بنام خدا

گزارش تمرین ۱

آرشام لؤلؤهری

۹۹۱۰۲۱۵۶

- **Working with arrays using PyLab**
    - **i.** Pylab is a module that provides a Matlab like namespace by importing functions from the modules Numpy and Matplotlib. Numpy provides efficient numerical vector calculations based on underlying Fortran and C binary libraries. Matplotlib contains functions to create visualizations of data.
    - **ii.** numpy.multiply performs element –wise multiplication for equal-size arrays or matrix multiplication for matrices (same as '*' operation). Unlike multiply, the '*' operation does not work for non-numpy arrays.

- **Coding Exercise 1: Python code to simulate the LIF neuron**

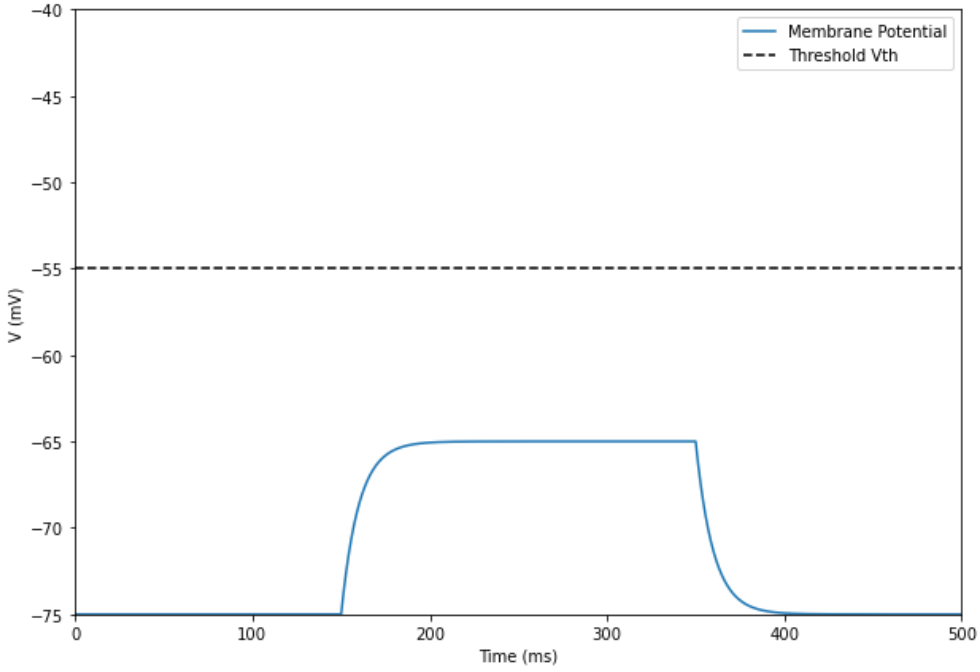
    - ✓ **Default_pars function description:**

      In this function, we are going to set some fixed parameters for our future model and functions. Parameters will get some default values unless we set a specific value when calling the function (This is done in the 'for' loop inside the function). In the end, we set the parameter 'range_t', which is a time vector in our simulation. This array in initiated using total simulation interval (T) and time steps (dt).

    - ✓ **Run_LIF function description:**

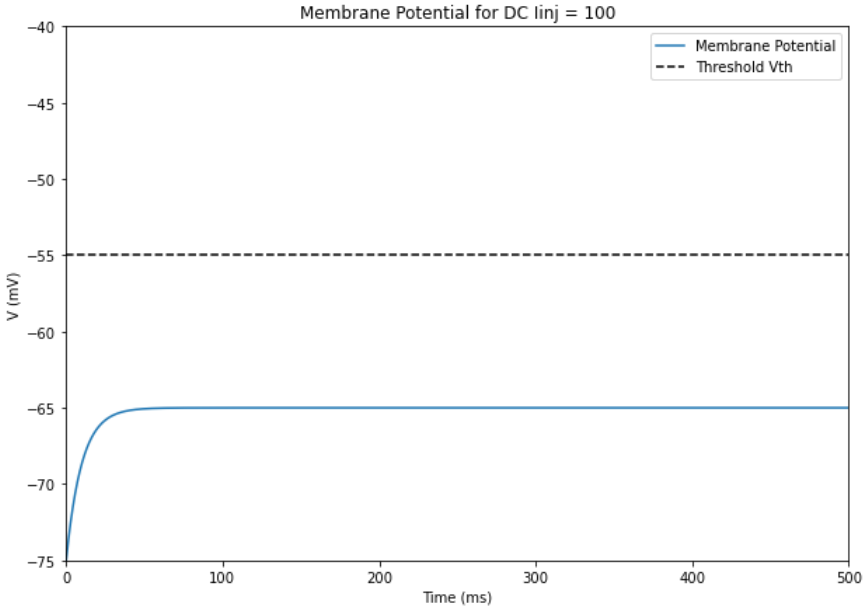
      First, at "Set Parameters" section, we initialize required variables using the parameter "pars".
      For voltage initialization, we set zeros for all indices in v, except the first one, which gets the initial value V_init.

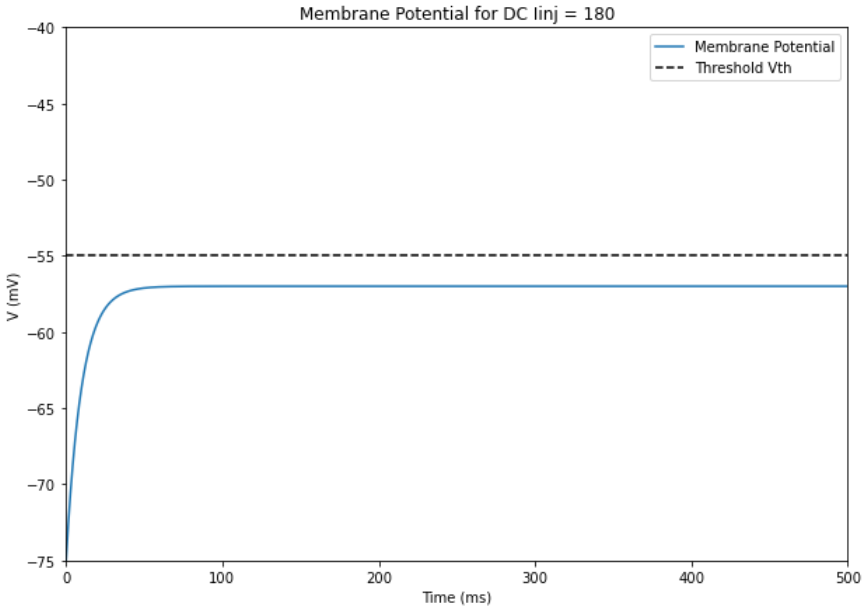
Now we initialize Iint. If the input Iint is either a vector, no changes needed. But if it's an integer, we consider a DC current with this value, flowing all the time.
If the input "stop" is true, we'll set a current pulse such that the Iint becomes zero for all samples, except 2000 middle samples.
In the for loop, we create the vector v sample by sample. When we have a non-zero value for tr (which acts like a refractory timer, starts when voltage reaches threshold), the voltage should reset (the timer tr also counts another sample). Else, we'll check the voltage whether exceeds Vth. In this case, the refractory timer tr gets the non-zero value, the voltage resets and the current spike time is saved in "rec_spikes".
Finally, it's time to calculate dv using equation (2) in the exercise. For each time sample, we need to add the previous v with this dv and put it in the next sample of v.
In the next cell, we are going to run this function, and then visualize the voltage v. we create the figure. Plot v against time vector, and a horizontal line on the threshold voltage. The resulting figure is as follows:

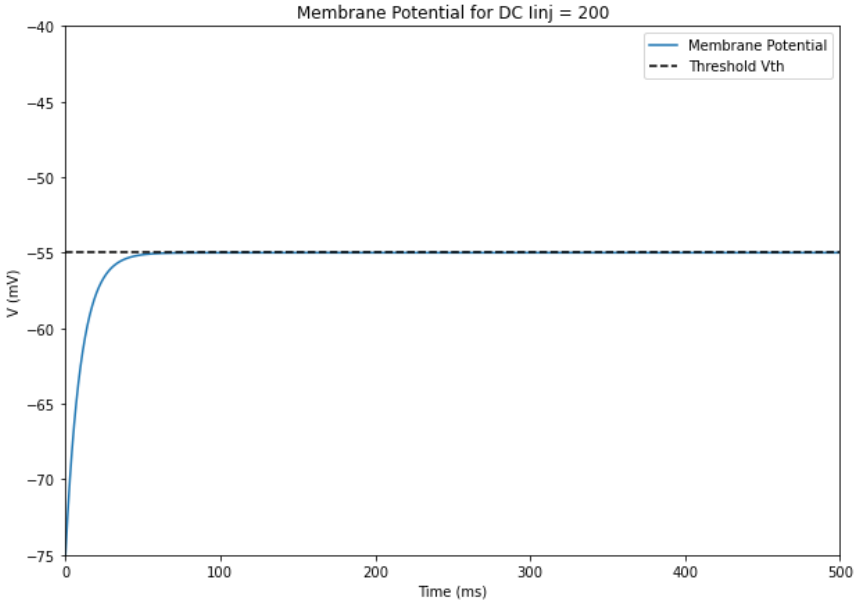


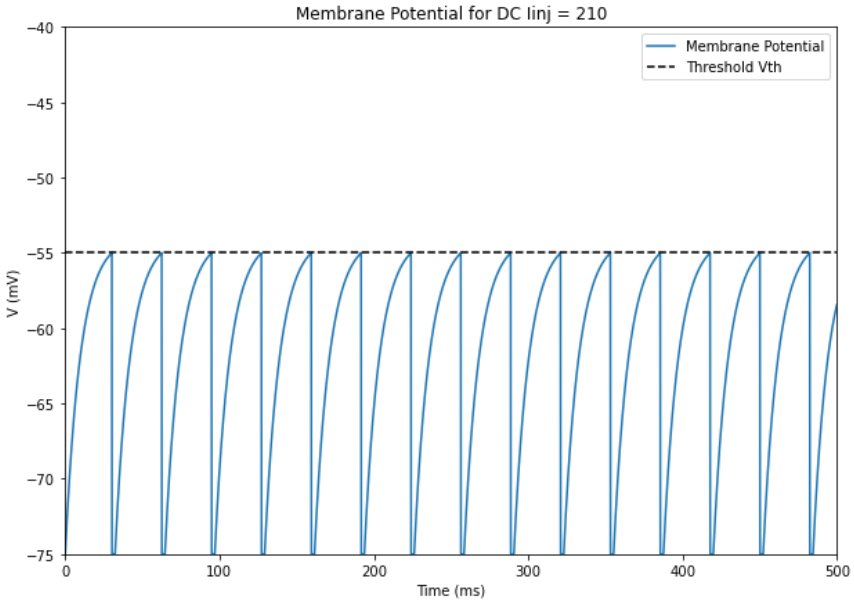
- **Response of an LIF model to different types of input currents**
    I. DC Current:

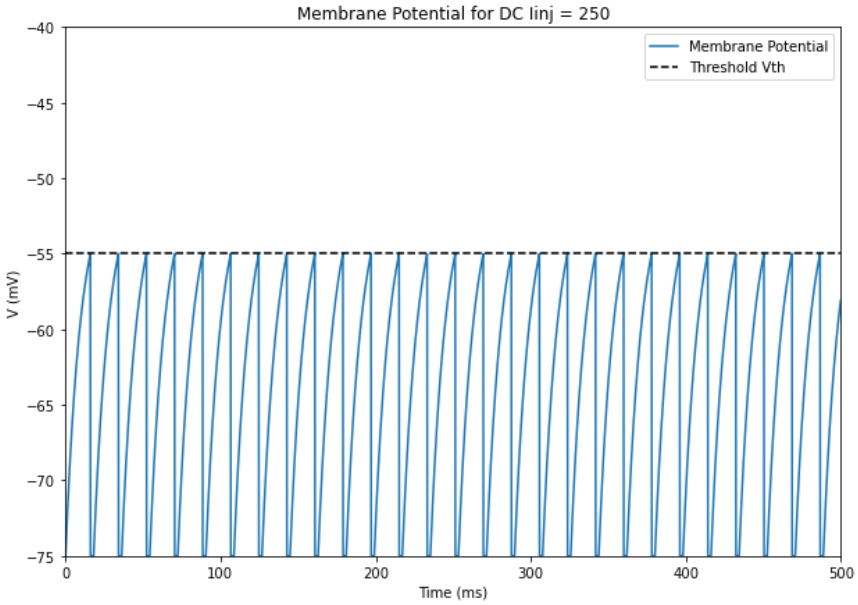
To make the current DC, we first change the “stop” value to “false”. The rest of visualization code is the same. The value of Iinj is set by variable ‘i’ and we plot the results for different values of i:







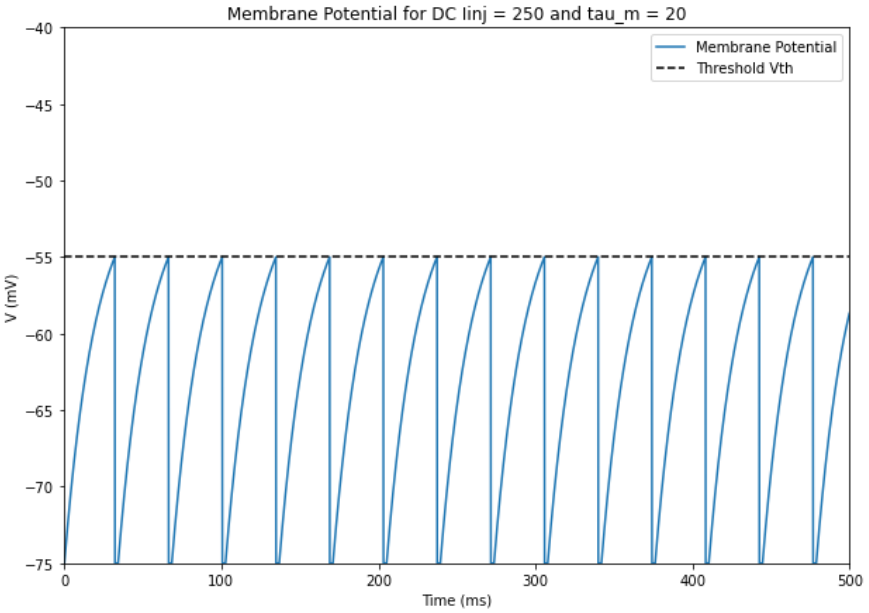




According to the plots, it is clear that DC currents with amplitude greater than 200 pA will cause spikes.

According to equation (2), increasing Iint increases dV/dt. Hence, the frequency of spikes increases too.

This time, we increase Membrane time constant from 10 to 20. With the same 250pA current, the following figure would be the result:



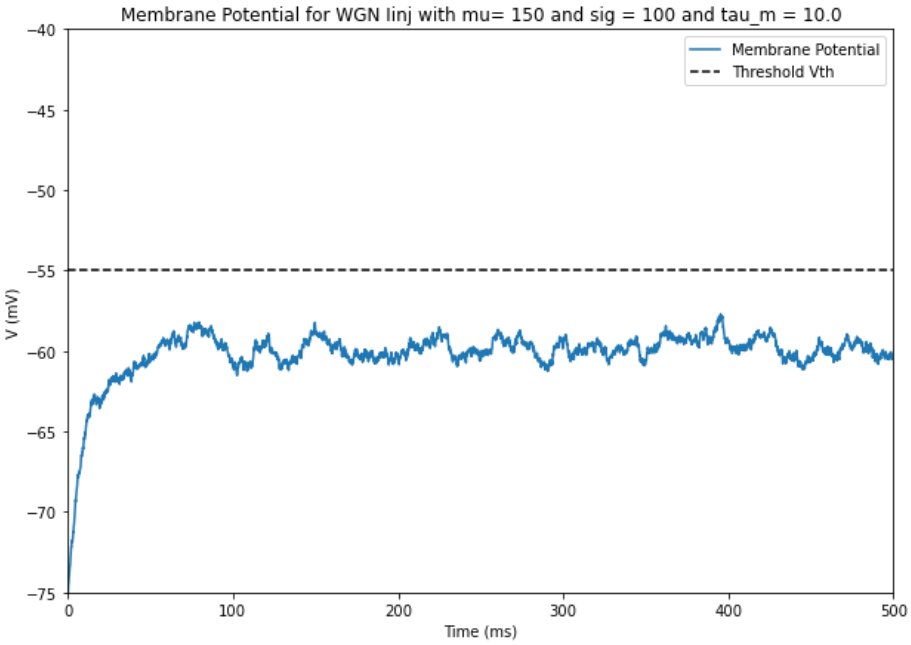
As expected from eq (2), increase in tau_m causes a decrease in dV/dt. Thus, more time is required to reach the threshold from reset voltage and the frequency of spikes decreases.

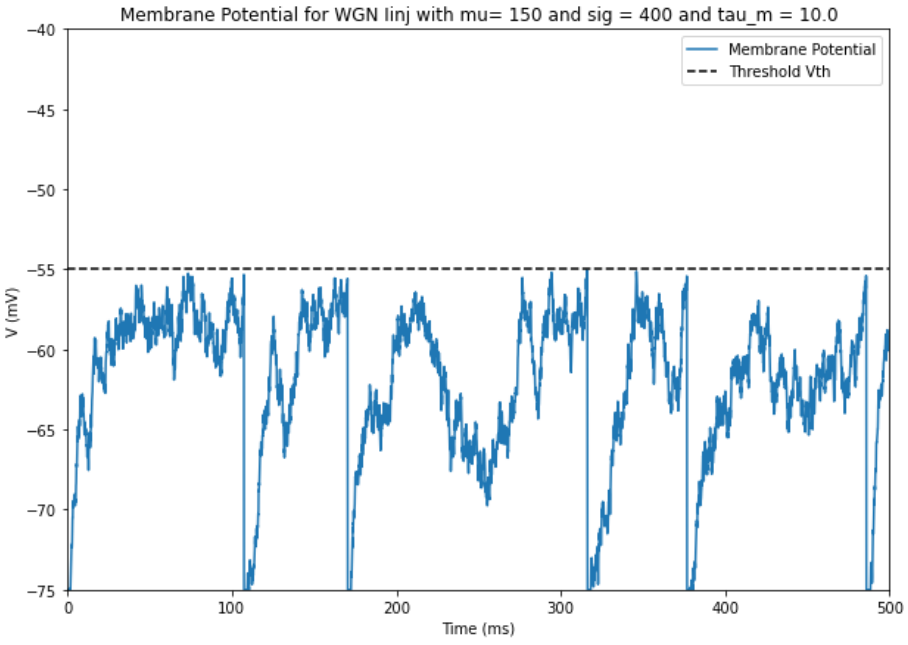
II. GWN Current:

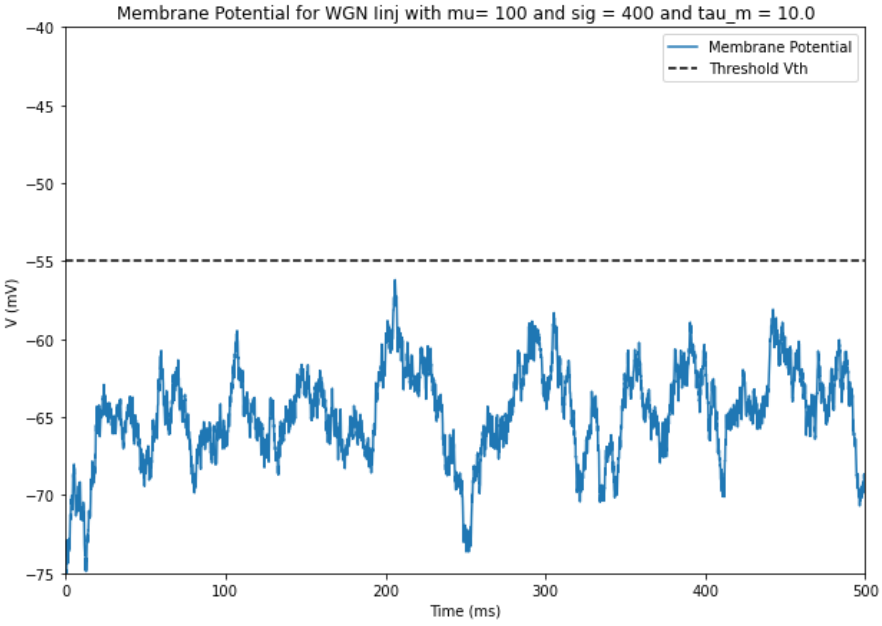
My_GWN function description:

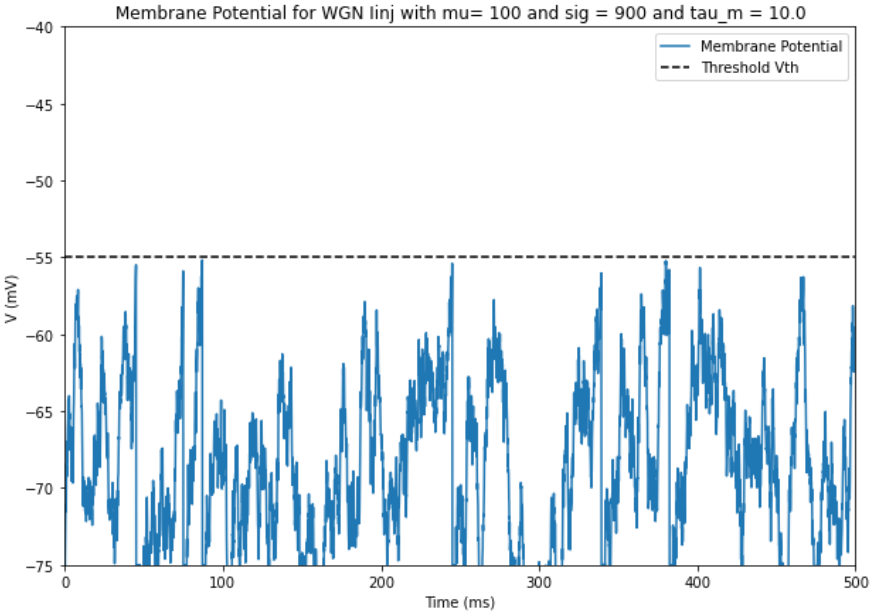
First we set the required parameters using default_pars function. Next, is the input ‘myseed’ is false, the seed for random WGN generation is not set. Else, we set a specific seed for our random process so each time we run this function, we get same WGN in the output.
Now we generate a gaussian noise. Np.random.randn generates an array of Gaussian noise with mu=0 and sig=1 (a normal Gaussian noise). If we want to change these two parameters, we need to multiply this array to sig, and finally add ‘mu’ to it. Our WGN current is now ready!

Now we're going to add a WGN to our Iinj. First, we look at the parameter 'sig'. As we increase the variance, the probability of choosing values larger than average increases (as well as lower values!). Hence, the minimum average 'mu' needed to make the neuron spike reduces with increase in 'sig' (because it is more probable that the current gets large enough to make the neuron reach its threshold voltage). Here are some example plots (remember from previous section that assuming sig=0, the minimum required mu for spike is 200):









It also can be seen that, as we increase sig, the signal and the spikes become more irregular, because the signal becomes noisier!

- **Analyzing GWN effects on spiking**
  As discussed above, while we increase either DC current value or mu in GWN, spikes rate increases too. We also saw that increasing sig causes more irregularity and makes the potential noisier. Thus, when we increase sig, we can reach threshold level with smaller mu, but the spikes would be irregular.

- **The Interspike Interval (ISI)**
  The ISI refers to the time between two consecutive spikes in neuron. Hence, if we have N spikes in a timeline (trial), the ISI

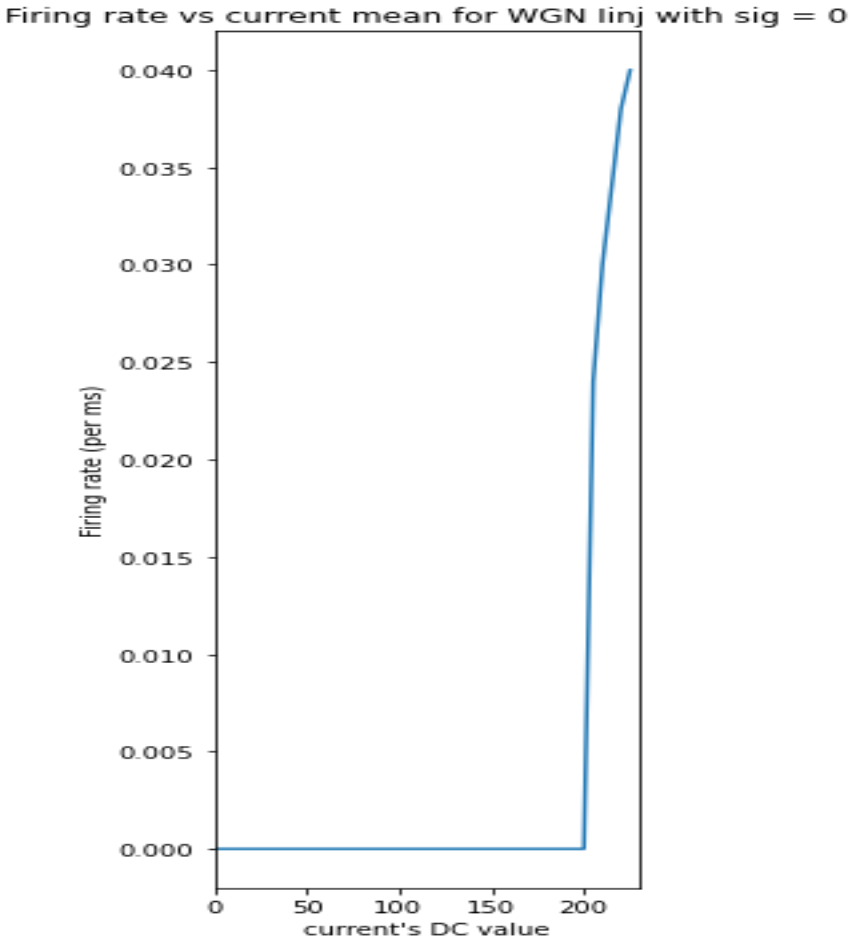
array contains N-1 elements that specify the times between these N spikes.

- # Firing rate and spike time irregularity

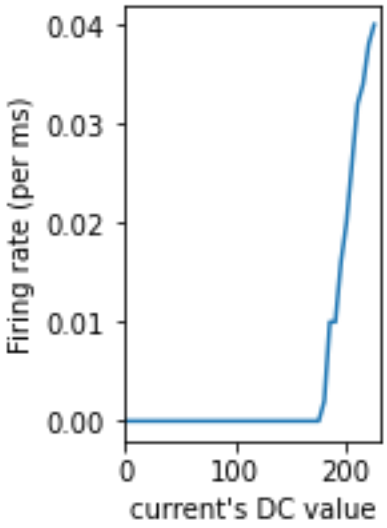
  - **F-I explorer for different sig_wgn**

    As seen above, when we increase sig for a low-mean WGN current, the spike probability increases because of high-amplitude fluctuations (i.e. the firing rate increases). The F-I curve plots firing rate as a function of DC part in the input current. Therefore, with an increase in sig, we expect that the slope of this curve increase too. Furthermore, the rise current (the current in which a spike happens for the first time) should decrease. However, the spikes would be more irregular, as mentioned before.

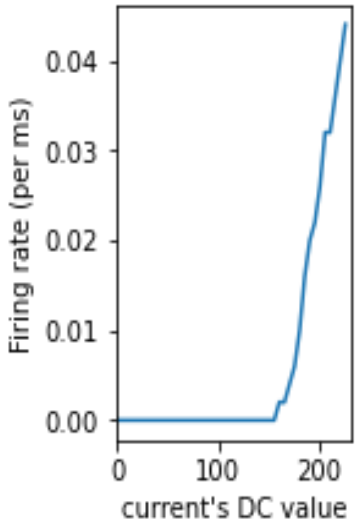
    A sample code has been written for plotting F-I curve for different values of sig. We have two ‘for’ loops. In the first one, the parameter ‘sig’ is changed from 0 to 500. In the second one, we compute firing rate for different values of mu, and obtain a firingRate_arr for our figure. The results are as follows:



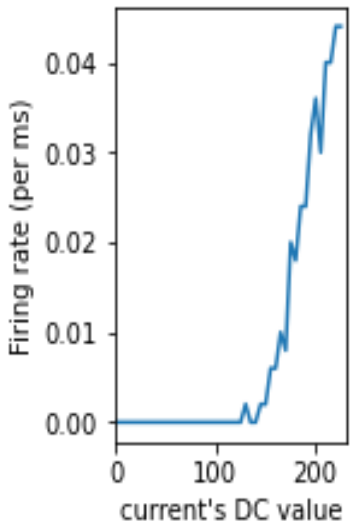
Firing rate vs current mean for WGN Iinj with sig = 100

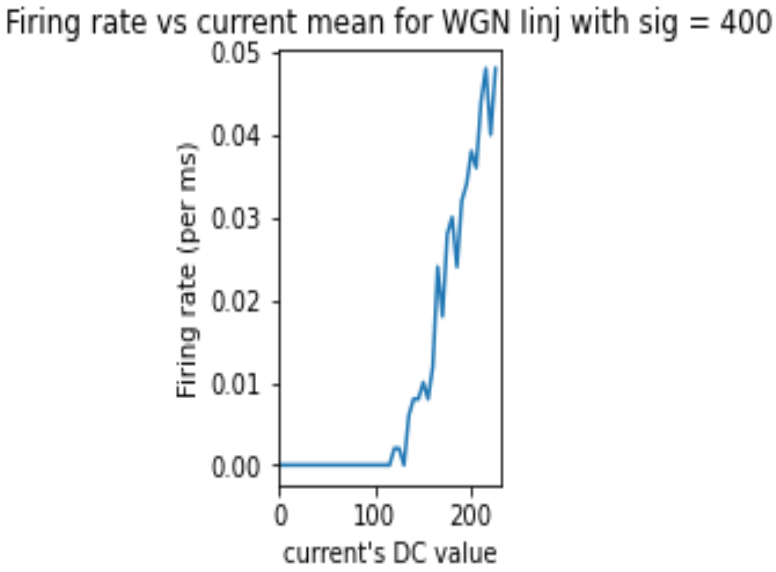


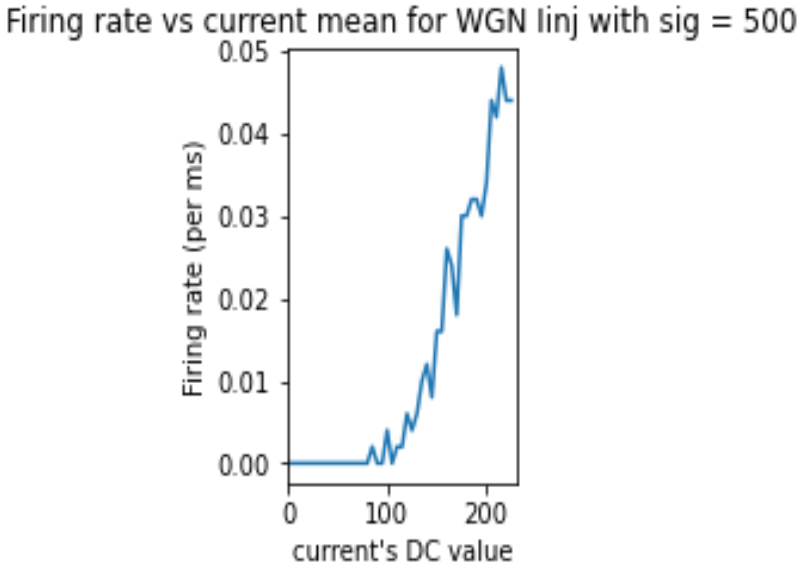
Firing rate vs current mean for WGN Iinj with sig = 200



Firing rate vs current mean for WGN Iinj with sig = 300





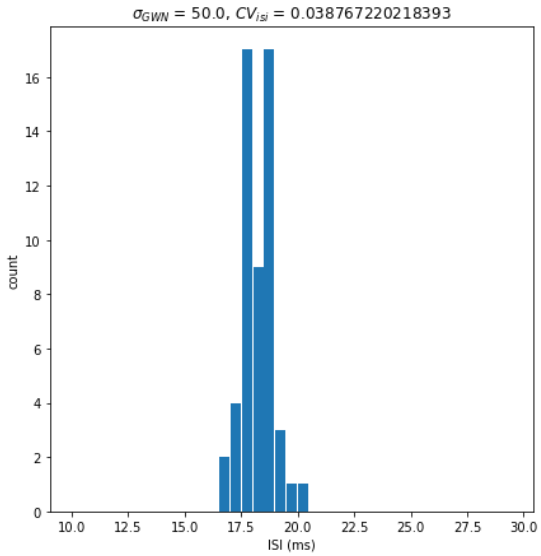


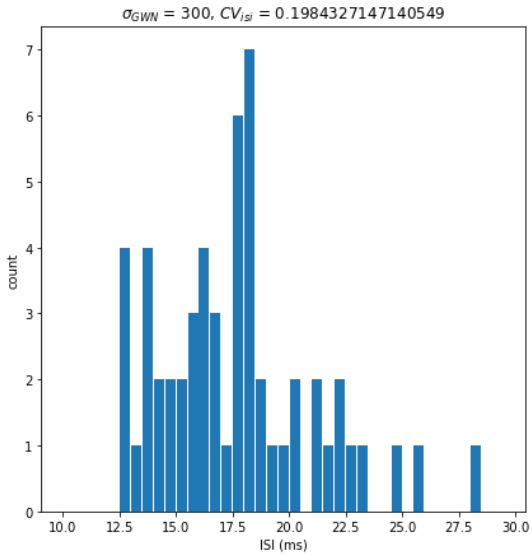
We can verify our expectations using the figures. As we increase sig, the rise current reduces (from 200 pA to under 100 pA), and the figure's slope increases such that the firing rate reaches 0.05 for sig=400 & 500.

- ✓ **isi_cv_LIF function description:**

  According to the definition of ISI, we can simply differentiate 'spike_times' to obtain the values of 'isi' (if its length is more than 1!). Then, by dividing its std to its mean, the value of CV is calculated.

  To visualize the effect of 'sig' on ISI, we plot two histograms for two different values of 'sig':



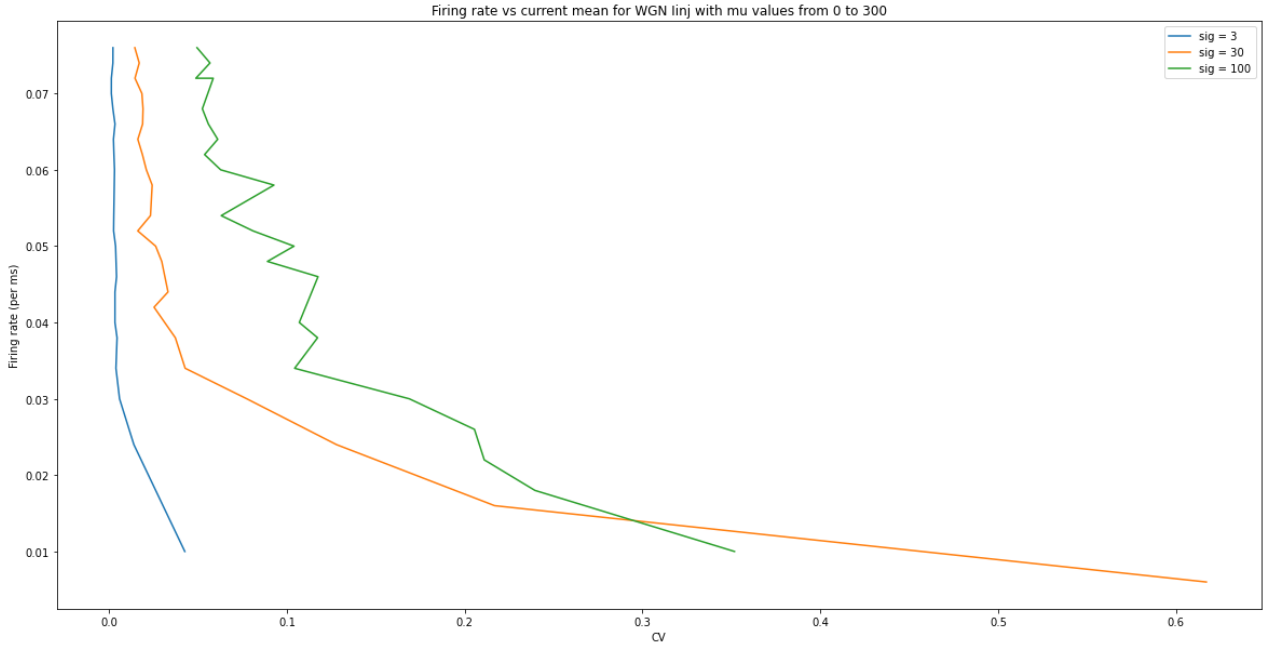


As we can see, increasing 'sig' increases the CV too. This is because the current and membrane potential become noisier, and the spike times get more irregular. Hence, the STD of ISI increases and histogram turns wider. Furthermore, we'll have a larger value for CV because of std(ISI) on its numerator.

- **Spike irregularity explorer for different sig_wgn**

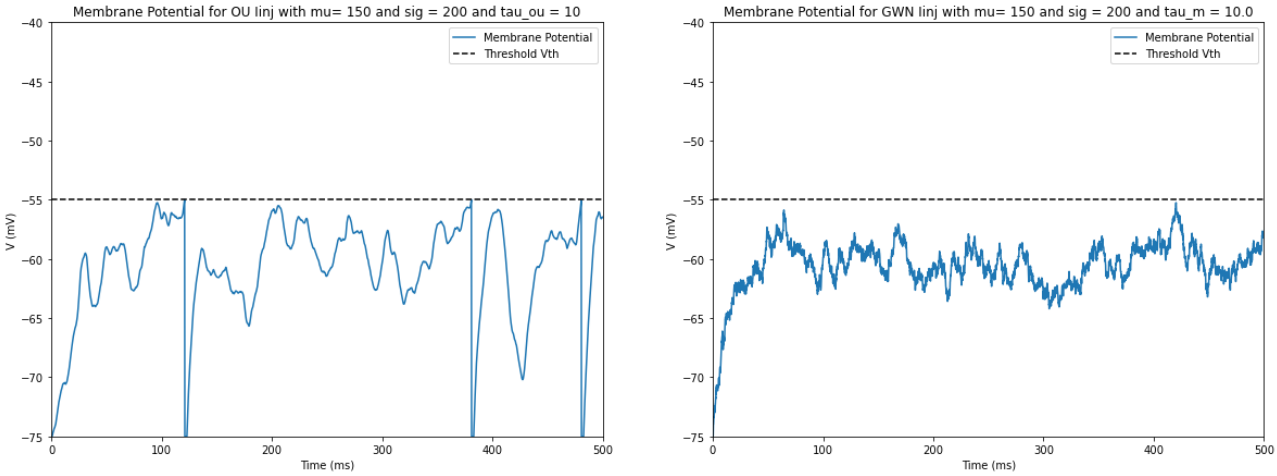
  As we increase sig, the current and Membrane potential turn noisier. Thus, the spike times and therefore the isi become more irregular. This irregularity increases the std of ISI and therefore, the parameter CV increases.

  The effect of sig on the F-I curve has been mentioned above. According to this discussion, we can conclude that if we plot firing rate vs CV for different values of 'mu', we may realize an indirect relationship between these two. Besides, we have concluded that increasing 'sig' makes CV larger. We can confirm these expectations in the figure below:



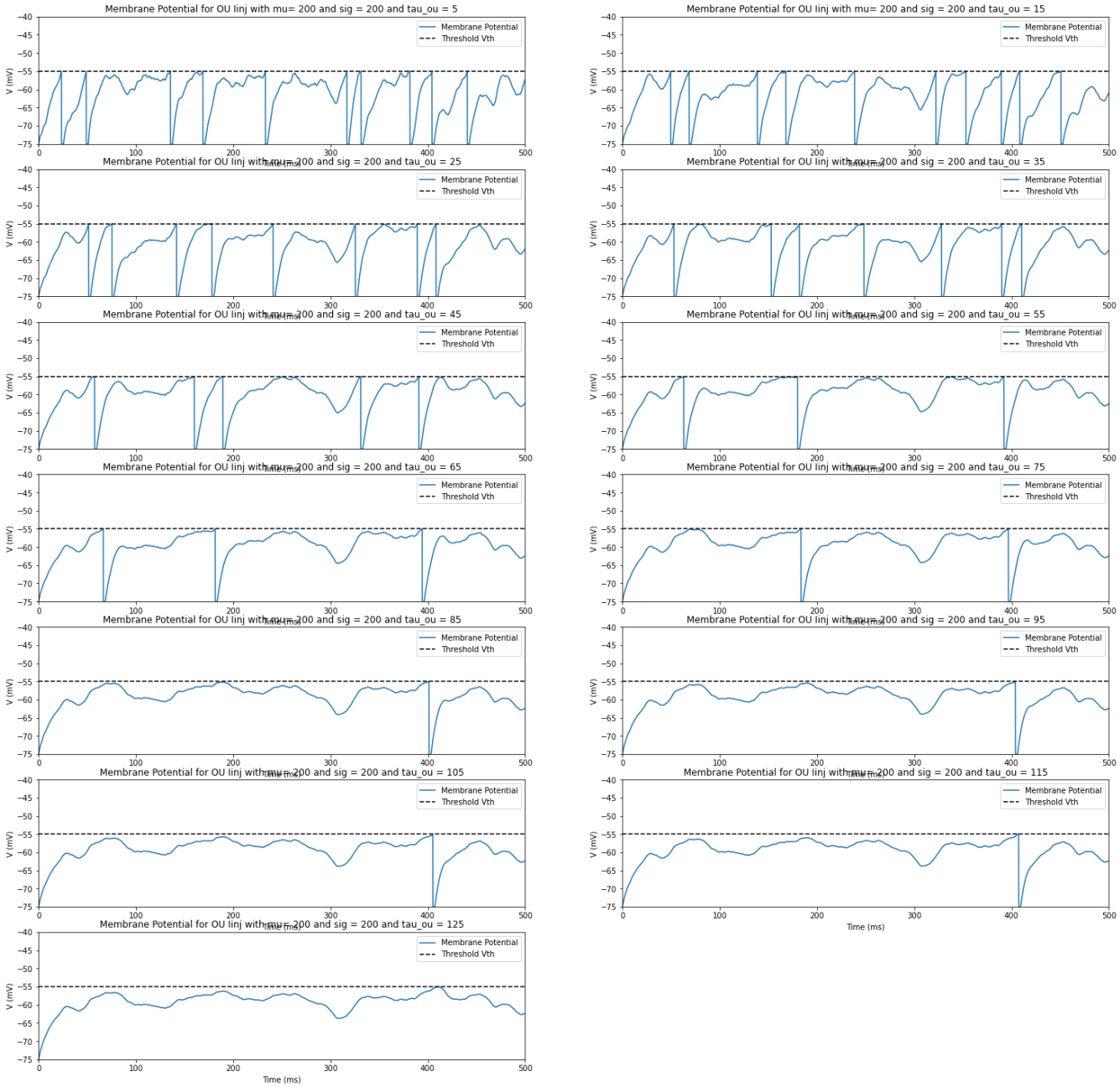
- **Ornstein-Uhlenbeck Process**
  - ✓ **My_OU function description:**
    We first set default parameters and also seed, just like my_GWN function. Then we define a normal Gaussian noise as $\zeta(t)$. Then we need to obtain an expression using the mentioned differential equation for OU process. We make I iteration-by-iteration and set it as the output of the function.

Now we compare the response of the neuron to GWN and OU process with same amplitude (sig) and average (mu) and same time constants (tau_m = tau_ou):



As expected, the high-frequency terms are removed in the OU process. Furthermore, with similar parameters, firing is more probable in OU input compared to GWN input. In other words, with same ‘sig’ in these two inputs, the amplitude of fluctuations in colored noise is higher than white noise. Hence, the potential reaches its threshold more frequently.

To consider the effect of time constant tau_ou, we plot Membrane potential for fixed values of ‘sig’ and ‘mu’, and different values of tau_ou. We already know that this parameter sets the time needed to reach the final value. Therefore, we expect that with a decrease in time constant, the potential reaches its final value and as a result, we see the spikes earlier in the time range. This means that the firing count (rate) is expected to increase in this situation. The results are as follows:



The figures confirm our expectations, as the firing rate is higher in small time constants.

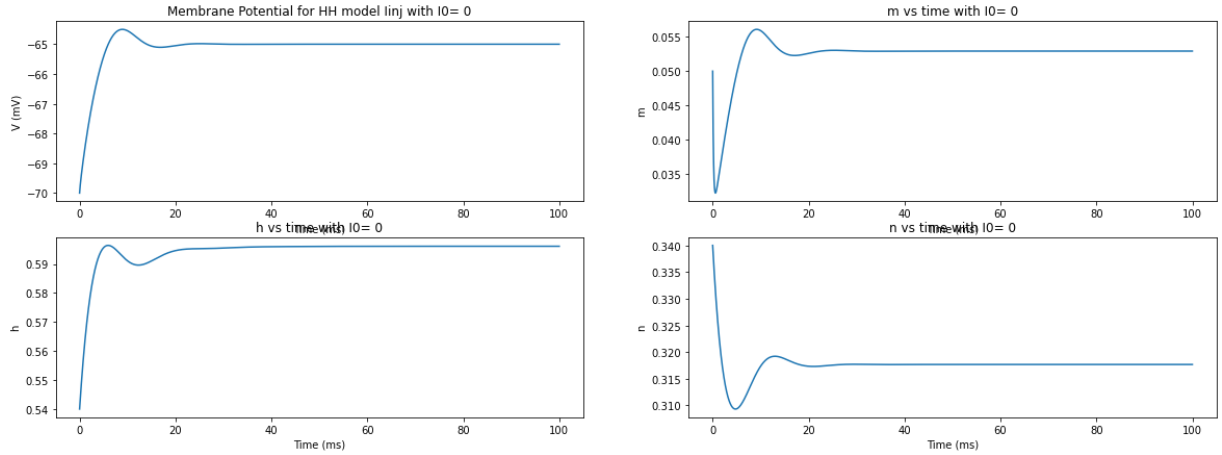
- # The Hodgkin-Huxley model
  - ✓ **HH function description:**

    First, it is important to note that the parameters ‘alpha’ and ‘beta’ are rates of transition from closed to open, and open to close, respectively. These parameters are functions of voltage and these functions have already been written in the code. Now in the HH function, we set total time samples, resting potential for every ion, and maximum conductance for each ion’s model.

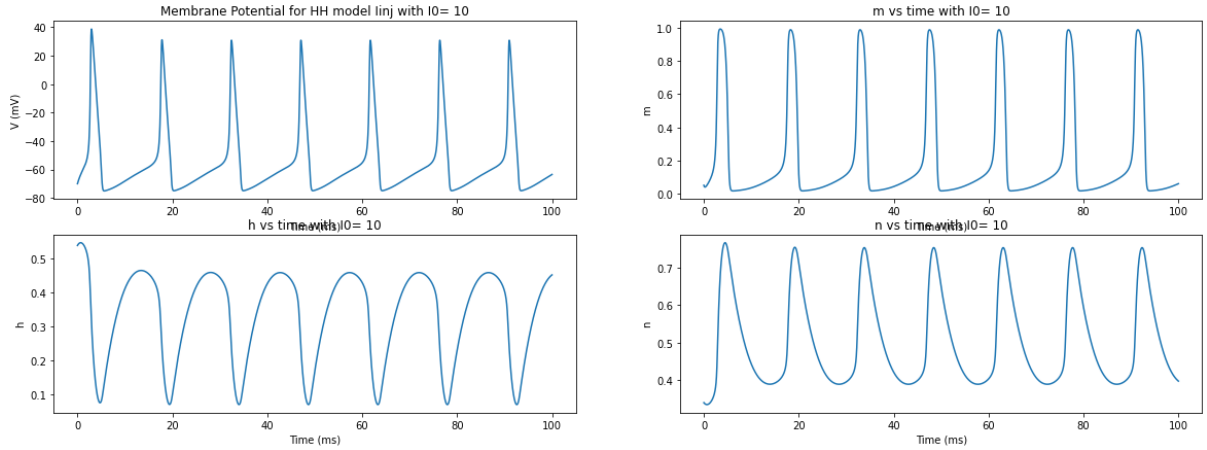
Now we derive expressions for 4 parameters V,m,n,h using our 4 differential equations, and compute their values iteration-by-iteration. Note that the actual conductance for the ions is a function of parameter m,h or n:

$$-C_m \frac{dV}{dt} = g_L(V - E_L) + \overline{g_K}\, n^4(V - E_K) + \overline{g_{Na}}\, m^3 h(V - E_{Na}) - I_e$$

We also include the effect of non-zero rest voltage in neurons, so we add 65mV to the voltages in the equation above (as added in alpha and beta equations).
Finally, we set the two inputs I0,T0, and plot these parameters in time. Here are the results for I0=0:



No spike happens as expected. Now we increase the current to 10:



This time a sequence of spikes happen.

✓ **Bonus:**

Here we update our HH function and return gL,gNa,gK arrays as our new outputs. We can compute the value of each conductance using the m,h,n parameters that mentioned above in the differential equation.

We now plot dynamics of gL,gNa,gK. It is clear that the leakage conductance gL is constant over time and does not change with other parameters such as V. But for the other two, we know that Na+ and K+ channels require a threshold voltage to open. While the input current increases the voltage, the voltage reaches Na+ channels threshold first and opens these channels (gNa+ rises). When Na+ channels open, the voltage continues to rise more rapidly so it reaches the K+ channels threshold soon (at this time, gK rises). This causes the voltage to reduce and at the same time, Na+ channels close again due to the parameter ‘h’ (inactivation gate). Hence, we will see a decrease in gNa and gK until they come back to their rest values. This cycle repeats as long as the input current is injected: