Computing restitution properties and plotting curves

Jonas van den Brink

j.v.brink@fys.uio.no

July 6, 2015

1. Action potential duration (APD_{90})
2. Effective refractatory period (ERP)
3. Conduction Velocity (C_V)
4. Wavelength (WL)

Of these four, we can compute the two first in a 0D cell model, while we need a 1D tissue strand to find nr 3 and 4.

Action potential duration—APD$_{90}$

Effective refractatory period—ERP

The Conduction Velocity—C_V

The conduction velocity is the speed a signal propagtes through the tissue. Recall that the signal propagation through the tissue is goverened by the monodomain equations, which is an example of a *reaction-diffusion* equation. The system is a set of grid-points, or nodes. Each node describe a collection of cells. As a cell fires an action potential, the rising membrane potential can diffuse to nearby cells, causing them to fire, and this propagates the potential.

The conduction velocity results from a complex set of factors, and is strongly dependant on both the cell model, as well as the tissue properties, such as the diffusion coefficient. There is therefore no simple way of calculating the conduction velocity, and we should simply simulate a propagating wave and measure it from the simulation.

To do this, we denote some sort of threshold for activation, when the potential in a grid point rises above the treshold, we say the cell activates. We then monitor two grid pints, x_0 and x_1 and measure the time they activate. The conduction velocity is then estimated from

$$C_V \simeq \frac{x_1 - x_0}{t_1 - t_0}.$$

Code specifics

As mentioned, the conduction velocity depends on a lot of different parameters. The cell model itself affects things, as does the tissue diffusion coefficient. The diffusion coefficient, D, again depends on numerical properties, such as the time constant and spacing h. To set D, we simply use the fact that the conduction velocity at a BCL of 1 second should be roughly 750 mm/s. I have already done this and set the diffusion coefficient for both the Koivumäki and FK models.

To simulate the tissue strand, we use 40 grid points, letting each be 0.5 mm apart, this means we simulate a piece of tissue that is 20 mm long. For every time step, we must solve the ODEs for the cell model for every single grid point, so this computation will be a lot more costly than the 0D measurements.

We start of by loading the steady cycle from the 0D cell model into the tissue strand. But we still should send at least a few pulses through the strand, so that the system approaches a steady cycle for the entire strand. In the program, we do 5 pulses at every given BCL, and print the conduction velocities for each pulse. The last result should be used when plotting the results. As the simulations are so costly in comparsion to the 0D measurements, we do them for fewer BCLs, maybe every 50 or so.

I have also implemented so you can plot in real-time, or not. Plotting is a great way to get a feel for what is going on and how things are working, but they slow down everything considerably. I therefore think you should run the scripts a few time in plot-mode to see how it works, and then without plotting to get a speed-up.

I have also implemented a changing time step, it is important to have a small time step when a cell is firing an action potential, as there are very rapidly changing states which we can only catch properily with a small Δt, but when the pulse has passed through the grid, and all cells are slowly moving back to resting, we can increase the Δt. You will see this very clearly if you run the program in plot-mode.

Running to code

Open the file /1D/measure_cv1D.py. You should only need to change the lines at the bottom to run the program for different cell models and BCL's. These lines looks something like this

```
solver = TissueStrand('hAM_KSMT_nSR', 0.31, ...)
#solver = TissueStrand('hAM_KSMT_cAF', 0.31, ...)
#solver = TissueStrand('FK_nSR', 0.077, ...)
#solver = TissueStrand('FK_cAF', 0.077, ...)

solver.pulse(1000, num_of_pulses=5, liveplot=True)
```

Here, the first line selects what cell model we are using, the rest of the parameters are just so the rest of the script works with the given cell model. You should not mess with these parameters, just choose the different cell models by using one of the first 4 lines (comment in and out using the number symbol #).

The sixth line here, is where the script is actually promted to produce results. The first number is the BCL in ms, here it is 1000, that means, 1 second, the `num_of_pulses` are the number of pulses to use and `liveplot` denotes whether or not to plot in real time while simulating. In this line, you can change all of these paramters.

The BCL should of course be changed, so that we can get a restitution curve. Live plotting should only be used to study the system, setting it to `False` yields a big speed-up, so that should be used when simulating for many different BCLs. The number of pulses can be reduced to get a faster running time, but you should first study how much the CV changes from pulse to pulse.

To run for many BCLs in a row, you can use the range function and leave the script running for a long while, for example:

```
solver = TissueStrand('hAM_KSMT_nSR', 0.31, ...)

for BCL in range(1000, 300, -50):
  solver.pulse(BCL, num_of_pulses=5, liveplot=False)
```

If you want to also set it up so that the script does this for all the ODEs, you can do it like this

```
solver_list = [TissueStrand('hAM_KSMT_nSR', 0.31, ...),
               TissueStrand('hAM_KSMT_cAF', 0.31, ...),
               TissueStrand('FK_nSR', 0.077, ...),
               TissueStrand('FK_cAF', 0.077, ...)]

for solver in solver_list:
    for BCL in range(1000, 300, -50):
        solver.pulse(BCL, num_of_pulses=5, liveplot=False)
```

Wavelength