Documentation for C++ model

Matt McDermott

February 22, 2014

1 Parameter Values:

`Tau`= τ: τ measures the timestep for the model and is currently $\frac{1}{4000}$ minutes.

`Contact Length`: This measures the length of time that an MT attached to the Cortex remains attached. It is currently `400*Tau`= $\frac{1}{10}$ minutes.

`Vg`= V_g: This measures the growth velocity of an MT and is currently $40\mu m$/min.

`Vs`= V_s: This measures the shrinking velocity of an MT and is currently $120\mu m$/min.

`Vs_c`= V_{sc}: This measures the shrinking velocity of an MT post contact and is currently $120\mu m$/min.

`kc`= k_c: This measures the catastrophe frequency of an MT and is currently $.1 \cdot 601$/min.

`kr`= k_r: This measures the catastrophe frequency of an MT and is currently $.4 \cdot 601$/min.

`R1_max`= R_1: This measures the maximum length of the ellipsoidal cell body along the x-axis and is currently $50\mu m$.

`R2_max`= R_2: This measures the maximum length of the ellipsoidal cell body along the y-axis and is currently $30\mu m$.

`Prad`= $P_{\mathbf{rad}}$: This measures the radius of the pronucleus and is currently $5\mu m$.

`Eta`= η: This measures the translational drag coefficient of the pronucleus and is currently $1\frac{\text{pN}}{\mu m \text{min}}$.

`Mu`= μ: This measures the translational drag coefficient of the pronucleus and is currently $5\eta = 5\frac{\text{pN}}{\mu m \text{min}}$.

2 Usage Instructions:

The three basic steps to to use this code are:
To Compile: Run `make` on the command line.
To Run: After compilation, run

```
./mtKineticModel <Number of Runs> <Output File> [temp/all]
```

on the command line. Here, `<Number of Runs>` denotes the number of runs of the given system the model should produce and `<Output File>` specifies the file to which the results should be written. Note that this file will be placed in the directory `nuclearKineticsModelling/data`. The final paramterer, `[temp/all]`, is an optional parameter. If omitted, the system will only store the final configuration of the system in the result file. If given as `temp`, it will store 30 second time slices in the result system for each model run, in sequence (i.e. it will list various parameters for 30 second slices for the first run, then those for the second, and so on and so forth). If given as `all`, it will write data for every timestep in the result file, in sequence, for each model run.
To View: There are a variety of matlab scripts in the `plottingCode` folder to visualize the results. Of particular note are the histogram scripts, `positionRotationHist.m` and `positionRotationHistCompare.m`, the single-run visualization tool, `readData.m`, and the movie-frame making scripts `movieMaker*.m`. All of these scripts are run with the following basic premise: Open `MATLAB`, set values `dataDir` and `dataFile` *(and/or `dataFile2`)* in the `MATLAB` console, then run the script. `dataDir` stands for the directory (as a sub-directory of `'nuclearKineticsModelling/data/`) and `dataFile` stands for the dataFile in question.

2.1 Basic Usage:

For basic usage, the only additional step beyond simple compilation and running is parameter modification. Here is a short description of some of the parameters we change most frequently *(All found in the `parameters.cpp` file).*

`springsOn` This boolean value controls whether the centrosomes are connected to the pronucleus rigidly (and thus not allowed to translate relative to the pronucleus) or via springs (of spring constant given later in the file).

`translation` This boolean value controls whether the pronucleus is allowed to translate. If it is `true`, the system proceeds in full generality. If it is `false`, the pronucleus center is fixed in the cortex (though it is allowed to rotate). If the springs are on, the centrosomes are also allowed to translate (and thus provide force via the springs to the pronucleus).

`kM/kD` These parameters control the relative spring constants for the springs connecting the mother (`M`) centrosome and daughter (`D`) centrosome to the pronucleus, respectively. If springs are off, then these parameters do nothing. We do not yet have good values for these constants.

`startX` This parameter controls the starting x coordinate of the pronucleus, with 0 being the center of the cell. We typically test starting at 0 and starting at 7.5μm.

`startPsi` This parameter controls the starting angular position of the pronucleus, in radians, with 0 being *horizontal*. For on-angle runs, we use $\frac{\pi}{2}$. To test the effects of angular variations, we typically use $\frac{\pi}{2} + k\frac{\pi}{8}$, with $1 \leq k \leq 4$, though it might be sensible to test larger k and finer gradations.

`numRegions/regionAngles/regionProbabilities` These are the band parameters, and control the number of uniform probability regions, the angular endpoints of those regions, and the associated probabilities with those regions.

2.2 Advanced Usage:

There are a variety of other parameters one may wish to modify, all of which found in the `parameters.cpp` file. For these modifications, the same procedure is used: modify the parameter, re-compile, and run.

3 Current Algorithm

3.1 High Level Description

The current algorithm is very straightforward. In particular, for each step in time, stepping by a set parameter `tau` through `Duration` minutes, the algorithm computes the net force and torque on the pronucleus and then uses an eulerian approximation of the solution to the DE inspired by friction dominated domain conditions to update the position and angle of rotation of the body. The algorithm also maintains a list of endpoints of a set number of MTs, updating these through growth, shortening, and respawning when necessary. It also uses the same risk formulas for catastrophe and rescue as did the previous matlab model. The only real differences are that several bugs are corrected, force and torque are calculated upon every iteration directly, and the implementation outputs final data to a file for another program to read in.

3.2 Implementation Details

- It is important to the current implementation that `mt_numb` is a constant value, as otherwise fixed-size arrays could not be used to store all the codes data.
- The `parameters.cpp` file defines not only fixed constants for the program, but also initializes global variables and sets up two very import `typedef`s. The first sets a new `float_T` type, which is used so that precision of the

model can be altered painlessly, and the second creates a `vec_T` type, which is used so that arrays of `float_T`s are easy to create.

- The two `operator` functions at the top of `main.cpp` are merely to aid in file output.
- The net force calc function is simple; It merely sums up the all the force directions for the MTs, then multiplies that sum by the force for any given MT. This will give the net force on that centrosome.
- The net force function is just a wrapper around net force calc.
- The `updatePNPos()` function is an extraction of the code to update the position of the pronucleus from the main loop. The actual mechanics of the function are very straightforward currently, and simply translate the components of the force into torque and motion inspiring forces, respectively.
- `advanceMT` is simply an encapsulation of some messy code from the loop. It grows or shrinks MTs.
- `respawnMT` creates a new MT.
- The main loop appropriately orders these functions, updates contact statistics, growth statistics, and growth velocities. It also checks for rescue and catastrophe and writes the data to the file.
- Note here that a major implementation detail between this algorithm and the matlab one is that as this algorithm computes the force on every iteration, not just on a new contact, the `MT_touch` array must be active every second, which in particular means that it cannot be used to distinguish between shortening velocities as it is in the matlab model. So, instead, an additional array is maintained, `MT_GrowthVel_*`, where * is either M or D. This stores the current growth velocity of the given MT.

4 Future Work

1. Test Stability

2. Write up results.