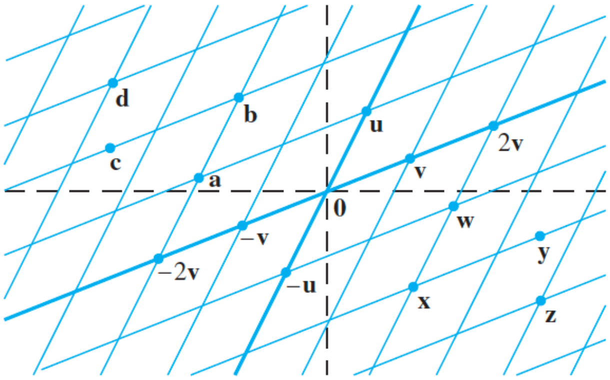
Neuro 120 Homework 1: Math and Hodgkin & Huxley

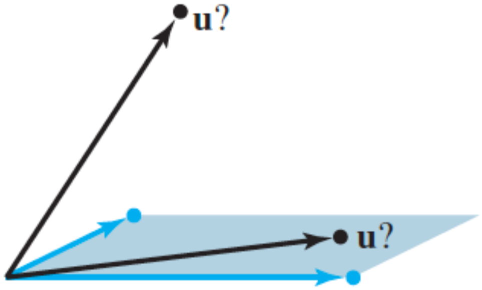
Due: Thursday 4 October 2018 at 5 pm

1.



Write each vector as a linear combination of **u** and **v**.
Is every vector in $\mathbb{R}^2$ a linear combination of **u** and **v**?

2.



Let $\boldsymbol{u} = \begin{pmatrix} 0 \\ 4 \\ 4 \end{pmatrix}$ *and* $A = \begin{pmatrix} 3 & -5 \\ -2 & 6 \\ 1 & 1 \end{pmatrix}$.
Is $\boldsymbol{u}$ in the plane in $\mathbb{R}^3$ spanned by the columns of A? Why or why not?

3. Are the column vectors linearly independent?

$$\begin{pmatrix} 1 & -2 & 3 & 2 \\ -2 & 4 & -6 & 2 \\ 0 & 1 & -1 & 3 \end{pmatrix}$$

4. For what value of x are these vectors linearly dependent?

$$\begin{pmatrix} 1 \\ -2 \\ -4 \end{pmatrix}, \begin{pmatrix} -3 \\ 7 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ x \end{pmatrix}$$

5. The matrix $A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ defines a linear transformation from $\mathbb{R}^2 \rightarrow \mathbb{R}^2$.

What does it do to the vectors $\boldsymbol{r} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$, $\boldsymbol{s} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$, and $\boldsymbol{r} + \boldsymbol{s}$? Illustrate the result.

6. Repeat for matrices

$B = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$, $C = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, $D = \begin{pmatrix} .5 & 0 \\ 0 & 2 \end{pmatrix}$, $E = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$, $F = \begin{pmatrix} 1 & 0 \\ .5 & 1 \end{pmatrix}$, $G = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$

In the programming section of this homework, you will simulate the Hodgkin-Huxley neuron model and investigate its dynamics in response to different kinds of inputs. Starter code for the assignment is provided on GitHub. Work in groups of 2-3 students. You may use any resources or code you found online, but must properly attribute it. Submit all code you write and the plots you produced to generate your answers.

1. **Numerical Integration**. The HH model is a nonlinear differential equation that can be written as

$$\frac{dy}{dt} = f(y,t) \qquad (1)$$

where the function $f(y,t)$ takes in the current state of the system and returns the temporal derivatives of each state variable. Here the state vector y is a column vector with four elements $y = [v\, m\, h\, n]^T$ corresponding to the state variables in the HH model.

The differential equation above is in continuous time (t can be any real number), but to simulate this on a computer we must approximately solve it at a discrete set of time points. The starter code provides several pieces necessary to simulate the HH dynamics, but is not quite finished. In particular, the le simulate_hh.m contains the main function for a simple simulation. The file euler_solver.m will solve the differential equation using Euler integration. The file hh_deriv.m contains the derivatives of each variable (it is the function f above). You should look through all the les to see how each calls the other.

To simulate the dynamics, your task is to write a simple Euler integrator to approximately solve the differential equation by filling in the blanks in the file euler_solver.m. Remember that a derivative is the rate of change of a function. Therefore, the change in state y from $y(t)$ to $y(t + t)$ a short time t later is

$$\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} \;. \qquad (2)$$

Using this in Eqn. 1, we have

$$\frac{\Delta y}{\Delta t} \approx f(y,t) \qquad (3)$$

$$\Delta y \approx f(y,t)\Delta t \qquad (4)$$

This is the basis of Euler integration, with time step Δt. In words it says that the change in the state over a short period of time Δt is equal to the derivative of the state multiplied by that time step. As the time step goes to zero, the solution becomes more accurate. To solve the differential equation from some starting initial state $y(0) = y_0$ until some final time T, you first initialize the state vector $y = y_0$; and then use a for loop to repeatedly update y using Eqn. 4.

(a) Implement euler integration. The file simulate_hh.m contains a flag use_euler which can be set to false to use a more sophisticated built-in ODE solver in matlab. You can use this to check your work. The basic settings of simulate_hh.m will simulate a HH model receiving a step input current and ring a series of spikes in response.

(b) Test the effect of the Euler integration step size (which is set in simulate_hh.m). The default used by simulate_hh.m is $\Delta t = 0.01$.

What happens when $\Delta t = 0.09$? What happens when $\Delta t = 0.07$?

2. **Basic spike generation**. With your simulator in hand, (or by setting use_euler=false), investigate the spiking behavior of the HH model. Zoom in to one action potential. Explain the generation of this action potential by examining the gating variables.

3. **Firing rate for constant inputs**. Now investigate the behavior of the HH model for different sustained input currents.

   (a) Generate a plot of the firing rate as a function of the applied input current. See the "configure input current" section of simulate_hh.m to adjust the input current. Try currents in the range [0 15]. You can do this by hand or by writing code to do it. The file simulate_hh.m contains a section which estimates the firing rate.

   (b) What is the minimum sustained firing rate for the HH model?

4. **Refractory period**. Uncomment the line in simulate_hh.m which corresponds to an input current of two brief pulses.

   (a) How many spikes are fired in response to two pulses separated by 12ms?

   (b) How many spikes are fired in response to two pulses separated by 3ms?

   (c) By investigating the gating variables, explain the source of this phenomenon.

5. **History dependence**. So far, we have looked at input currents that start at zero and at some time change to another value. What happens when a HH neuron has received a nonzero input for a long time and then steps to a new value?

   (a) Uncomment the input current which begins at $0\ \mu A$ and rises to $8\ \mu A$. What is the firing rate (after the rise to $8\ \mu A$)?

   (b) Uncomment the input current which begins at $10\ \mu A$ and drops to $8\ \mu A$. What is the firing rate (after the drop to $8\ \mu A$)?

   (c) Generate a firing rate curve as in part 2a, but starting at $10\ A$, and compare to that in 3a.

   (d) Uncomment the input current which begins at $10\ \mu A$ and then turns off ($0\ \mu A$). Explain the resulting behavior by investigating the gating variables.

6. **Oscillations**. Real neurons typically receive complex time-varying inputs. Uncomment the line in simulate_hh.m which corresponds to an input current of $I(t) = I_0 + I_1 \sin(\omega t)$.

   (a) Manipulate ω. What is the firing rate of the neuron?

   (b) Generate a firing rate curve by varying I_0.

   (c) Manipulate I_0 in the range where one spike is fired per oscillation cycle. What effect does I_0 have in this case?

7. **Extra Credit**: Leaky integrate and fire neurons. The full HH dynamics are complex enough that simulating large networks of neurons is difficult, as each neuron requires four variables. Those four variables interact, as you have seen, to create roughly all-or-nothing spikes. One simplification is the following: rather than simulate the exact shape of an action potential, we can instead track whether

   a given neuron has generated a spike or not. In particular, we will say that whenever the membrane potential v(t) crosses a threshold v_{th}, a spike is instantaneously emitted and the membrane potential immediately resets to 0. Otherwise, the dynamics of the membrane potential evolve as

$$C\dot{v}(t) = -v(t) + I(t).$$

   (a) Find an analytical expression the firing rate of this neuron as a function of the input current for a constant input current $I(t) = I_0$.

   (b) (Warning: Much harder) Consider inputs of the form $I(t) = I_0 + I_1\, sin(\omega t)$. Find an expression for the phase at which a spike is emitted relative to the input sinusoid, for the regime where one spike is emitted per oscillation cycle.