EE 243 HW #2

QUESTION 1

(a) False. When an action potential is fired, there is first a large current attributable to $Na^+$ channels opening up before $K^+$ channels.

(b) False. During an action potential, $Na^+$ currents serve to depolarize the cell and $K^+$ currents serve to hyperpolarize the cell.

(c) True

(d) False. It is possible to record action potentials with a voltage clamp.

(e) False. The poisson process is memoryless

(f) False. If the Fano factor of a neuron is greater than 1, the firing rate variance is greater than the firing rate mean

(g) True

(h) False. An exponential interspike interval distribution does not model the refactory period well.

(i) False. Measurement of action potentials at microsecond resolution

(j) True

(k) True.

(l) True

(m) False. Convolving spike trains w/ a Gaussian kernel to approximate a spike rate $r(t)$ is a type of low-pass filtering

(n) True

(o) False. During the relative refractory period, it is possible for a spike to be generated

QUESTION 2 $f(\theta) = c_0 + c_1 \cos(\theta - \theta_0)$

a. Show $\theta_0$ is preferred direction

Solution. Cos of $\emptyset$ is 1 and cosine values can only be between $\emptyset$ and 1. Assuming that $c_0$ and $c_1$ are positive, the max value for $f(\theta) = c_0 + c_1$. To get $c_1(1)$, $\cos(\theta - \theta_0)$ should equal 1 meaning that $\theta - \theta_0 = 0$ leading to $\theta = \theta_0$. This is why $\theta_0$ is the preferred direction.

b. $f(\theta) = -11 + 8\cos(\theta - 125)$

I would tell him "You've made a mistake". This is because $f(\theta)$ returns a negative value at all values of $\theta$ but we want distributions with values greater than $\emptyset$.

c. $\cos(\theta - \theta_0)$ ; $e^{j\theta} = \cos(\theta) + j\sin(\theta)$

$\rightarrow \cos(\theta - \theta_0) + j\sin(\theta - \theta_0) = e^{j(\theta - \theta_0)}$

$\rightarrow e^{j\theta - j\theta_0} = e^{j\theta} e^{-j\theta_0} = [\cos(\theta) + j\sin(\theta)][\cos(\theta_0) - j\sin(\theta_0)]$

$= \cos\theta\cos\theta_0 - j\cos\theta\sin\theta_0 + j\sin\theta\cos\theta_0 + \sin\theta\sin\theta_0$

$\cos(\theta - \theta_0) + j\sin(\theta - \theta_0) = \cos\theta\cos\theta_0 + \sin\theta\sin\theta_0 + j(\sin\theta\cos\theta_0 - \cos\theta\sin\theta_0)$

so, $\cos(\theta - \theta_0) = \cos\theta\cos\theta_0 + \sin\theta\sin\theta_0$

d. $f(\theta) = c_0 + c_1\cos(\theta - \theta_0)$

$f(\theta) = k_0 + k_1\sin(\theta) + k_2\cos(\theta)$

$f(\theta) = c_0 + c_1\cos(\theta)\cos(\theta_0) + c_1\sin(\theta)\sin(\theta_0)$

$k_0 = c_0$

$k_1 = c_1\sin(\theta_0)$ $\quad$ $k_2 = c_1\cos(\theta_0)$

e. $y_0 = k_0 + k_1\sin(0) + k_2\cos(0) = k_0 + k_2$

$y_{120} = k_0 + k_1\sin(120) + k_2\cos(120) = k_0 + \frac{\sqrt{3}}{2}k_1 - \frac{1}{2}k_2$

$y_{240} = k_0 + k_1\sin(240) + k_2\cos(240) = k_0 - \frac{\sqrt{3}}{2}k_1 - \frac{1}{2}k_2$

$y_{120} + y_{240} = 2k_0 - k_2$

$y_{120} + y_{240} + y_0 = 3k_0$

$$\boxed{\frac{y_{120} + y_{240} + y_0}{3} = k_0}$$

$$\boxed{k_2 = y_0 - \frac{y_{120} + y_{240} + y_0}{3}}$$

$$k_1 = \frac{2\left[y_{240} - \frac{y_{120} - y_{240} + y_0}{3} + \frac{1}{2}\left(y_0 - \frac{y_{120} - y_{240} + y_0}{3}\right)\right]}{\sqrt{3}}$$

$$Y_{240} + \frac{\sqrt{3}}{2}k_1 = k_0 - \frac{1}{2}k_2$$

$$Y_{240} + \frac{\sqrt{3}}{2}k_1 = \frac{Y_{120}+Y_{240}+Y_0}{3} - \frac{1}{2}\left[Y_0 - \frac{Y_{120}+Y_{240}+Y_0}{3}\right]$$

$$Y_{240} + \frac{\sqrt{3}}{2}k_1 = \frac{3}{2}\,\frac{Y_{120}+Y_{240}+Y_0}{3} - \frac{1}{2}Y_0$$
$$= \frac{1}{2}(Y_{120}+Y_{240}+Y_0) - \frac{1}{2}Y_0$$
$$= \frac{1}{2}Y_{120} + \frac{1}{2}Y_{240} \; ; \; \frac{\sqrt{3}}{2}k_1 = \frac{1}{2}Y_{120} - \frac{1}{2}Y_{240}$$
$$\sqrt{3}k_1 = Y_{120} - Y_{240}$$

$$k_1 = \boxed{\frac{Y_{120} - Y_{240}}{\sqrt{3}}}$$

fig: Jupyter

QUESTION 3

a. The exponential distribution does not incorporate the refractory period. This is because real data has few or no short ISI's due to the refactory period and the firing rates vary over time.

b.

$$\int_0^{0.001} \lambda e^{-\lambda t} = \int_0^{0.001} 50 e^{-50t}\,dt$$

$$= +\frac{50}{50} e^{-50t}\Big|_0^{0.001} = e^{-50t}\Big|_0^{0.001} = 0.0487$$

$$= \boxed{4.87\%}$$

$\lambda = \frac{60 \text{ spikes}}{\text{second}}$ ; $ISI = \frac{1}{60}$

QUESTION 4 Poisson process with rate $\lambda$

a. Mean of ISI $= \frac{1}{\lambda}$

b. $P(T > \frac{1}{\lambda}) = e^{-\lambda t}$ where $t = \frac{1}{\lambda}$
$$= e^{-\lambda \frac{1}{\lambda}}$$
$$= e^{-1}$$
$$= \frac{1}{e}$$

$P(T=t) = \lambda e^{-\lambda t}$

$T=t) = \lambda e^{-\lambda t}$

c. $E[T|T>\frac{1}{\lambda}] = \int_0^\infty t\, P(T=t\,|\,T>\frac{1}{\lambda})\, dt$

step function w/ val 1 @ $T>\frac{1}{\lambda}$

$$P(T=t\,|\,T>\tfrac{1}{\lambda}) = \frac{P(T=t)\,P(T>\frac{1}{\lambda}\,|\,T=t)}{P(T>\frac{1}{\lambda})} = \frac{P(T=t)}{P(T>\frac{1}{\lambda})} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda\frac{1}{\lambda}}} = \frac{\lambda e^{-\lambda t}}{e^{-1}}$$

$$P(T=t\,|\,T>\tfrac{1}{\lambda}) = \lambda e^{1-\lambda t}$$

$$\int_{\frac{1}{\lambda}}^\infty t\lambda e^{1-\lambda t}\, dt = \lambda e \int_{\frac{1}{\lambda}}^\infty t e^{-\lambda t}\, dt = \lambda e \cdot \frac{2}{e\lambda^2} = \boxed{\frac{2}{\lambda}}$$

d. $E[T|T<\frac{1}{\lambda}] = \int_0^\infty t\, P(T=t\,|\,T<\frac{1}{\lambda})$

step @ 1 from 0 to $\frac{1}{\lambda}$

$$P(T=t\,|\,T<\tfrac{1}{\lambda}) = \frac{P(T=t)\,P(T<\frac{1}{\lambda}\,|\,T=t)}{P(T<\frac{1}{\lambda})} = \frac{\lambda e^{-\lambda t}}{1-\frac{1}{e}}$$

$$\frac{\lambda}{1-e^{-1}}\int_0^{\frac{1}{\lambda}} t e^{-\lambda t}\, dt = \frac{e-2}{e\lambda^2}\cdot\frac{\lambda}{1-\frac{1}{e}} = \frac{e-2}{e\lambda(1-e^{-1})} = \frac{e-2}{e\lambda-\lambda} = \frac{e-2}{(e-1)}\cdot\frac{1}{\lambda}$$

e. $P(T>\frac{1}{\lambda}) = \frac{1}{e}$ $\quad P(T<\frac{1}{\lambda}) = 1-\frac{1}{e}$

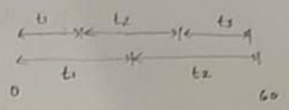
Expected # of spikes = $\sum_{n=1}^{\infty} n\left(\frac{1}{e}\right)\left(1-\frac{1}{e}\right)^{n-1}$

This is a geometric series that converges to $e$.

$\therefore$ Expected # of spikes = $e$

f. $e-1$ spikes with $T<\frac{1}{\lambda}$ and 1 spike at $T>\frac{1}{\lambda}$

$$\text{Waiting time} = (e-1)E[T|T<\tfrac{1}{\lambda}] + E[T|T>\tfrac{1}{\lambda}]$$
$$= (e-1)\left(\frac{1}{\lambda}\right)\left(\frac{e-2}{e-1}\right) + \frac{2}{\lambda}$$
$$= \frac{e-2}{\lambda} + \frac{2}{\lambda} = \boxed{\frac{e}{\lambda}}$$




(5) Electrode 1 - ISI of 20 ms

Electrod 2 - ISI of 30 ms

a. No neurons are detected in first 60 ms?

$N(s) = 1$ @ time > 60 ms for both electrodes

0 spikes in 60 ms $= e^{-\lambda(60)}$

$= e^{-(\frac{1}{20})(60)} = e^{-3}$ for 20 ms

$= e^{-(\frac{1}{30})(60)} = e^{-2}$ for 30 ms

P(0 spikes in 60 from both) $= e^{-5}$ = 0.67%

b. $Pr(T \geq t+s \mid T \geq s) = Pr(T \geq t) = e^{-\lambda t}$ ← this we know.

Now, accounting for $\lambda_1$ & $\lambda_2$: $\boxed{e^{-(\lambda_1+\lambda_2)t}}$

c. Probability of first spike at time t being from e1

1 spike in t ms: $e^{-(\frac{1}{20})t \cdot (\frac{1}{20})t}$

$e^{-(\frac{1}{30})t (\frac{1}{30})t}$

$N(t) = N_1(t) + N_2(t)$

$\int_0^\infty P(N_1(t) = 1, N_2(t) = 0)$ $P(N_1(t) = 1, N_2(t) = 0)$

$Pr(N_1(t) = 1, N_2(t) = 0 \mid N(t) = 1)$

Poisson

$= \int_0^\infty \lambda_1 e^{-(\lambda_1+\lambda_2)t} \, dt$

$= \int_0^\infty (0.05) \, e^{-(0.0833)t} \, dt$

$= 0.60 = \boxed{\frac{3}{5}}$

# hw2

April 28, 2018

## 1 HW2 Coding

This workbook will walk you through the plotting problem 2(f) in HW2. It will also provide familiarity with Jupyter Notebook and Python. Please print (to pdf) a completed version of this workbook for submission with HW2.

ECE C143A/C243A, Spring Quarter 2018, Prof. J.C. Kao, TAs T. Monsoor, X. Jiang and X. Yang

### 1.1 Import library

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
```

### 1.2 Define the function

The function below, `ptc()` accepts the average firing rates at three targets and learns the parameters $c_0, c_1$, and $\theta$ of the tuning curve. Please implement this function below. You may evaluate your execution by running the code under section "Plot the figure," which calculates the tuning parameters using your implementation of `ptc()`. The output should appear reasonable.

```
In [57]: def ptc(y0 , y1 , y2):
             #PTC calculates the tuning curve given average firing rates for certain directions.

             # ================================================================ #
             # YOUR CODE HERE:
             #  The function takes three inputs corresponding to the average
             #  firing rate of a neuron during a reach to 0 degrees (y0), 120
             #  degrees (y1) and 240 degrees (y2). The outputs, c0, c1, and
             #  theta0 are the parameters of the tuning curve.
             # ================================================================ #
             k0 = (y1+y2+y0)/3
             k1 = (y1-y2)/(np.sqrt(3))
             k2 = y0 - ((y1+y2+y0)/3)
             c0 = k0
             theta0 = np.arctan(k1/k2)
             c1 = k1/ np.sin(theta0)
             theta0 = theta0*180/(np.pi)
```

```
        # ================================================================= #
        # END YOUR CODE HERE
        # ================================================================= #

        return c0,c1,theta0
```

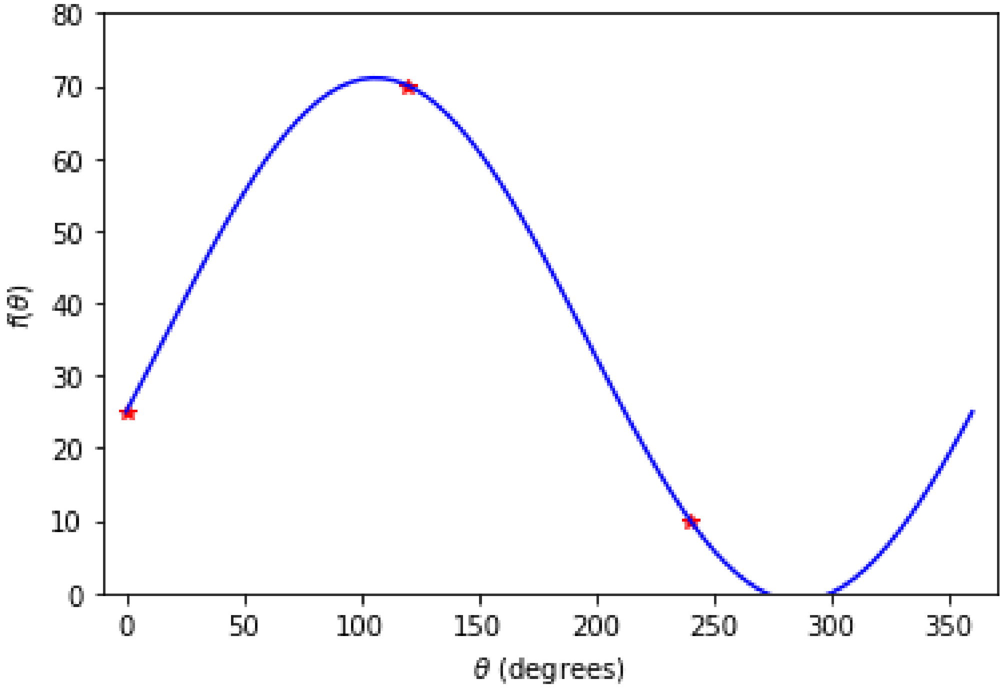
### 1.3 Plot the figure

The following cells execute your PTC function, printing out the values and plotting the tuning curve.

```
In [58]: c0,c1,theta0=ptc(25,70,10)
         print('c0 = ', c0)
         print('c1 = ', c1)
         print('theta0 = ', theta0)
```

```
c0 =  35.0
c1 =  -36.0555127546
theta0 =  -73.897886248
```

```
In [59]: theta = np.linspace(0, 2*np.pi, num=80)
         plt.plot([0,120,240],[25,70,10],'r*',10)
         plt.plot(theta * 180 / np.pi,c0 + c1 *np.cos(theta - theta0 * np.pi/180),'b',2)
         plt.xlim ([-10 ,370])
         plt.ylim ([0,80])
         plt.xlabel(r'$\theta$ (degrees)');
         plt.ylabel(r'$f(\theta)$');
         plt.show()
```




```
In [25]: a = np.array([[1, np.sin(0), np.cos(0)],
                      [1, np.sin((60*np.pi)/180), np.cos((60*np.pi)/180)],
                      [1, np.sin((120*np.pi)/180),np.cos((120*np.pi)/180)],
                      [1, np.sin((np.pi)), np.cos((np.pi))],
                      [1, np.sin((240*np.pi)/180), np.cos((240*np.pi)/180)],
                      [1, np.sin((300*np.pi)/180), np.cos((300*np.pi)/180)]
                     ])
         b = np.array([25, 40, 70, 30, 10, 15])
         k0,k1,k2 = np.linalg.lstsq(a,b)[0]
         c0 = k0
         theta0 = np.arctan(k1/k2)
         c1 = k1/ np.sin(theta0)
         theta0 = theta0*180/(np.pi)
         print('c0 = ',c0)
         print('c1 = ',c1)
         print('theta0 = ',theta0)
```

```
c0 =  31.6666666667
c1 =  -25.2212432507
theta0 =  -76.6271741919
```

```
In [17]: theta = np.linspace(0, 2*np.pi, num=80)
         plt.plot([0,60,120,180,240,300],[25, 40, 70, 30, 10, 15],'r*',10)
```

```
plt.plot(theta * 180 / np.pi,c0 + c1 *np.cos(theta - theta0 * np.pi/180),'b',2)
plt.xlim ([-10 ,370])
plt.ylim ([0,80])
plt.xlabel(r'$\theta$ (degrees)');
plt.ylabel(r'$f(\theta)$');
plt.show()
```