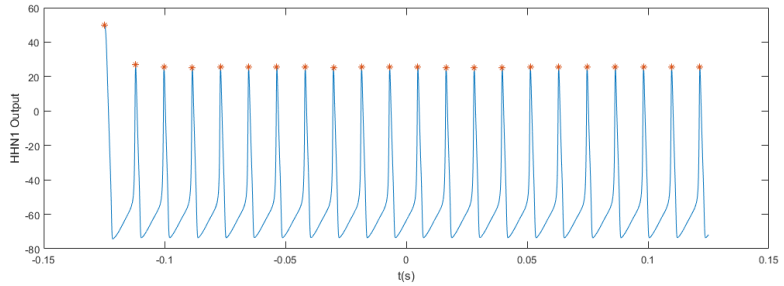
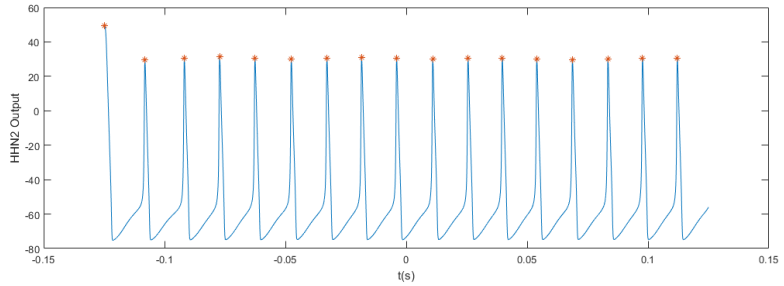
PROJECT #2

Name: Jiayi Zhang UNI: jz2856

(i)

Following the encoding circuits we can generate 2 sequences of spike times on the time interval. We should use convolution to simulate the h^1 and h^2 filtering $u(t)$. Besides, the feedback part of code should be include within the Hodgkin-Huxley loop.





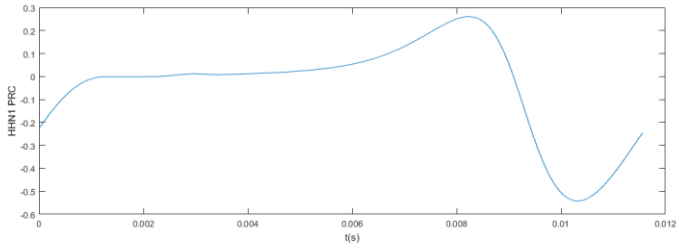

```
uh1 = conv(ut,h1)*dt;uh1=uh1(125000:375000);
uh2 = conv(ut,h2)*dt;uh2=uh2(125000:375000);
```

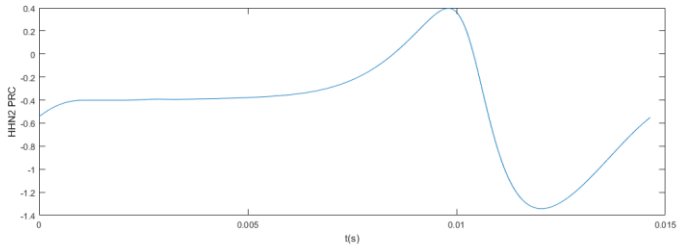
```
for i=2:length(t)
    v1(i) = uh1(i);
    v2(i) = uh2(i);
    if spikenumber1>0
        for j=1:spikenumber1
            v1(i)=v1(i)+h11(i-spikeindex1(j));
            v2(i)=v2(i)+h12(i-spikeindex1(j));
        end
    end
    if spikenumber2>0
        for j=1:spikenumber2
            v1(i)=v1(i)+h21(i-spikeindex2(j));
            v2(i)=v2(i)+h22(i-spikeindex2(j));
        end
```

```
    end
end
```

(ii)

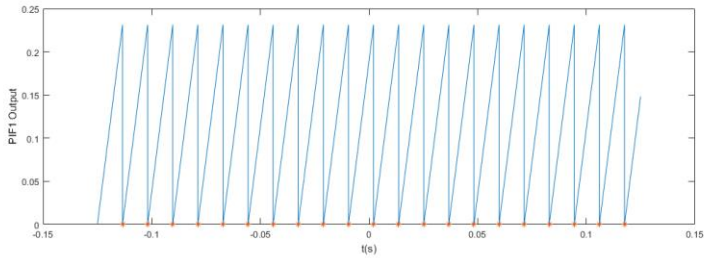
To replace Hodgkin-Huxley neurons with PIF neurons, we have to find the PRC of H-H neurons first.

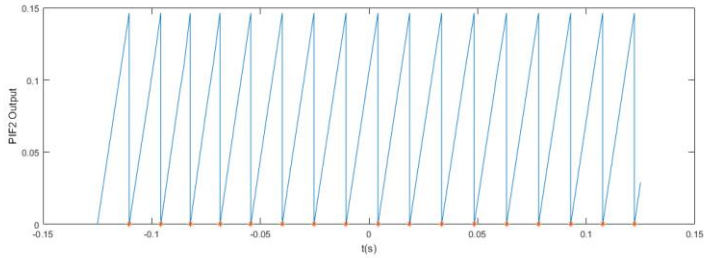





```
dt = 1e-6;
step = 15;
I = zeros(size(t));
V = hh(t,I,20);
refindex=find(findspike(V));
prc_N = refindex(end-1)-refindex(end-2);
prc_T1 = prc_N*dt;
PRC1 = zeros(1,prc_N);
for i=1:step:prc_N
    I = zeros(size(t));
    I(i+prc_N) = 2500;
    V = hh(t,I,20);
    shiftindex=find(findspike(V));
    PRC1(i:(i+step-1))= (shiftindex(end)-refindex(end))/prc_N*2*pi;
end
PRC1(PRC1>pi) = PRC1(PRC1>pi) - 2*pi;
PRC1(PRC1<-pi) = PRC1(PRC1<-pi) + 2*pi;
```

Then applying the integration of I/O equivalent PIF neuron: $\int b + u(s)\psi(s - t_k)ds$ and set the threshold to $b(t_{k+1} - t_k)$.



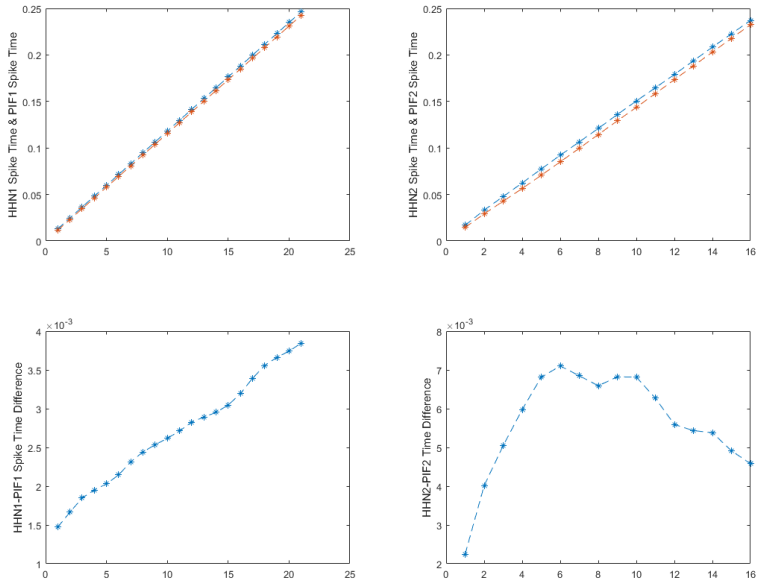



```
PRCrep1 = [PRC1 PRC1];
PRCrep2 = [PRC2 PRC2];
th1 = prc_T1*20;
th2 = prc_T2*10;
for i=1:length(t)
    v1(i) = uh1(i);
    v2(i) = uh2(i);
    if pspikenumber1>0
        for j=1:pspikenumber1
            v1(i)=v1(i)+h11(i-pspikeindex1(j));
            v2(i)=v2(i)+h12(i-pspikeindex1(j));
        end
    end
    if pspikenumber2>0
        for j=1:pspikenumber2
            v1(i)=v1(i)+h21(i-pspikeindex2(j));
            v2(i)=v2(i)+h22(i-pspikeindex2(j));
        end
    end

    intv1=intv1+dt*(20+v1(i)*PRCrep1(i_prc1));
    if intv1>th1
        intv1 = intv1 - th1;
        i_prc1 = 1;
        pspikenumber1 = pspikenumber1+1;
        pspikeindex1(pspikenumber1)=i;
    else
        i_prc1 = i_prc1+1;
```

```
    end
    PIF1(i)=intv1;
end
```

New spike times generated. Then compare the PIF spike time sequences and H-H neuron spike time sequences. (Blue for H-H and Orange for PIF)



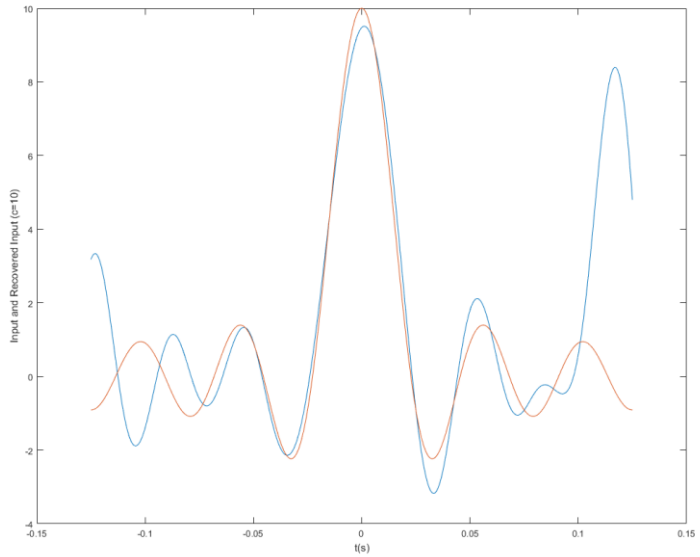
(iii)

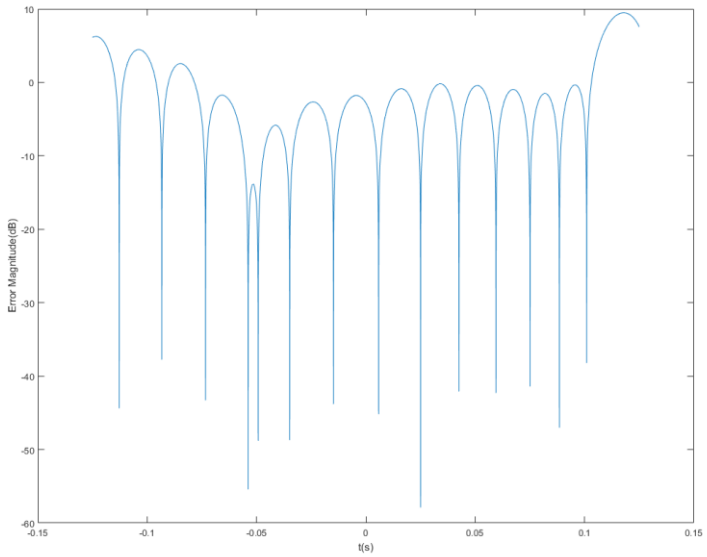
We can derive the $\mathbf{q}$ matrix and $\mathbf{G}$ matrix for the encoding circuit in the project first.

Considering the T-transform of PIF neuron: $\int_{t_k}^{t_{k+1}} u(s)\psi(s-t_k)ds = T(b) - b(t_{k+1}-t_k)$ and filter h^1 and h^2 as well as feedback $h^{11}, h^{12}, h^{21}, h^{22}$ in encoding process, we can find $q_k^1 = \int_{t_k^1}^{t_{k+1}^1}(u * h^1)(s)\psi_1(s-t_k^1)ds = \delta^1 - b(t_{k+1}^1 - t_k^1) - \psi_1(s-t_k^1)(\sum_{l\le k}\int_{t_k^1}^{t_{k+1}^1} h^{11}(s - t_l^1)ds + \sum_{t_l^2<t_k^1}\int_{t_k^1}^{t_{k+1}^1} h^{21}(s-t_l^2)ds)$ and q_k^2 should be in similar form.

Then we can derive matrix $\mathbf{G}$ as: $G_{lk}^{ij} = \int_{t_k^i}^{t_{k+1}^i}(h^i * \tilde{h}^j * g)(s - s_l^j)\psi_i(s - t_k^i)ds$.

Recover equation: $u(t) = \sum c_k^1 \varphi_k^1(t) + \sum c_k^2 \varphi_k^2(t)$ where $\mathbf{c} = \mathbf{G}^+\mathbf{q}$, $\mathbf{G} = \begin{bmatrix} \mathbf{G^{11}} & \mathbf{G^{12}} \\ \mathbf{G^{21}} & \mathbf{G^{22}} \end{bmatrix}$ and $\varphi_k^j(t) = (\tilde{h}^j * g)(t - s_k^j)$. (Blue for recovered signal and Orange for original signal)






```
q1 = zeros(1,length(PIF_tk1)-1);
for i = 1:length(PIF_tk1) - 1
    t_integral = PIF_tk1(i):dt:PIF_tk1(i+1);

    tk1_pick = PIF_tk1(PIF_tk1 <= PIF_tk1(i));
    temp11 = 0;
    for j = 1:length(tk1_pick)
       temp11 = temp11 + trapz(t_integral, h11(t_integral, tk1_pick(j)).*
prc1([1:length(t_integral)]));
    end

    tk2_pick = PIF_tk2(PIF_tk2 < PIF_tk1(i+1));
```

```
    temp21 = 0;
    for j = 1:length(tk2_pick)
        temp21 = temp21 + trapz(t_integral, h21(t_integral, tk2_pick(j)).*
prc1([1:length(t_integral)]));
    end
    q1(i) = thr1 - bias1 * (PIF_tk1(i+1) - PIF_tk1(i)) - temp11 - temp21;
end
```

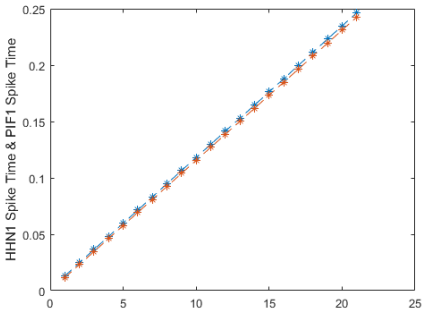
```
gt=gfunc(t);
g11=conv(conv(h1,fliplr(h1))*dt,gt*dt);
G11 = zeros(pspikenumber1-1,pspikenumber1-1);
for k=1:pspikenumber1-1
    for l=1:pspikenumber1-1
        G11(k,l)=0;
        for s=pspikeindex1(k):pspikeindex1(k+1)
            G11(k,l)=G11(k,l)+g11(s-
ceil((pspikeindex1(l+1)+pspikeindex1(l))/2)+375000)*PRCrep1(s-pspikeindex1(k)+1)*dt;
        end
    end
end
```

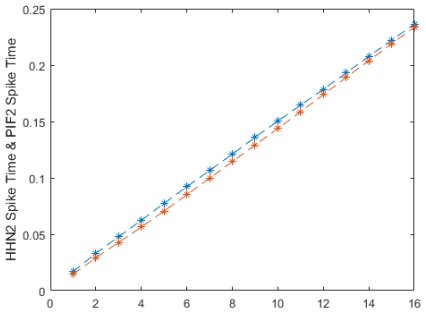
```
G = [G11,G12;G21,G22];
q = [q1,q2]';
c = pinv(G,1e-8)*q;
c1 = c(1:pspikenumber1-1);
c2 = c(pspikenumber1:end);
phi1=conv(fliplr(h1),gt*dt);
phi2=conv(fliplr(h2),gt*dt);
ut_rec = zeros(size(t));
for i = 1:length(t)
    for j = 1:length(c1)
        ut_rec(i)=ut_rec(i)+c1(j)*phi1(i-
ceil((pspikeindex1(j+1)+pspikeindex1(j))/2)+250000);
    end
    for j = 1:length(c2)
        ut_rec(i)=ut_rec(i)+c2(j)*phi2(i-
ceil((pspikeindex2(j+1)+pspikeindex2(j))/2)+250000);
    end
end
```

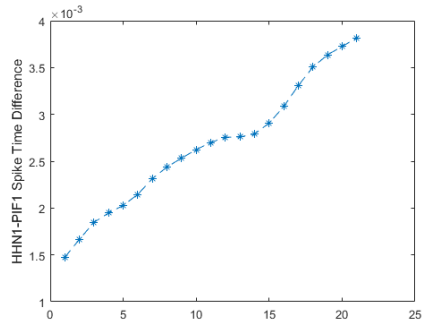
MSE=5.06671, and if amplitude normalized to 1 MSE'=0.05571.

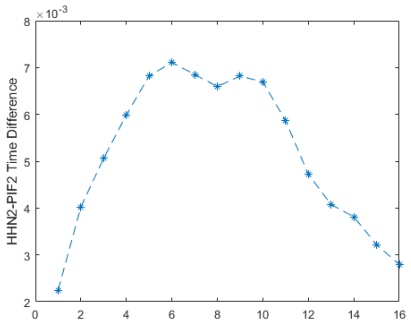
(iv)

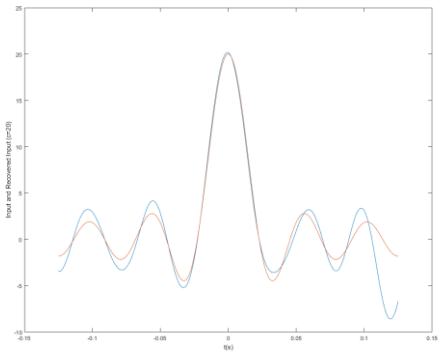
(a) For $c = 20$;

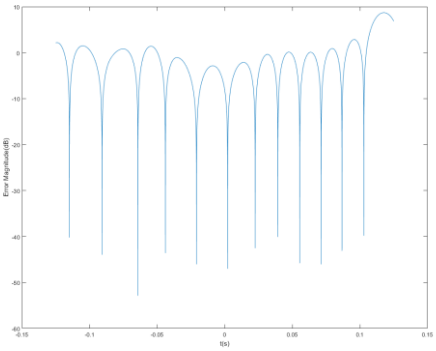






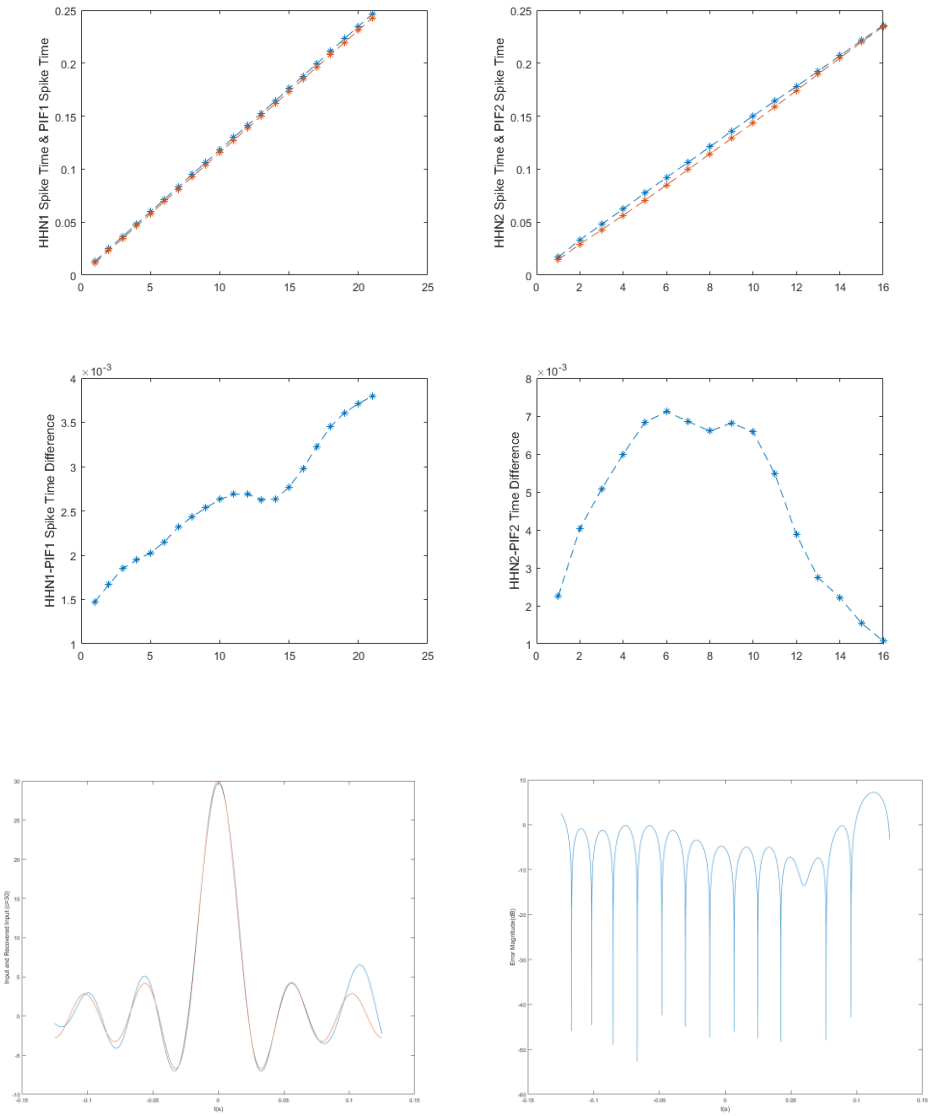






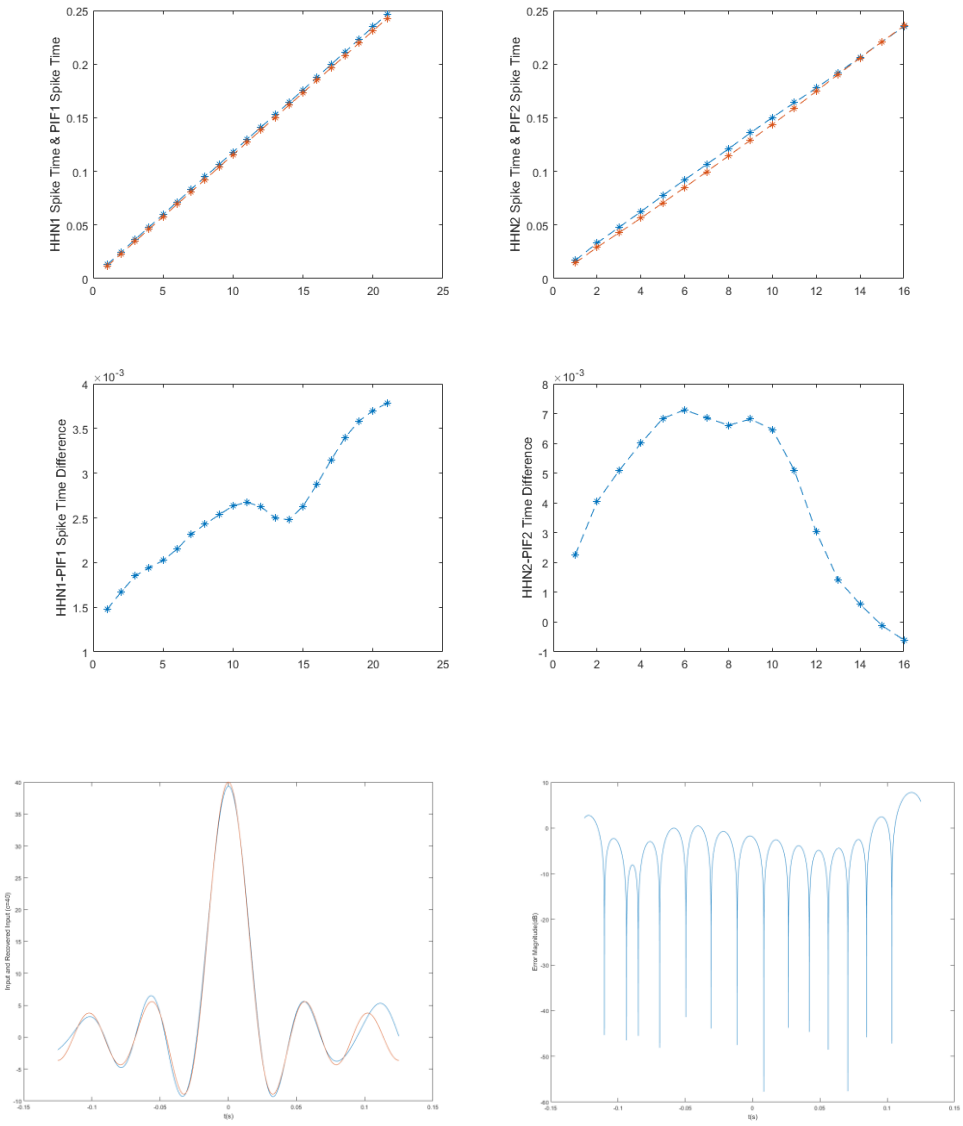
MSE= 3.38176, and if amplitude normalized to 1 MSE'= 0.00828.

(b) For $c = 30$;



MSE= 1.77162, and if amplitude normalized to 1 MSE'= 0.00206.

(c) For $c = 40$;



MSE= 2.15009, and if amplitude normalized to 1 MSE'= 0.00139.

Conculsion: If we normalize the input stimulus amplitude to 1, we can find that the MSE will decrease as c increase. That means the relative error becomes smaller when input gets bigger. From my point of view it might be caused by floating point error (which cannot be completely avoided when time domain is discrete). Compared to bias b, $u(s)\psi(s - t_k)ds$ is really small so the spike shift could also be small. However we store our spike time information as index in t matrix and the datatype is integer, let us say if the real spike shift is 1.6dt, it will be stored as 2dt in our simulation and the error will be 0.4dt. When input stimulus gets bigger, the spike shift will also gets bigger and the spike shift could be 5.6dt and be stored as 6dt. The error is also 0.4dt but compared to 5.6dt it is relatively small. Thus the error (after normalizing) will be smaller in our simulation if c gets bigger.