How CMC Works

At user-specified time intervals during a simulation, the CMC tool computes muscle excitation levels that will drive the generalized coordinates (e.g., joint angles) of a dynamic musculoskeletal model towards a desired kinematic trajectory. CMC does this by using a combination of proportional-derivative (PD) control and static optimization (figure below).

Before starting the CMC algorithm, initial states for the model are computed. The states comprise the generalized coordinates (joint angles), generalized speeds (joint angular velocities), plus any muscle states (e.g., muscle activation levels and fiber lengths). While the initial values of the generalized coordinates and speeds can be taken from the desired kinematics that you specify, the initial values of the muscle states are generally unknown. To compute viable starting muscle states, CMC is applied to the first 0.030 seconds of the desired movement. Because the muscle states are generally out of equilibrium and muscle forces can change dramatically during this initial time interval, the simulation results during this interval are generally not valid. Therefore, you should be sure to start CMC at least 0.030 seconds prior to the interval of interest.

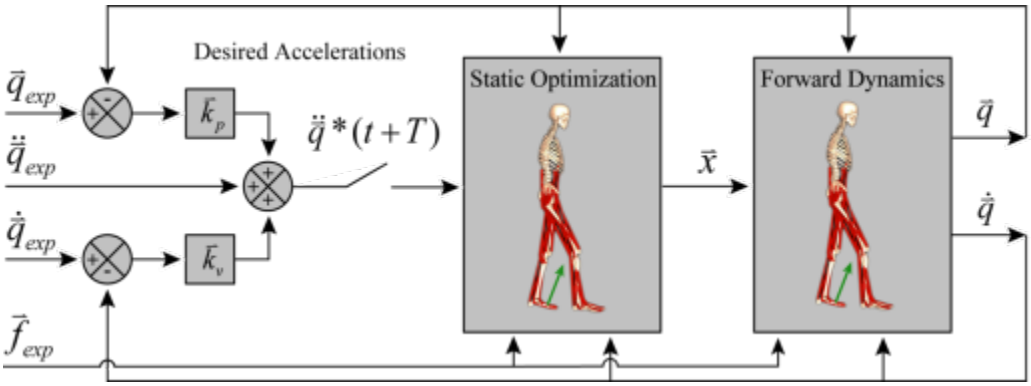

Figure: Schematic of the Computed Muscle Control Algorithm Applied to Gait [Thelen, D.G. and Anderson, F.C., "Using computed muscle control to generate forward dynamic simulations of human walking from experimental data, J. Biomech., 2006, 39(6):1107-1115]

The first step in the CMC algorithm is to compute a set of desired accelerations $\ddot{\vec{q}}^{*}$ which, when achieved, will drive the model coordinates $\vec{q}$ toward the experimentally-derived coordinates $\vec{q}_{\exp}$. The desired accelerations are computed using the following PD control law:

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

where $\vec{k}_v$ and $\vec{k}_p$ are the feedback gains on the velocity and position errors, respectively. Because the forces that muscles apply to the body cannot change instantaneously, the desired accelerations are computed for some small time *T* in the future. For musculoskeletal models, *T* is typically chosen to be about 0.010 seconds. This time interval is short enough to allow adequate control, but long enough to allow muscle forces to change.

If these desired accelerations are achieved, errors between the model coordinates and experimentally-derived coordinates will be driven to zero. To drive these errors to zero in a critically damped fashion (i.e., without over-shooting or over-damping), the velocity gains can be chosen using the following relation:

$$\vec{k}_v = 2\sqrt{\vec{k}_p}$$

For musculoskeletal models, it works well if the error gains are chosen to drive any errors to zero slowly. The error gains $\vec{k}_v = 20$ and $\vec{k}_p = 100$ will reduce tracking errors.

The next step in CMC is to compute the actuator controls $\vec{x}$ that will achieve the desired accelerations $\ddot{q}^{*}(t+T)$. Most of the time, the controls are predominantly comprised of muscle excitations, but this is not required. Any kind of actuator can be used with CMC (e.g., idealized joint moments). Static optimization is used to distribute the load across synergistic actuators. It is called "static" optimization because the performance criterion (i.e., the cost index) is confined to quantities that can be computed at any instant in time during a simulation. Using criteria like jump height or total metabolic energy over a gait cycle, for example, are not possible because these require simulating until the body leaves the ground or until the end of the gait cycle is reached.

Two formulations of the static optimization problem are currently available in CMC. The first formulation, called the slow target, consists of a performance criterion J that is a weighted sum of squared actuator controls plus the sum of desired acceleration errors:

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

The first summation minimizes and distributes loads across actuators and the second drives the model accelerations $\ddot{q}_j$ toward the desired accelerations $\ddot{q}_j{}^{*}$.

The second formulation, called the fast target, is the sum of squared controls augmented by a set of equality constraints $C_j = 0$ that requires the desired accelerations to be achieved within the tolerance set for the optimizer:

$$J = \sum_{i=1}^{n_x} x_i^2$$
$$C_j = \ddot{q}_j{}^{*} - \ddot{q}_j \qquad \forall j$$

The fast target is both faster and generally produces better tracking. However, if the constraints cannot be met, the fast target will fail and CMC will exit with an error message. Often the reason for the failure is that the musculoskeletal model is not strong enough.

To prevent the fast target from failing, it is possible to add a number of *reserve actuators* to a model that are able to make up for strength deficiencies in muscles should any be encountered. The reserve actuators have very low strength (or optimal force) and so require very high excitations to apply substantial load to the model. As a result, use of the reserve actuators is highly penalized in both the fast and slow formulations. When the forces (or moments) applied by the reserve actuators are written to file and plotted, their values are a good indication of which joints in the musculoskeletal model are not strong enough. If the model is strong enough, the reserve actuator forces/moments should generally be very small relative to the forces/moments applied by the primary actuators.

The final step in the CMC algorithm is to use the computed controls to conduct a standard forward dynamic simulation, advancing forward in time by T. These steps—computing the desired accelerations, static optimization, and forward dynamic simulation—are repeated until time is advanced to the end of the desired movement interval.

Once CMC finishes execution, you will typically want to compare the computed muscle excitation patterns with prototypical or measured electromyographical measurements. If desired, constraints can be placed on the upper and lower bounds of the controls $\vec{x}$ as a function of simulation time. The bounds on the controls $\vec{x}$ are specified in an XML input file. For muscle excitations, the default upper bound is typically 1.0 (full excitation), and the default lower bound is typically a small number just above 0.0 (no excitation), such as 0.01 or 0.02. The lower bound is not set at precisely 0.0 because mathematical models of muscle are often not as well-behaved when excitation goes all the way to 0.0. The format of the control constraints is shown in the following section.

You can view the excitations produced by CMC using the Excitation Editor.

Next: How to Use the CMC Tool

Previous: Getting Started with CMC

Home: Computed Muscle Control