Assignment 1
Neuroscience of Learning, Memory, Cognition
Dr.Aghajan

Mohammad Mowlavi

February 29, 2024

Contents

Chapter 1

Assignment 1

1.1 Part 1

1.1.1 Fourier Transform

Fourier Transform represents a time-domain or space-domain signal as a sum of sinusoidal functions with different frequencies, amplitudes, and phases. Actually, it converts a signal from the time or space domain to the frequency domain. There are two types of Fourier Transforms: 1. The standard continuous Fourier Transform is defined as an integral and 2. Discrete Fourier Transform (DFT) is used for discrete data using sigma. In figure 1.1 you can see some formulas of Fourier Transform.

But what is fft?

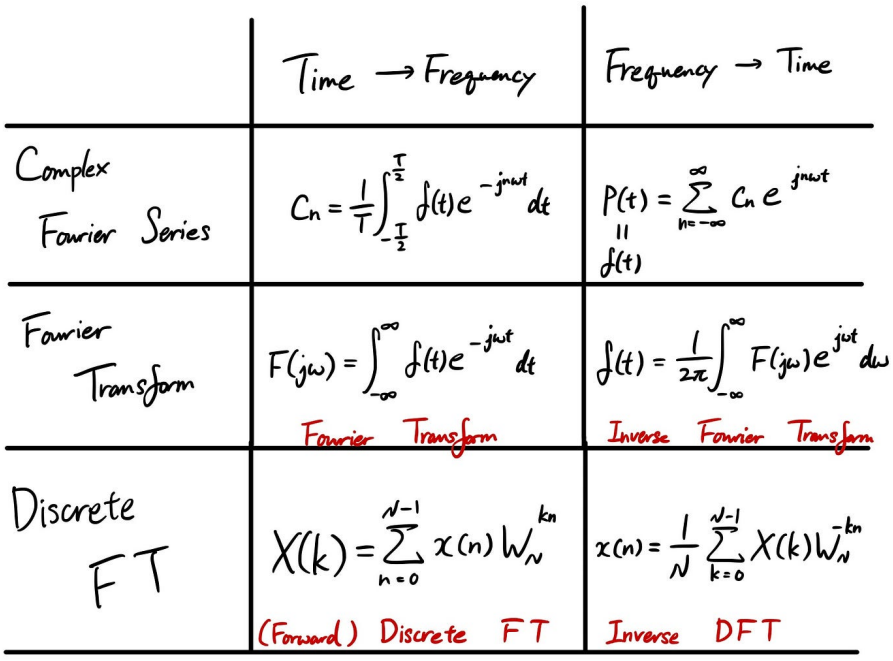

Figure 1.1: Some formulas of Fourier Transform.

Fast Fourier Transform (FFT) is an efficient algorithm for calculating the discrete Fourier Transform (DFT) of a sequence or time-domain signal. The FFT significantly reduces the computational complexity of the DFT from $O(n^2)$ to $O(n\log(n))$, making it practical for real-time and large-scale applications.

But how fft can help us in this practice?

As we want to process signals in this assignment, fft can help us in signal processing. Actually, fft is used for filtering, noise reduction, audio processing and etc. Also it enables us to to analysis and manipulate the signals with different frequencies in different domains. figure 1.2 shows the formulas of fft.

Produce a sinusoidal wave

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi nk}{N}} \quad (1)$$

$$x(n) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi kn}{N}} \quad (2)$$

Figure 1.2: Formulas of Fast Fourier Transform.

In this part I explain the results of the code. I write the code with np library and using sin function. Because after this question we add a noise with sd 1, I Set the sin amplitude to 5. As shown in figure 1.3 the signal and its FFT spectrum are calculated and plotted. The magnitude of FFT multiplied by 1000 because I set the time step 2000, and magnitude is multiplied by half of time step. **References for this question are:** Wikipedai,

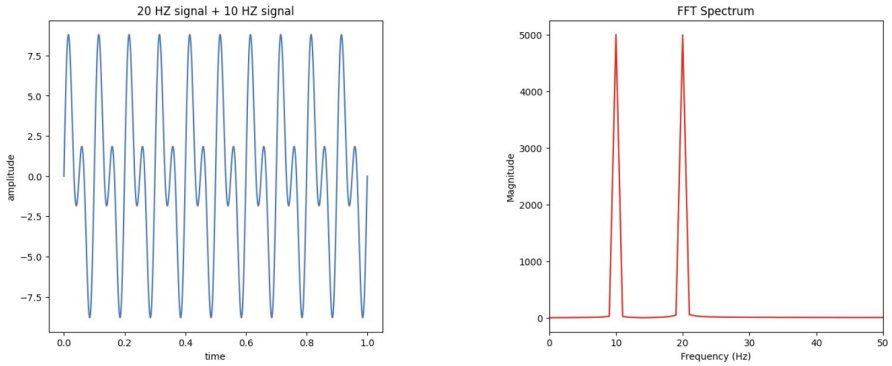

Figure 1.3: 10 HZ + 20 HZ signal.

thefouriertransform, for Fourier Transform and nti-audio, wikipedia and my friends for fft and its applications. (Before this course, I do not work with Fourier Transform and I do not know it.)

1.1.2 Gaussian white noise

Gaussian white noise is a random signal with a normal probability distribution and a flat power spectral density across all frequencies. The term *white* implies that the noise has equal power at all frequencies. The *Gaussian* property indicates that the noise samples follow a normal or Gaussian distribution. Figure 1.4 shows Gaussian distribution.

But how Gaussian white noise can help us in this practice?

As we want to process signals in this practice, Gaussian white noise can help us for to reference or benchmark signal for evaluating and testing various signal processing algorithms, such as noise reduction, filtering, and estimation techniques.

Corrupt the data with this noise and show the frequency spectrum.

In this part I corrupt the data with a Gaussian noise with mean 0 and sd 1, as you can see in figure 1.5.

References for this question are: Wikipedia, ni, etc.

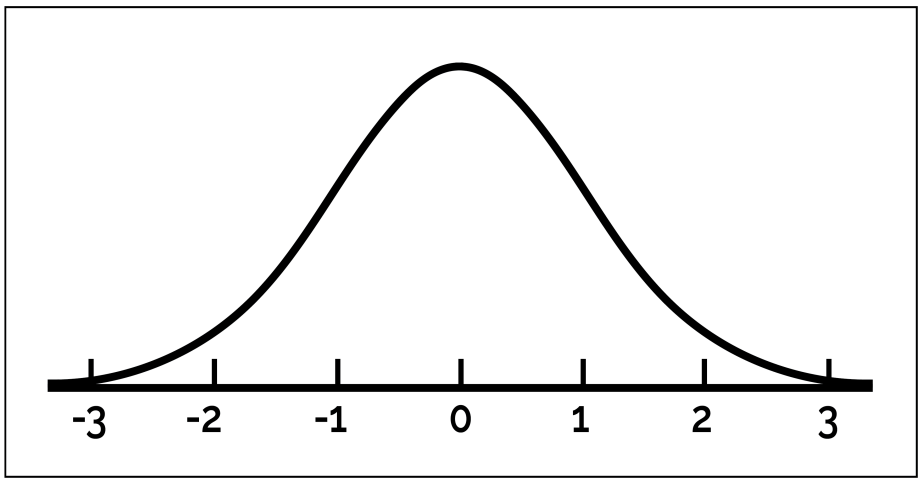

Figure 1.4: Normal distribution with mean 0 and sd 1.

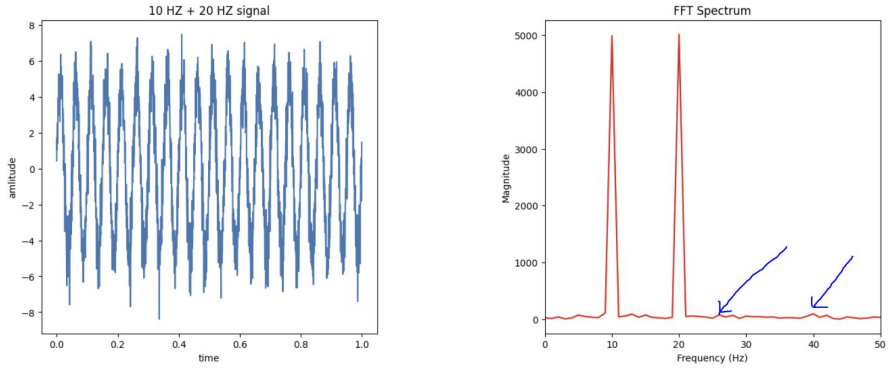

Figure 1.5: 10 HZ + 20 HZ signal with Gaussian noise with mean 0 and sd 1.

1.1.3 Bonus

What is Cross Spectrum?

Cross correlation is a complex value function which represents the correlation between two signals which have different frequencies and this function measures similarity between two signals as a function of the time lag between them. We can say Cross Spectrum is defined as the Fourier transform of the cross correlation function between the two signals. According to wikipedia, some formula are shown in figure 1.6.

As you can see in figure 1.7, cross power spectral density plot provides insights into the frequency components

Definition [edit]

Let (X_t, Y_t) represent a pair of stochastic processes that are jointly wide sense stationary with autocovariance functions γ_{xx} and γ_{yy} and cross-covariance function γ_{xy}. Then the **cross-spectrum** Γ_{xy} is defined as the Fourier transform of γ_{xy} [1]

$$\Gamma_{xy}(f) = \mathcal{F}\{\gamma_{xy}\}(f) = \sum_{\tau=-\infty}^{\infty} \gamma_{xy}(\tau)\, e^{-2\,\pi\,i\,\tau\,f},$$

where

$$\gamma_{xy}(\tau) = \mathrm{E}[(x_t - \mu_x)(y_{t+\tau} - \mu_y)]\,.$$

The cross-spectrum has representations as a decomposition into (i) its real part (co-spectrum) and (ii) its imaginary part (quadrature spectrum)

$$\Gamma_{xy}(f) = \Lambda_{xy}(f) - i\Psi_{xy}(f),$$

and (ii) in polar coordinates

$$\Gamma_{xy}(f) = A_{xy}(f)\, e^{i\phi_{xy}(f)}.$$

Here, the amplitude spectrum A_{xy} is given by

$$A_{xy}(f) = (\Lambda_{xy}(f)^2 + \Psi_{xy}(f)^2)^{\frac{1}{2}},$$

and the phase spectrum Φ_{xy} is given by

$$\begin{cases} \tan^{-1}(\Psi_{xy}(f)/\Lambda_{xy}(f)) & \text{if } \Psi_{xy}(f) \neq 0 \text{ and } \Lambda_{xy}(f) \neq 0 \\ 0 & \text{if } \Psi_{xy}(f) = 0 \text{ and } \Lambda_{xy}(f) > 0 \\ \pm\pi & \text{if } \Psi_{xy}(f) = 0 \text{ and } \Lambda_{xy}(f) < 0 \\ \pi/2 & \text{if } \Psi_{xy}(f) > 0 \text{ and } \Lambda_{xy}(f) = 0 \\ -\pi/2 & \text{if } \Psi_{xy}(f) < 0 \text{ and } \Lambda_{xy}(f) = 0 \end{cases}$$

Figure 1.6: Cross Spectrum formulas.

which are common between the two signals and higher values on the plot indicate a stronger correlation between the signals at those frequencies. Also the amplitude of each peak represents the strength of the correlation at that frequency.

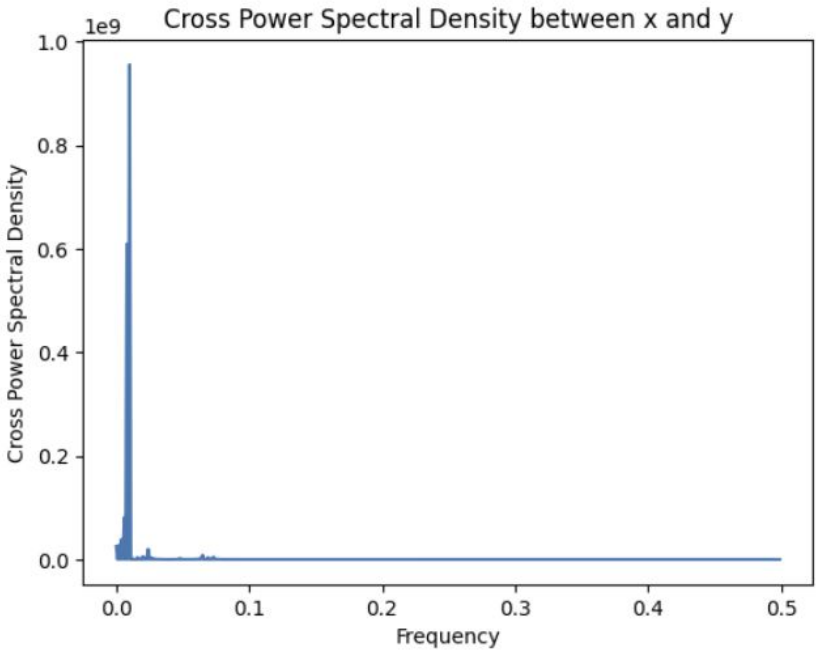

Figure 1.7: Plotting the result of cross power spectral density.

What does np.random.seed do?

Actually there are 2 types of random number generation. first approach really generates random numbers but second approach generates pseudo random numbers. Python and other programming languages generate random numbers in this way. Every pseudo random number generator uses a seed to generate first random number. Of course it may be different for various Pseudo number algorithms. For example below, seed is X_0:
$X_{i+1} = (aX_i + c)modm, i = 0, 1, 2, ...$

1.2 Part 2

1.2.1 The Leaky Integrate-and-Fire (LIF) model

What are the parameters of the the equation?
first equation is:

$$C_m \frac{dV}{dt} = -g_L(V - E_L) + I$$

In this equation, C_m is the membrane capacitance, V is the membrane potential, E_L is the resting potential, and I is the external input current and finally g_L is the leak conductance and it is actually the inverse if R_L which is the leak resistance. if we divide above equation to g_L then we have below equation:

$$\tau_m \frac{dV}{dt} = -(V - E_L) + \frac{I}{g_L}$$

In this equation we have τ_m which is the membrane time constant because dividing the capacitance to conductance resulting in time unit.
What are the results of altering them numerically?
As we said above, τ_m represents the membrane time constant, So Increasing it will slow down the rate of change of membrane potential, and it causes for lower responding time of neuron to inputs. For example in the figure 1.8 value of tau_m multiplied by 10 and rate change of membrane potential is so slower than before. $\frac{dV}{dt}$ shows the changing rate of V over time, if $\frac{dV}{dt}$ is positive shows increasing V and if negative, shows decreasing membrane potential.
$V - E_L$ shows the difference between the membrane potential and the resting potential, Actually resting potential is the membrane potential when neuron be in resting time, if $V - E_L$ be positive, then it will tend to bring membrane potential back towards the resting potential, as show in figure 1.9 which I change E_L value from -75 to -300
I represents the input current to the neuron, which can be negative or positive deponding on the neuron receiving excitatory or inhibitory inputs. Increasing this also make $\frac{dV}{dt}$ higher as you see in figure 1.10.
g_L shows the membrane conductance, which determines how easily ions can flow across the neuron's membrane. So, decreasing the g_L makes neuron more changeable to input current. For more information see figure 1.11, in this figure value of g_L is 5 and default value was 10.
What is the Euler method?
Euler method is a iterative algorithm for solving ordinary differential equations when you are given an initial condition. This method uses the idea of local linearity or linear approximation, where we use small tangent lines over a short distance to approximate the solution to an initial-value problem. Here is the formula of the Euler method:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

Describe code
i USE Euler method for calculating V over time steps in $simulate_LIF$ function. As you see, if $refactory_prioid$ is greater than 0, we should reset voltage or if V is greater than threshold voltage, we also should reset voltage and set refractory time. You can see the results of running this function in figure 1.12 with default values.

```
1 def simulate_LIF(pars, Iext=180):
2         # Initialize variables
3         V = np.zeros(len(pars['range_t']))
```

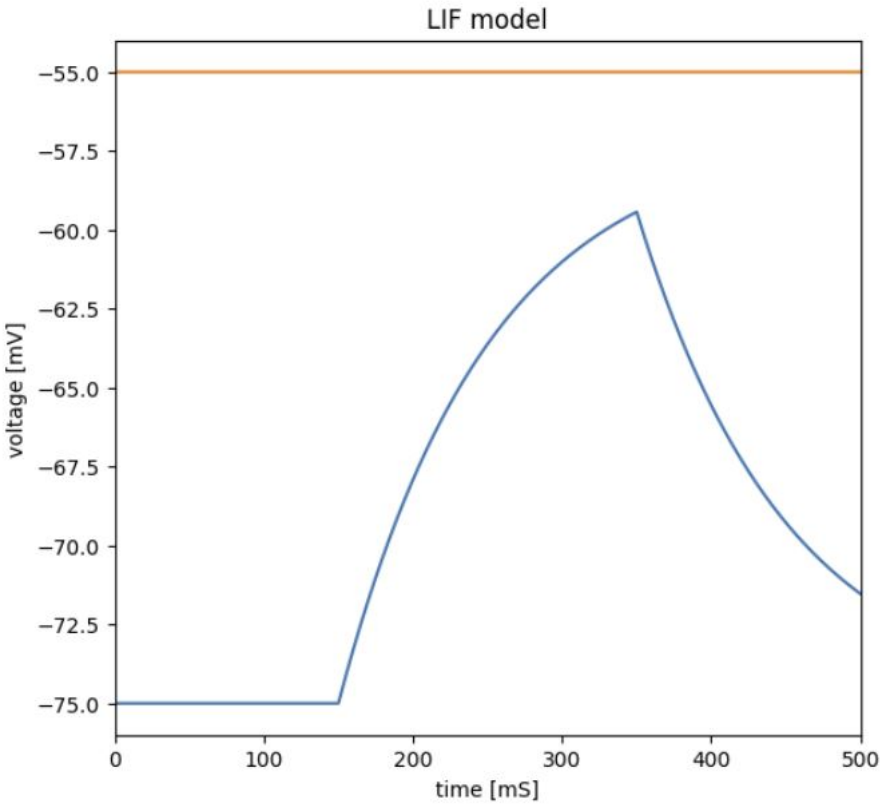

Figure 1.8: effect of increasing tau

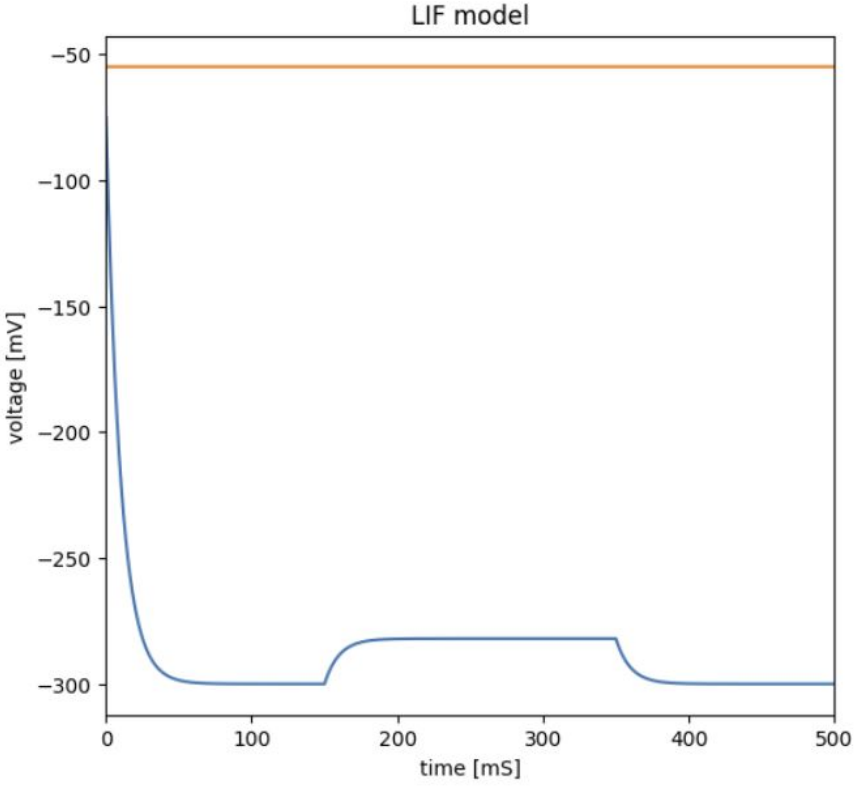

Figure 1.9: effect of changing the E_L

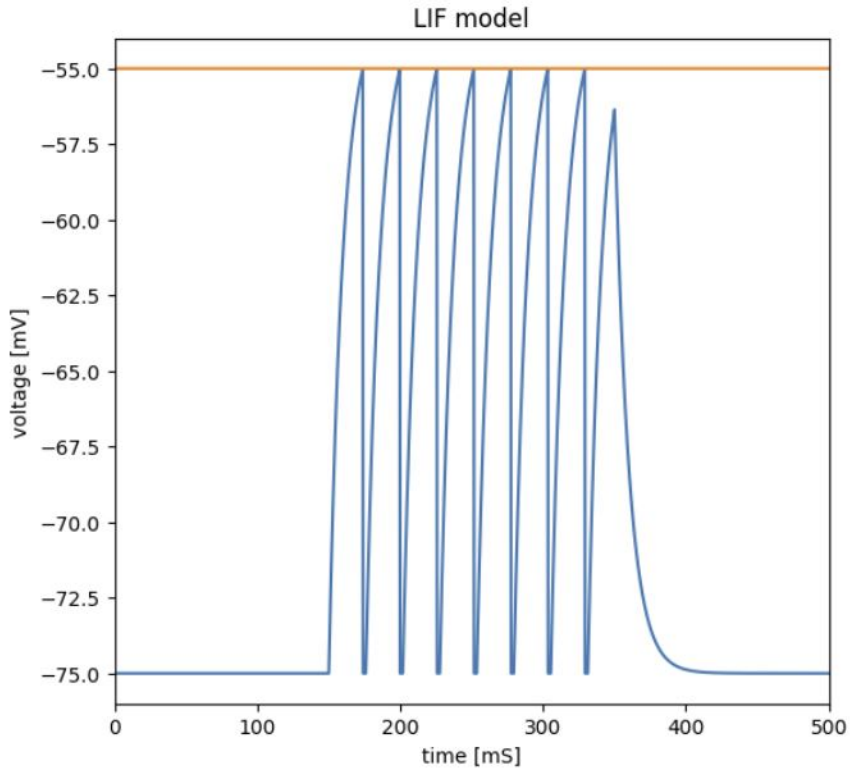

Figure 1.10: effect of changing the input current.

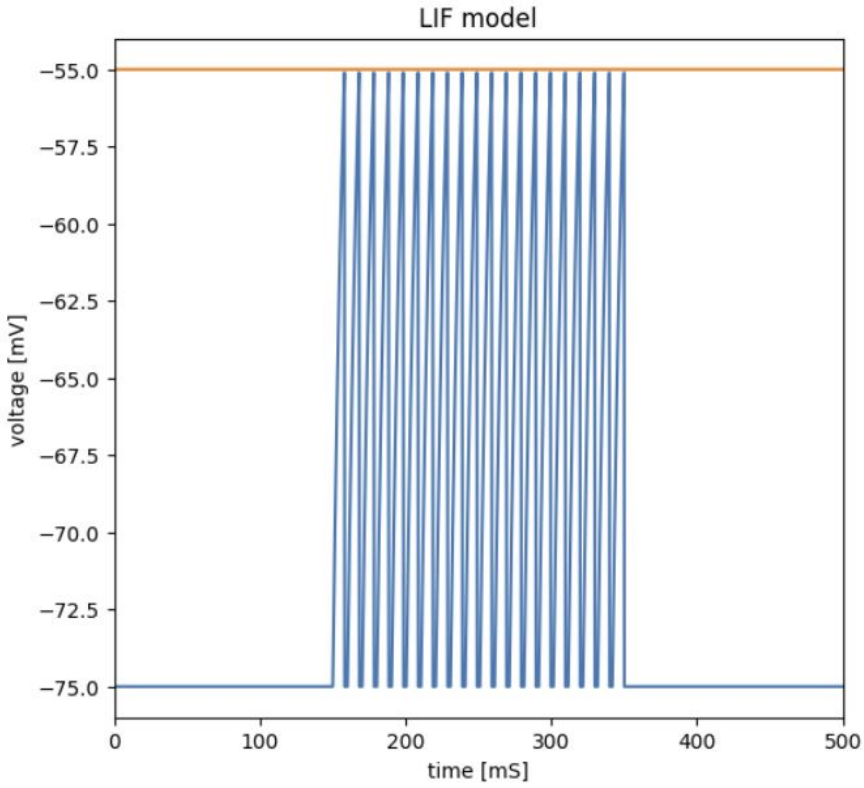

Figure 1.11: effect of changing the membrane conductance.

```
        V[0] = pars['V_init']
        Iext = Iext * np.ones(len(V))
        Iext[:int(len(Iext) / 2) - 1000] = 0
        Iext[int(len(Iext) / 2) + 1000:] = 0
        refactory_prioid = 0

        # runing itrative euler method for finding V
        for iteratation in range(len(V) - 1):
                if refactory_prioid > 0:
                        V[iteratation] = pars['V_reset']  # set voltage to reset
                        refactory_prioid = refactory_prioid - 1
                elif V[iteratation] >= pars['V_th']:
                        V[iteratation] = pars['V_reset']  # reset voltage
                        refactory_prioid = pars['tref'] / pars['dt']  # set refractory time

                # Increment potential
                dvdt = 1 / pars['tau_m'] * (pars['E_L'] - V[iteratation] + Iext[iteratation
                    ] / pars['g_L']) * pars['dt']

        # Update
                V[iteratation + 1] = V[iteratation] + dvdt

        return V, Iext
```

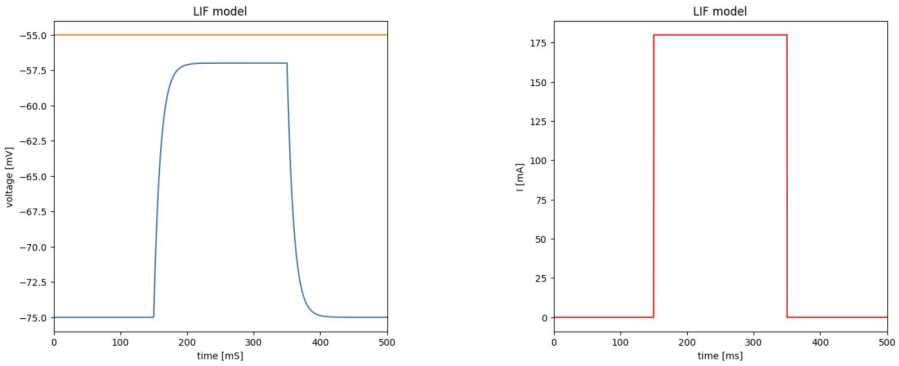

Figure 1.12: Default values of pars.

References for this question are: Wikipedia, calcworkshop and my previous knowledge about Euler method.

1.2.2 The Hodgkin-Huxley model

Describe the parameters of the Hodgkin-Huxley model in your report.
Step by step, we check each formula, here is the first formula of the Hodgkin-Huxley model:

$$C_M \frac{dV}{dt} = -g_{Na}(V - V_{Na}) - gK(V - V_K) - g_l(V - V_l) + I$$

- C_M**:** This parameter represents the membrane capacitance, the ability of membrane to store electrical charge. So, higher C_M means that membrane can store more charge and respond more slowly to input currents.
- $\frac{dV}{dt}$**:** This shows the change rate of membrane potential.
- g_{NA}**:** This shows the conductance of sodium ion channel, Sodium ions depolarize the membrane and generate action potentials. If ew increase g_{Na}, then it is more likely to reach to threshold voltage.

- V_{Na}**:** This parameter represents the equilibrium potential for sodium ions, in fact, it is the membrane potential which there is no net of flowing sodium ions.
- g_{Na}**:** Such as g_{Na} this parameter represents the conductance of K ions. Potassium ions repolarize the membrane, so with increasing this value, membrane back to its resting state faster.
- V_K**:** Such as V_{Na} this parameter shows the equilibrium of potassium ions.
- g_l**:** This parameter shows leak conductance, which accounts for the passive flow of ions through non-specific ion channels. Also it show the conductance for other ions, not specifically potassium or sodium ions. This parameter help neuron to maintain resting potential.
- V_l**:** This represents the equilibrium potential for leak conductance.
- I**:** This parameter shows the input current to neuron which can be positive to negative.

next formula is:

$$\frac{dh}{dt} = \alpha h(V)(1-h) - \beta h(V)h$$

- h**:** This term shows probability of Na ions channel being closed.
- $\frac{dh}{dt}$**:** This term represents the rate of change of the Na inactivation variable (h) over time.
- $\alpha h(V)$**:** This represents the rate at which the Na inactivation variable (h) transitions from the inactive state to the active state.
- $\beta h(V)$**:** This is the opposite of previous case, which represents the rate at which the Na inactivation variable (h) transitions from the active state to the inactive state.

next formula is:

$$\frac{dm}{dt} = \alpha m(V)(1-m) - \beta m(V)m$$

this one is such as above, we just have to change names :)

- m**:** This term shows probability of Na ions channel being opened.
- $\frac{dm}{dt}$**:** This term represents the rate of change of the Na activation variable (m) over time.
- $\alpha m(V)$**:** This represents the rate at which the Na activation variable (m) transitions from the inactive state to the active state.
- $\beta m(V)$**:** This is the opposite of previous case, which represents the rate at which the Na activation variable (m) transitions from the active state to the inactive state.

next formula is:

$$\frac{dn}{dt} = \alpha n(V)(1-n) - \beta n(V)n$$

this one also is such as above, we just have to change names :)

- n**:** This term shows probability of K ions channel being opened.
- $\frac{dn}{dt}$**:** This term represents the rate of change of the K activation variable (n) over time.
- $\alpha n(V)$**:** This represents the rate at which the K activation variable (m) transitions from the inactive state to the active state.
- $\beta n(V)$**:** This is the opposite of previous case, which represents the rate at which the K activation variable (n) transitions from the active state to the inactive state.

What are the dynamics of the Hodgkin-Huxley model equations?

First of all let's see what is the code.

```
### START CODE HERE ###
I0 = 0 # low input current
t = np.arange(0, T) * dt
V = np.zeros([T, 1])
m = np.zeros([T, 1])
h = np.zeros([T, 1])
n = np.zeros([T, 1])

V[0] = -75.0 # mV
m[0] = 0.1 # near to real prob
h[0] = 0.5 # near to real prob
n[0] = 0.3 # near to real prob

for i in range(0,T-1):

        V[i + 1] = V[i] + dt * (-1 * gNa0 * m[i] ** 3 * h[i] * (V[i] + 65 - ENa) - gK0 *
                                n[i] **4 * (V[i] + 65 - EK) - gL0 * (V[i] + 65 - EL) +
            I0)
        m[i + 1] = m[i] + dt * (alphaM(V[i]) * (1 - m[i]) - betaM(V[i]) * m[i])
        h[i + 1] = h[i] + dt * (alphaH(V[i]) * (1 - h[i]) - betaH(V[i]) * h[i])
        n[i + 1] = n[i] + dt * (alphaN(V[i]) * (1 - n[i]) - betaN(V[i]) * n[i])

### END CODE HERE ###
```

I choose I_0 0 to check the functionality of parameters when there is no current. Then I initialize the t, V, m, h, n variables and after that, I set V[0], m[0], h[0], n[0] to values near its resting value.

As we can see in the figure 1.13, because there is no input current, we do not see big change in membrane

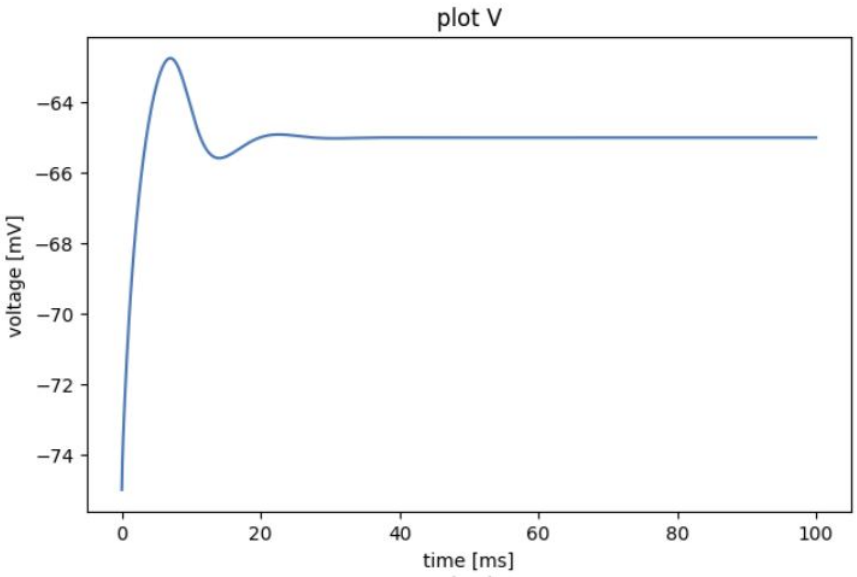

Figure 1.13: changing V over time.

potential. This little change is made by membrane leakage current and leakage conductance the potential of membrane. After some time, the potential will become stable and fixed and the final value after 100 ms is about -65 mV.

Behavior of n , m and h shows in figures 1.16, 1.14 and 1.15 As we know, the h, n and m variable are dependent to membrane potential. Based on figure 1.13 which shows the dynamic of membrane potential, potential being fixed after few moments and because of that, n, m and h being fixed and also there is no reason that sodium ions enter the neuron via their channels and also there is no reason that the potassium ions exit the

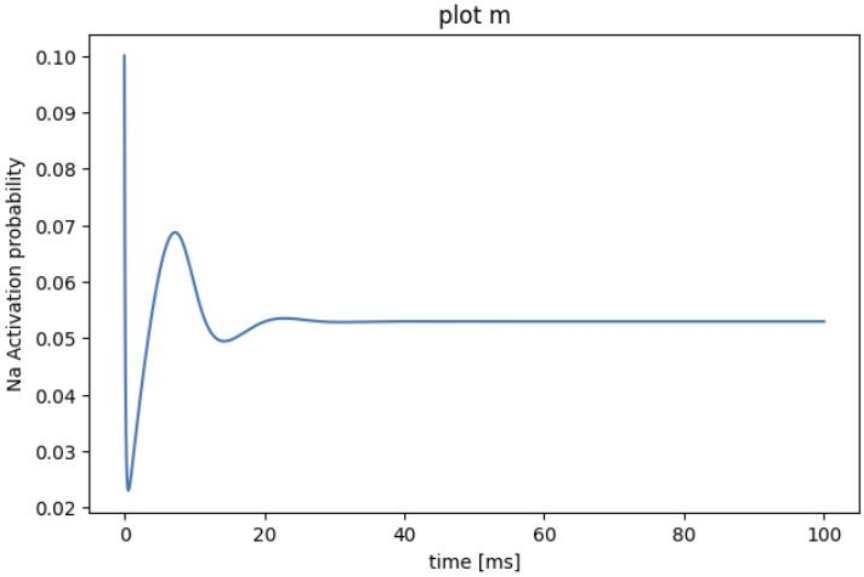

Figure 1.14: changing m over time.

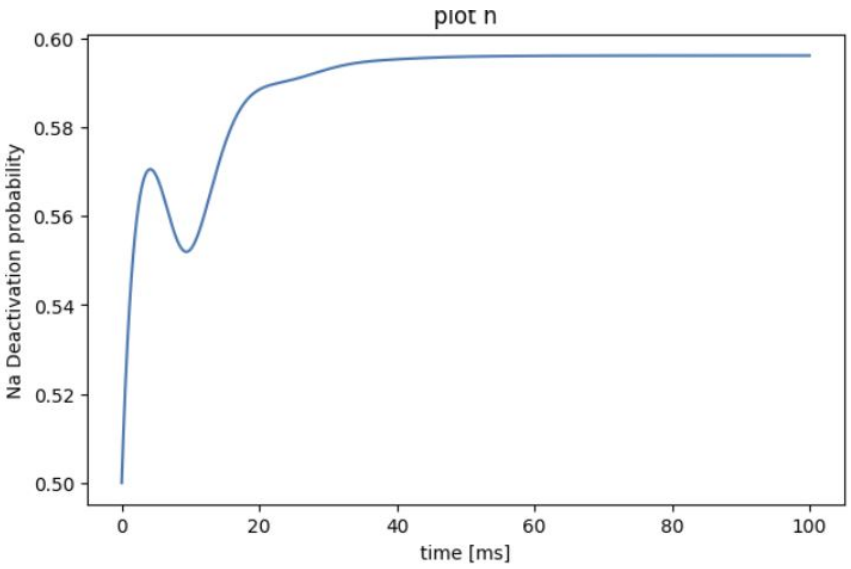

Figure 1.15: changing h over time.

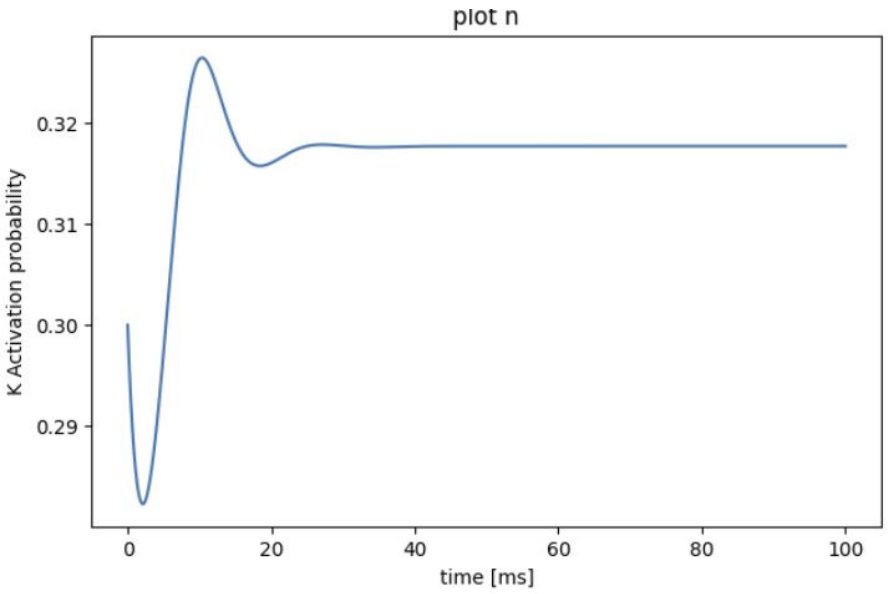

Figure 1.16: changing n over time.

neuron so we don't expect any reasonable change in probability of being open for gating variables. The steady state values of n, m, h are such as below:

$$m_\infty(v) = \frac{\alpha_m(v)}{\alpha_m(v) + \beta_m(v)}$$

$$h_\infty(v) = \frac{\alpha_h(v)}{\alpha_h(v) + \beta_h(v)}$$

$$n_\infty(v) = \frac{\alpha_n(v)}{\alpha_n(v) + \beta_n(v)}$$