Now we take a slightly tangential route

- Computational motor control
- Control in biological systems
- There’s something more than the control of the single joint
- Study how control is done in biology ↔ study how control has to be done in robotics

Also, something we haven’t discussed yet

- The study of the motor system is also the study of dynamics
  
  \[ F = ma \text{ instead of } x = f(x, v) \]
Optimization principles

- Don’t describe the kinematics directly, rather the movement is described abstractly
- Global measure (cost):
  - Total efficiency
  - Smoothness
  - Accuracy
  - Duration

Trajectory generation

- This fits in “front” of the “single joint” controller we’ve seen so far
- Q: how do we generate a sequence of reference points for the controller?

On the trajectory generation

- Note that the feedback controller by itself doesn’t necessarily generate suitable trajectories especially for a complex kinematic structure (e.g. arm)

Most studied behavior: reaching

- Despite variation of movement direction, starting point, etc. there are some kinematic invariants; most notably:
  - Straight trajectory
  - Bell shaped velocity profiles

Further…

- There are variation from straightness especially at the periphery of the workspace
- Why is it so surprising that trajectories are straight:
  - Joints are rotational → easier to get curved trajectories

In addition

- There might be differences (from the bell-shaped profile) when feedback plays a role
  - See a moving object and try to intercept it
- Intuition: when "open-loop" trajectories are stereotyped otherwise they get distorted by feedback
In formulas

\[ x(t) \quad t \in [0,T] \]

Cost \( g(x(t),t) \) instantaneous cost

\[ J = \int_0^T g(x(t),t) dt \]

\[ g \] represents what is costly for us

Minimum \( J \)

In general – 2 techniques:

- Dynamic programming
  - Computing all possible state transitions and cumulating the cost, then searching trajectories that minimize the cost \( \rightarrow \) need to discretize the state space (curse of dimensionality)
  - Variation calculus: finding \( x(t) \) such that \( J \) is minimized \( \rightarrow \) analytical

Examples

- Minimum Jerk (proposed by Hogan):
  \[ J = \int_0^T \left( \frac{d^3 x}{dt^3} \right)^2 dt \]
  - By calculus of variation it was shown that:
    \[ x(t) = x_0 + (x_f - x_0)10T \left( \frac{1}{T} \right)^3 - 15T\left( \frac{1}{T} \right)^4 + 6T\left( \frac{1}{T} \right)^5 \]
  - \( x \) is straight: obvious from the equation
Elaborations

• Don’t want to specify the duration of the movement

\[ J = \int \left[ \gamma \left( \frac{d^3 x}{dt^3} \right)^2 + 1 \right] dt \]

• This model predicts durations correctly

Further elaborations

• Minimize torque change \( \rightarrow \) similar to jerk minimization but in dynamical conditions

\[ J = \int \sum_{i=1}^{N} \left[ \frac{d^3 \tau_i}{dt^3} \right]^2 dt \quad i \in [1..N] \]

• This model is due to Kawato

Considerations

• This description doesn’t imply that the CNS is actually optimizing anything

Other issues

• Hpc: use P5(t) as a movement primitive (computed on-line)
• Superimpose primitives (which primitives?)
• Incrementally update \( x, \dot{x} \) in feedback so that the system responds to perturbations
• Neural net solution \( \rightarrow \) in practice the neural net does the minimization
• VITE model: feedback + variable gain might obtain results similar to the optimization techniques

Internal models

• A system that mimics the behavior of a natural process
• Does the brain rely on internal models? (see Miall & Wolper paper)

• Types of models:
  • Forward models
  • Inverse models

Forward models

• Given the current state and input predict the next state of the system

• In physiology need to also estimate the state (measured, sensed) from the raw sensory input (it might be a complex computational problem – e.g. 3D from 2D information, etc.)
Prediction of the causal flow

- The forward model can be seen as a prediction (anticipation) of the causal flow
- Being “internal” it can be faster than reality
- Example: the prediction of the state of the motor system due to the outgoing motor commands

Example

Formally

\[ \min_{\hat{u}} \|\hat{y} - y\| < \varepsilon \]

Use of models: canceling sensory reafferences

- Important for distinguishing our own motion from the environmental motion

Inverse model

- More difficult: the underlying forward model can be a one to many, thus not invertible unless additional constraints are provided

Forward models again

- They’re always well defined
- They could be one to one or one to many
- Another example:
  - Kinematics: computing the position in space of the end-effector as a function of the joint angles
In biology

- Ego-motion cancellation in pursuing a target
- Efference copy: a copy of the command
- Corollary discharge: the prediction of a signal computed by the CNS

State estimation

- How can we (the CNS in fact) integrate motor and sensory information in estimate the state of the arm (for example)?
- Observer:

Internal feedback to overcome delays

- Feedback:
  - Robust, doesn’t require a precise model of the system to be controlled
  - Issue: it suffers from delays
- Feedforward:
  - Requires a precise model
  - Doesn’t care of delays since the control is computed in advance

Delays in the CNS

- We live delayed of 30-300ms!
- A fast arm movement can last around 200ms
- Feedback controllers are required!

The Smith predictor model

In practice

- A forward model + delay estimates the feedback signal
- This signal is compared with the delayed feedback and provides a correction due to feedback to the state estimation (slow, with some delay, low gain)
- State estimation proceeds open-loop otherwise directly from the model (fast, little delay)
Moreover...

- State estimation of course could be extended into prediction
- Humans can get to zero delay in tasks where the target follows a predictable trajectory

In essence

- Under certain conditions Kalman filter is optimal (linear system, quadratic cost, Gaussian noise)

\[
\begin{align*}
    x_{t+1} &= f(x_t, u_t) + k(y_t - g(f(x_t, u_t))) \\
    x_{t+1} &= f(x_t, u_t) + \xi_t \
    y_t &= g(x_t) + \eta_t
\end{align*}
\]

f is linear

g is linear

Learning the models

- What does it mean to learn the models?

\[
\begin{align*}
    \hat{y} &= f_w(u) \\
    \min_w \frac{1}{2} \| \hat{y} - y \|^2 &= \min_w \frac{1}{2} \| f_w(x) - y \|^2 
\end{align*}
\]

How do I get the samples?

- Direct-inverse modeling
- Feedback error learning
- Distal supervised learning
- Reinforcement learning
- ...

Direct-inverse

- Simply send “certain” inputs to the system and measure the output. Use the set of samples collected to find the min of the cost
  - If there are many solutions to the problem (e.g., redundancy) the direct-inverse approach is not well behaved
  - For linear or otherwise simple problems the approach can work
**Example**
- Archery problem: goal of the controller is to determine the angle

**Feedback error learning**
- Use something simpler to bootstrap learning of something more complicate

**Example**
- Distal supervised learning

**Reinforcement learning**
- Reduced feedback from the environment

**RL (2)**
- $r$ is a scalar, much harder problem than anything we've seen so far
- Interaction with the environment is explicit
- Link of RL to dynamic programming, in practice RL is an approximation of DP
- It can solve difficult problems and it can generate controllers that perform better than the teacher
Why is it so hard?

• Need to reconstruct a gradient from a scalar information (at best), in many cases information is even poorer (imagine playing chess: you only get information at the end of the game)

supervised learning (anything we've seen so far)

reinforcement learning (get only the magnitude of y)

\[ y \] starting point

\[ ||y|| \] starting point

RA 2008 target

supervised learning (anything we've seen so far)

reinforcement learning (get only the magnitude of y)

\[ y \] starting point

\[ ||y|| \] starting point

RA 2008 target