Exercise 3: Temporal representation in spiking neurons

3.1 Constructing useful subroutines

1. Create a Matlab subroutine that will generate a signal $x(t)$, with Fourier coefficients **A**.

   [x,A] = genSignal(T,dt,rms,bandwidth,randomSeed)

   Where:

   - T is the length of the signal in seconds
   - dt is the time step size (1 or 0.1 milliseconds)
   - rms is the desired root mean square power level of $x(t)$: $rms = \sqrt{\frac{1}{T}\int x(t)^2 dt}$
   - bandwidth is a vector with the minimum and maximum frequency components allowed
   - randomseed is a positive integer that can be used to seed randn to guarantee the generation of the same ('random') signal over and over again

   Submit:

   Plots (a) and (b) for

   - Gaussian white noise: $P(\omega) = 1$ for (-bandwidth<=ω<=bandwidth) and $P(\omega) = 0$ otherwise
   - The power spectrum $P(\omega) = e^{-(\omega - minfreq)^2/2bandwidth^2}$; (i.e. a Gaussian power spectrum).

   (a) Plot of x for T = 1.0 seconds, dt = 0.001, rms = 0.5, bandwidths = 2π*(5, 10, 50 Hz).

   (b) Plot of the power spectrum (norm of Fourier coefficients) used to create the signal x at 10Hz (averaged over several, e.g. 100, examples).

   - Complete, well-commented code

   Hints:

   - Useful Matlab functions: FFT, IFFT, FFTSHIFT, RANDN
   - When switching to the time domain, be sure to only take the real part, as numerical error may cause small imaginary values to appear, which shouldn't.
   - The norm of a complex number can be found with abs(x) in Matlab.
   - The power spectrums are to be used in conjunction with the equations from the lecture notes.

- The purpose of this exercise is to get you to write a reusable signal generator for randomly picking signals from some statistically defined ensemble, as discussed in the lecture.
- To ensure your RMS=x, pick a signal and then normalize it to x (both the signal and the amplitudes).
- Remember you should have complex amplitudes and the negative frequencies are complex conjugates of the positive ones.
- Run the script several times for a fixed bandwidth to see how the signals change shape for different calls to randn, then change the bandwidth and note how the character of the signals change.
- $\Delta\omega = 2\pi/T$, $N = 2 * floor(T/2\Delta t) + 1$

2. Write a program that generates spikes for one LIF neuron.

   The analytic solution for the LIF firing rate (spikes/s) is given by

$$a(x) = \frac{1}{\tau^{ref} - \tau^{RC} \ln\left(1 - \frac{J_{th}}{J_M(x)}\right)}, \tag{3.1}$$

   where $J_M(x) = \alpha x + J^{bias}$.

   Set $\tau^{ref} = 2$ ms, $\tau^{RC} = 20$ ms. Find α and J^{bias} such that at $a(x=0) = 40$ and $a(x=1) = 150$. These are the parameters to use for the remainder of this exercise. Hint: Do the calculation by rescaling $J_{th} = R = 1$.

   Submit:

   (a) Plot of the spike trains for $x = 0$ and $x = 1$ for $t = 1$ s with these parameters and report the number of spikes.

   (b) Plot of the spike trains for $x(t)$ generated by the signal generator in the previous example with Gaussian white noise over $0-30$ Hz, rms=0.5, randomSeed=52, t=1. Overlay $x(t)$.

   (c) For (b), save V(t) and plot it for the first .2 s. Report the spike rate.

   Hints:

   - Solve $\dot{V}(t) = -\frac{1}{\tau^{RC}}(V - J_M(x(t))R)$ numerically, i.e. at discrete time steps, dt=0.001 sec.
   - Detect when $V(t) = 1$, save the spike time, reset to $V = 0$ and wait τ^{ref}.

   Bonus:

   - How can you improve the spike time accuracy without changing dt=0.001.

3. Write a program that generates spike trains for a symmetrical pair of on/off neurons with the same parameters used in question 2.

   [onOffCount, tonOff] = genOnOffSpikes(x(t), dt, parameters)

   Submit:

   (a) Plot the signal $x(t)$ and overlay the on and off spikes for:

      i. $x(t) = 0$, (should get ~40 spikes/second for each)
      ii. $x(t) = 1$ (only the 'on' at ~150 spikes/sec, the 'off' is off)
      iii. $x(t) = \frac{1}{2}\sin(2\pi \cdot 5 \cdot t)$ (A sinwave at 5Hz).
      iv. $x(t)$ = genSignal(T=2.0, dt=0.001, rms=0.5, bandwidth=[0, 2*pi*5], randomSeed=99) (Gaussian white noise from 0 to 5 hz.)

   (b) Complete and well-commented code.

   Hints:

   - onOffCount should be a 1x2 vector with the number of on and off spikes generated.
   - tonOff should be a 2 x floor(dt*TimeSteps/Tref)) vector, where you put in the time that on and off spikes occur and leave it zero otherwise.
   - Make sure the program can be generalized to more neurons later on, i.e. write the subroutine so that both neurons are updated together at each time step.
   - Note that you now have to make sure that $V(t) \geq 0$ while solving the differential equations (i.e. assume the neuron has a floor at 0).
   - Don't include a plot of $V(t)$ (i.e., don't save the voltage at each time).
   - α for the off neuron = $-\alpha$ for the on neuron, J^{bias} is the same.
   - Initialize the on voltage $V(t = 0) = 0$, and the off voltage $V(t = 0) = 0.5$.

3.2 Finding and using optimal filters

1. Download, comment, and implement the code for finding optimal filters.

   Submit:

   (a) Time and frequency plots of optimal filter for the signal in 3.a.iv above.

   (b) The signal, estimate, and spike plot for the signal in 3.a.iv above.

   (c) The signal, spike, and correlated power spectra for 3.a.iv above.

   (d) Compute and plot for the signal in 3.a.iv (note windowing):

i. the residual error at each frequency assuming no noise

$$MSE(\omega) \equiv \Delta A^2(\omega) = \left\langle |A(\omega) - \hat{A}(\omega)|^2 \right\rangle_{\mathbf{A}} \tag{3.2}$$

ii. the signal to noise ratio

$$SNR(\omega) = \frac{\left\langle |\hat{A}(\omega)|^2 \right\rangle_{\mathbf{A}}}{\Delta A^2(\omega)} \tag{3.3}$$

(Bonus: What is this signal to noise *really* telling you?)

iii. the information per frequency channel,

$$Info(\omega) = \frac{1}{2} log_2(1 + SNR(\omega)) \tag{3.4}$$

iv. report the information per spike, and the information rate

(e) Time plots of the optimal filter for Gaussian white noise with bandwidths from zero to $2, 10, 30$ Hz. Describe the effects on the temporal filter as the bandwidth increases.

(f) Comparison of plots of the time filter and its power spectrum for runs of 1, 4, and 10 seconds at 30Hz. What happens to the shape of the power spectrum and the filter?

(g) Generate a signal with bandlimited white noise in the frequency range of 30-70Hz. Run with rms=0.05, 0.1, and 0.5. What happens to the information transfer rate, why?

(h) The completed and commented code.

Hints:

- The code is on the website.
- Comments should be filled in where there are single '%' signs.
- The comments should indicate that you know what is going on in the code (refer to the text, class notes, and appendices for help).
- All graphs should be appropriately titled (i.e., replace all '???').
- This code should be made to work with the subroutines you wrote in the previous exercises. I recommend altering your code to do this.
- Recall that $\langle \cdot \rangle_{\mathbf{A}}$ includes windowing by definition.

3.3 Using PSCs

1. Generating post-synaptic currents. The general form for these is

$$PSC(t) = t^n e^{-t/\tau_{PSC}}$$

(normalized the area to 1).

Submit:

(a) A plot of (normalized) PSCs for $n = 0, 1, 2$ with $\tau_{PSC} = 7$ ms. What two things do you expect increasing n will do to $\hat{x}(t)$?

(b) A plot of (normalized) PSCs for $n = 0$ with $\tau_{PSC} = 2, 5, 10, 20$ ms. What two things do you expect increasing τ_{PSC} will do to $\hat{x}(t)$?

2. Use the code in the exercise 3.2 to estimate signals with PSCs. Use $n = 0$ and $\tau_{PSC} = 7$ ms.

   Submit:

   (a) The same plots as in 1.a, 1.b, and 1.d.iv in exercise 3.2 for Gaussian white noise with a bandwidth of 0 to 30 Hz

   (b) The same plots as in 1.b and 1.d.iv in exercise 3.2 a bandwidth of 0 to 5 Hz (do not recalculate the optimal filter, see below).

   Hints:

   - To scale the PSC appropriately, you must find the original optimal filter and ensure that the PSC you are using has the same area.
   - You will have to control the length of the vector representing the PSC appropriately to use it in place of the optimal filter.
   - You should only have to add a maximum of 10 lines of code to do this.
   - Sometimes, things work best in the time domain.