Continuous Decoding Problem Set
42-699B/86-595 Neural Data Analysis
Due at the beginning of class on Tuesday, 12/4/12.

1. (20 pts.) You decide you can speed up the implementation of your Kalman filter by skipping the one-step prediction and instead using your estimate of velocity on the current timestep as the prior for the next timestep. That is, where normally you would solve the measurement update and one-step predictions iteratively on each timestep:

Measurement Update

$$P(v_t|\{f\}_1^t) = \frac{P(f_t|v_t)P(v_t|\{f\}_1^{t-1})}{P(f_t|\{f\}_1^{t-1})}$$

One-step prediction

$$P(v_t|\{f\}_1^{t-1}) = \int P(v_t|v_{t-1})P(v_{t-1}|\{f\}_1^{t-1})dv_{t-1}$$

you instead decide to just iterate the Measurement Update step, by directly plugging in your previous estimate as the prior:

$$P(v_t|\{f\}_1^t) = \frac{P(f_t|v_t)P(v_{t-1}|\{f\}_1^{t-1})}{P(f_t|\{f\}_1^{t-1})}$$

Describe qualitatively what will happen to your estimate of velocity over time if you do this.
(Hint: when in doubt, try (1) simulating it, or (2), solving the 1D case…)

2. (75 pts.) Matlab analysis: Continuous arm trajectory prediction.

In this problem, you will build and test different decoders that take spike counts and use them to predict arm movements in a 2D plane.

Download the data set ps6_data_2D.mat. It has several data structures in it:

arm_vel: cell array of arm trajectories to predict. Each cell is a separate trajectory.
(E.g., arm_vel{3}(5,:) is the 5th time point of the 3rd trajectory.)

spikes: cell array of binned spike counts corresponding to each trajectory. There are 63 neurons included; spikes{3}(:,5) is a 63-element long vector of spike counts for the 5th time point of the 3rd trajectory.

spikes2: same as spikes, but corresponding to a *hidden* set of trajectories! These will be important for question 2(c).

dt: time-step. Needed for converting counts to rates and (hint!) for the Poisson ML decoder.

cosine_fit: structure containing the tuning curve information. It has three elements:

.b0s: vector of baseline firing rates for each of the 63 neurons

.Bs: matrix of tuning curve coefficients (its size is # neurons x 2).

.sigma: covariance matrix of the fit, 63-x-63. Notice that it's diagonal; we've assumed the neurons are conditionally independent given the velocity.

2a. (20 pts.) Implement a PVA decoder. Start by implementing it on the raw spike counts – that is, don't smooth the spike counts. Plot the predictions you make for the 2nd trajectory. I have included a file plot_trajectory.m that should help you make the plot.

Now investigate the effects of smoothing on your predictions. Try smoothing the spike counts by averaging the spike counts over the last T time bins, where T varies from 1 to 10. What value of T gives you the best estimate? I have included a file, evaluate_fit_2D.m to help you determine which fit is best. It computes the R^2, or coefficient of determination, of your fit. The closer it is to 1, the better. Plot the R^2 as a function of T for your PVA decoder. Then plot the predictions you make for the 2nd trajectory using the optimum smoothing.

2b. (20 pts.) Implement an OLE decoder. Again, start by implementing it on the raw spike counts. Plot the predictions you make for the 2nd trajectory.

Again smooth the spike counts by averaging the spike counts over the last T time bins, where T varies from 1 to 10. What value of T gives you the best estimate? Plot the R^2 as a function of T for your OLE decoder, and plot the predictions you make for the 2nd trajectory using the optimum smoothing.

2c. (35 pts.) Turn in (as a .mat file) the velocities you predict from the second array of spike counts, spikes2. We will evaluate it against the real velocities. Your grade will be determined by how well you predict the real velocities, based on the scale below.

R^2>.5: 10 pts. R^2>.75: 15 pts. R^2>.9: 25 pts. R^2>.92: 30 pts. R^2>.93: 35 pts.
R^2>.95: 35 pts, PLUS 5 pts extra-credit towards the final.

To get an R^2>.95, you'll have to go beyond the OLE. You will even have to go beyond the Kalman. It is possible, however. (Try building a maximum likelihood estimator that assumes Poisson distributed spike counts. It will have to be solved numerically, for example, using MATLAB's fminunc function…)

3. (5 pts.) About how much time did you spend on each question of this problem set? Which problem taught you the most, and which taught you the least? How could this problem set be improved?