BE 130 Problem set 3 - Due Tuesday April 16

(1) Numerically approximating solutions to Minimum Jerk et al using gradient descent.
Numerically solve the minimum acceleration, minimum jerk, and minimum snap problems using gradient descent. Compare these solutions to each other and to the analytically-derived solutions from problem set 2.
Do this **(a)** using lsqnonlin in matlab for all three problems and **(b)** using your own hand-written gradient descent function for the minimum acceleration problem. For **(b)**, comment on you chose / settled on the learning rate, and examine (and comment on) what occurs during the gradient descent trajectory – for this it may be helpful to plot out the current solution every 100 iterations or so during convergence as an animation.

(2) The brachistochrone problem
(a) Analytically solve the brachistochrone (minimum descent time) problem discussed in class. **(b)** Numerically solve the brachistochrone problem with friction. You can again use the lsqnonlin function in MATLAB. As a reminder this is the problem of designing the shape of the (curved) ramp that allows a ball to roll to a particular downhill point as fast as possible. Plot the results for values of the coefficient of friction ranging from 0 to 1 (for example, u=[0:0.2:1]) **(i)** for an endpoint that is displaced by 1 unit laterally and 1 unit vertically and **(ii)** for an endpoint that is displaced by 2 units laterally and 1 unit vertically.

(3) Numerical Simulations of the Full Hodgkin Huxley Model
Using Matlab, simulate action potentials from the Hodgkin Huxley model. Note this model is essentially a 4th order differential equation (4 coupled 1st order differential equations – 3 nonlinear & 1 linear), so use one of Matlab's differential equation solvers (ode45, ode23, etc) to simulate the HH model. To use ode23, for example, you will need to write a matlab function (m-file) that takes time (t), and a state vector = [E, m, h, n] as input returns as output the time derivative of the state vector (4 elements: dE/dt, dm/dt, dh/dt, dn/dt). For initial conditions, use the resting potential for E (-61.2mv), and the asymptotic values of m, h, & n at E=-61.2mV. Note that for the nominal parameter values given in the accompanying handout, if we suppress the units then the equations work out to give time in milliseconds, potential in millivolts, and current in microamps/cm^2. Note also that the form the of the activation and inactivation kinetics in the Hodgin Huxley model is the same for the m, h, & n particles – the only difference is their dependence on voltage (E) derived from the different dependences of a(E) and b(E) on voltage.

(1) Plot minf, hinf, and ninf as a function of voltage (over the range of E_K to E_{Na}). Label the plot and indicate the resting potential.

(2) Stimulate the model with a constant current of 1 to 10 microamps/cm^2 and plot the membrane potential for 30 msec. You can use a command like:

```
for k=1:10, i_const=k; [T,Y] = ode45('hh_fun',[0:.001:30],y0); subplot(5,2,k); plot(T,Y(:,1)); end
```

(3) Stimulate with a current pulse of 1 msec duration and magnitude of 10 microamps/cm^2, and then plot membrane potential, and the sodium and potassium conductances as function of time.

(4) Stimulate with a current pulse of 1 msec duration and a magnitude varying from 0 to 10 microamps/cm^2. Plot the relationship between stimulating current pulse magnitude and peak membrane voltage.

(5) Stimulate with an initial 10uA current pulse of 1 msec duration and then with a second pulse with magnitude varying from 1 to 20 ([1:20]) microamps/cm^2, and then plot the relationship between current pulse magnitude and peak membrane voltage for the 2nd pulse when it comes 1-30 ms after the first pulse (1ms intervals [1:30]).

(6) Plot the relationship between firing rate and stimulating current (for constant (non-pulsed) stimulating currents). To do so you'll need to determine a reasonable range of current densities to use.

For each part above write a terse (1-3 sentence) description & explanation of the behavior observed.

Notes / Hints:

Problem 1:

The 'lsqnonlin' function (which stands for: the **L**east **SQ**uares solution for a **NON-LIN**ear function) searches (using a version of gradient descent) for the parameter set p that will allow the error function be as small as possible in a least squares sense, which amounts to minimizing sum(f(t,p).^2). Since we want to minimize CSJ = sum(jerk.^2) we can pass the jerk function to lsqnonlin as f(t,p) and set f*(t)=0. Note that using the 'lsqnonlin' with an anonymous function for jerk in matlab can allow you to code this very efficiently although the syntax takes a little getting used to. Thus for the minimum jerk part of problem 4a, note that jerk is the 3rd derivative of position and so can be coded in matlab as jerk = diff(x,3)/(dt)^3 for a given x.

Note that dt is fixed so 1/(dt)^3 is a constant, so minimizing CSJ=sum(jerk.^2) is the same as minimizing sum(diff(x,3).^2), so we can write something like:

```
N=1000;          % N = the number of free params (time steps) in the function that we will optimize
x0=zeros(1,N);   % assign an initial starting point for the optimization
                 % note x0 could be anything, e.g. we could instead have x0=randn(1,N) or x0=ones(1,N)
                 % but x0 must have the right size (& our csj function below assumes it will be a row vec)
csj = @(x,xf) diff([0,0,0,x,xf,xf,xf],3);  % 'anonymous' one-line function for csj
                                           % note that we pad x on both sides so that the
                                           % initial & final conditions are automatically fulfilled
x = lsqnonlin(@(x) csj(x,10), x0);   % run the optimization
x_ = [0,x,10];                       % augment the optimized function with the initial & final positions
Tf=0.5; t=[0:(N+1)]/(N+1)*Tf; dt=Tf/(N+1);  % construct the time vector from 0 to Tf, and determine dt
figure(100);
subplot(311); plot(t,x_); ylabel('Position'); hold on;                    % plot the result
subplot(312); plot(t(2:end),diff(x_)/dt); ylabel('Velocity'); hold on;
subplot(313); plot(t(2:end-1),diff(x_,2)/dt^2); ylabel('Acceleration'); xlabel('Time (sec)'); hold on;
```

See notes for problem 3 on the following pages…

Hodgkin-Huxley Action Potential Model

This formula is used to calculate the membrane potential assuming some initial state. The calculation is based on Sodium ion flow, Potassium ion flow and leakage ion flow (which lumps all the less significant & non-voltage-dependent (NVD) currents – mainly the NVD K+ & Cl-). If there is a mother of computational neuroscience this is it. It resulted in an Noble Prize for the authors, and more than half a century later it is still probably the single important quantitative model in neuroscience (Hodgkin, A. L. and Huxley, A. F. (1952) "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve" Journal of Physiology 117: 500-544)

The Main formula

$$C\frac{dV}{dt} = I_{tot} = I_{Na} + I_K + I_L + I_{ext} = -g_{Na}(V - V_{Na}) - g_K(V - V_K) - g_r(V - V_r) + I_{injected}$$

The Variable Definitions

The parameter names in bold are fixed variables.

- C: Membrane capacitance (about 1 microFarad/cm^2)
- I : the total ionic current across the membrane (in microAmps/cm^2)
- m : the probability that 1 of the 3 required activation particles has contributed to the activation of the Na gate (m^3 : the probability that all 3 activation particles have produced an open channel)
- h : the probability that the 1 inactivation particle has *not* caused the Na gate to *close*
- **G_Na** : Maximum possible Sodium Conductance (about 120 mOhms^-1/cm2)
- **E** : total membrane potential (about -61.2 mV)
- **E_Na** : Na membrane potential (about 55 mV)
- n : the probability that 1 of 4 activation particles has influenced the state of the K gate.
- **G_K** : Maximum possible Potassium Conductance (about 36 mOhms^-1/cm2)
- **E_K** : K membrane potential (about -72 mV)
- **G_L** : Maximum possible Leakage Conductance (about .3 mOhms^-1/cm2)
- **E_L** : Leakage membrane potential (about -50 mV)

Parameters and Values

Parameter	**Value**
G_Na 120	mOhms^-1/cm2
G_K 36	mOhms^-1/cm2
G_L .3	mOhms^-1/cm2
E	-61.2 mV
E_Na	55 mV
E_K	-72 mV
E_L	-50 mV

The Activation Particle probability(m) for Sodium

$$\frac{dm}{dt} = a_m (1 - m) - b_m m = \frac{m_{inf} - m}{T_m}$$

$$T_m = \frac{1}{a_m + b_m} \qquad m_{inf} = \frac{a_m}{a_m + b_m}$$

Empirical Formula for the workings of the Sodium gate activation (m)

$$a_m(E) = \frac{-.1\,(40 + E)}{\exp\left(\frac{-(40 + E)}{10}\right) - 1}$$

$$b_m(E) = 4 \exp\left(\frac{-(65 + E)}{18}\right)$$

Variable Definitions

- m : the probability that 1 of the 3 required activation particles has contributed to the activation of the Na gate (m^3 : the probability that all 3 activation particles have produced an open channel)
- t : time in msec
- a_m : rate constant for particle not activating a gate
- b_m : rate constant for particle activating a gate
- m_inf : the steady state value of m (m-infinity)
- T_m : the time constant of m
- E : total membrane voltage
- a_m : rate constant for particle not activating a gate
- b_m : rate constant for particle activating a gate

The Inactivation Particle probability(h) for Sodium

$$\frac{dh}{dt} = a_h (1 - h) - b_h h = \frac{h_{inf} - h}{T_h}$$

$$T_h = \frac{1}{a_h + b_h} \qquad h_{inf} = \frac{a_h}{a_h + b_h}$$

Empirical Formula for the workings of the Sodium gate inactivation (h)

$$a_h(E) = .07 \exp\left(\frac{-(E+65)}{20}\right)$$

$$b_h(E) = \frac{1}{\left(\exp\left(\frac{-(35+E)}{10}\right)+1\right)}$$

The Variable Definitions

- h : the probability that the 1 inactivation particle has *not* caused the Na gate to *close*
- t : time in msec
- a_h : rate constant for particle inactivating a gate
- b_h : rate constant for particle not inactivating a gate
- h_inf : the steady state value of h (h-infinity)
- T_h : the time constant of h
- E : total membrane voltage
- a_h : rate constant for particle inactivating a gate
- b_h : rate constant for particle not inactivating a gate

The Activation Particle probability(n) for Potassium

$$\frac{dn}{dt} = a_n(1-n) - b_n n = \frac{n_{inf} - n}{T_n}$$

$$T_n = \frac{1}{a_n + b_n} \qquad n_{inf} = \frac{a_n}{a_n + b_n}$$

Empirical Formula for the workings of the Potassium gate activation (n)

$$a_n(E) = \frac{-.01(55+E)}{\exp\left(\frac{-(55+E)}{10}\right) - 1}$$

$$b_n(E) = .125\exp\left(\frac{-(E+65)}{80}\right)$$

The Variable Definitions

- n : the probability that 1 of 4 activation particles has influenced the state of the K gate.
- t : time in msec
- a_n : rate constant for particle not activating a gate
- b_n : rate constant for particle activating a gate
- n_inf : the steady state value of n (n-infinity
- T_n : the time constant of n