BE567 Fall 2010 Lab Assignment 3

For this lab you will need to create multiple Matlab M-files. The first type will be scripts, which call functions in other files. When asked to include answers as comments, place them in the main scripts. Additional comments that clearly explain the purpose of individual commands will be considered in grading for partial credit. The set of files should be grouped in a single archive before being uploaded to the Blackboard Dropbox by 5pm on the due date; the most common archiver is ZIP, available under Windows, Mac and Linux. The archived file should be named

`be567_lab3_LastFirst.zip`

For example, my file name would be

`be567_lab3_RittJason.zip`

If you are unsure about using the Blackboard Dropbox feature, you should test the system by uploading a file named "test" prior to the deadline.

NOTE: Students may discuss lab projects in general terms with each other, but may not share answers, written materials, code, or files of any kind. When in doubt, defer to the instructor or teaching fellow.

This lab assignment assumes a basic familiarity with Matlab. Any students without Matlab experience should contact the professor and/or teaching fellow as soon as possible.

In this lab we will demonstrate numerical experimentation, in which we use the ODE solver as if it returned information from an experimental preparation under different trial conditions, rather than try to visualize the entire phase space at once. This approach is often taken to assess the validity of a dynamical model under conditions more general than that in which it was derived, or to produce hypotheses for new experiments in the real system if the model is already believed to be accurate for the system. When models become sufficiently complex or high dimensional, numerical experimentation is usually the first and sometimes only option to understand its behavior. Accordingly, we will explore a much more complicated example system than we have encountered previously, although still simple compared to many simulations encountered in typical research problems.

Problem 1. The first part of the lab walks through the coding of the Hodgkin-Huxley model for "action potential" conduction in a neural axon. As we discussed in class in the context of the Fitzhugh-Nagumo model, we consider a small section of the membrane of a neuron. The membrane separates internal and external solutions, and contains a number of ion channels, which selectively allow various ion species to cross the membrane. The distribution of ions on either side of the membrane results in a chemical gradient (each ion species tends to diffuse down its concentration gradient) and an electrical gradient (due to the difference in charge carriers on either side of the membrane). Flow of ions produces a current which charges or discharges the capacitive membrane. For the important class of "voltage gated" ion channels, the degree to which they allow ions to flow depends on the membrane potential, and a positive feedback loop between the ionic current and the membrane potential produces a fast ($\sim$1 msec), large amplitude ($\sim$100 mV) voltage swing known as an action potential or "spike". These spikes travel down the axon, carrying a signal that is relayed to other neurons or muscles, with the entire network of these signaling events ultimately producing thoughts and behaviors in some mysterious way yet to be understood. In their original work, Hodgkin and Huxley based their model on recordings from a segment of the "giant axon" of the squid, which in the intact network transmits a signal triggering an escape response. Unfortunately, as with the Fitzhugh-Nagumo model, the voltage convention flipped in the last half century, and so the HH model shows action potential as fast *negative* deflections in membrane potential; I've left the mathematical description in their original form.

The small patch of membrane is modeled by the potential V across it, and the "states" (fraction open or closed) of populations of ion channels. We start with conservation of charge

$$C\frac{dV}{dt} = I_{Na} + I_K + I_l + I_{stim} \tag{1}$$

Here, C is the membrane capacitance and the LHS gives the current due to capacitive charging, I_{Na} is the current due to flow through voltage gated sodium channels, I_K is the current due to flow through voltage gated potassium channels, I_l is the current due to flow through "leak" channels (mostly chloride, and not voltage gated), and I_{stim} is current due to stimulation, which we control through an electrode (in general, one could add other currents due to "synaptic" connections between neurons, and/or other types of ion channels). Each ionic current is assumed to be Ohmic, $I_x = g_x(E_x - V)$, where g_x is the (in general time varying) conductance given by the fraction of open channels, and the constant E_x (called for operational reasons a "reversal potential") represents a driving force produced by the electrochemical gradient of the ion species ($x = Na$,Kl, or l).

The fundamental contribution of the H-H model was the determination of a dynamical form for the various currents that produced the experimentally observed action potential behavior, without corresponding to specific molecular mechanisms. Through a series of elegant experiments, Hodgkin and Huxley isolated the dynamics of each channel type, so that individually they could be fit to directly measurable current-voltage relationships. They modeled the conductances as products of state variables, which themselves followed simple rate equations. For the sodium conductance, $g_{Na} = g_{\bar{N}a}m^3h$, where the constant $g_{\bar{N}a}$ is the maximal sodium conductance, and the "gating variables" m and h satisfy

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m \tag{2}$$

and

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h. \tag{3}$$

Note that the form of the rate equations keep the gating variables between 0 and 1 (think about the signs of the derivatives for values outside this range). Thus powers and products of gating variables are also between 0 and 1, and we see why $\bar{g_{Na}}$ is the maximum conductance (the minimum is zero). The potassium conductance satisfies $g_K = \bar{g_K} n^4$, where the constant $\bar{g_K}$ is the maximal potassium conductance, and

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n. \tag{4}$$

It is the dependence of the rate functions α and β on V that required extensive experiments and curve fitting. Hodgkin and Huxley found

$$\alpha_m(V) = 0.1(V+25)/(e^{(V+25)/10} - 1) \tag{5}$$
$$\beta_m(V) = 4e^{V/18} \tag{6}$$
$$\alpha_h(V) = 0.07e^{V/20} \tag{7}$$
$$\beta_h(V) = 1/(e^{(V+30)/10} + 1) \tag{8}$$
$$\alpha_n(V) = 0.01(V+10)/(e^{(V+10)/10} - 1) \tag{9}$$
$$\beta_n(V) = 0.125e^{V/80} \tag{10}$$

The leak current is simply $I_l = \bar{g_l}(E_l - V)$. The remaining parameters required for a full specification are $C = 0.775\ \mu F/cm^2$, $\bar{g_{Na}} = 120\ mS/cm^2$, $\bar{g_K} = 36\ mS/cm^2$, $\bar{g_l} = 0.3\ mS/cm^2$, $E_{Na} = -115\ mV$, $E_K = 12\ mV$ and $E_l = -10.5989\ mV$.

Physiologically, the sodium current tends to depolarize the membrane (increase the voltage in the modern convention, decrease it in these simulations). The sodium gating variable m increases with depolarizing V (due to the choice of α_m and β_m), thus increasing the sodium current, and this is the source of the positive feedback that leads to the action potential. The "inactivation" gating variable h decreases with depolarizing V, producing negative feedback but at a slower rate, and so clamps down on the sodium current during sustained activation. The potassium current also provides negative feedback, and tends to restore the membrane potential towards its resting value after a spike. Again, remember that the V in the HH model is more or less the negative of the membrane potential as it is defined today. This convention also means that *negative* values of I_{stim} are stimulating, and positive values hyperpolarize (inhibit) the neuron.

One more general remark: the mathematical and conceptual framework of the H-H model is now a universal standard in neuroscience research, but the exact functional forms and parameters have to be fit for each different type of neuron (there are *many* types of neurons...). Also, in more realistic and detailed simulations, the spatial extent of the neuron is modeled by coupling sets of H-H-like equations to capture how voltage changes spread along neighboring segments of the membrane, a practice known as compartmental modeling. In essence, large sets of ODEs are used to approximate a partial differential equation describing how the membrane potential varies both in time and at different spatial locations. It is now common practice to use neural models with thousands of equations (actually, there are now simulations with >10 million equations, but typically at this scale the equations are simplified to sacrifice some accuracy in favor of computational tractability) and many thousands of parameters. The only practical way these models can be compared to measurements from real neurons is to employ numerical experiments like those below (if slightly more involved in practice).

(1A). We could incorporate the entire H-H model in a RHS function handle definition as in previous labs, but the expression would be unwieldy and visually impenetrable. We will instead create a separate function m-file that takes in a point in phase space and returns the values of the derivatives. Create the file `hh_rhs.m`, the first line of which should be

```
function dpdt = hh_rhs(t,p)
```

The input argument `p` and output argument `dpdt` follow the same logic as the output of the function handle in the previous lab. In this case, `p=[V; m; h; n]` is a point in the four dimensional phase space of the model, and the four elements of the vector `dpdt` must be set to the RHS of the ODEs corresponding to this point. We can get a numerical solution via a call to `ode45` such as

```
[t,p] = ode45('hh_rhs',[0 50],[0; 0; 1; 0.5]);
```

Note the primary difference from the call with a function handle is that the name of the function is passed into `ode45` as a string (that's what the quotes `'` are for).

The value returned by the RHS function is whatever `dpdt` is last set to before the end of the file. All other variables are invisible outside the function (e.g., they do not show up in the workspace). One way to keep organized is to compute intermediate steps using variables with suggestive names, building up to an expression for `dpdt` on the last line. The pseudo-code might look like

```
function dpdt = hh_rhs(t,p)
V=p(1);m=p(2);h=p(3);n=p(4);
Define all parameters, including Istim
Compute gNa, gK, gl
Compute Ina, Ik, Il
Compute dVdt = (Ina+Ik+Il+Istim)/C;
Compute dmdt, dhdt and dndt
Set dpdt = [dVdt; dmdt; dhdt; dndt];
```

Note: `dpdt` must be a *column* vector (that is the reason for including the semicolons in the vector definition). Following this pseudo-code and the definitions above (for now setting `Istim=0;`), create the RHS function file.

(1B). Next we will call `ode45` and plot the results, but we need to do one more housekeeping step concerning the initial condition. Create a script (in a new file `run1_hh.m`) that sets a solution time range `Tsolve=[0 50];` and sets an initial condition `init_cond=[0; 0; 1; 0.5];`. Find a numerical solution via

```
[t,p] = ode45('hh_rhs',Tsolve,init_cond);
```

V is in volts, while m, n and h are dimensionless variables between 0 and 1. To plot them together despite these different scales use subplots

```
figure
subplot(2,1,1)
plot(t,p(:,1)) % V(t)
xlabel('Time (msec)')
ylabel('Membrane potential (mV)')
subplot(2,1,2)
plot(t,p(:,2),t,p(:,3),t,p(:,4)) % m(t), h(t), n(t)
legend('m','h','n')
xlabel('Time (msec)')
ylabel('Gating variables')
```

Our choice of initial conditions was ad hoc, but at the end of the above simulation you should see the system relax to the unforced ($I_{stim} = 0$) steady state, a more natural initial condition for later experiments (if you didn't see an approach to a fixed point, there is likely an error in your RHS function). Instead of trying to compute the steady state analytically, we reset our initial condition to the ending value of this "trial" solution, sometimes called "burning in" the initial condition: `init_cond = p(end,1:4);`. If you call `ode45` with this new initial condition you should get constant solutions, because you will be starting at (really, just near) the fixed point.

Instead of having to burn in the condition every time you run the script, manually replace the first definition of `init_cond` with the values you get from running `ode45` once. You will need to view the values separately because V and the gating variables are on different scales (e.g. type `disp(p(end,1))`, `disp(p(end,2:4))` at the command line). We will keep this initial condition fixed for the rest of Problem 1.

Note: even after burning in for 50 time units, we will still be far enough from the fixed point that starting from "rest" will produce small voltages fluctuations on the order of $10^{-3}\ mV$. We will ignore these; an action potential goes through hundreds of millivolts. Pay attention to the voltage scale!

(1C). Now we have the preliminaries out of the way and can begin the experiments. Thanks to our choice of initial conditions, if we set `Istim` to some non-zero value in the RHS function, a new call to `ode45` will return a solution corresponding to "turning on" stimulation at time zero for a system initially (e.g. at negative times) at rest. Thus we can model the effects of step current inputs simply by keeping the same initial conditions and repeatedly solving the ODE for different choices of `Istim`. Note: the most direct way to vary `Istim` is to change it in `hh_rhs.m` and resave the file each time before calling `ode45` from `run1_hh`. If you know how to use global variables (or read the help files on how to pass additional arguments into `ode45`), you can do this efficiently in loops within `run1_hh.m` without having to alter `hh_rhs`.

To within two decimal places, find the maximum value of `Istim` for which the solution includes at least one action potential during time interval `[0 50]`, and list this value in a comment (recall that stronger stimulation occurs for more negative values of I_{stim}). At this stimulation value, is the system excitable (spikes followed by a steady state voltage) or oscillatory (periodic spiking)? What do solutions do when the stimulation current is 0.01 above this minimum value? What type of bifurcation does this suggest occurs as `Istim` is decreased (why)?

(1D). In a comment, describe the changes in action potential shape and frequency as `Istim` is made increasingly negative (include approximate ranges of `Istim` values in your descriptions). Is there a value for which action potentials cease to occur? Is there a threshold at which this change occurs?

Problem 2. Copy `run1_hh.m` to `run2_hh.m`. Following the same procedure as above, burn in the initial condition corresponding to `Istim=4`. Using this initial condition, compute and plot the solution for `Istim=0`. Describe the behavior of the solution in a comment. How would you describe what is happening physiologically, given the decsription of the gating variables above (you might also want to think back to our analysis of the Fitzhugh-Nagumo and van der Pol equations)? Can you get the same behavior using the resting initial condition from Problem 1 for m, h and n but with $V(0)$ replaced by the new initial condition? Physiologically, what is the difference between these two situations? Is there a new value for $V(0)$ that does produce an action potential? Why?

Problem 3. Now we return to thinking about the dynamics in general. Although the H-H model is four dimensional, we can ask if the behavior is "really" 4D, or hovers close to a lower dimensional manifold in the full phase space.

(3A). Add new plots to `run1_hh.m` that compare pairs of variables during the action potential, that is, plot `p(:,j)` against `p(:,k)` for the 6 distinct choices of `j` and `k`, with `Istim` above threshold and the initial condition at rest. Be sure to label the x and y axes correctly. Are there pairs for which the resulting projection of the solution curve looks close to being a plot of a single valued function (i.e. such that for each x value there is nearly just one y value)? Put your answer in a comment line.

If so, we could try a simplifying approximation in which one of the variables is replaced with an algebraic function suggested by this plot. Recall we set $y = F(x)$ as an approximation for part of the limit cycle solution of the van der Pol oscillator, where F was the cubic function giving the nullcline, because the x variable "instantaneously snapped to" the nullcline while y slowly varied. Through numerical experimentation in this more complicated system, we might find a similar behavior even if the appropriate function is not obvious a priori.

(3B; Extra Credit). Find an appropriate algebraic approximation for one of the dynamic variables in terms of the others, and numerically solve the resulting 3 dimensional system starting from rest, with a superthreshold stimulation current. How does the solution compare to that for the full system?

Problem 4 (Extra Credit). It takes a certain amount of time for a superthreshold stimulation current to elicit an action potential. Try replacing `Istim` with a single square wave in time, such that you can vary both the amplitude and duration of the step, and use numerical experimentation to determine how the action potential threshold covaries with these two parameters. You should end up with a plot of the minimal amplitude required for each choice of duration to produces at least one action potential.