# Exercise 2: Representation in populations of neurons

## 2.1 Representation of scalars

1. Write a Matlab program that implements the neural representation of a scalar using a *rectified* neuron model with equal numbers of on and off neurons. Make the neuron $x$-intercepts drawn from an even distribution over [-1,1], with a maximum firing rate evenly distributed between 100-200Hz. Generate figures that

   (a) Display the neuronal responses $a_i(x)$ for $N = 16$ (i.e., 8 on and 8 off neurons).

   (b) Show the optimal linear decoders, $\phi_i$, as in equations (2.4-2.5) and Appendix A of the textbook.

   (c) Show $\hat{x} = \sum_{i=1}^{N} \phi_i a_i(x)$. Overlay a plot of $y = x$ to see the distortion errors. Overlay another plot (with a different y-axis (matlab: 'plotyy')) of just the error by plotting $\hat{x} - x$. Report the RMS error value.

   (d) Show $\hat{x} - x = \sum_{i=1}^{N} \phi_i (a_i(x) + \eta_i) - x$ where $\eta_i$ is Gaussian random noise with $\bar{\eta} = 0$ and $\sigma = .2$. Use the $\phi_i$ found in (c). Compare this plot to that in (c). Report the RMS error value.

   (e) Find the optimal linear decoders $\phi_i$ *under noise* as in equations (2.7-2.10) of the textbook. Now plot (c) again (but using these new decoders). Finally, reproduce plot (d) *exactly* as before (using the decoders found in (b)), but with an overlay of the same plot using the $\phi_i$ found under noise in (e).

   (f) Compare the respective RMS errors in a 2x2 table ($\phi_i$ found with and without noise; decoding $a_i$ with and without noise), and comment briefly on them.

   Hints:

   - Use the *rand* function in Matlab for the even distribution for $x$-intercepts and max firing rates.
   - Evaluate the integrals as using $dx = 0.05$, certainly no smaller than 0.01.
   - Implement the rectified neuron model function $[f(x)]_+$ as $a(x) = (f(x) > 0) .* f(x)$ in Matlab. For this model, $f(x) = mx + b$.
   - Recall the Gaussian distribution, $\rho(\eta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\eta-\bar{\eta})^2/(2\sigma^2)}$. Plot this if you are unfamiliar with it. Use the matlab *randn* function to generate Gaussian distributed noise (scale and shift the results of *randn* to get the desired distribution).
   - You can solve for the $\alpha_i$ for the 'on' half of the population and then just negate them to give the 'off' half.

- Note that the standard deviation of 0.2 is *relative to a normalized maximum firing rate* of 1 (i.e., possibility of about 10% error). So when you use use real firing rates, *scale the noise* accordingly.
- Debug the program using $N = 4$, not $N = 1000$.

2. Use the previous program to examine the sources of error.

   (a) Plot the error due to distortion, $E_{dist}$, and the error due to noise, $E_{noise}$, as a function of $N$. Use the loglog plot and run simulations for $N = 4, 8, 16, 32, 64, 128, \ldots$ until you run out of patience. Do about 5 runs at each population size and average the results. Approximately fit (by shifting up or down) $1/N$ and $1/N^2$ to these lines, and plot them. Use $\phi_i$ under noise, and $\sigma = .1$.

   (b) On a different graph, examine the effects of the noise variable on $E_{dist}$ and $E_{noise}$ by using $\sigma_\eta = 0.1$ and 0.01, and regenerating some of the points in (a). Put all four plots (i.e. each kind of error at both variances) on one graph. What do the graphs in (a) and (b) tell you about sources of error in neural populations?

3. Change the program to run with the nonlinear neuronal response function that arises from the leaky integrate and fire (LIF) model of neuron spike generation:

$$\begin{aligned} a_i(x) &= G_i\left[J_i(x)\right] \\ &= \begin{cases} \dfrac{1}{\tau_i^{ref} - \tau_i^{RC} \ln\left(1 - \frac{J_i^{threshold}}{J_i(x)}\right)}, & \text{if } \alpha_i x + J_i^{bias} > J_i^{threshold} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2.1}$$

   where

$$J_i(x) = \alpha_i x + J_i^{bias}.$$

   Again make sure the $x$-intercepts are evenly distributed, $J^{threshold} = 1$, and that the firing rates are evenly distributed between 100-200Hz. Typical values for $\tau_i^{ref}$ are between 0.5-2ms and for $\tau_i^{RC}$ are between 5-50ms (pick some and leave them constant; note that the equation is scaled to seconds). See figure 4.3 in the book for information on how changing these variables affects the response functions. Generate the same plots as in 1.a and 1.d, finding the optimal decoders under noise (don't compare it to plot 1.c). Report the RMSE.

   Suggestions:

   - You have two unknowns and two points on the function (the $x$-intercept and the firing rate at $x$=1 or -1). Solve the equation for one of the unknowns without substituting the points in first. Then proceed as usual (i.e., substituting the points, solving for the other unknown and then finding the first unknown). Recall that the firing rate is zero at $J(x) = J^{threshold}$.
   - The LIF functions will look odd if you plot or keep the whole thing. Only keep the part of the function that matters, i.e. $x > x_{intercept}$ (Matlab: ai.*(x>xintercept)) for 'on' neurons and $x < x_{intercept}$ for 'off' neurons.

## 2.2 Representation of vectors

1. Vector tuning curves:

   (a) Plot the tuning curve of a LIF neuron whose preferred direction vector is at -$\pi/4$ in the x-y plane, with an intercept of 0, and a max firing rate of 100Hz (along the preferred direction). This is a 3D plot.

   (b) Plot the cosine tuning curve for this neuron over a range of $[0, 2\pi]$ centered on the perferred direction. Fit this with a cosine and plot both. How good is the fit, why is it good at all, and why does it deviate?

   Notes:

   - Do a least squares fit for (b). Either directly or using *lsqcurvefit* in Matlab. You can use the form $A\cos(B\theta + C) + D$.
   - Other useful functions for both of these questions include: repmat, meshgrid, surf, reshape

2. Vector representation:

   (a) Plot a set of 100 unit vectors drawn from an even distribution (i.e. use *rand*) around the unit circle (try using Matlab's *quiver* function for plotting vectors).

   (b) Find the optimal decoding *vectors* for LIF neurons under noise as before, and plot them. How do they compare to the encoding vectors?

   (c) Using 20 unit vectors distributed around the circle as the input, $\mathbf{x}$ (i.e. don't worry about the magnitude of the vectors), plot the decoded estimates and compare the mean square error (MSE) for 1) the optimal decoders from (b) and 2) Georgopoulos' preferred direction decoding (i.e. using the preferred direction encoders as decoders) for *direction* only.

   (d) Using 20 vectors whose magnitude and direction are distributed within the unit circle, plot the decoded estimates and compare the MSE between the two decodings for direction and magnitude combined.

   Notes:

   - The 100 unit vectors are the prefered direction, or encoding, vectors for the neurons.
   - The neurons should be chosen with parameters distributed as in 1. Each neuron chosen will then be associated with a particular encoding vector (so those parameters are defining activity along the encoding vector).
   - $\Gamma$ is the same form as before, but will take longer to compute (again, test with a few neurons).

- To compare direction only, decode the vectors as usual, and then compute the angle around the circle. Use the actual and decoded angles to compute the MSE. Only plot the decoded angle as a scalar.
- To compare magnitude and direction, encode and decode the vectors as normal and use the actual original vector and it's decoded version to compute MSE.