Computational Methods in Biology (Spring 2015)

Assignment 3 (due in class on March 16)

This assignment focuses on stochastic models and simulation methods, using ion channel gating as the model system. At the end of this sheet there is a brief description of how random (actually pseudo-random) numbers are generated on a computer.

1. [20 points] Consider the gating of a single two-state ion channel, with $C \rightarrow O$ transition probability k^+ and $O \rightarrow C$ probability k^- (units of both are ms^{-1}). Write a program for a Monte Carlo simulation of the gating of this channel for a duration of 500 msec, and time step $\Delta t = 0.5$ msec. Let $S = 0$ when the channel is in a closed state, and $S = 1$ when it is in an open state. Plot S vs. time. Also, compute the running sample mean of S and plot it along with S. [Sample mean $= \frac{\sum_i^n S(i)}{n}$, where n is the current iteration number, $t = n\Delta t$, and $S(i)$ is S at iteration i.] Do this for three combinations of the transition probabilities (k^+, k^-): (0.05,0.05), (0.3,0.3), (0.05,0.3). Comment on how the channel dynamics and sample means differ for these different combinations. Use your simulation to calculate the sample mean open and closed dwell times. Plot these vs. time. Compare the values of the sample mean dwell times at time $t = 500$ ms to the expected equilibrium mean dwell times from probability theory.

2. [20 points] Next consider an ensemble of 5 identical and independent ion channels. This could represent, for example, a patch of neural membrane containing 5 channels whose activity is monitored by a patch clamp electrode.

(a) Simulate the stochastic gating of the ensemble using five Markov variables, one for the state of each channel. Suppose that the transition probabilities for each channel are $(k^+, k^-) = (0.05, 0.05)$. Turn in a plot with three panels. The first two should show the states of two of the five channels versus time. The third panel should have two superimposed curves. One curve should show the number of open channels versus time. The other curve should show the sample mean of the number of open channels. According to probability theory, what is the expected mean number of open channels? Is this consistent with your sample mean from the Monte Carlo simulation?

(b) Simulate the stochastic gating of the ensemble using a single Markov variable for the number of open channels. What is the kinetic scheme for this simulation? Turn in a plot with three panels, corresponding to transition probabilities $(k^+, k^-) = (0.05, 0.05)$, $(0.3, 0.3)$, and $(0.05, 0.3)$. In each, plot the number of open channels versus time and the running sample mean of the number of open channels.

Does the sample mean match the expected mean from probabiility theory?

3. [20 points] Gillespie's method provides an alternative to Monte Carlo that is accurate for large ensembles and much faster than Monte Carlo. It is somewhat analogous to variable-stepsize numerical ODE solvers.

(a) Use this method to simulate the ensemble of five ion channels. Do this for the transition probabilities $(k^+, k^-) = (0.05, 0.05)$, $(0.3, 0.3)$, and $(0.05, 0.3)$. Turn in a 3-panel figure showing number of open channels and the sample mean versus time. Does the running sample mean match your expectations? For the case $(k^+, k^-) = (0.2, 0.2)$ also turn in a figure showing the transition or dwell times τ versus time. Also, include a curve showing the running sample means of the transition times. Do this mean match your expectations? Explain.

(b) Gillespie's method is particularly useful when performing simulations with large ensembles. With transition probabilities $(k^+, k^-) = (0.2, 0.2)$, do simulations for ensembles of 5, 20, and 100 elements. Turn in a plot with three panels, one for each ensemble. In each panel, plot the number of open channels and the temporal mean of the number of open channels versus time. Also turn in a plot of the temporal mean of the dwell times. This plot should have three curves in one panel: a curve for N=5, one for N=20, and one for N=100 channels. Comment on the relative sizes of the mean dwell times. Are the differences expected from the probability theory? Explain. Finally, turn in a plot of the running coefficients of variation versus time for each of your ensembles (again, one panel with three curves). Comment on the relative sizes of the CVs for the different ensembles. Are the differences expected from the probability theory? Explain.

4. [20 points] The aim of this problem is to add stochastic gating of a population of voltage-dependent ion channels to the **Morris-Lecar** excitable membrane model. The voltage equation is

$$\frac{dV}{dt} = -[I_K + I_{Ca} + I_l - I_{ap}]/C$$

where I_K, I_{Ca}, and I_l are potassium, calcium, and leak currents, respectively (details given in class). For our purposes, the K^+ channel is most important, $I_K = g_k w(V - V_K)$, where g_k is the maximum conductance and V_K the Nernst potential. We can solve an ordinary differential equation for the mean fraction of open K^+ channels (w), resulting in a **deterministic** model:

$$\frac{dw}{dt} = [w_\infty(V) - w]/\tau_w \ .$$

You should get the ode file (**deterministic_ML.ode**) from my website at www.math.fsu.edu/~bertram/course_software, for use with XPP. (Use this version rather than the one you wrote earlier in the semester, since parameter values are somewhat different.)

If the cell has only a few K^+ channels, then it is not appropriate to use the mean open fraction. Instead, channel gating should be simulated as a stochastic process, using a Monte Carlo approach.

(a) Write a computer program that uses forward Euler (with time step $\Delta t = 0.1$ msec) to solve the voltage equation, and that uses a Monte Carlo approach to solve for the fraction of open K^+ channels w. Note that w must be updated at each time step. Use a population of $N = 50$ K^+ channels and set $I_{ap} = 80$. Carry out the simulation for 1000 msec. Turn in a figure with two panels. The top panel should show V vs. t as calculated with the stochastic model, and superimposed on this should be V vs. t calculated with the deterministic model (solved using XPP, making sure that you use the right initial conditions). The bottom panel should show w vs. t for both models. You will most likely see occasional large fluctuations in V and w in the stochastic model. Why do these occur?

(b) Turn in figures similar to the one above with $N = 200$ and $N = 1000$ K^+ channels. [For $N = 1000$ reduce the time step to 0.01 to account for the much larger number of channels.] Describe the changes in system behavior that occur as the number of channels is increased.

(c) Now increase the applied current to $I_{ap} = 150$. This will put the deterministic system into an oscillatory state. Turn in a figure showing superimposed solutions of the deterministic and stochastic models ($N = 50$). Describe how the behaviors of the models differ from one another.

(d) Next, change the parameters $v3 = 30$, $v4 = 60$, and $I_{ap} = 80$. How do the nullclines and steady states of the system compare with those calculated with the original parameter values of $v3 = 2$, $v4 = 30$, and $I_{ap} = 80$? Make the parameter changes in your stochastic program. Turn in a plot showing superimposed solutions of the deterministic and stochastic models ($N = 50$). Describe how the behaviors of the models differ from one another. What is the reason for the difference?

Pseudo-Random Numbers

Pseudo-random numbers can be generated in many ways. One algorithm often used is called a *linear congruential generator*. This generates a sequence of integers I_1, I_2, ..., each between 0 and $m-1$, using the recurrence relation

$$I_{j+1} \;=\; aI_j + c \pmod{\mathrm{m}} \;.$$

Here m is the *modulus*, and a and c are positive integers called the *multiplier* and the *increment*, respectively. The sequence of integers will eventually repeat, with a period no greater than m. However, if m, a, and c are chosen correctly the period will be maximal (m). In this case, all possible integers between 0 and $m-1$ occur at some point, so any initial "seed" choice of I_0 is as good as any other. Different seeds will generate different sequences of (pseudo) random numbers. For more information, see Numerical Recipes by Press et al.

Both C and FORTRAN compilers implement linear congruential generators through the function "rand()". In C, you can set the seed at the beginning of the program with the function srand. A good way to do this is to set the seed according to the current time. So in C, declare the variable *seed* as an integer, then set the seed to the current time:

```
int seed;
seed=time(NULL);
srand((int)seed);
```

You then get your first random number "ran" later in the code with the command:

```
ran=rand();
```

This will return an integer between 0 and $m-1$=Rand_max-1. In most cases, we want a real number between 0 and 1. To get such a number, use:

```
ran=rand()/(RAND_MAX+1.0);
```

This will give a number uniformly distributed between 0 and 1.