Overview of OpenSim Workflows

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. Contents include:

The OpenSim Model

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

Additional information is also available in the section on OpenSim Models. A large repository of existing models is available (see curated list or do a search on SimTK). These include models of the lower extremity, head and neck, spine, wrist, and many other musculoskeletal regions for both humans and other animals. We encourage you to contribute your own models to SimTK to enable other researchers to build on your work and further advance the field.

A model consists of different components. For example, in a model used for the simulation of human walking, 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).

Simulation Pipelines (Workflows)

Different simulation pipelines have been developed to answer different questions. Our webinar on "Which Simulation Pipeline Should I Use? An Overview of Common Workflows" can help you select a suitable simulation pipeline. Our paper on Best practices for verification and validation of musculoskeletal models and simulations of movement also has valuable information about how to design a study.

You can also use the questions below to guide you in selecting a suitable simulation pipeline. The section below on Examples of Choosing a Simulation Pipeline also provides useful tips.

1. Do you:

    a. Have data about movement (e.g., from optical motion capture or inertial measurement units) you have collected from subjects AND

    b. Want to estimate joint angles and coordinates, joint moments, joint torques, muscle forces, muscle activity, musculotendon dynamics, metabolic cost or other values that are a function of the states of the model?

    If you answered "yes" to (a) and (b), you may have an Inverse Problem. See the Inverse Problem simulation options below.

2. Do you want to generate new movement given a set of prescribed controls, such as muscle excitations or joint torques? Or do you want to generate a movement to achieve a specific objective function (e.g., maximizing jump height)?

    If yes, you may have a Forward Problem. See the Forward Problem simulation options below.

3. Do you want to estimate some change in kinematics that occurs due to a perturbation (e.g., changing segments masses or simulating the effects of a surgery)?

    If yes, you may have a Forward Problem. See the Forward Problem simulation options below.

4. Do you have a problem that does not fit any of the previously mentioned scenarios?

        If yes, you can take advantage of OpenSim's extensibility to develop a novel pipeline. Some examples include:

        - *A MATLAB pipeline to perform static optimization with muscle synergies (publication, webinar)*

- *Combining OpenSim with finite element analysis*

    - *Combine OpenSim and FEBio (finite element analysis software) to study contact loads during walking (publication)*

    - *Interfacing musculoskeletal and finite element models to study bone structure and adaptation (webinar + links to publications)*

    - *Combining multi-scale data and finite element modeling of the knee (webinar + links to publications)*

- *Probabilistic analyses of simulations*

    - *A probabilistic tool to quantify the effects of population variability and model uncertainty (webinar + links to publication and tool)*

    - *Enabling stochastic simulations of movement with high throughput computing on the Open Science Grid (webinar + links to publications and sample files)*

Common Pre-Processing Steps

For several of the simulation pipelines discussed above, the first steps are to import your experimental data and scale your model. For cases where the simulation pipeline relies on experimental data, you can read about importing various types of data below. For cases where you are simulating and analyzing the movement of a specific subject, you can read more about scaling a model below.

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 (EMG)

See Preparing Your Data for detailed information about preparing and importing this kind of experimental data.

As of OpenSim 4.1, we have also begun to add capabilities to analyze data from Inertial Measurement Units (IMUs). Read more about analyzing IMU Data here:

- OpenSense - Kinematics with IMU Data

If you have imaging data, such as MRI, which you would like to use to create subject-specific musculoskeletal models, you can try one of these tools, which are compatible with OpenSim, created by others in the community:

- Musculoskeletal Atlas Project (MAP) client
- NMSBuilder

Scaling

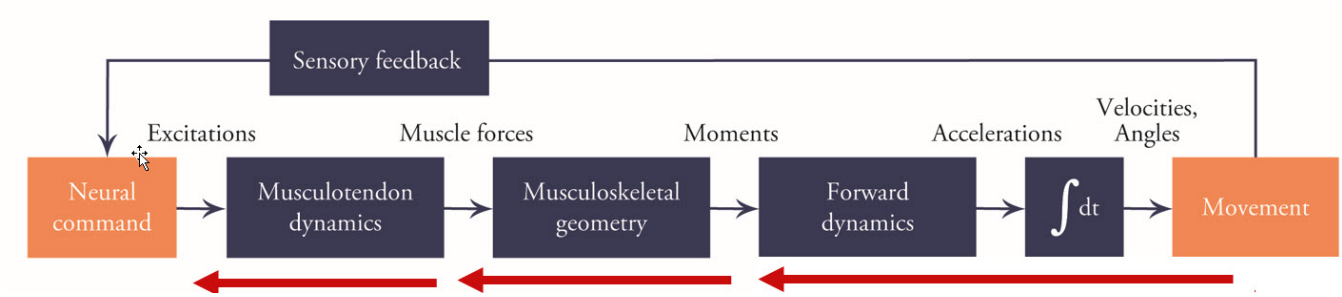
If you are using a generic model from the existing library of models, the next step is to scale the model to match the experimental data collected for your subject—functionality provided by the Scale Tool in OpenSim. The purpose of scaling a generic musculoskeletal model is to modify the anthropometry,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

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

The Inverse Problem

Inverse methods use data measured from observed moments to estimate joint angles and coordinates, joint moments, joint torques, muscle forces, muscle activity, musculotendon dynamics, and other values that are a function of the model's states. The states of the model generally include its coordinates, coordinate velocities, muscle activations, and muscle fiber lengths.

In the figure below, the black arrows show the relationship between different biological processes. The red arrows highlight how the inverse method can utilize data about observed moments to compute quantities involved in generating that movement.



The table below compares and contrasts different inverse methods. Not all methods are available from within the OpenSim graphical user interface (GUI) (see the "Available Interfaces" column below).

METHOD	GOAL	KEY CONSIDERATIONS	AVAILABLE INTERFACES				RESOURCES
			GUI	Command Line*	C++ & Scripting**	Other	
Inverse dynamics	Calculate joint torques from a measured motion	Straightforward; minimal assumptions	X	X	X		Overview User Guide: Inverse Dynamics Hands-on Example (Beginner): Scaling, Inverse Kinematics, and Inverse Dynamics
Static optimization	Estimate muscle force/activations from a measured motion	Fast estimation; assumes rigid tendons; minimizes activation squared at each time step	X	X	X		Overview User Guide: Static Optimization Hands-on Example (Intermediate): Working with Static Optimization Hands-on Example (Intermediate): Estimating Leg Muscle Forces in Stance and Swing
Computed muscle control (CMC)	Estimate muscle excitations from a measured motion	Excitation-activation dynamics; accounts for tendon stretch; minimizes activation squared at each time step	X	X			Overview User Guide: Computed Muscle Control Hands-on Example (Intermediate): Computed Muscle Control Hands-on Example (Intermediate): Estimating Leg Muscle Forces in Stance and Swing CMC Theory and Publications
EMG-informed methods	Estimate musculotendon parameters given a measured motion and muscle activity	Normalizing muscle activity is necessary				X	Overview Calibrated EMG-Informed Neuromusculoskeletal Modeling (CEINMS) Toolbox

*"Command Line" refers to the interactive, text-based interface within OpenSim.

**"Scripting" refers to calling commands from other languages, specifically MATLAB and Python.

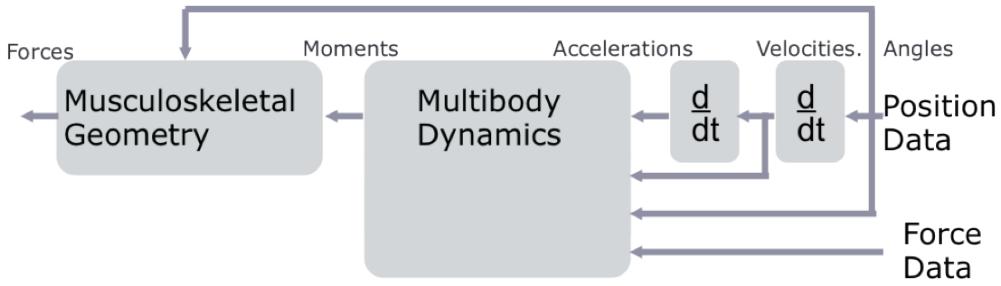
Inverse Kinematics

The Inverse Kinematics (IK) Tool in OpenSim finds values for the generalized coordinates (joint angles and positions) in the model that best match the experimental kinematics recorded for a particular subject (see 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 position the model in a pose that "best matches" experimental marker and coordinate values for that time step. Mathematically, the "best match" is expressed as a weighted least-squares problem, whose solution aims to minimize both marker and coordinate errors.

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

Inverse Dynamics

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). 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 actuated to generate that motion. To determine these internal forces and moments, the equations of motion for the system are solved with external forces (e.g., ground reaction forces) and accelerations given (estimated by differentiating angles and positions twice). The equations of motion are automatically formulated using the kinematic description and mass properties of a musculoskeletal model in Simbody™.



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

Static Optimization

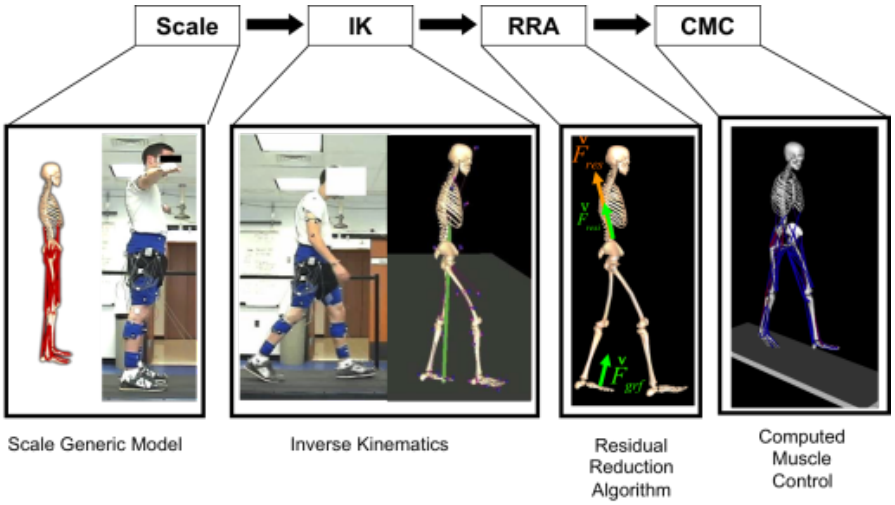
Static optimization is an extension of inverse dynamics that further resolves the net joint moments into individual muscle forces at each instant in time based on some performance criteria, like minimizing the sum of squared muscle forces. See Static Optimization for more details.

Computed Muscle Control

The Computed Muscle Control (CMC) Tool estimates muscle excitations from a measured motion. It does so by dividing a motion into 10ms windows. Within each window, the algorithm performs an optimization to compute the muscle excitations needed to match the motion at the end of the 10ms window.

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.

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



EMG-Informed Methods

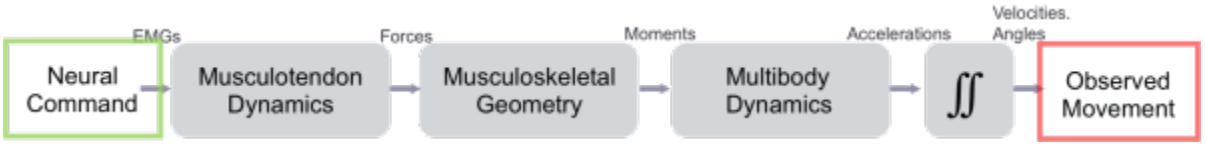
It is possible to include experimental electromyography (EMG) data as an additional input to your inverse problem to estimate musculotendon parameters. We refer to such approaches as "EMG-informed methods." There are two ways to incorporate EMG data:

1. EMG can be used as constraints to muscle excitations in Computed Muscle Control (see the Control Constraints section in our CMC documentation for more information). This was used in work by Hamner *et al.* in simulations of running in order to constrain some muscle activity by EMG measured from the experiments. Their project page can be found here.
2. Use the Calibrated EMG-Informed Neuromusculoskeletal Modeling (CEINMS) Toolbox (Pizzolato, *et al.*, 2015) to calibrate subject-specific models and/or generate EMG-informed simulations

EMG-informed methods provide better estimates of muscle and tendon parameters than CMC because of the additional input data. One key point, however, is to carefully normalize muscle activity data to get good results.

The Forward Problem

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



The table below compares and contrasts different forward methods. Not all methods are available from within the OpenSim graphical user interface (GUI) (see the "Available Interfaces" column below).

METHOD	GOAL	SPEED	KEY CONSIDERATIONS	AVAILABLE INTERFACES				RESOURCES
				GUI	Command Line*	C++ & Scripting**	Other	
Forward dynamics with known controls	Generate a motion based on specified muscle excitations, joint torques, and /or other applied forces	Fast (seconds to minutes)	Easy to set up and can quickly get results; difficult to use for more complex motions (e.g., walking) without adding a controller	X	X	X		Overview User Guide: Forward Dynamics
Shooting methods	Generate a motion based on high-level tasks quantified by an objective function	Slow (hours to days)	Model and controller simplifications are common; controllers are usually motion-specific; can support controllers based on realistic feedback loops (e.g., force)			X	X	Overview SCONE software Webinar: Predictive Simulation of Biological Motion Using SCONE

Reinforcement learning (RL)	Generate a motion based on high-level tasks quantified by an objective function	Very slow (days to weeks)	Model simplifications are common; may require very large amount of computing power; minimal input needed from user so workflow can be extended to many motions				X	Overview osim-rl tool Webinar: Robust Control Strategies for Musculoskeletal Models Using Deep Reinforcement Learning
Direct collocation	Quickly generate a motion based on high-level tasks quantified by an objective function; intermediate solutions do not necessarily satisfy physical constraints	Middling (minutes to hours)	Capacity to scale to more complicated models; difficult to implement (e.g., constraints, providing derivatives); difficult to add feedback loops (i.e., for reflexes)			X	X	Overview OpenSim Moco Webinar: OpenSIm Moco: Software to Optimize the Motion and Control of OpenSim Models

*"Command Line" refers to the interactive, text-based interface within OpenSim.

**"Scripting" refers to calling commands from other languages, specifically MATLAB and Python.

Forward Dynamics with Known Controls

In a forward dynamic simulation of motion, muscle excitations, joint torques, and/or other applied forces are used to drive the motion of a model. Or in some cases, the model may have its own controller (e.g., that models reflexes).

The Forward Dynamics Tool takes a set of controls to drive a model's motion by integrating forward in time. An example of such control signals are muscle excitations, which could be collected experimentally or generated via a tool like the Computed Muscle Control (CMC) Tool described above. Note that a forward simulation using such excitation signals without any additional controller will generally not create a stable gait as smaller integration errors can accumulate and experimentally measured EMG is not an exact measure of the underlying muscle activity.

A few examples of forward dynamic simulations include:

- How muscle fiber lengths and velocities affect muscle force generation as humans walk and run at different speeds (Arnold, *et al.*)
- Preparatory co-activation of the ankle muscles may prevent ankle inversion injuries (DeMers, *et al.*)
- Tutorial example of a drop landing simulation

Refer to the Forward Dynamics section of the User Guide for additional information.

Shooting Methods

Developing controllers to generate a forward dynamics simulation can be challenging. As such, methods have been developed that can generate controls in an automated manner to achieve the desired behavior or movement in forward dynamics. Shooting methods are one such method. In shooting methods, a simulated trajectory is "shot" forward in time (e.g., a forward simulation is run). At the end of each simulation, the results are evaluated by an objective function, a mathematical representation of the criteria you want your simulation to meet. Example objective functions include minimizing metabolic cost or minimizing joint contact. An optimizer then updates the controller parameters for the next iteration of the forward simulation. This process is repeated until the objective function is met within the specified tolerance. The SCONE software, built on top of OpenSim, enables researchers to use shooting methods for their studies.

Reinforcement Learning (RL)

Reinforcement learning (RL) is another method to generate controls for forward dynamics simulations in an automated manner. Like shooting methods, RL generates a motion based on objective functions, the mathematical representation of the criteria you want your simulation to meet, but it takes a different approach to developing the controller. The RL method consists of three components:

- State-based Controller or Policy: RL creates a state-based controller, which tracks parameters (states) that describe the environment or behavior of interest. Example states are joint angles or center of mass acceleration. In RL language, this state-based controller is called the policy.
- Objective Function or Reward: The policy seeks to maximize the objective function, often called a reward.
- Agent: In RL, there is an agent, or model, which is performing actions within an environment.

Based on a given action, the objective function (reward) is computed, along with the controller values needed to change the state. Then, just as with shooting methods, RL iterates over this process. Over more iterations, the policy or controller is updated to get better and better rewards. Thus, the policy is being trained to create a controller without input from the user. osim-rl, built on top of OpenSim, incorporates musculoskeletal modeling into a reinforcem ent learning environment.

Direct Collocation

Forward methods, particularly shooting methods and reinforcement learning, can be slow. Direct collocation aims to generate a forward simulation with the speed of inverse methods. It does so by concurrently optimizing the whole motion trajectory and muscle excitations (or other control values). This can speed up the process quite a bit with physical constraints satisfied at the end of the optimization. While direct collocation problems have traditionally been challenging to set up and solve, the OpenSim Moco project provides an open-source toolkit to facilitate the use of direct collocation methods in human and animal movement studies.

Frequently Asked Questions About Choosing a Simulation Pipeline

- **Which method is better: forward or inverse dynamics?** Forward and inverse methods are both valuable methods with different goals, and neither method is always better than the other. It is important to choose the method that is needed to answer your research question, and the webinar provided examples of many studies to help guide this process. In general, we suggest choosing the easiest and quickest methods that are sufficient for your problem.
- **Are forward and inverse dynamics simulation more suitable for upper and lower limb modelling, respectively?** Neither method is always better than the other, as mentioned in the response to the previous question. We suggest choosing the easiest and quickest methods that are sufficient for your program. For example, forward dynamic simulations of the upper extremity can be easier to generate compared with simulations of the lower extremity, as upper extremity motions don't generally require simulating complicated foot-floor contact.

- **Which of the pipelines can be used if marker (motion capture or MOCAP) data is not available, for example to estimate muscle forces reliably? -**In general, if accurate muscle force estimation is needed, we highly recommend inverse methods over forward methods. Since the motion is used as a constraint in inverse methods, it provides more confidence in these estimations. Furthermore, with forward methods, simplified models are often used, which may group muscles together (e.g., one muscle for three vasti muscles) and thus can only give estimates for muscle groups. There are many datasets (e.g., Grand Challenge to Predict in Vivo Knee Loads, Gait in Children with Spastic Cerebral Palsy, R unning at Multiple Speeds) that have been shared by the community that we encourage you to use if you are not able to collect your own motion capture. If a forward method is necessary, all forward methods can estimate muscle forces, although there is less confidence in these estimates. If your motion is simple enough, using the forward dynamics tool is a quick way to get rough estimates. Direct collocation could be used for more complicated motions, and can generate simulations faster than shooting methods or reinforcement learning.

- **Which is better for joint reaction force analysis: CMC or SO?** Static optimization (SO) and CMC have both been successfully used for joint reactions analysis as they estimate the muscle forces necessary for calculating joint reaction forces. Often, the two methods will give similar results with respect to the timing of when muscles activate; however, CMC will tend to give higher forces since it typically yields solutions with more co-contraction. When your research question depends on the interaction between muscle and tendon during a motion or when elastic storage of energy in tendon is a known or likely contributor to the motion of interest (e.g., high-force motions like running), then you should use CMC or a framework that models tendon-dynamics. For either static optimization or CMC, validating that the muscle forces are reasonable for your motion is the most important step to understanding which method is best for your problem.

- **I am adding an external torque to the right foot to simulate high ankle sprain injuries during gait. Can I modify existing gait files (such as ground reaction force files) and run forward dynamics to observe fibular motion?** For problems in which you want to study the effect of changing the skeletal motion or the ground contact forces, it can be difficult to use inverse methods. For this problem, we would suggest looking at a direct collocation package such as OpenSim Moco.

- **I have real experimental data for osteoarthritis patients and am trying to determine if a hip or ankle exoskeleton would be best to assist walking. I have joint angles, moments, and power. What is the best pipeline for this?**
  A good place to start could be to use CMC to find a motion that has the same joint angles and moments but with an ideal torque actuator at the hip or ankle. This was done previously for studying hip, knee, and ankle devices for walking in an article titled "Simulating ideal assistive devices to reduce the metabolic cost of walking with heavy loads", and for running in an article titled "Simulating Ideal Assistive Devices to Reduce the Metabolic Cost of Running".

- **My research question is whether lateral sway of the bicycle is the optimal strategy for cyclists to use during sprint cycling, i.e., travel 250 m as fast as possible. Which pipeline would you recommend?**
  It can be helpful to break down complex questions into smaller pieces that are testable by more targeted simulations. For instance, if you are interested in whether the sway of the bicycle itself is important, then a simulation without a complex musculoskeletal muscle could be useful. It could also be possible to gain insight through inverse methods by using data of individuals cycling with lateral sway and without lateral sway. If it is necessary to predict a new motion with a complicated model, direct collocation is likely the best option of the methods for this case, and careful validation should be used to gain confidence in the results (e.g., showing you can accurately predict known motions).

- **Can I run forward dynamics using torques as input data to control a model?** Yes, torques can be used to drive a motion. You will need to modify your model to include new components to simulate the torques. Two components that are helpful for this are the TorqueActuator components, which apply torques between two bodies, and CoordinateActuators, which apply generalized forces and torques to a specific coordinate. Each of these can be used along with a PrescribedController, which can be used to feed in input signals to the actuators. To find out the XML format for adding these components to your model, refer to the XML Browser in the GUI (under the "Help" dropdown menu, click "XML Browser").

- **If I do not have muscle attachments and PCSA for a non-human animal, can I still run an inverse analysis?** The requirements for the model are the same for human and non-human animals. If you only need to calculate joint torques via inverse dynamics, only the skeletal model is needed, and thus, muscles and their parameters are not needed for inverse dynamics. If you need to estimate muscle activity, you need a model of the animal of interest that contains accurate muscle and skeletal geometry relevant to the muscles and joint motions of interest.

- **Can CMC be run for models that do not include a torso (i.e., lower extremity models with pelvis)?** CMC can be run on a model without a torso. However, you would have to adjust the model to account for the missing mass of the torso or else there will be large discrepancies between the ground reaction force and the kinematics. Even after making these adjustments, it will be important to make sure that your residuals from the CMC solution are small enough to trust your simulation results for your use case.

- **If I have a problem that is best addressed by reinforcement learning or direct collocation, would it be a good idea to use a CMC solution as the initial guess?** One way to improve the speed of more complicated algorithms such as reinforcement learning or direct collocation is to give it a good initial guess. Using a quicker method, such as CMC, to give a good guess is a great way to do just this. We have seen users leverage this strategy for both reinforcement learning and direct collocation.

Analyzing Simulations

Answering your research questions often 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 (including 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 relative to 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; 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 yield the resultant body force on the contacting bodies separately, expressed in the ground frame. For constraints, the same is true, except the forces are expressed relative to the most distal common ancestor body. Whenever a constraint involves ground, this is the ground body; however, if (for example) a model of the arm has a hand with fingers touching via a point constraint, then the forces are expressed in the nearest common ancestor, which would be the palm (if modeled as a single body).

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

Next: Preparing Your Data