Homework

Warning: I tend to write long problem sets, but most of it is informational, there is not that much for you to actually do. So please don't be put off by the length.

All instructions that you will be graded on are in ***bold italics.***

Notation: boldface small letters, like $\mathbf{r}$, represent column vectors; $\mathbf{r}^{\mathrm{T}}$ is a row vector, the transpose of $\mathbf{r}$; boldface capital letters, like $\mathbf{W}$, represent matrices; $\mathbf{W}^{\mathrm{T}}$ is the transpose of $\mathbf{W}$; non-boldface letters represent numbers, either scalars or the individual elements of vectors or matrices.

Consider a two-population model of firing-rate neurons: one excitatory (E) population and one inhibitory (I) population. r_E and r_I are the firing rates of the excitatory and inhibitory populations, respectively, represented by the vector $\mathbf{r} = \begin{pmatrix} r_E \\ r_I \end{pmatrix}$. The matrix of connections between them is $\mathbf{W} = \begin{pmatrix} w_{EE} & -w_{EI} \\ w_{IE} & -w_{II} \end{pmatrix}$, where w_{XY} represents the (positive) strength of the connection from y to x. We let the vector of external inputs to the two populations be $\mathbf{i}$. We let $f(\mathbf{v})$ be a nonlinear function applied element-wise to the elements of the vector $\mathbf{v}$, *i.e.* $f(\mathbf{v})$ is a vector with i^{th} element $f(\mathbf{v})_i \equiv f(v_i)$. We assume the steady-state firing rate $\mathbf{r}_{SS}$ for a given input is given by f applied to each unit's input: $\mathbf{r}_{SS} = f(\mathbf{W}\mathbf{r} + \mathbf{i})$. We assume the network approaches its instantaneous steady state with first-order dynamics: letting $\mathbf{T} = \begin{pmatrix} \tau_E & 0 \\ 0 & \tau_I \end{pmatrix}$ be the diagonal matrix of E and I time constants, we have

$$\mathbf{T}\frac{d}{dt}\mathbf{r} = -\mathbf{r} + f(\mathbf{W}\mathbf{r} + \mathbf{i}) \tag{1}$$

Problem 1: The inhibition-stabilized network (ISN). Suppose there is a fixed point $\mathbf{r}_{SS}$.

1. Assume you have linearized about the fixed point. You know that, letting f'_E and f'_I be the derivative of f evaluated at the E and I components of $\mathbf{W}\mathbf{r}_{SS} + \mathbf{i}$, respectively, the linearized weights are $\begin{pmatrix} \partial f_E/\partial r_E & \partial f_E/\partial r_I \\ \partial f_I/\partial r_E & \partial f_I/\partial r_I \end{pmatrix} = \begin{pmatrix} f'_E w_{EE} & f'_E w_{EI} \\ f'_I w_{IE} & f'_i w_{II} \end{pmatrix}$; to make notation simpler, let's define this to be $\mathbf{J} = \begin{pmatrix} j_{EE} & -j_{EI} \\ j_{IE} & -j_{II} \end{pmatrix}$ (we'll assume $f(x)$ is a monotonically increasing function of x, so that all the f'_X's are positive and hence all the j_{XY}'s are positive). Let $\mathbf{i}_{SS}$ be the steady-state input that yields the fixed point $\mathbf{r}_{SS}$. If there is a deviation $\Delta\mathbf{i}$ from $\mathbf{i}_{SS}$, in the linearized equation this becomes $\delta\mathbf{i} \equiv \begin{pmatrix} f'_E \Delta i_E \\ f'_i \Delta i_I \end{pmatrix}$. Define small deviations in response from the steady state by $\mathbf{r} = \mathbf{r}_{SS} + \delta\mathbf{r}$.

Then the equation for the dynamics linearized about the fixed point is

$$\mathbf{T}\frac{d}{dt}\delta\mathbf{r} = -\delta\mathbf{r} + \mathbf{J}\delta\mathbf{r} + \delta\mathbf{i} \tag{2}$$

This should all be familiar to you, but if it's not, satisfy yourself that this is all true.

Now for a steady-state input perturbation $\delta\mathbf{i}$, ***write down the equation*** for the steady-state response $\delta\mathbf{r}$ (this should involve the matrix $(\mathbf{1}-\mathbf{J})^{-1}$ where $\mathbf{1}$ is the identity matrix and the superscripted -1 indicates the matrix inverse). For a 2×2 matrix $\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the inverse is given by $\mathbf{M}^{-1} = \frac{1}{\mathrm{Det}\,\mathbf{M}}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$. Recall that, for the fixed point to be stable, we must have $\mathrm{Det}\,(\mathbf{J}-\mathbf{1}) > 0$. ***Show*** that, for a stable fixed point, if a perturbation is given only to the I cells ($\delta\mathbf{i} \propto \begin{pmatrix} 0 \\ 1 \end{pmatrix}$), the steady-state response is "paradoxical" (response to a positive input is negative; response to a negative input is positive) if and only if $j_{EE} > 1$. ***Show*** that $j_{EE} > 1$ is precisely the condition that the E subpopulation alone is unstable: that is, if r_I were a fixed constant, fixed to its steady state value, so that the only dynamics were the E equation with the fixed r_I, then the fixed point would be unstable.

Thus, you've established that in an ISN – a network at a stable fixed point, for which the dynamics of the excitatory subnetwork alone would be unstable at that fixed point – the steady-state response to input to inhibition is "paradoxical", *e.g.* adding excitation to the I cells causes the I cells to decrease their firing rate in the new steady state.

2. Now show the same things using nullclines. Again assume that the function $f(x)$ is a monotonically increasing function of x. The equations for the E and I nullclines are the E and I components of the fixed-point equation: $\mathbf{r} = f(\mathbf{W}\mathbf{r} + \mathbf{i})$. We will draw the nullclines with r_E on the x axis and r_I on the y axis.

   For the I nullcline, ***compute its slope***, dr_I/dr_E; you should find that it is given by $\frac{j_{IE}}{1+j_{II}}$. This means that the nullcline always has positive slope.

   Now for the E nullcline, ***compute the inverse of its slope***, dr_E/dr_I; you should find that this inverse slope is $\frac{j_{EI}}{j_{EE}-1}$. This means that the slope is positive if the E subnetwork is unstable, and negative if the E subnetwork is stable.

   Show that the condition that $\mathrm{Det}\,(\mathbf{J}-\mathbf{1}) > 0$, which is necessary for stability, is equivalent to the I nullcline having a larger slope than the E nullcline. So for a fixed point to be stable, it is necessary that the I nullcline have a larger slope than the E nullcline at their crossing that defines the fixed point.

   So, we'll ***graph two versions*** of the nullclines: one that is an ISN, one that is not. First draw the I nullcline, which will be the same for both versions. Imagine that, for

r_E small, the I-nullcline solution for r_I should be small, while for r_E large, r_I is large; so the nullcline could start toward the bottom left, and make, for example, a sigmoidal shape to the top right, with positive slope always.

Now, draw the E nullclines, assuming a stable fixed point. Imagine that when r_I is high, r_E is low, so the nullcline starts in the upper left corner; while when r_I is low, r_E is high, so it ends up in the lower right corner. In the non-ISN version, it has a negative slope all the way. In the ISN version, it has a positive slope in a middle portion, so the nullcline looks like a sideways S; and the fixed point is on this positive-sloping middle portion.

Draw the arrows indicating the direction of flow in the different regions of the nullcline plane. ***Show on the plot*** that, in negative-sloping regions of the E nullcline, if r_I is kept fixed, small perturbations off the E nullcline will flow back to the nullcline; while in positive-sloping regions, it will flow away. This also tells you that in positive-sloping regions, the E subnetwork alone is unstable, while in negative-sloping regions it is stable.

Now, suppose you add a positive input to the I cells. ***Show*** that the resulting change in the I nullcline is to reduce r_E by the same amount for any given r_I, that is, to move the I nullcline leftward. There is no change in the E nullcline. ***Show on the plot*** that, for a stable fixed point, if the network is an ISN, the result is to decrease both r_E and r_I in moving to the new fixed point; while for a non-ISN, the result is to decrease r_E but increase r_I. (For the ISN, assume that the new fixed point, like the old one, is on the positive-sloping portion of the E nullcline.)

In the ISN case, ***draw the dynamical path*** followed by r_E, r_I from the old fixed point to the new fixed point after adding the input. This addition of input instantaneously moves the nullcline; the resulting derivative at the old fixed point has an upward component, becoming horizontal as the flow crosses the I nullcline, and then going downward to the new fixed point (it might spiral into the fixed point if there are complex eigenvalues, or go straight down to it if eigenvalues are real). Note, regarding the old fixed point as a perturbation from the new fixed point, that, even though the new fixed point is stable, the dynamics move further away from the new fixed point (the upward movement) before ultimately flowing back to it. This is an effect of non-normal dynamics. (Recall that biological weight matrices, of the form $\mathbf{J} = \begin{pmatrix} j_{EE} & -j_{EI} \\ j_{IE} & -j_{II} \end{pmatrix}$ with all j_{XY}'s positive, are non-normal, meaning that their eigenvectors are not orthogonal, because $\mathbf{J}\mathbf{J}^{\mathrm{T}} \neq \mathbf{J}^{\mathrm{T}}\mathbf{J}$, which is the necessary and sufficient condition for non-normality.)

Problem 2: Non-normal dynamics.

Now we're going to restrict attention to linear dynamics, which you can think of as the linearized dynamics about a fixed point, but for simplicity we'll use $\mathbf{W}, \mathbf{r}, \mathbf{i}$ instead of $\mathbf{J}, \delta\mathbf{r}, \delta\mathbf{i}$. To further simplify, we're going to restrict to a class of connection matrices that has $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ as one of its eigenvectors: $\mathbf{W} = \begin{pmatrix} w_1 & -(w_1+x) \\ w_2 & -(w_2+x) \end{pmatrix}$ where $x > 0$. ***Verify*** that the (unnormalized) eigenvectors are $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ with eigenvalue $\lambda_1 = -x$, and $\begin{pmatrix} \frac{w_1+x}{w_2} \\ 1 \end{pmatrix}$ with eigenvalue $\lambda_2 = w_1 - w_2$. We assume the system is stable, that is $\lambda_1 < 1$ and $\lambda_2 < 1$. Note that the two eigenvectors are not orthogonal to each other, and that they are similar to one another in that both have all-positive entries, i.e. both point into the upper right quadrant. We'll also assume that $\tau_E = \tau_I = \tau$.

We're going to transform to an orthogonal basis that makes the weight matrix as simple as possible, namely upper triangular – only zeros below the diagonal (with a transformation to the non-orthogonal basis of the eigenvectors, the matrix can be made even simpler – diagonal; but upper triangular is as simple as we can make it with a transformation to an orthogonal basis). This is called the Schur transformation. To do this, we choose one eigenvector as the first Schur basis vector, and choose the other vector orthogonal to this one (more generally, in higher dimensions, you take some ordering of the eigenvectors and then do Gram-Schmidt orthonormalization to produce an orthogonal Schur basis; the transformation is not unique, because each ordering produces a different Schur basis). We'll choose our (normalized) Schur basis vectors to be $\mathbf{s}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and a vector orthogonal to it, $\mathbf{s}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. Since the component of a vector along an orthonormal basis vector is just given by the dot product of the vector with the basis vector, ***show*** that $\mathbf{r}$ has components $\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} \frac{r_E+r_I}{\sqrt{2}} \\ \frac{r_E-r_I}{\sqrt{2}} \end{pmatrix}$ in the $\mathbf{s}_1, \mathbf{s}_2$ basis, that is, the components r_1 and r_2 represent the sum and difference of E and I activities, respectively.

You already know that $\mathbf{W}\mathbf{s}_1 = -x\mathbf{s}_1$. ***Show*** that $\mathbf{W}\mathbf{s}_2 = (w_1 + w_2 + x)\mathbf{s}_1 + (w_1 - w_2)\mathbf{s}_2$, and so in the $\mathbf{s}_1, \mathbf{s}_2$ basis, $\mathbf{W}$ takes the form $\begin{pmatrix} \lambda_1 & w_{FF} \\ 0 & \lambda_2 \end{pmatrix}$ where the "feedforward weight" w_{FF} is given by $w_{FF} = w_1 + w_2 + x$.[1] We call w_{FF} a feedforward weight because it is an effective feedforward weight between activity patterns; the pattern $\mathbf{s}_2$ projects to $\mathbf{s}_1$

[1] If this being the form of the matrix representation isn't clear to you: in a given basis of vectors $\mathbf{b}_1, \mathbf{b}_2$, we can write any vector $\mathbf{v}$ as $\mathbf{v} = v_1\mathbf{b}_1 + v_2\mathbf{b}_2 = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$. Suppose $\mathbf{W}\mathbf{b}_1 = w_{11}\mathbf{b}_1 + w_{21}\mathbf{b}_2$ and $\mathbf{W}\mathbf{b}_2 = w_{12}\mathbf{b}_1 + w_{22}\mathbf{b}_2$. Then $\mathbf{W}\mathbf{v} = (w_{11}v_1 + w_{12}v_2)\mathbf{b}_1 + (w_{21}v_1 + w_{22}v_2)\mathbf{b}_2 = \begin{pmatrix} w_{11}v_1 + w_{12}v_2 \\ w_{21}v_1 + w_{22}v_2 \end{pmatrix}$. This is what you

with strength w_{FF}, and there is no projection back, it is a strictly feedforward connectivity between patterns, without loops; in addition, there are the self-loops $\mathbf{s}_1 \to \mathbf{s}_1$ with strength λ_1 and $\mathbf{s}_2 \to \mathbf{s}_2$ with strength λ_2.) The fact that $w_{FF} \neq 0$ is what makes the matrix nonnormal; this is clear in the $\mathbf{s}_1, \mathbf{s}_2$ basis, i.e. in this basis $\mathbf{W}\mathbf{W}^{trans} = \mathbf{W}^{trans}\mathbf{W}$ if and only if $w_{FF} = 0$. ***Show*** this also in the r_E, r_I basis: if $w_{FF} = w_1 + w_2 + x = 0$, then the matrix $\mathbf{W}$ in this basis becomes a symmetric matrix, $\mathbf{W} = \mathbf{W}^{\mathrm{T}}$, and therefore is normal; and the eigenvectors become orthogonal, *i.e.* the second eigenvector becomes $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

Solve the linear dynamics $\tau \frac{d}{dt}\mathbf{r} = -r + \mathbf{W}\mathbf{r} + \mathbf{i}$ for constant $\mathbf{i}$ in the $\mathbf{s}_1, \mathbf{s}_2$ basis. You will have to first solve for $r_2(t)$, then solve for $r_1(t)$ with $w_{FF}r_2(t)$ as one of the inputs. You should find

$$r_2(t) = r_2(0)e^{-(1-\lambda_2)t/\tau} + \frac{i_2}{1-\lambda_2}\left(1 - e^{-(1-\lambda_2)t/\tau}\right) \quad (3)$$

$$r_1(t) = r_1(0)e^{-(1-\lambda_1)t/\tau} + \frac{i_1 + w_{FF}i_2/(1-\lambda_2)}{1-\lambda_1}\left(1 - e^{-(1-\lambda_1)t/\tau}\right) \quad (4)$$

$$+w_{FF}\left(r_2(0) - \frac{i_2}{1-\lambda_2}\right)\frac{e^{-(1-\lambda_1)t/\tau} - e^{-(1-\lambda_2)t/\tau}}{\lambda_1 - \lambda_2} \quad (5)$$

Graph the function $\frac{e^{-(1-\lambda_1)t/\tau} - e^{-(1-\lambda_2)t/\tau}}{\lambda_1 - \lambda_2}$ for some choice of λ_1 and λ_2 as real numbers less than 1 (they can be negative). Note that this is a transient pulse that rises with time constant $\min\left(\frac{\tau}{1-\lambda_1}, \frac{\tau}{1-\lambda_2}\right)$ and then falls with time constant given by the max of the two time constants. It can be quite large because it is multiplied by $w_{FF} = w_1 + w_2 + x$, which will be large if the weights are large. It is this transient that can cause the dynamics to transiently move far from the steady-state fixed point, although as $t \to \infty$ the dynamics will approach the fixed point. For some choice of $w_1 > 1$, $w_2 > 1$ and $x > 1$ that gives a stable system, ***graph the timecourse*** of r_1, r_2, r_E, and r_I on the same plot, starting from an initial condition of $r_E = 1$, $r_I = 0$ (translate this into $r_1(0)$ and $r_2(0)$) and assuming no external input. Note that r_E or r_I are just found as a weighted sum or difference of r_1 and r_2, you don't have to separately compute their time courses. Make the same graph starting from the initial condition of $r_E = 0$, $r_I = 1$.

Now focus on the steady state for nonzero input, $r_2 = \frac{i_2}{1-\lambda_2}$, $r_1 = \frac{i_1 + w_{FF}i_2/(1-\lambda_2)}{1-\lambda_1}$. Note that small inputs i_2 to the difference of E and I can, for large w_{FF}, cause large steady state response of the sum of E and I. This is the effect underlying the paradoxical response: if i_2 is negative, representing tilting of the difference towards I, the result can be that I as well as E decreases in the new steady state. When precisely does this occur? Consider an input only to

get from applying $\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ to $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$. Since this is true for an arbitrary vector $\mathbf{v}$, $\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ is the representation of $\mathbf{W}$ in the $\mathbf{b}_1, \mathbf{b}_2$ basis. We can summarize this by saying $w_{XY} = \mathbf{b}_x^T \mathbf{W} \mathbf{b}_y$.

I. ***Show*** that this corresponds to $i_2 = -i_1$, with the input to I positive or negative according to whether i_1 is positive or negative. ***Show*** that for this case ($i_2 = -i_1$), the steady-state I response is $r_I = \frac{1}{\sqrt{2}}(r_1 - r_2) = \frac{i_1((1-\lambda_1)+(1-\lambda_2)-w_{FF})}{\sqrt{2}(1-\lambda_1)(1-\lambda_2)}$. Note that the nonnormal effect – the effect involving w_{FF} – has sign opposite to i_1; show that this represents a paradoxical effect. The remaining terms represent the normal effect – the effect in a normal matrix in which the eigenvectors are the Schur basis vectors and are orthogonal, so $w_{FF} = 0$; these terms have the same sign as i_1, representing a non-paradoxical effect. Thus the paradoxical effect arises precisely when the effect of nonnormality exceeds the remaining effects. Intuitively, if you give negative input to r_2 and an equal but opposite positive input to r_1, this pushes r_2 down and pushes r_1 up, both representing a nonparadoxical increase in inhibition; but since r_2 has a feedforward connection to r_1, the pushing down of r_2 also pushes down r_1 by the nonnormal effect (the feedforward connection). When the nonnormal effect exceeds the normal effects, the result is a paradoxical change.

Finally, ***show*** that the r_I response is paradoxical – negative for positive i_1 – if and only if $w_1 > 1$, which means if and only if the excitatory subnetwork by itself is unstable. To show this, you'll need to use the facts that $\lambda_2 < 1$ and $\lambda_1 < 1$.

In the general case – a weight matrix $\begin{pmatrix} w_{EE} & -w_{EI} \\ w_{IE} & w_{II} \end{pmatrix}$ – if the eigenvectors and eigenvalues are real, the results are similar: each eigenvector has both of it entries of the same sign (which you can take to be positive), so they both represent weighted sums of E and I; and, taking one of them to be the first Schur vector, the 2nd Schur vector, which will make a feedforward connection to the first, must have its two entries of opposite signs, representing a weighted difference of E and I. So it remains true that differences in E and I are amplified, via w_{FF}, into sums of E and I. When the eigenvectors and eigenvalues are complex, it gets a little more complicated, but there is a sense in which the same picture continues to be true.

Intro theory class: Homework

G. Sean Escola

March 25, 2017

1. ...

2. The continuous dynamics of a recurrent neural network (RNN) of N neurons are:

$$\begin{aligned} \tau\dot{\mathbf{x}}(t) &= -\mathbf{x}(t) + \mathbf{J}\mathbf{r}(t) + \mathbf{V}\mathbf{s}(t) \\ \hat{\mathbf{y}}(t) &= \mathbf{W}\mathbf{r}(t) \end{aligned} \tag{1}$$

where $r_n = F(x_n)$ for some nonlinearity $F(x)$ (e.g., $F(x) = \tanh x$), τ is the neural time constant, $\mathbf{x}$ and $\mathbf{r}$ are length N vectors (of the "fields" and firing rates of the neurons respectively), $\mathbf{J}$ is the $N \times N$ matrix of recurrent weights, $\mathbf{s}$ is an S dimensional stimulus vector, $\mathbf{V}$ is the $N \times S$ matrix of input weights, $\hat{\mathbf{y}}$ is the K dimensional network readout, and $\mathbf{W}$ is the $N \times K$ matrix of readout weights. To simulate such a network, the discrete time formation is needed:

$$\mathbf{x}(t+\Delta) = \mathbf{x}(t)\left(1 - \frac{\Delta}{\tau}\right) + \frac{\Delta}{\tau}\left(\mathbf{J}\mathbf{r}(t) + \mathbf{V}\mathbf{s}(t)\right),$$

where Δ is the time-step.

To do batch learning of RNNs à la Jaeger and Hass, 2004, you first run the following dynamics for T time-steps:

$$\mathbf{x}(t+\Delta) = \mathbf{x}(t)\left(1 - \frac{\Delta}{\tau}\right) + \frac{\Delta}{\tau}\left(\mathbf{J}^0\mathbf{r}(t) + \mathbf{U}\mathbf{y}(t) + \mathbf{V}\mathbf{s}(t)\right) + \sigma\sqrt{\frac{\Delta}{\tau}\left(2 - \frac{\Delta}{\tau}\right)}\boldsymbol{\xi} \tag{2}$$

where $\mathbf{y}$ is a K dimensional target function, $\mathbf{J}^0$ is the matrix of pre-learning recurrent weights, $\mathbf{U}$ is an $N \times K$ matrix of feedback weights, and $\boldsymbol{\xi}$ is an N dimensional gaussian noise vector (i.e., $\xi_n \sim \mathcal{N}(0,1)$). (The scale on the noise, $\sigma\sqrt{\frac{\Delta}{\tau}\left(2 - \frac{\Delta}{\tau}\right)}$, makes the standard deviation of each x_n exactly σ if the recurrent, feedback, and input weights were all zero. This is easy to prove to yourself if you'd like.) You then collect all of the rate vectors into a single $N \times T$ matrix $\mathbf{R} = [\mathbf{r}(\Delta)\ \mathbf{r}(2\Delta) \cdots \mathbf{r}(T\Delta)]$ and all the target vectors into a single $K \times T$ matrix $\mathbf{Y} = [\mathbf{y}(\Delta)\ \mathbf{y}(2\Delta) \cdots \mathbf{y}(T\Delta)]$. Least squares gives you the solution for the output weights:

$$\mathbf{W} = \mathbf{Y}\mathbf{R}^{\mathrm{T}}\left(\mathbf{R}\mathbf{R}^{\mathrm{T}} + \alpha\mathbf{I}\right)^{-1}.$$

(The formula we used in class has $\alpha = 0$. $\alpha > 0$ saves you if $\mathbf{R}\mathbf{R}^{\mathrm{T}}$ is not invertible, which may sometimes be the case especially if T is small. Conceptually this is unimportant, but you will need it for the programming assignment in Problem 3 below.)

After training, simulating Equation 1 with $\mathbf{J} = \mathbf{J}^0 + \mathbf{UW}$ should give you $\hat{\mathbf{y}}(t) \approx \mathbf{y}(t)$. However, as we discussed in class, Jaeger and Hass could only get this to work by using a lot of noise and by training for a long time (large σ and T). Sussillo and Abbott, 2009, were able to significantly reduce the training time and get rid of the need for noise by instead updating the weights at every time-step. Since $\mathbf{YR}^{\mathrm{T}} = \sum_{j=1}^{T} \mathbf{y}(j\Delta)\mathbf{r}(j\Delta)^{\mathrm{T}}$ and $\mathbf{RR}^{\mathrm{T}} = \sum_{j=1}^{T} \mathbf{r}(j\Delta)\mathbf{r}(j\Delta)^{\mathrm{T}}$ we can define:

$$\begin{aligned}
\mathbf{A}_i &= \sum_{j=1}^{i} \mathbf{y}(j\Delta)\mathbf{r}(j\Delta)^{\mathrm{T}} \\
&= \mathbf{y}(i\Delta)\mathbf{r}(i\Delta)^{\mathrm{T}} + \sum_{j=1}^{i-1} \mathbf{y}(j\Delta)\mathbf{r}(j\Delta)^{\mathrm{T}} \\
&= \mathbf{y}(t)\mathbf{r}(t)^{\mathrm{T}} + \mathbf{A}_{i-1}, \qquad (3)
\end{aligned}$$

and similarly:

$$\mathbf{C}_i = \mathbf{r}(t)\mathbf{r}(t)^{\mathrm{T}} + \mathbf{C}_{i-1}, \qquad (4)$$

with $\mathbf{A}_0 = \mathbf{0}$, $\mathbf{C}_0 = \alpha\mathbf{I}$, and $t = i\Delta$.

Then, we can update the readout weights at every time-step with $\mathbf{W}_i = \mathbf{A}_i\mathbf{C}_i^{-1}$. To train the weights, we now run Equation 2 but replace $\mathbf{y}(t)$ with $\mathbf{W}_i\mathbf{r}(t)$. Since $\mathbf{W}_i\mathbf{r}(t)$ will not be exactly $\mathbf{y}(t)$, this method will sample the noise in the directions that are most unstable, leading to stabilization of those dimensions. Thus we can significantly reduce σ or even set it to zero during training.

However, the recursive least squares algorithm described in Sussillo and Abbott (the "FORCE" algorithm), instead uses the following update:

$$\mathbf{P}_i = \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{r}(t)\mathbf{r}(t)^{\mathrm{T}}\mathbf{P}_{i-1}}{1 + \mathbf{r}(t)^{\mathrm{T}}\mathbf{P}_{i-1}\mathbf{r}(t)} \qquad (5)$$

$$\mathbf{W}_i = \mathbf{W}_{i-1} - \left(\mathbf{W}_{i-1}\mathbf{r}(t) - \mathbf{y}(t)\right)\mathbf{r}(t)^{\mathrm{T}}\mathbf{P}_i, \qquad (6)$$

with $\mathbf{P}_0 = \alpha^{-1}\mathbf{I}$ and $\mathbf{W}_0 = \mathbf{0}$. (FYI, the update for $\mathbf{P}_i$ is a special case of the more general Woodbury matrix identity.) Importantly, Equations 5 and 6 prevent the need for a matrix inversion on every time-step, which has the prohibitive computational complexity of $\mathcal{O}(N^3)$.

Prove that: (i) $\mathbf{P}_i$ is the inverse of $\mathbf{C}_i$. To do so, you just need to show that $\mathbf{C}_i\mathbf{P}_i = \mathbf{I}$ assuming that $\mathbf{C}_{i-1}\mathbf{P}_{i-1} = \mathbf{I}$. (This is technically a proof by induction; it's trivial to show that $\mathbf{C}_0\mathbf{P}_0 = \mathbf{I}$.) And (ii) Equation 6 gives the same result as $\mathbf{W}_i = \mathbf{A}_i\mathbf{C}_i^{-1}$.

3. ...