Lecture 9: Dynamic transformations (cont.)

9.1 Controlled integrator

- Let's quickly reconsider the controlled integrator (or more accurately controlled filter that becomes an integrator as a special case; see slide).
- To be more careful about our analysis, let's work with the same diagram, but explicitly write out the synaptic filters at each step. We know:

$$\begin{aligned} x(t) &= h_u(t) * u(t)B' + h_p(t) * f(p_1, p_2) \\ p_1(t) &= h_{A'}(t) * A' \\ p_2(t) &= h_x(t) * x(t) \\ f(p_1, p_2) &= p_1(t)p_2(t) \end{aligned} \tag{9.1}$$

- we know that each of the filters is of the form:

$$h_\alpha(s) = \frac{g_\alpha}{1 + s\tau_\alpha}$$

- Solving equation (9.1) gives:

$$x(s) = \frac{h_u(s)u(s)B'}{1 - h_p(s)h_{A'}(s)h_x(s)A'}$$

so, the network temporal filter is given by

$$\begin{aligned} h_{net}(s) &= \frac{h_u(s)B'}{1 - h_p(s)h_{A'}(s)h_x(s)A'} \\ &= \frac{g_u B'(1 + s\tau_x)(1 + s\tau_{A'})(1 + s\tau_p)}{(1 + s\tau_u)\left[(1 + s\tau_x)(1 + s\tau_{A'})(1 + s\tau_p) - g_p g_{A'} g_x A'\right]} \end{aligned}$$

- That's pretty ugly, so let's assume that the recurrent time constant τ_x is much longer than all the rest (so it dominates). This gives

$$\begin{aligned} h_{net}(s) &\approx \frac{g_u B'(1 + s\tau_x)}{(1 + s\tau_x) - g_p g_{A'} g_x A'} \\ &= \frac{g_{net}(1 + s\tau_x)}{1 + s\tau'_{net}} \end{aligned}$$

where

$$\begin{aligned} g_{net} &= \frac{g_u B'}{1 - g_p g_{A'} g_x A'} \\ \tau_{net} &= \frac{\tau_x}{1 - g_p g_{A'} g_x A'}. \end{aligned}$$

As usual, we expect the denominator to be less than one so the network time constant is much longer than that of any of the populations. We can see that this becomes an integrator as $-g_p g_{A'} g_x A'$ goes to one, as expected. (We also know, given the derivation from last time, that this will operate best when $\tau_u = \tau_x + \tau_p$, although this is lost here.)

- Writing these equations out explicitly shows what kind of assumptions must be made when constructing such networks. Also notice that it makes clearer that the high-level translation between standard and neural control theory assumes that the τ used effectively summarizes all of the time constants in the neural network.
- Also note that there are at least two possible implementations of this network in a neural system. The first would be as discussed, where you construct a middle layer that multiplies the recurrent and A' signals. The second doesn't repeat the $x(t)$ representation in the middle layer, but rather has that representation be $x(t)$ (we'd have to rederive the dynamics eqns above to know how this will act; this works better).

9.2 Attractor networks

9.2.1 Introduction

- In dynamical systems theory, an attractor is a point towards which a system tends over time (see slide).
- The standard analogy is to imagine that the system is a ball on a hilly surface (the state space). When there is a large dent in the surface, the ball will tend towards the bottom of that dent, or basin. The point at the bottom of the basin is thus a *point attractor*
- In neural modeling, attractor networks (networks with attractors) have long been thought relevant for understanding certain behavior (e.g., memory, integration, off-line updating of representations, repetitive pattern generation, noise reduction, etc.)
- The neural integrator and working memory are both examples of attractor networks.
- The first can be thought of as a *line attractor*. Though it's really only an approximation to one (see slide).
- The second can be thought of as a *plane attractor* (similarly an approximation) – the 2D analog (see slide).
- Similarly we can construct a *ring attractor*. It can be thought of as a line attractor that's wrapped in a circle. This kind of attractor is a natural way to describe systems that can encode and hold positions over a repeating axis (e.g., directional heading in hippocampus).

- Attractor networks were first extensively examined in the ANN community (e.g. Hopfield nets). Amit suggested that observed persistent neural activity could be associated with recurrent biological networks as well. Persistent activity is common, it's found in motor, premotor, posterior parietal, prefrontal, frontal, hippocampal, and inferotemporal cortex, as well as the basal ganglia, midbrain, superior colliculus, and brainstem.
- However, work so far has focussed on simple attractors (namely the line and ring). But the concept seems easily generalizable – and we now have techniques to build arbitrary, complex attractors.

9.2.2 Generalization

- We can generalize attractors in two ways
- Representationally, we know that any attractor we can define for a scalar, we can define for functions and vectors as well. This is what the working memory model does. This is straightforward.
- We can use the means of quantifying representation discussed earlier to see what effect they have on dynamics. As expected, heterogeneity seems to make a system have more fixed points than the other (non-even) distributions – again this distribution provides for a nice trade off between useful properties (dynamic fixed points) and expense (i.e. you don't have to precisely space the intercepts; see slide).
- More interestingly, we can also generalize the dynamics. Because we can relate our neural systems to the standard state equations:

  $$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$

  knowing how to pick interesting values for $\mathbf{A}$, can help us build biologically plausible networks with interesting dynamics.
- One important case is the simple harmonic oscillator. Here, $\mathbf{A}$ is given by:

  $$\mathbf{A} = \left[\begin{array}{cc} 0 & \omega \\ -\omega & 0 \end{array} \right].$$

- This gives rise to a *cyclic* (or *periodic*) *attractor* (see slide: note, can't use the 'ball' analogy any more).
- Technically, this isn't a simple cyclic attractor, since the amplitude depends on the initial conditions. As a result, just as we can control the stable point of the line attractor by adjusting $u(t)$, so we can control the cycle (i.e., amplitude) at which the oscillator attractor operates by adjusting $\mathbf{u}(t)$.
- One notable difference between the oscillator and the line attractor is that the oscillator also includes the variable ω which controls the speed with which the system traverses the attractor.

- This kind of attractor is useful for describing repetitive behaviours (as in the lamprey model below).
- Notes:
  - Attractors can be subsystems of a system being modeled (e.g., there is an integrator that is a part of the otolith model)
  - Dynamic stability is important for a number of reasons.
    - it provides a kind of short-term memory that can aid in the processing of complex problems.
    - it allows the system to react to environmental changes on multiple time scales (stability permits systems to act on longer time scales than it might otherwise, which is important for certain kinds of behaviors such as prediction, navigation, hunting, etc.)
    - stability can make the system more robust (i.e., more resistant to undesirable perturbations.) This makes such networks good at noise reduction. Because
    - the same mechanisms that support noise reduction make attractor networks suitable for categorization tasks, as has been emphasized by Hopfield (note: we can construct networks with stable states non-evenly distributed around the space, just as we may have category clusters).
  - there are kinds of attractors we haven't examined yet, but could (chaotic (done Lorenz elsewhere), torus, etc.)
  - Attractors themselves can be thought of as dynamic. E.g., perhaps reaching for a target location is equivalent to forming an attractor point at that location in motor space. Then unexpected perturbations could be easily over come.

9.3 Lamprey model

- The lamprey example implements a cyclic attractor (simple harmonic oscillator).
- There are two techniques of interest in this example.
- The first is that we assure the stability of the function representation over the lamprey by explicitly damping out higher-order terms (high-frequencies) in the dynamics matrix.
- The second is that we introduce an intermediate level of representation between the 'higher' function representation and 'lower' neural representation.
- First, we have the observed swimming motions:

$$s(z,t) = A\sin(\omega t - kz). \tag{9.2}$$

- This gives an expression that describes the muscle tensions that are needed to give rise to these observed swimming motions:

$$T(z,t) = \kappa(\sin(\omega t - kz) - \sin(\omega t)), \tag{9.3}$$

- We then define a representation of the dynamic pattern of tensions in terms of the coefficients $x_n(t)$ and the orthonormal spatial harmonic functions $\Phi_n(z)$:

$$T(z,t;\mathbf{x}) \quad = \kappa \left(x_0 + \textstyle\sum_{n=1}^{N} x_{2n-1}(t) \sin(2\pi nz) + x_{2n}(t) \cos(2\pi nz) \right). \tag{9.4}$$

- Let's write the error

$$E = \left\langle [T(z,t) - \hat{T}(z,t;\mathbf{x})]^2 \right\rangle_{t,z}, \tag{9.5}$$

  Solving (9.5) results in the expression:

$$\begin{aligned} E \quad = \quad & \left\langle \frac{1}{2}(x_1 + \cos(\omega t))^2 + \frac{1}{2}(\sin(\omega t) + x_0)^2 \right. \\ & \left. + \left(\frac{x_0 + x_2}{2} \right)^2 + \left(\frac{x_0 - x_2}{2} + \sin(\omega t) \right)^2 + \sum_{n=3} x_n^2(t)_t \right\rangle_t . \end{aligned} \tag{9.6}$$

- To ensure that this error goes to zero over time, each of the terms in this expression must also go to zero.
  - The third term suggests that we need to damp the sum of x_0 and x_2 over time to keep the error small.
  - Together with the fourth term, it also suggests that we must enforce the constraint $x_0 = -x_2 = -\sin(\omega t)$.
  - The last term suggests all higher order coefficients must go to zero too.

- We can see that the first and second terms together identify an oscillator since

$$\begin{aligned} x_1 &= -\cos(\omega t) \\ x_0 &= -\sin(\omega t). \end{aligned}$$

- These dynamics can be written in standard control theory form as

$$\begin{aligned} \dot{x}_0 &= -\omega x_1 \\ \dot{x}_1 &= \omega x_0 \end{aligned}$$

  or,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x},$$

  where

$$\mathbf{A} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}.$$

  i.e., a cyclic attractor.

- Skipping a number of important steps... (see the text)
- As mentioned, the final term tells us that we must damp the higher-order terms in the orthogonal representation. We can combine these damping terms in a single matrix. This matrix then damps all sources of error introduced by our representation:

$$\mathbf{A}_{damp} = \begin{bmatrix} -\alpha_0 & 0 & -\alpha_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\alpha_0 & 0 & -\alpha_0 & 0 & 0 \\ 0 & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 & \ddots \end{bmatrix}.$$

- We have now effectively defined the dynamics matrix, $\mathbf{A}_D$, so let us turn to the input matrix, $\mathbf{A}_I$. This matrix is essential for starting the swimming motion, and could also be used to increase or decrease the amplitude of the swimming wave. For convenience, we have chosen startup dynamics defined by the startup matrix $\mathbf{A}_I$:

$$\mathbf{A}_I = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

  Substituting this matrix into (9.7) and allowing $\mathbf{u} = \mathbf{x}$ shows that the resulting equation is $\dot{\mathbf{x}} = \mathbf{x}$, giving an exponential startup while satisfying the constraints imposed by (9.6).

- Putting these matrices together, we can create a set of dynamical equations that result in the desired behavior:

$$\dot{\mathbf{x}} = \mathbf{A}_D\mathbf{x} + \mathbf{A}_I\mathbf{u} = [\mathbf{A}_x + \mathbf{A}_{damp}]\mathbf{x} + \mathbf{A}_I\mathbf{u}(t - t_s), \qquad (9.7)$$

- The second insight to be gained from this model is that we define the model over an 'intermediate' representation. This representation lies between the function representation and the neural representation. Essentialy it is like a population-level representation (where each tuning curve represents the activity of the neurons at one vertebrae). This is very useful for simulations, as it is significantly cheaper than a full neuron simulation by more realistic than simulating the functions directly.
- A simple choice for this intermediate level representation is to use one Gaussian encoding function for each vertebrae segment. We can then proceed as usual:

$$T(t, z) = \sum_i a_i(t)\phi_i(z), \qquad (9.8)$$

  where we can find the decoding functions by minimizing the error as before. The $a_i(t)$ is a scalar that can be thought of as the 'segment activity', defined by

$$a_i(t) = \left\langle \tilde{\phi}_i(z) T(t, z) \right\rangle_z. \qquad (9.9)$$

  Together, (9.8) and (9.9) form an overcomplete representation of $T(z, t)$.

- To use this representation, while preserving the higher-level dynamics, we can construct a projection operator between the higher-order orthogonal space and this space. This operator effectively allows us to move between these spaces:

$$\Theta = \left[\tilde{\boldsymbol{\phi}}\mathbf{\Phi}\right]^{-1}.$$

- We can use Θ to transform the dynamical equations for the Fourier amplitudes, $\mathbf{x}$, given by (9.7), to dynamical equations in the intermediate space, $\mathbf{a}$:

$$\dot{\mathbf{a}} = \mathbf{A^a}_D\mathbf{a} + \mathbf{A^a}_I\mathbf{a}(t - t_s), \tag{9.10}$$

  where

$$\begin{aligned}\mathbf{A^a}_D &= \Theta^{-1}\mathbf{A}_D\Theta\\ \mathbf{A^a}_I &= \Theta^{-1}\mathbf{A}_I\Theta.\end{aligned}$$

  We can now simulate the lamprey's swimming in the intermediate space using this equation. Not surprisingly, the lamprey swims as expected.

- Note that NESim doesn't support this kind of intermediate representation.