Simulation-Based Design to Reduce Metabolic Cost

Overview

The purpose of this exercise is to demonstrate how the OpenSim software platform can be used to design and evaluate devices to assist locomotion. You will work with a simplified model of the musculoskeletal system and a simulation of a typical, unloaded gait. The subject in this exercise is a (approximately) 72kg, 1.8m tall male, walking at 1.2 m/s on a treadmill.

You will learn how to generate muscle-driven simulations with and without simple passive and active assistive devices and analyze how these devices affect the metabolics of gait. Optional sections explore setting up a model with metabolic probes/calculators and using the OpenSim API libraries in the scripting shell to add simple assistive devices to a model.

The exercise includes the following sections:

I. Explore the Model

First, we will explore the components that make up the musculoskeletal model. You'll launch OpenSim and use the Navigator panel to explore the bodies, joints, and muscles in the model. We are using a model that has been simplified from the models we typically use to study gait so that you can quickly generate simulations and iterate on assistive device designs. In particular:

- The upper extremity has been lumped into one torso segment to represent the trunk, arms, and head.
- The model includes only one degree of freedom at each hip and ankle, and between the torso segment and pelvis.
- The number of muscles in the model has been reduced by lumping the key flexor/extensor muscles in the lower extremities.

To further simplify the exercise and allow you to focus on muscle-driven simulation and metabolic cost, we'll start with a model that has already been scaled and run through the Residual Reduction Algorithm. In particular:

- The generic model has been scaled to match the subject's anthropometry (i.e., mass and segment lengths) using the Scale Tool.
- The Inverse Kinematics Tool was used to determine the model coordinates, as functions of time, that yield the observed marker trajectories.
- The Residual Reduction Algorithm Tool used the scaled model and the inverse kinematics results to make small adjustments to segment inertial properties and the joint kinematics for the trial to achieve a model and kinematics that are dynamically consistent with the measured ground reaction forces.

You can read more about this process in the User's Guide chapters on Scaling, Inverse Kinematics, and the Residual Reduction Algorithm.

A. Launch OpenSim

On Windows, you can launch OpenSim from the Start menu. If you used default installation settings, simply select **Windows>All Programs>OpenSim**. You will see multiple information panels and an empty main view.

B. Load and explore the model

1. In the OpenSim GUI, select **File>Open Model...**.
2. Navigate to your **OpenSim resources directory** and find the **Models/Gait10dof18musc** directory.
3. Select **subject01_metabolics.osim**. This is the final model created in the RRA step described above, with metabolic probes/calculators included. The OpenSim GUI will now show a view window containing a model with a skeleton.
4. Go to the **Navigator** panel and use the **+ icon** to expand the list of components in the model. From here, you can explore the bodies, joints, and forces (i.e., muscles) that make up this model.

5. Go to the **Coordinates** panel, which lists the degrees of freedom in the model. Move the sliders to change the values of the coordinates.
6. Try changing the model view in the **Visualizer Window** to zoom in on the ankle joint (see Navigating the Visualizer Window).
7. Compare the RRA model (which you just loaded) to the generic model:
    a. Reset the scaled model (walk_subject01) to its default pose by going to the **Coordinates** panel and selecting **Poses>Default**. Zoom out to bring the full model into view. You can use the box icons in the top-left corner of the Visualizer Window to center the camera on the X-, Y-, or Z-axis. (See Navigating the Visualizer Window for more tips.)
    b. Open the generic model by selecting **File>Open Model...**, navigating to **Models/Gait10dof18musc**, then selecting **gait10dof18musc.osim**. This newly loaded model is now the Current Model, with its name shown in bold in the Navigator panel.
    c. Compare the mass of the torso segment between the two models. Highlight the body of interest in the Navigator panel, then find the mass property in the Properties window.
    d. Compare the model sizes in the Visualizer Window.
    e. When finished, close the generic model by right-clicking the model's name in the Navigator panel and selecting Close from the drop-down menu.

Questions

1. How many degrees of freedom does the model have? How many muscles? How many bodies?
2. Do any muscles cross the lumbar joint? If you're unsure, highlight a muscle in the **Navigator** panel, find the **Properties** window, and click the "**...**" button next to **GeometryPath**. Do you think the model's muscles will be strong enough to control the motion of the trunk during gait? How might we compensate for this in the simulation?
3. Which model (generic or subject-specific) do you think has a lower BMI (body mass index)? Recall that

$$\text{BMI} = \frac{\text{mass[kg]}}{(\text{height[m]})^2}$$

II. Simulate Unassisted Walking

In this section, you will generate a simulation of unassisted walking using OpenSim's Computed Muscle Control (CMC) Tool. As noted above, we will begin with the kinematics data that are output by the RRA Tool. The CMC algorithm finds a set of muscle excitations that track the input kinematics (within an error tolerance). CMC solves the muscle redundancy problem by minimizing the sum of squared muscle activations. You can read more about CMC in the User's Guide chapter on Computed Muscle Control. After we generate the simulation, we'll assess the quality of the results, a vital step in the simulation process.

A. Load and explore the motion

1. Load the RRA-adjusted kinematics file by selecting **File>Load Motion...** and navigating to **RRA/ResultsRRA**. Select **subject_adjusted_Kinematics_q.sto**.
2. Play the motion using the **Motion Slider Panel** (blue controls in the top-center of the screen). The motion slider panel's controls allow you to play and pause the motion, step frame by frame, control playback speed, and loop the motion. You can use the slider to scroll through the motion.
3. It is often useful to visualize the ground reaction forces (GRFs) for the motion. Find **Motions>Coordinates** in the **Navigator** panel. **Right-click** and select **Associate Motion Data...**. Navigate to **subject01_walk_grf.mot** and select **Open**. Now when you play the motion, you will see green arrows corresponding to the experimentally measured ground reaction forces.
4. By examining the motion, determine the times of key gait cycle events. A *gait cycle* is defined as the period between consecutive heel strikes of the same foot. What time range defines a gait cycle for this trial (starting and ending with the right foot)?

B. Use Computed Muscle Control (CMC) to generate a muscle-driven simulation

1. Go to **Tools>Computed Muscle Control** to launch the CMC Tool. In the CMC panel, you will specify the settings for generating a simulation. The tool operates on the model currently selected in the GUI (**walk_subject01** in this case).
2. We have configured CMC for you and saved the settings in a file. Load these settings by selecting **Load...** and opening the file **walk_Setup_CMC.xml**.
3. In the **Main Settings** tab, you will first find the **Input** pane.
    a. For *Desired kinematics*, we've selected the RRA-adjusted kinematics you loaded in section A above (subject_adjusted_Kinematics_q.sto).
    b. You do not need to filter kinematics; these data are already smooth because they resulted from a run of RRA.
    c. The *Tracking tasks* file, gait10dof_Kinematics_Tracking_Tasks.xml, defines the coordinates that OpenSim should track (in this case, all coordinates in the model) and a corresponding set of weights. (You can use the pencil/paper icon to explore the contents of the tracking tasks file.)
    d. We specified a time range that corresponds to one gait cycle, as you determined in section A above (right heel strike to right heel strike is approximately 0.6 to 1.9 seconds).
4. To run CMC, we also need to add residual and reserve actuators to the model and specify external loads (e.g., GRFs). These are specified in the **Actuators and External Loads** tab.
    a. We've added an *Additional force set file*, gait10dof_Reserve_Actuators.xml. This file creates reserve actuators at each of the model's joints and 3 residual actuators at the pelvis. These actuators are forces/torques that are added about each joint and at the pelvis to augment the muscle forces, if necessary, in order to track the desired kinematics.
    b. The "Append to model's force set" radio button is selected. This means that we want to add these reserve and residual actuators to the set of muscle actuators already included with the model.
    c. Finally, the *External loads specification file*, subject01_walk_grf.xml, defines how the forces in the subject01_walk_grf.mot file are applied to the model's foot segments (calcn_r and calcn_l).
5. Click **Run**. You can now **Close** the tool (CMC will continue running). You will see the model animate as results are generated.

C. Evaluate the simulation results

1. **Play** the results using the **Motion Slider Panel**. Observe the muscles changing color from blue to red as activation increases.
2. Now we'll dive into more detail to make sure the kinematics from the CMC simulation are a good match with the input kinematics.

    a. Select **Tools>Plot...** to open a Plotter window.
    b. Click on **Y-Quantity...** and select **Load file...** at the bottom of the list. Navigate to the folder where the CMC results were saved (**CMC /ResultsCMC** by default) and find the file **walk_subject_Kinematics_q.sto**. Check the boxes for **pelvis_tilt**, **hip_flexion_r**, **knee_angle _r**, and **ankle_angle_r** and hit **OK**.
    c. Click on **X-Quantity...** and select **time**. Click **Add** to add these curves to the plot.
    d. Now we'll load the input kinematics. Repeat steps b and c, but select the **subject_adjusted_Kinematics_q.sto** file from the **RRA /ResultsRRA** folder.
    e. Examine the kinematics errors. Hover over any point on the curve to see its value. You can zoom in on an area of interest by drawing a box around the region (click and drag from north-west to south-east). Click and drag in the opposite direction (from south-east to north-west) to zoom out.
    f. **Minimize** the plot window (don't close) to save for viewing later.
3. You did the plotting above by hand, but you can also generate plots in OpenSim using scripts. We've prepared a CMC plotting script for you to evaluate the quality of CMC results. This script will come in handy as we test several assistive devices. To run the script:
    a. Select **Scripts>Run..**.
    b. Find the folder **Scripts** and select **CMC_EvaluateResults.py**.
    c. When the script starts running, you will be prompted to choose a folder. Select the folder where you saved your CMC results.
    d. Examine the three plots generated by the script:
        i. The kinematic tracking errors. This is the difference between the RRA and CMC curves you plotted above. These errors should generally be 2 degrees (0.035 radians) or less for gait simulations, as indicated by the plotted thresholds.
        ii. The residual forces and moments applied to the pelvis (the "hand-of-God" forces).
        iii. The reserve actuator torques applied to each of the joints.
    e. **Minimize** the plot windows; we'll come back to them once we've simulated assisted walking.
4. Finally, let's look at the **muscle forces** from the simulation. For now, we'll focus on muscles that cross the ankle joint.
    a. Select **Tools>Plot...** to open another Plotter window.
    b. Click on **Y-Quantity...** and select **Load file...**. Choose the **walk_subject_Actuation_force.sto** file from the CMC results folder.
    c. Check the boxes for the gastrocnemius, soleus, and tibialis anterior muscles on the right leg (gastroc_r, soleus_r, and tib_ant_r).
    d. Click on **X-Quantity...** and select **time**.
    e. Click **Add** to add the curves to the plot. Click on **Properties...** in the Curves List pane and change the title of the plot to 'Muscle Forces'.

Questions

1. Which coordinate has the largest kinematic errors? What is the maximum value of its error? Were the kinematics tracked within 2 degrees (0.035 radians), the recommended threshold?
2. What is the maximum value of the residual forces? Residual forces are generally considered "Good" for gait simulations if they remain below 10 N throughout the simulation and "Okay" if they remain below 25 N (see Best Practices). Why might the residual be only "Okay"? (Residual moments are considered "Good" if they remain below 50 N-m.)
3. Why is the lumbar extension reserve so much larger than the reserves for the hip, knee, and ankle?
4. When do plantarflexor forces peak? What about the dorsiflexors?

How do you know whether your residual forces are low enough? The thresholds we used for this exercise are rough guidelines, which will vary from study to study. In our case, the peak power of the residuals is under 4 Watts, while the ankle muscles and iliopsoas have peak powers of over 120 Watts during the walking trial. Thus, our residuals are relatively insignificant in terms of energetics. Further, you can get less kinematic chatter and smoother power curves if you use a small CMC time window (see How CMC Works). This increases the number of simulation steps (and, therefore, simulation time), but is a good refinement process to consider when conducting research studies (vs. running through a simplified exercise).

III. Explore Metabolics of Unassisted Walking

We can also use the muscle-driven simulation to explore the metabolic cost of the walking motion. The metabolics calculators in OpenSim can calculate the rate or total amount of energy consumption for each muscle in the model, and for the whole body. The models of metabolics from Umberger et al. (2003) and (2010) account for the effect of muscle mass, the ratio of fast- and slow-twitch fibers in the muscle, and the lengthening/shortening velocity; the OpenSim implementation includes extensions by Uchida et al. (2016). The model you've been working with in Sections I and II above includes a set of metabolic probes to calculate the energy consumed by all the muscles in the model, as well as each individual muscle on its own. Read more about the metabolic probes on their doxygen page.

A. Explore the metabolic probes

1. In the **Navigator** panel, find the **Probes** set for the model.
2. Use the **Property Editor** to find the metabolics probe included with the model.
    a. Find the **probe_operation** property. What would happen if you changed the operation from 'value' to 'integrate'?
    b. Find the **aerobic_factor** property. Are we assuming that the activity is aerobic or anaerobic? Do you think this is a reasonable assumption for walking?

B. Analyze the metabolic calculations for the walking simulation

1. Now let's plot the results. When you ran CMC, a ProbeReporter analysis ran in the background to calculate the metabolic rate for each muscle and the total metabolic power consumed for all muscles, over the gait cycle.
2. Select **Tools>Plot...** to open a Plotter window.
3. Click on **Y-Quantity...** and select **Load File...**. Navigate to the folder where you saved your CMC results and find the **walk_subject_MetabolicsR eporter_probes.sto** file. Check the box for **metabolics_TOTAL** and click **OK**.
4. Click on **X-Quantity...** and select **time**.
5. Click **Add**. The plot shows the total instantaneous metabolic power consumed (in units of Watts) over the walking trial, since the operation of the probe is set to 'value'.

6. Create a second plot that includes the metabolic probe output for the plantarflexor muscles of the right leg (gastroc_r and soleus_r).

Questions

1. For which parts of the gait cycle is the total rate of metabolic energy consumption highest?
2. When we plotted muscle forces in Section II.C above, the soleus had a force peak at around 0.9 s and a second force peak in late stance at around 1.3 s. The rate of energy consumption for the soleus is much lower in mid-stance than late stance. What are some of the possible reasons for the difference between force production and metabolic cost? (Note that this first peak of soleus activity is not commonly observed in walking. Recall that the model has been simplified for the purposes of this exercise.)
3. Bonus Question: What is the metabolic energy consumed, in Joules, for one walking trial? The total mass of the subject is 75 kg. What is the average cost over the walking trial in J/kg/s (or W/kg)? Is this value higher or lower than values reported in the literature of around 4–6 W/kg for walking at a speed of 1.2 m/s? You can compute the integral offline or re-run CMC with the probe_operation set to 'integrate'.

IV. Model Building (Optional)

In this section, we will create two models, each with a different simple assistive device on the right limb, using the scripting shell and Property Editor. Although the devices are simple for the purposes of the exercise, the OpenSim software platform has an extensive library of actuators and controllers. The gait10dof18musc folder already includes these two augmented models, so you can skip these model-building steps and go directly to section V if you wish.

To create the models, we'll use the scripting shell and the Property Editor.

You can't paste multiple lines of code at the command prompt (>>>) in the ScriptingShell Window. We recommend creating a new (.py) file to store your code for model building and running in the GUI. As of OpenSim 3.2, if you mistakenly paste multiple lines of code, you can quit by hitting the Esc key.

A. Create a model with a torsional ankle spring

We will model a torsional spring at the ankle joint as a CoordinateLimitForce. This force element is generally used to limit the range of motion of a coordinate by hitting a stop that is modeled as a spring and damper, which is engaged smoothly as the coordinate reaches and begins to exceed its predefined limit. We will use the CoordinateLimitForce to act like a spring with some damping at the ankle joint when the ankle exceeds a specified angle. The CoordinateLimitForce that we'll add to the model will generate forces when the dorsiflexion angle is greater than 5 degrees, according to the constant stiffness value K_upper (10 N-m/degree), which will generate a plantarflexion moment. Note that the *transition* property determines the region in which the stiffness transitions from zero (before the limit) to K beyond the limit.

Make sure the subject01_metabolics.osim model that we were working with above is loaded in the GUI and made current (**walk_subject01** should appear in boldface in the Navigator panel). Find the ScriptingShell window (below the View window) and the command prompt >>>. Enter the commands below to create a new model and add a torsional spring to this model:

```
# Get a handle to the current model and create a new copy
baseModel = getCurrentModel()
ankleSpringModel = baseModel.clone()
ankleSpringModel.setName(baseModel.getName() + '_ankle_spring')

# Create the spring we'll add to the model (a CoordinateLimitForce in OpenSim)
ankleSpring = modeling.CoordinateLimitForce()
ankleSpring.setName('AnkleLimitSpringDamper')
# Set the coordinate for the spring
ankleSpring.set_coordinate('ankle_angle_r')

# Add the spring to the model
ankleSpringModel.addForce(ankleSpring)

# Load the model in the GUI
loadModel(ankleSpringModel)
```

Now that we've added the spring to the model, we can set its properties using the Property Editor in the GUI.

1. Find the spring in the model you created: **Navigator>walk_subject01_ankle_spring>Forces>Other Forces**.
2. Select the **AnkleLimitSpringDamper** object and update its properties:
   a. upper_stiffness = 10.0 This is the stiffness of the ankle spring assist during ankle dorsiflexion.
   b. upper_limit = 5.0 The spring is "engaged" when the ankle angle is greater than 5 degrees (dorsiflexion).
   c. lower_stiffness = 1.0 The CoordinateLimitForce will also generate forces preventing excess plantarflexion, as specified by the lower_stiffness and lower_limit values.
   d. lower_limit = -90.0 Allow the full range of plantarflexion motion.
   e. damping = 0.01 A small damping term.
   f. transition = 2.0 This term dictates the transition from zero to constant stiffness as the coordinate exceeds its limit (upper or lower), in degrees.
3. Save the new model to a file. Right-click on the model's name (walk_subject01_ankle_spring) in the Navigator panel and choose **Save As...**. Name the model file subject01_metabolics_spring.osim.

After you've loaded a model in the GUI, you can't add new components to it (you can only change the properties of existing components). Use the clone, add, and load workflow described above to add components to a model.

B. Create a model with a biarticular path spring

We will also create a model with a passive spring that acts along a path between the femur and foot segments. The path spring element takes a GeometryPath (similar to muscles), along with resting length, stiffness, and dissipation values. We will use the spring to augment the gastrocnemius muscle with a stiffness of 10000 N/m.

Once again, make sure the subject01_metabolics.osim model that we were working with in Sections I, II, and III is loaded in the GUI and made current (**wal k_subject01** should appear in boldface in the Navigator panel). Find the ScriptingShell window and the command prompt >>>. Enter the commands below to create a new model and add a path spring to that model. This time, we'll set all the path spring's properties before we add it to the model instead of using the GUI Property Editor to do so, demonstrating a second modeling workflow.

```
 # Get a handle to the current model and create a new copy
baseModel = getCurrentModel()
pathSpringModel = baseModel.clone()
pathSpringModel.setName(baseModel.getName() + '_path_spring')

# Create the spring we'll add to the model (a PathSpring in OpenSim)
name = 'BiarticularSpringDamper'
restLength = 0.4
stiffness = 10000.0
dissipation = 0.01
pathSpring = modeling.PathSpring(name,restLength,stiffness,dissipation)

# Set geometry path for the path spring to match the gastrocnemius muscle
gastroc = pathSpringModel.getMuscles().get('gastroc_r')
pathSpring.set_GeometryPath(gastroc.getGeometryPath())

# Add the spring to the model
pathSpringModel.addForce(pathSpring)

# Load the model in the GUI
loadModel(pathSpringModel)

# Save the model to a file
fullPathName = baseModel.getInputFileName()
newName = fullPathName.replace('.osim', '_path_spring.osim')
pathSpringModel.print(newName)
```

V. Simulate Walking with Passive Devices

Now that we've established a baseline for unassisted walking in Section III, let's see whether we can reduce metabolic cost with an assistive device. We'll try two passive devices first: a torsional spring at the ankle and a biarticular linear spring that follows the path of the gastrocnemius muscle. Use the models you created above or the models included in the gait10dof18musc folder.

A. Generate a muscle-driven simulation with a torsional ankle spring and plot the results

1. **Open** the model with the ankle spring (**subject01_metabolics_spring.osim**) and make sure it is current (bold) in the GUI.
2. Find the ankle spring in the list of model **Forces**. What are the properties of the spring? You can read more about the torsional spring in the Model Building section above.
3. Launch the CMC Tool and load the settings file **walk_Setup_CMC.xml**.
4. Change the *Output Directory* to **ResultsCMC_AnkleSpring**. Leave all other settings the same; CMC is run using the current model.
5. Hit **Run** and then **Close** the CMC Tool window.
6. Let's plot the baseline (unassisted) results for comparison. Run the script **plotBaselineResults.py**. When prompted, choose the folder where you saved the baseline CMC results (e.g., CMC/ResultsCMC). This script will generate plots of the total cumulative metabolic energy consumption during the gait cycle and the metabolic rate over the gait cycle for the gastrocnemius, soleus, tibialis anterior, and iliopsoas muscles. *Note: These plots must remain open for the rest of the exercise because we'll add additional curves to them in the next section.* ***Minimize****, don't close.*
7. Now we'll add the results from the simulation with the torsional ankle spring. Run the script **addPlotDeviceResults.py**. When prompted, choose the folder where you saved the Ankle Spring CMC results (e.g., CMC/ResultsCMC_AnkleSpring). This script will add curves corresponding to the ankle spring to each of the plots created in the previous step.
8. **Minimize** the plots so that we can add the results for the path spring in the next section.

B. Generate a muscle-driven simulation with a biarticular path spring and plot the results

1. **Open** the model with the path spring (**subject01_metabolics_path_spring.osim**) and make sure it is current (bold) in the GUI.
2. Find the path spring in the list of model **Forces**. What are the properties of the spring? You will also see the path spring visualized in green in the View panel. What path does the spring follow? You can read more about the path spring in the Model Building section above.
3. Launch the CMC Tool and load the **walk_Setup_CMC.xml** settings file.
4. Change the *Output Directory* to **ResultsCMC_PathSpring**. Leave all other settings the same.
5. Hit **Run** and then **Close** the CMC Tool window.
6. Now we'll add the results from the simulation with the biarticular path spring. Run the script **addPlotDeviceResults.py** again. Use **Run Recent>** to quickly run the script you used in part A above. When prompted, choose the **ResultsCMC_PathSpring** folder. This script will add curves corresponding to the path spring to each of the plots created in the previous step.

Questions

1. Examine the Total Metabolic Energy plot. Which device decreases the cost of gait and at what parts of the gait cycle?
2. Examine the Muscle Metabolic Rate plots for the right-limb gastrocnemius, soleus, and tibialis anterior muscles in early- to mid-stance. Do the devices tend to increase or decrease the rate of energy consumption by the plantarflexors from baseline? How do the devices affect the metabolic rate of the tibialis anterior? Why do you think you observe these differences?
3. Examine the Muscle Metabolic Rate plots for the soleus and iliopsoas in late stance and swing. Do these results explain the savings observed with the biarticular path spring?
4. You should make sure that adding the spring didn't introduce any significant tracking errors, residuals, or reserve forces. Re-run the script **CMC_EvaluateResults.py** twice, first choosing the **ResultsCMC_AnkleSpring** folder and then choosing the **ResultsCMC_PathSpring** folder. Do you notice any significant increase in tracking errors, residuals, or reserve forces due to adding the devices?

VI. Explore on Your Own: Simulate Walking with an Active Device

We've also provided a model and setup files to simulate the effects of an active biarticular path actuator. In fact, the actuator can take any control signal (e.g., one that you've measured from a physical device) over the time course of the gait simulation. The path of the actuator is the same as the passive path spring we examined in the section above. If desired, this geometry path can be modified as well.

A. Generate a simulation with an active path actuator

To generate a simulation with the path actuator, take the following steps:

1. Load the model **subject01_metabolics_path_actuator.osim**.
2. Launch the **CMC** tool and **Load...** the settings file **walk_Setup_CMC_PathActuator.xml**.
3. In the **Navigator** panel, find the path actuator in the model, **PlantarFlexAssist**. What is its optimal force? The optimal_force property scales the input control signal (e.g., if the input control signal is 1 and the optimal force value is 10, the tension in the actuator will be 10 N).
4. Note the differences between the CMC setup files loaded in the previous sections. Here, we've specified an Actuator constraints file, controls_43p_r.xml. This file contains the desired control signal for the path actuator. To view the controls, click the pencil icon, check the PlantarFlexAssist box, and click Add. This will open the Excitation Editor, where you can view and edit the signal. (Click the Help button in the panel for instructions.)
5. Change the output results directory to **ResultsCMC_PathActuator** or another meaningful name.
6. Click **Run**, then Close the **CMC** tool.
7. When CMC completes, you can use the plotting scripts used above to display the results.

B. Create your own control signal

You can also generate your own control signal for the path actuator (or another model component like a torque actuator). You can generate a new control signal using the excitation editor, as described above. Or you can modify the signal in Excel and then import this signal into OpenSim using the following steps:

1. Go to a **folder browser** and find the folder **PlantarFlexorAssistCurves**. This folder includes a set of storage (.sto) files that have control signals for the PlantarFlexAssist path actuator (with different onset times).
2. Open one of the storage files in Excel (e.g., **exoskeleton_actuation_toeoff_13p_r.sto**, which starts actuation at 13% of the gait cycle).
   a. The file has two columns: one for time and one for the PlantarFlexAssist.excitation. You can use the existing curves or vary the signal to match some desired pattern.
   b. Time must be the first column and note that the time stamp must match the simulation that you are using (e.g. for the current simulation heel strike occurs and 0.6 and then 1.9 seconds).
   c. You can add additional control signals if your model has multiple devices. The name of each column must match the name of the actuator and have the suffix ".excitation", as in the example files.
   d. You can add a min and max value for the excitation signals by adding columns with the suffix "_min" or "_max" (e.g. PlantarFlexAssist.excitation_min). If you enter no min and max, the script we'll use in the next step will assume that you want to apply the signal exactly.
3. Go back to **OpenSim** and select **Scripts>Run... Find the file** createControlsFromStorage.py (see <Resources Dir>/Code/GUI). At the first prompt, select the storage file from step 2 above (e.g. **PlantarFlexorAssistCurves/exoskeleton_actuation_toeoff_13p_r.sto**) and hit **Open**. At the second prompt, go to the CMC folder then enter a file name (e.g. **CMC/controls_13p_r.xml**). Hit **Save**.
   a. The createControlsFromStorage has an optional buffer. Edit the script to add an offset for the max and min values of the control (e.g. the specified signal plus or minus 0.1). This is useful if CMC has tracking your input control signal.

This will generate an Actuator constraints file that you can use with CMC (following the process in A, above).

VII. Explore on Your Own: Using the Metabolic Probes with Other Models

Musculoskeletal models can contain dozens of muscles; manually adding a metabolic probe to each would be tedious and error-prone. We have provided a script (Scripts/addMetabolicProbes.py) that adds a metabolic probe to each muscle in the current model. These probes or calculators determine the rate at which each muscle in the model consumes energy throughout a simulation. A whole-body metabolic probe is also added to calculate the total energy consumed by all muscles in the model.

Note that the ratio of slow- to fast-twitch fibers in a muscle is required to compute the muscle's metabolic cost. We have provided examples of these data for the gait10dof18musc and Gait2392 models in the folder Scripts. The text files metabolicsSlowTwitchRatios_gait10dof18musc.txt and metabolicsSlowTwitchRatios_Gait2392.txt are used by the script to set the appropriate slow-twitch fiber ratio property for each probe. Note that a slow-twitch ratio of "-1" can be entered for muscles whose ratio of slow- to fast-twitch fibers is unknown; the default ratio of 0.5 will be used for these muscles.

Metabolic probes can be added to the Gait2392 model by following these steps:

1. **Open** the Gait2392 model in the OpenSim GUI (Models/Gait2392_Simbody/gait2392_simbody.osim).
2. Run the **addMetabolicProbes.py** script.
3. When prompted, select the file containing the slow-twitch ratios for the muscles in the model (**metabolicsSlowTwitchRatios_Gait2392.txt**).

A new model with the suffix "_probed" will be saved and loaded in the GUI. You can follow a similar process for the gait10dof9musc model. If you have a model with a different set of muscles, you can use the script as long as you create a text file containing the appropriate slow-twitch fiber ratios.

More information about the metabolic probes can be found on their doxygen page.

References

1. Umberger, B.R., Gerritsen, K.G.M., Martin, P.E. (2003) A model of human muscle energy expenditure. Computer Methods in Biomechanics and Biomedical Engineering, 6(2):99–111.
2. Umberger, B.R. (2010) Stance and swing phase costs in human walking. Journal of the Royal Society Interface, 7(50):1329–1340.
3. Uchida, T.K., Hicks, J.L., Dembia, C.L., Delp, S.L. (2016) Stretching your energetic budget: how tendon compliance affects the metabolic cost of running. PLOS ONE, 11(3):e0150378.