Assignment 2

Terry Stewart

January 28, 2014

1 SYDE556/750 Assignment 2: Spiking Neurons

- Due Date: February 14th at dawn (7:22am)
- Total marks: 20 (20% of final grade)
- Late penalty: 1 mark per day
- You can use any programming language you like, but it is recommended that you use a language with a matrix library and graphing capabilities. Two main suggestions are Python and MATLAB.

1.1 1) Generating a random input signal

1.1) Gaussian white noise

Create a function called that generates a randomly varying $x(t)$ signal. This should be like the `generate_signal` function used in the course notes.

The inputs to the function are:

- `T`: the length of the signal in seconds
- `dt`: the time step in seconds
- `rms`: the root mean square power level of the signal. That is, the resulting signal should have $\sqrt{\frac{1}{T}\int x(t)^2 dt} = rms$
- `limit`: the maximum frequency for the signal (in Hz)
- `seed`: the random number seed to use (so we can regenerate the same signal again)

Notes:

- To do Fourier transforms in MATLAB, see here
- To do Fourier transforms in Python, see here

- In both cases, the transform takes you from t to ω (or back the other way). Importantly, ω is frequency in *radians*, not in Hz. To convert Hz to radians, multiply by 2π.
- $\Delta\omega$ will be $2\pi/T$
- To keep the signal real, $X(\omega) = X(-\omega)^*$ (the complex conjugate: the real parts are equal, and the imaginary parts switch sign)

- When randomly generating $X(\omega)$ values, sample them from a Normal distribution $N(\mu = 0, \sigma = 1)$. Remember that these are complex numbers, so sample twice from the distribution; once for the real component and once for the imaginary.
- To implement the limit, set all $X(\omega)$ components with frequencies above the limit to 0
- a) [1 mark] Plot $x(t)$ 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.
- b) [1 mark] Plot the average $|X(\omega)|$ (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 should have a different `seed`)

1.2) Gaussian power spectrum noise

Create a modified version of your function from question 1.1 that produces noise with a different power spectrum. Instead of having the $X(\omega)$ values be 0 outside of some limit and sampled from $N(\mu = 0, \sigma = 1)$ inside that limit, we want a smooth drop-off of power as the frequency increases. In particular, instead of the `limit`, we sample from $N(\mu = 0, \sigma = e^{-\omega^2/(2*b^2)})$ where b is the new `bandwidth` parameter that replaces the `limit` parameter.

- a) [1 mark] 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.
- b) [1 mark] Plot the average $|X(\omega)|$ (the norm of the Fourier coefficients) over 100 signals generated with `T`=1, `dt`=0.001, `rms`=0.5, and `bandwidth`=10 (each of these 100 signals should have a different `seed`)

1.2 2) Simulating a Spiking Neuron

Write a program to simulate a single Leaky-Integrate and Fire neuron. The core equation is $\frac{dV}{dt} = \frac{1}{\tau_{RC}}(J - V)$ (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 τ_{ref} amount of time. Also, if the voltage goes below zero at any time, reset it back to zero. For this question, τ_{RC}=0.02 and τ_{ref}=0.002

Since we want to do inputs in terms of x, we need to do $J = \alpha e \cdot x + J^{bias}$. For this neuron, set e to $+1$ and find α and J^{bias} such that the firing rate when $x = 0$ in 40Hz and when $x = 1$ it is 150Hz. To find these α and J^{bias} values, use the approximation for the LEF neuron $a(J) = \frac{1}{\tau_{ref} - \tau_{RC} ln(1 - \frac{1}{J})}$.

- a) [1 mark] Plot the spike output for a constant input of $x = 0$ over 1 second. Report the number of spikes. Do the same thing for $x = 1$. Use `dt`=0.001 for the simulation.
- b) [1 mark] 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?
- c) [1 mark] Plot the spike output for $x(t)$ generated using your function from part 1.1. Use `T`=1, `dt`=0.001, `rms`=0.5, and `limit`=30. Overlay on this plot $x(t)$
- d) [1 mark] Using the same $x(t)$ signal as in part (c), plot the neuron's voltage over time for the first 0.2 seconds, along with the spikes over the same time.
- BONUS: 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 2 marks for actually implementing it and showing that it works.

1.3 3) Simulating Two Spiking Neurons

Write a program 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, use exactly the same settings as in question 2.

- a) [0.5 marks] Plot $x(t)$ and the spiking output for $x(t) = 0$ (both neurons should spike at ~40 spikes per second)
- b) [0.5 marks] Plot $x(t)$ and the spiking output for $x(t) = 1$ (one neuron should spike at ~150 spikes per second, and the other should not spike at all)
- c) [1 mark] Plot $x(t)$ and the spiking output for $x(t) = \frac{1}{2}sin(10\pi t)$ (a sine wave at 5Hz)
- d) [1 mark] Plot $x(t)$ and the spiking output for a random signal generated with your function for question 1.1 with `T`=2, `dt`=0.001, `rms`=0.5, and `limit`=5

1.4 4) Computing an Optimal Filter

Compute the optimal filter for decoding pairs of spikes. Instead of implementing this yourself, here is an implementation in Python and an implementation in Matlab.

- a) [1 mark] Document the code and connect it with the code you wrote for part (3). Comments should be filled in where there are `#` signs (Python) or `%` signs (Matlab). Replace the `'???'` labels in the code with the correct labels.
- b) [1 mark] Plot the time and frequency plots for the optimal filter for the signal you generated in question (3d)
- c) [0.5 marks] Plot the $x(t)$ signal, the spikes, and the decoded $\hat{x}(t)$ value for the signal in question (3d)
- d) [0.5 marks] Plot the $|X(\omega)|$ power spectrum, $|R(\omega)|$ spike response spectrum, and the $|\hat{X}(\omega)|$ power spectrum for the signal in question (3d)
- e) [1 mark] Generate time plots for the optimal filter for different `limit` values of 2Hz, 10Hz, and 30Hz. Describe the effects on the time plot of the optimal filter as the `limit` increases. Why does this happen?
- f) [1 mark] Generate time and power spectrum plots for the optimal filter for different `T` values of 1 second, 4 seconds, and 10 seconds. What happens to the shape of these plots as `T` increases? Why does this happen?

1.5 5) Using Post-Synaptic Currents as a Filter

Instead of using the optimal filter from the previous question, now we will use the post-synaptic current instead. This is of the form $h(t) = t^n e^{-t/\tau}$ normalized to area 1.

- a) [1 mark] Plot the normalized $h(t)$ for n=0, 1, and 2 with τ=0.007 seconds. What two things do you expect increasing n will do to $\hat{x}(t)$?
- b) [1 mark] Plot the normalized $h(t)$ for τ=0.002, 0.005, 0.01, and 0.02 seconds with n=0. What two things do you expect increasing τ will do to $\hat{x}(t)$?
- c) [1 mark] Decode $\hat{x}(t)$ from the spikes generated in question (3d) using an $h(t)$ with n=0 and τ=0.007. Do this by generating the spikes, filtering them with $h(t)$, and using that as your activity matrix A to compute your decoders. Plot the time and frequency plots for this $h(t)$. Plot the $x(t)$ signal, the spikes, and the decoded $\hat{x}(t)$ value.
- d) [1 mark] Use the same decoder and $h(t)$ as in part (c), but generate a new $x(t)$ with `limit`=5Hz. Plot the $x(t)$ signal, the spikes, and the decoded $\hat{x}(t)$ value.