BIOE 302 – Modeling Human Physiology: Muscle Dynamics Model

Manuel Alejandro Ramírez García B.S.
Bradley P. Sutton Ph.D.
Department of Bioenginering

Table of Contents

Simulink Model

The Simulink model we are going to present uses microscopic contractile processes known as crossbridge dynamics to explain the macroscopic muscle mechanics as understood by A.V. Hill's Force-Velocity curve. It is loosely adapted from Hoppensteadt and Peskin (2002) crossbridge dynamics model. The Simulink model crossbridge.mdl calculates the attachment and detachment of crossbridges and looks at the fraction of attached crossbridges over time, the total force they generate as well as the distribution of crossbridges with varying displacements as a population density function. All initialization values with units are preloaded into the Simulink file through the callback function in model properties, and are also available in the parameters_crossbridge.m file. The population density is calculated with the popdensiy.m MATLAB function.

The events leading to the contraction of skeletal muscle are understood to be part of the excitation-contraction coupling mechanism. Action potentials travel through motor neurons, and join muscles at the neuromuscular junction. The chemical synapse allows for action potentials to travel down muscle fibers.

Muscle cells are long and multinucleated. Skeletal muscles contain many myofibrils which are striated, giving it a banded appearance due to the thick and thin filament organizational structure. Thick and thin filaments arranged longitudinally and cross-sectionally in units known as sarcomeres. Myofibrils are also surrounded by a sarcoplasmic reticulum containing calcium ions and are crisscrossed by transverse tubules (T-tubule) which can be depolarized by action potentials. The events leading to the contraction of skeletal muscle are known as the excitation-contraction coupling mechanism.

The basic unit of muscular function is the sarcomere. The thick filaments are made up of myosin, and the thin filaments of actin, tropomyosin and troponin. Myosin has two “heads” and a “tail” region. The “head” of the myosin protein has an actin-binding site, necessary for crossbridge formation. Actin has various myosin binding sites however, at rest, the binding sites are covered by tropomyosin preventing actin and myosin from interacting. Troponin is a complex of three proteins, Troponin T (for tropomyosin), troponin I (for inhibition) and Troponin C (for calcium).[2] Along with tropomyosin, Troponin I and Troponin T are bound to each other and at rest, prevent the myosin binding site in actin to be exposed.

In the excitation-contraction coupling mechanism, when an action potential is propagated through the T tubules, they depolarize causing a cascade allowing the calcium ions stored in the sarcoplasmic reticulum to be released. The binding of calcium ions to Troponin C causes a conformational change in the troponin complex moving tropomyosin out of the way allowing the myosin heads to bind to the actin molecules forming a crossbridge.

The interactions between myosin and actin are part of the crossbridge cycling mechanism. Two products of these interactions are the generation of force through the power stroke, and the movement of the thin filaments relative to the thick filaments (known as sliding) in the sarcomere. In the sarcomere, the actin filaments extend from the edges (Z-line) to the beginning of the H hand. The myosin proteins span across the A band, beginning from the center of the sarcomere (M line). The heads of the myosin protrude towards the actin filaments.

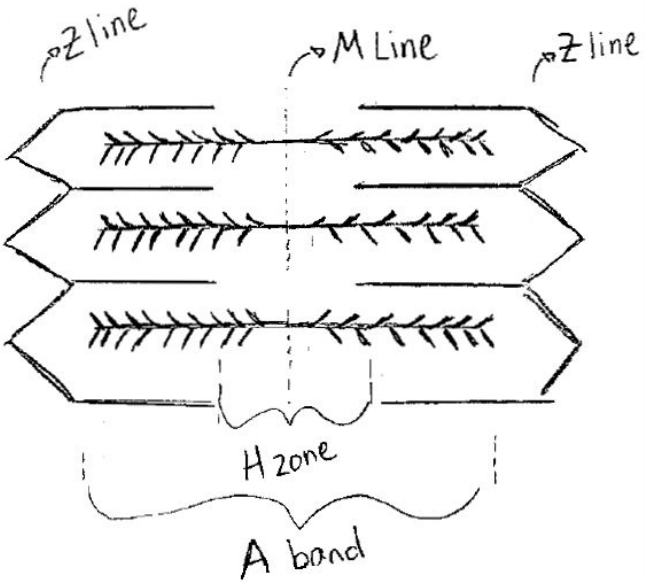

Figure 1: The sarcomere unit. The Z line denotes the outer boundaries of the sarcomere. The A-band contains the overlapping myosin thick filament and actin thin filament area. The H-zone contains only thick filaments, thus no crossbridges can be formed as there is no actin for the myosin to bind to. The M-line is the middle of the sarcomere.

In our simulation, we look at various components of muscle dynamics. We create 10,000 crossbridges and randomly determine if they are attached or detached through an algorithm based on the MATLAB code provided by Hoppensteadt and Peskin (2002). As mentioned previously, as the thick and thin filaments undergo crossbridge cycling the thin filaments move in relation to the thick filaments, and we track the displacement of the attached crossbridges (myosin bound to actin) over time. The displacement of a myosin head from its equilibrium position will determine the force that it contributes to the contractions. We are interested in the total force that all the crossbridges generate over time. Ultimately, we are concerned with the amount of force developed in a muscle undergoing a constant shortening velocity. The relationship between muscle shortening velocity and force generated by muscles is known as the Hill Force Velocity Curve (Figure 2).

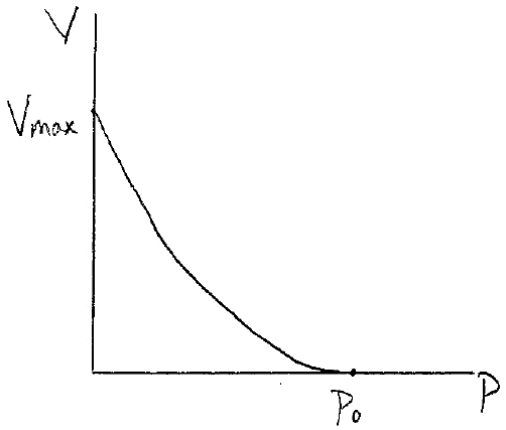

Figure 2: A.V. Hill's Force Velocity Curve. On the vertical axis we have Shortening Velocity where V_{max} denotes the maximum muscle shortening velocity. On the horizontal axis we have muscle force P where P_0 denotes the isometric force when the velocity is zero.

This model is unusual in Simulink because it doesn't have any differential equations for Simulink to solve. Instead, it takes advantage of the fixed time steps to propagate attachment/detachment/displacements of a vectorization of the problem, instead of relying on separate MATLAB programs to propagate attachment/detachment and generation of force, as in Hoppensteadt and Peskin (2002). Without underlying differential equations in the Simulink system, this approach only offers improved visualization of the mathematical process.

The model analyzes crossbridge dynamics by having them attach and detach randomly, while displacing with the fixed muscle shortening. In the end, we take a look at the fraction of attached crossbridges *U*, the population density of attached crossbridges with respect to displacement from equilibrium and the total force generated by the attached crossbridges.

In the simulation there are two main states, which correspond to the two main vectors we use. The vectors are of size n_0x1, where n_0 is the number of crossbridges in a half-sarcomere, the unit that is being simulated. The first state, held in the vector *a*, is for keeping track of whether a crossbridge is attached or detached. An attached crossbridge is represented with a value of 1, while a detached crossbridge is represented with a value of 0. The second state corresponds to the vector *x*, the displacement of a crossbridge from its equilibrium position. This state tracks the movement and locations of all the crossbridges whether attached or detached, as determined by vector *a*. We will see that unattached crossbridges do not have a displacement from equilibrium and their displacements will be ignored.

The process of attaching and detaching crossbridges is random, and occurs in the subsystem "calculating a". The process of determining displacements of crossbridges, occurs in subsystem "calculating x".

To determine the fraction of attached crossbridges *U*, we take the sum of the vector *a* and divide by the total number of crossbridges, n_0.

$$U = \frac{\sum a}{n_0} \qquad (1)$$

To find the population density of the crossbridges *u*, we created a MATLAB function called popdensity, similar to the function used in Hoppensteadt and Peskin. The function utilizes the histogram function and assigns bins for each location of a crossbridge and plots the density of crossbridge in each bin. It requires three inputs: the vector *a*, the vector *x*, and the number of bins you wish to plot in (a quantity which can be changed in the parameters_crossbridge.m file.

Finally, to calculate the total force generated by the sarcomere, we sum the individual attached crossbridges forces together. For each crossbridge, myosin will attach to an actin filament with its head extended. After attachment, the head ratchets to generate force. We will consider that a myosin head has an equilibrium displacement for its attachment position at x=0, where it will not create any force. For displacements larger than this, such as when a cross-bridge is first formed, there will be a large force generated where the force is dependent on the displacement. A simply model would be a spring, where the force ,p, of a crossbridge might be related to displacement, x, as p=kx. A cross-bridge will form with a positive displacement x=A, and will stay attached, displacing with the fixed shortening velocity of the muscle. When crossbridges stay attached, the displacement can go negative. This will result in a pushing back force from the cross-bridge, subtracting from the force generated by the muscle.

To get the force for the whole muscle, indicated by P, we sum up the forces of the individual crossbridges, given by p(x). Instead of a simple spring, we use the crossbridge force model from Hoppensteadt and Peskin.

$$p(x) = p_1 \times (e^{\gamma x} - 1) \qquad (2)$$

$$P = \int_{-\infty}^{A} p(x)u(x)dx \qquad (3)$$

Note that the integral for the force only goes up to A as a maximum, this is because crossbridges form at a displacement of x=A due to the ratcheting myosin head and the muscle shortens from there. No crossbridges are attached at a distance greater than x=A.

In our model, we assume the crossbridge population is at steady state. We assume crossbridges are only formed at a displacement of A. The crossbridges are formed at a rate of α per unattached bridge and break at a rate β per attached bridge, independent of displacement. Crossbridges also slide in the direction of decreasing x at the fixed muscle shortening velocity of V (Figure 3). V_0 is the macroscopic shortening velocity of the whole muscle in nm per second. Because half-sarcomeres in series add displacements, the relationship between the velocity of shortening of a half-sarcomere V and the whole muscle shortening V_0 is such that: $V_0 = N*V$, where N is the number of half sarcomeres in series in the muscle.

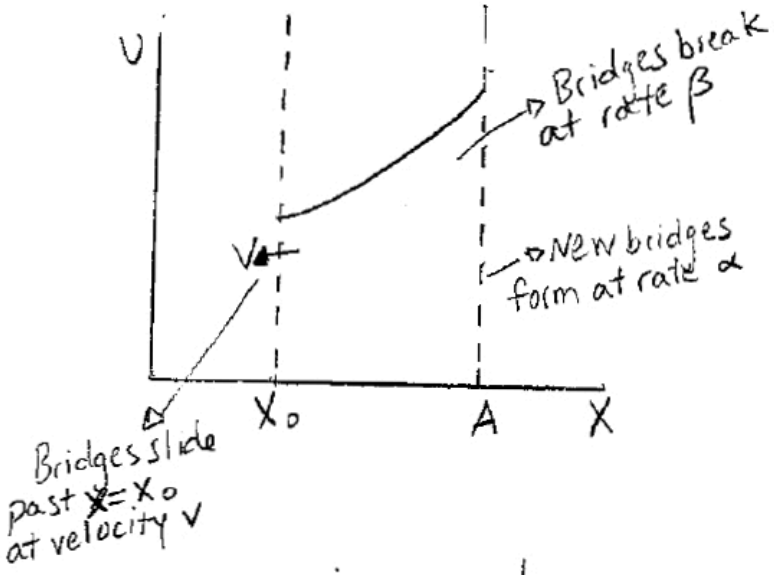

Figure 3: Dynamics of the crossbridge population density u(x) for a control volume between x_0 and A . Within this differential element, as steady state, crossbridges are formed at the same rate that they either break or leave the control volume. Crossbridges are formed at x = A at a rate α. Crossbridges break at all displacements at a rate β. And crossbridges at a displacement of x=x0 will displace with the fixed muscle shortening velocity v and leave the control volume.

Crossbridge Dynamics

Our Simulink model revolves around two main subsystems, Calculating a and Calculating x. Crossbridge formation and destruction is performed in Calculating a and crossbridge displacement is performed in Calculating x. The subsystems create the necessary vectors to solve Equations 1 and 2. Equation 1 finds the population of attached crossbridges *U* by finding the sum of the attached crossbridge and dividing it by the number of crossbridges. In our model we use a Sum of Elements and Gain block. Equation 2 calculates the total force generated by the crossbridges over time. In our model we find the force generated by each crossbridge in the p(x) subsystem (Figure ...) and add them together with a Sum of Elements block. The main window of the Simulink file is shown below.

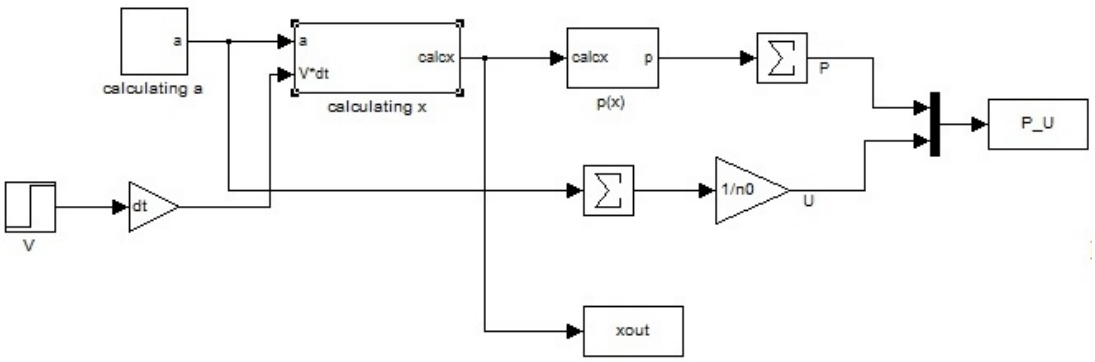

Figure 3: Main Window of Crossbridge.mdl.

In a half-sarcomere, there are a given amount of available crossbridges that can attach to cause muscle contraction. We initialize vectors *a* and *x* in the MATLAB workspace as zero vectors with a length equal to the total number of available crossbridges in a half-sarcomere.

The first step of our simulation is to determine the state of our crossbridges as attached or detached. In the Calculating a subsystem, we determine which crossbridges will detach, attach, or remain detached/attached. We do this by the two subsystems labeled Detachment and Attachment found within the Calculating a subsystem.

To help visualize the process for crossbridge attachment/detachment, it is useful to think of each crossbridge as a step function, where a value of 1 is reserved for an attached crossbridge, and a value of 0 for a detached crossbridge.

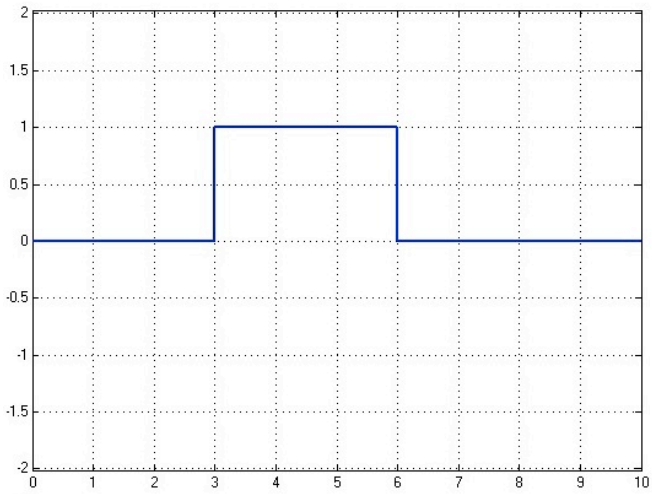

Figure 4: a vector seen analogously to a(t), a unit step function where a value of 1 signifies attachment, and 0 detachment. In this example, the crossbridge was attached from 3 to 6 seconds, and detached elsewhere.

If we have a(t) as the state of crossbridges over time, what would the change of crossbridge states over time, da/dt, represent? From our analogy, there are four possible values and therefore scenarios for da/dt. A detached crossbridge could attach, an attached crossbridge could remain attached, an attached crossbridge could detach, or a detached crossbridge could remain detached.

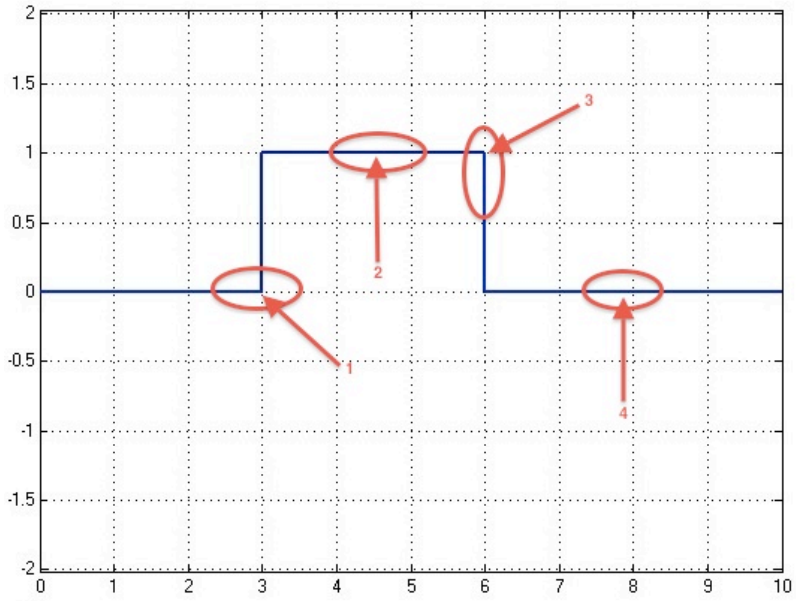

Figure 5: a(t) step signal annotated with 4 situations. Situation 1) Crossbridges changes states from detached to attached, 2) Crossbridge remains attached, 3) Crossbridge changes states from attached to detached, 4) Crossbridge remains detached

If we think of the changes in a(t) in these four situations, we can think of its derivative da/dt. If a crossbridge changes state from detached to attached we would expect a positive value from its derivative, if a crossbridges changes state from attached to detached we would expect a negative value from its derivative, and if a crossbridge remains attached or detached we would expect no change in its derivative. If we think in terms of slopes and a(t), we have a positive slope, for a positive derivative, a negative slope for a negative derivative, and a zero slope for no change. Graphically, this is shown in Figure 6.

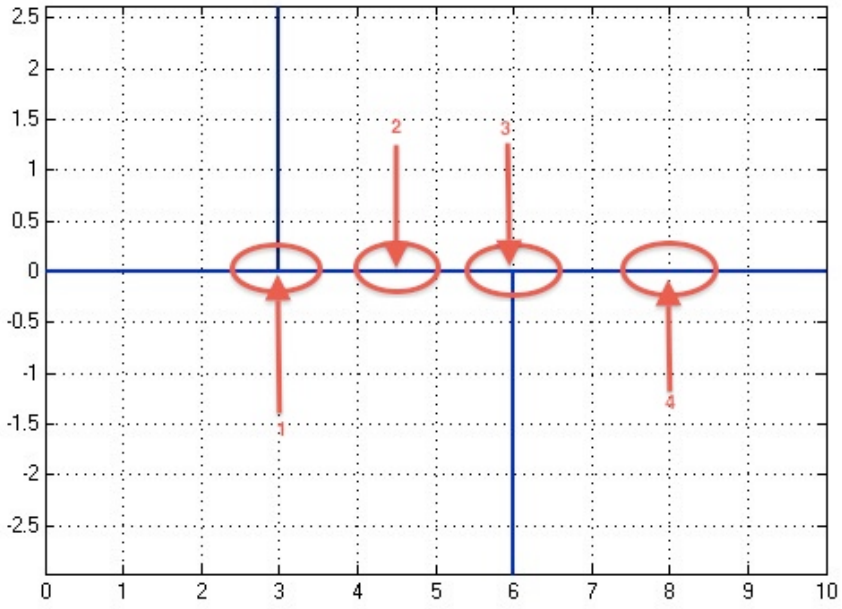

Figure 6: da/dt with annotations. Scenarios 1-4 match those in Figure 5. The values

for scenarios 1 and 3 are non-zero, and the values for scenarios 2,4 are zero.

We have now reached a fundamental concept for our Simulink model. Remember, we utilize differential elements (the integrator and derivative block) as a method to vectorize, not to solve differential equations. The *a* vector can hold two states, attachment (with a value of 1) and detachment (with a value of 0), while its derivative da/dt represents the change of states with three values. A change from detached to attached (with a positive value) a change from attached to detached (with a negative value) and no change in state (with a value of zero). Understanding this concept is critical to grasp the operations performed in the Calculating A and Calculating x subsystems.

"Calculating a" Subsytem

We initialized all crossbridges as a zeros vector, meaning they are detached and therefore, available for attachment. The initialized vector is propagated to the Detachment subsystem and attached subsystem. Before the vector reaches the Attachment subsystem, the signal is subtracted from 1. This subtraction determines how many crossbridges are available for attachment. Remember, an attached crossbridge holds a value of 1, if a crossbridge was attached, the input into Attachment would be 0, meaning no attachment can happen for that crossbridge.

The attachment and detatchment blocks will output either a 1 (attachment block indicated attachment has occurred) or a -1 (detachment block indicating a detachment has occurred, but never for the same entry in the vector. We then update our attachment state vector, *a*, by integrating the sum of these outputs with an initial condition of the current state vector, *a*. You will notice that we multiply our output of the attachment/detachment summation by a scaling factor of 1/dt, this is to create a change of 1 in our integration.

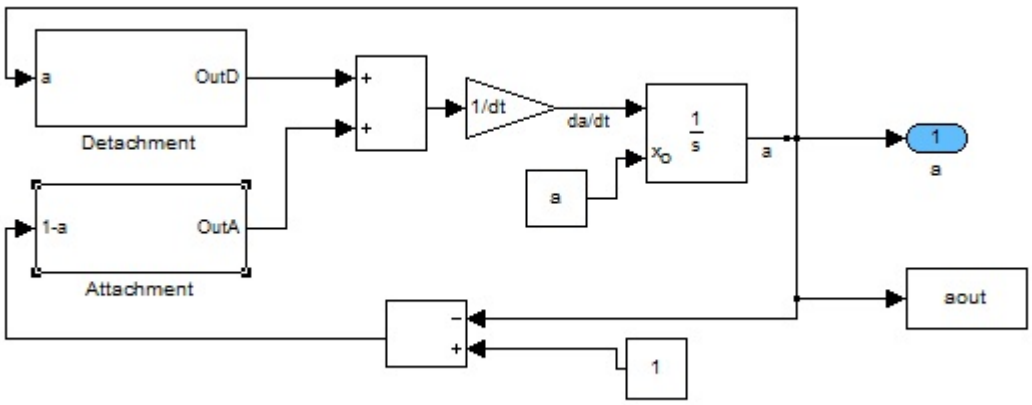

Figure 7: Calculating a subsystem. The initialized zeros vector is found in the constants block a. The relationship between a and da/dt can best be understood with the step function analogy in Figures 4,5,6

Attachment Subsystem

For each of the crossbridges available for attachment, a random number generator generates a Uniform Random Number in the range from zero to one. This number is compared to the small probability of attachment, using a compare to constant block. The probability of attachment is $\alpha \times dt$ where α is the probability of attachment per unit time and dt is the duration of our timestep in the simulation. The result of the comparison is a logical value of zero or one, where zero means the number generated by the Uniform Random Number block was greater than the probability of attachment, and therefore will not attach, while a 1 means the opposite. The logical values are converted to doubles using a data type conversion block.

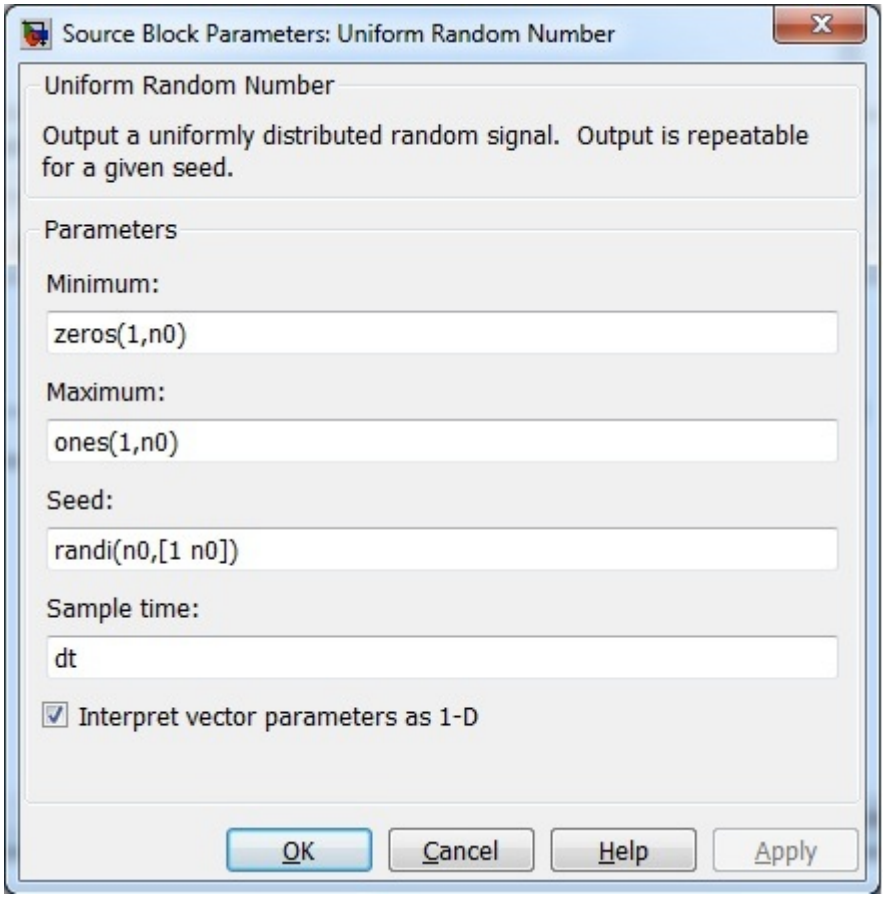

Figure 8: Attachment Subsystem Uniform Random Number Parameters window.

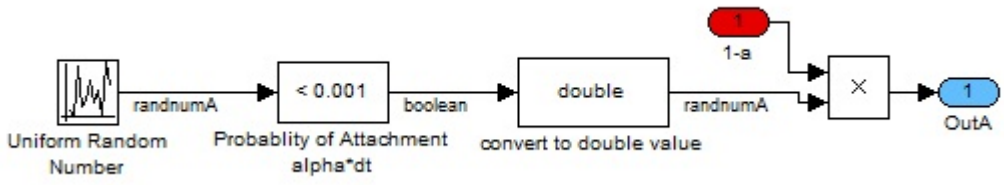

Figure 9: Attachment Subsystem in Calculating a subsystem

Finally, the (1-*a*) vector signal is used to mask the output attachment vector by multiplication. If a crossbridge is available for attachment, then the element in the *a* vector will be a 1, if it is not available for attachment it will result in multiplying by 0. This generates a vector called OutA of zeros and ones where zero indicates no change has occurred in crossbridge attachment, and one indicates that a detached crossbridge has attached.

Detachment Subsystem

From the main Calculating a subsystem the integrated *a* vector signal is similarly propagated to the Detachment subsystem. In the Detachment subsystem, a very similar process occurs as in the Attachment subsystem. A Uniform Random Number block generates a vector signal (let's call it randnumD) with the same parameters as its attachment counterpart. The signal is compared using a compare to constant block to the small probability of detachment β×dt where β is the probability per unit time for detachment and dt is the duration of our timestep in the simulation. The outcome is a vector signal of zeros and ones where zero means the number generated by the Uniform Random Number block was greater than the probability of detachment, and therefore will not detach, while a one means the opposite. The logical values are converted to doubles using a data type conversion block.

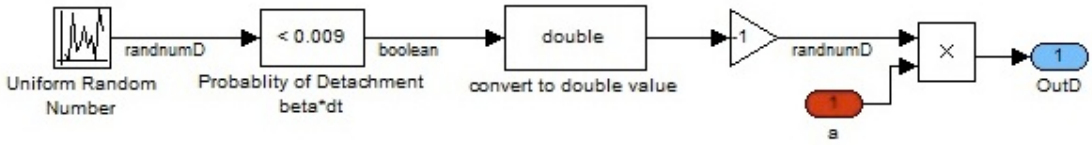

Figure 10: Detachment subsystem in Calculating a subsystem

Afterwards, the result is multiplied by negative one to indicate the differential signal of detachment when compared to the state vector *a*. This leaves randumD as a signal of zeros and negative ones where zero indicates no detachment, and negative one indicates detachment. Again, the state vector a is used to mask the detachments to only attached crossbridges by multiplying randumD by the *a* vector signal. This generates a vector signal OutD with values of zeros and negative ones, where zero indicates no detachment is to occur, and negative one means a detachment should occur.

In conclusion, for OutA, we have values of zeros and ones. A value of zero indicates a remainder in the detached state or that the crossbridge was already attached, while a value of one indicates a switch from a detached to an attached state. For OutD, we have values of zeros and negative ones. A value of zero indicates a remainder in the attached state or that a crossbridge was already detached, while a value of negative one indicates a switch from an attached to detached state. We have now determined the four values for all situations of crossbridge cycling, as explained previously.

Note about random numbers in Simulink

Before we leave the attachment and detachment modules, we will say a few words about random number generators. Much like in MATLAB, in Simulink random numbers are not truly random, but are generated as a fixed series of numbers via an algorithm. They are based on a seed, which tells the random number generator where to start generating numbers. In reality the random number generators are pseudo-random as there is a defined process that determines the next value each time. For this reason, we set the seed in our subsystem to be determined from a set of random integers.

da/dt

Afterwards, the OutA and OutD signals are added together, divided by dt and then integrated (Figure 5). The integrations that occur in the Calculating a Subsystem for da/dt will find the area in the changing slope of a(t). This area is base*height, where base is dt and height is 1. However, we want the result of the integral to be 1, so we divide by dt beforehand with a gain block.

Viewing the vector a and its "derivative" in this manner helps understand the next subsystem, where the displacements of the crossbridges are determined.

"*Calculating x*" Subsystem

For each state of a crossbridge, there are four situations which lead to that result, as outlined above. Now, we must determine what the displacement and locations for each crossbridge should be.

Crossbridges are only formed at the arbitrary value of x = A at a rate of α per unattached bridge, and they break at a rate β per attached bridge. Crossbridges slide at a velocity V where V is the macroscopic shortening velocity (nm/s) of the whole muscle divided by the number of half-sarcomeres in series along the muscle.

For each attached crossbridge remaining attached, it will displace a distance of V×dt. Detached crossbridges will remain at their location of x = 0. For a crossbridge attaching it's new location will be x = A, and if detaching its new location must be set to x = 0. This can be described by a decision tree for how to update the displacement of a crossbridge, based on the state vector a and the time derivative to indicate changes of state.

$$\text{When a=1} \begin{cases} \frac{da}{dt} = 0 \rightarrow \text{ remains attached, must displace by V*dt} \\ \frac{da}{dt} = 1 \rightarrow \text{ changed states from detached to attached, set x=A} \end{cases}$$

$$\text{When a} = 0 \begin{cases} \frac{da}{dt} = 0 \rightarrow \text{remains detached, keep x=0} \\ \frac{da}{dt} = -1 \rightarrow \text{changed states from attached to detached, set x=0} \end{cases}$$

To accomplish the selection of each choice in vector *x*, we utilize three switch blocks in the Calculating x subsystem (Figure 11). The switch on the leftmost side of the subsystem, uses the value of vector *a* as the criteria for passing the first input, related to changes in displacement for the attached crossbridges. The first input of the switch will be triggered if the second input (the *a* vector) is non-zero. If it is zero, then the third input of the switch will be selected, related to changes in displacement for the detached crossbridges.

The next set of switches deal with the two subdivisions generated by da/dt. Like the previous switch bock, the switches first input will be triggered if the second input (in this case da/dt taken by the derivative block of the imported *a* vector) is non-zero. If it is zero, then the third input of the switches will be selected. The selections for the switches can be followed from the description above.

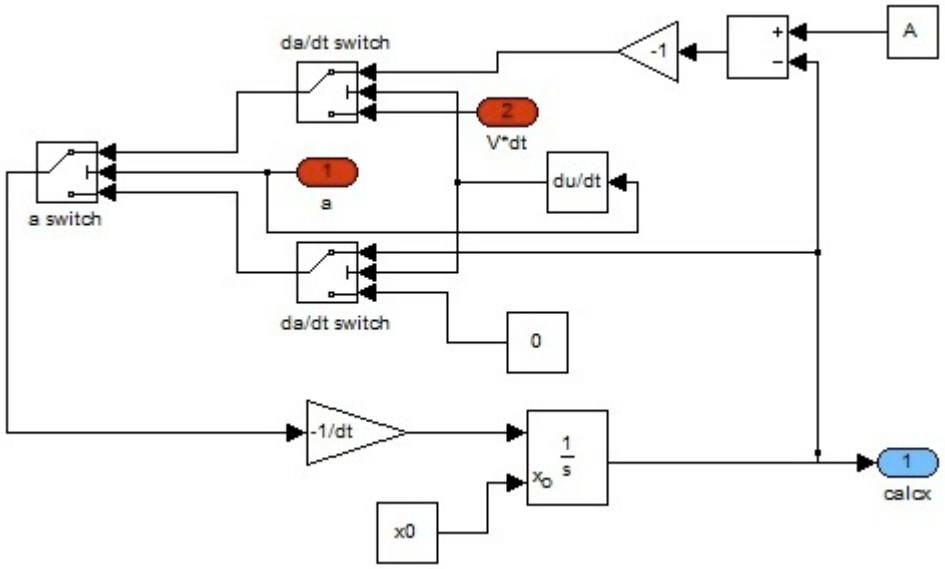

*Figure 11: "Calculating x" Subsystem. In each switch block, the second (middle) input is used as the criteria to choose between the first (top) and third (bottom) inputs. In each switch block, the first input will be selected if the second input is non-zero. The red a and V*dt blocks represent import blocks. The a vector is imported from the Calculating a subsystem, and the V*dt is calculated in the main Window.*

Velocity of shortening:

Our simulation will have two segments in time. We start with a muscle at rest, velocity, V, of 0 nm/s. Then at some time, T_{start}, we impose a fixed shortening velocity on the muscle as it contracts. To achieve these two different velocity regimes in the simulation, we utilize a step block multiplied by a gain block found in the main Simulink window of crossbridge.mdl (Figure 3). The step block has a step time of T_{start}, an initial value of zero, and a final value of V. The velocity is then multiplied by dt through a gain block. Before we integrate, we must divide again by dt, for the same reasons as explained in the Calculating a

subsystem. Furthermore, we multiply by -1 because the velocity is shortening the muscles, i.e. making the displacements of the attached crossbridges smaller.

Calculating Force of Attached Crossbridges:

Now that we have the *x* vector, we can determine the force exerted by the individual crossbridges. Notice from Equation (2) *p(x)* is an increasing function, sketched in Figure 12. It has a relationship between displacement and force and exerts negative force (pushing back) when displacements are shorter than its equilibrium point at x=0.

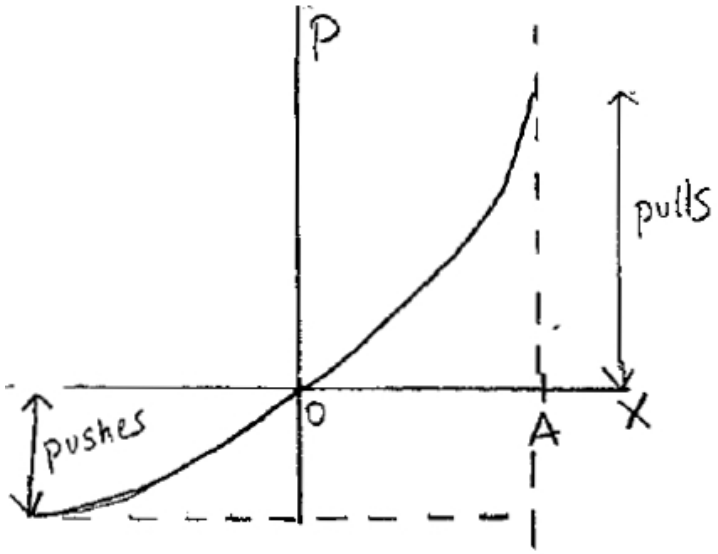

Figure 12: Force extension curve p(x) for an individual crossbridge. For positive values of x the crossbridge will be pulling, and for negative values of x the crossbridge will be pushing.

We model Equation (2) in Simulink in the *p(x)* subsystem using math operations blocks. The values for γ and p_1 are given in the parameters_crossbridge.m file.

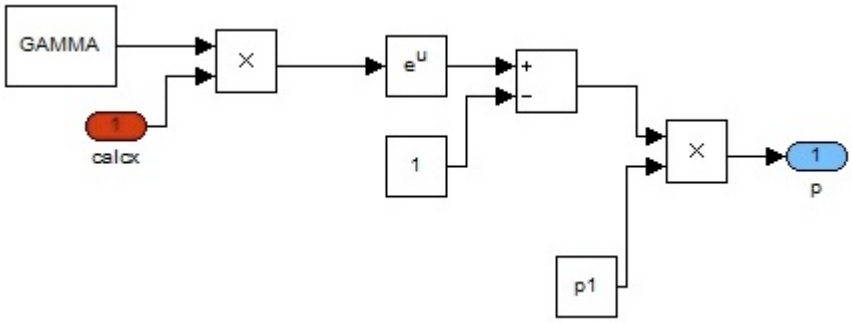

Figure 13: p(x) subsystem. Equation (2) was gathered from Hoppensteadt and Peskin (2002)

We sum the individual forces generated by each crossbridge to find the total force of the sarcomere using a Sum of Elements block in the main Simulink window of crossbridge.mdl.

Running the Simulation:

One final element we must discuss before we can run the simulation, are the configuration parameters. We set up our simulation to be fixed-step with ode1 (Euler) as our solver. We have already mentioned our step size of dt, but not our stop time. The stop time of our simulation is dt*klokmax. By multiplying klokmax (the number of time steps) by dt, we reduce the simulation to the amount of time we want it to run.

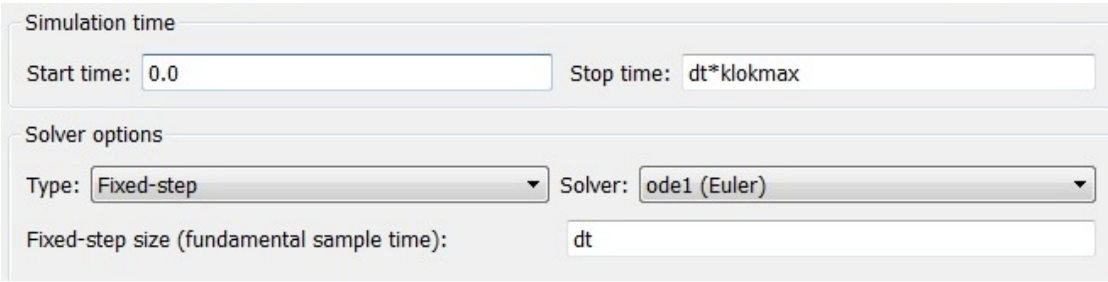

Figure 14: Configuration Parameters Window

Once run, the simulation will export the *a* and *x* vectors into the MATLAB workspace for plotting with the plotcrossbridge.m file. The top subplot of Figure 15 shows the total force generated by the sarcomere over time. At T_{start}, where we set the muscle to shorten, we see that as the muscle shortens the force the crossbridges can generate decreases.

Before T_{start}, the muscle was undergoing isometric contraction. Our model does a good job in imitating the macroscopic muscle mechanics using crossbridge dynamics. In the bottom plot shows the fraction of attached crossbridges calculated with Equation (1). Crossbridge dynamics are independent of muscle shortening (as they attach and detach at random.

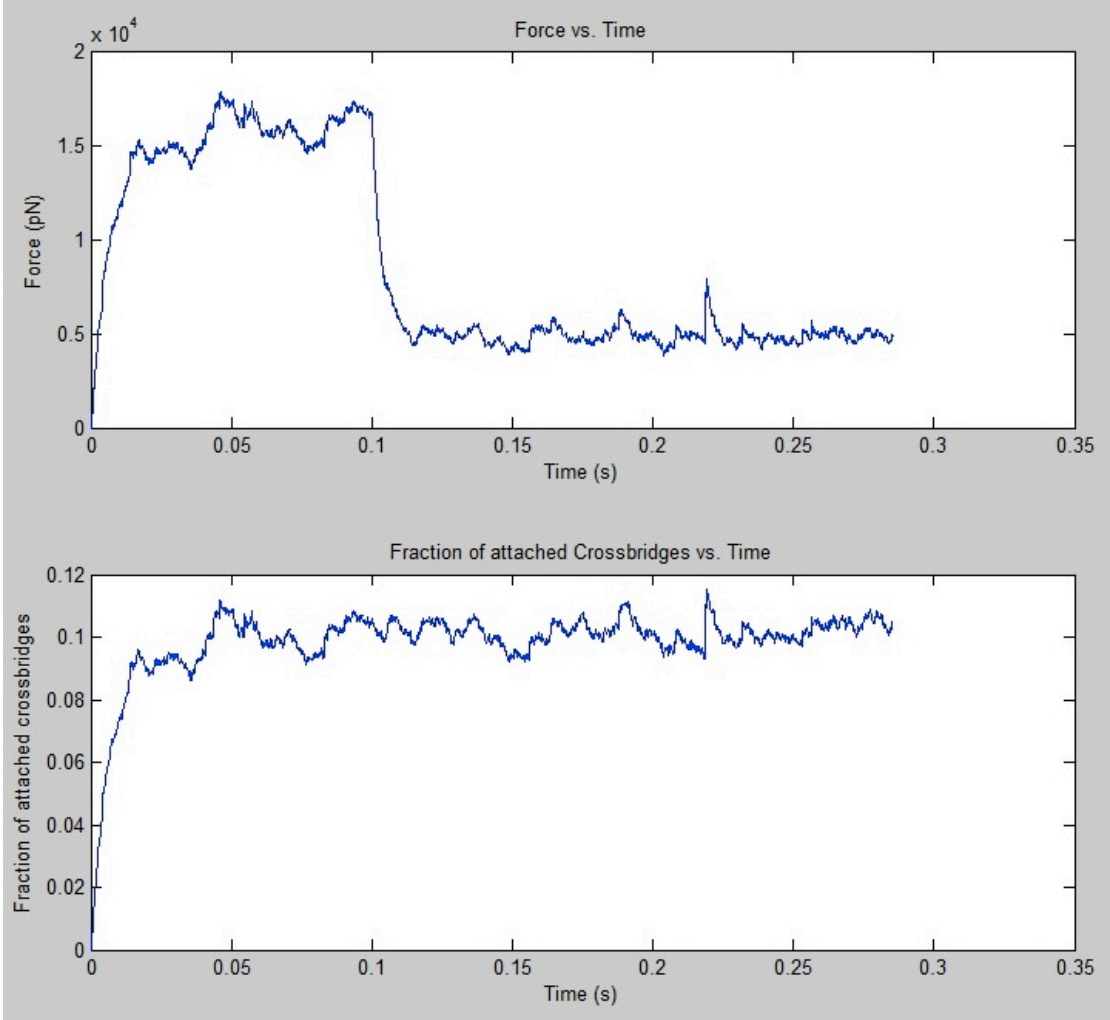

Figure 15: Main plots of Crossbridge.mdl. Top subplot demonstrates the total force generated by the sarcomere over time. The bottom subplot shows the fraction of attached crossbridges over time.

Crossbridge Population Density

The exported *x* and *a* vectors are used to calculate the crossbridge population density. We are interested in knowing the distribution of crossbridges throughout the different displacements *x* throughout the sarcomere. To accomplish this, we utilize the MATLAB function popdensity (Appendix 2). The function utilizes the *x* and *a* vectors, as well as a set number of bins pre-established in the parameters_crossbridge.m file.

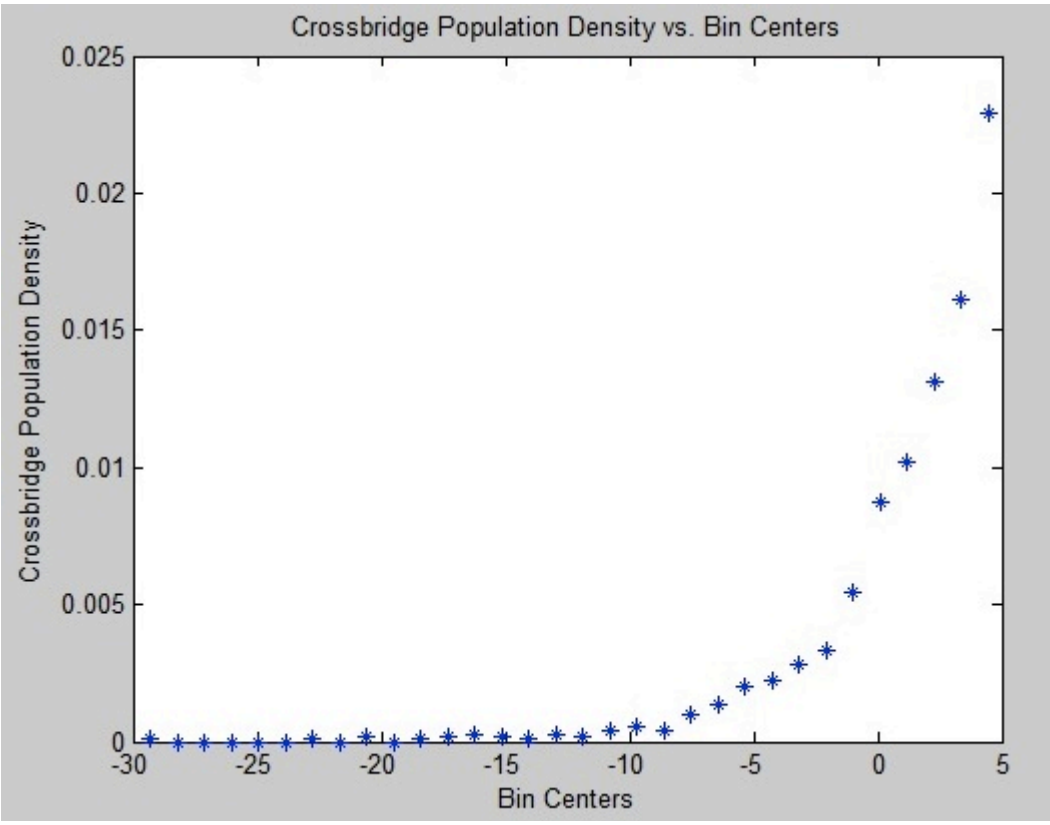

Figure 17: Crossbridge Population Density vs. Bin Center

Works Cited

[1] Hoppensteadt, Frank C., Peskin, Charles S. Modeling and Simulation in Medicine and the Life Sciences. 2nd Ed. New York: Springer Science+Business Media, Inc, 2002. Print

Appendix 1 – Parameters File

```
%This Model has been adapted from the MATLAB code presented in
Hoppensteadt
%and Peskin "Modling and Simulation in Medicine and the Life Sciences

%Constants for p(x)
p1=4;                       %units: pN
GAMMA=0.322;                %units: /nm

%Constants for u
alpha=14;             %probability per unit time for attachment: /s
beta=126;             %probability per unit time for detachment: /s
n0=10000;             %number of crossbridges
A=5;                  %displacement of newly attached crossbridge: nm
dt=0.01/(alpha+beta); %duration of time step: s
klokmax = 4000;

%number of bins for population density function:
nbins=round(sqrt(n0*alpha/(alpha+beta)));

%Model parameters
Tstart=14/(alpha+beta);    %time at which motion begins: s
V=500;                     %shortening velocity for t>Tstart: nm/s

%All crossbridges are initially detached.
a=zeros(1,n0);
x0=zeros(1,n0);
```

Appendix 2 – MATLAB Functions

popdensity.m

```
function popdensity(xout,aout,nbins)
%find population density of attached crossbridges

%input variables:
%  xout = array of crossbridge displacements
%  aout = array of crossbridge states (0=detached,1=attached)
%  nbins = number of bins

%Create x and a vectors from output arrays from Simulink
xout=xout(size(xout,1),:);
aout=aout(size(aout,1),:);

n0=size(xout,2); %total number of crossbridges
Nabridges=length(xout(find(aout)));  %number of attached crossbridges

%nabridges=number of attached crossbridges in each bin;
%xc=centers of the bins:
[nabridges,xc]=hist(xout(find(aout)),nbins);

%normalize nabridges to find crossbridge population density
xstep=xc(2)-xc(1);

u = (nabridges/n0)/xstep;  % Why is this? (Hint: look at units)
plot(xc,u,'*') %plot population density
```

plotcrossbridge.m

```
%This script will plot the total force generated by the crossbridges,
and
%the fraction of attached crossbridges.

%Store the arrays for future plotting:
time=P_U.time;
P=P_U.signals.values(:,1);
U=P_U.signals.values(:,2);

%Plot Results
figure(1)
subplot(2,1,1), plot(time,P);
title('Force vs. Time');
ylabel('Force (pN)');
xlabel('Time (s)');
subplot(2,1,2), plot(time,U);
title('Fraction of attached Crossbridges vs. Time');
ylabel('Fraction of attached crossbridges');
xlabel('Time (s)');

%Call the Popdensity Function
figure(2)
popdensity(xout,aout,nbins);
```

Copyright

Document

Software

Developed by: Manuel A. Ramirez Garcia, Bradley P. Sutton
Magnetic Resonance Functional Imaging Lab (MRFIL)
Department of Bioengineering
University of Illinois Urbana-Champaign
https://wiki.engr.illinois.edu/display/SimMeasPhysio/Home