Updated Documentation for `C++` model

Erin Angelini

August 21, 2018

1 Updates to `main.cpp`:

The main file now includes code that writes the end time and angular orientation of the spindle when it has completed rotation. The first part of this code is the part that determines what "complete" rotation means. Experimental results from the Dawes lab indicate that nuclear envelope breakdown occurs within 90% of the full rotation from $\pi/2$ to the horizontal. Therefore, under the `runModel` function, we place a check on the spindle's angle to see if it is within $\pi/2 \pm 0.9(\pi/2)$. If it has reached this threshold, we consider to be done rotating, so we write the final time and position and end the run.

The new function that writes the final time for us is called `writeFinalTime`; its writes the final spindle position, angle, and time to rotation. As of right now, this feature(i.e. recording time to rotate) is turned on and off manually by respectively uncommenting and commenting out these two blocks of code. These blocks of code can be found by searching the document for the comment string `//EA:`.

1.1 Length dependence

There is also a version of `main.cpp` that includes length dependent pulling and pushing forces. This change is implemented under the functions `mtForceCalcM` and `mtForceCalcD`, which are the functions that calculate the total force generated by the mother (M) and daughter (D) asters, respectively. As with the current implementation of the mean-field model, each MT force scales with its length `L` by a factor of $\beta = 1$. That is, force(MT_i) = L_i for each MT. Like the other updates, this code is marked by the comment `//EA:` and must be turned on and off manually.

2 Updates to `parameters.cpp`:

For the off-center spindle position option (i.e. `startPsi` > 0), the off-center coordinate is no longer fixed at 10.0 but instead scales with `R1_max`. More specifically it is set to be $0.4\times$`R1_max`, which is the 70:30 egg-length point (consistent with the results form Coffman *et al.*).

The `parameters.cpp` file also now includes code for implementing a lower anterior MT-dynein binding probability ($P_a = 0.65$, coded into `regionForceMultipliers`) *without* push bands present. In this implementation, the anterior and posterior regions are defined by the parameter `myPos`, which is the Cartesian angle that marks the 60:40 egg-length point (0.2×`R1_max`). This parameter is used to divide the cortex into 3 regions: (upper) posterior, anterior, and (lower) posterior. Like the new code in `main.cpp`, turning these features 'on' or 'off' must be done manually by uncommenting/commenting out the appropriate code.

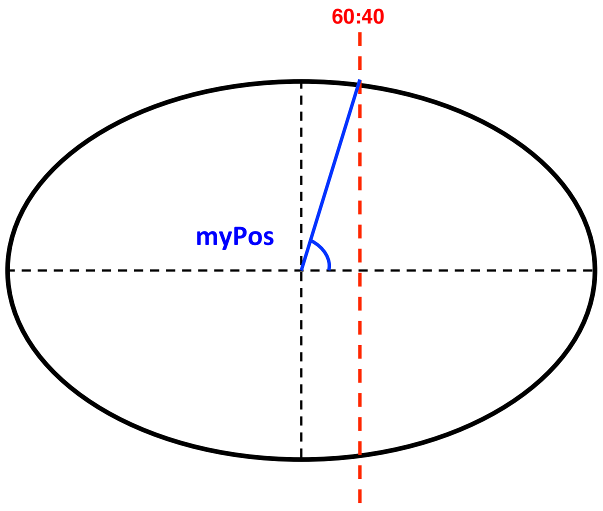


Figure 1: Demarcation of the 60:40 anterior-to-posterior split by Cartesian angle in the computational model.

2.1 Changing parameters with aspect ratio

Certain parameters need to be updated when you change the value of either `R1_max` or `R2_max` (i.e. the length of either axis of the ellipse). The instructions for doing so are as follows. So far all changes must be done manually (i.e. copy and paste for each new aspect ratio)

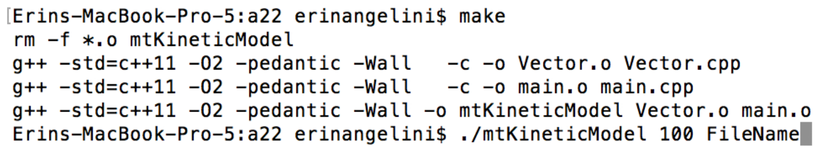
`myPos`: The position of the anterior-to-posterior break. If implementing the model with an anterior-to-posterior difference in binding probabilities without a push band, then this parameter must be updated with the aspect ratio. The pre-calculated values are located under the 'band and window code' directory in the file `APpos.cpp`.

`start` and `end`: These are the Cartesian angles the demarcate the start and the end of the push band regions, respectively. If implementing the model with the push bands, these parameters also change with aspect ratio. The pre-calculated values are located under the ‘band and window code’ directory in the file `pushBands.cpp`.

`numberContactWindows` and `contactWindowAngles` These inputs describe the number of cortical dyneins and their positions (in *polar* angles, NOT Cartesian), respectively. These must also be updated with aspect ratio. The pre-calculated values are located under the ‘band and window code’ directory in the file `aspectContactWindows.cpp`.

3 General use

This code requires Xcode to run on Mac computers. To run the model, first open terminal and cd into the folder containing `main.cpp` and the other model files (`parameters.cpp`, `mtKineticModel.exe`, etc.). On the command line, enter `‘make’` –this command compiles the model. Then enter `‘./mtKineticModel n fileName’`, where `n` is the desired number of runs and `fileName` is the name of the .csv file to which the final data will be written (see screenshot below). To read more about the options for writing final data, see the ‘README_2016’ file (original documentation written by Matt McDermott). If no options are passed into the command line about what data is written into the .csv file, then the program automatically writes the final position of the pronucleus (x and y coordinates) and the final angular orientation of the spindle (radians). If the `writeFinalTime` function is in place, then the program also writes the time at which the spindle completed rotation (in minutes). If the spindle does not complete rotation, no time is written and the final time is the total length of the simulation (as given in the `parameters.cpp` file). The .csv file and question is written to the folder titled ‘data,’ which is located one directory up from the model code itself.



```
[Erins-MacBook-Pro-5:a22 erinangelini$ make
rm -f *.o mtKineticModel
g++ -std=c++11 -O2 -pedantic -Wall   -c -o Vector.o Vector.cpp
g++ -std=c++11 -O2 -pedantic -Wall   -c -o main.o main.cpp
g++ -std=c++11 -O2 -pedantic -Wall -o mtKineticModel Vector.o main.o
Erins-MacBook-Pro-5:a22 erinangelini$ ./mtKineticModel 100 FileName
```

Figure 2: Sample command for running C++ model in the terminal.