# Lecture 8: Dynamic transformations

## 8.1 'Neural' control theory

- History:
  - Cybernetics (40s) - neurophysiologists (Warren McCulloch and Arturo Rosenblueth), one a mathematicians (Norbert Weiner) and one an engineer (Julian Forrester). Too behaviourist, classical control.
  - GOFAI (60s-) - representation, and computation. Turing machines, von Neumann architecture, etc. Ignores time
  - Contemporary (90s-) - time is essential, obvious to neurophysiologists. Idea: reintroduce modern control theory

### 8.1.1 Standard control theory

- There are two equations that effectively summarize linear control theory (See slide):

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{8.1}$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \tag{8.2}$$

- these are the *state equations* of a system because the vector $\mathbf{x}(t)$ consists of variables that together describe the (internal) state of the system.
- *state vector*, $\mathbf{x}(t)$, serves to summarize the effects of all past input
- $\mathbf{A}$ is the dynamics matrix
- $\mathbf{B}$ is the input matrix
- The transfer function, $h(t)$ in standard control theory is integration.
- In the Laplace domain, $h(s) = \frac{1}{s}$.

### 8.1.2 Neural control theory

- In neurons, the synaptic dynamics dominate the overall population dynamics. So, we need to use the intrinsic synaptic dynamics to characterize 'neural' control theory.
- recall, PSCs are given by

$$h(t) = \frac{1}{\tau} e^{-t/\tau}$$

- Laplace of this

$$h(s) = \frac{1}{1 + s\tau}$$

- we can now put this into a 'neural' control diagram, analogous to the stardard one above (see slide).
- we leave out the $\mathbf{C}$ and $\mathbf{D}$ matrices because they are accounted for by input matrices in populations this system is connected to.
- Now, we can rewrite the state equation with this new transfer function

$$\begin{aligned} \mathbf{x}(s) &= \frac{1}{1+s\tau}\left[\mathbf{A}'\mathbf{x}(s)+\mathbf{B}'\mathbf{u}(s)\right] \\ &= \frac{\tau^{-1}}{\tau^{-1}+s}\left[\mathbf{A}'\mathbf{x}(s)+\mathbf{B}'\mathbf{u}(s)\right]. \end{aligned}$$

So,

$$\begin{aligned} (\tau^{-1}+s)\mathbf{x}(s) &= \tau^{-1}\left[\mathbf{A}'\mathbf{x}(s)+\mathbf{B}'\mathbf{u}(s)\right] \\ s\mathbf{x}(s) &= \tau^{-1}\left[\mathbf{A}'-\mathbf{I}\right]\mathbf{x}(s)+\tau^{-1}\mathbf{B}'\mathbf{u}(s). \end{aligned} \tag{8.3}$$

- Noting, in addition,

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s)+\mathbf{B}\mathbf{u}(s)$$

for standard control theory, we can solve to find $\mathbf{A}'$ and $\mathbf{B}'$ as

$$\mathbf{A}' = \tau\mathbf{A}+\mathbf{I} \tag{8.4}$$

$$\mathbf{B}' = \tau\mathbf{B}, \tag{8.5}$$

- Of course, this answer depends on our assumed model for PSCs, but the techniques are the same for any model.
- This, then, is the desired 'translation' between standard and neural control theory

### 8.1.3 A generic neural subsystem

- we can use this derivation to describe a generic neural subsystem: i.e., a theoretical subsystem that can be mapped to any spiking neural population involved in some dynamic transformation.
- First, we can draw a subsystem that simply integrates the PSCs with some weights to give rise to neural spiking (see slide). This is a standard, though control-like description of neural population behavior. This is a description of 'basic-level' behaviour.
- However, we can also describe the behaviour of the population without worrying about spikes, but just about the dynamics of state vector being represented (see slide). This is a description of 'higher-level' behaviour.
- Now, of course, we can bring these two descriptions together in a 'generic' neural subsystem description that includes both - plus our characterization of representation (see slide). In effect, this diagram summarizes our entire approach.

- We can see directly from this diagram that the standard form for connection weights for a population implementing certain dynamics transformations is

$$\omega_{ij}^{\alpha\beta} = \left\langle \tilde{\boldsymbol{\phi}}_i^{\alpha} \mathbf{M}^{\alpha\beta} \boldsymbol{\phi}_j^{\alpha F\beta} \right\rangle_m$$

- As well, we can restate the three principles we saw at the very beginning of class in a precise way (see slides)

- **Principle 1**: *Neural representations are defined by the combination of nonlinear encoding and weighted linear decoding*.

- Neural encoding is defined by

$$\sum_n \delta(t - t_{in}) = G_i \left[ \alpha_i \left\langle \tilde{\boldsymbol{\phi}}_i \mathbf{x}(t) \right\rangle_m + J_i^{bias} \right].$$

  Neural decoding is defined by

$$\hat{\mathbf{x}}(t) = \sum_i a_i(\mathbf{x}(t)) \boldsymbol{\phi}_i^{\mathbf{x}},$$

  where

$$\begin{aligned} a_i(\mathbf{x}(t)) &= \sum_n h_i(t) * \delta(t - t_{in}) \\ &= \sum_n h_i(t - t_{in}). \end{aligned}$$

  In both cases, $i$ indexes neurons in population and $n$ indexes spikes transmitted from one population to another.

- **Principle 2:** *Transformations of neural representations are functions of the variable that is represented by the population. Transformations are determined using an alternately weighted linear decoding*.

- Assuming the encoding in principle 1, we can estimate a function of $\mathbf{x}(t)$ as

$$\hat{f}(\mathbf{x}(t)) = \sum_i a_i(\mathbf{x}(t)) \boldsymbol{\phi}_i^f,$$

  where, $a_i(\mathbf{x}(t))$ is defined as before. The only difference between this decoding and the representational decoding are the decoders themselves, $\boldsymbol{\phi}_i^f$.

- **Principle 3:** *Neural dynamics are characterized by considering neural representations as control theoretic state variables. Thus, the dynamics of neurobiological systems can be analyzed using control theory*.

- Allowing $\mathbf{x}(t)$ to be a state variable and $\mathbf{u}(t)$ to be the input, we have the following general expression for the encoding:

$$\sum_n \delta(t - t_{in}) = G_i \left[ \alpha_i \left\langle \tilde{\boldsymbol{\phi}}_i \left( h_i(t) * [\mathbf{A}'\mathbf{x}(t) + \mathbf{B}'\mathbf{u}(t)] \right) \right\rangle_m + J_i^{bias} \right].$$

## 8.2 Controlling eye position

- There has been a lot of work done the the 'neural integrator', part of the brain that controls eye position. There is part of the brain (NPH and vestibular nucleus), that stores the current desired eye position in the absence of cues.
- This is a nice example because integration, in general is a useful and difficult problem to solve. And, it's simple to formulate.
- We can draw a block diagram for this control system (see slide)
- Let's redo the derivation of what the behaviour of the system will be, this time assuming that the input and recurrent time constants can be different:

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{y}(s)$$

and

$$\begin{aligned}
\mathbf{x}(s) &= \frac{1}{1+s\tau_x}\mathbf{A}'\hat{\mathbf{x}} + \frac{1}{1+s\tau_y}\mathbf{B}'\hat{\mathbf{y}} \\
(1+s\tau_x)\mathbf{x}(s) &= \mathbf{A}'\hat{\mathbf{x}} + \frac{1+s\tau_x}{1+s\tau_y}\mathbf{B}'\hat{\mathbf{y}} \\
s\mathbf{x}(s) &= \frac{1}{\tau_x}\left(\mathbf{A}'\hat{\mathbf{x}} - \mathbf{x}\right) + \frac{1}{\tau_x}\frac{1+s\tau_x}{1+s\tau_y}\mathbf{B}'\hat{\mathbf{y}}
\end{aligned}$$

now assume $\mathbf{x} = \hat{\mathbf{x}}$, $\mathbf{y} = \hat{\mathbf{y}}$, $\mathbf{A}' = 1 + \mathbf{A}\tau_x$ and $\mathbf{B}' = \mathbf{B}\tau_x$

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x} + \frac{1+s\tau_x}{1+s\tau_y}\mathbf{B}\mathbf{y}$$

this is the same as the original dynamics equation if $\tau_x = \tau_y$. This is important to know since it tells us that having non-equal synpatic time constants on the input and recurrence can significantly affect the system.

- Now we can solve for the two matrices for the integrator (actually just scalars in this case) now:

$$\begin{aligned}
B' &= \tau \\
A' &= 1
\end{aligned}$$

- Now we can put this in a circuit:

$$a_j(t) = G_j\left[\alpha_j\left\langle x(t)\tilde{\phi}_j\right\rangle + J_j^{bias}\right].$$

Substituting for $x(t)$, and leaving the higher-level representation of the input, $u(t)$, gives

$$\begin{aligned}
a_j(t) &= G_j\left[\alpha_j\left\langle h(t) * \tilde{\phi}_j\left[A'\sum_i a_i(t)\phi_i^x + B'u(t)\right]\right\rangle + J_j^{bias}\right] \quad (8.6) \\
&= G_j\left[h(t) * \left[\sum_i \omega_{ji}a_i(t) + B'\tilde{\phi}_j u(t)\right] + J_j^{bias}\right], \quad (8.7)
\end{aligned}$$

where $\omega_{ji} = \alpha_j A' \phi_i^x \tilde{\phi}_j$.

- That was easy... but, let's think about the integrator for a minute. If we expect any error at all in our representation of $x(t)$, we won't be able to build a perfect integrator (and of course, we expect error - from the repn, from spikes, and from noise).

- Think of error in $\hat{x}$ as being captured by some gain, $kx$. This gain is playing the role of $A$ in the circuit. And we know the circuit will be sensitive to any deviations of $A$ away from $0$ (or $1$ in the neural version), because

$$\begin{aligned} x(s) &= \frac{1}{s}(Ax(s) + Bu(s)) \\ &= \frac{A}{s}x(s) + \frac{B}{s}u(s). \end{aligned}$$

- That is, if $A$ is non-zero, we get moved away from the current value of $x$ even with no input. So it acts like a rate constant, i.e.,:

$$A = -\frac{1}{\tau_{\mathit{eff}}}.$$

  Since we know the equivalent value of $A$ in the neural circuit, we know that

$$\begin{aligned} -\frac{1}{\tau_{\mathit{eff}}} &= \frac{A' - 1}{\tau} \\ \tau_{\mathit{eff}} &= \frac{\tau}{1 - A'}. \end{aligned}$$

- So (see slide)

  1. the synaptic time constant, $\tau$, of constituent neurons affects the integration abilities of the circuit of which they are a part;
  2. that the goodness of the $x(t)$ representation affects the integration abilities of the circuit.

- a longer $\tau$ makes for a longer effective time constant (note that a perfect integrator has an infinite effective time constant), so neurons with longer synaptic time constants (e.g., neurons with NMDA receptors) are more suited to integration

- also, as $A'$ nears unity, the effective time constant becomes infinite. Thus, setting $A' = 1$ means that all of the integration error is a result of representation error. So, the better our representation of $x(t)$ (and thus the closer $A'\hat{x}(t)$ matches $A'x(t)$), the better job we can do of implementing stable dynamics

- So representation error is very important for dynamics.

- But note that error is a function of $x$. This is why it's often useful to graph (as you have been doing) $\hat{x} - x$.

- The standard linearity graph is useful for understanding dynamics (see slide). You can trace the temporal evolution of the system on this graph.
- The 'difference' graph makes it easy to see stable points, and drift speeds. Note that the drift velocity is proportional to the magnitude of this curve.
- Now we can measure drift velocity and look at these kinds of properties to understand the behavior as we change network parameters. The slides show changing the number of neurons and changing the RC time constants of neurons in the population (see slide).
- Just for fun we can now build a network that dynamically changes $A'$, thereby affecting the parameters that control the dynamics. This essentially makes the integrator circuit into a tunable filter. (see slides):
- Let

$$A'(t) = \sum_l b_l(t)\phi_l^{A'},$$

and substitute this into (8.6) to give

$$a_j(t) = G_j\left[\alpha_j\left(h(t) * \tilde{\phi}_j\left[\sum_l b_l(t)\phi_l^{A'}\sum_i a_i(t)\phi_i^x + B'u(t)\right]\right) + J_j^{bias}\right] \quad (8.8)$$

We know that to perform a multiplication between the recurrent signal, $x(t)$, and the control signal, $A'(t)$, we can introduce a new population whose dimensionality is the sum of those being multiplied (see section **??**). So let

$$\mathbf{p}(t) = \sum_m c_m(t)\boldsymbol{\phi}_{\boldsymbol{m}}^{\mathbf{p}}$$

be such a population. So, we can write (8.8) as

$$a_j(t) = G_j\left[\alpha_j\left(h(t) * \tilde{\phi}_j\left[\sum_m c_m(t)\boldsymbol{\phi}_{\boldsymbol{m}}^{\mathbf{p}} + B'u(t)\right]\right) + J_j^{bias}\right], \quad (8.9)$$

where

$$c_m(t) = G_m\left[\alpha_m\left(x(t)\tilde{\phi}_m^{c_1} + A'(t)\tilde{\phi}_m^{c_2}\right) + J_m^{bias}\right] \quad (8.10)$$

$$= G_m\left[\alpha_m\left(\sum_i a_i(t)\phi_i^x\tilde{\phi}_m^{c_1} + \sum_l b_l(t)\phi_l^{A'}\tilde{\phi}_m^{c_2}\right) + J_m^{bias}\right] \quad (8.11)$$

- The nice thing here is that we can figure out exactly how single cell properties affect network behavior. This has often been a point of contention in neuroscience (is integration a network effect, or a cellular effect?). Using this approach we can try to figure out what parts of each give rise to the observed behaviour.
- Also, (see slide) we can tune our model to match what is known about integrators in real systems. E.g., we can incorporate centripetal only drift. And change tuning curve distribution to match those systems.

## 8.3 Working memory

- I have essentially nothing new to say about this example, since it's exactly the same dynamics as the integrator, but over function representation. So, the derivation is the same, using vectors (the function coefficients) instead of scalars.
- Interestingly, this network works well, and exhibits behaviour that people haven't been able to incorporate in the past (See slides).