

PSYCH-GA.221 / NEURL-GA.2201 – Fall 2024
Mathematical Tools for Cognitive and Neural Science

MATLAB Lab 1 (Homework 0)

Due: 10 Sept 2024

Welcome to MathTools! In this lab, you will try out some exercises related to stuff you learned this week. You can submit the exercises as an optional homework. If you choose to submit answers, the TA's will give you feedback on code correctness, programming style, and submission format, which will help you perform better on later non-optional homeworks. [\[Click here for submission instructions\]](#)

All homeworks can be done in either MATLAB or Python. For Python, try to find equivalent numpy (often imported as `np`) and matplotlib.pyplot (often imported as `plt`) commands for the various MATLAB functions.

If you're already comfortable with the programming language of your choice, you can just go ahead and start with an empty script. If you don't know where to start and want some reminder of programming basics, please refer to the tutorial files in the folder.

1. Projections, change of coordinates, and angles.

- (a) Write a function, "project" that takes two non-zero vectors, v and unit vector u as input and projects v onto u . Have the function output the length of the projected vector. Test the function on a few known cases.
- (b) Write a function, "changeOfCoords" that takes two 2D vectors, v and u as input. Create a new "coordinate system" such that the vector u is one of the axes (you'll need to make this a unit vector, and generate the other axis, an orthogonal unit vector w). Express the coordinates of v in the coordinate system defined by $[u, w]$. The function should return these coordinates, along with the matrix $[u, w]$. Test the function by showing that rescaling the new coordinate axes by these coordinate values and adding the produces the original vector.
- (c) Write a function, "angleBetween" that takes two vectors, v and u , and calculates the angle between them. Verify the answers on a few known cases. Hint: the inverse cosine function `acos()` may prove useful.

2. Plotting a Unit Circle A unit circle is a circle of radius 1, centered on the origin. When we get into matrix transformations, circles are often a nice way to visualize how a space can be transformed by a matrix. Let's think about how to create the unit circle. HINT: Use trigonometry!

3. Vectors as operators Imagine there is a neuron with three dendrites. Each dendrite receives the same 8 inputs (has 8 synaptic terminals), but uses a different computation on those inputs to determine whether it should signal the cell to fire. Dendrite 1 acts as a simple averager. Dendrite 2 acts as windowed averager, only factoring in the last four inputs, Dendrite 3 acts as a component selector and only extracts the value of the last (eighth) dimension. The cell decides to fire if a majority of the dendrites surpass a threshold.

- (a) Create a function `neuron()` which takes a 8x1 input vector and a scalar threshold, and outputs the cell's decision to fire as a boolean value (0=no firing, 1 = firing). Randomly generate an input vector and test your function using an appropriate threshold value.
- (b) Now imagine that instead of those 8 inputs representing different synaptic terminals, they represent inputs over time. Write a function `neuronOverTime`, that takes as an input a threshold and a scalar `T`, a length of time. Randomly generate a vector of inputs with the length `T`, and use the 'neuron' function to generate the cell's response at each successive time point by calculating the response to subset of inputs, `input(t:t+8)`. To measure the activity of the neuron at time $t = 1$, pad the input vector with zeros so that the first random number is input 8.
- (c) Create a vector of output values (1s and 0s) and plot them as a "spike plot" using the function `stem()`.