University of Waterloo
Faculty of Engineering
Department of System Design Engineering

SYDE 556 - Assignment 2
Spiking Neurons

200 University Avenue West
Waterloo, Ontario, Canada
N2L 3G1

Prepared by
Sugandha Sharma
ID: 20401693
userid: s72sharm
4B Electrical Engineering
7 March 2015

Table of Contents

List of Figures 3
1 Generating a Random Input Signal 5
  1.1 Gaussian White Noise 5
    a) Plot of x(t) 5
    b) Plot of the average |X(ω)| 6
  1.2 Gaussian power spectrum noise 7
    a) Plots of x(t) 7
    b) Plot the average |X(ω)| 9
2. Simulating a Spiking Neuron 10
  a) Spike output for a constant input of x=0 and x =1 11
  b) Discussion on the observed number of spikes 13
  c) Spike output for x(t) with T = 1 14
  d) Spike output for x(t) with T = 0.2 15
  e) Bonus Question 16
3 Simulating Two Spiking Neurons 18
  a) Plot of x(t) and the spiking output for x(t)=0 18
  b) Plot of x(t) and the spiking output for x(t)=1 20
  c) Plot of x(t) and the spiking output for x(t)=12sin(10πt) 23
  d) Plot of x(t) and the spiking output for a random signal 24
4. Computing an Optimal Filter 25
  a) Document the code 25
  b) Time and frequency plots for the optimal filter 25
  c) x(t) signal, the spikes, and the decoded xhat(t) value 26
  d) Power Spectrum Plots 27
  e) h(t) time plots for the optimal filter for different limit values 30
  f) h(t) time and |H(ω)| power spectrum plots for different 'T' values 33
5. Using Post-Synaptic Currents as a Filter 36
  a) Plot of normalized h(t) for n=0, 1, and 2 36
  b) Plot of normalized h(t) for τ=0.002, 0.005, 0.01, and 0.02 seconds 37
  c) Decode xhat(t) using spikes from LIF neurons 38
  d) generate a new x(t) with limit=5Hz 40

List of Figures

Figure 1. Plot of x(t) with frequency limit 5Hz 5
Figure 2. Plot of x(t) with frequency limit 10Hz 5
Figure 3. Plot of x(t) with frequency limit 20Hz 6
Figure 4. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz. 6
Figure 5. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz (blown-up version) 7
Figure 6. Plot of x(t) with frequency limit 5Hz 8
Figure 7. Plot of x(t) with frequency limit 10Hz 8
Figure 8. Plot of x(t) with frequency limit 20Hz 9
Figure 9. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz. 9
Figure 10. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz (blown-up version) 10
Figure 11. Demo of spike generation with spiky LIF neuron model 11
Figure 12. Voltage output for a constant input of x=0 over 1 second 11
Figure 13. Spike output for a constant input of x=0 over 1 second 12
Figure 14. Voltage output for a constant input of x=1 over 1 second 12
Figure 15. Spike output for a constant input of x=1 over 1 second 13
Figure 16. Voltage output for x(t) over T = 1 second with dt = 0.001 15
Figure 17. Spike output for x(t) over T = 1 second with dt = 0.001 15
Figure 18. Voltage output for x(t) over T = 0.2 seconds with dt = 0.001 16
Figure 19. Spike output for x(t) over T = 0.2 seconds with dt = 0.001 16
Figure 20. Output for X(t) over T = 0.2 seconds with Method 2 18
Figure 21. x(t) and the spiking output for x(t)=0 and encoder = 1 19
Figure 22. x(t) and the Voltage output for x(t)=0 and encoder = 1 19
Figure 23. x(t) and the spiking output for x(t)=0 and encoder = -1 20
Figure 24. x(t) and the Voltage output for x(t)=0 and encoder = -1 20
Figure 25. x(t) and the Voltage output for x(t)=1 and encoder = 1 21
Figure 26. x(t) and the Voltage output for x(t)=1 and encoder = 1 21
Figure 27. x(t) [red] and the voltage output for x(t)=1 and encoder = -1 22
Figure 28. x(t) [red] and the spiking output for x(t)=1 and encoder = -1 22
Figure 29. Voltage Output x(t)=12sin(10πt) for e=-1 shifted down by 1.5 23
Figure 30. Spike Output x(t)=12sin(10πt) for e=-1 shifted down by 1.5 23
Figure 31. x(t) and the voltage output for a random signal (neuron with e=-1 shifted down) 24
Figure 32. x(t) and the Spike output for a random signal (neuron with e=-1 shifted down) 24
Figure 33. Time Plot of an optimal Linear Decoder for a pair of LIF neurons 25
Figure 34. Frequency Plot of an optimal Linear Decoder for a pair of LIF neurons 25
Figure 35. x(t), xhat(t) and the spikes 26
Figure 36. |X(ω)| power spectrum 27
Figure 37. |R(ω)| power spectrum 28
Figure 38. |XHAT(ω)| power spectrum 29

Figure 39. Frequency and time plot of optimal filter with limit = 2Hz 31
Figure 40.Frequency and time plot of optimal filter with limit = 10Hz 31
Figure 41. Frequency and time plot of optimal filter with limit = 30Hz 32
Figure 42. Plot of optimal filter h(t) with limit = 2HZ (red), 10Hz (green), 30Hz (blue) 32
Figure 43. Plots of optimal filter and |H(w)| power spectrum for T = 1 second 33
Figure 44. Plots of optimal filter and |H(w)| power spectrum for T = 4 seconds........................... 33
Figure 45. Plots of optimal filter and |H(w)| power spectrum for T = 10 seconds......................... 34
Figure 46. Power spectrum plots for T = 1 (red), T = 4 (green), T = 10 (blue) 35
Figure 47. Optimal filter in time domain for T = 1 (red), T = 4 (green), T = 30 (blue)................. 35
Figure 48. Plot of h(t) for n = 0 (red), n = 1 (green), n = 2 (blue) with tau = 0.007s..................... 36
Figure 49. Plot of h(t) for tau = 0.002 (red), tau = 0.005 (green), tau = 0.01 (blue) with tau = 0.02 (majenta). 37
Figure 50. Time plot of the filter h(t)... 38
Figure 51. Frequency plot of the filter h(t) 38
Figure 52. Plot of x(t) [Red], xhat(t) [cyan], neuron with e =1 [yellow], neuron with e = -1 [magenta] 39
Figure 53. Plot of new x(t) [Red], xhat(t) [cyan], neuron with e =1 [yellow], neuron with e = -1 [magenta] 40

1 Generating a Random Input Signal

1.1 Gaussian White Noise

A function that generates a randomly varying x(t) signal.

a) Plot of x(t)

Plot of x(t) for three randomly generated signals with limit at 5, 10, and 20Hz. For each of these, T=1, dt=0.001, and rms=0.5.

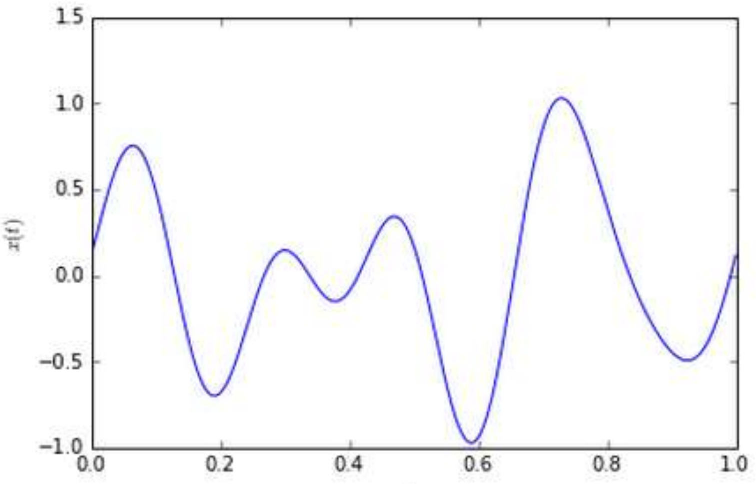


Figure 1. Plot of x(t) with frequency limit 5Hz

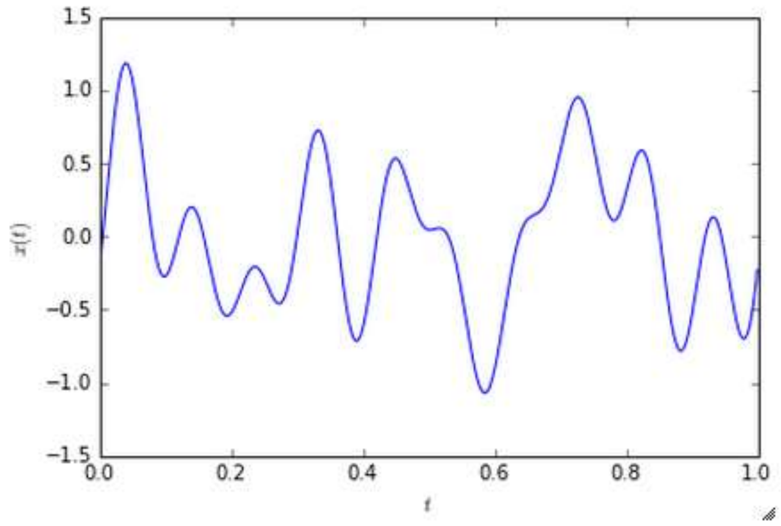


Figure 2. Plot of x(t) with frequency limit 10Hz

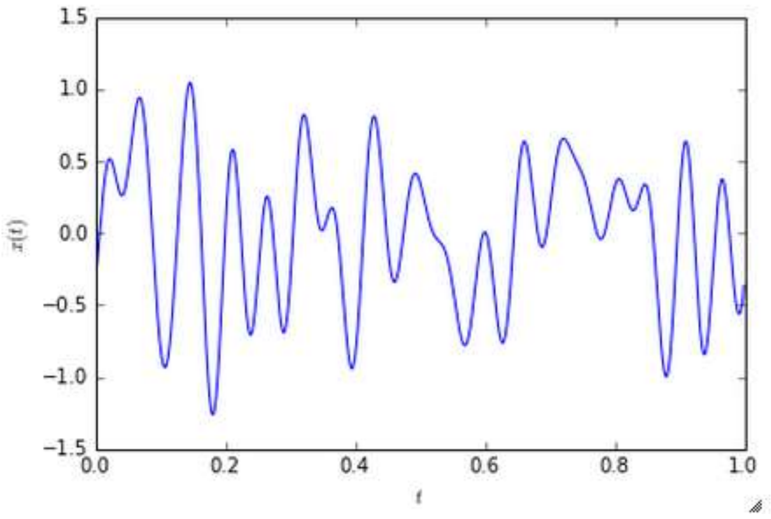


Figure 3. Plot of x(t) with frequency limit 20Hz

b) Plot of the average |X(ω)|

Plot of the average |X(ω)| (the norm of the Fourier coefficients) over 100 signals generated with T=1, dt=0.001, rms=0.5, and limit=10 (each of these 100 signals have a different seed). The plot shows the different ω values on the x-axis and the average |X| value for that ω on the y-axis.

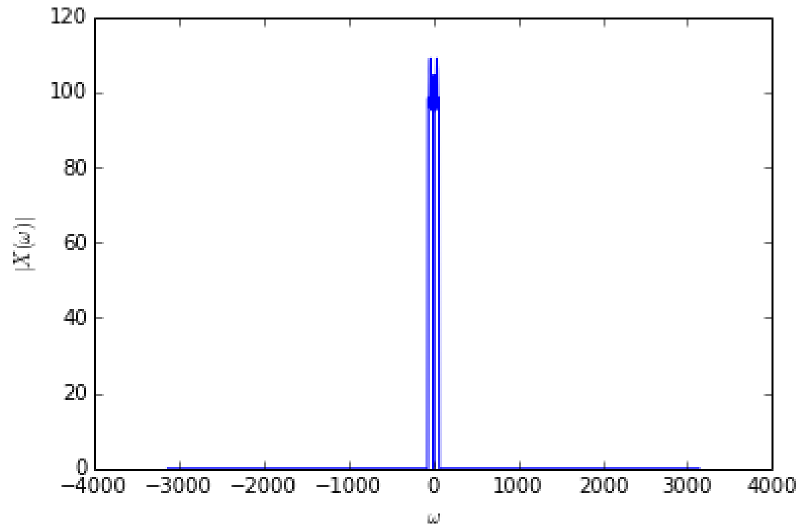


Figure 4. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz.

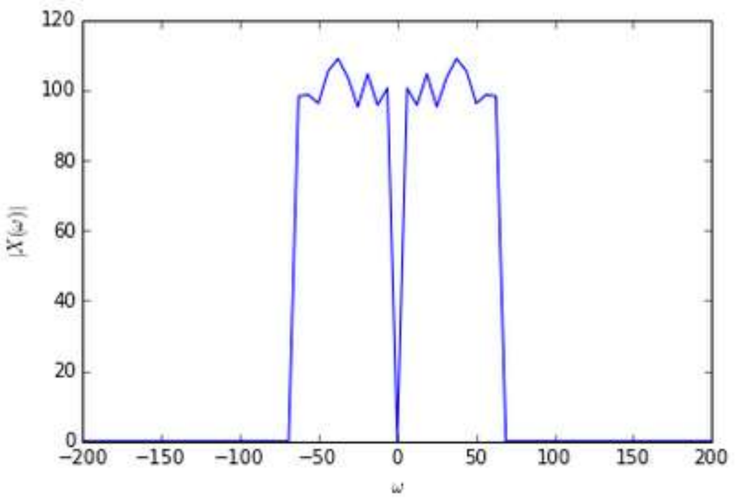


Figure 5. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz (blown-up version)

It was expected that X(w) would be equal to zero for frequencies greater than 10Hz i.e for values of mega greater than omega = 2*pi*10 = 62.83 radians. The result obtained in figure 5 complies well with what was expected.

1.2 Gaussian power spectrum noise

Created a modified version of the function from saection 1.1 that produces noise with a different power spectrum. Instead of having the X(ω) values be 0 outside of some limit and sampled from N(μ=0,σ=1) inside that limit, the new implementation has a smooth drop-off of power as the frequency increases. In particular, instead of the limit, the sampling is done from N(μ=0, σ=exp(−ω2/(2*b2))) where b is the new bandwidth parameter that replaces the limit parameter.

a) Plots of x(t)

Plot x(t) for three randomly generated signals with bandwidth at 5, 10, and 20Hz. For each of these, T=1, dt=0.001, and rms=0.5

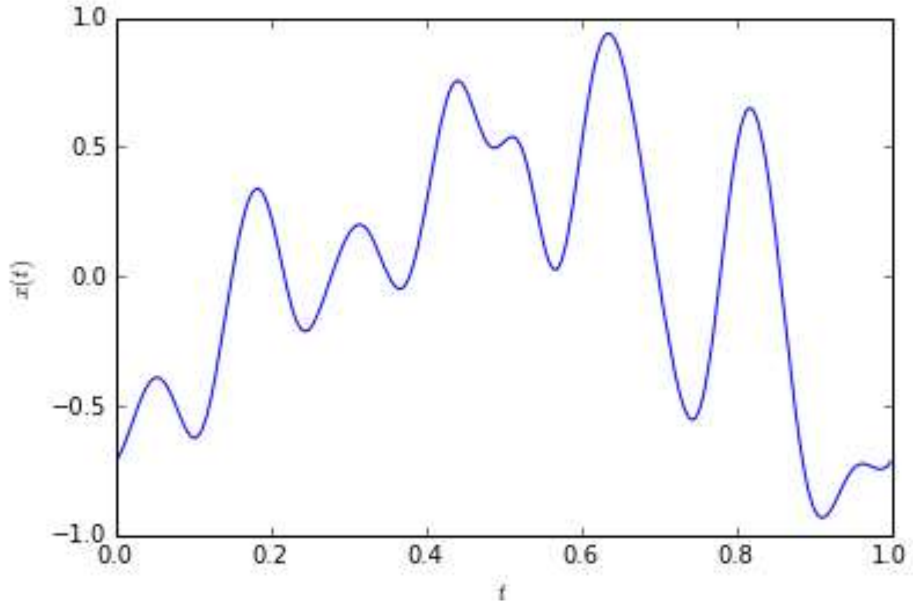


Figure 6. Plot of x(t) with frequency limit 5Hz

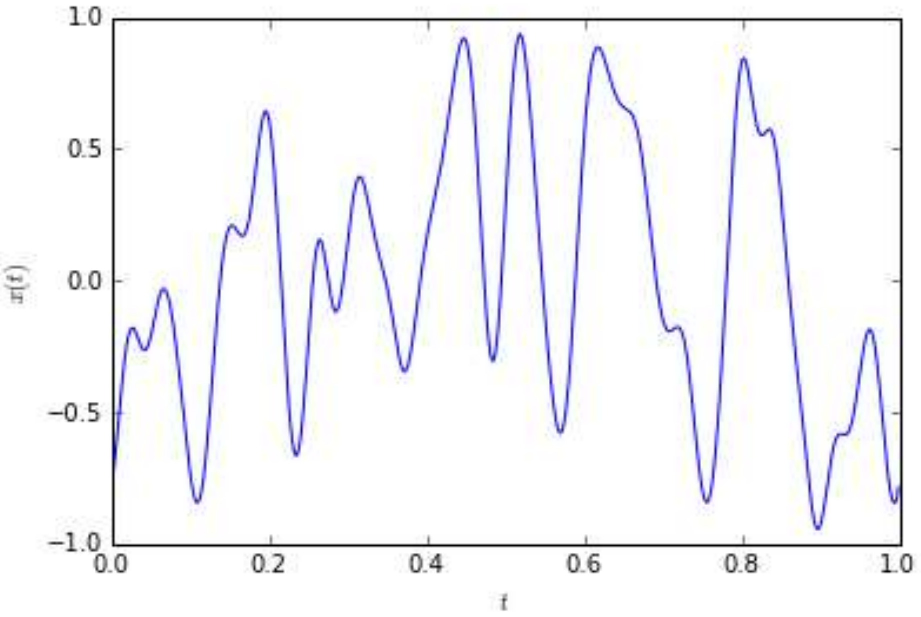


Figure 7. Plot of x(t) with frequency limit 10Hz

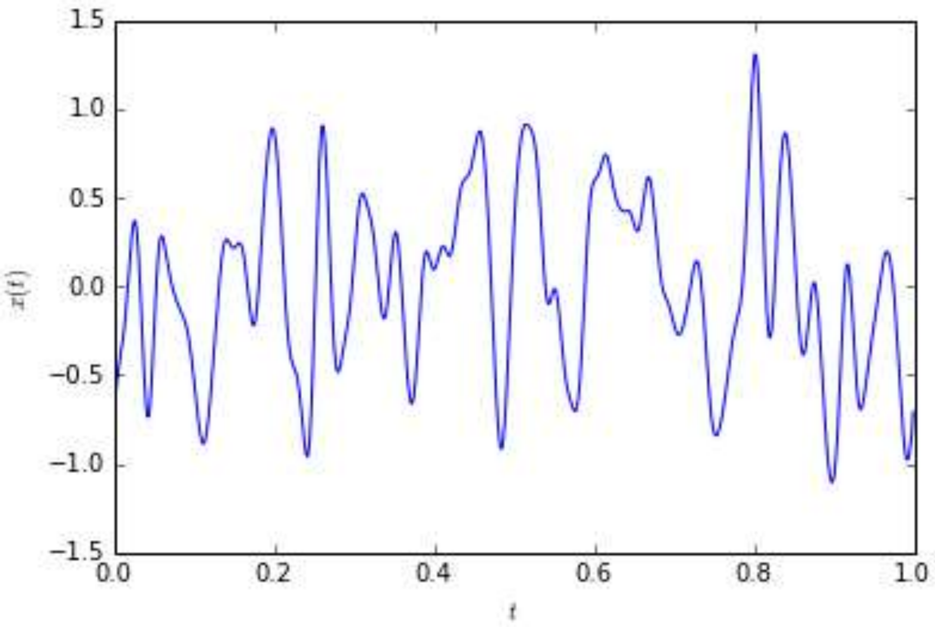


Figure 8. Plot of x(t) with frequency limit 20Hz

b) Plot the average |X(ω)|

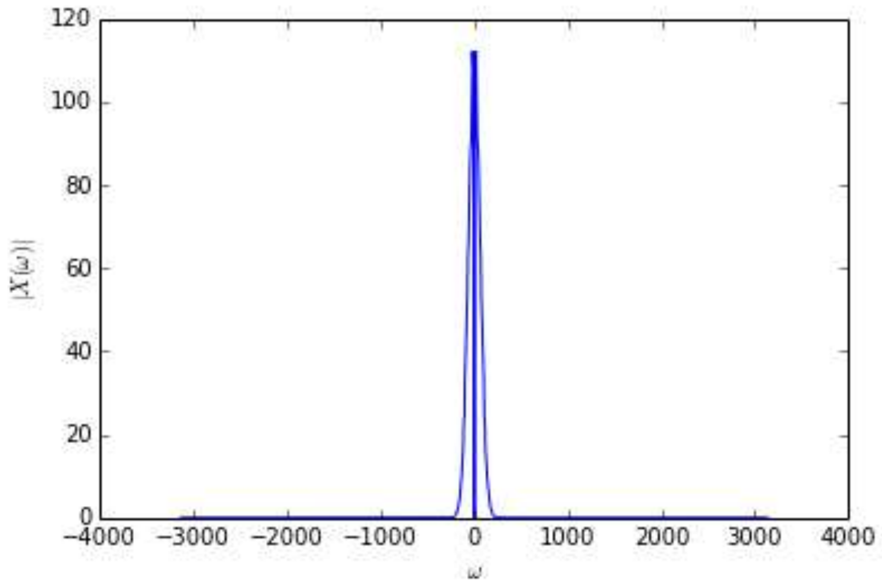


Figure 9. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz

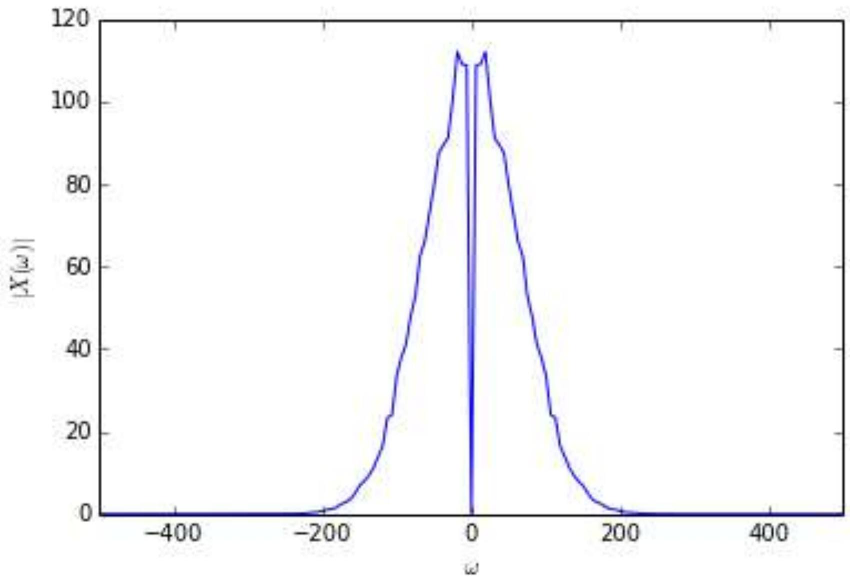


Figure 10. Plot of the Average |X(ω)| over 100 signals with limit = 10Hz (blown-up version)

In this case since the limit was 10Hz, the value of b = 10 in the Gaussian distribution N(μ=0, σ=exp(−ω2/(2*b2))). Thus it was expected that the bandwidth of the X(w) plot would be equal to 2*b2 which is equal to 200. Thus the result obtained in figure 10 complies well with what was expected.

2. Simulating a Spiking Neuron

A program to simulate a single Leaky-Integrate and Fire neuron. The core equation is dV/dt=1/τRC * (J−V) (to simplify life, this is normalized so that R=1, the resting voltage is 0 and the firing voltage is 1). This equation can be simulated numerically by taking small time steps (Euler's method). When the voltage reaches the threshold 1, the neuron will spike and then reset its voltage to 0 for the next τref amount of time. Also, if the voltage goes below zero at any time, reset it back to zero. For this part, τRC=0.02 and τref=0.002

Since we want to do inputs in terms of x, we need to do J=αe.x+Jbias. For this neuron, e was set to +1 to find α and Jbias such that the firing rate when x=0 is 40Hz and when x=1 it is 150Hz. To find these α and Jbias values, the approximation for the LIF neuron a(J)=1/(τref−τRC ln(1−1/J)) was used.

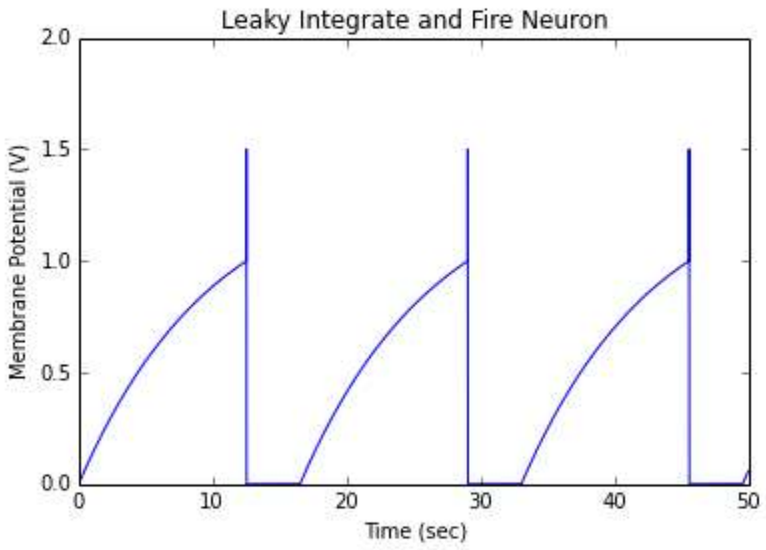


Figure 11. Demo of spike generation with spiky LIF neuron model

a) Spike output for a constant input of x=0 and x =1

For x = 0 , number of Spikes = 40

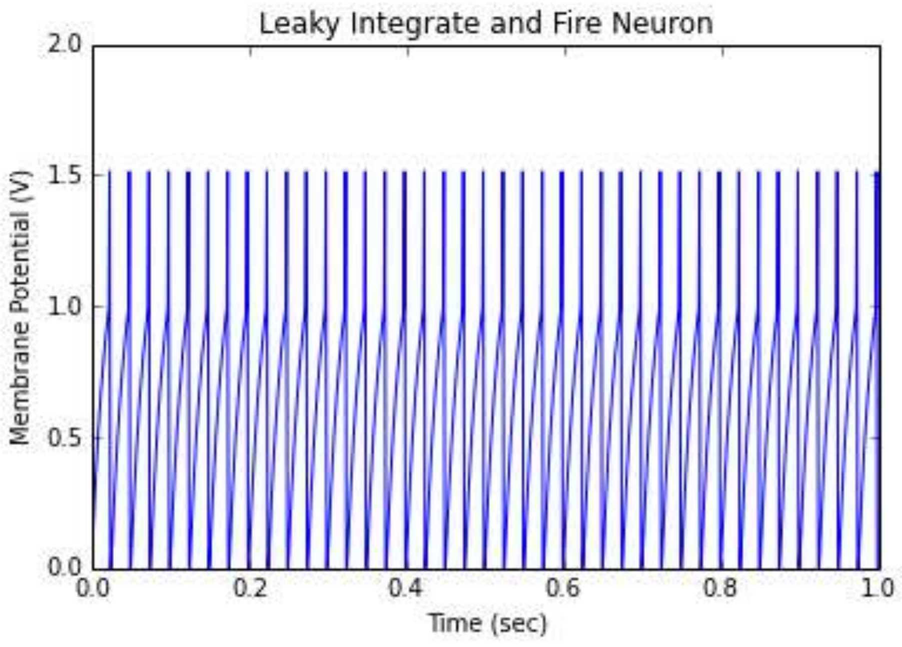


Figure 12. Voltage output for a constant input of x=0 over 1 second

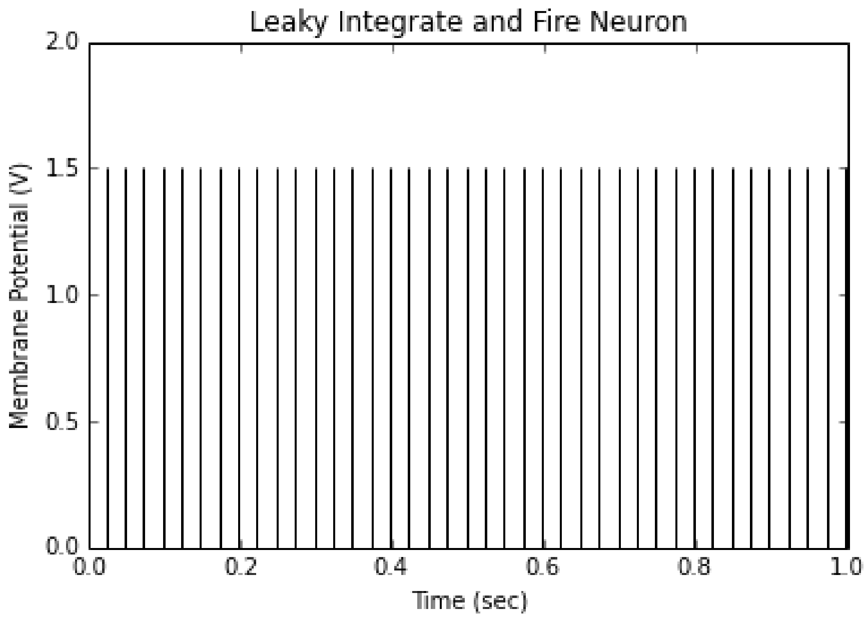


Figure 13. Spike output for a constant input of x=0 over 1 second

For x=1, number of Spikes = 143

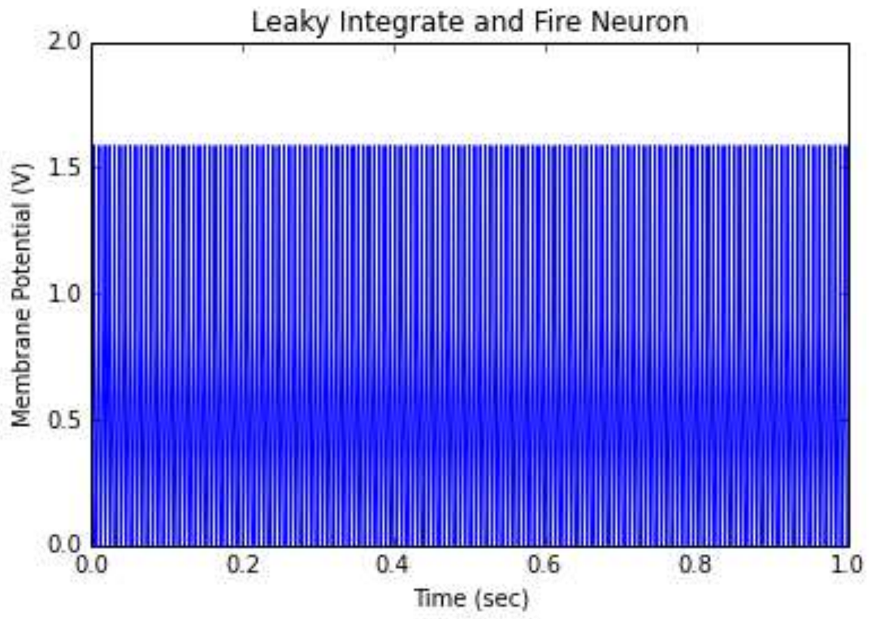


Figure 14. Voltage output for a constant input of x=1 over 1 second

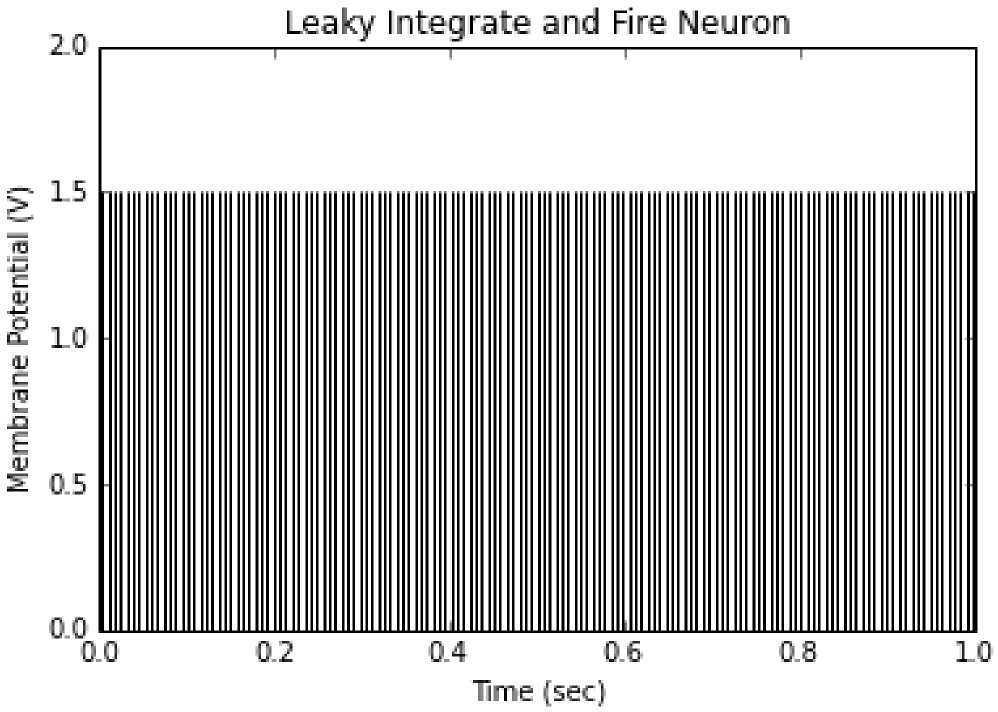


Figure 15. Spike output for a constant input of x=1 over 1 second

b) Discussion on the observed number of spikes

Does the observed number of spikes in the previous part match the expected number of spikes for x=0 and x=1? Why or why not? What aspects of the simulation would affect this accuracy?

No the number of spikes produced does not match the expected number of spikes for x =0 and x = 1. When simulating the core equation for the calculation of the membrane potential and generation of spikes, Euler's method was used. This means that the interval of 1 second was divided into discrete time steps in order to compute the membrane potential a t each time step (differential equation was converted to a difference equation) . However, in reality the current is a continuous function input coming into the neurons and the voltage at time t depends on all the past current. This means that in reality the number of spikes produced over a period of 1 second will be greater in both the cases (x=0 and x=1). Therefore the number of spikes do not match the expected number. The main reason for this is the discrete time step calculation and the accuracy of this calculation is highly dependant on the discrete time step size. [Refer to Method 1 under section 2e) for details].

For x = 0, the number of spikes observed was equal to 40 whereas for x =1, the number of spines observed was 143. The membrane potential at each time step was calculated using the following core equation for an LIF neuron as mentioned below:

V[i] = V[i-1] + ((J*R - V[i-1]) / self.tau_rc) * dt

However, the current 'J' was calculated using the equation i.e J = gain*x.e + Jbias where gain and Jbias were computes using an approximated equation of the LIF neuron. It can be seen that the voltage at any time step is dependent on the value of the voltage at the previous time step and the current J (R, tau_rc and dt are constants).

According to this model, it was expected that the number of spikes for x=1 should be greater that that for x=0, since for x = 0, J = Jbias and thus as the value of the current is small, it takes more time steps for the voltage to reach the threshold voltage where the spike occurs.

However, for x=1, J = gain*(1) + Jbias. This implies a higher value of current which means that the voltage would now reach the threshold in fewer time steps as compared to the x=0 case which means that more number of spikes would be produced. Thus, it was observed that the number of spikes obtained for x = 1 was 143 which is larger than 40 spikes produced for x = 0.

Quantitatively,

Jbias = 1.46
gain = 3.34

J (x=0) = 1.46
J (X=1) = gain + Jbias = 4.8

From the values of the current, it can be intuitively predicted that the number of spikes in x = 1 case would be almost 3 to 4 times greater than the number of spikes obtained for x=0. The results obtained show that x = 1 produces around 3.5 times more spikes than x = 0.

c) Spike output for x(t) with T = 1

Plot of the spike output for x(t) generated using the function from part 1.1.

Used T=1, dt=0.001, rms=0.5, and limit=30. Overlaid x(t) on this plot

Number of spikes = 46

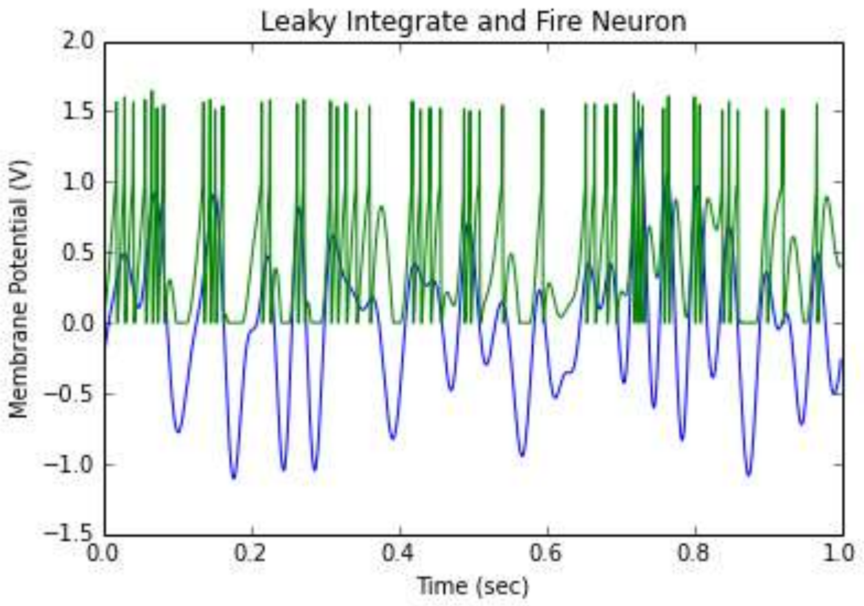


Figure 16. Voltage output for x(t) over T = 1 second with dt = 0.001

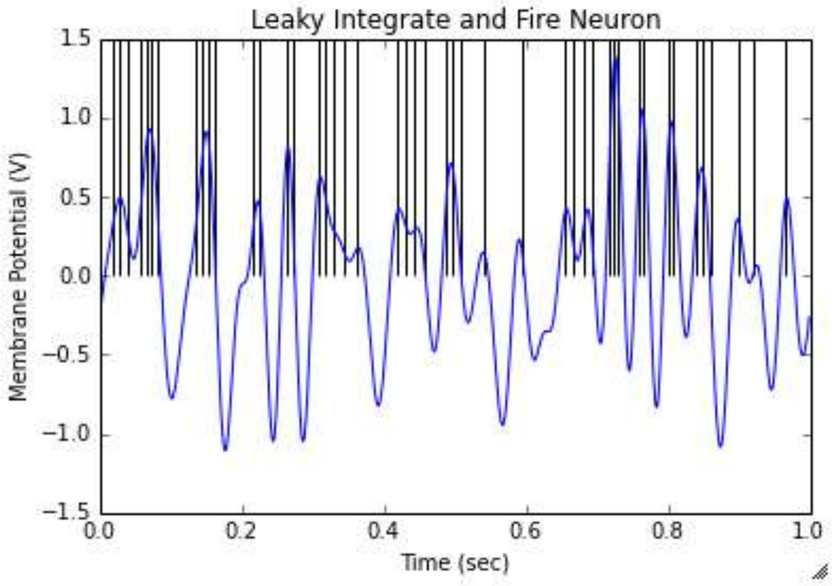


Figure 17. Spike output for x(t) over T = 1 second with dt = 0.001

d) Spike output for x(t) with T = 0.2

Number of Spikes = 11

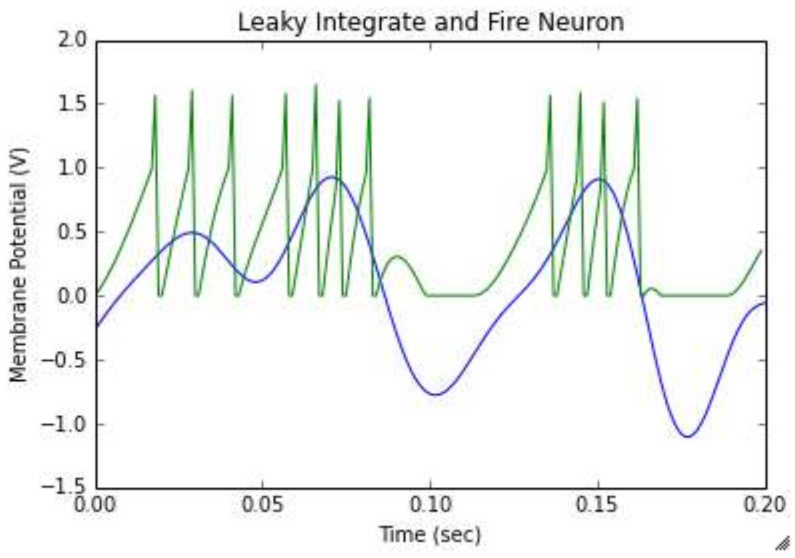


Figure 18. Voltage output for x(t) over T = 0.2 seconds with dt = 0.001

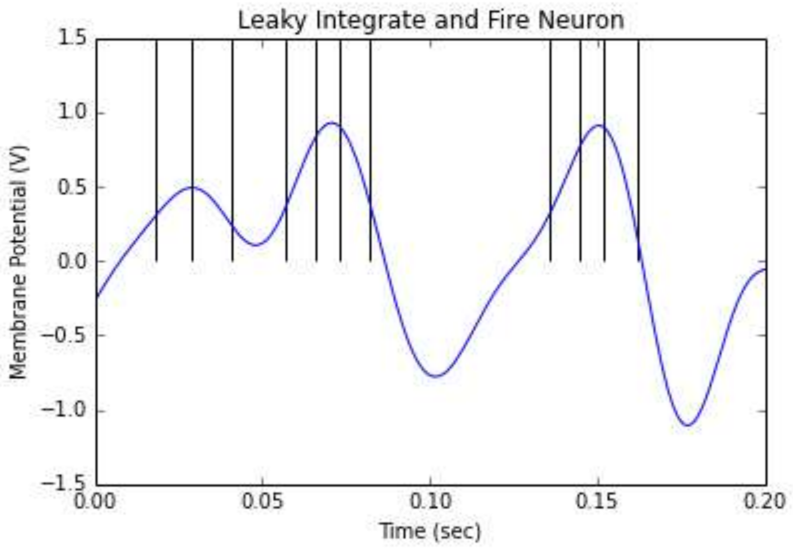


Figure 19. Spike output for x(t) over T = 0.2 seconds with dt = 0.001

e) Bonus Question

How could you improve this simulation (in terms of how closely the model matches actual equation) without significantly increasing the computation time? 0.5 marks for having a good idea, and up to 1 marks for actually implementing it and showing that it works.

Method 1
For this simulation, Euler's method was used in order to calculate the membrane potential at each time step, since the interval of 1 second was divided into discrete time steps. Thus the following differential equation is converted into a difference equation.

$$dV/dt = (J-V) / \tau RC$$

However, in reality the current is a continuous function input coming into the neurons. This mean s that in reality the number of spikes produced over a period of 1 second will be slightly greater in both the cases (x=0 and x=1). Thus as the time step dt is decreased (upto a certain minimum valu e), the number of spikes produced should increase thus improving the simulation to match the real ity.

Method 2
The problem with Method 1 is that the computation time increases as the time step dt is decreased . Hence this method is not scalable.

Method 2 is to solve the following equation to compute membrane potential as a function of time:

$$dV/dt = (J-V) / \tau RC$$

On evaluating the above integral and assuming that J is constant, following equation is obtained (Appendix B of the book 'Neural Engineering', P-304, 305) :

$$V(t) = JR (1 - \exp^{-t/\tau RC})$$

The above equation is a more accurate equation for the calculation of membrane potential at any given time and it also takes the past current input into account. Since now we know that the soluti on to the equation is exponential, using exponential approximation instead of a linear approximati on should give us better results.

This is justified by the following data found experimentally:

- With Method 2, for dt=0.001 , number of spikes =209 over a period of 1 seconds as shown in Figure 20
- With Euler's Method, for dt =0.001, number of spikes = 46 over a period of 1 second as shown in Figure 17

Thus, even while keeping the time step value dt same as before, the number of spikes was found to increase using the exponential approximation. This shows that the method 2 is more accurate.

Number of Spikes = 209

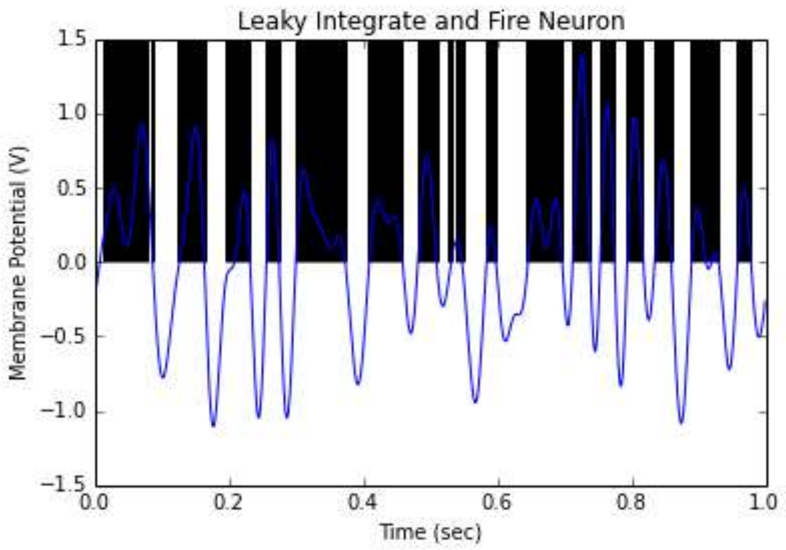


Figure 20. Output for X(t) over T = 0.2 seconds with Method 2

3 Simulating Two Spiking Neurons

In this part, a program was written that simulates two neurons. The two neurons have exactly the same parameters, except for one of them e=1 and for the other e=−1. Other than that, exactly the same settings as in section 2 were used.

a) Plot of x(t) and the spiking output for x(t)=0

Number of spikes = 40 for both cases: encoder = 1, -1

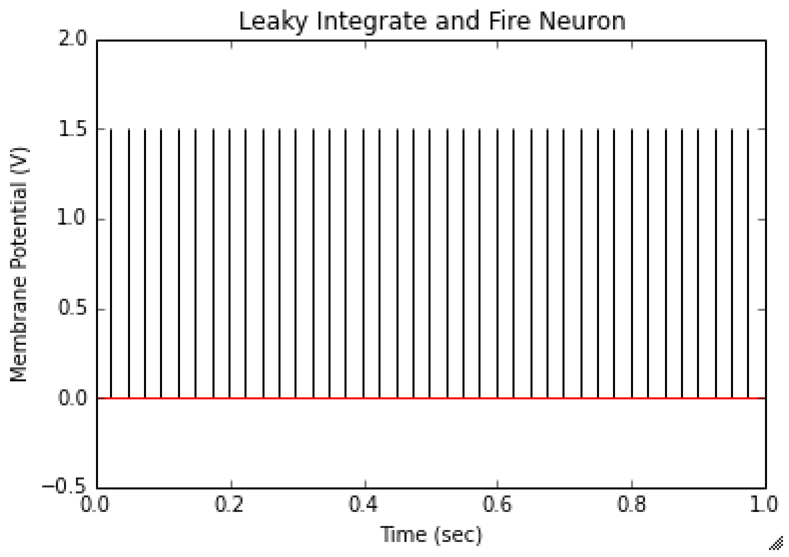


Figure 21. x(t) and the spiking output for x(t)=0 and encoder = 1

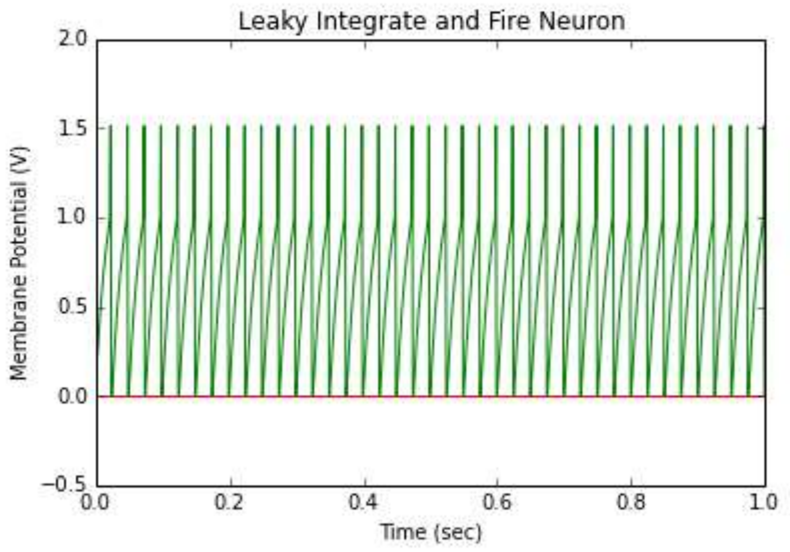


Figure 22. x(t) and the Voltage output for x(t)=0 and encoder = 1

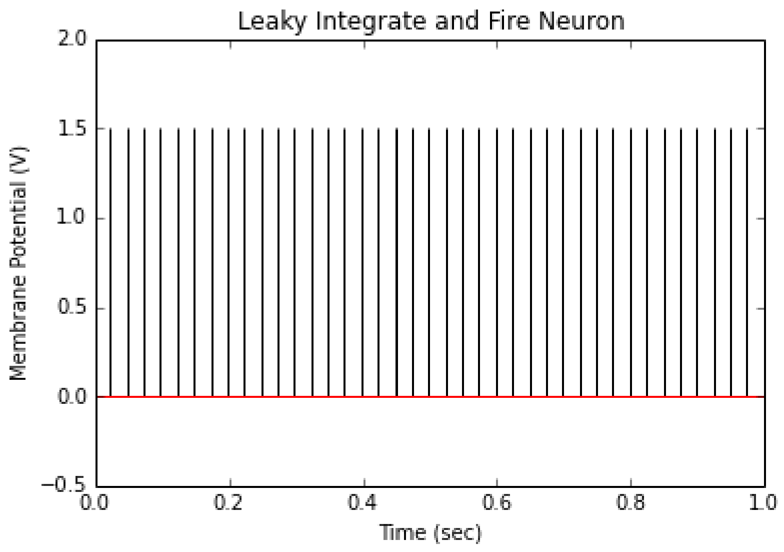


Figure 23. x(t) and the spiking output for x(t)=0 and encoder = -1

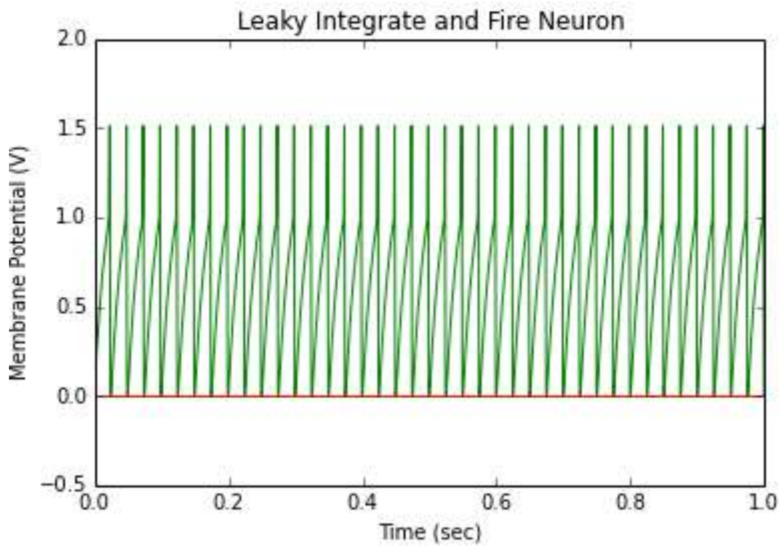


Figure 24. x(t) and the Voltage output for x(t)=0 and encoder = -1

Number of spikes in both cases shown in Figure16 and Figure17 = 40
The graphs are same despite the different encoders (preferred directions). This is because since x = 0 , J = Jbias which is the same in both cases.

b) Plot of x(t) and the spiking output for x(t)=1

Number of spikes = 143

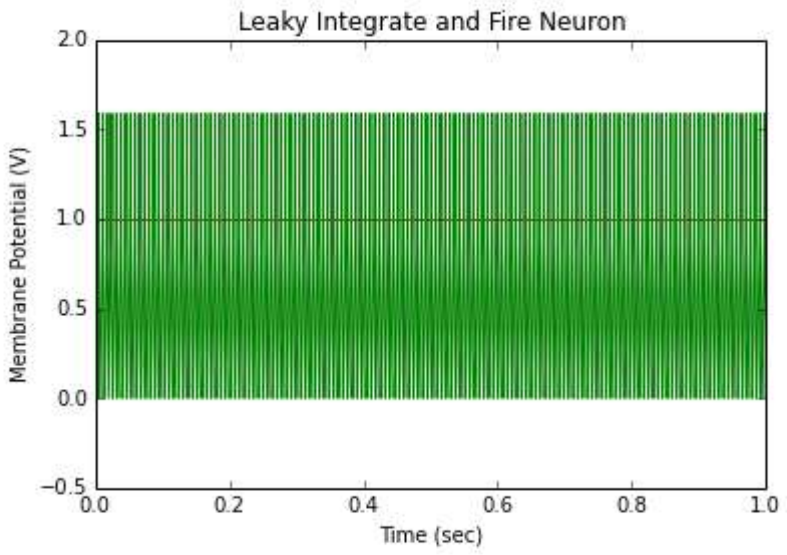


Figure 25. x(t) and the Voltage output for x(t)=1 and encoder = 1

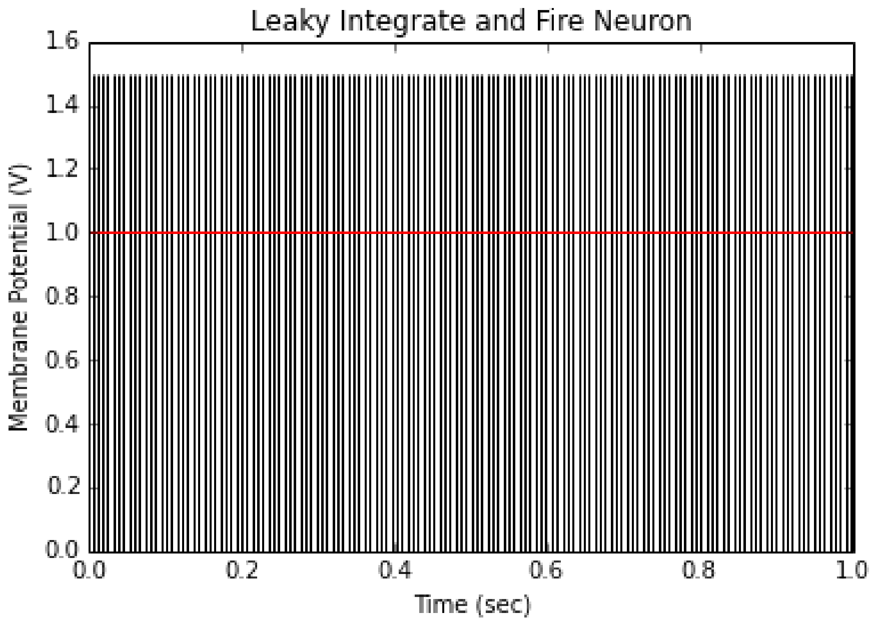


Figure 26. x(t) and the Voltage output for x(t)=1 and encoder = 1

Number of spikes = 0

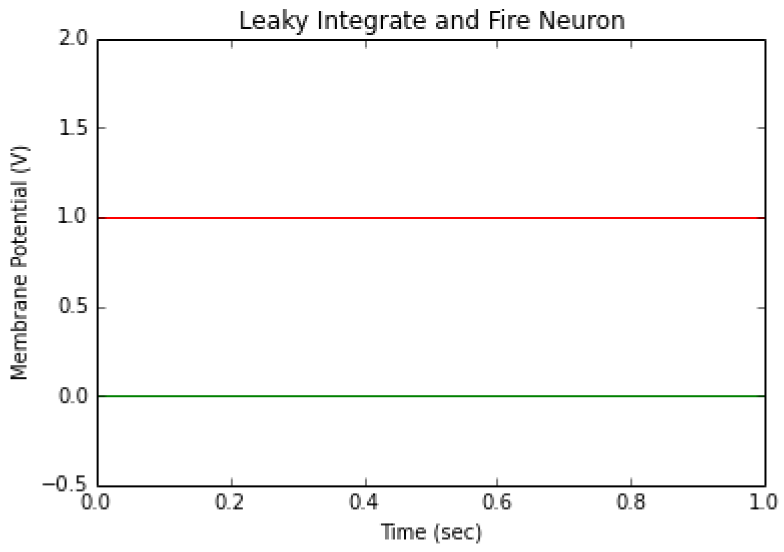


Figure 27. x(t) [red] and the voltage output for x(t)=1 and encoder = -1

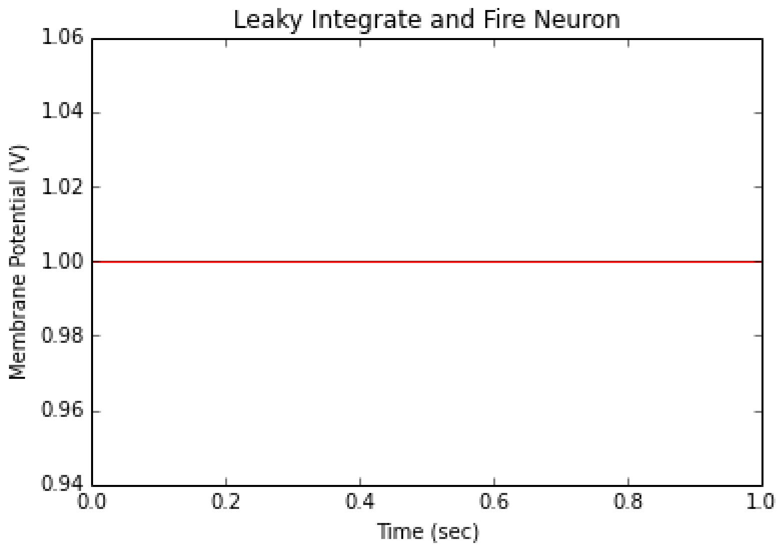


Figure 28. x(t) [red] and the spiking output for x(t)=1 and encoder = -1

c) Plot of x(t) and the spiking output for x(t)=12sin(10πt)

Plotting the spikes generated by both the neurons with e = 1 and e = -1 on the same plot. In the plots, the spike train for the neuron with e = 1 is plotted as it is but the spike train for the neuron with e=-1 is shifted down by 1.5 so that the two spike trains do not overlap.

Number of Spikes = 40 each for both neurons with encoder = 1, -1

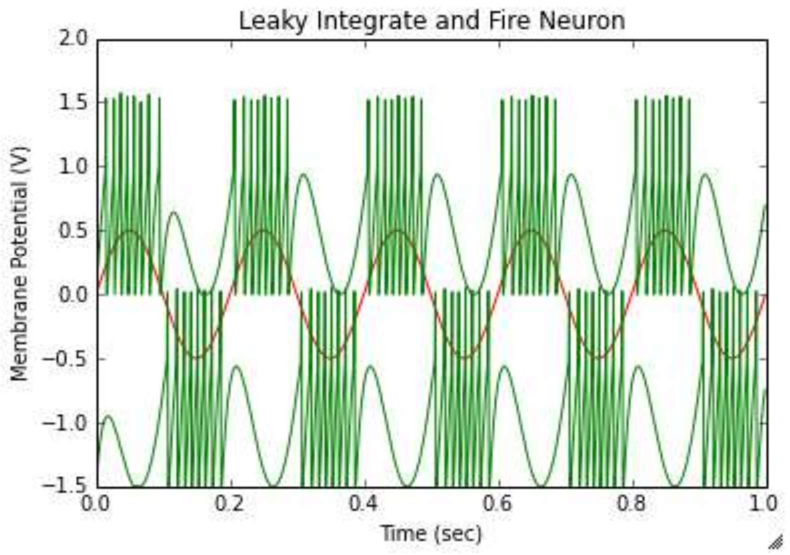


Figure 29. Voltage Output x(t)=12sin(10πt) for e=-1 shifted down by 1.5

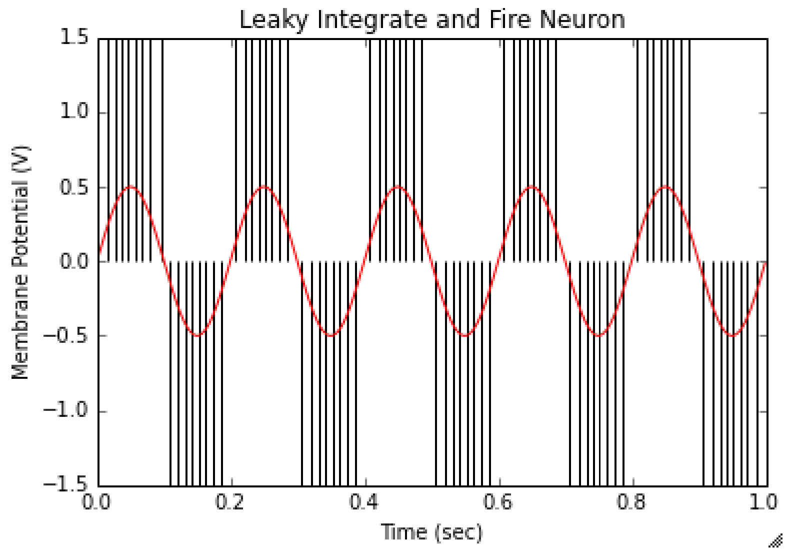


Figure 30. Spike Output x(t)=12sin(10πt) for e=-1 shifted down by 1.5

d) Plot of x(t) and the spiking output for a random signal

Plotting the spikes generated by both the neurons with e = 1 and e = -1 on the same plot.. In the plots, the spike train for the neuron with e = 1 is plotted as it is but the spike train for the neuron with e=-1 is shifted down by 1.5 so that the two spike trains do not overlap.

Number of spikes = 94 for neuron with e = 1

Number of spikes = 90 for neuron with e = -1

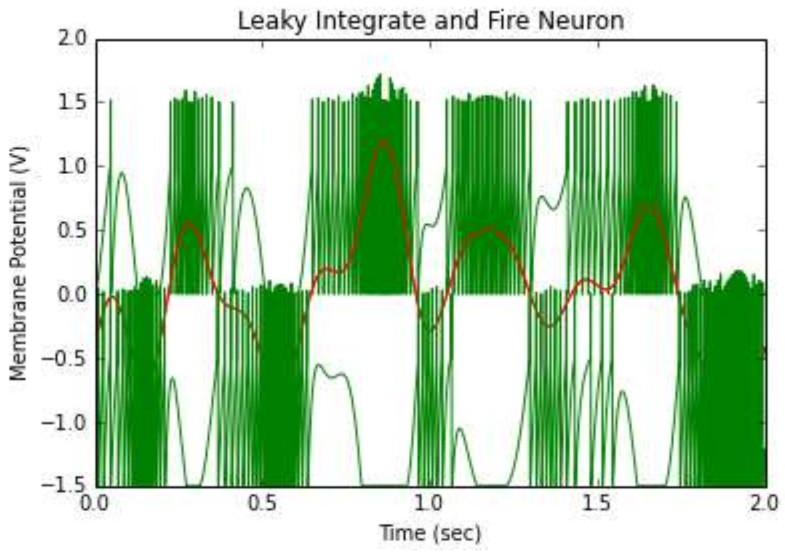


Figure 31. x(t) and the voltage output for a random signal (neuron with e=-1 shifted down)

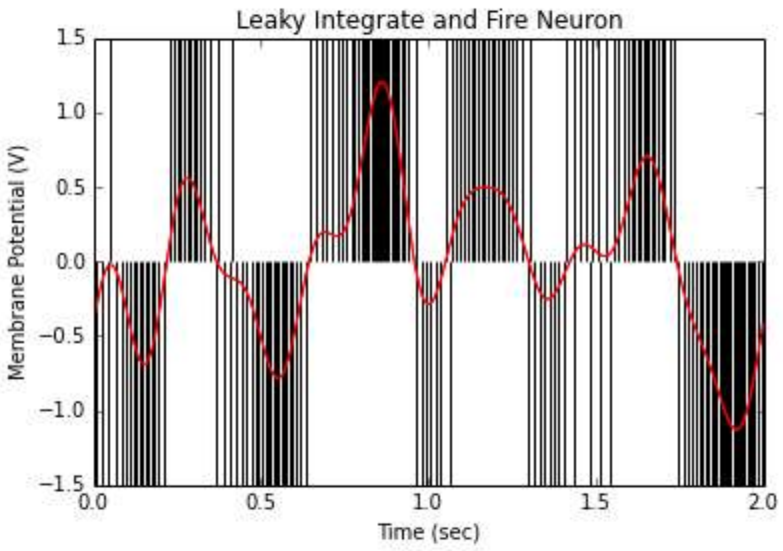


Figure 32. x(t) and the Spike output for a random signal (neuron with e=-1 shifted down)

4. Computing an Optimal Filter

a) Document the code

b) Time and frequency plots for the optimal filter

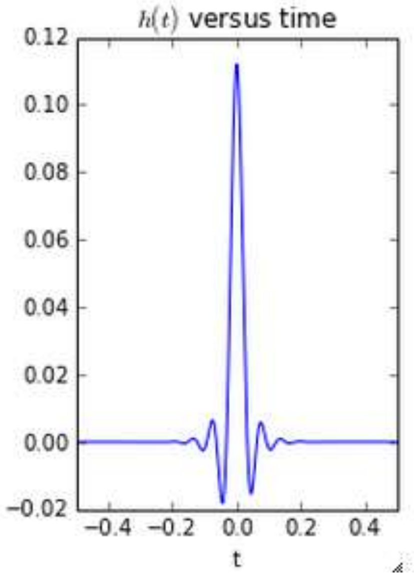


Figure 33. Time Plot of an optimal Linear Decoder for a pair of LIF neurons

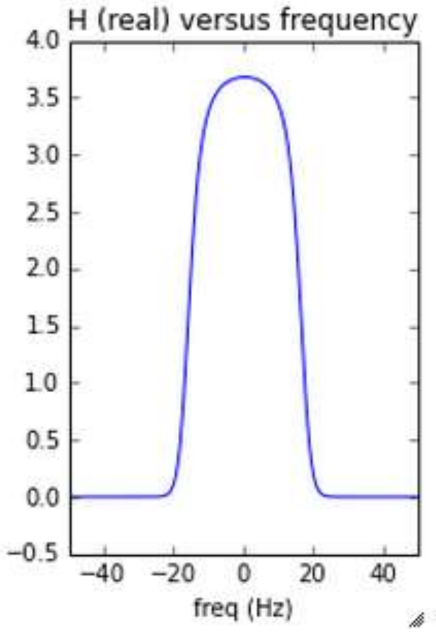


Figure 34. Frequency Plot of an optimal Linear Decoder for a pair of LIF neurons

c) **x(t) signal, the spikes, and the decoded xhat(t) value**

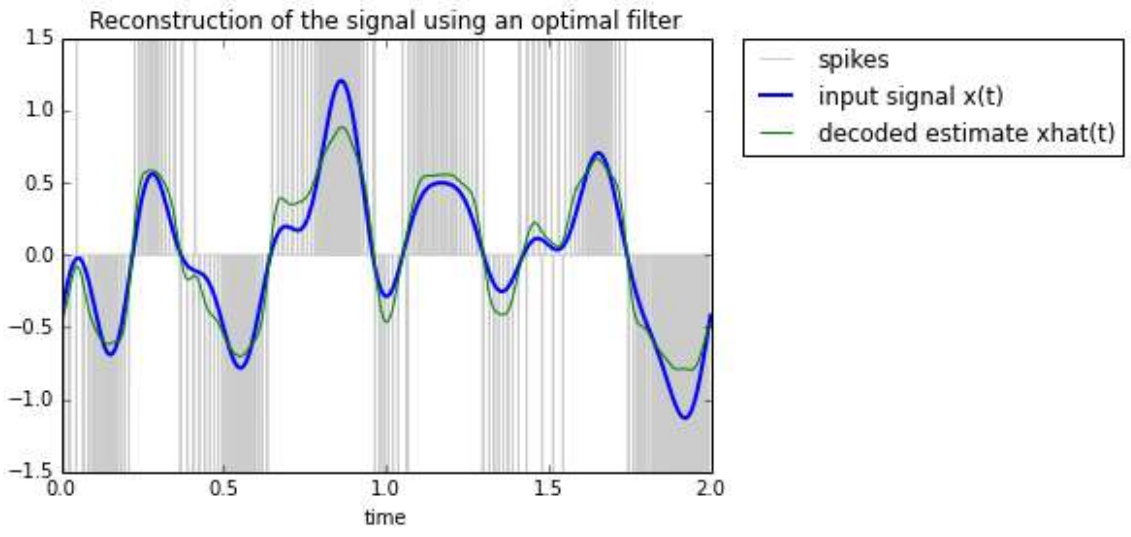


Figure 35. x(t), xhat(t) and the spikes

d) Power Spectrum Plots

Plots of |X(ω)| power spectrum, |R(ω)| spike response spectrum, and the |X^(ω)| power spectrum.

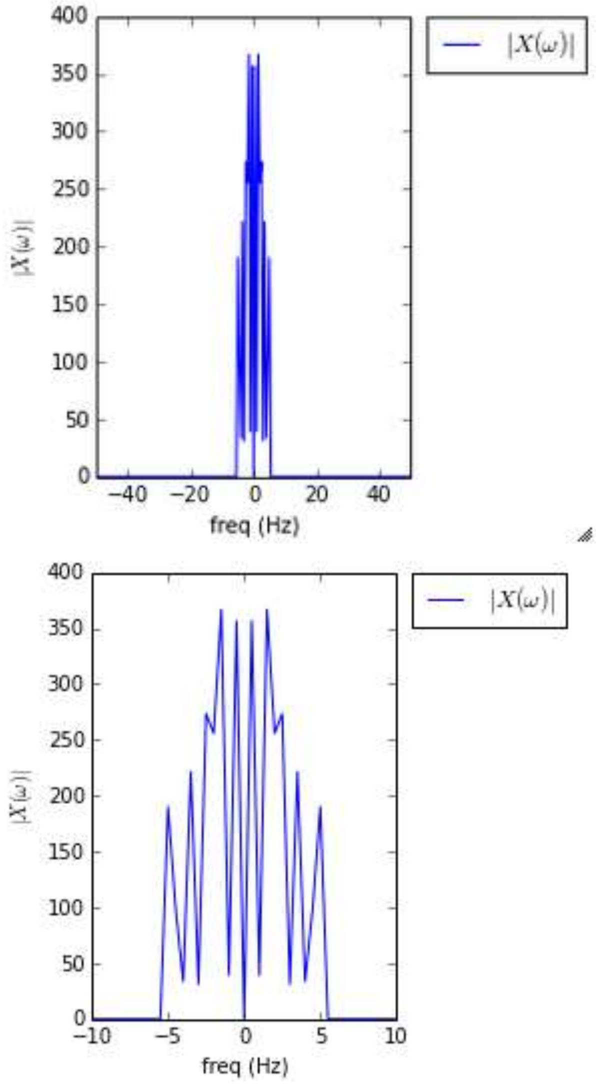


Figure 36. |X(ω)| power spectrum

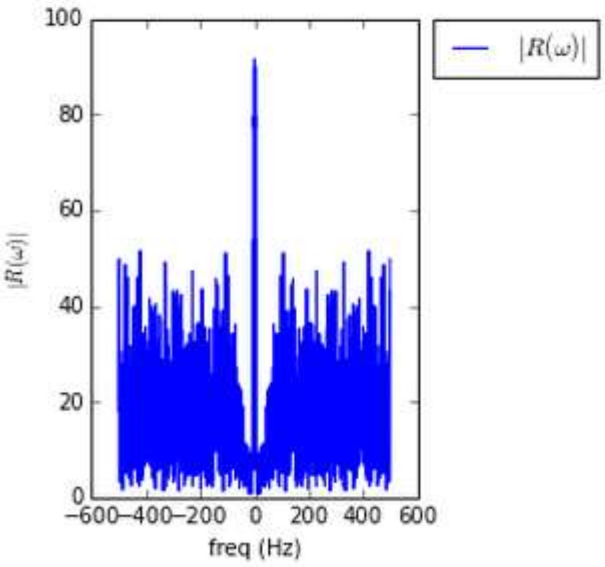
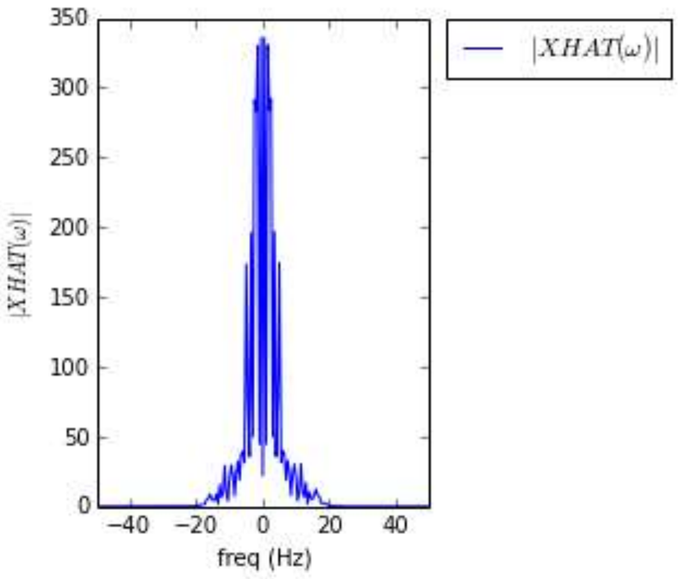


Figure 37. |R(ω)| power spectrum



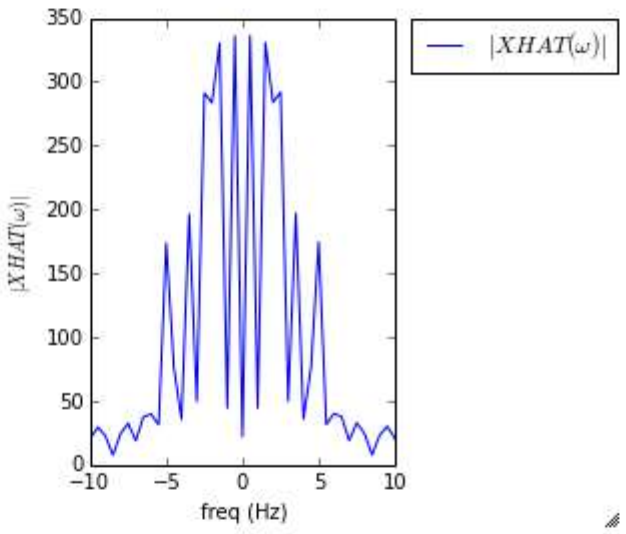


Figure 38. |XHAT(ω)| power spectrum

<u>Relation to the optimal Filter:</u>

H(ω) is the Fourier transform of the optimal filter h(t)

H(ω)= [(X(ω)R(ω)*) * W(ω)] / [|R(ω)|^2 * W(ω)]

XHAT(ω)=R(ω)H(ω)

It can be seen that encoding the original signal using LIF neurons introduces spurious power especially at high frequencies as shown in figure 38 - R(w) power spectrum. However, the power at frequencies that are actually in the signal X(w) are also well preserved. Thus one of the main functions of the decoding filter is to remove these spurious high frequency components without altering the spectrum, i.e basically acting as a low pass filter. It can be seen from figure 35 that the optimal filter does resemble a low pass filter and figure 34 shows that the filter is localized in time. The cutoff frequency of the optimal filter is 20Hz (figure 35) and therefore it can be seen in figure 38 that XHAT(w) also has frequency components between -20 to 20 Hz.

Note: The width of the optimal filter in the time domain depends on the kind of signal that is used to find it. Example: for high frequency signals it will be thinner in the time domain, thus increasing the cutoff frequency of the low pass filter in the frequency domain. On the other hand, for a low frequency signal, it will be wider in the time domain, thus decreasing the cutoff frequency in the frequency domain.

e) h(t) time plots for the optimal filter for different limit values

The limit refers to the maximum frequency of the signal in Hz. We know that the width of the optimal filter depends on the kind of signal that is used to compute it. As the frequency of the signal increases, it is expected that h(t) time plot for the optimal filter would be thinner and taller. At the same time the plot in the frequency domain would be wider and shorter since the cutoff frequency of the optimal filter in the frequency domain would increase.

This is confirmed by the results obtained in Figure 40, 41, 42. It is clear that as the limit frequency was increased, the width of the optimal filter decreased in time domain, but the cutoff frequency increases in the frequency domain. This happens because if the original signal x(t) has higher frequency components, then the optimal filter needs to be wider (higher bandwidth) in the frequency domain so that it can preserve the power at all the frequencies that are included in the original signal x(t).

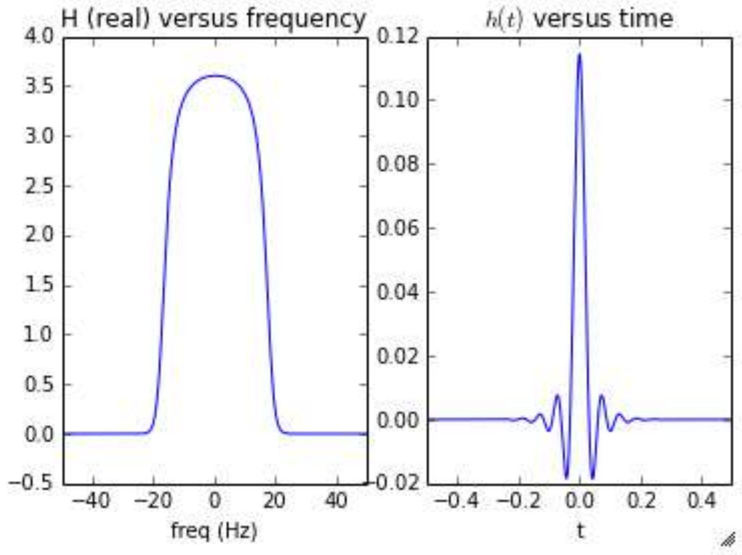


Figure 39. Frequency and time plot of optimal filter with limit = 2Hz

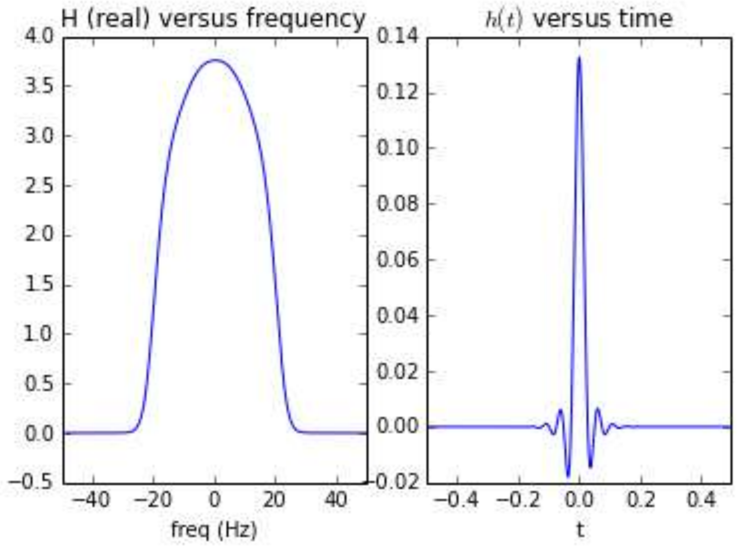


Figure 40.Frequency and time plot of optimal filter with limit = 10Hz

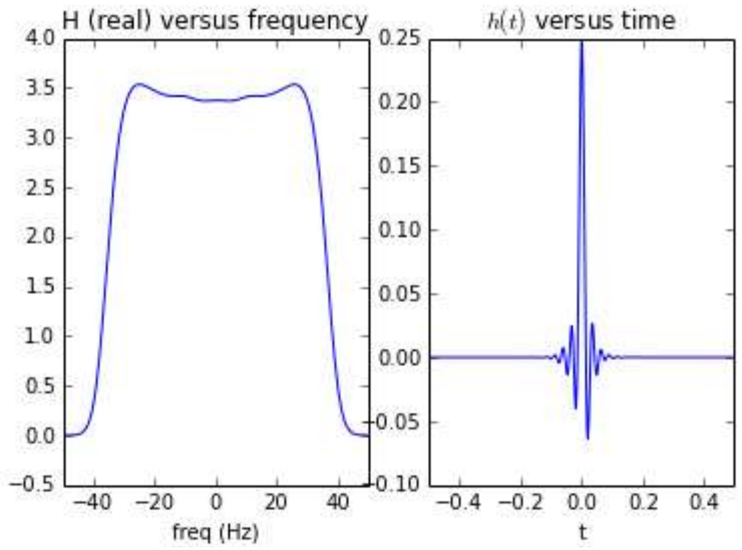


Figure 41. Frequency and time plot of optimal filter with limit = 30Hz

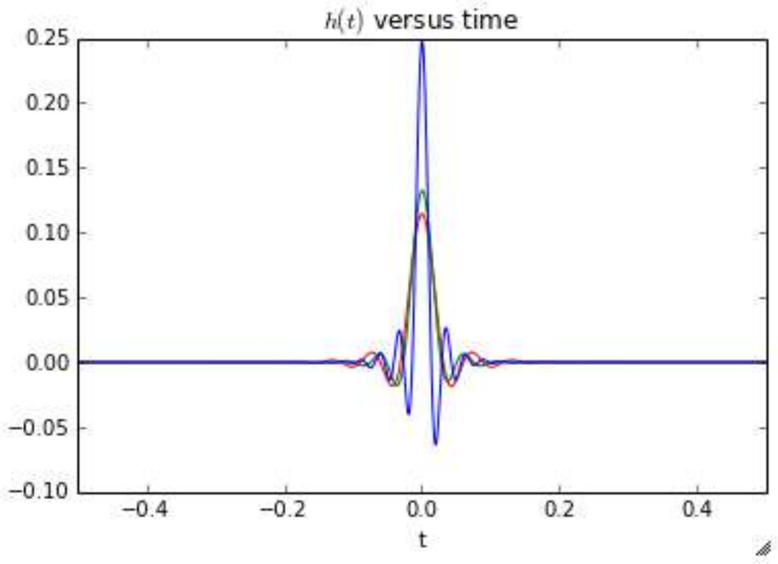


Figure 42. Plot of optimal filter h(t) with limit = 2HZ (red), 10Hz (green), 30Hz (blue)

f) h(t) time and |H(ω)| power spectrum plots for different 'T' values

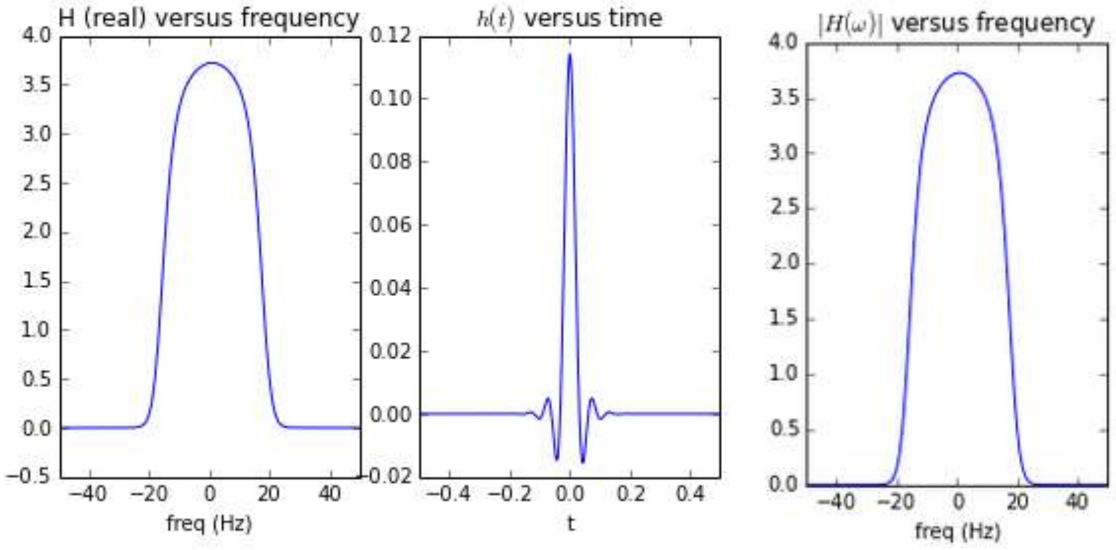


Figure 43. Plots of optimal filter and |H(w)| power spectrum for T = 1 second

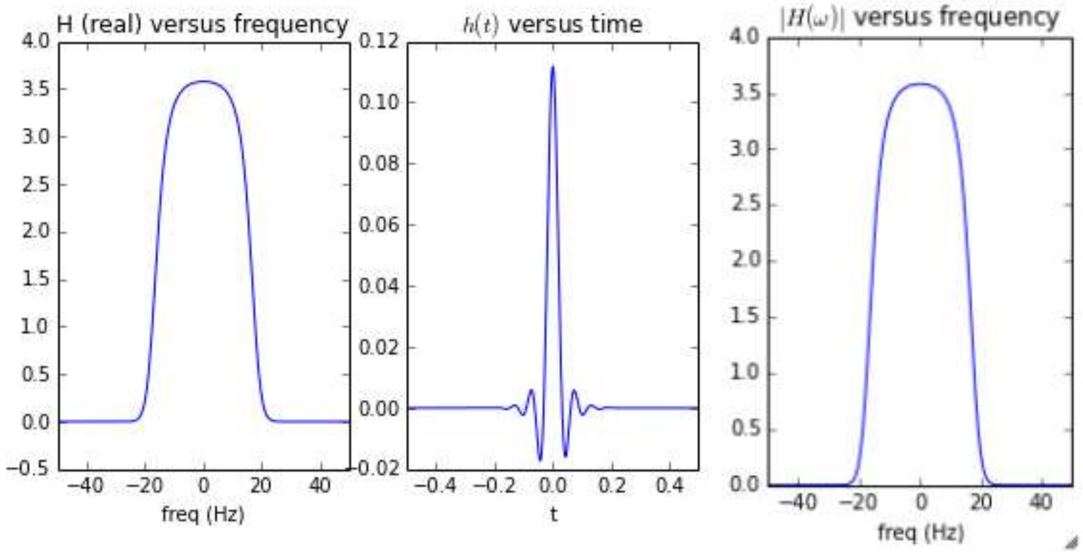


Figure 44. Plots of optimal filter and |H(w)| power spectrum for T = 4 seconds

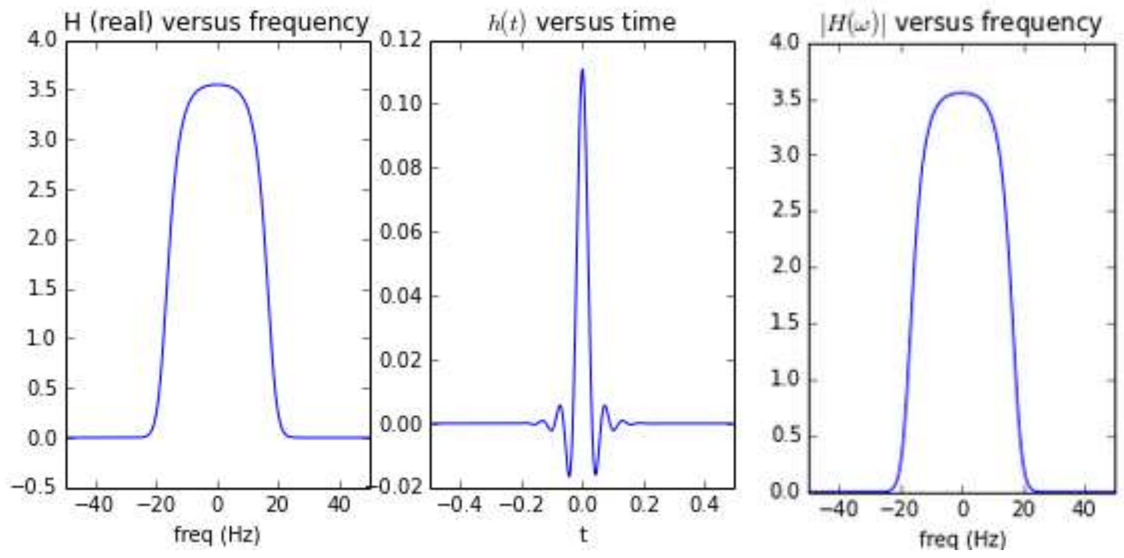


Figure 45. Plots of optimal filter and |H(w)| power spectrum for T = 10 seconds

From Figures 43, 44, 45, it can be seen that changing the duration of the signal does not have much impact on the optimal filter as well as on the |H(w)| power spectrum. This is confirmed on taking a closer look as shown in Figure 46 and 47. The optimal filter h(t) remains the same since the frequency components of the three signals are the same due to having the same limit value. The H(w) power spectrum also has almost the same cutoff frequencies for the same reason. However, the H(w) spectrum flattens and becomes more square as the duration of the signal increases, since longer run times provide better sampling.

Since we are using a Gaussian Window, we know that the Fourier transform of a Gaussian is another Gaussian which itself is a well localized low pass filter. Thus the continuous Gaussian filtering allows us to make better use of the data available providing a better estimation because the estimation process itself does not introduce any sources of error. Thus it is not extremely necessary to average over longer time periods to have accurate results.

On the other hand, if we were using say a box window, the Fourier transform of that would be a sync function which contains ringing out to high frequencies. Thus any loss of these frequencies would likely be reflected in our estimate introducing errors. Averaging over longer runtimes would help to reduce those errors in this case.

This is the advantage of using a Gaussian Window over a Box Window.

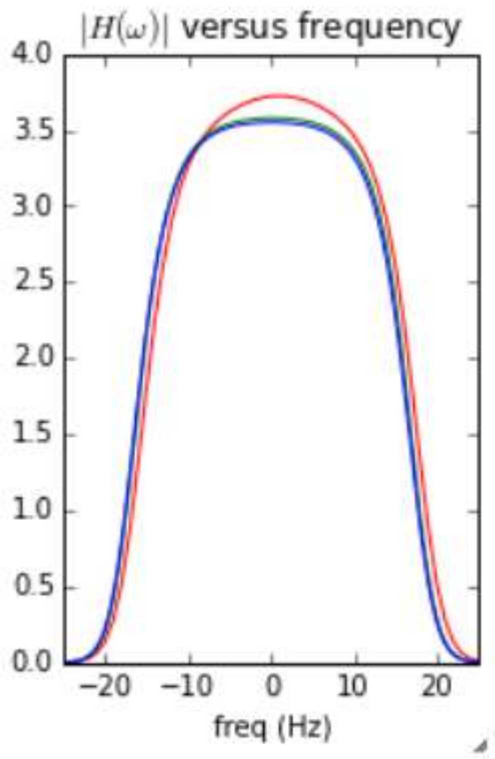


Figure 46. Power spectrum plots for T = 1 (red), T = 4 (green), T = 10 (blue)

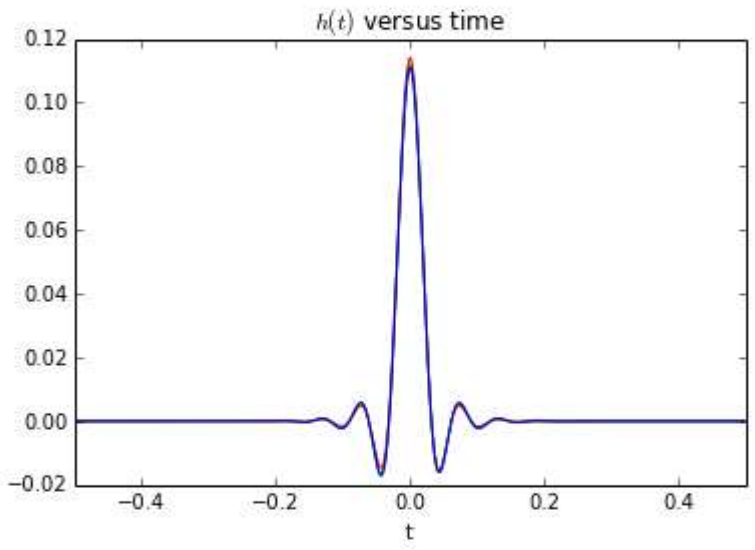


Figure 47. Optimal filter in time domain for T = 1 (red), T = 4 (green), T = 30 (blue)

5. Using Post-Synaptic Currents as a Filter

Decoding with PSCs instead of using optimal filters results in a small reduction in information transmission, but a large gain in neural plausibility. However, increasing the number of neurons can make up for the reduction in the coding accuracy.

a) Plot of normalized h(t) for n=0, 1, and 2

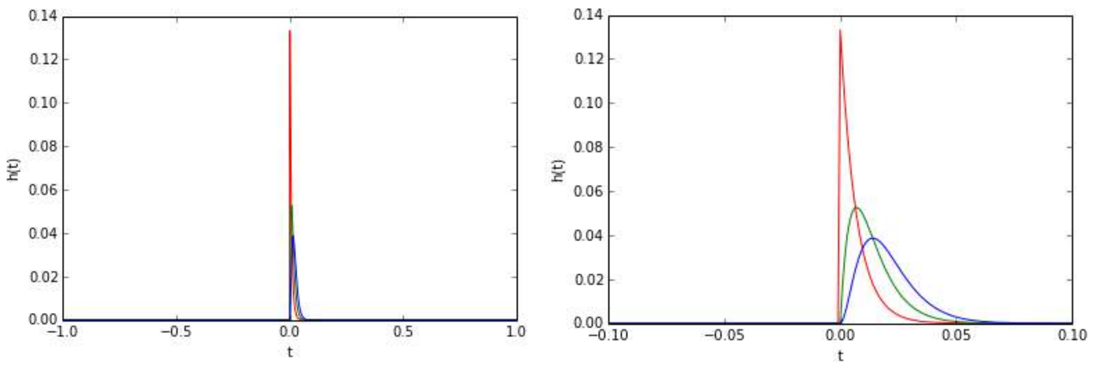


Figure 48. Plot of h(t) for n = 0 (red), n = 1 (green), n = 2 (blue) with tau = 0.007s

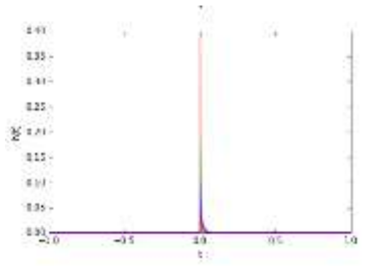
On increasing n, the peak value of the filter decreases and it is pushed forward in time.

1. Peak Value - Indicates the relative weightage of earlier versus later aspects of the response.
1. Pushed forward in time - shows how fast is the information about the initial state lost over a period of time

For a high peak value, earlier as aspects of the response carry higher weightage as compared to the later aspects of the response. Thus as n increases, the weightage gets more and more evenly distributed over time. This results in greater averaging of the signal over time leading to a smoother signal.

Moreover since there is a push forward in time, this suggests that there will be some time delay in filtering. Thus the smoothness of the filtered signal comes at the expense of a time delay.

b) Plot of normalized h(t) for τ=0.002, 0.005, 0.01, and 0.02 seconds



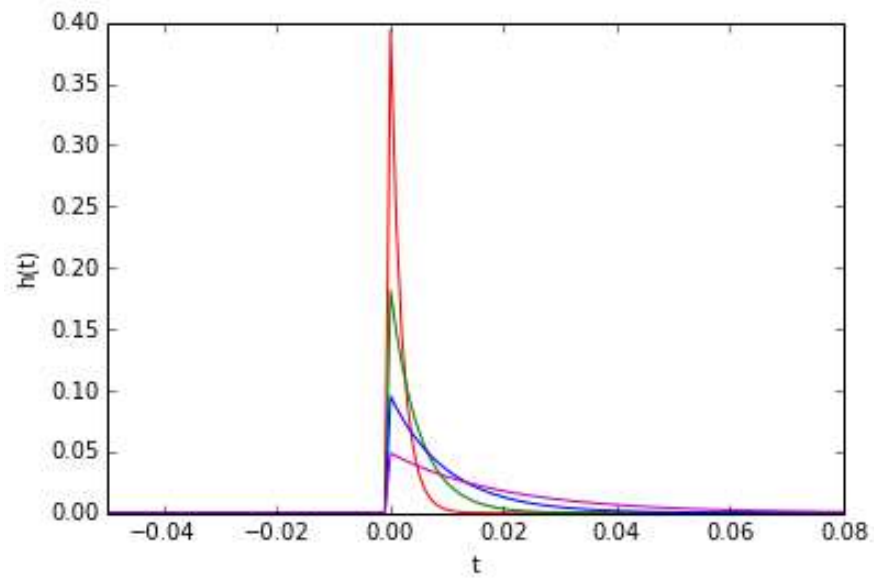


Figure 49. Plot of h(t) for tau = 0.002 (red), tau = 0.005 (green), tau = 0.01 (blue) with tau = 0.02 (majenta).

On increasing tau, the peak value of the filter as well as the slope of decay of the filter (decay rate) decreases.

The peak value - Indicates the relative weightage of earlier versus later aspects of the response.

Decay rate - shows how fast is the information about the initial state lost over a period of time

For high peak value, earlier as aspects of the response carry higher weightage as compared to the later aspects of the response. Thus as tau increases, the weightage gets more and more evenly distributed over time. So as tau increases, xhat will have greater averaging of the signal over time leading to a smoother signal.

Lower decay rate means that the information about the initial state of the signal over time is not lost very quickly thus leading to a better approximation of xhat as tau increases. However this comes at the expense of some time delay in filtering.

c) Decode xhat(t) using spikes from LIF neurons

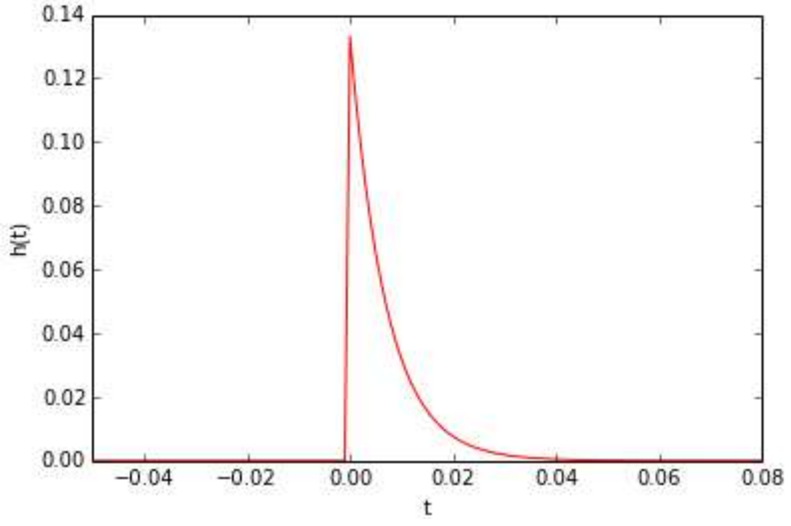


Figure 50. Time plot of the filter h(t)

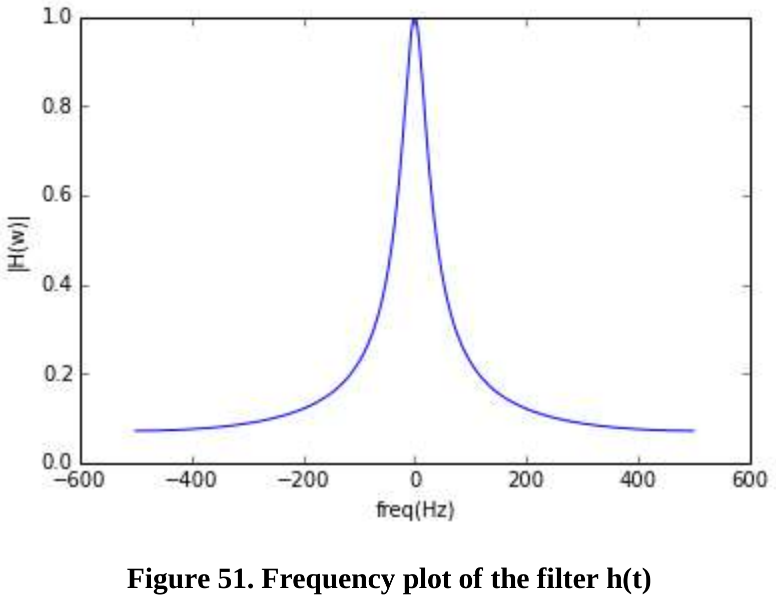


Figure 51. Frequency plot of the filter h(t)

In Figure 53, the spikes of the neurons with encoder = -1 are shifted down by 1.5 in order to avoid the overlapping of the spikes produced by the two neurons.

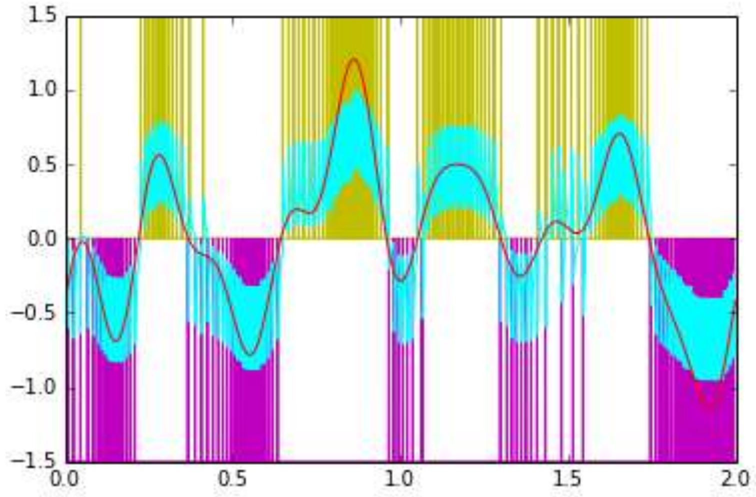


Figure 52. Plot of x(t) [Red], xhat(t) [cyan], neuron with e =1 [yellow], neuron with e = -1 [magenta]

d) generate a new x(t) with limit=5Hz

In this part, a new signal x(t) was generated using a different seed and was decoded using the decoders from part c.

RMSE of the original signal = 0.25162070483

RMSE of the new signal = 0.260889287038

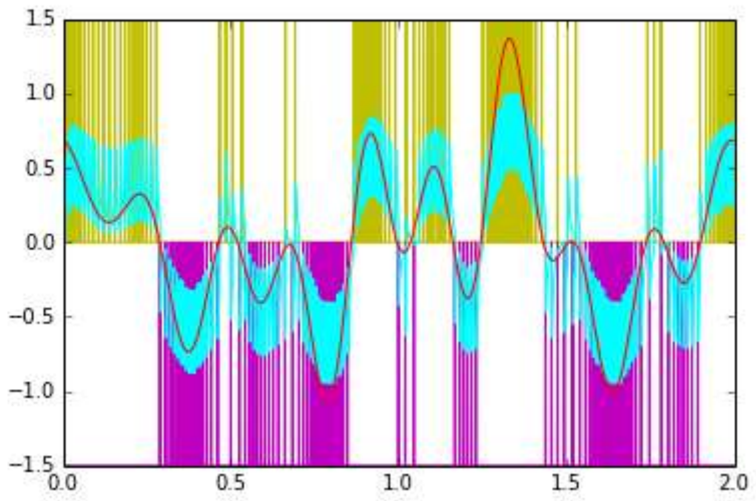


Figure 53. Plot of new x(t) [Red], xhat(t) [cyan], neuron with e =1 [yellow], neuron with e = -1 [magenta]

It can be seen that the RMSE in both cases is very similar although it is slightly higher for the new signal which is acceptable. This tells us that the decoders computed using the original signal did a pretty good job of decoding the new signal. Therefore the filter we created for a 5Hz signal is generalizable to another 5Hz random signal as well.