

# Series4 L2-structure worksheet

Sudhanshu B

December 14, 2024

This worksheet is intended to be a quick introduction to analyzing the L2 structure; and consists of a bunch of basic problems/tasks. Coding/answering these questions like an assignment will help you work with the L2 structure and neural data in general. There are two sections in this worksheet. Section-A contains some simple operations to just get used to the L2 structure, whereas Section-B involves specific analysis that are commonly used. The appendix contains some additional material as well, with A.1 and A.3 (an assignment-0 for people who are new to MATLAB) being quite important

Note: You might require knowledge of additional concepts (or not know what a term means), so do google/YouTube/chatGPT along the way, or ask any one of us for help. Infact learning these additional concepts online will teach you a lot . So take time, and make the most of this worksheet. All of you would be using visual channel map data, so this a great opportunity to discuss with other interns. I hope you have a fun time solving some of these questions! :)

## Contents

<b>Setup</b>	<b>1</b>
<b>Introduction</b>	<b>2</b>
Visual Channel Map . . . . .	2
L2 Structure . . . . .	3
Visionlablib . . . . .	3
<b>A Section-A</b>	<b>4</b>
<b>B Section-B</b>	<b>6</b>
B.1 Response Consistency . . . . .	6
B.2 Face selectivity . . . . .	6
B.3 Face decoding . . . . .	7
B.4 Making sense of population activity using dimensionality-reduction . . . . .	7
<b>A Appendix</b>	<b>9</b>
A.1 A Very brief intro to our neural data . . . . .	9
A.2 Creating L1 and L2 structures . . . . .	9
A.3 TCN Assignment-0 . . . . .	10

## Setup

Before getting started, make sure that everything is setup properly. A basic checklist is listed below:

1. Install Matlab (any version should do), along with all necessary Matlab toolboxes (if it looks important, install it). I can't recall all the toolboxes necessary, but a non exhaustive list is [Parallel Computing Toolbox, Statistics and Machine Learning Toolbox, Curve Fitting Toolbox,

MATLAB Coder, Embedded Coder, Signal Processing Toolbox, Image Processing Toolbox, Simulink]. In addition you also might want to install Circular Statistics Toolbox (external "app")

2. Also install MonkeyLogic. Please install the correct version of MonkeyLogic (from `X:/software`), NOT the publicly released version.
3. Add visionlablib to your MATLAB path. If you don't have visionlablib, it can be found in the files section of Vision lab teams group (under visionlabdocs). This is a shared folder, and you should add it using onedrive so that it updates automatically when any of the library codes are updated. Ensure that the folder on your PC is read-only so that you don't accidentally change these codes.

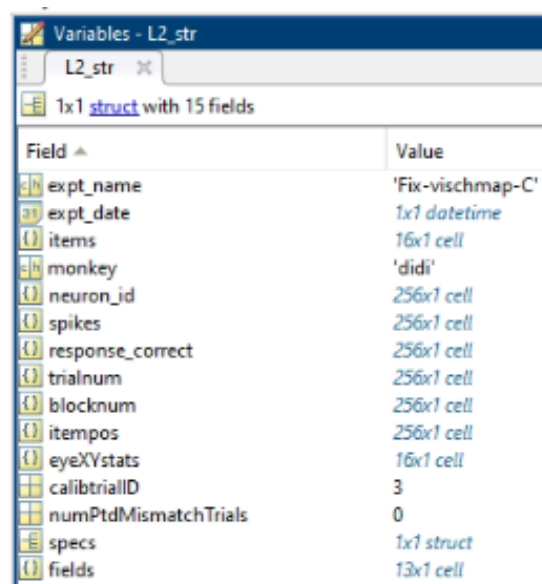
You might have to map the lab network drives to your PC as well.

## Introduction

Note: The terms cells/electrodes/neurons/channels are used interchangeably here. This is fine because in this worksheet you are working with MUA. But, if you have spike-sorted data, the number of neurons and electrodes are not the same.

## Visual Channel Map

This our lab's basic fixation task, that we run on every recording day to look at the consistency of the data. The stimuli consists of 16 cropped images (faces and objects). We collect at least 16 correct repetitions of each stimulus across different trials. Each trial shows 8 random stimuli which the monkey has to fixate on. Each stimuli is shown for 200ms, and there is an inter-stimulus-interval between images is 200 ms. The inter-trial-interval varies between trials, and depends on when the monkey initiates the trials with the hold button.



The screenshot shows the 'Variables - L2\_str' window in MATLAB. It displays a 1x1 struct with 15 fields. The fields and their values are as follows:

Field	Value
expt_name	'Fix-vischmap-C'
expt_date	1x1 datetime
items	16x1 cell
monkey	'didi'
neuron_id	256x1 cell
spikes	256x1 cell
response_correct	256x1 cell
trialnum	256x1 cell
blocknum	256x1 cell
itempos	256x1 cell
eyeXYstats	16x1 cell
calibtrialID	3
numPtdMismatchTrials	0
specs	1x1 struct
fields	13x1 cell

Figure 1: Visual Channel Map in variable view

## L2 Structure

You might remember from the recordings, we have different PC's recording different information (behavioural data, neural data, event-markers etc.). All this data is first combined into the L1-structure. This structure contains all relevant information/data about the experiment. It is organised as a collection of trials, and isn't optimised for analysis. The L2-structure is a more refined/smaller structure and is particular to the task type (fixation, TSD...). These standardised structures ensure that all the data across series4 experiments is organised in a similar way, and the data can be loaded up and analyzed by anyone.

Load up the L2-structure (for visual-channel-map) into MATLAB. It should look something like Fig1. The structure is well documented in the `specs` field. **Please go through the same once.**

The main item of relevance are the spikes (the [appendix](#) contains a very brief intro to neural data). The fixation L2-str contains spikes organised by stimuli (i.e. all repetitions of a stimulus from different trials are grouped together), with *spike-times being relative/aligned to the time the stimulus appears on screen*. The spikes are organised as  $nElectrodes \times nStim \times nTrials$ . Each of these items is an array of spike-times (or a spike-train) relative to stimulus-onset.

Note: This stimulus-onset-alignment across different repeats is not trivial for other tasks, especially in tasks involving motor movements (which takes variable time across repetitions) or other complicated behaviour. For example, even for a TSD (temporal same different) task one could align spike-times - to the onset of the stimulus, or the moment the response is made; both offering interesting analysis, and which you would use depends on the question.

In this light, do think about how one could create an L2-structure for a Nat-task. You might have interesting ideas, and since the library codes haven't been finalized yet, your ideas might help make the L2 structure more useful. Also think about what event your particular Nat-tasks should be aligned to, to best answer its questions

## Visionlablib

This shared onedrive folder contains all the common lab codes. `series4lib` is relevant to series4 data analysis. `visionlablib\series4lib\L2` contains codes to solve many problems given in this worksheet. In case you're stuck on one question, feel free to browse through the library codes to get inspired. `visionlablib\series4lib\wmLib` contains some codes used to create these L2-structure. You can go through them to get a better understand these structures.

## A Section-A

### 1. The stimuli and task:

- (a) Visualise all the stimuli in visual-channel-map (16 images). You will find the stimuli in `L2_str.items`.
- (b) For each of these 16 stimuli, print the number of repetitions collected, and the number of repetitions from correct trials. You will find the necessary information in `L2_str.response_correct`

2. **Firing rate:** The most common way to analyze spike-neural data, is to count the number of spikes in a certain interval, and compute the firing rate of the neuron.

- (a) Each item in the `spikes` field of the L2-str consists of spikes aligned to the stimulus presentation(i.e. stimulus is presented at  $t=0$ ). We only store spikes in a certain interval around this presentation time. Find this interval, `spk_window` , used to create this structure in the L2 structure.
- (b) Create a function to compute firing rates for all the stimuli, averaged across different repetitions. Your function should have the signature

```
function firing_rate = FetchL2Rates_custom(L2,t1,t2,qElectrodes)
% Inputs
%   L2           = L2 structure
%   t1,t2        = start and end of spike window to be used. Only spikes
%                 in this interval should be counted.
%   qElectrodes = indices of electrodes to use
% Outputs:
%   firing_rate = nElectrodes x nstim matrix of firing rates (spikes/s)
```

Basically, in each spike-train count the number of spikes such that  $t1 \leq spike - time \leq t2$ . Average this firing rate across all correct repetitions, to get the trial-averaged firing rate for a stimulus. Do this for all the electrodes listed in `qElectrodes`, and return the resulting matrix of firing rates.

- (c) Using this function, compute the firing rate for all stimuli from electrode 129/198/152, for 50ms bins  $\in [-100 : 50 : 300]ms$ . Plot the firing rate across time intervals as a histogram/bar-plot and mark the stimulus onset and stim off times:  $[0, 200]ms$ . Plot the images on the same figure. Compare the neural response to different stimuli (make sure you use the same scale across different subplots). Do you notice any selectivity or preference that different neurons have?

*This firing rate across time is also called the PSTH: Peri-stimulus time histogram.*

- (d) Find the peak response time (when the mean firing rate is the highest) using the PSTH for all IT electrodes. Plot a histogram of the response latencies for these 128 electrodes.

3. **Visualising data using global rasters and stimplots:** Visualising the firing of neurons, and noticing certain selectivity patterns/trends is a very important step in neural data analysis. Once you notice certain patterns that match your hypothesis, you can then code up the relevant analysis.

- (a) Plot the stimplots and L2 global rasters using `plotL2_stimplot` and `plotL2_globalraster` and save them

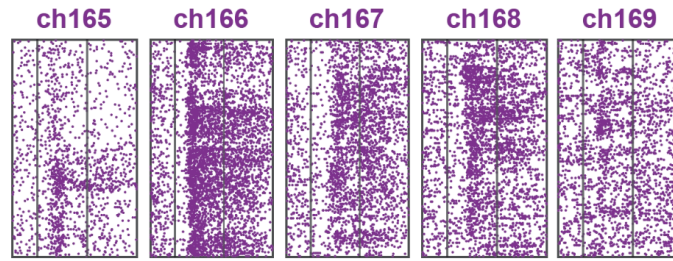


Figure 2: Example Rasterplots for some channels. It marks spiketimes on the x axis ( $xlim = [-100, 400]ms$ ). The black lines indicate the time the stimuli are turned on and off ( $[0, 200]ms$ ). Each row shows one presentation for a stimulus. The rows are grouped such that repetitions of the same stimulus, are continuous. Thus, if a particular stimulus has higher neural activity, a bunch of continuous rows will show high firing rate.

- (b) The L2 global raster contains the rasterplots for all electrodes. Example rasterplots(for 5 electrodes) are explained in Fig2. Once you understand what is being plotted, go through the stimplot and note down your observations on 8 electrodes which have interesting neural response patterns.
- (c) A **visual responsive channel** in which there is a visually-triggered response. That is, the firing rate shows a consistent change sometime after stimulus presentation. Manually count and report the number of visually responsive channels from the L2 global raster
- (d) The stimplots are similar to the PSTH plots you created in question 2b. Go through the stimplots of all channels, and note down interesting selectivity patterns for at least 8 channels. A Channel is selective if has a higher response only for particular preferred stimuli. Also find 5 visual channels that are selective to faces over non-faces.

## B Section-B

Note: This section consists of some commonly used neural data analysis. It needs some additional effort to learn some new concepts, but is worth the effort. If you have the L2-structure for your experiment, you can get started and create stimplots/globalrasters, and parallelly learn the concepts involved for the next section (Linear Algebra and basic ML) over 2 weeks. This section is generally harder, so feel free to skip certain questions if it's too hard.

For this entire section use the firing rates calculated with a 100ms bin between 100ms and 200ms after stimulus onset.

### B.1 Response Consistency

If the same stimulus is shown twice in different trials, the neural response will be slightly different, because of noise sources (the inherent stochastic nature of neurons, noise that arises because of the recording, and the pre-processing). Some of the noise is reduced because of using trial-averaged firing rates, but it is important to quantify the inter-trial variability, as it sets the upper bound for the various analysis we could perform.

- (a) The most common way to quantify response reliability is using the **split-half correlation**.

Select a good electrode (129/198/152). For each stimulus, calculate the trial-averaged neural response for odd repetitions and even repeats separately. Thus you have two vectors of responses (each of length  $n_{Stim}$ ), one for odd and the other for even repeats of the same images. Calculate the correlation coefficient between these vectors to obtain the split-half correlation.

- (b) Calculate the split-half correlation for all IT electrodes. Find the electrodes with the best 3 split half correlations, and the worst 3 split-half correlations, and observe them on the global-raster.

### B.2 Face selectivity

While looking at stimplots, you would have noticed a few cells that respond selectively to faces. This analysis is to quantify that face selectivity.

- (a) Calculate the response to face stimuli (8) and non-face stimuli (8) for an electrode that looks face selective from the stimplot. Run a simple statistical test to check if the mean response to face and non-face stimuli are different.
- (b) We will quantify the face selectivity using a simple metric, Face selectivity Index

$$FSI = \frac{meanResponse_{faces} - meanResponse_{nonFaces}}{meanResponse_{faces} + meanResponse_{nonFaces}}$$

Calculate this index for all neurons using the 8 non-face and 8-face stimuli. Report the neurons with the highest face selectivity. Look at the stimplots and verify your results.

- (c) Selectivity only measures the difference in neural response between faces and non-faces, it doesn't tell us if the responses discriminate between different faces. To investigate this, pick 3 neurons with high FSI and run a n-way ANOVA between the trialwise neural firing rate to check if the neural response discriminates between different faces.

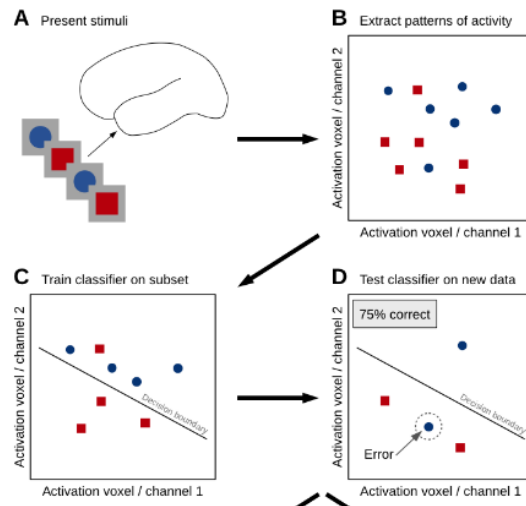


Figure 3: An illustration of neural decoding or classification in neural space. The goal is to learn the boundary between between different classes in the space

### B.3 Face decoding

[Requires knowledge of classification/linear-classifiers/basic ML. You can use the `decodeL2` library function to simplify your work (or you can use the `matlab` function `classify()` if you want to learn more ML). Even if it takes time solve this question because you'll learn basic ML, classification and neural decoding analysis.]

If the response of a neuron is different for face and non-face stimuli, one must be able classify a stimulus to be face/non-face just using the neural response. This can be done by training linear ML models to predict the "label" (face/not-face) on some training stimuli. The model's performance is then tested on some unseen test data to get the accuracy of the model.

Fig3 illustrates the core idea behind behind classification. Suppose you have neural responses( $\hat{X}$ ) to different stimuli which have a label,  $y$  (face/nonface). If neural population encodes properties about the stimuli, different stimuli should be located in parts of this space. The goal of linear classifiers is to find an optimal decision boundary. The decision boundary is a hyperplane that best separates points belonging to different classes. (If your data is 1-dimensional(single neuron), you the best hyperplane is a scalar value of a firing rate that separates face/non-face stimuli.) Now using this decision boundary, new points can be classified as face/non-face with good accuracy.

- (a) Select an electrode with high FSI, and use its neural responses. Label each stimuli with a 0/1 variable to indicate if it is a face stimuli. This is the target variable( $y$ ), which we want to predict using the neural responses( $\hat{X}$ )

Use a leave-one-out method to generate train-test splits between the stimuli, and train the classifier on the train stimuli to discriminate between face/non-face. Test the performance of the model on the test-stimuli (performance is just fraction of correctly predicted labels). Report the mean leave-one-out classification accuracy. Repeat the analysis on a 2 more electrodes with different FSI.

- (b) Select the top-5 FSI neuron, and their neural response. Now  $\hat{X}$  is matrix of size  $nStim \times nElectrodes(5)$ . Use this population response to predict face/non-face labels the same way as in the previous questions. Does the mean leave-one-out accuracy increase?

### B.4 Making sense of population activity using dimensionality-reduction

[Requires knowledge of Principal-Component Analysis(PCA)]



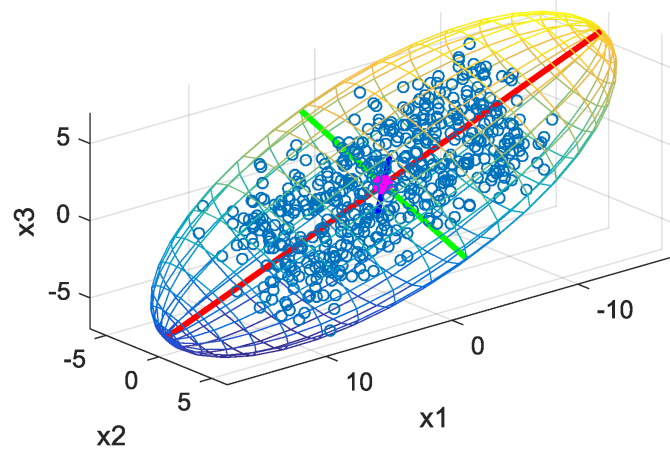


Figure 4: Illustration of PCA. Think of each dimension as the firing rate of one neuron. As seen in the diagram, points spread in 3D space. But because the different dimensions (responses of different neurons) share certain correlations, the points lie mostly along a certain axis - **the first principal component**. Similarly including the **the second principal component** allows us to capture most of the variation in 3D space, in just 2 dimensions. Thus if your data consists of more than 3 dimensions, using dimensionality reduction, you can still view the significant trends/clusters in your data in 2D/3D

So far you have only looked at the responses of single neurons, or different neurons separately. The population activity in neural space (a vector of firing rates of length  $n_{Electrodes}$ ) encodes the combined information across all neurons, and is capable of showing more smaller effects. However, the population data is high dimensional, making it hard to visualize/look at, and analyse (look up curse of dimensionality). That is where dimensionality reduction methods come in handy.

- (a) Preprocessing: Different neurons have firing rates of different magnitudes, and sometimes it might be beneficial to equalize all the scales. In addition, dimensionality reduction techniques expect centered input data ( $\mu_{\hat{x}} = 0$ ), refer to Fig5 for an example. Therefore one usually **z-score**'s the responses of each neuron separately, that is  $X_{z-scored} = \frac{\hat{X} - \mu_{\hat{X}}}{\sigma_{\hat{X}}}$  (the mean and standard deviation are across different stimuli). This also makes the data distribution closer to a multivariate normal distribution, which has several nice properties and is one of the assumptions of PCA.  
Z-score the neural responses for each electrode.

- (b) Dimensionality reduction: There are several methods to reduce dimensionality among which PCA is the most basic, and is briefly explained in Fig4. To summarise, PCA tries to find the axis which capture the most variation in the data (or where the data points are the most spread).

Take the z-scored matrix of neural responses across all IT electrodes, and using PCA, reduce the dimensionality to [2, 5, 10] dimensions. Calculate the cumulative explained variance for different number of output dimensions and plot it. Why does the curve saturate fast?

- (c) Visualisation using PCA: Take the top 2 principal components (PCA to 2 dimensions). Plot them on a scatterplot for different stimuli. Add the stimuli name, or preferably the stimuli image as a marker to this 2D scatter plot. Do you observe any trends/clusters? What visual features do the principal components seem to code for?

*There are better methods to visualize high dimensional data, like MDS, TSNE and UMAP. A common practise is to use PCA to get the dimensionality down to 20-50 dimensions and then use these methods. For a summary of different methods, I usually refer to [scikit-learn's documentation](#). You can try to use MDS and compare the final plot to the PCA plot.*



- (d) Classification using population activity: Use the top 5 principal components and repeat the analysis done in B.3 FaceDecoding (b). How does the accuracy compare?
- (e) LDA for dimensionality reduction: Linear Discriminant analysis is a common classification method, but can also be used to reduce the dimensionality of the data which has labels. For example consider face and non-face stimuli. LDA finds the axis along which the two clusters(face and non-face stimuli) are the most separated. Use LDA for dimensionality reduction utilizing the face/non-face labels. Reduce dimensionality down to 2 dimensions and plot the stimuli on a scatter plot. Compare the explained variance to PCA.

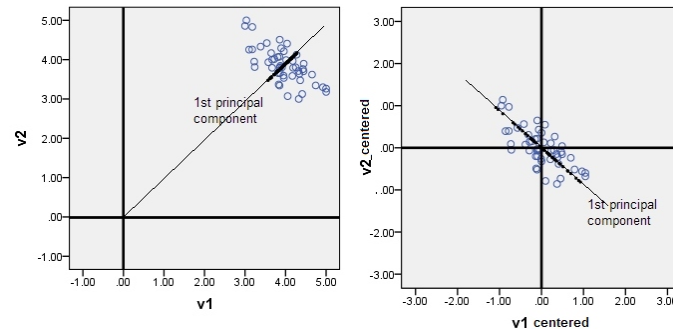


Figure 5: Necessity of having centered data. If your data doesn't have zero mean(left), the PCA is very misleading.

## A Appendix

### A.1 A Very brief intro to our neural data

We simultaneously record data from 256 electrodes in the IT, PMv and PFC regions from 2 monkeys, Didi and Coco. Electrodes 129-256 are the IT electrodes, which are of special interest to the Vision lab. Electrodes 1-128, for Didi are PMv electrodes, and for Coco are PMv+PFC.

The raw neural data recorded (BIN file) is a voltage trace over the recording duration. This data is split into spikes and LFP to further analyze it; Fig6 has some more details. We mostly work with spikes in our lab (NEX file). Each spike is an action-potential. You will likely use Multi-unit activity (where the spikes are from a few neurons near the electrode). Spike-sorting refers to the process grouping all spikes from the same neuron together, by looking at the spike-waveform similarities. Spikes are represented as spike-times in the L2 structure. Usually we count the number of spikes in a certain time interval, to obtain the firing rate of a neuron. This is what you'll be using in your analysis, and what is commonly called the "neural response". The space occupied by the population vector of the neural responses (across different units) usually called "neural-space".

### A.2 Creating L1 and L2 structures

The L1 structure can be created using `wm_createL1str()` from `wmLib` (in `visionlablib\series4lib`). You need to pass the paths to the bhv file, ecube folder, and nex file. The bhv file contains all the behavioural data, the ecube folder contains eventcode information. The raw neural data is initially stored as a bin file. In order to create a nex file(spikes) from a bin file use `wm_getmua()`. You can also obtain MUA using OFS, a software on FGPC/OPPC.

The L2 structure is created using L2 creation codes of the task types. `wm_createL2str_fixtasks` is what is used for fixation tasks, but TSD, cor-tasks and nat-tasks use their own functions.

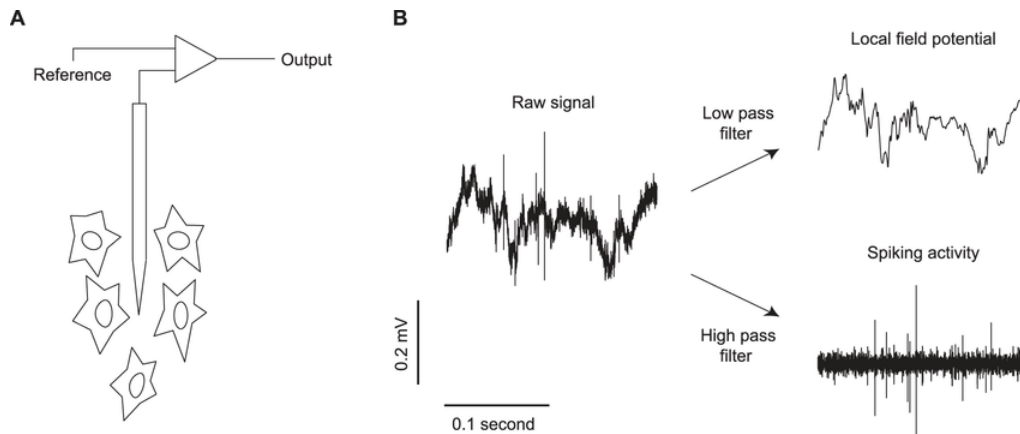


Figure 6: Decomposition of Neural data into spikes (high frequency component) and LFP(low frequency component).

## Creating the structure with LFP

For Nat-tasks, LFP might be especially quite useful. To create an L1 str with LFP, simply pass the `binFileFullPath`. This can either be the original binfile, or for larger recordings, a binfile of reduced size using `wm_createlfpbin` or `decimate`. But if you plan to use LFP, remember that LFP is often analyzed in the frequency domain (in different frequency bands like Gamma, Alpha)

### A.3 TCN Assignment-0

Note: Theoretical and Computational Neuroscience is a course that Arun jointly offers. The course has an assignment-0 for MATLAB practise. If you are new to MATLAB, and feel like you need more MATLAB practise before starting this worksheet, do solve some of these questions as well.

1. Write a function in MATLAB to calculate the factorial of a number.
2. Write a function 'myfunc' which takes an array as input and returns another array containing the maximum value, minimum value, mean, median and variance of the input array. Write another program to generate 100 different arrays of random numbers which calls 'myfunc'. The output of this program should be a 100x5 matrix which has all the 5 parameters for each of the 100 arrays of random numbers.
3. Generate 5 arrays of lengths 10, 100, 1000, 5000 and 10000 respectively where the entries of the arrays are randomly drawn from a normal distribution with mean 2 and standard deviation 8. Plot the histograms of all the 5 arrays in a single figure using matlab function subplot. Now calculate the mean and standard deviations of each array. Which of these is more accurate?
4. Create two matrices A and B such that  $A*B$  and  $A.*B$  are feasible operations. Do both kinds of multiplications and figure out what is the difference between them.
5. Generate a 200x150 grayscale image with pixel values randomly drawn from (a) Uniform distribution (b) Gaussian distribution. Display the image and the plot the histograms of the pixel values and see if they are indeed from the correct distributions. Plot the histograms again by varying the number of bins from 1 to 1000.
6. Learn how to create and manipulate other data structures like cell-arrays, strings, structures and tables. Write a few lines of code creating and manipulating these structures if you don't have experience.

7. Find out about following functions in matlab using matlab help and write few lines of code showing what these functions do. sort, sum, num2cell, save , load, clear, bar, stem, scatter, intersect, setdiff, ismember, unique, switch, hold on, find, legend, ones , zeros, transpose.