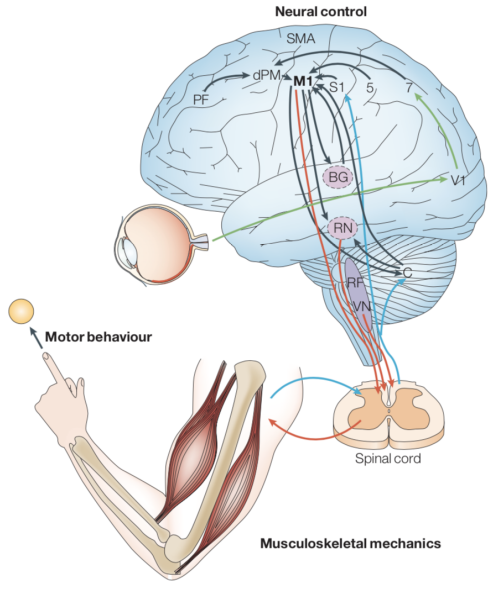
Neuromechanics of Human Motion

A (Soft) Introduction to OFC Modelling

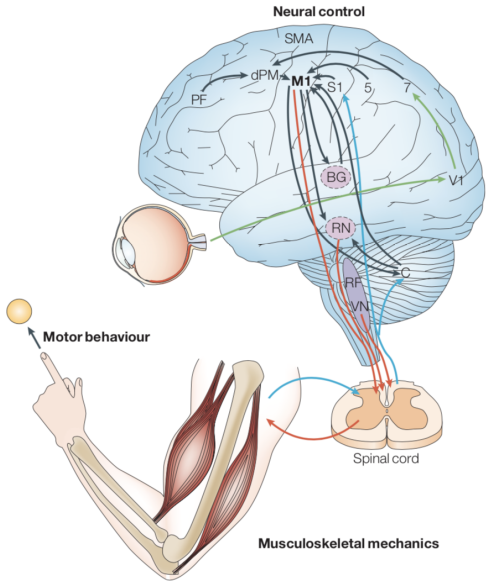
Joshua Cashaback, Ph.D.

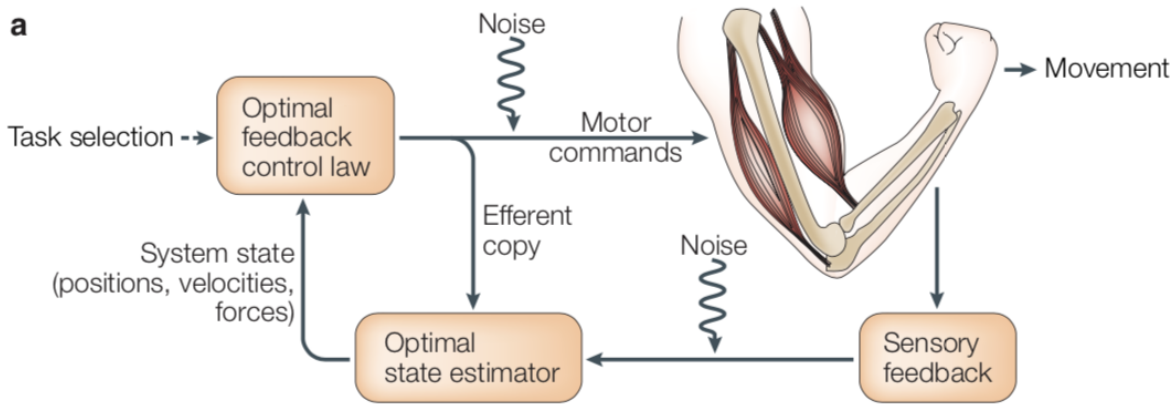
OFC Recap



(Scott, 2004)

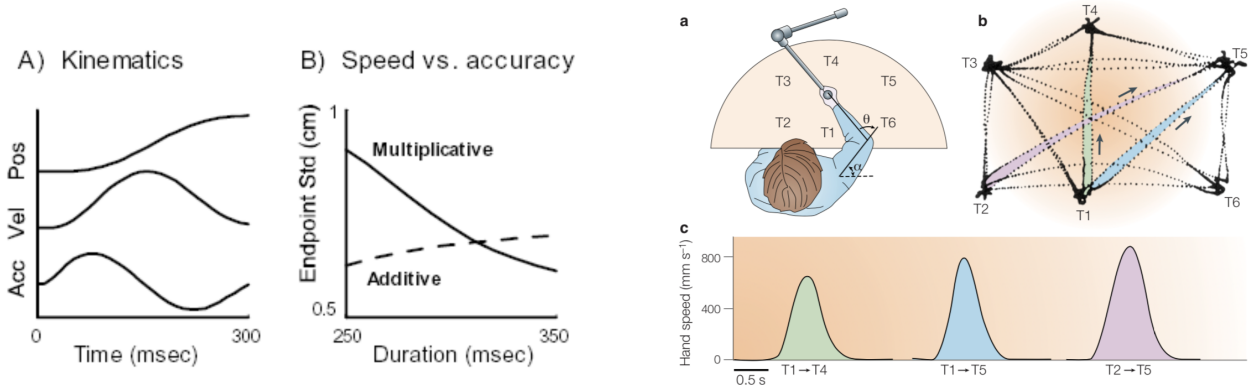
OFC Recap



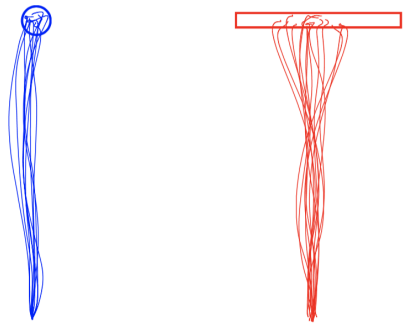


(Scott, 2004)

OFC Recap



speed-accuracy tradeoff and bell-shaped velocity(Scott, 2004; Todorov, 2005)



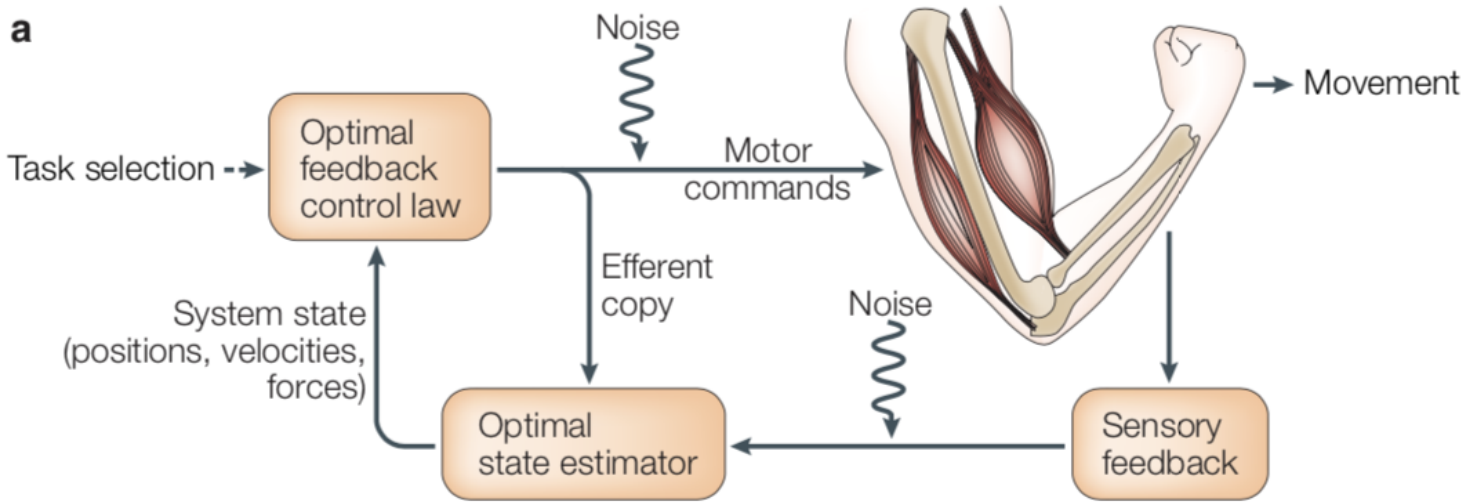
Minimum Intervention Principle (Nashed et al, 2012)

Lecture Objectives

1. General Concept of Linear Quadratic Gaussian Controllers
   a. Linear Quadratic Regulator (LQR) Controll
   b. Kalman Filters
2. Program a simple OFC model!

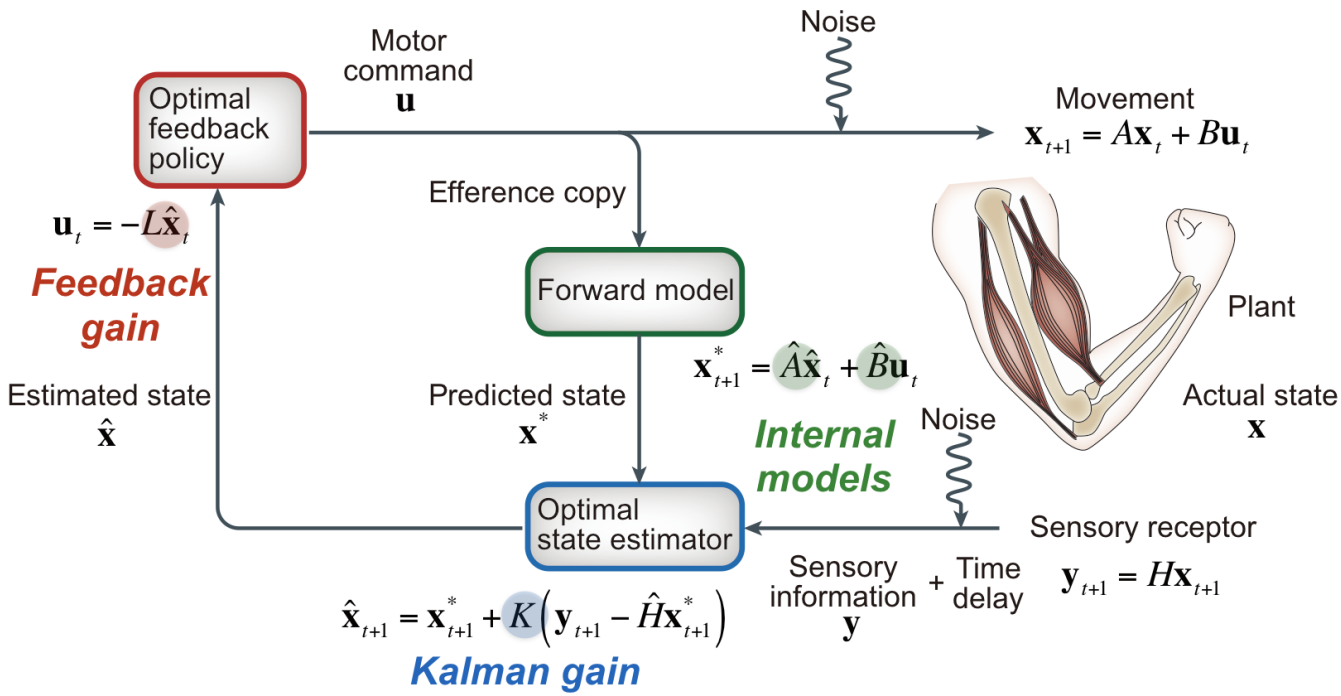
Lecture Objectives

1. General Concept of Linear Quadratic Gaussian Controllers
   a. Linear Quadratic Regulator (LQR) Controll
   b. Kalman Filters
2. Program a simple OFC model!



Lecture Objectives

1. General Concept of Linear Quadratic Gaussian Controllers
   a. Linear Quadratic Regulator (LQR) Controll
   b. Kalman Filters
2. Program a simple OFC model!



What is LQG?

Linear Quadratic Gaussian (LQG) control: Provides a framework to optimally control a dynamical system with input and output noise (i.e., stochastic, optimal control).

1. Linear Quadratic Regulator (LQR)
2. Kalman Filter

$$x_{k+1} = A_d x_k + B_d u_k + v_k$$
$$y_k = C x_k + w_k$$

x_{k+1} are future states, x_k is the current state, u is the signal input, A represents the dynamics, B describes the relationship between signal input (u) and force generation, C is an observation matrix, v_k is additive state noise, and w_k is additive sensory noise.

LQR

LQR finds the minimal u (e.g., minimal effort) that lets us reach our desired state (e.g., stop on some target).

Let's consider only the dynamics with no noise,

$$x_{k+1} = Ax_k + Bu_k$$

We want to minimize the cost, J, as follows:

$$J = \sum_{k=0}^{N} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

This problem can be solved analytically using Hamilton-Jacobi-Bellman equation or Calculus of Variations (Lagrange multipliers). Check out this resource for the derivation: Click Me: ▸ Link

LQR

The optimal control sequence, u, that minimizes the cost (J) is given by:

$$u_k = -F_k x_k$$

s.t.,

$$F_k = (R_k + B_d^T P_{k+1} B_d)^{-1} (B_d^T P_{k+1} A_d)$$

where, P_k is found by iterating backwards in time using the Riccati equation as follows:

$$P_k = A_d^T P_{k+1} A_d - (A_d^T P_{k+1} B_d) \left(R_k + B_d^T P_{k+1} B_d\right)^{-1} (B_d^T P_{k+1} A_d) + Q_k$$

Initial Condition: $P_{k+1} = Q_N$

Sensorimotor Noise

BUT, WE HAVE NOISE IN OUR SENSORIMOTOR SYSTEM

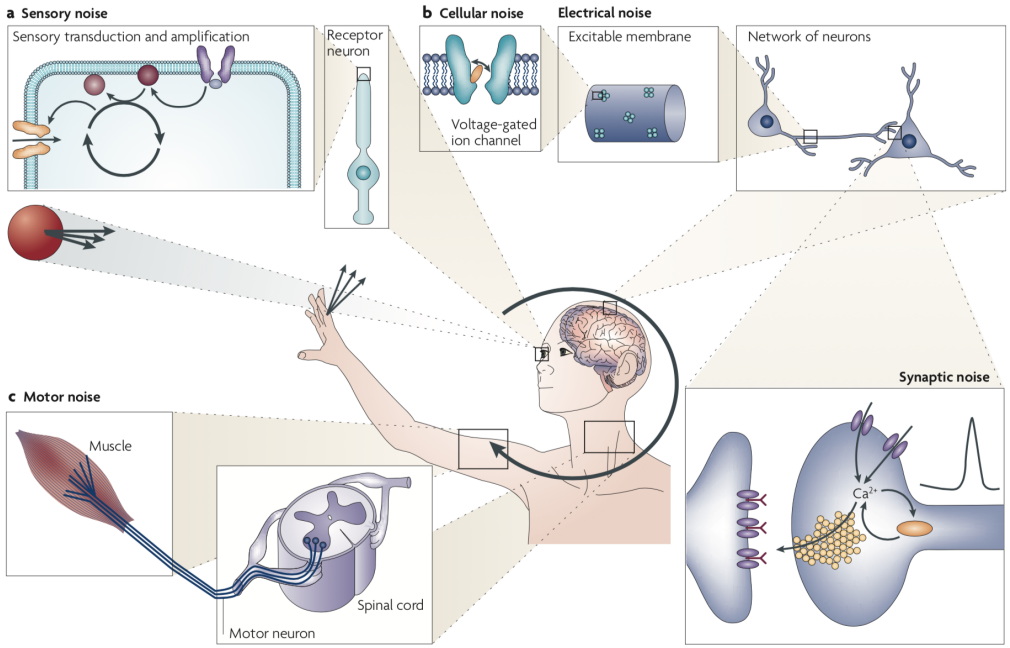
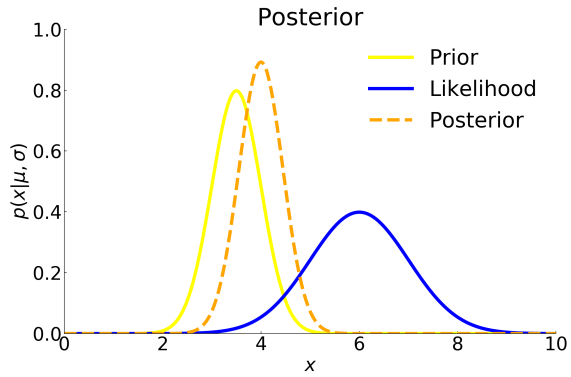


Figure 1 | **Overview of the behavioural loop and the stages at which noise is present in the nervous system.** **a** | Sources of sensory noise include the transduction of signals. This is exemplified here by a photoreceptor and its signal-amplification cascade. **b** | Sources of cellular noise include the ion channels of excitable membranes, synaptic transmission and network interactions (see BOX 2). **c** | Sources of motor noise include motor neurons and muscle. In the behavioural task shown (catching a ball), the nervous system has to act in the presence of noise in sensing, information processing and movement.

HOW DO WE DEAL WITH THIS?

(Faisal et al., 2008)

Kalman Filter — Priors, Likelihood, Posteriors



1. Prior: Our initial guess of where we are going to end up given some control input and known dynamics
2. Likelihood: What we observe from our senses
3. Posterior: Where we most likely are given our sensory observation, known dynamics, and previous control input

Kalman Filter

Predict:

$\hat{x}_{k+1}^{prior} = A_d \hat{x}_k^{post} + B_d u_k$ Predicted (a priori) state estimation

$P_{k+1}^{prior} = A_d P_k^{post} A_d^T + V_k$ Predicted (a priori) estimate of covariance

Update:

$\tilde{y}_{k+1} = y_{k+1} - C\hat{x}_{k+1}^{prior}$ State Innovation (pre-fit residual)

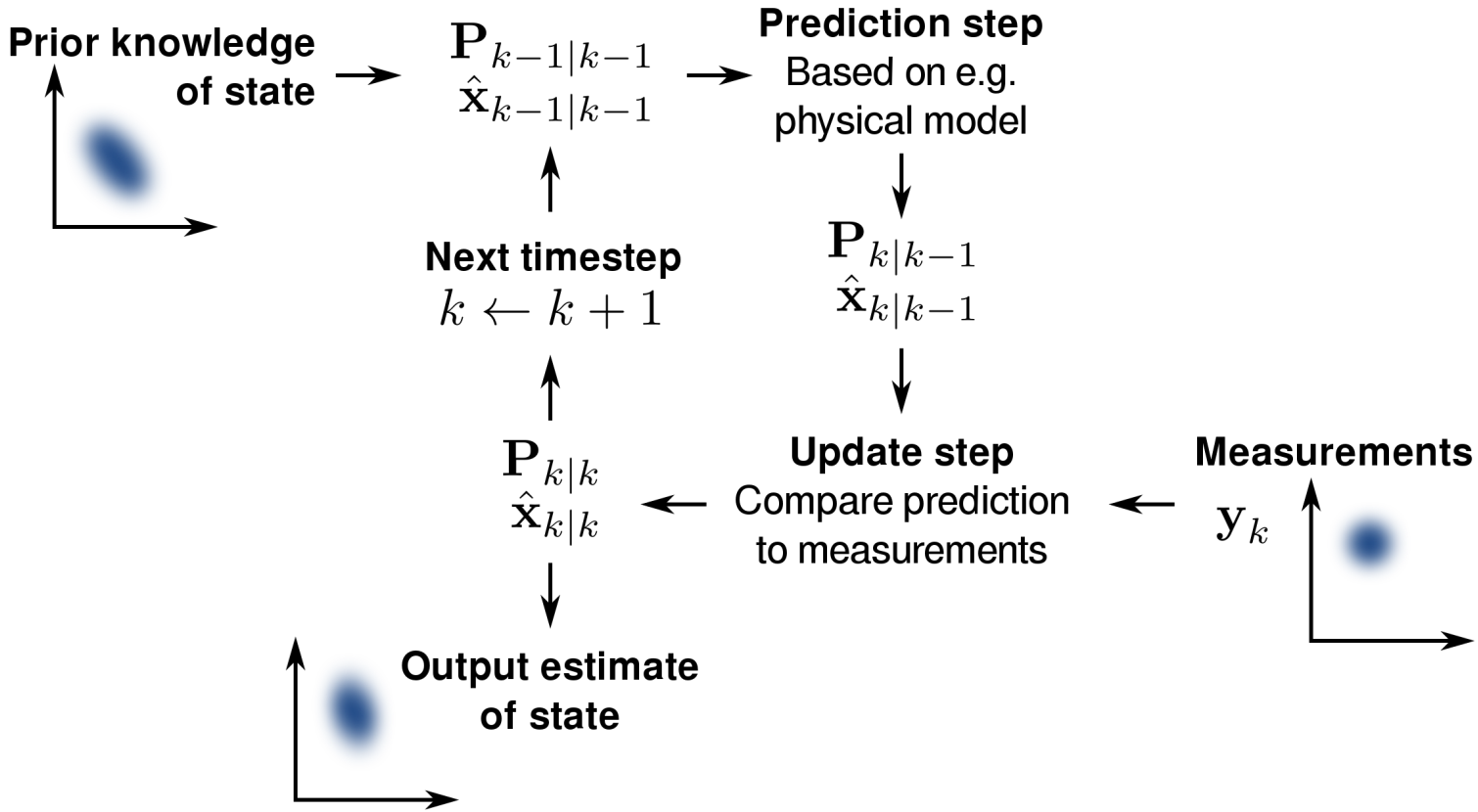
$S_{k+1} = CP_{k+1}^{prior} C^T + W_{k+1}$ Covariance Innovation (pre-fit residual)

$K_{k+1} = P_{k+1}^{prior} C^T S_{k+1}^{-1}$ Optimal Kalman Gain

$\hat{x}_{k+1}^{post} = \hat{x}_{k+1}^{prior} + K_{k+1}\tilde{y}_{k+1}$ Updated (a posteriori) state estimation

$P_{k+1}^{post} = (I - K_{k+1}C) P_{k+1}^{prior}$ Updated (a posteriori) covariance estimation

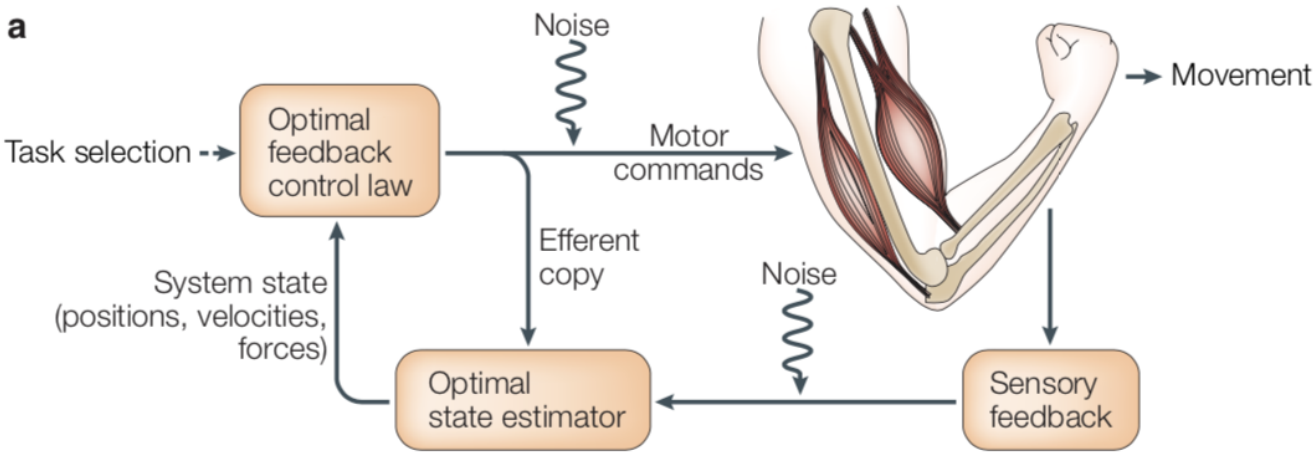
Kalman Filter — State and Covariance



LQG — Combining LQR and Kalman Filters

Linear Quadratic Gaussian (LQG) Control combines the idea of:

1. Optimal Feedback Control
2. State Estimation



LQG — Combining LQR and Kalman Filters

$$u_k = -F_k \hat{x}_k^{post}$$

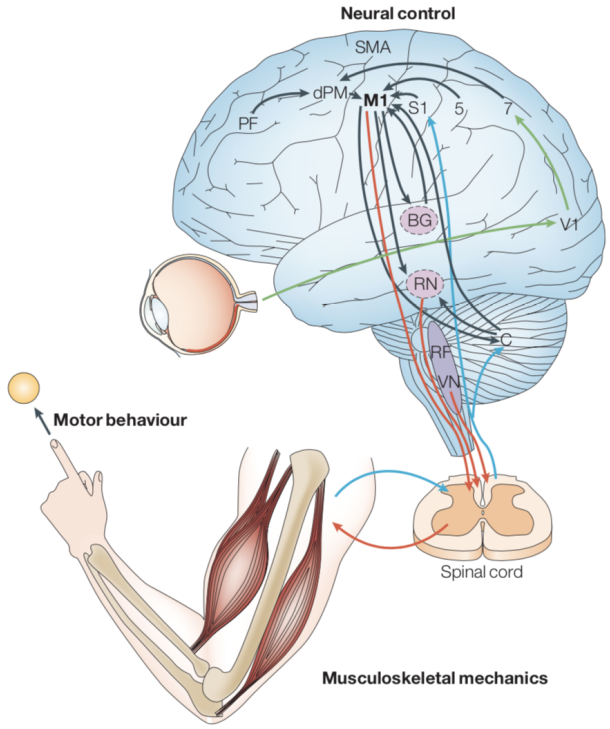
$$x_{k+1} = A_d x_k + B_d u_k + v_k$$

$$y_{k+1} = C x_{k+1} + w_{k+1}$$

$$\hat{x}_{k+1}^{prior} = A_d \hat{x}_k^{post} + B_d u_k$$

$$\tilde{y}_{k+1} = y_{k+1} - C \hat{x}_{k+1}^{prior}$$

$$\hat{x}_{k+1}^{post} = \hat{x}_{k+1}^{prior} + K_{k+1} \tilde{y}_{k+1}$$

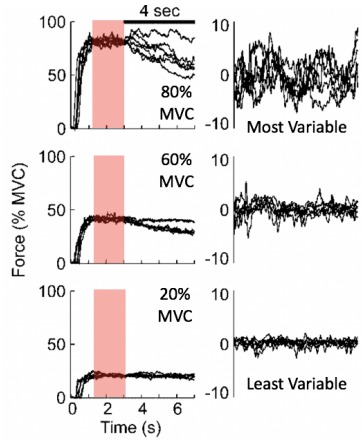


Other Considerations

1. Time Delays
2. Signal Dependent Noise
3. Non-Gaussian Noise
4. Nonlinear Dynamics
5. Constraints
6. Knowledge of Dynamics
7. Adaptation
8. Error-Based vs. Reinforcement-Based Learning

Time Delays — Matrix Augmentation

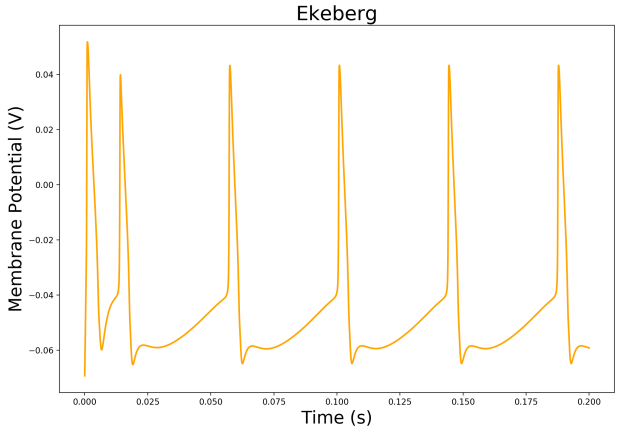
Signal Dependent Noise

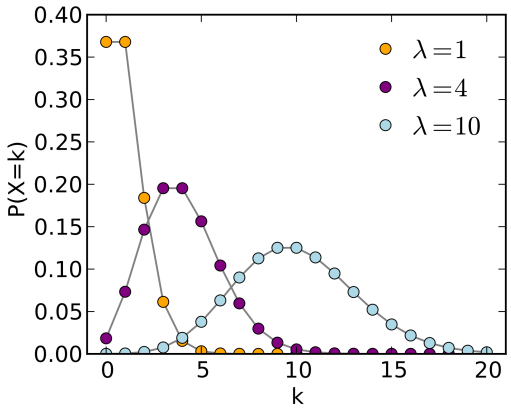


1. Signal Dependent (Multiplicative) Noise
2. Multiplied by control input, u
3. Solve LQR and Kalman simultaneously

   . put in same loop until some tolerance level is met
4. Todorov, E. (2005). Neural computation, 17(5), 1084-1108.

Non-Gaussian Noise

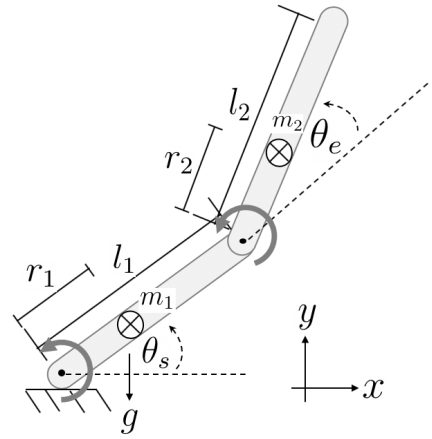


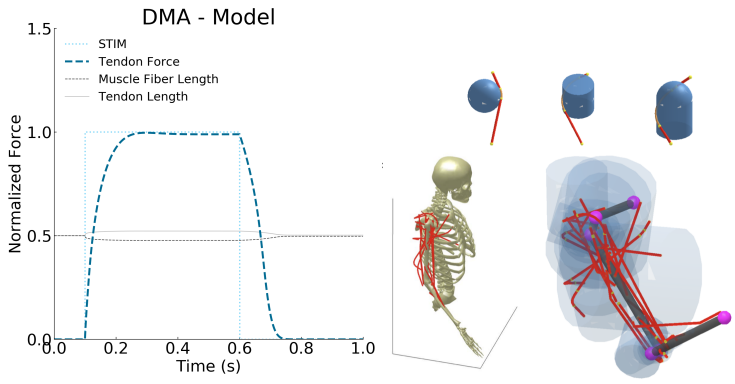


Bounded, Poisson distributed neuron firings

1. Extended Kalman filter
2. Unscented Kalman filter
3. Particle Filters

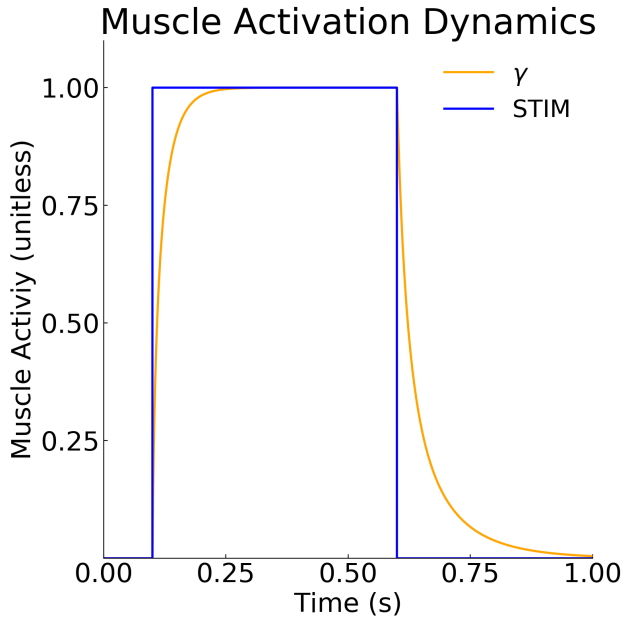
Nonlinear Dynamics

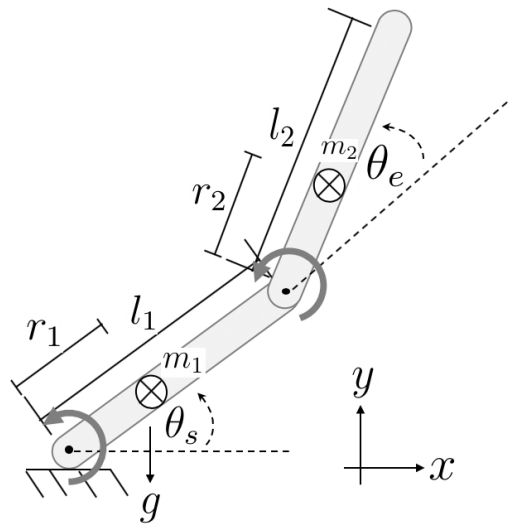




Model Predictive Control

Constraints

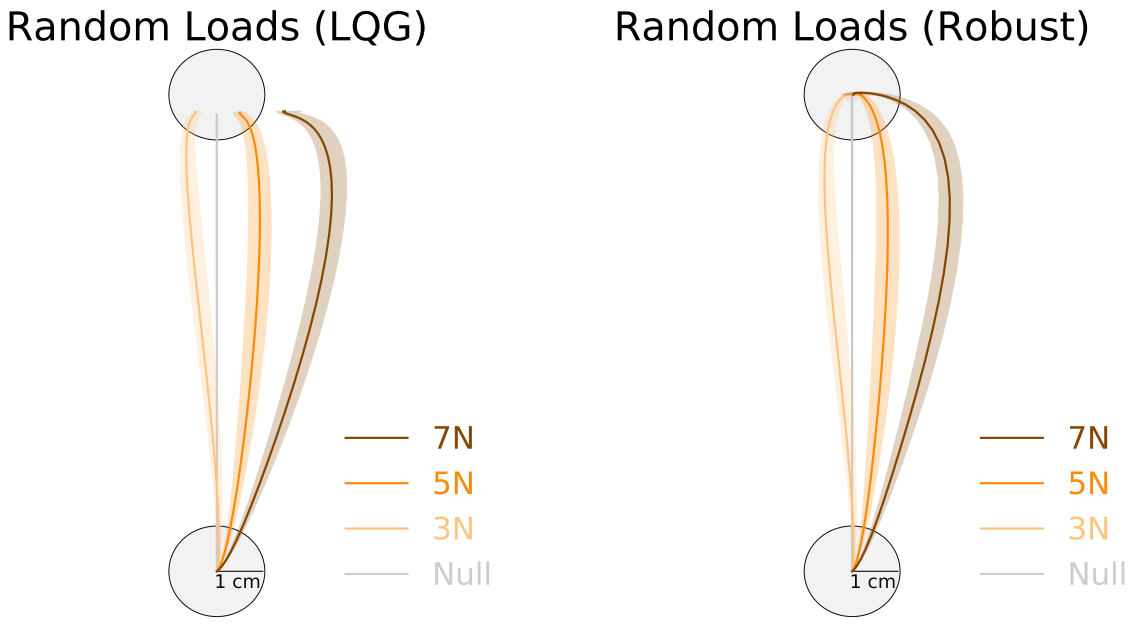




Model Predictive Control

- muscle activation dynamics (0-1)
- joint range of motion

Knowledge of Dynamics



H_∞ (Robust) Control

. Internal models

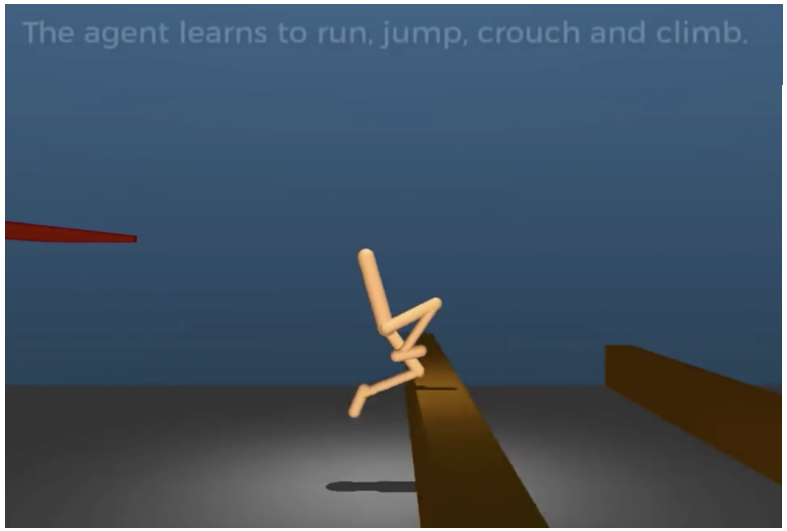
. Imperfect knowledge

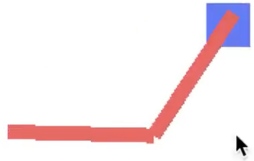
Adaptation

Adaptive Control

- Need to update control policy
  - e.g., plane losing fuel
- We are not born optimal
- Multiple time-scales

Error verses Reinforcement

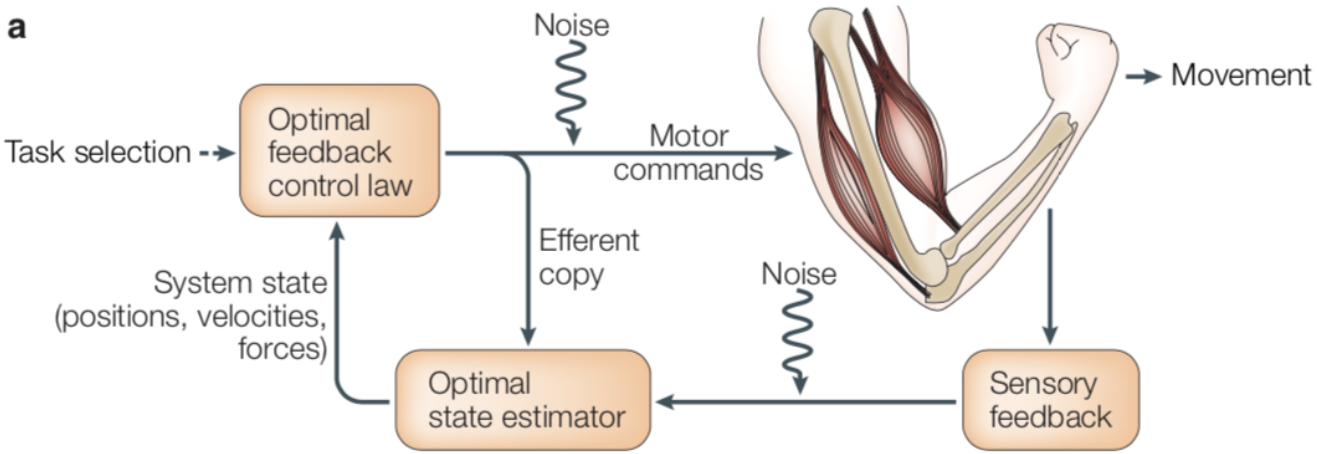




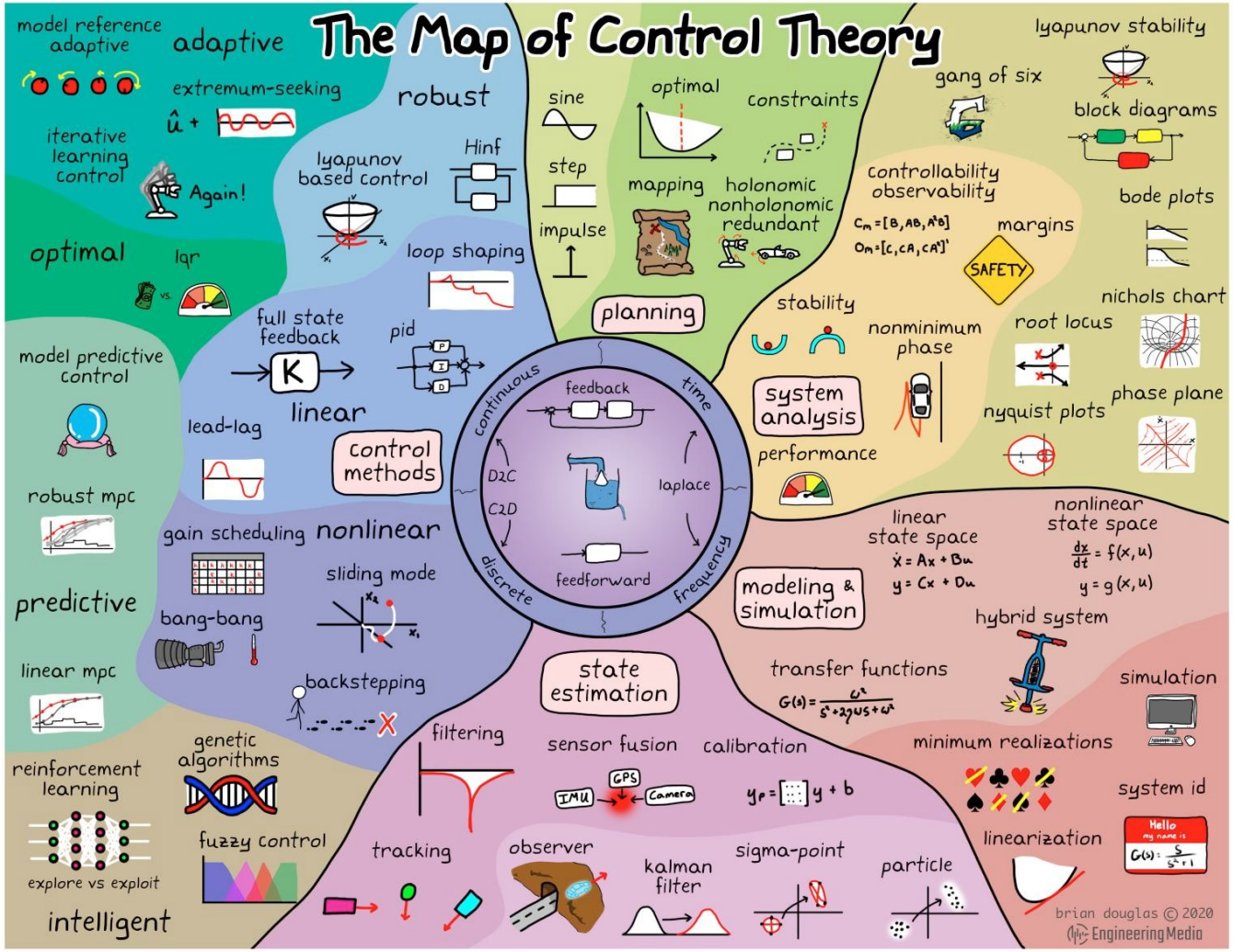
1. Error vectors vs. reward signals
2. Neural Basis
   . cerebellum & basal ganglia

Take Homes

1. OFC (LQG) Captures MANY classic features of human behaviour
2. Human Control Involves
   a. (Near) Optimal Control
   b. State Estimation
3. Many Factors to Consider



Control Theory



QUESTIONS???

Next Class

Cost Functions (Objectives) of the Nervous system

1. What do we optimize?
2. Role of Evolution?
3. A Common Currency?

Assignment

see handout

tips and tricks, below (we'll go over these together)

General Form

$$\dot{x}(t) = Ax(t) + Bu(t) + v(t)$$

$$y(t) = Cx(t) + w(t)$$

$\dot{x}(t)$ are future states, $x(t)$ is the current state, $u(t)$ is the signal input, A represents the dynamics, B describes the relationship between signal input and force generation, C is an observation matrix, $v(t)$ is additive state noise, and $w(t)$ is additive sensory noise.

We want to (a) define the equations of motion and then (b) transform the continuous system of equations above into a discrete system of equations.

Step 1 — Define the Equations of Motion

Here we will consider a point-mass that is constrained to move along a line (x-axis). The point-mass has a weight (m), and is linked to a viscous damper (b) and spring (k). Show that

$$m\ddot{x} = -b\dot{x} - kx + F$$

s.t.,

$$u = F$$

can be expressed in matrix form as ($\dot{x} = Ax + Bu$)

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \begin{bmatrix} u \end{bmatrix}$$

Step 2 — Convert to a Discrete System

Convert the continuous system, $\dot{x}(t) = Ax(t) + Bu(t)$, to a discrete system, $x_{k+1} = A_d x_k + B_d u_k$. This can be done by:

$$x_{k+1} \approx (I + A \cdot h)x_k + (B \cdot h)u_k$$

$$x_{k+1} \approx A_d x_k + B_d u_k$$

where, h is the time step (0.01) and I is an identity matrix:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Note: Here we are simply performing Euler integration by redefining the A matrix, whereas B is scaled by the time-step

Step 2 Cont'd

For example, with m (4.0), b (1.0), k (0.25), h(0.01):

$$A_d = \begin{bmatrix} 1 & 0.01 \\ -.000625 & 0.9975 \end{bmatrix}$$

and

$$B_d = \begin{bmatrix} 0 \\ 0.0025 \end{bmatrix}$$

and

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Now we have our system of equation in the discrete form of

$x_{k+1} = A_d x_k + B_d u_k + v_k$

$y_k = C x_k + w_k$

Note: $y_{k+1} = C x_{k+1} + w_{k+1}$

Step 3 — Define the Cost Function

To numerically optimize the cost,

$$J = \sum_{k=0}^{N} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right)$$

we have to define R_k and Q_k for all time points $(0, 1, ..., N)$

$$R_k = \begin{bmatrix} 0.0000001 \end{bmatrix}$$

and

$$Q_k = \begin{cases} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, & \text{if } k \neq N \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \text{if } k = N \end{cases}$$

Step 4 — Find the Optimal Feedback Gains

Use a 0.01 step size (h) from 0.0 to 0.5s (51 time steps). Perform the following equations to numerically solve for J:

Initial Condition: $P_{k+1} = Q_{N-1}$

$$F_k = (R_k + B_d^T P_{k+1} B_d)^{-1} (B_d^T P_{k+1} A_d)$$

$$P_k = A_d^T P_{k+1} A_d - (A_d^T P_{k+1} B_d) \left(R_k + B_d^T P_{k+1} B_d\right)^{-1} (B_d^T P_{k+1} A_d) + Q_k$$

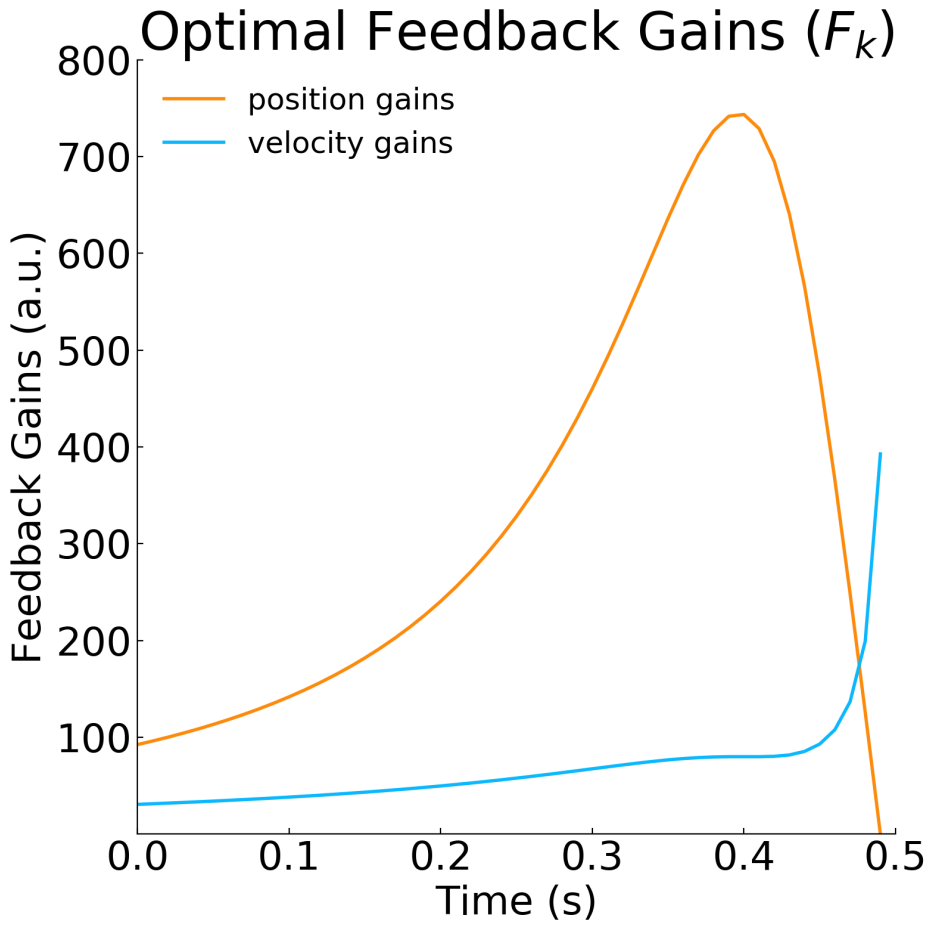
Remember, P_k is found by iterating BACKWORDS in time!

Note: F_k (and U_k) both have length $N = 50$ and we are solving for $k = N-1, N-2, ..., 0$. Specifically, we are solving for $k = 49, 48, ..., 0$

The optimal control policy, which we will use further below, is defined as

$$u_k = -F_k x_k$$

Step 5 — Plot the Feedback Gains



Step 6 — Run LQR Controller

1.) Run the following dynamics **without** noise

Initial conditions: $x_0 = \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix}$

$$u_k = -F_k x_k$$

$$x_{k+1} = A_d x_k + B_d u_k$$

2.) Run the same algorithm but now **with** noise

Initial conditions: $x_0 = \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix}$; $v_k = \mathcal{N}\left(\begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}, \begin{bmatrix} 0.005 & 0.0 \\ 0.0 & 0.005 \end{bmatrix} \right)$

Tip: v_k = np.random.normal(0,0.005,(2,1)) — yields a 2x1 noise vector

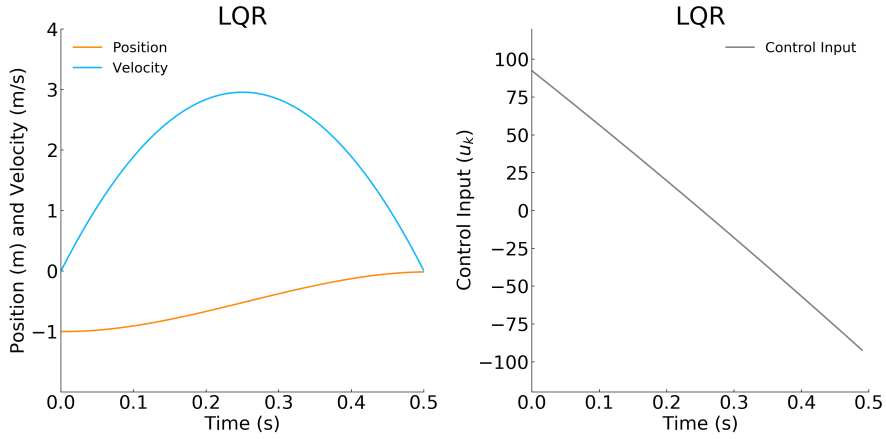
$$u_k = -F_k x_k$$

$$x_{k+1} = A_d x_k + B_d u_k + v_k$$

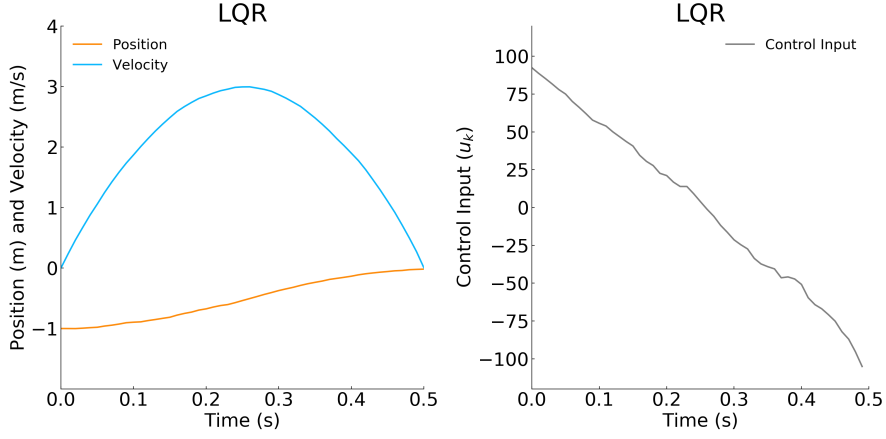
Note: x_k has length $N + 1(51)$ and we are solving for $k = 0, 1, ..., 50$

Step 7 — LQR Plots

1.) Plot the States (x_k) and Control Input (u_k) **without** noise



2.) Plot the States (x_k) and Control Input (u_k) **with** noise



Step 8 — Calculate the Kalman Gain

Covariance matrices for sensory noise (W) and state noise (V):

$$W = \begin{bmatrix} 0.01 & 0.0 \\ 0.0 & 0.01 \end{bmatrix} \quad V = \begin{bmatrix} 0.005 & 0.0 \\ 0.0 & 0.005 \end{bmatrix}$$

Initial Condition: $P_0^{prior} = W$

$$S_k = C P_k^{prior} C^T + W$$

$$K_k = P_k^{prior} C^T S_k^{-1}$$

$$P_k^{post} = (I - K_k C) P_k^{prior}$$

$$P_{k+1}^{prior} = A_d P_k^{post} A_d^T + V$$

K_k has length $N + 1(51)$ and we are solving for $k = 0, 1, ..., 50$

Note: these equations are the same as defined on the earlier slide. We have just changed the notation from (k+1) to (k) which allows for a seamless integration of the LQR and Kalman filter

Step 10 — LQG

Finally, we combine LQR with state estimation (Kalman Filter).

Initial Conditions:

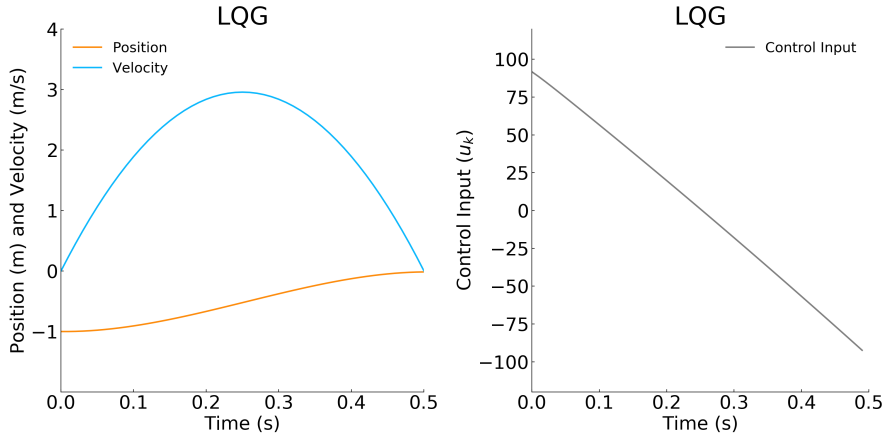
$$x_0 = \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix} \; ; \; w_k = \mathcal{N}\left(\begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}, \begin{bmatrix} 0.01 & 0.0 \\ 0.0 & 0.01 \end{bmatrix} \right) \; ;$$

$$y_0 = x_0 + w_0 \; ; \; \hat{x}_0^{post} = y_0$$

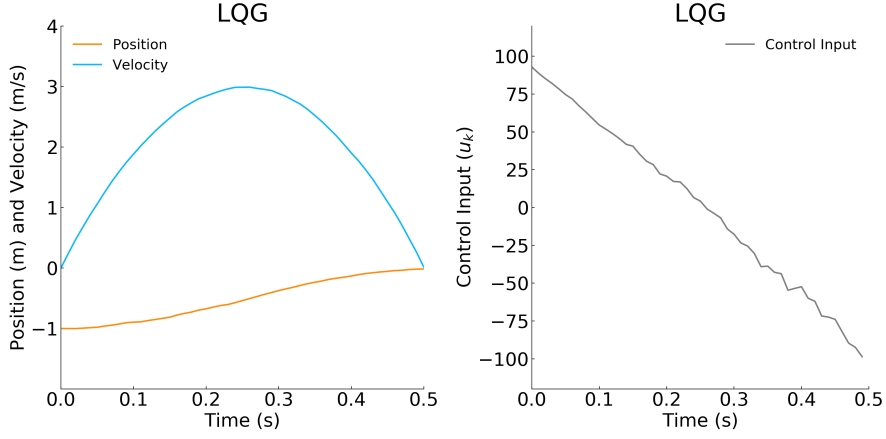
$$\begin{aligned} u_k &= -F_k \hat{x}_k^{post} \\ x_{k+1} &= A_d x_k + B_d u_k + v_k \\ y_{k+1} &= C x_{k+1} + w_{k+1} \\ \hat{x}_{k+1}^{prior} &= A_d \hat{x}_k^{post} + B_d u_k \\ \tilde{y}_{k+1} &= y_{k+1} - C \hat{x}_{k+1}^{prior} \\ \hat{x}_{k+1}^{post} &= \hat{x}_{k+1}^{prior} + K_{k+1} \tilde{y}_{k+1} \end{aligned}$$

Step 11 — LQG Plots

1.) Plot the States (x_k) and Control Input (u_k) **without** noise



2.) Plot the States (x_k) and Control Input (u_k) **with** noise



THAT'S ALL!

Acknowledgements

Phillip Sabes

Fred Crevecoeur

Gunnar Blohm

Stephen Scott