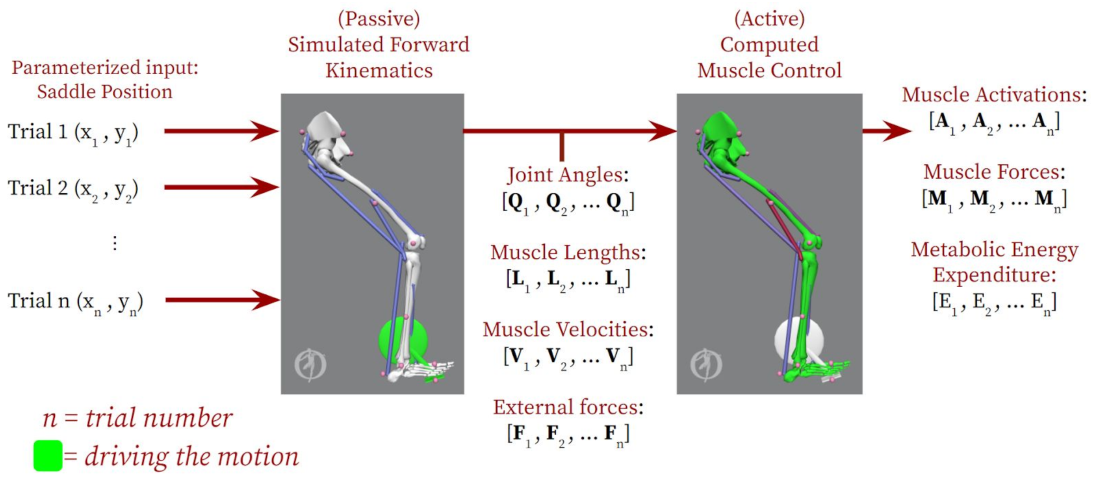
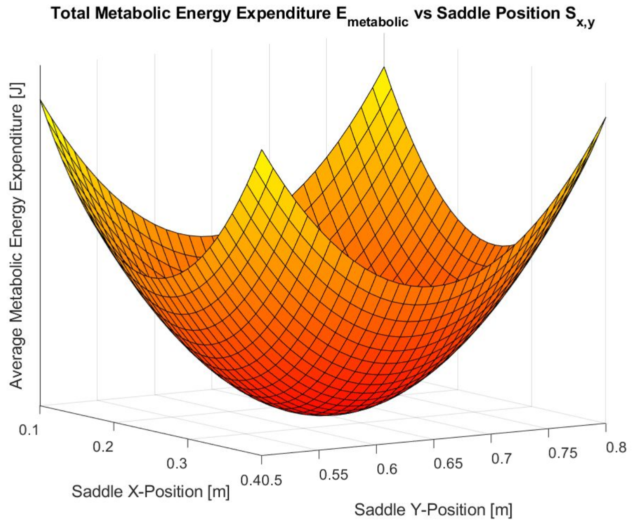
Memo: Key Aspects of OpenSim used for the 2021 ME329 Specialized Project

Summary (TL;DR)


Parameterized input: Saddle Position
Trial 1 (x₁ , y₁)
Trial 2 (x₂ , y₂)
⋮
Trial n (xₙ , yₙ)
(Passive) Simulated Forward Kinematics
Joint Angles: [Q₁ , Q₂ , ... Qₙ]
Muscle Lengths: [L₁ , L₂ , ... Lₙ]
Muscle Velocities: [V₁ , V₂ , ... Vₙ]
External forces: [F₁ , F₂ , ... Fₙ]
(Active) Computed Muscle Control
Muscle Activations: [A₁ , A₂ , ... Aₙ]
Muscle Forces: [M₁ , M₂ , ... Mₙ]
Metabolic Energy Expenditure: [E₁ , E₂ , ... Eₙ]
n = trial number
🟩= driving the motion

1. We select a saddle position $S_{x,y}$ by choosing a fore-aft distance, x, and a vertical distance, y, with respect to the bicycle bottom bracket.
2. We perform "Simulated Forward Kinematics", which is not unlike recording motion capture data in a human performance lab, to record the joint angles, muscle deformation (muscle lengths), speed of muscle contractions (muscle velocities), and forces at the midfoot for the selected saddle position (external forces).
3. The aforementioned outputs are fed into OpenSim's "Computed Muscle Control", which is an inverse dynamics solver that solves the following problem. *"Generate the desired motion with a set of muscle excitations while minimizing the deviation of simulated model kinematics from desired kinematics as well as distributing forces across all available muscles."*
4. The muscle excitation data that CMC calculates directly allows for computing of muscle forces (when paired with knowledge of muscle lengths and muscle velocities). And finally, with full knowledge of muscle forces, metabolic energy expended by every muscle in the model is known.
5. We iterate this mapping process for a number n saddle positions and generate corresponding n values for metabolic energy expended throughout the movement. We expect the final results to look something like the following figure:



Workflow Breakdown:

$T : S_{x,y} \Rightarrow E_{metabolic}$

Not a cycle but an iterative mapping process

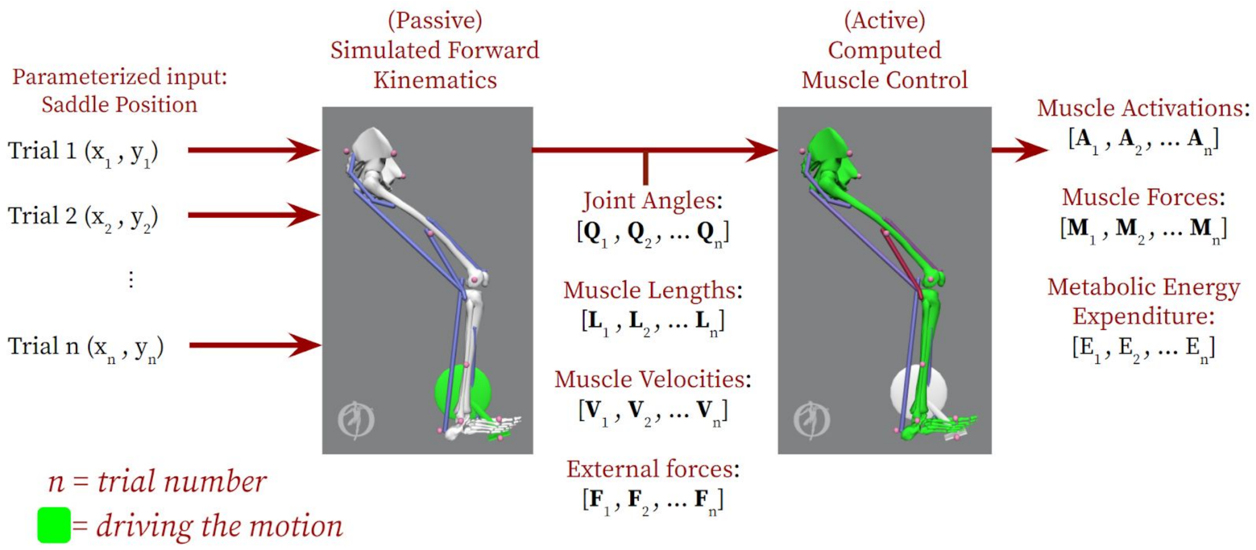

(Passive)
Simulated Forward Kinematics
(Active)
Computed Muscle Control
Parameterized input: Saddle Position
Trial 1 (x_1 , y_1)
Trial 2 (x_2 , y_2)
⋮
Trial n (x_n , y_n)
Joint Angles: [$\mathbf{Q}_1$, $\mathbf{Q}_2$, ... $\mathbf{Q}_n$]
Muscle Lengths: [$\mathbf{L}_1$, $\mathbf{L}_2$, ... $\mathbf{L}_n$]
Muscle Velocities: [$\mathbf{V}_1$, $\mathbf{V}_2$, ... $\mathbf{V}_n$]
External forces: [$\mathbf{F}_1$, $\mathbf{F}_2$, ... $\mathbf{F}_n$]
Muscle Activations: [$\mathbf{A}_1$, $\mathbf{A}_2$, ... $\mathbf{A}_n$]
Muscle Forces: [$\mathbf{M}_1$, $\mathbf{M}_2$, ... $\mathbf{M}_n$]
Metabolic Energy Expenditure: [E_1, E_2, ... E_n]
n = trial number
🟩= *driving the motion*

***Figure 1**: Iterative Mapping Process Workflow. Mapping from Seat Position to average Metabolic Energy Expenditure: $T : S_{x,y} \Rightarrow E_{metabolic}$. A saddle position is chosen to collect data for, and a single value representative of metabolic energy expenditure is computed. Simulated Forward Kinematics emulates experimental motion capture data for the selected saddle position and is used to inform the Computed Muscle Control Tool. The Computed Muscle Control calculates a set of muscle activations and forces to reproduce the desired body kinematics of Simulated Forward Kinematics, and in turn metabolic energy expenditure of the muscles for the movement is calculated.*

The flow in ***Figure 1*** summarizes the steps we are taking and the data we are outputting for each trial's simulation iteration. For every n-th trial, we select a saddle position with a specific x-coordinate (fore-aft distance) and y-coordinate (vertical distance) with respect to the bicycle bottom bracket. We then use Simulated Forward Kinematics to generate vector outputs of joint angle trajectories, $\mathbf{Q}_n$, for all of the coordinates / degrees of freedom (ie pelvis_tilt, ankle_angle, etc.) in our model, as well as muscle lengths, $\mathbf{L}_n$, muscle velocities, $\mathbf{V}_n$, and external pedal forces, $\mathbf{F}_n$. These outputs are the data we feed into the Computed Muscle Control Tool.

Computed Muscle Control (CMC) is an inverse dynamics solver which solves an optimization problem to determine the muscle activations required to perform a particular motion. The optimization problem is a least-squares error calculation which aims to minimize the error between desired kinematics (joint angle trajectories) and model kinematics due to muscle activation that are computed in real time during the simulation. The CMC Tool has a performance criterion proportional to the weighted sum of squared actuator controls on the body plus the sum of desired acceleration errors.

$$J = \sum_{i=1}^{n_x} x_i^2 + \sum_{j=1}^{n_q} w_j \left(\ddot{q}_j^{\,*} - \ddot{q}_j\right)^2 \tag{1}$$

q''^{*}_{j} = desired (input) accelerations for the j-th coordinate

q''_{j} = model accelerations for the j-th coordinate

w_j = coordinate weights

x_i = muscle excitations / actuator controls for the i-th actuator

In other words, CMC **attempts to distribute the load on the body across all available muscles** rather than having only a small subset of muscles bearing the load and generating the motion. Additionally, it always **penalizes deviations from the desired kinematics,** which are inputs into the solver and depend on the relative weighting assigned to the degrees of freedom. (More on the specific control law and files in the Computed Muscle Control section.)

From the CMC Tool, chosen actuator controls / muscle excitations, metabolic energy expenditure is calculated using muscle specific data (volume, maximum force, range of motion, etc.). The **total metabolic energy expenditure** (a scalar value, E_n) across all muscles is our desired output at the end of every n-th trial.

Figure 2 gives an example of what our input (saddle position) and output (metabolic energy expenditure) mapping may resemble with n data points corresponding to n values for total sum metabolic energy expenditure across a domain of saddle positions.

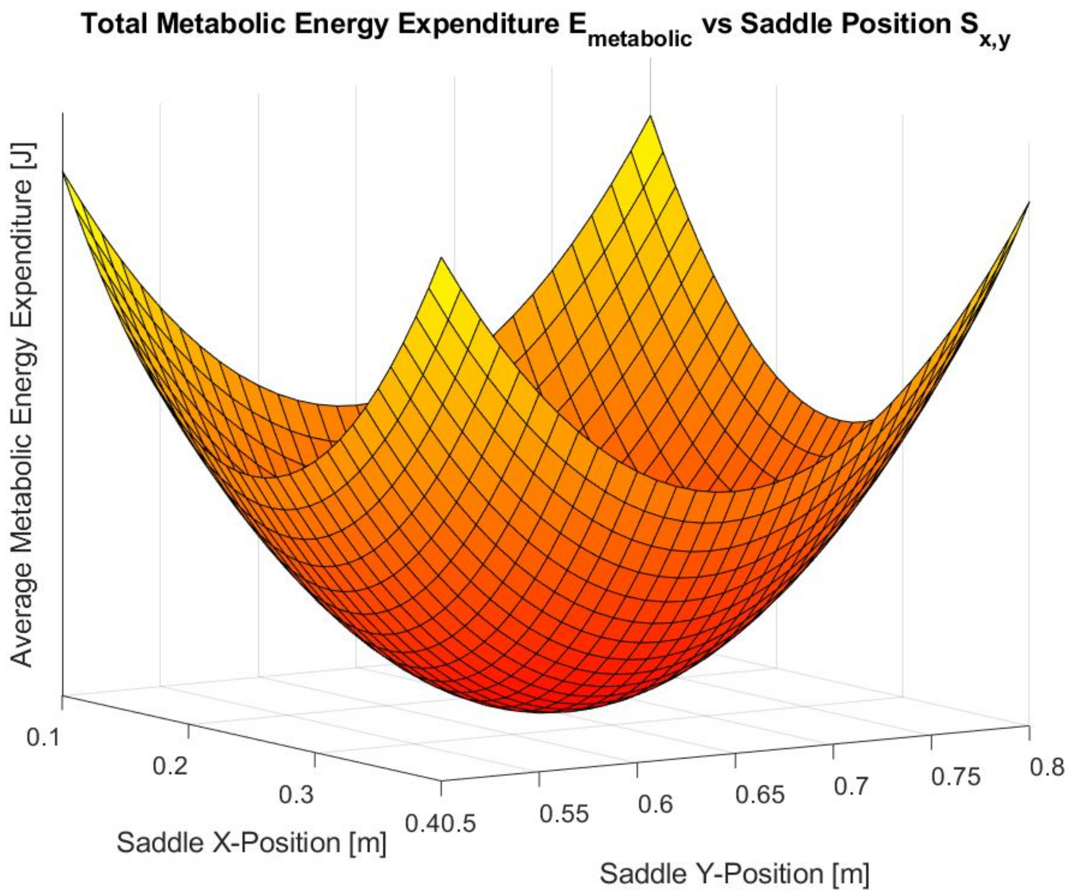


Figure 2: *Mock data output mapping of saddle position input (X,Y) and total sum metabolic energy expenditure output. This is a visualization of what we imagine our final results will be after we iterate our workflow for an n number of data points. Based on our personal experience and intuition, saddle positions that are relatively far away and very close to the bottom bracket are uncomfortable for the cyclist and result in inefficient cycling. This directly implies a higher energy expenditure required of the muscles in these saddle positions.*

Simulated Forward Kinematics

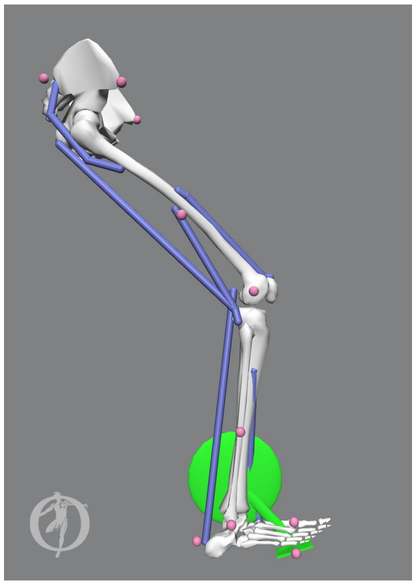


Figure 3: *During the step Simulated Forward Kinematics, the motion of the model in OpenSim is driven by a prescribed function for the crank angle. Constraints between the midfoot and the pedal of the bottom bracket as well as locking the position of the pelvis force the model to passively follow a pedaling motion as the crank angle turns. As of right now, a linearly increasing function is prescribed to the crank angle, which results in smooth, constant cadence pedaling body kinematics. This step is akin to collecting marker data from experimental motion capture with human subjects. Joint angles, muscle lengths, muscle velocities, and external forces are the data outputted by this step.*

In this project, the term 'Forward Kinematics' is very similar to what is mathematically considered forward dynamics: allowing external forces to drive the motion of the system. However, we are **not** making use of OpenSim's **Forward Dynamics Tool**, which requires input files for reaction forces and muscle excitations. Instead, we make the distinction and call this step "**Forward Kinematics**". What we do specifically for our forward kinematics is fix the pelvis in space so that it cannot translate or rotate in any manner. We constrain the position of the midfoot (ie the 'ball' of the foot) to the pedal to achieve the cycling motion. We then prescribe a function to the crank angle coordinate (rotational DOF/revolute joint), specifying a constant pedaling speed. *A prescribed function will force the crank angle to linearly increase in time, and applies all necessary forces to the body to ensure that this occurs for the duration of the simulation. (The prescribed function can be imagined as massless motors that are attached to the joints of the body to generate the specified motion.)*

The result of this setup is that *a constant cadence pedaling motion is imposed on the leg* through the crank of the bottom bracket. This directly gives us **joint angles (Q) vs time** for the entire leg. We can then use OpenSim's **Analyze Tool** to compute values such as **muscle lengths (L)**, **muscle velocities (V)**, and **external forces (F)** on the midfoot over time.

Computed Muscle Control "CMC"

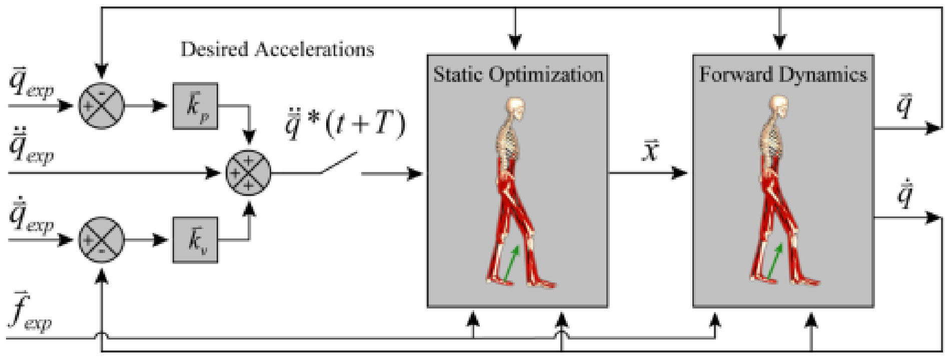


Figure 4a:
*The feedback loop OpenSim's Computed Muscle Control Tool uses to track desired joint angle accelerations. Actuator / muscle controls, **x**, are chosen following a specific performance criterion such that the simulated model kinematics track the desired joint angle accelerations.*

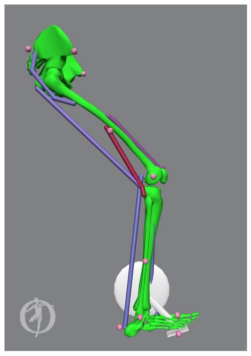


Figure 4b:
Motion generated by leg musculature instead of being driven by crank angle.

Figure 4: *During the step Computed Muscle Control, we utilize OpenSim's Computed Muscle Control Tool to calculate a set of muscle excitations that generate a set of model joint angles equal to the desired joint angle input plus or minus some finite error. Muscle excitation data, paired with knowledge of muscle lengths and muscle velocities, exactly determines muscle forces in the model and subsequently the metabolic energy expended by every muscle. Summing these energy values together results in our final desired output of total metabolic energy expended throughout the movement.*

The proportional-derivative control law employed in *Figure 4a* is given below:

$$\ddot{\vec{q}}^*(t+T) = \ddot{\vec{q}}_{\exp}(t+T) + \vec{k}_v\left[\dot{\vec{q}}_{\exp}(t) - \dot{\vec{q}}(t)\right] + \vec{k}_p\left[\vec{q}_{\exp}(t) - \vec{q}(t)\right] \tag{2}$$

Or in simpler terms:

$$\textit{Desired acceleration} = \textit{Experimental acceleration} + \textit{(Velocity gain)} \times \textit{(Velocity error)} + \textit{(Position gain)} \times \textit{(Position error)}$$

Following this control law, CMC chooses actuator controls, ***x***, for all muscles present on the body, as well as “reserve actuators”.

Aside: ***Reserve actuators*** *are objects in OpenSim that allow for imposing additional forces on the body to help follow the desired kinematics when the available muscles are not quite strong enough to provide forces to perform the desired kinematics.*

As mentioned previously, the inverse dynamics solver chooses a specific solution such that **the forces on any small subset of muscles is being minimized** in favor of distributing loads across muscles with similar lines of action. Additionally, the dynamics solver **will always penalize deviations from the desired kinematics**, which helps ensure that the muscles generate a sensible and desired motion when compared to the input kinematics.

The CMC tool carries out these optimization calculations based on the specified weighting of each of the coordinates (DOF). With each time-step in the CMC Tool run, the CMC Tool calculates a set of muscle forces to drive kinematics to “best match” (the desired kinematics from the forward kinematics simulation that is run prior to running the CMC Tool). This “best match” is solved in OpenSim via the weighted least squares optimization problem to minimize error at DOF between desired kinematics and kinematics produced by muscle forces. The weighting is specified by the user prior to setting up the CMC Tool and is a main area in which the user may need to make adjustments between CMC trials.

Muscle states

What values define the muscle state?

Biomechanical quantities extracted from our simulated forward kinematics give us insights into how efficiently the muscles are being used and are very important in understanding how the muscle force is generated. These quantities are derived completely from the kinematics and are unaffected by muscle activation, as seen in Equation (3). However, with muscle activation, which we get from the CMC simulation step, these quantities are used to calculate the force generated from the muscle-tendon unit.

The muscle force ($F^M(t)$) generated is a function of the maximum isometric muscle force (F_O^M), the muscle activation $a(t)$, and parameters from each the muscle's active force-length curve (f^L), force-velocity curve (f^V), and passive force-length curve (f^{PE}). The only parameter that depends on the muscle activation, is $a(t)$, which requires the CMC simulation step. All other terms can be derived from the muscle's intrinsic properties or form the forward kinematics simulation steps' outputs.

$$F^{\mathrm{M}}(t) = F_{\mathrm{o}}^{\mathrm{M}}\left[a(t)\,\mathrm{f}^{\mathrm{L}}\left(\tilde{\ell}^{\mathrm{M}}(t)\right)\mathrm{f}^{\mathrm{V}}\left(\tilde{v}^{\mathrm{M}}(t)\right) + \mathrm{f}^{\mathrm{PE}}\left(\tilde{\ell}^{\mathrm{M}}(t)\right)\right] \tag{3}$$

Ref: Uchida, Thomas K., et al. Biomechanics of Movement: The Science of Sports, Robotics, and Rehabilitation. The MIT Press, 2021.

Why are they useful to know? (work in progress)

There are a number of papers that we are currently reviewing that utilize knowledge of *only* muscle length and muscle velocity to draw conclusions about energy cost of a movement. We plan to mention these papers and their findings in our upcoming report, but in summary: we can use the data of muscle lengths and muscle velocities from our Simulated Forward Kinematics step to help support / validate our values for metabolic energy expenditure. According to these papers, there should be a clear correlation between the two approaches.

The OpenSim Model “.osim”

The model that we import into the OpenSim software is an “.osim” file type. It is essentially an .xml file with information on objects (skeletal geometries, other bodies), joints, controllers, constraints, forces (muscles), and markers.

Objects

Objects such as the **bones** of the model and the **bicycle components** (i.e. bottom bracket, pedal, etc.) are bodies in the .osim with mass, inertia, coordinate frames, physical meshes, and visual geometries. Any physics we have in our simulation acts on these bodies. When reporting forces in plots, force vectors are either with respect to the ground (i.e. global) coordinate frame or with respect to a body (local) coordinate frame (e.g. femur body frame).

Joints

Joints are where we define kinematic relationships between bodies. More importantly, it is where we define **coordinates** aka **degrees of freedom, DOF,** for the model. For each joint, a “parent frame” and a “child frame” are defined
(e.g. *Coordinate = Hip Angle; Parent = Pelvis, Child = Femur*). Default assignments for initial value and speed can be set for every joint, and joints can also be told to follow a smooth function when forward kinematics are being simulated. Constraints may override these default assignments. (The field is referred to as a prescribed function and, in short, it is what allows us to force the crank angle to turn at a constant rate.)

Controllers (not used in our model; included for completeness)

Controllers are not used in our model at this time. If present, controllers would define control laws in terms of proportional and velocity gains that “actuators” would follow to guide the model and simulations are run. These actuators can be thought of as invisible motors attached to the coordinates that exert torques to help achieve the desired movement.

Constraints

Constraints are where dependencies between coordinates are set, in other words, where we **couple (or otherwise relate) degrees of freedom**. This set of fields is somewhat involved and we do not need to adjust too much in here for our purposes, save for our 2-legged model exploration. One example of a constraint OpenSim is defining the forward/backward sliding of the tibia when extending or flexing the knee.

Forces

Muscles are represented as force components and thus **all muscles** included in the model are defined in this field (actuators would also be defined here if we were to utilize any for our model). Muscle objects are defined using a specific muscle object constructor (e.g. ThelenMuscle2003, Millard2012EquilibriumMuscle). Some muscle object constructors are more complex than others, but all of them contain key physiological information of the muscle being defined (see below).

All muscles require biomechanical information quantities about their **force-length** and **force-velocity** relationships, optimal fiber length, max velocity, max force output, max muscle strain, max tendon strain, and so on. These values govern the behavior of the muscles in the model. A result of how the muscles were originally defined for our model is that the muscles are "**stiff**". Essentially, an activity such as cycling is considered a **"high-flexure task**" which requires large amounts of mechanical strain on the leg muscles. When the muscles are stiff, they act not unlike preloaded springs, and thus forces carried by the muscles can become unusually large as the mechanical strain approaches the muscle's maximum.

These values for the muscles are <u>critical</u> to consider when performing simulation (whether it is forward kinematics or inverse dynamics via Computed Muscle Control). If at any point OpenSim's solver returns a solution for a muscle that violates or contradicts its physical limits, errors will be thrown, the simulation will stop, and any data collected up until that point in the simulation is unreachable through the GUI.