Mitotic Spindle Simulation Manual

I

How to Download and Use the Program

This program was written in MATLAB and Simulink. In order to run the program, you must have MATLAB and Simulink R2010b or higher installed with a C compiler. The mathematical model can be downloaded from: http://cismm.cs.unc.edu/downloads. The findings from this model are published in *The Journal of Cell Biology*, “Pericentric Chromatin Loops Function as a Non-linear Spring in Mitotic Force Balance” Stephens et al., 2013. Once you have downloaded the program, please unzip it and leave all the files together in the extracted folder. The folder should contain the following files:

Coupled_Model.mdl

Population.m

Timelapse.m

doubleToSingle.m

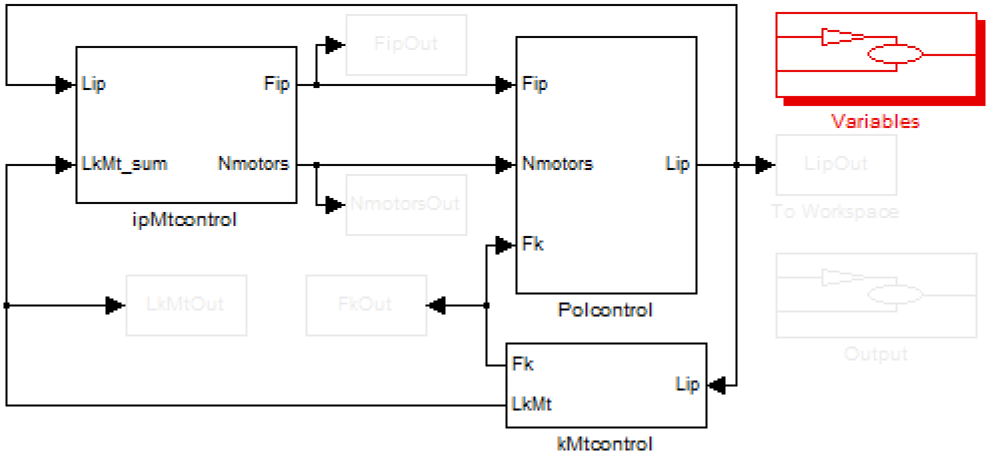
freeToSingleipMt.m

singleToDouble.m

singleToFree.m

xlswrite1.m

To make changes to the initial conditions of the model, open *Coupled_Model.mdl* and change the conditions in the variables control panel. For more information on what these parameters should be, please see the table II in the main text.



Map 1: Variables

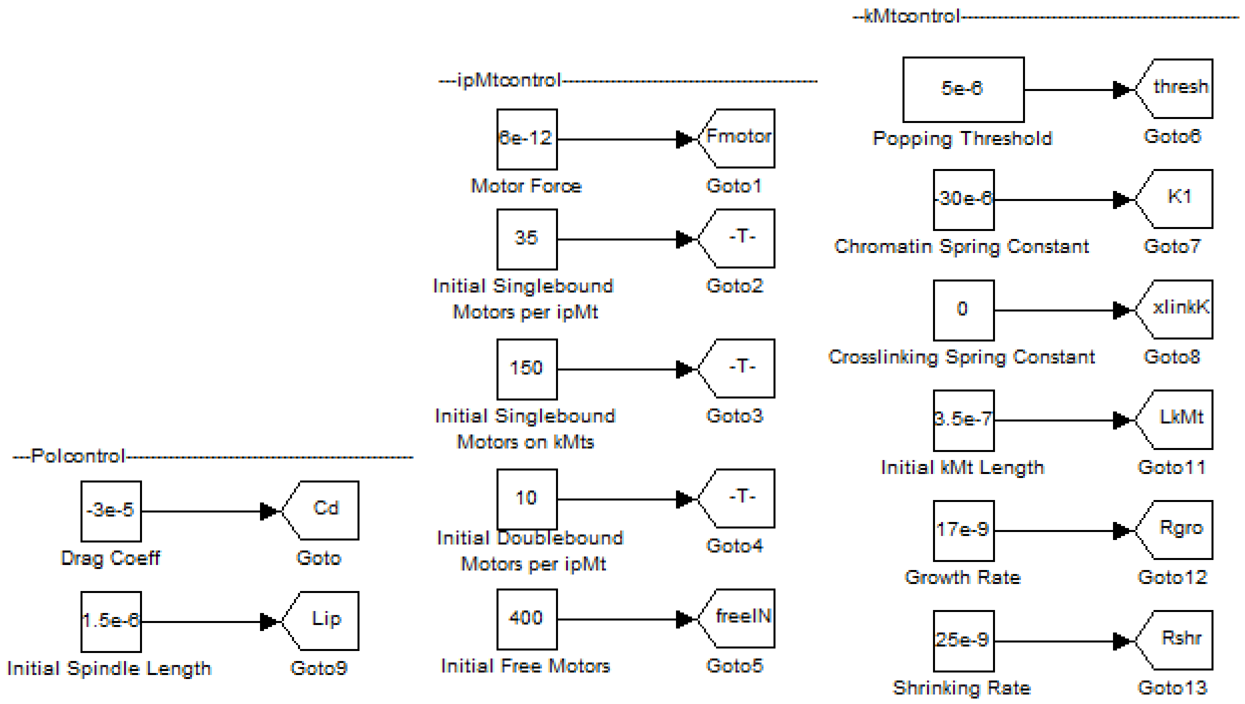


Figure 1: Variable List

To run the model like a time lapse, open *Timelapse.m*. Make sure that the workspace folder in MATLAB is the folder containing the extracted files. Click the run function button. A dialog box will appear to query how many times you want to run it. Then a standard save file window will appear. Please give it a filename without any extension and do not use the '.' character in the filename. A number indicating which run it is will be appended to the end of the file name. For example, if you run the file twice and give it the name "filename", you will then have two files, "filename1" and "filename2." During the function run, a pop up will ask you if you want to save the changes you have made to the Excel file. Once the run is over, please select 'Save Changes.' Timelapse output is used for determining the spindle variation. The first 50 steps are ignored as during this time the simulation is equilibrating.

To run the model for population measurements, open *Population.m*. Make sure that the workspace folder in MATLAB is the folder containing the extracted files. This function runs the same as *Timelapse.m*. Population output is used for determining the average lengths and percentage of stretching.

If you wish to run analysis in MATLAB with your own functions, or to look at scopes in the model, it may be run simply by opening it and pushing the run model button. Alternatively, the MATLAB built in function sim(*Coupled_Model.mdl*) may be used. A table of workspace outputs and their meanings may be found at the end of this document.

It is not necessary to interact with any of the other files provided, but they must remain in the same folder for the model to run properly.

II

Model Information and Subsystem Overviews

The system is made of three subsystems, kMtcontrol, ipMtcontrol, and Polcontrol.   The system is coupled: each subsystem produces outputs that would serve as inputs for another subsystem(s). We will introduce the basic functions of the model by going through each subsystem individually.

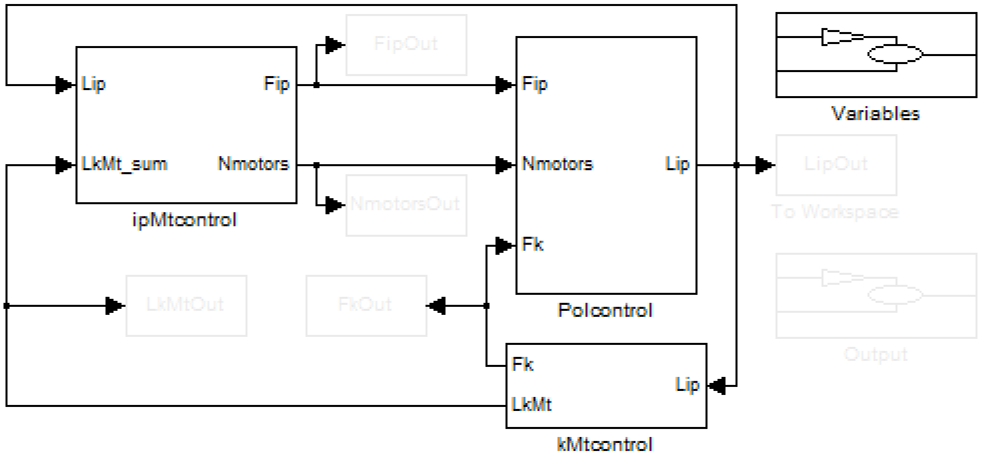
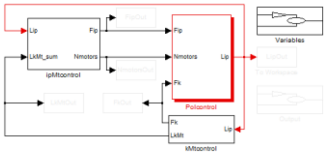


Figure 2: Subsystems

Polcontrol Subsystem

Inputs: Fip, Fk, Nmotors

Outputs: Vip, Lip



Map 2: Polcontrol Location

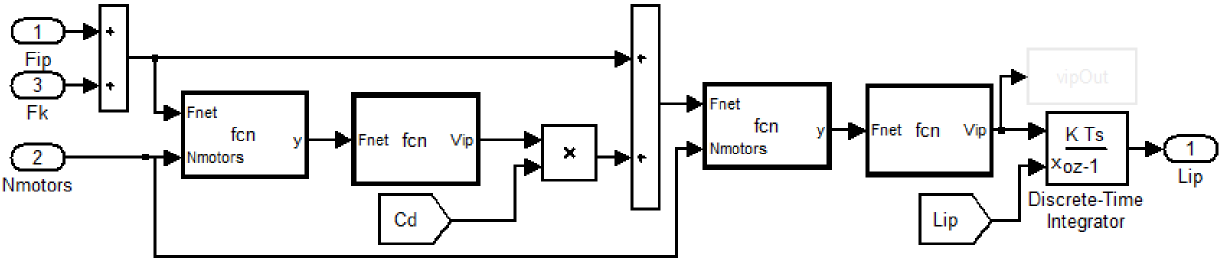


Figure 3: Polcontrol

The velocity is calculated by first figuring out the total force exerted on each motor, and then using a velocity vs. force curve to determine the velocity. Since the interpolar force per motor is always 6 pN and the spring force is always negative, when the net force is zero, the forces cancel each other out and the motor stalls. Likewise, at 6 pN, the spring force is zero, and the motor is able to run at its unloaded motor velocity of 50 nm/pN. The force per motor should never be greater than 6 pN. This original velocity is calculated without respect to the drag felt from the cytoplasm and serves as a predictor step.

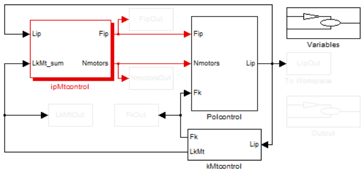
We then perform a corrector step to calculate this velocity with drag. The initial calculated velocity is multiplied by the drag coefficient to get the force of the drag on the spindle. We then sum this along with the interpolar and spring forces, and recalculate a velocity using the same velocity vs. force curve.

The length of the spindle is simply the integral of the velocity set with an initial condition (1.5 μm).

ipMtcontrol Subsystem

Inputs: Lip, LkMt_sum

Outputs: Nmotors, Fip



Map 3: ipMtcontrol Location

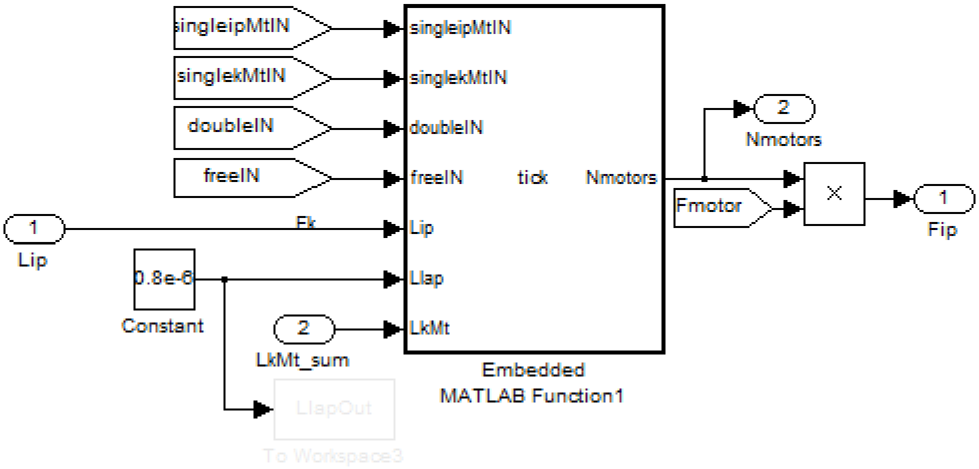


Figure 4: ipMtcontrol

The ipMtcontrol subsystems simulate the ipMt's and the associated motor behavior. We assume the overlap zone is of a constant length (0.8 μm).

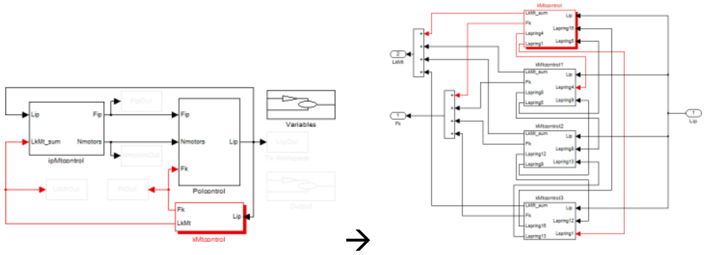
Fmotors, Llap, Fk, LkMt, and Lip, are put into the Embedded MATLAB Function, which determines the number of motors doubly bound over the overlap region. This function calls five functions included in the downloaded folder (*doubleToSingle.m, freeToSingleipMt.m, freeToSinglekMt.m, singleToDouble.m and singleToFree*.m). These functions distribute a population of motors based on on/off probabilities.

Fip, the aggregate force of all motors, is then determined by multiplying the number of motors with force of individual motor: $Fip = Nmotors \cdot Fmotor$.

kMtcontrol Subsystem

Inputs: Lip

Outputs: Fk, LkMt_sum



Map 4: kMtcontrol Location

The kMtcontrol subsystem has two components, kMt dynamics and chromatin spring behavior. Because these components interact closely with each other, it makes sense to put them into the same subsystem. The kMtcontrol subsystem appears to be broken into 4 separate subsystems, but this is out of coding convenience, and all 4 subsystems interact with each other, as if they were one system. The subsystem appears complex, but that is mainly due to the repetition of 16 springs and 32 kinetochore microtubules.

kMt Dynamics

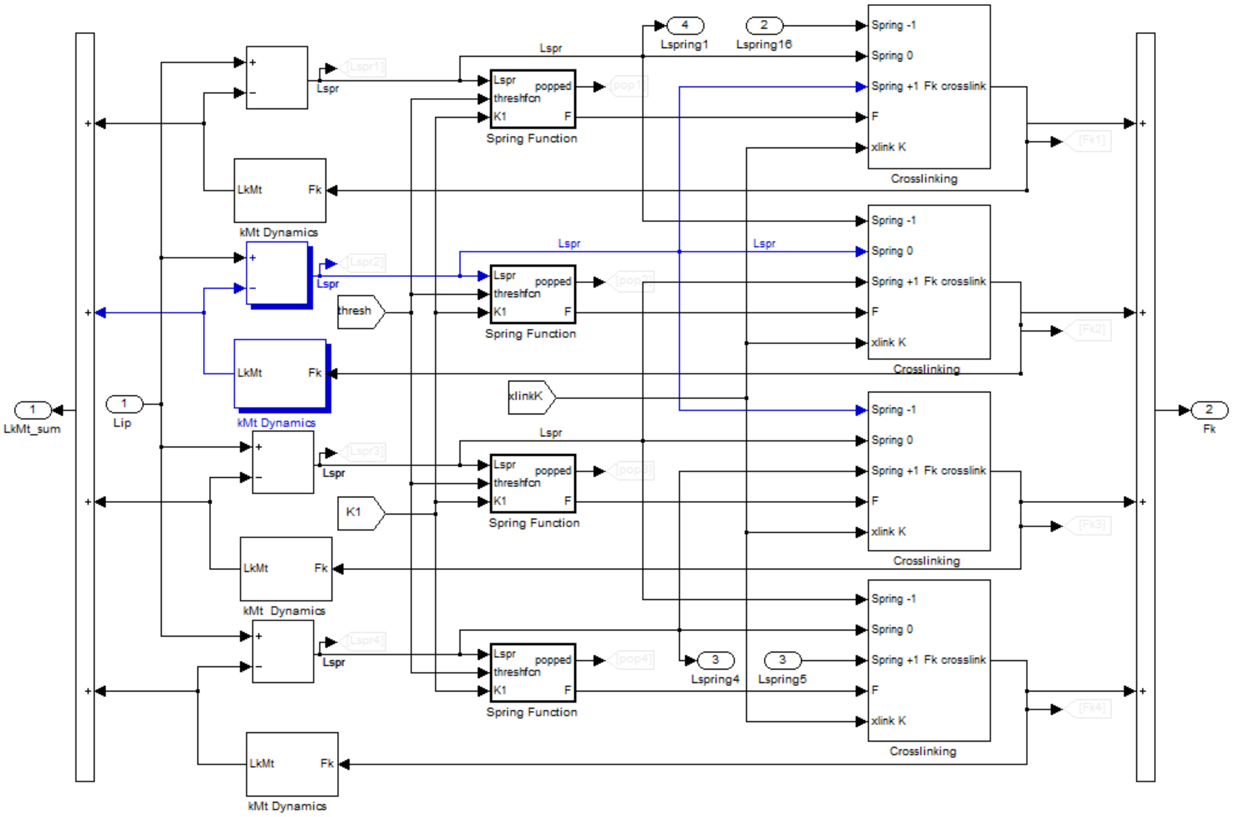
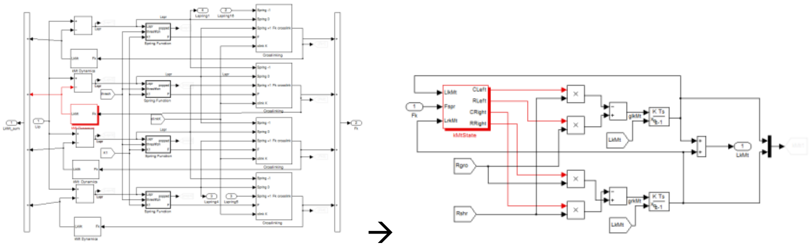


Figure 3: kMt Dynamics

kMt dynamics is a well-studied subject (see text). A microtubule undergoes two states—catastrophe (shrinkage) and rescue (growth). Gradients to switch into these states are determined by kMt length and individual spring force, respectively. The rescue rate of microtubule is about 17nm/s, and that of catastrophe is about 25 nm/s. There are 16 identical kMt Dynamics subsystems of the same set up, one per chromosome each with a left and right kMt.



Map 5: kMtstate Location

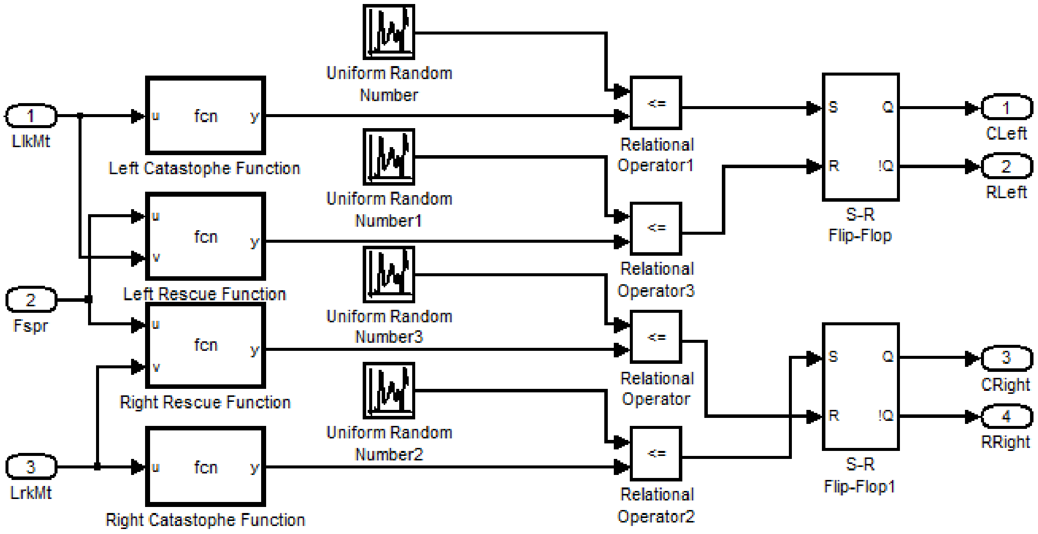


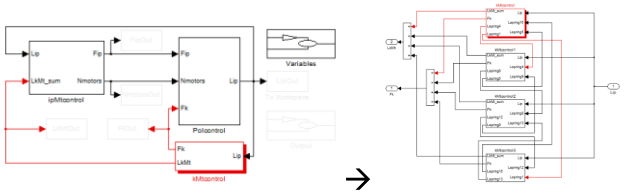
Figure 4: Rescue and Catastrophe

The kMtState subsystem controls microtubule dynamics internally. It uses length based catastrophe and tension based rescue formulas to determine the probabilities of entering rescue and catastrophe. A random number generator and truth table are used to determine which state the system goes into as follows:

Rescue	Catastrophe	State
TRUE	FALSE	Rescue
FALSE	TRUE	Catastrophe
TRUE	TRUE	Same as Previous Step
FALSE	FALSE	Neither

We use the growth and shrinking rates to determine how much the microtubules grow or shrink in that time step.

Chromatin Spring Behavior



Map 6: kMtcontrol Location

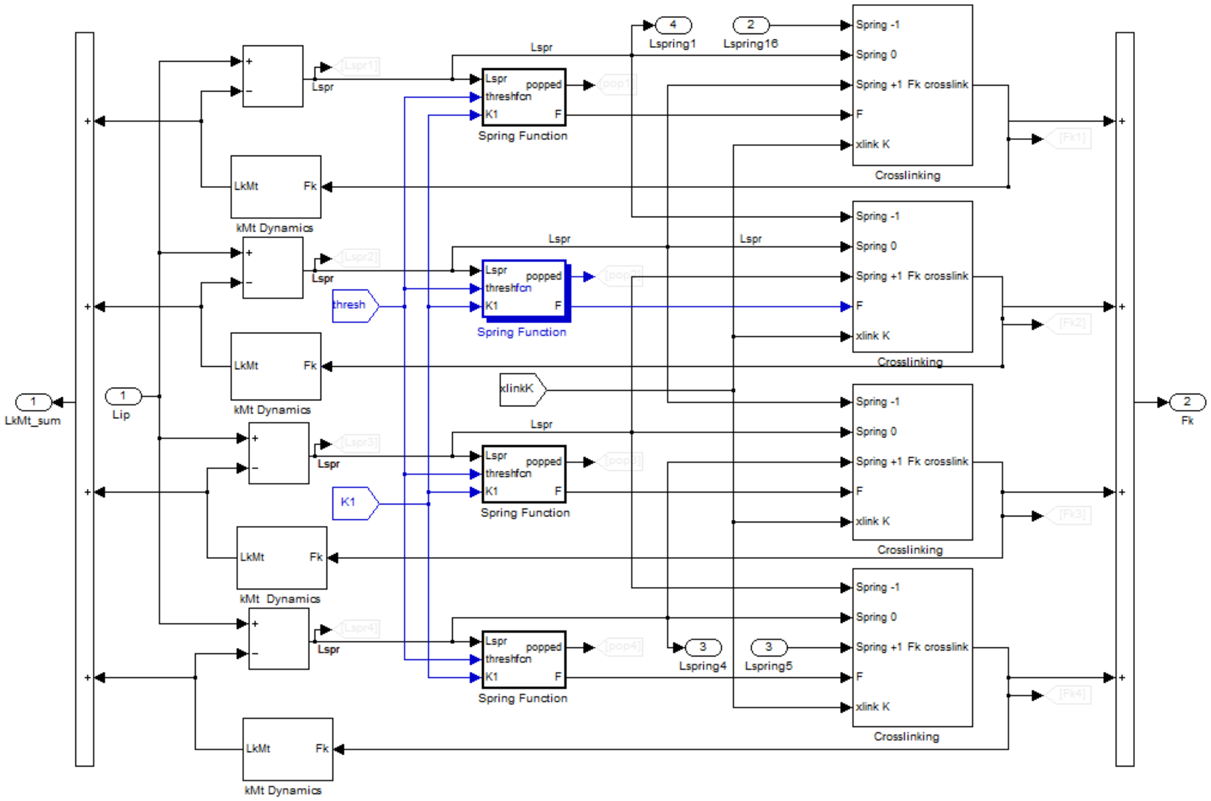


Figure 5: Chromatin Spring Behavior

The length of the spring is determined by subtracting LkMt from Lip. This is fed into our equation for the popping spring. This is a piecewise function of two lines that are discontinuous at the popping threshold length. The parameters for the function can be set in the control panel using the popping threshold and chromatin spring constant (thresh and K1 in Fig.5 above). In order to run the model like the Coupled Linear Stochastic, the threshold should be set to a spring length that will not be reached whereas the chromatin spring constant (k_1) is the variable parameter. The default value of 5 µm will not be reached by the lowest reported spring constant (5 pN/µm). To run the model as the Coupled Non-linear Stochastic the chromatin spring constant should be a set parameter (30 pN/µm default) whereas the threshold (thresh) is the variable parameter (effective range 800-1100 nm). The first line follows Hooke's Law $(F = k\Delta x)$

with a spring constant of k_1 and a rest length of 200nm. The second line follows Hooke's Law with a rest length of 650 nm and a weaker spring constant given by:

$$k_2 = k_1 \cdot \frac{Lspr}{Lspr+popLength}$$

Cross-linking

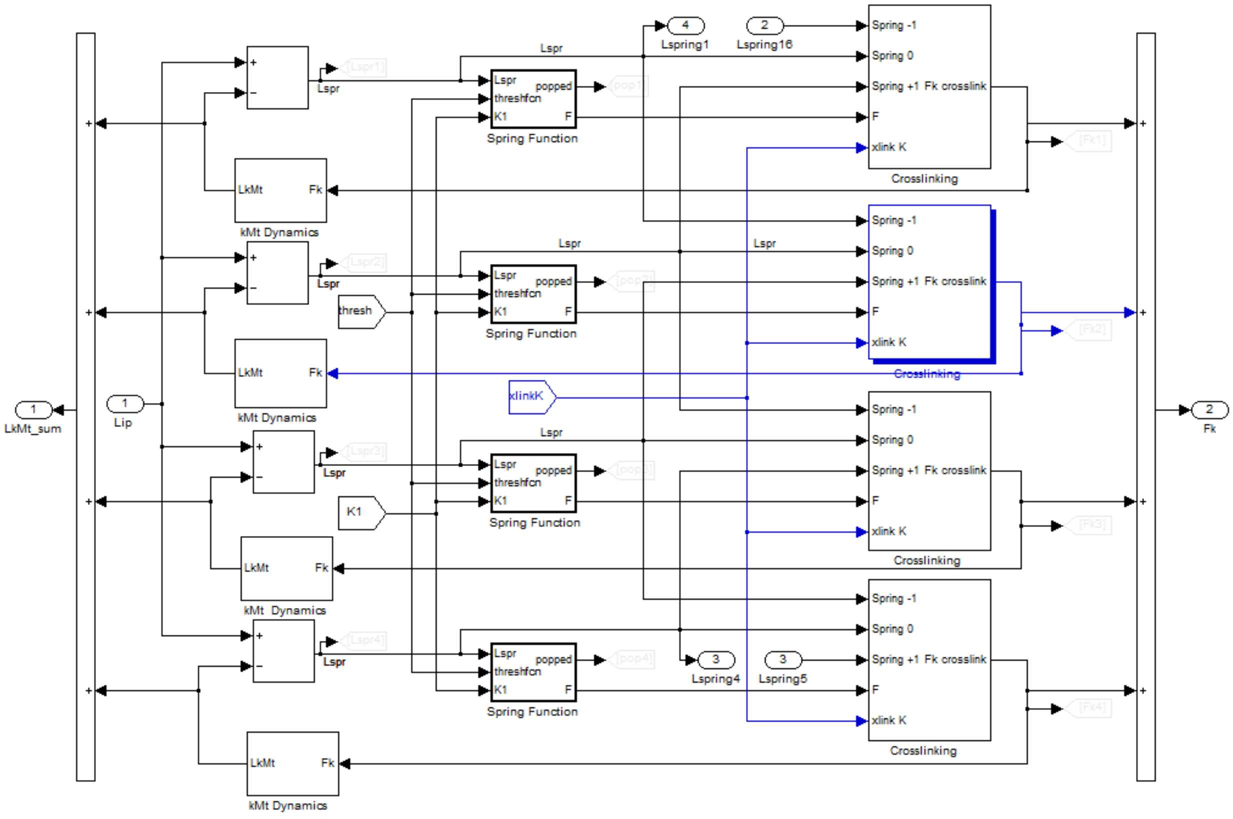


Figure 6: Cross-linking

The force found with the previous function is then fed into the cross-linking subsystem. This subsystem determines the force applied on the spring if it is connected to its two neighbors using Hookean Springs F= -(xlink K)(Δx), where xlink k is the cross-linking spring constant and Δx is the difference between the length of Spring 0 and each of its neighboring springs (Spring +1 and -1). It therefore takes as input the length of the springs adjacent to it. The amount of crosslinking can be changed in the control panel by changing the cross-linking spring constant (xlink K in fig.6 above). If no crosslinking is desired, the crosslinking should be set to 0 pN/µm. Ranges for cross-linking explored between 0- 15 pN/µm, which represented 0-0.5X a set chromatin spring constant (30 pN/µm default value). For simplicity we do this in one dimension because the angle between the two is small enough to use small angle approximation. These three forces are summed to determine the total force of the spring.

III

Workspace Outputs

Output	Name	Size
Interpolar Force	FipOut	1001x1
Cumulative Spring Force	FkOut	1001x1
Interpolar Length	LipOut	1001x1
Cumulative Kinetochore microtubule length	LkMtOut	1001x1
Overlap Length	LlapOut	1001x1
Number of Double bound motors	NmotorsOut	1001x1
Loop Popping	PoppedOut	1001x16
Individual Spring Force	allFksOut	1001x16
Individual Spring Lengths	allLsprOut	1001x16
Individual Microtubule Lengths	kMtsOut	1001x32
Time	tout	1001x1
Velocity	vipOut	1001x1