Documentation for `C++` model

Matt McDermott

February 3, 2014

1 Parameter Values:

`Tau`$= \tau$**:** 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 $40\mu m/\text{min}$.

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

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

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

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

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

`Mu`$= \mu$**:** 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 clean && make` on the command line.
To Run: After compilation, run `./main` on the command line.
To View: After running the code, run the matlab script `readData.m` within matlab to produce a graphic representation of the evolution of the system.

2.1 Basic Usage:

For basic usage, the only additional step beyond simple compilation and running is parameter modification. At this lever, there are two relevant parameters to change, both found in the `parameters.cpp` file.

1. Most commonly, the `MT_numb` parameter. This parameter indicates how many microtubules stem from a single centrosome. To modify it, simply alter its value in the `parameters.cpp` file, re-compile, and run again.

2. Secondly, the `translation` boolean value controls whether the pronucleus is allowed to translate or not. If `false`, no translation is permitted, only rotation. If `true`, both rotation and translation are permitted. Again, merely modify the parameter, re-compile, and run again.

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. However, there are several other changes that are conceivable that require more comprehensive changes. The first relates to data export. Currently, though this may change soon, there is a boolean value `ONLY_COMMA` defined in the `main.cpp` file itself. This controls the format of the output: `true` indicates that no seperates beyond comma are used and requires a thorough understanding of the output order for the data to be useful, while `false` allows the data to be parsed into component parts easily (though understanding what these component parts are still requires a complete understanding of the output). By default it is `true`, as it is required to be true for the `matlab` code to work.

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.

The matlab files in this folder, particularly `readData.m` can be used to read in the data from the `../data` directory and plot a "movie" of the growth, as well as the torque, force magnitude, and force angle.

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. This makes the main loop easier to read and also allows for easier generalization. Specifically, I am beginning to think it will, in fact, be necessary to use a DE solver, and this compartmentalization will be very useful then. 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

- I think we will need to use a DE solver, as often the forces are large enough that the pronucleus rotates well past the ideal.
- I think that we need to rethink the risks of catastrophe and rescue. They were implemented incorrectly in the matlab model and, when corrected, do not behave in a way that we like.
- The parameters files should be merged: the following resources look promising.
  `https://code.google.com/p/yamlmatlab/`
  `https://code.google.com/p/yaml-cpp/`
  `YAML` is a great format for this as it is language independent and thus extensible. If later, we want to do some statstics with `R`, or visualization with `python`, this would be just as portable there. There are other solutions to read `C++` files into matlab, or `matlab` files into `C++`, but the latter is just as complicated as using `YAML` and the former requires using `MEX`, which isn't compatible with the compiler version necessary to compile the `C++` code here as it uses some `C++11` stuff.