Assignment 1: Neuroscience of Decision Making PSY 307 (Monsoon 2023)

Name: Lakshay Bhushan
Roll Number: 2021397

Instructions: Please write your own responses and do not copy or lift text/code from any source (including the paper). If you are referring to credible external sources other than the attached paper for your answers, please cite those sources (within the body of text and the provide a reference list at the end) in the APA citation format (https://www.mendeley.com/guides/apa-citation-guide). Word limits given are indicative and less than the indicated numbers may also be used.

Please download this MS word question-cum-response template to TYPE your answers and feel free to add sheets as required. Convert this document to a PDF and rename the file: name_RollNo. before submitting. Please note that answers in this template only will be evaluated and hand-written or scanned answer sheets will not be evaluated.

[Strict deadline for submission: 18 September, Monday, 9:00 AM]

Q1) Fill out the google form: https://forms.gle/k93XPaUspgxg4NN6A

Q2) An experimenter recorded a SINGLE cortical neuron's activity from the part of the brain processing visual information as the organism viewed a visual stimuli on the screen. The collected dataset is attached herewith: 'Data1_NDM.mat'. It is a Three-dimensional MATLAB array. Note the following.

- **Dimension 1 = Stimulus orientations (45 degree to 202.5 degree with increments in step size of 22.5 degree visual angle).**
- **Dimension 2 = Time (sampling frequency of the neuronal activity = 1000 Hz); Total neuronal recording duration = 3.5 seconds; Time = 0 (stimulus not moving);Time = 500 (stimulus moving) ; Time = 2500 (stimulus off);**
- **Dimension 3 = Trials or stimulus repetitions**
- **Value 1 = action potential (spike) fired by the neuron; Value 0 = no action potential fired by the neuron**

Now solve the following. Insert a figure (wherever required) and paste the MATLAB/Python/R code for the same. Any figure must provide all information necessary to interpret it including axes labels, captions/legends (see Fig.1 of the attached paper for a sample; simple figure titles as captions are not enough).

[links about the 3-dimensional array in MATLAB and importing MATLAB data arrays into Python and R

- **https://in.mathworks.com/help/matlab/math/multidimensional-arrays.html**
- **https://in.mathworks.com/help/matlab/matlab_external/matlab-arrays-as-python-variables.html**
- **https://stackoverflow.com/questions/11671883/importing-an-array-from-matlab-into-r]**

```
# Importing the required libraries
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
```

A) Create a Raster plot of the neuron for ALL EIGHT orientations of the stimulus and mark the onset of stimulus movement and offset of the stimulus by a vertical green and red line respectively on the same individual subplots. Mark the spikes (action potentials) with solid black circles. [5 marks]

Hint: Create a larger figure with eight subplots (positioned as 4 rows x 2 columns); Indicate the stimulus orientation on top of each subplot as subplot title

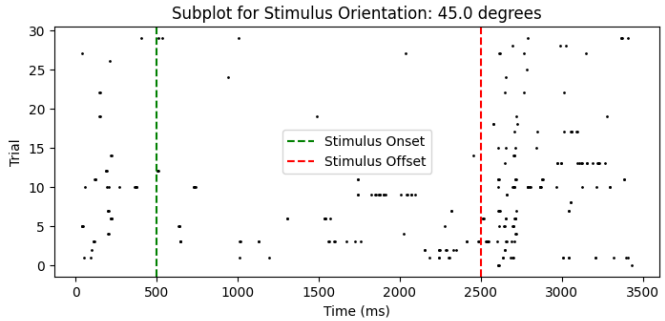
[Solution]

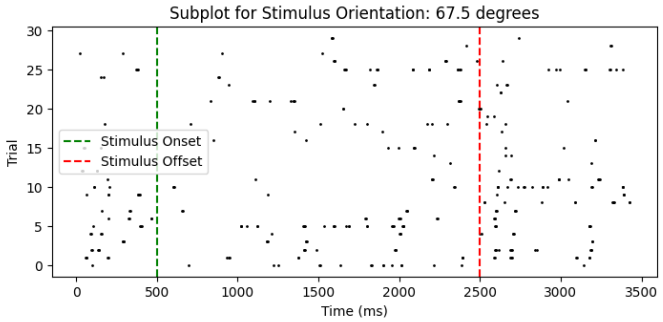
```
# Loading the MATLAB dataset `Data1_NDM.mat` into a variable called
`neuron_data`. (Naor, 2009)
neuron_data = sio.loadmat('Data1_NDM.mat')['Data1_NDM']

# Now we are calculating dimensions of our Neuron data using `shape`
function from `numpy` library and storing them in variables.
num_orientations = neuron_data.shape[0]
num_time_points = neuron_data.shape[1]
num_trials = neuron_data.shape[2]

# Creating a time vector that represents the time points at which the
neuron's data is recorded. It will be used to plot the spikes of the
neurons over time.
time = np.arange(0, num_time_points) (“Numpy.Arange() in Python,” 2017)

# Specifiying the time at which we want to start & end observation in
milliseconds.
stimulus_onset_time = 500
stimulus_offset_time = 2500

# Here the whole figure is represented by `_` as it's not going to be
used.`subplot_array` is an array of all the subplots in the figure with a
shape of 4 rows and 2 columns and each subplot is of size 15 by 15 inches.
_, subplot_array = plt.subplots(4, 2, figsize=(15, 15))
_.tight_layout()
```

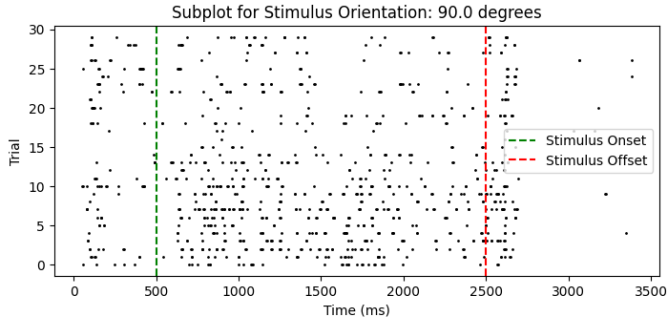
```
# Plotting the spikes of neurons over time for each orientation. Here
`orientation_index` is iterating for each orientation.
for orientation_index in range(num_orientations):

    # Calculating row and column index for each subplot
    row,col = divmod(orientation_index, 2) (Python Divmod() (with
Examples), n.d.)

    # Extracting a subset of `neuron_data` for a specific orientation
    orientation_data = neuron_data[orientation_index, :, :]

    for trial in range(num_trials):

        # Extracting the spike times for each trial
        spike_times = np.where(orientation_data[:, trial] == 1)[0]
(Numpy.Where – NumPy v1.26 Manual, n.d.)

        # Plotting the spikes of neurons over time for each trial
        main_plot = subplot_array[row, col].plot(time[spike_times], trial
* np.ones_like(spike_times), 'o', markersize=1, color='black')

    # As we only have the strimulus orientation ranging from 45 to 202.5
degrees, we are calculating the orientation value for each orientation
index.
    orientation_value = 45 + orientation_index * 22.5

    # Setting the x and y markers denoting the stimulus onset and offset
time for each subplot.
    subplot_array[row, col].axvline(x=stimulus_onset_time, color='green',
linestyle='--', label='Stimulus Onset')
    subplot_array[row, col].axvline(x=stimulus_offset_time, color='red',
linestyle='--', label='Stimulus Offset')

    # Setting the subplot title to indicate stimulus orientation for each
subplot.
    subplot_array[row, col].set_title(f'Subplot for Stimulus Orientation:
{orientation_value} degrees')
```

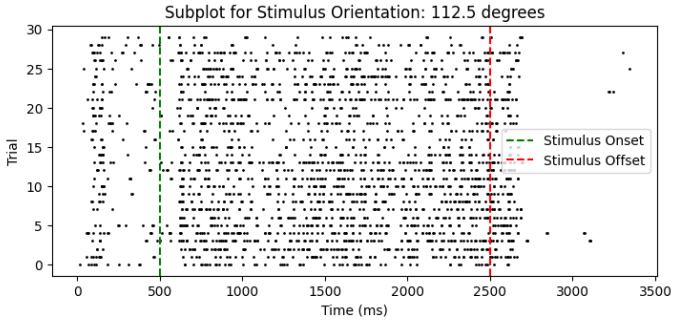
```
    # Setting x and y axis labels & legend for each subplot.
    subplot_array[row, col].set_xlabel('Time (ms)')
    subplot_array[row, col].set_ylabel('Trial')
    subplot_array[row, col].legend()

# Adjusting spacing between subplots
plt.subplots_adjust(hspace=0.50, wspace=0.25)

# Displaying the plots
plt.show()

--- END OF CODE FOR QUESTION 2 (A) ---
```

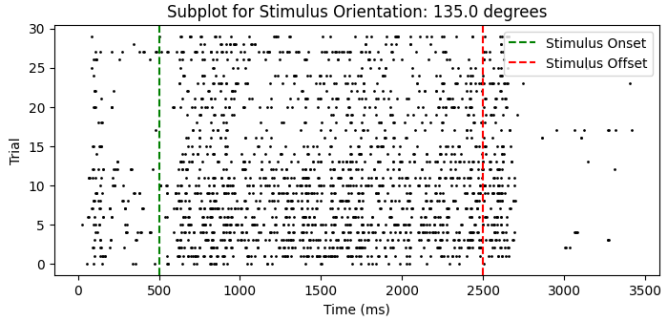
PLOT FOR QUESTION 2 (A)

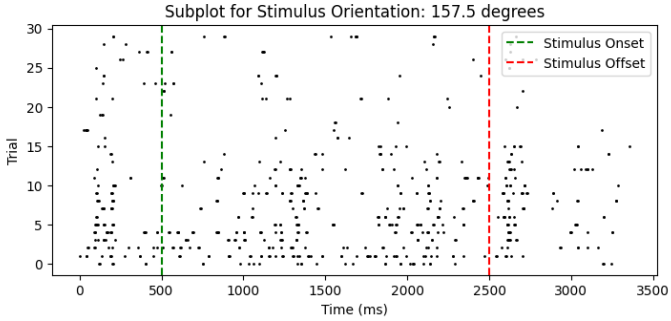


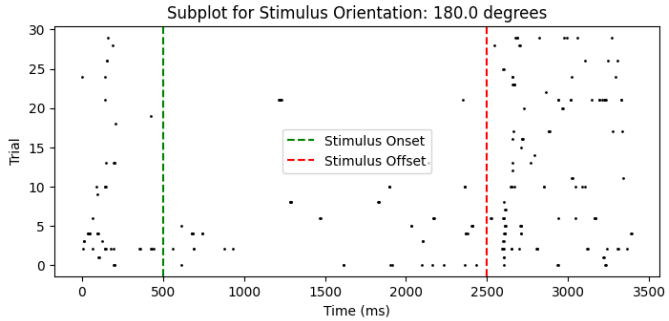


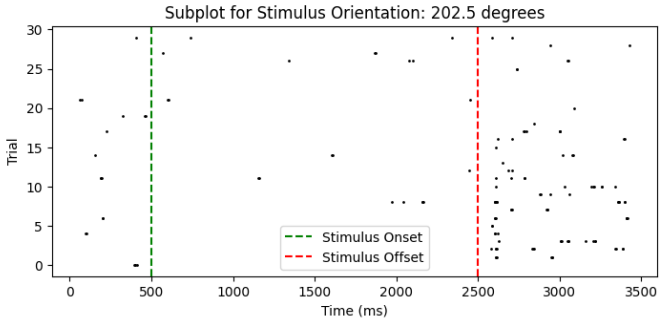












Observation: We can see a cluster (maximum number of neurons) for orientation at 112.5 and it is little less in 135 degrees which implies preferred orientation for our cortical neuron is 112.5 degrees.

B) Create a Peri Stimulus Time Plot of the neuron for ALL EIGHT orientations of the stimulus and mark the onset of stimulus movement and offset of the stimulus by a vertical green and red line respectively on the same individual subplots. Before computing the histogram, smooth the data for each subplot over a time window of 61 ms. [5 marks]

Hint: Create a larger figure with eight subplots (positioned as 4 rows x 2 columns); Smooth the data by moving average method; A line plot would suffice and depict the trend; Indicate the stimulus orientation on top of each subplot as subplot title

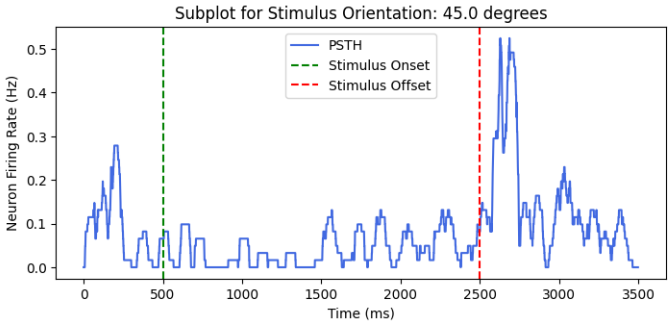
[Solution]

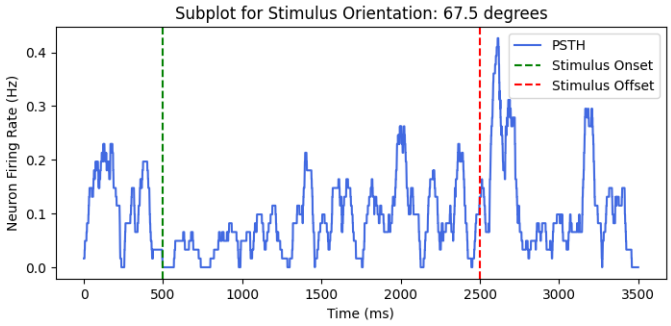
```
# Loading the MATLAB dataset `Data1_NDM.mat` into a variable called
`neuron_data`. (Naor, 2009)
neuron_data = sio.loadmat('Data1_NDM.mat')['Data1_NDM']

# Now we are calculating dimensions of our Neuron data using `shape`
function from `numpy` library and storing them in variables.
num_orientations = neuron_data.shape[0]
num_time_points = neuron_data.shape[1]
num_trials = neuron_data.shape[2]

# Creating a time vector that represents the time points at which the
neuron's data is recorded. It will be used to plot the spikes of the
neurons over time.
time = np.arange(0, num_time_points) (“Numpy.Arange() in Python,” 2017)

# Specifiying the time at which we want to start & end observation in
milliseconds.
stimulus_onset_time = 500
stimulus_offset_time = 2500

# Defining a smoothing window in milliseconds.
smoothing_window = 61

# Here the whole figure is represented by `_` as it's not going to be
used.`subplot_array` is an array of all the subplots in the figure with a
shape of 4 rows and 2 columns and each subplot is of size 15 by 15 inches.
_, subplot_array = plt.subplots(4, 2, figsize=(15, 15))
_.tight_layout()
```

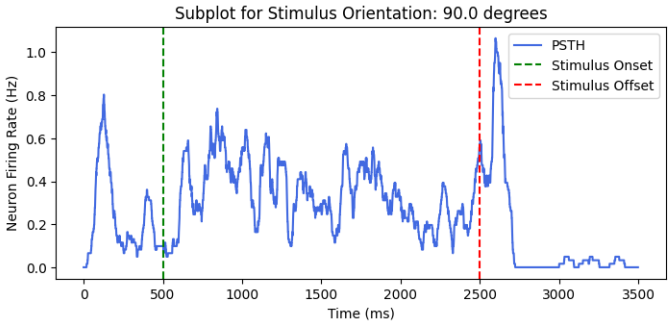
```
# Plotting the spikes of neurons over time for each orientation. Here
`orientation_index` is iterating for each orientation.
for orientation_index in range(num_orientations):

    # Calculating row and column index for each subplot
    row,col = divmod(orientation_index, 2) (Python Divmod() (with
Examples), n.d.)

    # Extracting a subset of `neuron_data` for a specific orientation
    orientation_data = neuron_data[orientation_index, :, :]

    # Smoothing the data using a moving average method and convolving it
with `smoothing_window`.
    smoothed_data = np.convolve(np.sum(orientation_data, axis=1),
np.ones(smoothing_window), 'same') / smoothing_window (Sondi, 2022; yatu,
2019)

    # Plotting the Peri-Stimulus Time Histogram (PSTH) for each
orientation using the smoothed data.
    subplot_array[row, col].plot(time, smoothed_data, color='royalblue',
label='PSTH')

    # As we only have the strimulus orientation ranging from 45 to 202.5
degrees, we are calculating the orientation value for each orientation
index.
    orientation_value = 45 + orientation_index * 22.5

    # Setting the x and y markers denoting the stimulus onset and offset
time for each subplot.
    subplot_array[row, col].axvline(x=stimulus_onset_time, color='green',
linestyle='--', label='Stimulus Onset')
    subplot_array[row, col].axvline(x=stimulus_offset_time, color='red',
linestyle='--', label='Stimulus Offset')

    # Setting the subplot title to indicate stimulus orientation for each
subplot.
```

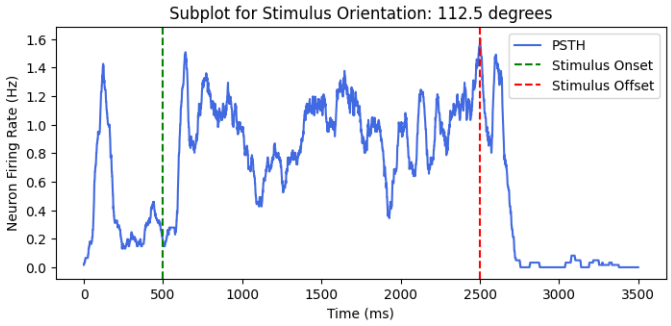
```
    subplot_array[row, col].set_title(f'Subplot for Stimulus Orientation:
{orientation_value} degrees')

    # Setting x and y axis labels & legend for each subplot.
    subplot_array[row, col].set_xlabel('Time (ms)')
    subplot_array[row, col].set_ylabel('Neuron Firing Rate (Hz)')
    subplot_array[row, col].legend()

# Adjusting spacing between subplots
plt.subplots_adjust(hspace=0.50, wspace=0.25)

# Displaying the plots
plt.show()

--- END OF CODE FOR QUESTION 2 (B) ---
```

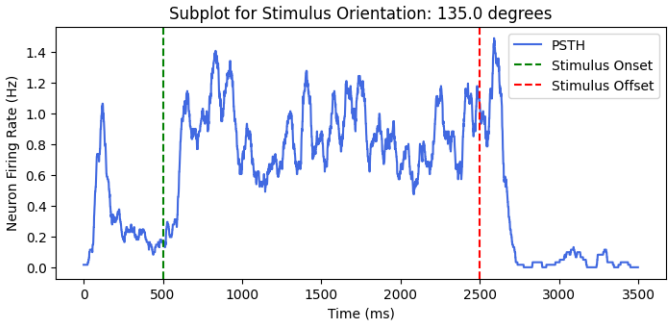
PLOT FOR QUESTION 2 (B)

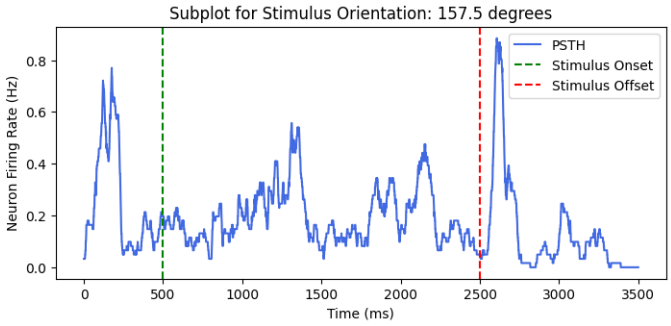


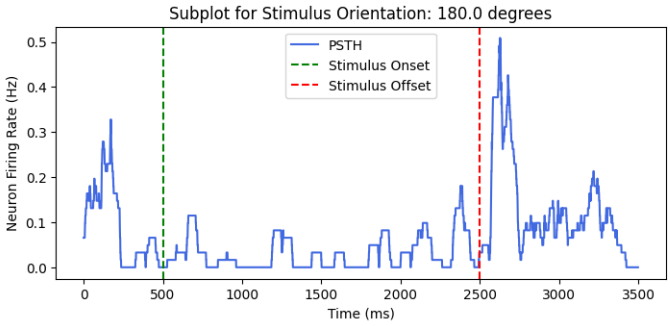


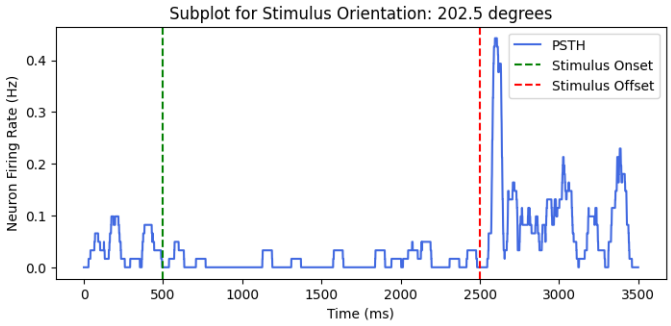












Observation: Neurons at orientations 180 and 202.5 degrees have low neuron firing rate between 500ms to 2500ms which implies low action potential.

C) Create a figure representing the Tuning Curve of the neuron from the average firing rate of the neuron (between 600 – 2500 ms and all trials) for each orientation. Computationally calculate the 'preferred orientation' of the neuron. Report the same on the title of the plot and mark it on the plot. [5+5 marks]

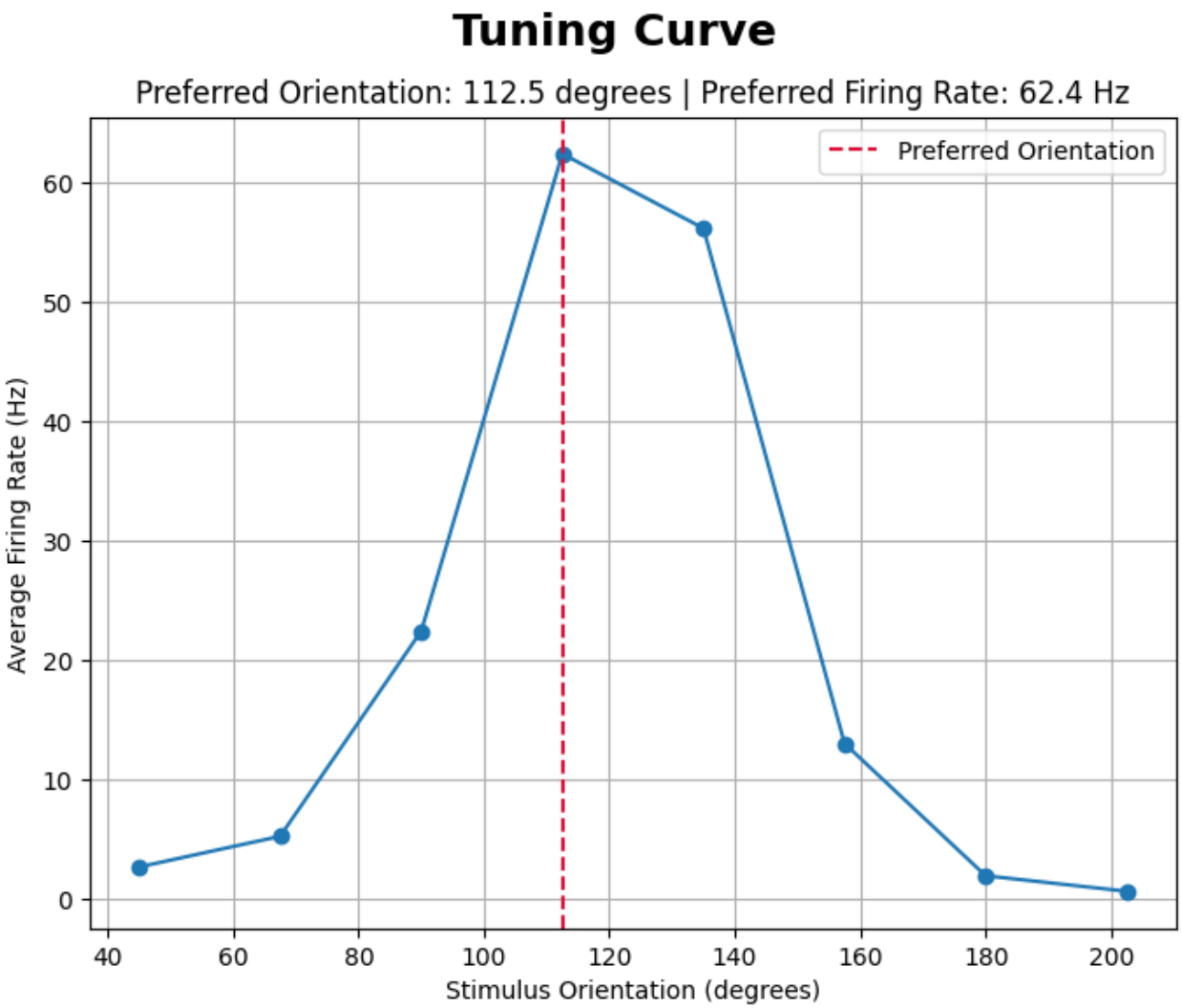
[Solution]

```
# Loading the MATLAB dataset `Data1_NDM.mat` into a variable called
`neuron_data`. (Naor, 2009)
neuron_data = sio.loadmat('Data1_NDM.mat')['Data1_NDM']

# Now we are calculating dimensions of our Neuron data using `shape`
function from `numpy` library and storing them in variables.
num_orientations = neuron_data.shape[0]
num_time_points = neuron_data.shape[1]
num_trials = neuron_data.shape[2]

# Creating a time vector that represents the time points at which the
neuron's data is recorded. It will be used to plot the spikes of the
neurons over time.
time = np.arange(0, num_time_points) ("Numpy.Arange() in Python," 2017)

# Specifiying the time at which we want to start & end observation in
milliseconds.
stimulus_onset_time = 500
stimulus_offset_time = 2500

# Creating an array of zeros to store the firing rates of each neuron for
each orientation.
firing_rate_array = np.zeros(num_orientations)

# Creating a list of all the orientations ranging from 45 to 202.5
degrees.
orientations = np.arange(45, 203, 22.5) ("Numpy.Arange() in Python," 2017)

for orientation_index in range(num_orientations):

    # Selecting neuronal data for the current orientation
    orientation_data = neuron_data[orientation_index, :, :]
```

```
     # Returing a boolean array with `True` values for time points between
stimulus onset and offset time.
    time_window_bool = (time >= stimulus_onset_time) & (time <=
stimulus_offset_time)

    # Calculating the average firing rate for this orientation
    firing_rate = np.mean(np.sum(orientation_data[time_window_bool, :],
axis=0))

    # Storing the firing rate
    firing_rate_array[orientation_index] = firing_rate

# Calculating the preferred orientation by finding the index of the
maximum firing rate along the orientation axis.
preferred_orientation = orientations[np.argmax(firing_rate_array)] (Malli,
2022)

# Calculating the preferred firing rate by finding the maximum firing rate
along the orientation axis.
preferred_firing_rate = firing_rate_array[np.argmax(firing_rate_array)]
(Malli, 2022)

# Plotting the tuning curve
plt.figure(figsize=(8, 6))
plt.plot(orientations, firing_rate_array, marker='o', linestyle='-')
plt.title(f'Preferred Orientation: {preferred_orientation} degrees |
Preferred Firing Rate: {preferred_firing_rate} Hz')
plt.suptitle(f'Tuning Curve', fontsize=18 , fontweight='semibold')
plt.xlabel('Stimulus Orientation (degrees)')
plt.ylabel('Average Firing Rate (Hz)')
plt.grid(True)

# Marking the preferred orientation with a vertical dashed line
plt.axvline(x=preferred_orientation, color='crimson', linestyle='--',
label='Preferred Orientation')
plt.legend()
```

```
# Displaying the tuning curve plot
plt.show()

# Printing the preferred orientation of the neuron in degrees.
print(f"The preferred orientation of the neuron is {preferred_orientation}
degrees.")

# Printing the average firing rate of the neuron at the preferred
orientation in Hz.
print(f"The average firing rate at the preferred orientation is
{preferred_firing_rate} Hz.")

--- END OF CODE FOR QUESTION 2 (C) ---
```

PLOT FOR QUESTION 2 (C)



Terminal Output:

```
The preferred orientation of the neuron is 112.5 degrees.
The average firing rate at the preferred orientation is 62.4 Hz.
```

Bibliography / References

Malli. (2022, August 7). How to Use NumPy argmax in Python. *Spark By {Examples}*. https://sparkbyexamples.com/numpy/how-to-use-numpy-argmax-in-python/

Naor, G. (2009, May 17). *Answer to "Read .mat files in Python."* Stack Overflow. https://stackoverflow.com/a/874488

Numpy.arange() in Python. (2017, July 15). *GeeksforGeeks*. https://www.geeksforgeeks.org/numpy-arrange-in-python/

numpy.where—NumPy v1.26 Manual. (n.d.). Retrieved September 18, 2023, from https://numpy.org/doc/stable/reference/generated/numpy.where.html

Python divmod() (with Examples). (n.d.). Retrieved September 18, 2023, from https://www.programiz.com/python-programming/methods/built-in/divmod

Sondi, J. (2022, November 23). *What is the numpy.convolve() Method in Python?* Scaler Topics. https://www.scaler.com/topics/numpy-convolve/

yatu. (2019, February 11). *Answer to "How to calculate rolling / moving average using python + NumPy / SciPy?"* Stack Overflow. https://stackoverflow.com/a/54628145