13/11/2018

motor

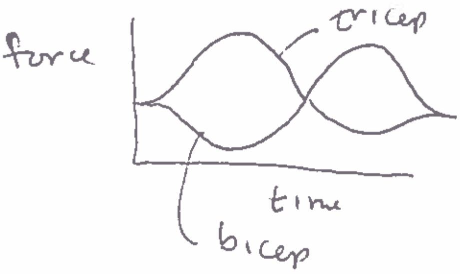
The only goal of all organisms: learn to turn intention into action

including reflexes

1. how do you learn to turn intention into action at the algorithmic level?
2. how do you implement that in neural circuitry?
3. what's the role of feedback?

Is this a hard problem? What's the complexity of the input-output transformation?

- we have ~650 muscles
- we have to generate 650 time-dependent signals



- this has to be done with a network of spiking neurons!
- "complexity" = "number of parameters" (loosely)

A dumb solution

$$f(t) = \sum_n a_n \cos(w_n t + \theta_n)$$

forward model

- just need a network to generate sines and cosines
- small assemblies of neurons oscillating at different frequencies

Intention $\rightarrow$ $a_n(I)$, $\theta_n(I)$

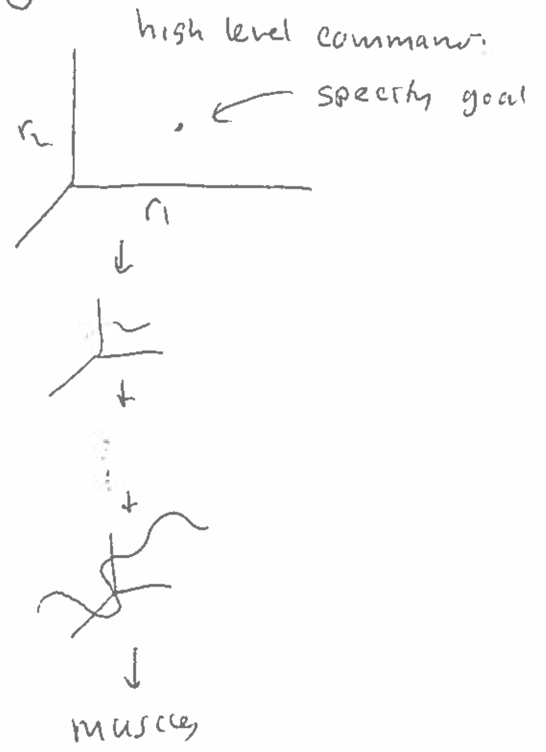
inverse model

- If the intention is a trajectory,
  the inverse model is an inverse Fourier transform.

  $\min \left[f(t) - \sum_n a_n \cos(\omega_n t + \theta) \right]^2$

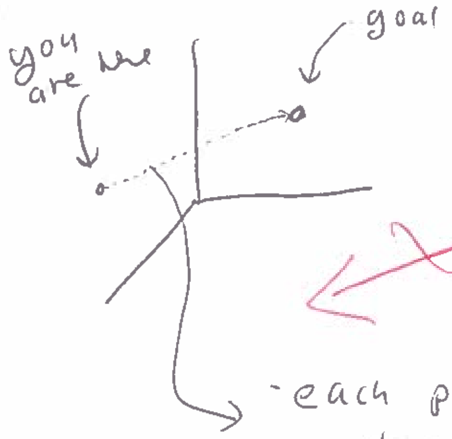
- ~~but usually (often??)~~
  but intention is usually an endpoint goal
- mapping endpoint goal to trajectory is nontrivial.
  muscles and limbs ~~have~~ have mass and nontrivial properties, and everything changes depending on level of fatigue
- rest of this talk: approaches to solving this problem

general picture



end-point approach (Emilio Bizzi)

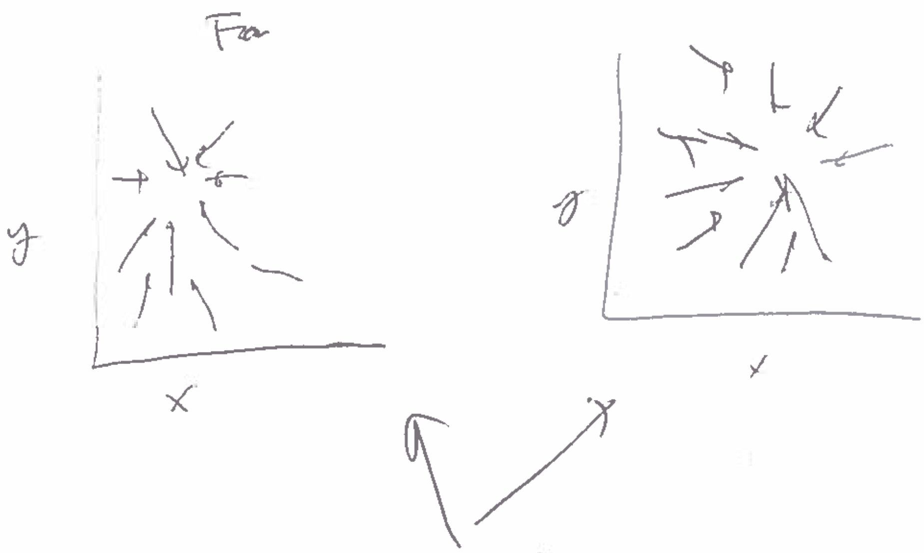
specify goal ~~[illegible]~~ in physical space



Still have a complicated transformation

- each point corresponds to muscle equilibrium

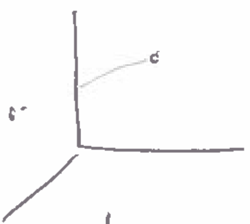
- consistent with experiments in frog



each point corresponds to stimulation of different points in the frog spinal cord

(1992)

Command



↓

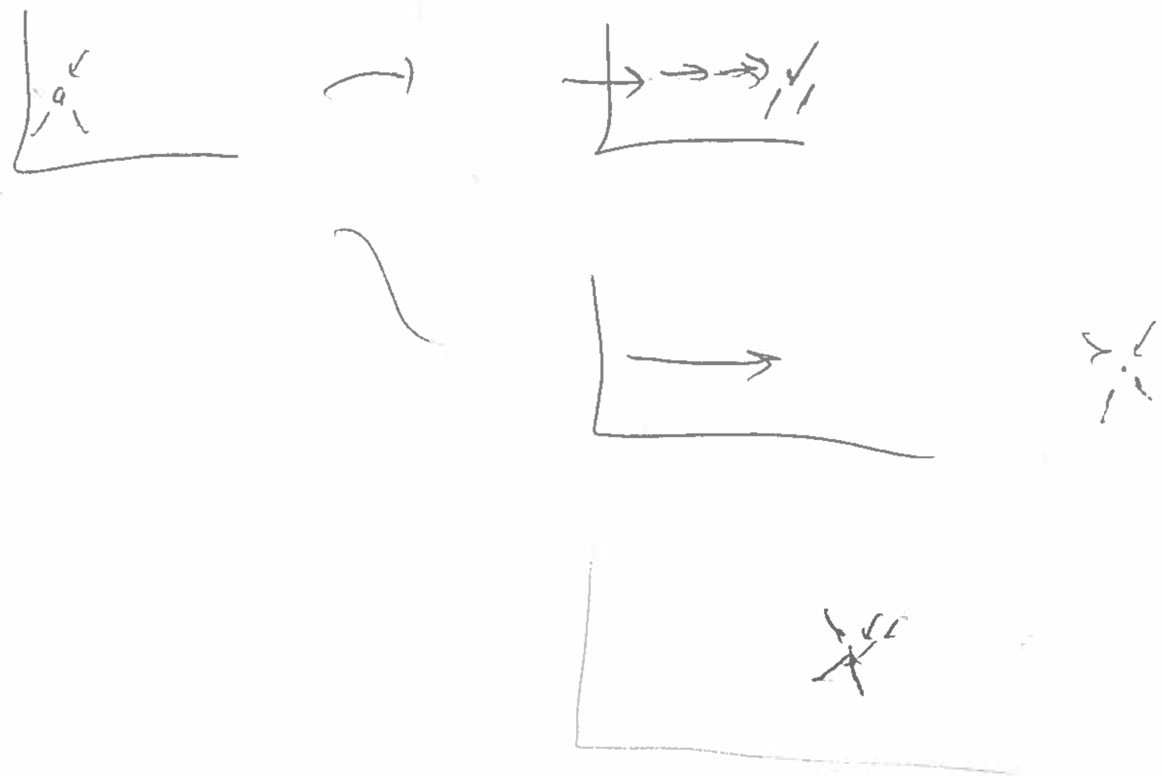
motor ctx

↓

spinal cord

Still have a complicated transformation but at least you don't have to worry about dynamics.

- endpoint 序 approach is more complicated than advertised. in particular, moving the endpoint faster doesn't give you a simple transformation. imagine instantaneous mu



Optimal Control

$$\dot{x}_i = f\left(\sum_j w_{ij} x_j\right) + u_i(t)$$

$$y_\mu = \sum_{j=1}^{N} D_{\mu j} x_j$$

$\mu = 1, \ldots, 3$ point in 3-D space

Choos $u_i(t)$ such that: $y_\mu(T) = y_\mu^*$

$\int dt\, u_i(t)^2$ as small as possible

- doesn't quite make sense, but it's ~~easier~~ not so hard
- what we really want to do is minimize some function of y_μ

~~Speed~~

speed	$\lvert\dot{y}_\mu\rvert^2$	bad idea
acceleration	$\lvert\ddot{y}_\mu\rvert^2$	better
jerk	$\lvert\dddot{y}_\mu\rvert^2$	good description of simple trajectories

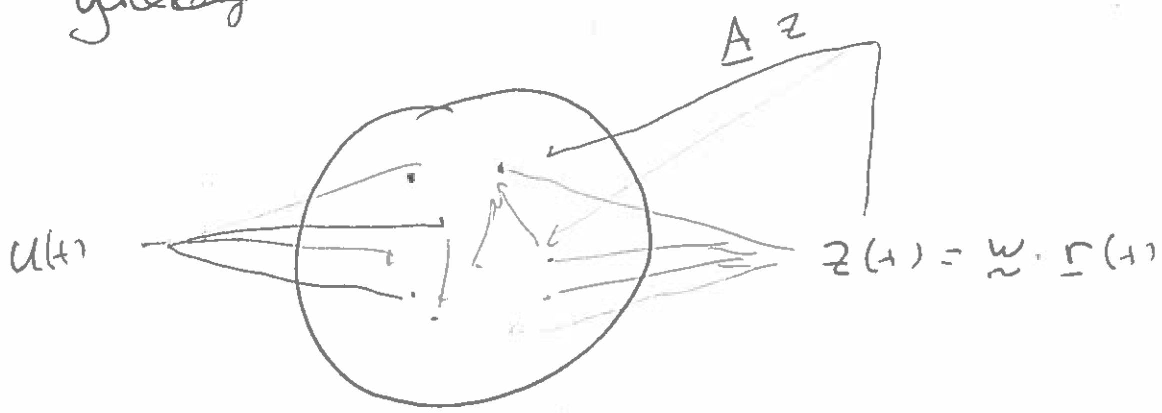
infinite time problems. minin

$$\left| \underline{\underline{D}} \cdot \underline{X}_{\infty}(\underset{\sim}{u}) - y^{t} \right|^{2} + \frac{\lambda \, \underline{u} \cdot u}{2}$$

- looks like endpoint approach of Bati!!!

- these are principles for constructing optimal circuit.
- but ultimately everything has to be learned
- revisit sine and cosine idea, but in a more incomprehensible way

greedy



$$\tau \dot{V}_i = -V_i + \sum_j J_{ij} \underbrace{r_j}_{\tanh(v_j)} + A_i z + B_i u$$

$$z_q(t) = \sum_j w_j r_j$$

$$\tau \dot{\underset{\sim}{V}} = -\underset{\sim}{V} + \underline{\underline{J}} \cdot \underline{r} + \overset{a}{\cancel{A}}\, z + \underline{B}\, u$$

$$z = \underline{w} \cdot \underline{r}$$

$$\underline{r} = \tanh(\underline{v}) \qquad (r_i = \tanh v_i)$$

goal: train $\underset{\sim}{w}$ so that $\underline{w} \cdot \underline{r} \cong z^*(t)$

easy approach:

feed in z^*; train weights

greedy algorithm: ~~$(\underline{w}\cdot r - z^*)^2$~~

$$\delta = z^* - \underline{w}\cdot r$$

minimum $\frac{\delta^2}{2} = \frac{1}{2}(z^* - \underline{w}\cdot\underline{r})^2$

$$\frac{d\delta^2/2}{\partial \underline{w}} = -(z^* - \underline{w}\cdot\underline{r})\,\underline{r} = -\delta\underline{r}$$

$$\dot{\underline{w}} = -\eta\frac{d\,\delta^2/2}{d\underline{w}} = \eta\,\delta\underline{r}$$

- conditions for it to work

r_i complex & differ

$$\tau\dot{\underline{v}} = -\underline{v} + \underline{\underline{J}}\cdot\tanh(\underline{v}) + \underline{A}z^* + \underline{B}u$$

$\underline{\underline{A}}, \underline{\underline{J}}$ small $\quad \underline{v} \to 0$

$\underline{A}$ huge: $\quad V_i = A_i z \quad r_i = \pm 1 \quad$ all doing the same thing.

A small, J just right: chaotic

A must be big enough to control chaos

even if you can train, no gauruntee of stability!

example: 1-D, ignore tanh

$$\tau \dot{V} = -V + a z^* \qquad z^* = z_0$$

$$V^* = a z_0$$

$$z = \frac{V}{a} = z_0 \qquad w = \frac{1}{a}$$

$$\tau \dot{V} = -V + a\frac{V}{a} = -V + V = 0 \qquad \text{neutral stability!}$$

$$\tau \dot{V} = -V + \varepsilon(V - V_0) + a z^* \qquad z^* = z_0$$

$$= -(1-\varepsilon)V + a z^* - \varepsilon V_0$$

$$V^* = \frac{a z_0 - \varepsilon V_0}{1-\varepsilon}$$

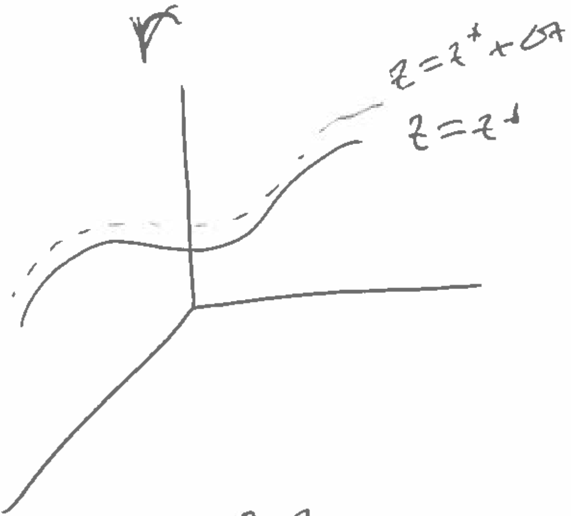
$$z_0 = w V^* \Rightarrow w = \frac{(1-\varepsilon) z_0}{a z_0 - \varepsilon V_0}$$

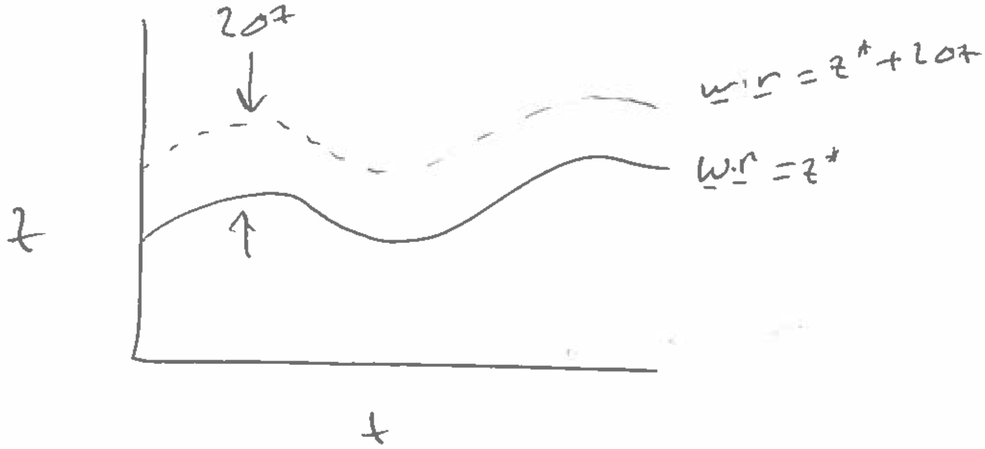
$$\tau \dot{V} = -(1-\varepsilon)V + \frac{a(1-\varepsilon) z_0}{a z_0 - \varepsilon V} V$$

$$= -(1-\varepsilon)\left[1 - \frac{a z_0}{a z_0 - \varepsilon V_0}\right] V$$

$$= \underbrace{\frac{\varepsilon(1-\varepsilon) V_0}{a z_0 - \varepsilon V_0}}_{\text{can be positive}} V$$

more generally





in addition, learning is slow & brittle

- brittleness can be fixed by feeding back the true signal

- can also use RLS (recursive least squares) aka FORCE learning

$$\text{minim} \sum_{\tau=1}^{T} \left(z^d(t) - \underset{\sim}{w}^{(t)} \cdot \underline{r} \right)^2$$

$$\underline{w}(t) = \underbrace{\left(\sum_{\tau=1}^{T} \underline{r}(t)\, \underline{r}(t) \right)^{-1}}_{\underline{P}} \cdot \underbrace{\sum_{\tau=1}^{T} \underline{r}(t)\, d(t)}_{q}$$

$$\underline{w}(T+1) = \left(\sum_{\tau=1}^{T+1} \underline{r}(t)\, \underline{r}(t) \right)^{-1} \cdot \sum_{\tau=1}^{T} \underline{r}(t)\, d(t)$$

$$\left(\sum_{\tau=1}^{T} \underline{r}(t)\, \underline{r}(t) + \underline{r}(T+1)\, \underline{r}(T+1) \right)^{-1}$$

$\underline{q}(T) - d(T+1)\,\underline{r}(t)$

$$= \left(\underline{P}^{-1} + \underline{r}(t+1)\, \underline{r}(t+1) \right)^{-1}$$

$$= \underline{P}^{(T)} - \frac{\underline{P}^{(T)} \cdot \underline{r}(t+1)\, \underline{r}(t+1) \cdot \underline{P}}{\underline{r}(t+1) \cdot \underline{P}(T) \cdot \underline{r}(t+1)}$$

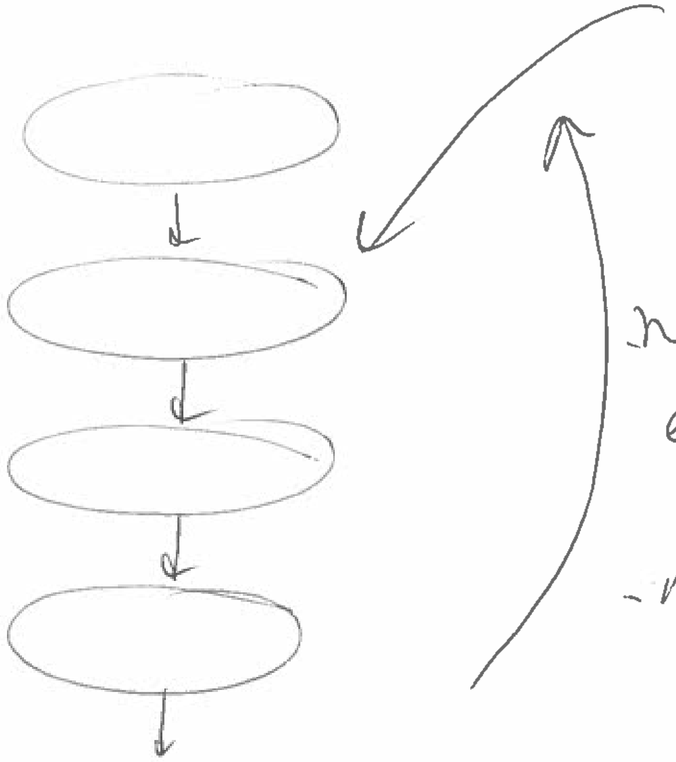
$$\parallel$$

$\underline{P}(T+1)$

- learns fast!!
- tends to be stable!!

---

not clear how to chain together multiple network



also want to modify recurrent weights

need to propagate error all the way back

- not biologically plausible!!

$z(t)$ target $= z^*$

but this is what the brain does!!

feedback

$\dot{x} = -x + f(x) + u(t) +$

~~keep f(t) on~~

- $z^*(t)$ is always present:
- however, it comes with a delay and noise
- ~~need to learn how~~
- ~~other~~ this makes the learning rule hard
  - not a lot of work in neuroscience, at least that I know of

Summary

- motor control is hard because of the inverse problem: given a goal, generate correct muscle activation
- endpoint ~~[illegible]~~ hypothesis: still need to learn something, but it's easier than learning dynamics. ~~[illegible]~~ doesn't really work for fast movement
- reservoir approach: glorified sines + cosines. gotta learn
- unsolved: multiple dynamic areas, feedback w/ delays + noise.