Examples of Muscle Activation-Contraction Dynamics—Revised

Excitation dynamics

Here are some Matlab functions which integrate the first-order differential equation describing muscle excitation, that is, the dynamics between neural input to a muscle and its muscle activation.

Here’s a function which provides the state derivative:

```matlab
function adot = excit(t,a);
  % provides state derivative for excitation dynamics of muscle
  u = input_func(t);
  tact = .010;    % activation time constant
  beta = 0.2;     % ratio of activation to deactivation constants
  adot = -1/tact * (beta + (1 - beta) * u) * a(1) + 1/tact * u;
```

And a function which provides a square-wave input between 0.2 and 0.25 sec:

```matlab
function u = input_func(t);
  % provides input waveform as function of time
  u = 0;
  if t > 0.2 & t < 0.25,
    u = 1;
  end;
```

The following commands in Matlab integrate the differential equation from t = 0 to t = 0.5, and plot the results:

```matlab
[t,a]=ode45('excit',0,.5,0);
for i=1:length(t), uvec(i) = u(t(i)); end; % construct an input vector
plot(t,a,'-',t,uvec,':') % plot a with solid line, u with dotted
```

And here are the results:

![Graph](image)

Notice that the time constant for activation is much faster than that for deactivation.
Kick simulation

Consider a simplified model of the quadriceps muscle group, in which the patellar tendon is assumed to be very stiff. The following functions are used to generate a simple simulation of a kick of the lower leg, with the upper leg held fixed at a level position. The simulation starts with the knee at 90° and shows how it moves given full muscle activation.

```matlab
function f = fl(x);
% normalized force-length curve
% input x is normalized wrt optimal muscle length
w = 0.5;
Lopt = 1; % this is a normalized optimal length
f = 1 - ((x-Lopt)/(w*Lopt))^2;
if f < 0, f = 0; end;

function f = fv(v);
% normalized force-velocity relation
% input is velocity normalized wrt maximum shortening velocity
af = 0.25; % shape parameter
f = (1 - v)./(1 + v/af);

function xdot = kick(t,x);
% state derivative for kicking simulation
% states: phi, phidot where phi is angle of leg
phi = x(1);
phidot = x(2);
I = 0.1832; % moment of inertia of lower leg (kg-m^2)
m = 4.88; % mass (kg)
g = 9.81; % grav constant
lopt = 0.09; % optimal muscle length in meters
vmax = 0.45; % about 6 lopts/s max shortening vel
Fmax = 12000; % max isometric force (N)
rf = 0.033; % moment arm of quadriceps muscle
rcm = 0.264; % distance b/w center of mass & joint
a = 1; % maximum activation
l = rf*(pi/2 - phi)/lopt + 1; % find normalized length as function of phi
v = rf*phidot/vmax; % normalized shortening velocity
F = Fmax * fl(l) * fv(v) * a; % muscle force
M = F * rf;
Mg = -m * g * rcm *sin(phi - pi/2); % gravitational moment
phiddot = (M + Mg) / I; % second derivative of phi
xdot = [phidot; phiddot];
```

The results from the simulation are plotted below. Notice that the muscle quickly reaches close to its maximum shortening velocity and reaches a fairly short length, and that muscle force decreases quickly.
Helpful functions for simulating muscle contraction

An arctangent function is sometimes used for the force-velocity curve. Here is a sample function which inverts that relationship:

```
function v = fvi(f);
% provides inverse of force-velocity relation:
% force in, velocity out
% force is normalized wrt max isometric force,
% velocity is normalized wrt maximum shortening velocity
% c1 = -0.18713;
% c2 = 0.32094;
% c3 = 1.06485;
% c4 = 0.1850;
% v = -c1*cot(c2*f.^2 + c3*f + c4);
if f > 1.4, v = -.15; end;   % make sure function is defined above f = 1.4
```

Here is a sample function for the passive muscle force-length curve:

```
function f = flp(x);
% normalized passive force-length curve for muscle, where
% input x is normalized length wrt optimal muscle fiber length
f = 8*(x-1).^3;
% below is a trick which works for vector inputs; ignore if it’s confusing
f(x < 1) = 0*find(x < 1);   % set to zero below x = 1
```
Simulating muscle contraction

There are several options for setting up a muscle simulation with tendon. The tendon elasticity adds an additional state to the equations, and there are a number of possible choices. For example, the muscle force itself can be a state. In this case, it is necessary to solve for the velocities of the muscle and tendon. Assuming normalized versions of the force-length and force-velocity curves:

\[
\ell_T = f^{-1}_SE\left(\frac{F_T}{A_T}\right), \ell_{slack}^T
\]

\[
\ell_{MT} = \ell^M + \ell^T
\]

tendon stretch found by inverting the normalized tendon force-length curve

muscle length can be solved for from the total muscle-tendon length (defined as muscle + tendon stretch, neglecting tendon slack length)

\[
\frac{v^M}{v_{max}^M} = f_v^{-1}\left(\frac{F_T}{F_{max}^M \cdot f_\ell(\ell^M/\ell^M_0)}\cdot a\right)
\]

muscle shortening velocity can be found by inverting the force-velocity curve. When the denominator of the input goes to zero or when the muscle force goes below zero, set \(v^M\) to -0.15, which is approximately its yield velocity

\[
v^T_{s} = v^M_{s} + v^T_{s}
\]

muscle shortening velocity and tendon shortening velocity are related to overall shortening velocity the state equation for muscle force—note that if muscle force is below 0 then make sure \(F^T\geq 0\), because muscles cannot push. Also note that this equation makes use of the slope of the tendon force-length curve.

\[
\frac{dF_T}{dt} = \frac{dF_T}{d\ell} \cdot \frac{d\ell}{dt} = -\frac{dF_T}{d\ell} \cdot v^T
\]

Muscle or tendon length can also be chosen as states, and the constitutive equations, compatibility conditions, and boundary conditions must be manipulated to arrive at the corresponding state equations.