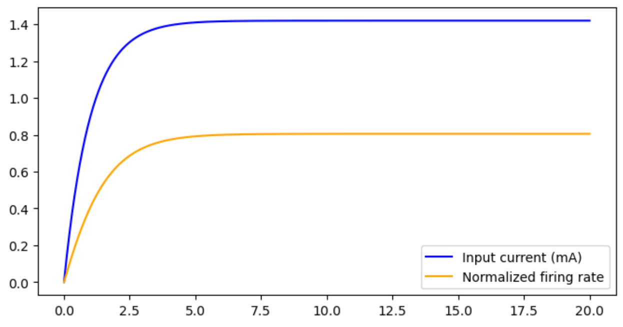
-HW2 Report-

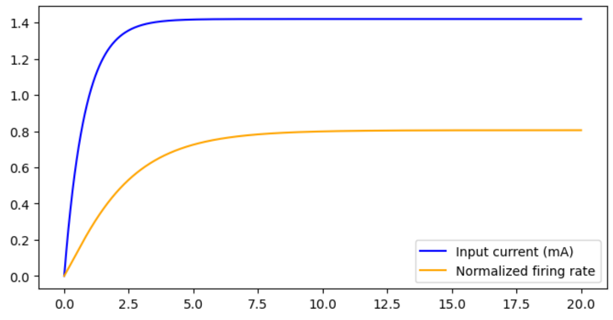
Nima Haji

400100973

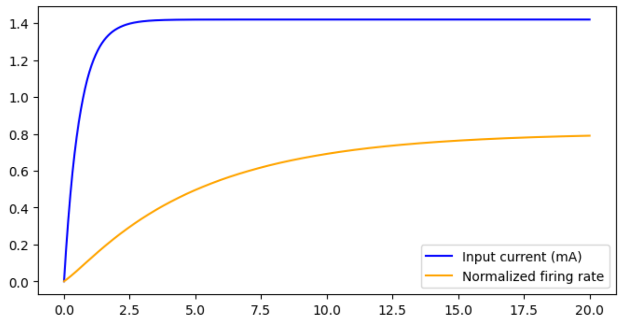
Questions:

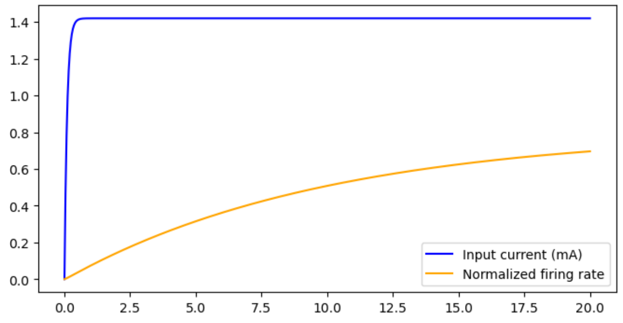
1.Manipulate the time constants. Can you make the input reach a steady state much faster than the output?





$\tau_r = 1$ $\tau_s = 1$ **$\tau_r = 2$ $\tau_s = 0.8$**



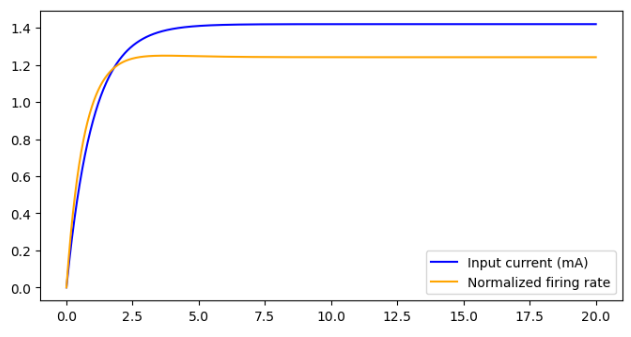


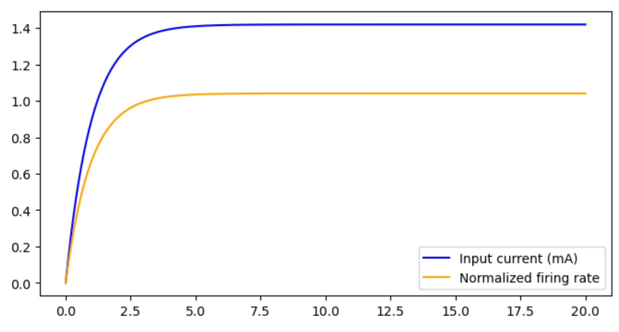
$\tau_r = 5$ $\tau_s = 0.5$ **$\tau_r = 10$ $\tau_s = 0.1$**

By manipulating time constants, input current and normalized firing rate reach their steady state at different times.

If $\tau_s \ll \tau_r$ the synaptic input converges quickly to its steady state value and as opposed to the network dynamics, output firing rate takes longer to converge.

2.Try out the model for other non-linear functions. Do you spot any difference?





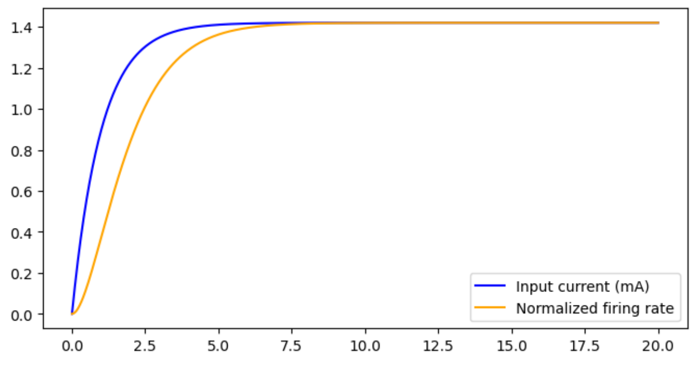
(1) $\mathbf{1 + e^{-I}}$ **(2)** $\mathbf{1 + \frac{0.1}{1+I}}$

As we see, the steady state values of new plots are different from the first plot.

First plot has higher values while second one has lower steady state values.

Now we choose function as below which could model our system properly.

$$F(I_s) = \begin{cases} 0 & if\ I_s < 0 \\ I_s & if\ I_s \geq 0 \end{cases}$$



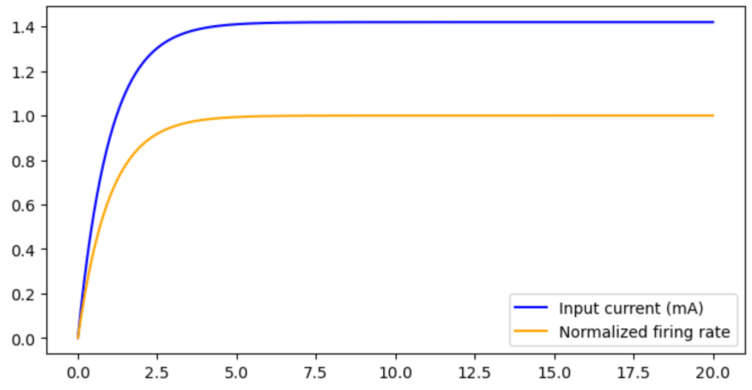
Max(0,I)

The steady state value for this function compare to the first one, is higher.

(All results has the same time constants) ($\tau_r = 1$, $\tau_s = 1$)

3.Can you compare this model with a perceptron network? Can we classify input using this network as well?

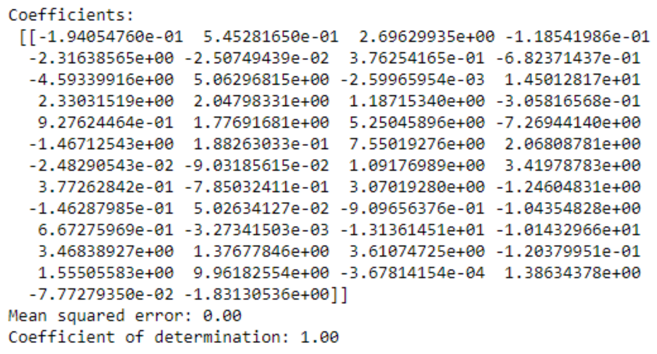
We can assign function F to the step function with non-zero threshold to generate perceptron network. This model has lower steady state value compare to the first one.



u(t)

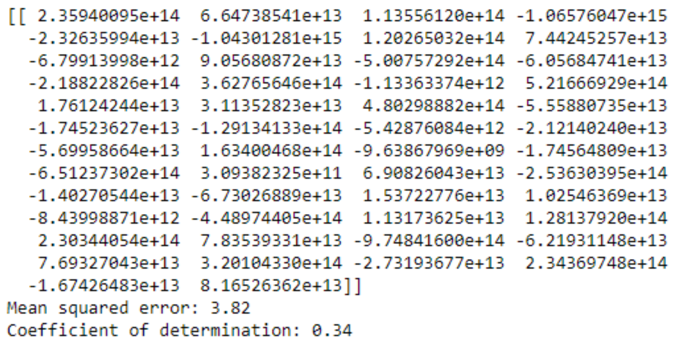
We can classify input using this network. A feedforward neural network (FNN) usually has one or more hidden layers between the input and output layers. Such layered FNN is called an MLP.
There are M layers, and layer m has Jm nodes. All the input nodes are collectively treated as the first layer.
For classification, all hidden and output neurons are with a sigmoidal function or hard limiter. For function approximation, all hidden neurons are with a sigmoidal function, but the output neurons in layer M use a linear activation function.

Bonus :



```
Coefficients:
 [[-1.94054760e-01  5.45281650e-01  2.69629935e+00 -1.18541986e-01
  -2.31638565e+00 -2.50749439e-02  3.76254165e-01 -6.82371437e-01
  -4.59339916e+00  5.06296815e+00 -2.59965954e-03  1.45012817e+01
   2.33031519e+00  2.04798331e+00  1.18715340e+00 -3.05816568e-01
   9.27624464e-01  1.77691681e+00  5.25045896e+00 -7.26944140e+00
  -1.46712543e+00  1.88263033e-01  7.55019276e+00  2.06808781e+00
  -2.48290543e-02 -9.03185615e-02  1.09176989e+00  3.41978783e+00
   3.77262842e-01 -7.85032411e-01  3.07019280e+00 -1.24604831e+00
  -1.46287985e-01  5.02634127e-02 -9.09656376e-01 -1.04354828e+00
   6.67275969e-01 -3.27341503e-03 -1.31361451e+01 -1.01432966e+01
   3.46838927e+00  1.37677846e+00  3.61074725e+00 -1.20379951e-01
   1.55505583e+00  9.96182554e+00 -3.67814154e-04  1.38634378e+00
  -7.77279350e-02 -1.83130536e+00]]
Mean squared error: 0.00
Coefficient of determination: 1.00
```

Previous part



```
[[ 2.35940095e+14  6.64738541e+13  1.13556120e+14 -1.06576047e+15
  -2.32635994e+13 -1.04301281e+15  1.20265032e+14  7.44245257e+13
  -6.79913998e+12  9.05680872e+13 -5.00757292e+14 -6.05684741e+13
  -2.18822826e+14  3.62765646e+14 -1.13363374e+12  5.21666929e+14
   1.76124244e+13  3.11352823e+13  4.80298882e+14 -5.55880735e+13
  -1.74523627e+13 -1.29134133e+14 -5.42876084e+12 -2.12140240e+13
  -5.69958664e+13  1.63400468e+14 -9.63867969e+09 -1.74564809e+13
  -6.51237302e+14  3.09382325e+11  6.90826043e+13 -2.53630395e+14
  -1.40270544e+13 -6.73026889e+13  1.53722776e+13  1.02546369e+13
  -8.43998871e+12 -4.48974405e+14  1.13173625e+13  1.28137920e+14
   2.30344054e+14  7.83539331e+13 -9.74841600e+14 -6.21931148e+13
   7.69327043e+13  3.20104330e+14 -2.73193677e+13  2.34369748e+14
  -1.67426483e+13  8.16526362e+13]]
Mean squared error: 3.82
Coefficient of determination: 0.34
```

After adding noise

If we add Gaussian noise, our accuracy will be less than 1. (here we have 34%)