enabled". Similarly, after performing an analysis to find the coordinates to satisfy a static pose, calling setDefaultsFromState(state) will update the Coordinate's default values for the coordinate value from the desired state. Next time the model is initialized, it will be in the desired pose.

11.4 OpenSim Application Programming Interface (API)

In order to build custom components, it is necessary to have a general understanding of which objects (classes) are responsible for what actions/behaviors. The functions (methods) that OpenSim's public classes provide (that other applications/programs can call) define its Application Programming Interface or API.

We have already seen four methods that a model component must implement to behave as a ModelComponent in OpenSim. This defines the ModelComponent interface. Each type of ModelComponent, in turn, specifies additional methods in order to satisfy that type of component. For example, a Force in OpenSim must implement a computeForce() method (in addition to the ModelComponent methods), a Controller must implement computeControls(), etc. The set of all Classes and their interfaces defines the OpenSim API.

The OpenSim API is undergoing rapid development and improvement. We therefore rely on Doxygen to automatically generate html documentation of the latest source, which describe the classes that are available and the accessible methods. The Doxygen pages can be viewed using a web browser and are available with your OpenSim installation in: <OpenSim_Install_Dir>/sdk/doc/index.html.

11.5 What is an OpenSim plug-in?

When creating a new component (like a force, controller) or a new analysis, you may want to include it in an existing model, run it with existing tools, and/or share your contribution with colleagues. An OpenSim plug-in is a way of packaging of your code in a dynamically linked library so that an existing OpenSim application can recognize it, load it, and make your code "runnable". For an example of creating an analysis as a plug-in please see <OpenSim_Install_Dir>/sdk/examples/plugin.

11.6 What is an OpenSim "main" program?

A main program in C/C++ results in a standalone executable that you can run from a command prompt or by double clicking in Windows. All C/C++ programs have a main() function, which can be as simple as printing "Hello World" or it can invoke several libraries to produce complex applications, like Word and Excel. By including the OpenSim libraries, your main program can call the OpenSim API, and you may also include any other (C++) libraries that provide additional computational and/or visualization resources. Main programs are extremely flexible, but they are particularly useful for streamlining/automating processes independent of the GUI. For example, ik.exe, id.exe, and cmc.exe (available with the OpenSim distribution) are main programs that take setup files and perform tasks related to the OpenSim workflow. Alternatively, users have created their own main programs to systematically scale strengths of all muscles in a model, run forward simulations with their own controllers, perform design optimizations, etc. An advantage of a main program (compared to a plug-in) is that any classes you define in the project are immediately useable by your program. This can make prototyping and testing of your new component or analysis faster and easier without having to wrap, load, and call your plug-in from the GUI.

11.7 OpenSim Developer's Guide

The developer's guide provides a step-by-step example of calling the OpenSim API to build a model, including muscles and contact forces, and to perform a simulation in a main program. Please refer to the OpenSim Developer's Guide for more details.

11.8 Command Line Utilities

All of the OpenSim Tools are available as command-line utilities that take as input the same setup (or settings) file loaded into or saved from the OpenSim GUI application. For example, to perform Inverse Kinematics from the command line (the Command Prompt in Windows) one can execute the following command:

```
ik –S arm26_Setup_InverseKinematics.xml
```

Similarly, this command line arguments work for CMC or any other tool, with the complete set of command line executables available in <OpenSim_Install_Dir>/bin. In addition to the –***S****etup* option, there are –***H****elp* , -***P****rint****S****etup* and –***P***roperty***I***nfo options. Help provides this list of options. Print Setup prints a default setup file for that Tool with all available properties (XML tags) for Tool settings.

The –***P***roperty***I***nfo option can be a very handy resource to obtain information about existing settings for Tools and ModelComponents including the XML tags needed in the model and/or setup file. This is the same information listed in the "Available Objects..." panel under the Help menu in the OpenSim GUI. Executing ik –PI lists all the available classes (components, analyses, utilities and tools) available in OpenSim. For more information about a particular object, such as adding a point constraint to the model, executing

```
ik -PI PointConstraint
```

yields:

```
PROPERTIES FOR PointConstraint (5)
1. isDisabled
2. body_1
3. body_2
4. location_body_1
5. location_body_2
```

The information returned lists the properties for defining a point constraint in OpenSim.

11.9 MATLAB Utilities for Data Import

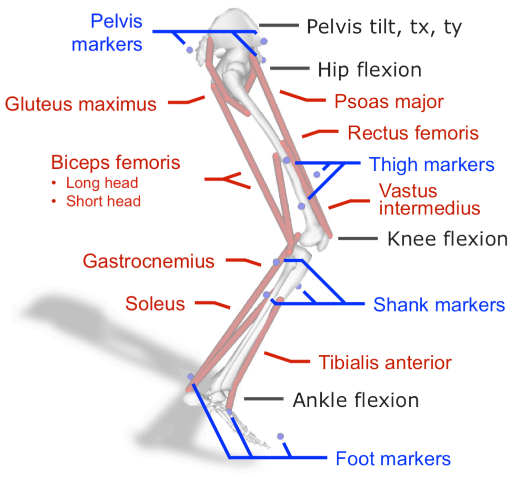
There are several MATLAB scripts for reading .trc, .c3d, .mot, and .sto files into MATLAB and writing out the data file formats required by OpenSim. Scripts provided by the Neuromuscular Biomechanics Lab at Stanford are available on the OpenSim Utilities project on SimTK.org: https://simtk.org/home/opensim-utils. Additional utilities by OpenSim users are posted on SimTK.org and can be found using the search tool on SimTK.org.

12 Example: Forward Simulation with RRA and CMC

Creating a Muscle-Driven Simulation of the Stance Phase of Gait

A. Opening the Model and Viewing the Data

1. Load the *leg6dof9muscles.osim* model from the examples directory (e.g., *examples \Leg6Dof9Musc*)
2. Preview the kinematics and ground-reaction forces with the model
   a. Load the motion file *leg69_IK_stance.mot* and hit play.
   b. Under File, choose Preview Motion Data and select *leg69_stance_grf.mot*
   c. Sync the two motions by selecting both motions: hold the control key, right click to sync motions, and hit play.



Leg6Dof9Musc.osim: The model we'll use in this exercise is of a right leg and pelvis with 6 coordinates and 9 muscles.

B. Use Inverse Dynamics to determine the amount of residual force that is required for the model's dynamics to be consistent with applied ground reaction forces:

1. Right click the leg69 model and make it the current model.
2. Launch the Inverse Dynamics tool.
3. Under Input select Motion→Loaded motion→ ik trial.
4. Check the box to filter kinematics at 6Hz.
5. Specify the time range as 0.5s to 1.5s, the period in which ground reaction forces are defined.
6. Specify an output directory (e.g. *<YourWorkingDir>\Stance\ID_Results*)
7. Select the External Loads tab and check the External Loads box.
8. Edit the External Loads settings by clicking the pencil icon.
   a. Select the *leg69_stance_grf.mot* as the Force data file. This file describes the force applied at the foot's center of pressure.
   b. Select *leg69_IK_stance.mot* as the Kinematics for external loads.
   c. Select Filter kinematics and specify 6Hz.
   d. Forces listed in the motion file are added as individual forces by hitting the Add button.
      i. Provide a name (e.g., "Right_GRF")
      ii. Applied to body (e.g., calcn_r)
      iii. Check Applies Force and select Point Force
      iv. Force Columns select "ground_force_vx", y & z selected automatically

v. Point Columns select “ground_force_px”
vi. The GRF free moment is a torque, so check “Applies Torque”
vii. Torque Columns, scroll down and select “ground_torque_x”
viii. Both the GRF and CoP are expressed in the ground (lab) frame
ix. Click OK

e. Hit Save and enter a filename for the External Force (e.g., *leg69_right_GRF.xml*)

9. Save settings (e.g. *leg69_Setup_ID_stance.xml*), then hit Run.
10. Plot the forces acting on the pelvis (e.g., pelvis_tilt_moment, pelvis_tx_force, etc.) and the net joint moments for the hip, knee and ankle from *inverse_dynamics.sto*.
Which is the largest residual? What accounts for the large residual forces?

C. Use the RRA tool to reduce residuals. In other words, adjust the model to compensate for model inconsistency with the applied GRFs.

1. Launch the Reduce Residuals tool.
2. Specify Desired Kinematics, which is the IK motion *leg69_IK_stance.mot*.
3. Check the box to filter kinematics at 6Hz.
4. Specify the tracking tasks for RRA. Specify the task file provided in *Stance/Reference/ leg69_Tracking_Tasks.xml*. This file specifies the coordinates to be tracked and the corresponding weights. Use an XML editor (e.g., Notepad++) to view the tasks.
5. Specify Actuator control constraints file *leg69_residuals_motors_control_limits.xml*. These constraints define the maximum and minimum control limits for all actuators.
6. Check “Adjust model”. Click on the folder icon, make sure you are in the Stance folder, and specify a new model name (e.g. *leg6dof9musc_adjusted_COM_pelvis.osim*). Click Save.
7. Specify the Body COM to adjust as pelvis. The center-of-mass of this body will be adjusted to reduce residual. Typically we choose the segment incorporating the torso.
8. Specify the time range as 0.5s to 1.5s.
9. Specify an output directory (e.g. *Stance\RRA*)
10. Select the Actuators and External Loads tab and choose “<u>Replace</u> model’s force set” to replace the model’s muscles with residual and joint motor actuators since we are creating a torque-driven simulation. Click Edit, then click Add. Then click the folder button next to the red text box. Select *leg69_RRA_residuals_motors.xml*. Click OK.
11. Check the External Loads box and specify the file you created for Inverse Dynamics (e.g. *leg69_right_GRF.xml*).
12. Save your settings to an RRA setup file (e.g., *leg69_Setup_RRA_stance.xml*).
13. Hit Run.
Why does the model “float” down and up?
14. With the original model, preview the model motion with the GRF again.
Are all of the forces being applied correctly? Thinking about this model and motion, what time range of the gait should you restrict your RRA analysis?
15. Repeat RRA with the original model over the new time interval (and save the new settings). Close the RRA Tool.
Does the model still “float” up or down? If so, what else could be causing this?

16. Open up the messages window and locate the recommendation from the last run of RRA (e.g., "dmass = 44.037"). Note, the units are in kilograms (kg).
    What is the recommended mass adjustment? Why would the mass adjustment be so large?
17. Edit the adjusted model to make the recommended mass adjustments by RRA to the pelvis body.
    a. To make mass adjustments open the Property Editor (Window→Properties). Navigate to the pelvis Body of the adjusted model. Edit the mass of the pelvis and save the model. Alternatively, use an XML editor to edit the model file.
    b. Rename the model in the Navigator window (e.g. leg6dof9musc_adjusted) by right clicking on the current model name and selecting Rename.
18. Re-run RRA with the adjusted model. Be sure to close and re-open the RRA Tool so you are working on the adjusted model.
    Is the mass adjustment suggested by RRA smaller than before?
19. Plot the RRA residual actuator forces (i.e., MZ, FX, FY) from *leg6dof9musc_Actuation_Force.sto*.
    How do they compare to the forces acting on the pelvis from your ID results?
20. Plot tracking kinematics outputted by RRA from *leg6dof9musc_Kinematics_q.sto* vs. kinematics from IK. You can also get the tracking error values directly from the file *leg6dof9musc_pErr.sto*.
    Which coordinates have large tracking errors?
21. Increase the tracking task weights for coordinates that show poor tracking (via Edit → File). Decrease the tracking weight for coordinates that are within a degree, since the optimizer can use this to reduce residuals. Note: The plotter will display most angles in radians. 1 degree = 0.017 radians. See the handout for more information about getting good results from RRA.
22. Re-run RRA. Repeat steps 19 and 20 to check the residual forces and tracking errors.
    Are the residual forces and tracking improved?

D. Use the Computed Muscle Control (CMC) Tool to determine the muscle excitations, activations and forces that generate a forward dynamics simulation of the stance-phase of gait.

1. Load the final adjusted model from RRA
2. Consider the residual and motor actuators necessary for CMC (e.g., modify *leg69_RRA_residuals_motors.xml* using Edit→File or an XML editor)
    a. With muscles present, reduce the optimal force of joint motors to 1 so that they are penalized during CMC and muscles are favored to generate joint moments.
    b. Save the edited the actuators as a new file (e.g., *leg69_CMC_residuals_motors.xml*).
3. Launch the CMC Tool.
4. Specify Desired Kinematics as output from RRA (e.g., *leg6dof9musc_Kinematics_q.sto*). Note, no filtering is required as the kinematics are smooth since they result from a simulation.

5. Apply tracking tasks. Use the same tasks file as for RRA.
6. Include limits on muscle actuators by using *leg69_muscles_residuals_motor_control_limits.xml* as Actuator constraints. Open the file in an XML editor to compare with the Actuator constraints used for RRA.
   What is the difference between the actuator constraints file used in CMC and the file for RRA?
7. Define time range for the stance simulation (use the same range you determined from Step 14 in RRA).
8. Specify the output directory (e.g., *Stance\CMC*).
9. Under the Actuators and External Loads tab, select "Append to the model's force set" to include joint motor and residual actuators in addition to existing muscles in the model. Use the Edit button to specify *leg69_CMC_residuals_motors.xml* to be appended.
10. Specify the external loads (same as for RRA).
11. Save your settings to a file.
12. Run CMC. If CMC does not execute completely, review the tips and tricks in the handout for help with troubleshooting.
13. Plot the muscle activation patterns from the states file, *leg6dof9musc_added_mass_states.sto*. Tip: Using the pattern filter in the plotter selection tool, type "activation" to quickly select the activations.
    Are the simulated activations close to what you expected?
14. Check the quality of the simulation by examining the tracking errors, the residual forces, and the motor moments (i.e., reserves). The motor moments are in *leg6dof9musc_added_mass_Actuation_force.sto*
    How do the kinematics compare to the IK solution for stance? How big are the residuals forces from CMC? How do they compare to residuals from RRA? Are the motor moments (i.e., reserves) at the hip, knee and ankle significant? Are the residuals below 2% of body-weight?
15. If time permits, run a forward simulation with the controls from CMC and initial states from CMC. Running Forward Dynamics after CMC is a method of verifying that the controls from CMC generate a forward simulation consistent with the observed kinematics and applied GRFs.
    How do the kinematics from Forward compare to the original IK solution?

13 Example: Model Editing

13.1 Purpose

In this example, you will add a body and an actuator to an existing OpenSim model by editing its xml file. You can find the model file (arm26.osim) in the examples folder under your OpenSim installation directory.

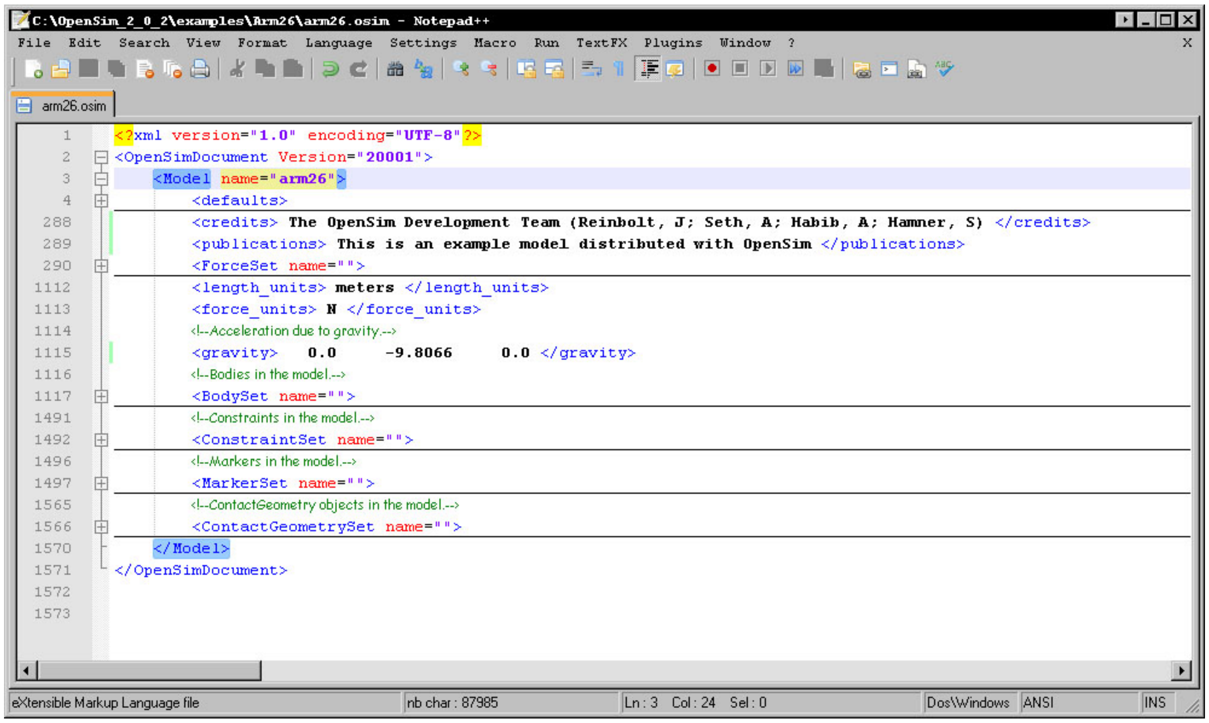
You will need to download an XML editor to work through this example. Some free options include:

- notepad++ (http://notepad-plus.sourceforge.net/uk/site.htm)
- XMLMarker (http://symbolclick.com/download.htm)

13.2 Connecting an Additional Segment to the Model

- **Open Model File in XML**. Use an XML editor (e.g., Notepad++) to open the OpenSim model file (e.g., *arm26.osim*). When collapsed to the 3rd level (e.g., **Alt+3** in Notepad+), your model file should look like the figure below.

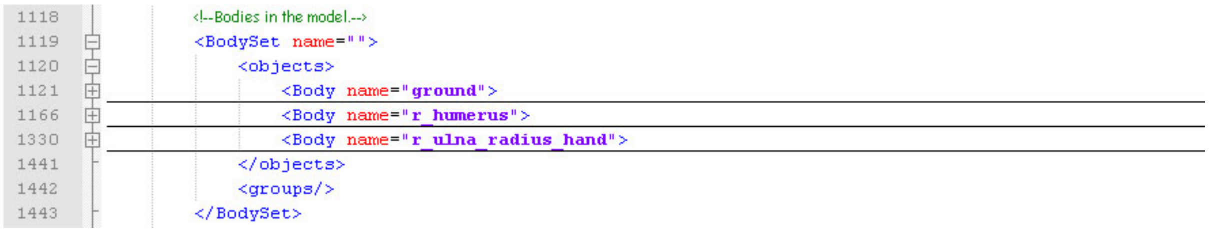
- **Explore the Model.** The **Model** tag has five main sets named, **ForceSet, BodySet**, **ConstraintSet**, **MarkerSet**, and **ContactGeometrySet**.


C:\OpenSim_2_0_2\examples\Arm26\arm26.osim - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
arm26.osim

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSimDocument Version="20001">
    <Model name="arm26">
        <defaults>
        <credits> The OpenSim Development Team (Reinbolt, J; Seth, A; Habib, A; Hamner, S) </credits>
        <publications> This is an example model distributed with OpenSim </publications>
        <ForceSet name="">
        <length_units> meters </length_units>
        <force_units> N </force_units>
        <!--Acceleration due to gravity.-->
        <gravity>    0.0      -9.8066      0.0 </gravity>
        <!--Bodies in the model.-->
        <BodySet name="">
        <!--Constraints in the model.-->
        <ConstraintSet name="">
        <!--Markers in the model.-->
        <MarkerSet name="">
        <!--ContactGeometry objects in the model.-->
        <ContactGeometrySet name="">
    </Model>
</OpenSimDocument>
```

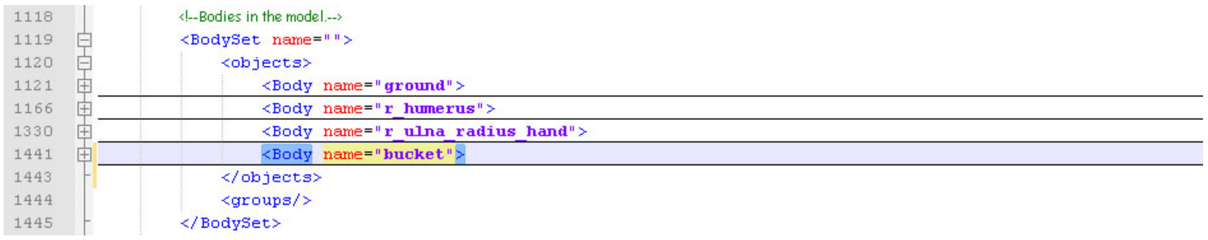
eXtensible Markup Language file
nb char : 87985
Ln : 3 Col : 24 Sel : 0
Dos\Windows
ANSI
INS

- **Explore the Body Set.** The **BodySet** tag has three **Body objects** named **ground**, **r_humerus**, and **r_ulna_radius_hand**.



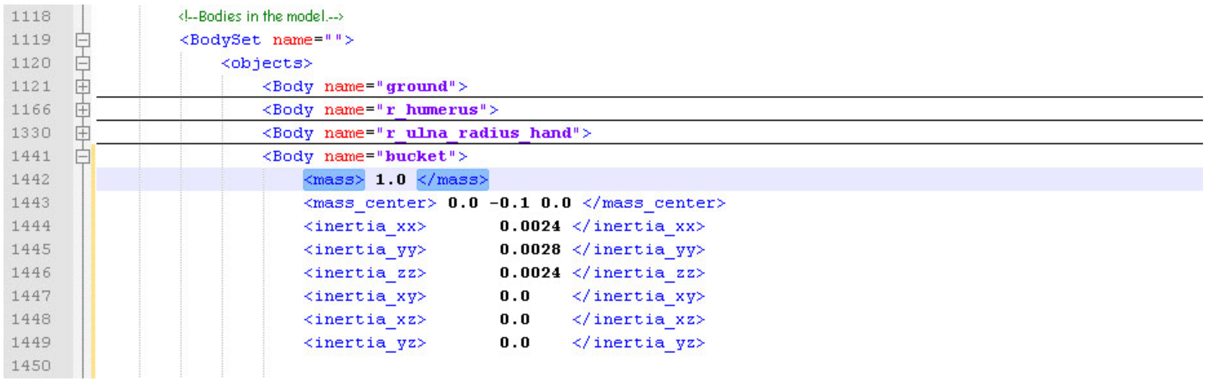
```
<!--Bodies in the model.-->
<BodySet name="">
    <objects>
        <Body name="ground">
        <Body name="r_humerus">
        <Body name="r_ulna_radius_hand">
    </objects>
    <groups/>
</BodySet>
```

- **Add New Body.** Add a new **Body** named **bucket** immediately below the **Body** named **r_ulna_radius_hand**.



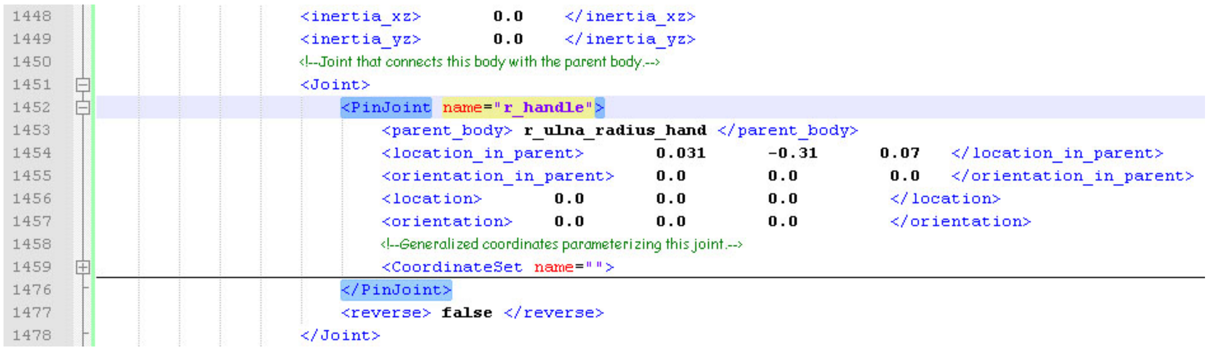
```
<!--Bodies in the model.-->
<BodySet name="">
    <objects>
        <Body name="ground">
        <Body name="r_humerus">
        <Body name="r_ulna_radius_hand">
        <Body name="bucket">
    </objects>
    <groups/>
</BodySet>
```

- **Specify Mass Properties.** Add tags and enter values for the **mass**, **mass_center**, **inertia_xx**, **inertia_yy**, **inertia_zz, interia_xy**, **inertia_xz, and inertia_yz** for the bucket as seen below:



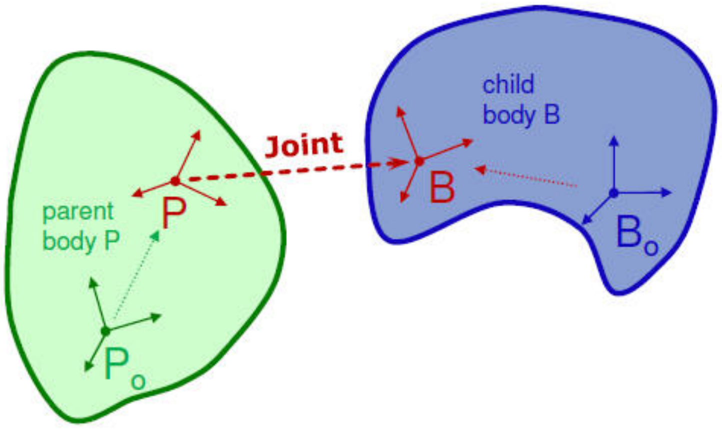
```
<!--Bodies in the model.-->
<BodySet name="">
    <objects>
        <Body name="ground">
        <Body name="r_humerus">
        <Body name="r_ulna_radius_hand">
        <Body name="bucket">
            <mass> 1.0 </mass>
            <mass_center> 0.0 -0.1 0.0 </mass_center>
            <inertia_xx>      0.0024 </inertia_xx>
            <inertia_yy>      0.0028 </inertia_yy>
            <inertia_zz>      0.0024 </inertia_zz>
            <inertia_xy>      0.0    </inertia_xy>
            <inertia_xz>      0.0    </inertia_xz>
            <inertia_yz>      0.0    </inertia_yz>
```

- **Specify Joint.** Add tags and names for the **PinJoint** and **parent_body**, and tags and values for **location_in_parent**, **orientation_in_parent, location**, and **orientation** as seen below:



```
                <inertia_xz>        0.0     </inertia_xz>
                <inertia_yz>        0.0     </inertia_yz>
                <!--Joint that connects this body with the parent body.-->
                <Joint>
                    <PinJoint name="r_handle">
                        <parent_body> r_ulna_radius_hand </parent_body>
                        <location_in_parent>        0.031       -0.31       0.07    </location_in_parent>
                        <orientation_in_parent>     0.0         0.0          0.0    </orientation_in_parent>
                        <location>          0.0         0.0         0.0         </location>
                        <orientation>       0.0         0.0         0.0         </orientation>
                        <!--Generalized coordinates parameterizing this joint.-->
                        <CoordinateSet name="">
                    </PinJoint>
                    <reverse> false </reverse>
                </Joint>
```

The **location_in_parent** and **orientation_in_parent** tags define the position and orientation of the joint with respect to the parent body origin. In the above example, the vector from the parent body *r_ulna_radius_hand* to the joint *r_handle* is [0.031 -0.031 0.07] and the orientation of the joint (in a zero pose) is exactly the same as the orientation of the parent body (in a zero pose). The **location_in_parent** and **orientation_in_parent** vectors should be given in the coordinate frame of the parent body. In most cases, the **location_in_parent** and **orientation_in_parent** tags are all that's required to specify the new joint (in which case, the child body origin, e.g., *bucket*, would be located at the joint center of *r_handle*). Additionally, the local location and orientation offset of the joint center with respect to the child body origin can be specified using the **location** and **orientation** tags. The above description of joints in OpenSim is illustrated below:

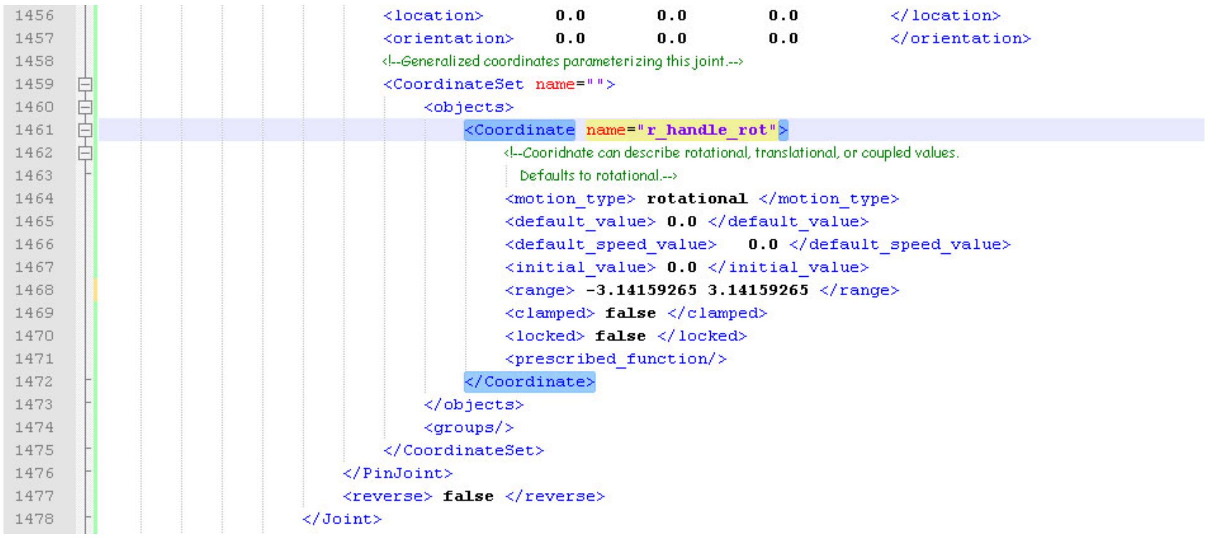


B specified by joint `location` and `orientation`

P specified by joint `locationInParent` and `orientationInParent`

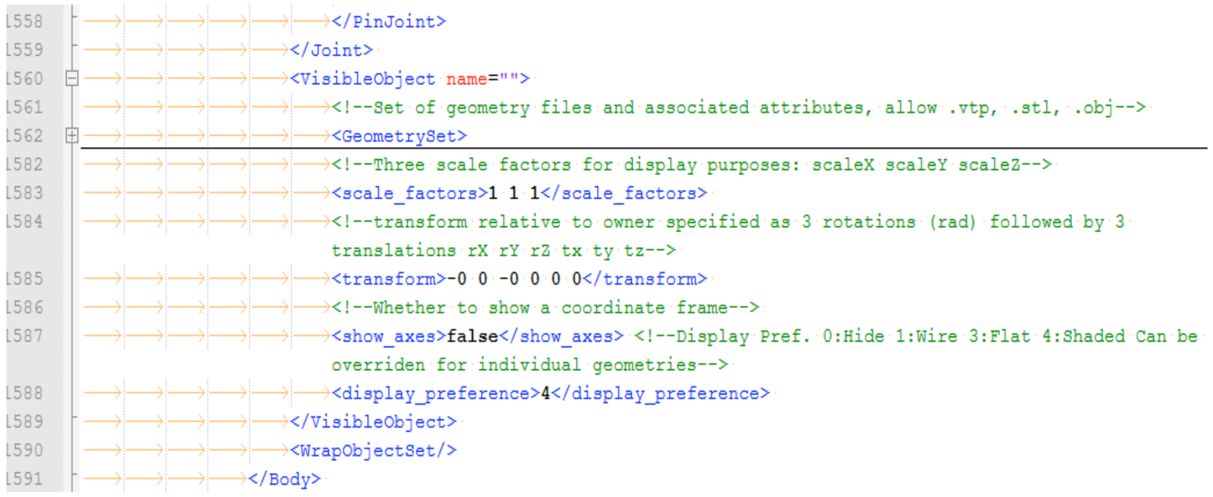
Joint coordinates specify the kinematics of B relative to P

- **Specify Generalized Coordinate.** Add a tag and name for the **Coordinate**, and add tags and values for **motion type, default_value, default_speed_value, initial_value, range, clamped**, and **locked** as seen below:



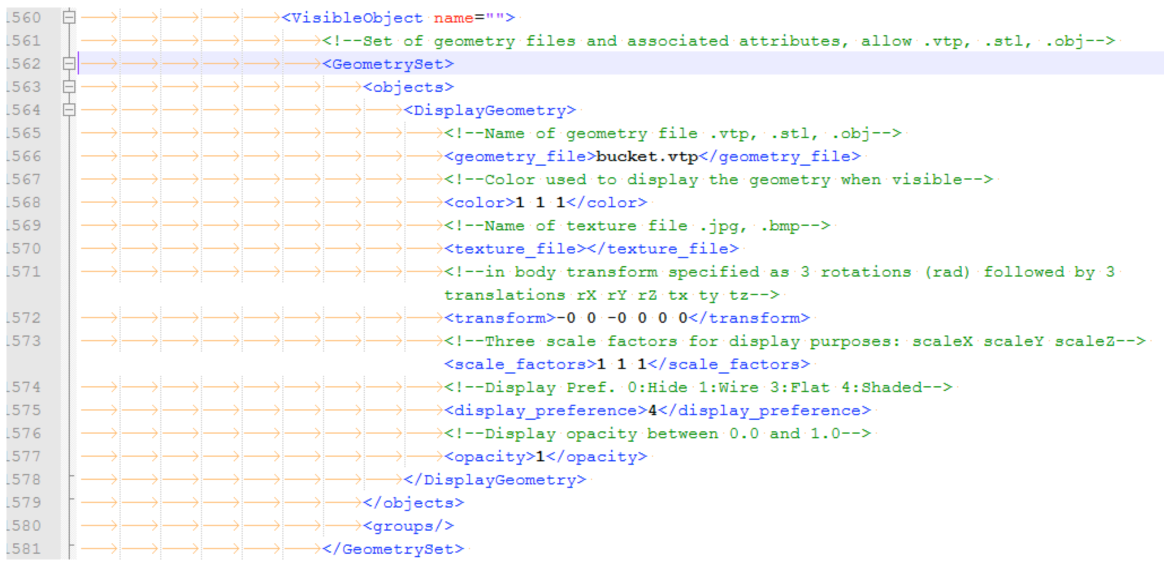
```
				<location>      0.0      0.0      0.0      </location>
				<orientation>   0.0      0.0      0.0      </orientation>
				<!--Generalized coordinates parameterizing this joint.-->
				<CoordinateSet name="">
					<objects>
						<Coordinate name="r_handle_rot">
							<!--Cooridnate can describe rotational, translational, or coupled values.
							    Defaults to rotational.-->
							<motion_type> rotational </motion_type>
							<default_value> 0.0 </default_value>
							<default_speed_value>   0.0 </default_speed_value>
							<initial_value> 0.0 </initial_value>
							<range> -3.14159265 3.14159265 </range>
							<clamped> false </clamped>
							<locked> false </locked>
							<prescribed_function/>
						</Coordinate>
					</objects>
					<groups/>
				</CoordinateSet>
			</PinJoint>
			<reverse> false </reverse>
		</Joint>
```

- **Specify Geometry File.** Add a tag for VisibleObject and add the appropriate tags for **GeometrySet** and other properties as seen below:



```
					</PinJoint>
				</Joint>
				<VisibleObject name="">
					<!--Set of geometry files and associated attributes, allow .vtp, .stl, .obj-->
					<GeometrySet>
					<!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
					<scale_factors>1 1 1</scale_factors>
					<!--transform relative to owner specified as 3 rotations (rad) followed by 3
					translations rX rY rZ tx ty tz-->
					<transform>-0 0 -0 0 0 0</transform>
					<!--Whether to show a coordinate frame-->
					<show_axes>false</show_axes> <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded Can be
					overriden for individual geometries-->
					<display_preference>4</display_preference>
				</VisibleObject>
				<WrapObjectSet/>
			</Body>
```

- **Specify DisplayGeometry** including **Geometry_File** in the **GeometrySet**

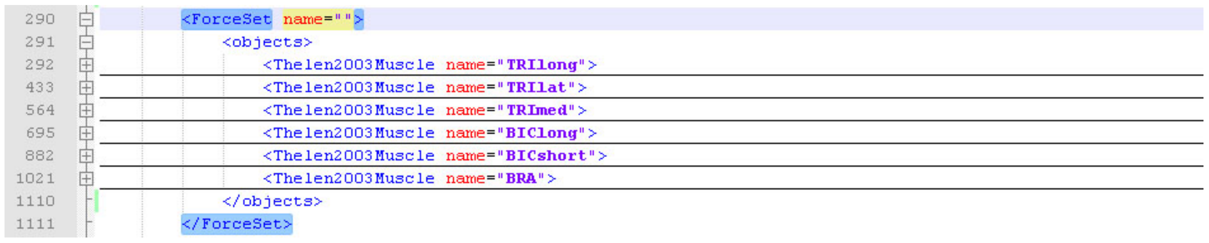


```
<VisibleObject name="">
    <!--Set of geometry files and associated attributes, allow .vtp, .stl, .obj-->
    <GeometrySet>
        <objects>
            <DisplayGeometry>
                <!--Name of geometry file .vtp, .stl, .obj-->
                <geometry_file>bucket.vtp</geometry_file>
                <!--Color used to display the geometry when visible-->
                <color>1 1 1</color>
                <!--Name of texture file .jpg, .bmp-->
                <texture_file></texture_file>
                <!--in body transform specified as 3 rotations (rad) followed by 3 translations rX rY rZ tx ty tz-->
                <transform>-0 0 -0 0 0 0</transform>
                <!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
                <scale_factors>1 1 1</scale_factors>
                <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded-->
                <display_preference>4</display_preference>
                <!--Display opacity between 0.0 and 1.0-->
                <opacity>1</opacity>
            </DisplayGeometry>
        </objects>
        <groups/>
    </GeometrySet>
```

- **Save Model File.** From the XML editor, save the OpenSim model file (e.g., *arm26_with_bucket.osim*). Now open your new model in OpenSim to see the bucket!

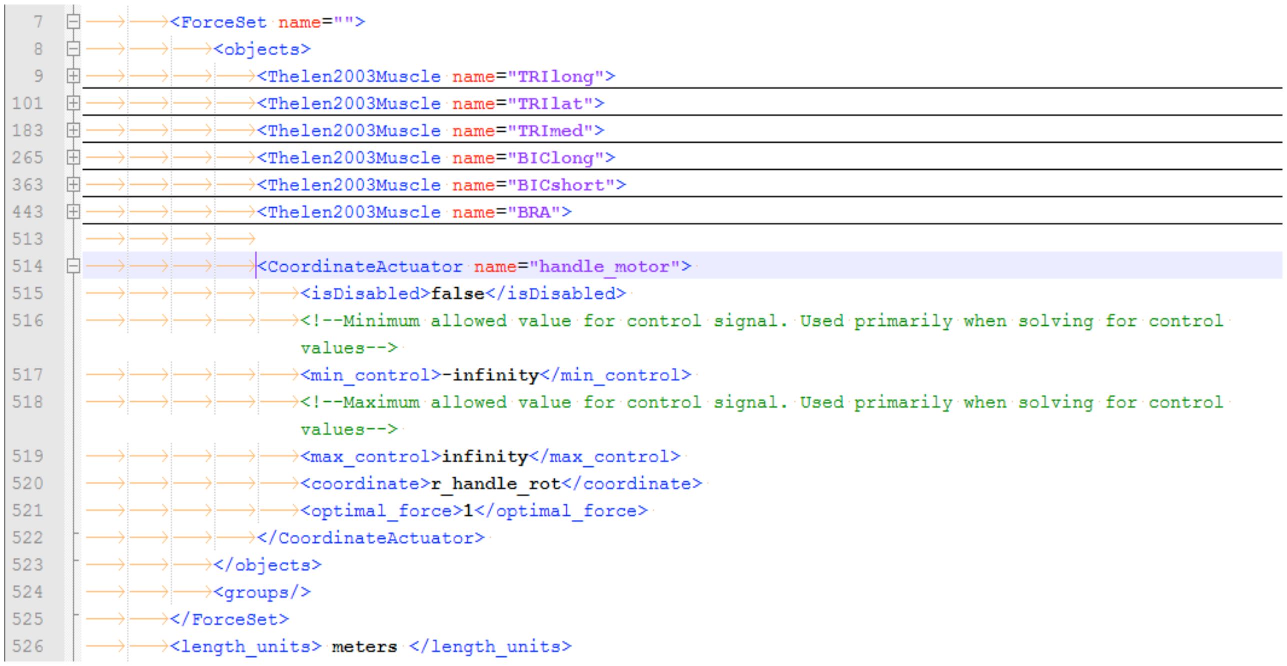
13.3 Adding an Additional Actuator

- **Explore the ForceSet.** The **ForceSet** tag has six **Thelen2003Muscle objects** named **TRIlong**, **TRIlat**, **TRImed**, **BIClong**, **BICshort**, and **BRA**.



```
<ForceSet name="">
    <objects>
        <Thelen2003Muscle name="TRIlong">
        <Thelen2003Muscle name="TRIlat">
        <Thelen2003Muscle name="TRImed">
        <Thelen2003Muscle name="BIClong">
        <Thelen2003Muscle name="BICshort">
        <Thelen2003Muscle name="BRA">
    </objects>
</ForceSet>
```

- **Add New Actuator.** Add a **CoordinateActuator object** named **r_handle_rot_force** immediately below the **Thelen2003Muscle** named **BRA**. Associate this **CoordinateActuator** with the **r_handle_rot coordinate** and specify an **optimal_force** of 1000.



```
        <ForceSet name="">
            <objects>
                <Thelen2003Muscle name="TRIlong">
                <Thelen2003Muscle name="TRIlat">
                <Thelen2003Muscle name="TRImed">
                <Thelen2003Muscle name="BIClong">
                <Thelen2003Muscle name="BICshort">
                <Thelen2003Muscle name="BRA">

                <CoordinateActuator name="handle_motor">
                    <isDisabled>false</isDisabled>
                    <!--Minimum allowed value for control signal. Used primarily when solving for control
                    values-->
                    <min_control>-infinity</min_control>
                    <!--Maximum allowed value for control signal. Used primarily when solving for control
                    values-->
                    <max_control>infinity</max_control>
                    <coordinate>r_handle_rot</coordinate>
                    <optimal_force>1</optimal_force>
                </CoordinateActuator>
            </objects>
            <groups/>
        </ForceSet>
        <length_units> meters </length_units>
```

- **Save Model File.** From the XML editor, save the OpenSim model file (e.g., *arm26_with_bucket.osim*).