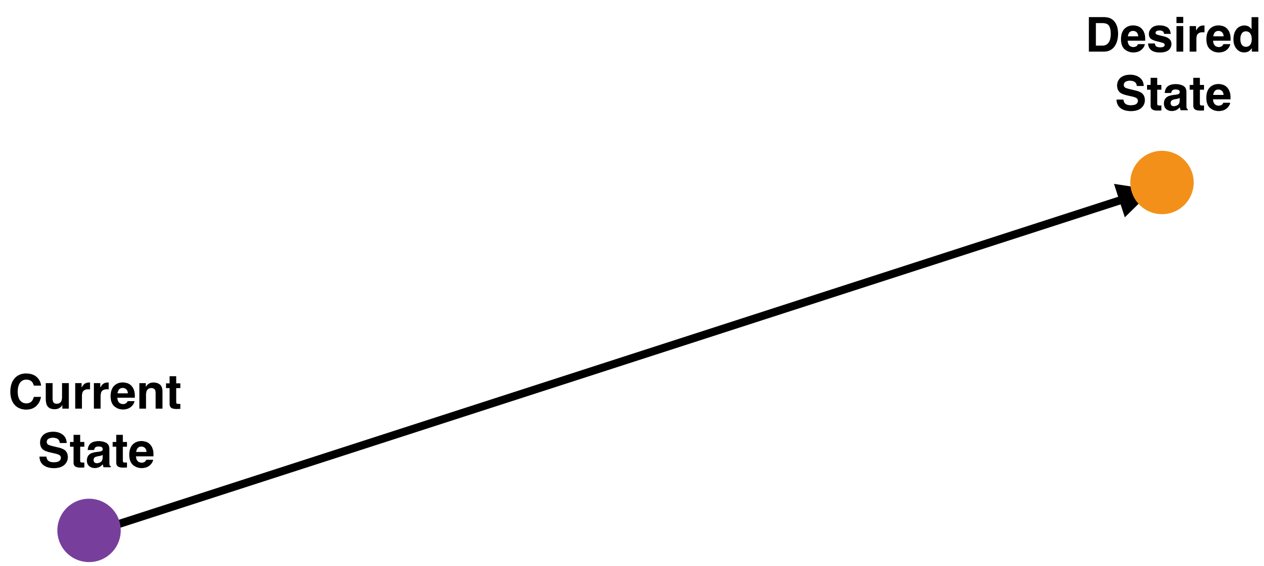
Robot Autonomy

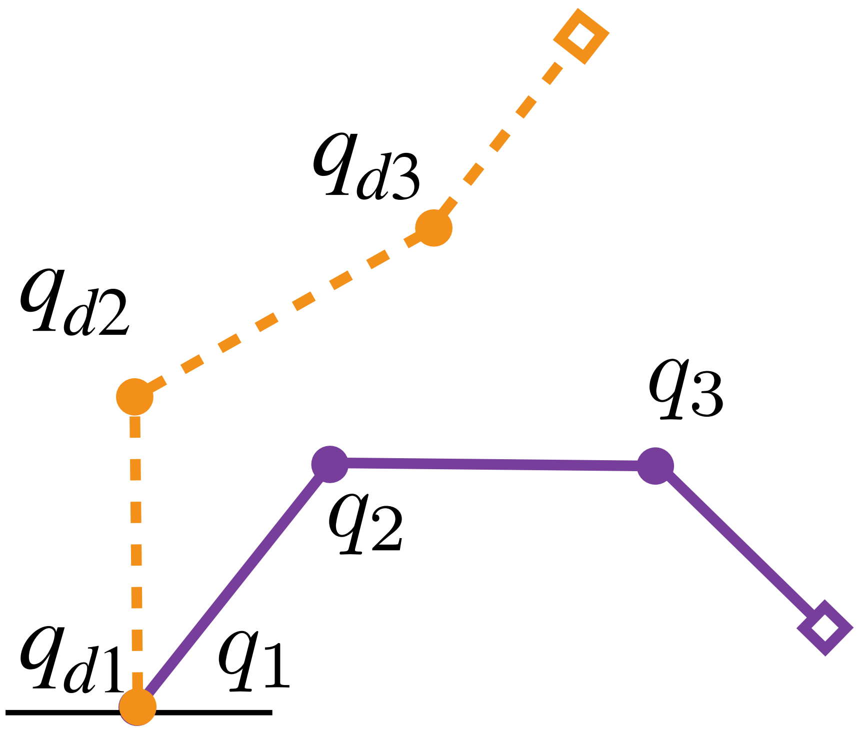
Lecture 2: Control

Oliver Kroemer

Motivation

- Assume robot is given a desired trajectory or pose



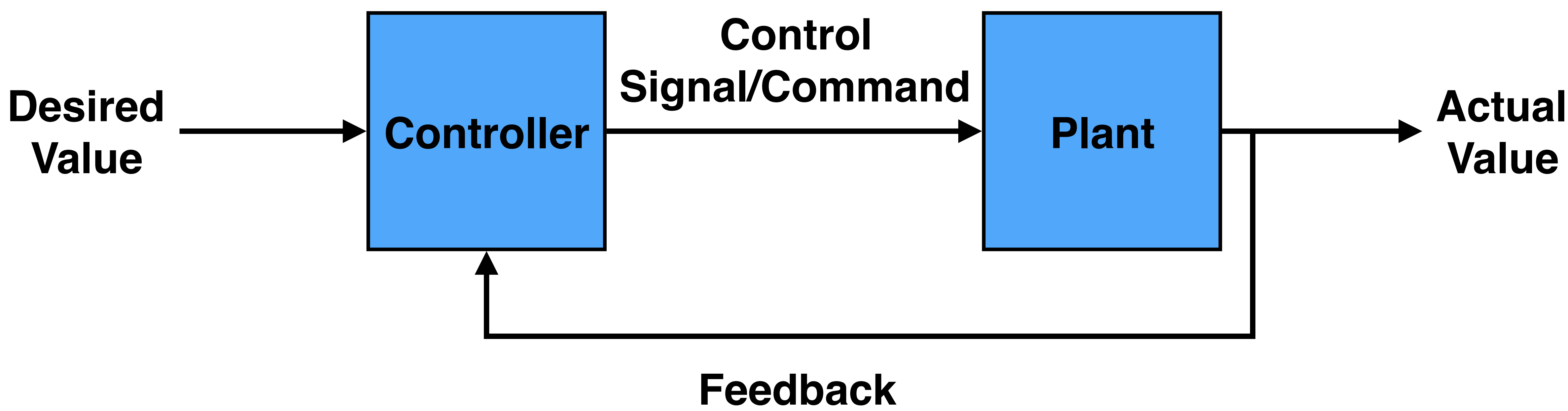


- Need to get robot to move to pose or follow trajectory
  - Robust to perturbations during execution
  - Control interaction forces when in contact

Feedback Basics

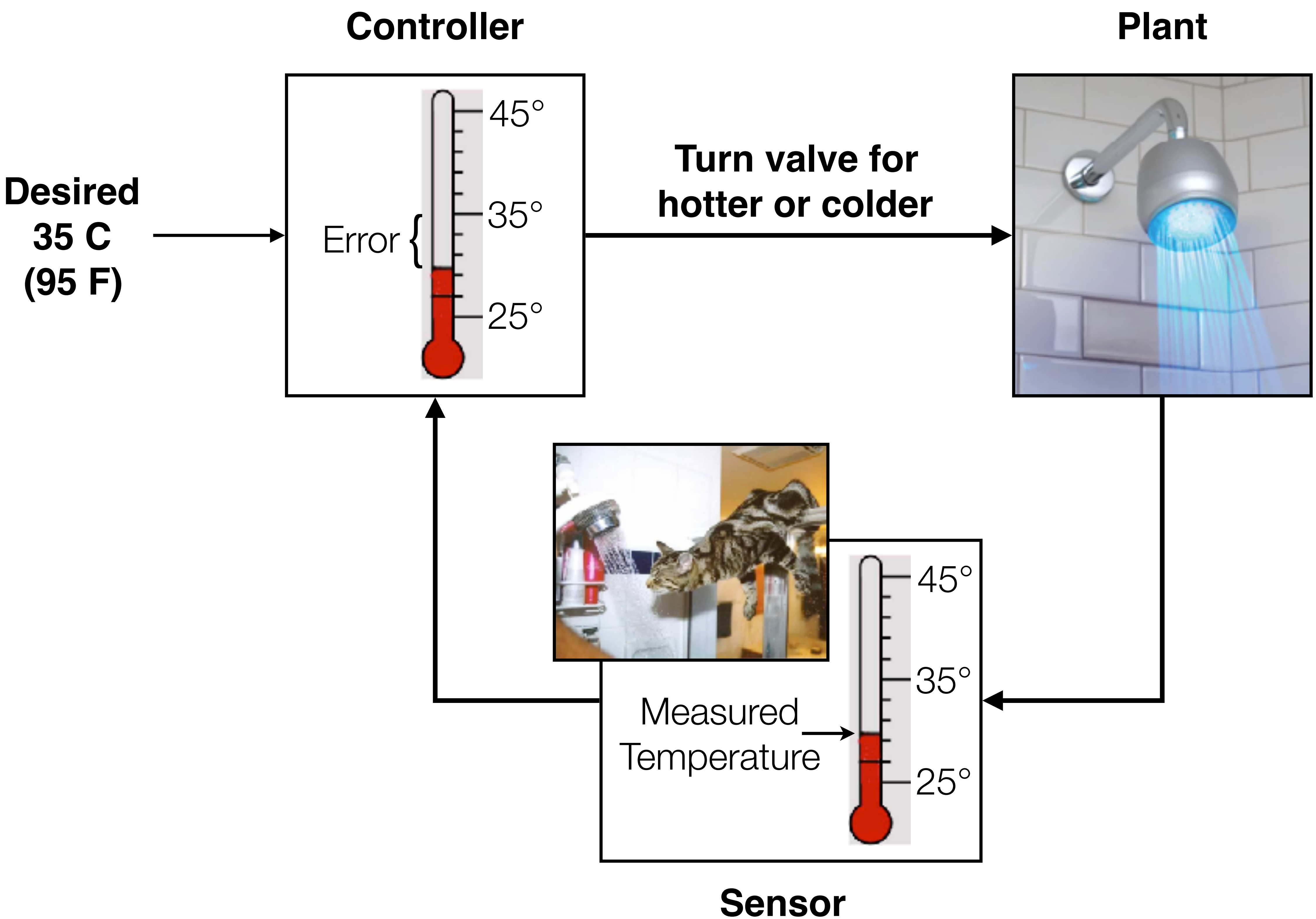
Feedback Controller

- Need a controller to control the robot's movements
  - adapt control signal based on error between desired and actual

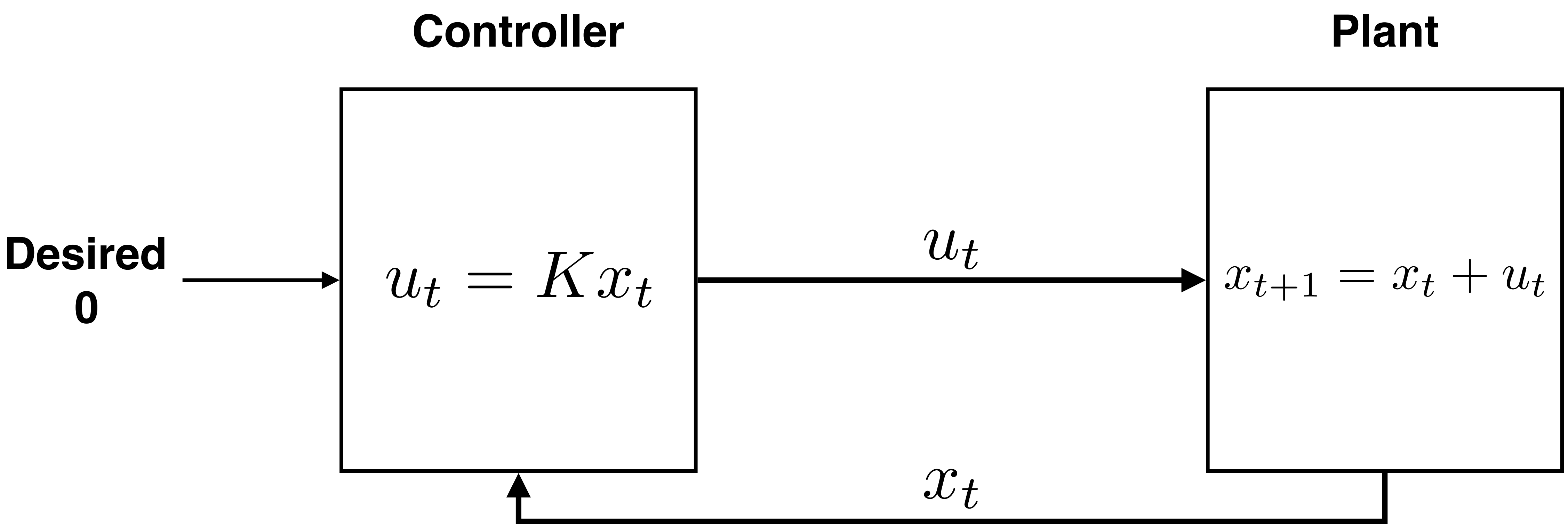


  - Feedback allows robot to compensate for errors/perturbations

Shower Example (Celsius)



Shower Example



$$x_{t+1} = Ax_t + Bu_t$$

Linear System

- Linear system with linear control

$$x_{t+1} = Ax_t + Bu_t \qquad\qquad u_t = Kx_t$$

  - Assume desired state is 0 without loss of generality

$$x_{t+1} = Ax_t + B(Kx_t)$$

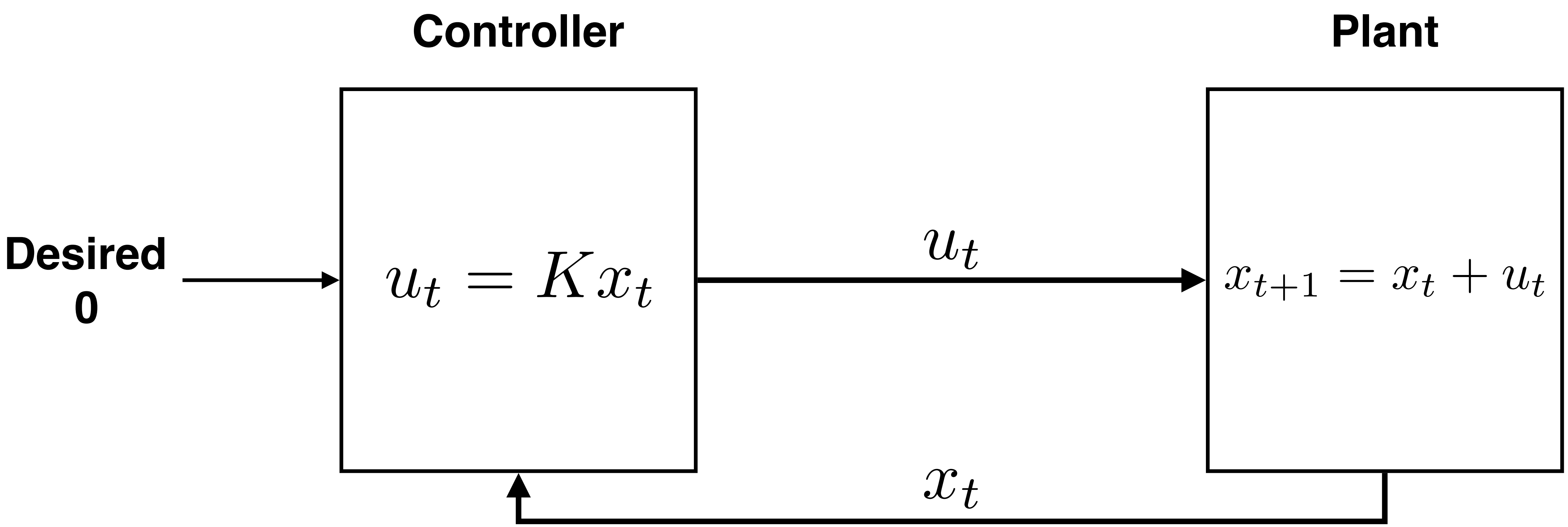
$$x_{t+1} = (A + BK)x_t$$

$$x_{t+n} = (A + BK)^n x_t$$

- We want state to tend to zero as n tends to infinity
- Absolute eigenvalues need to be less than one

$$|\mathrm{eig}(A + BK)| < 1$$

Simple Example



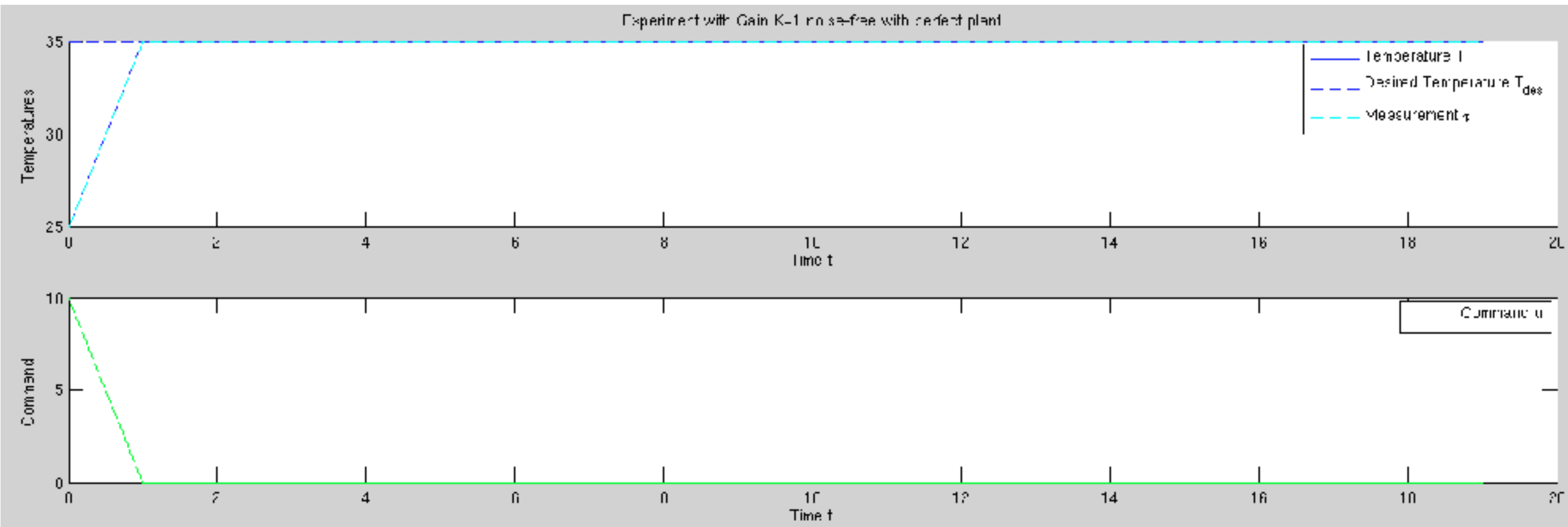
$$x_{t+1} = Ax_t + Bu_t$$

$$A = 1, B = 1, K = -1 \rightarrow A + BK = 0$$

Simple Example

$$A = 1, B = 1, K = -1 \rightarrow A + BK = 0$$

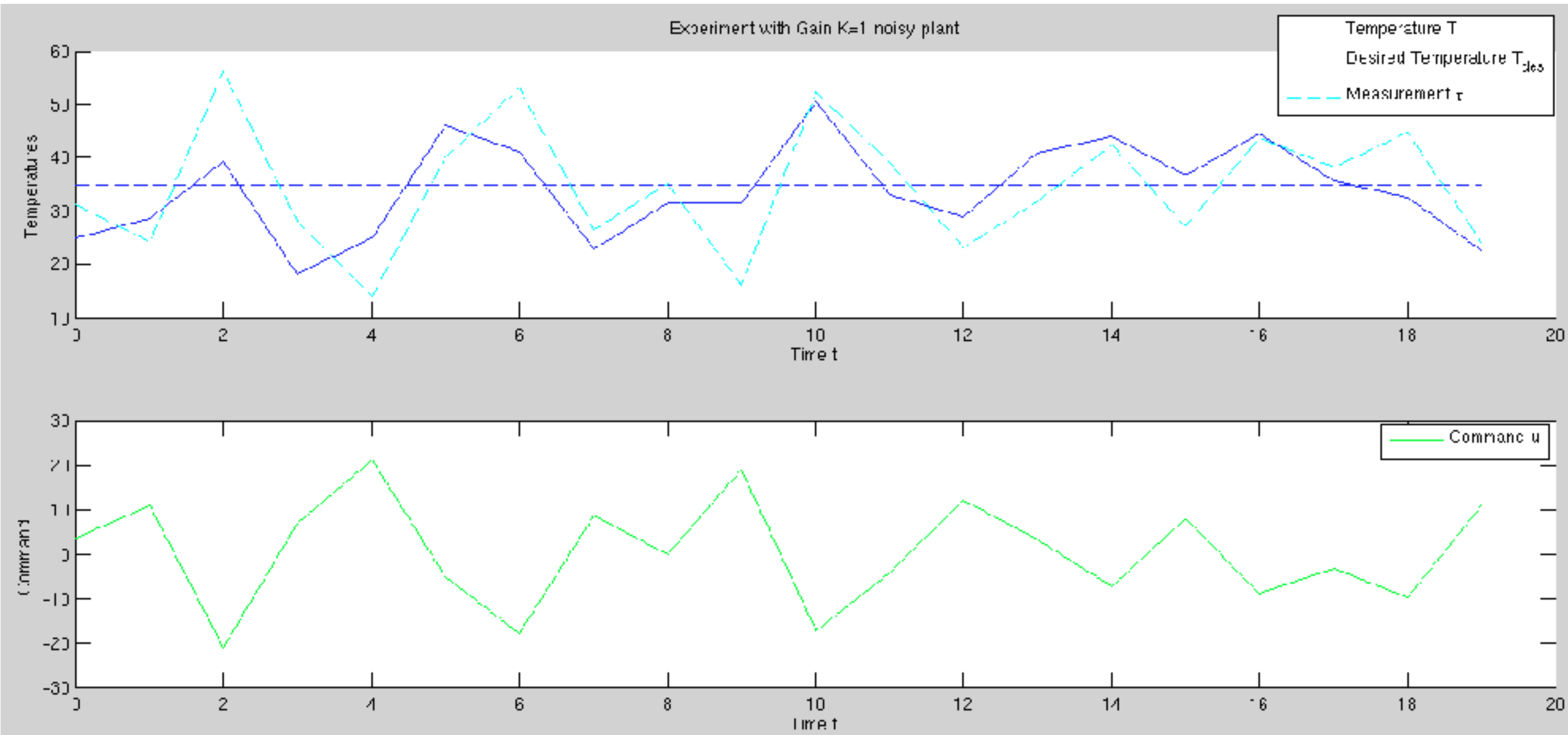
- Ideal plant and no noise:



Simple Example

$$A = 1, B = 1, K = -1 \rightarrow A + BK = 0$$

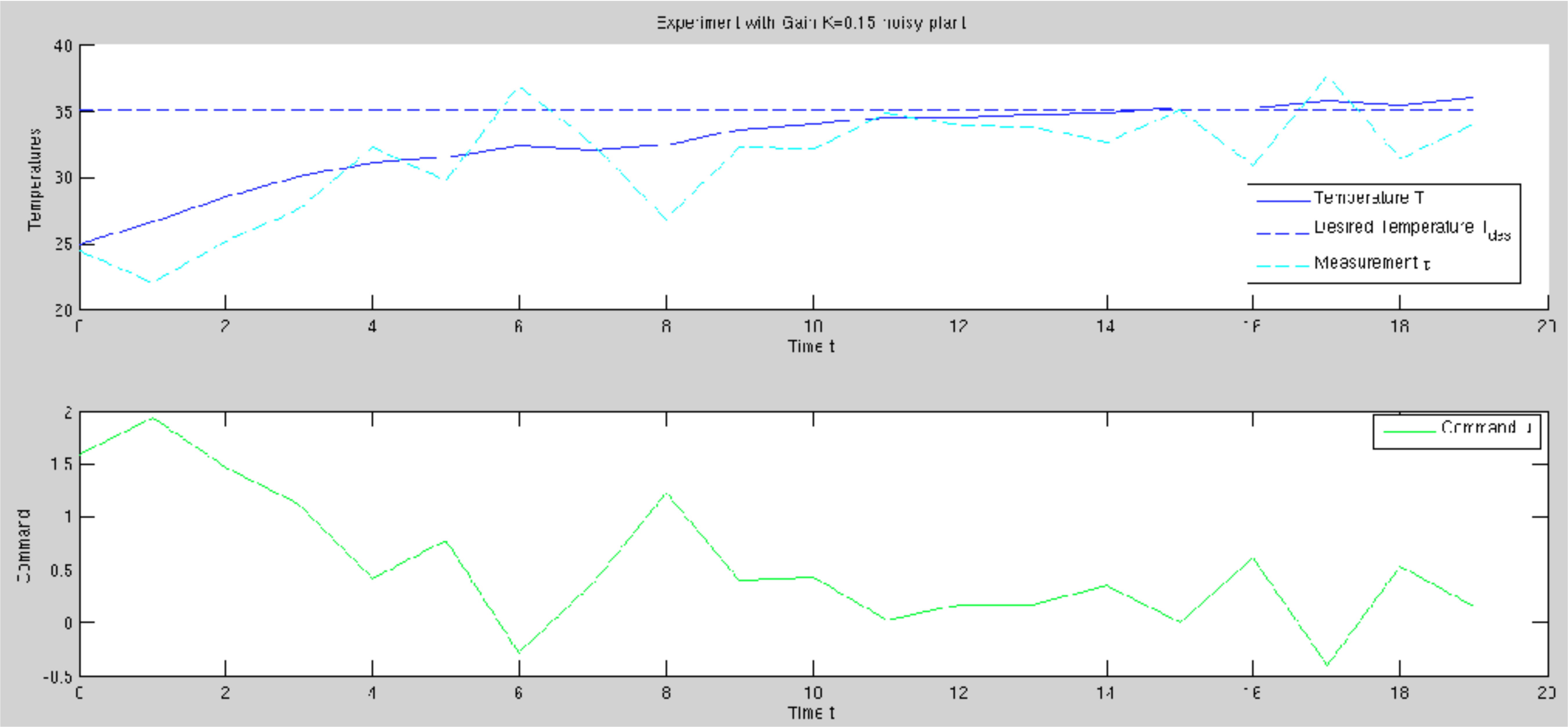
- With measurement noise:



Simple Example

$$A = 1, B = 1, K = -0.15 \rightarrow A + BK = 0.85$$

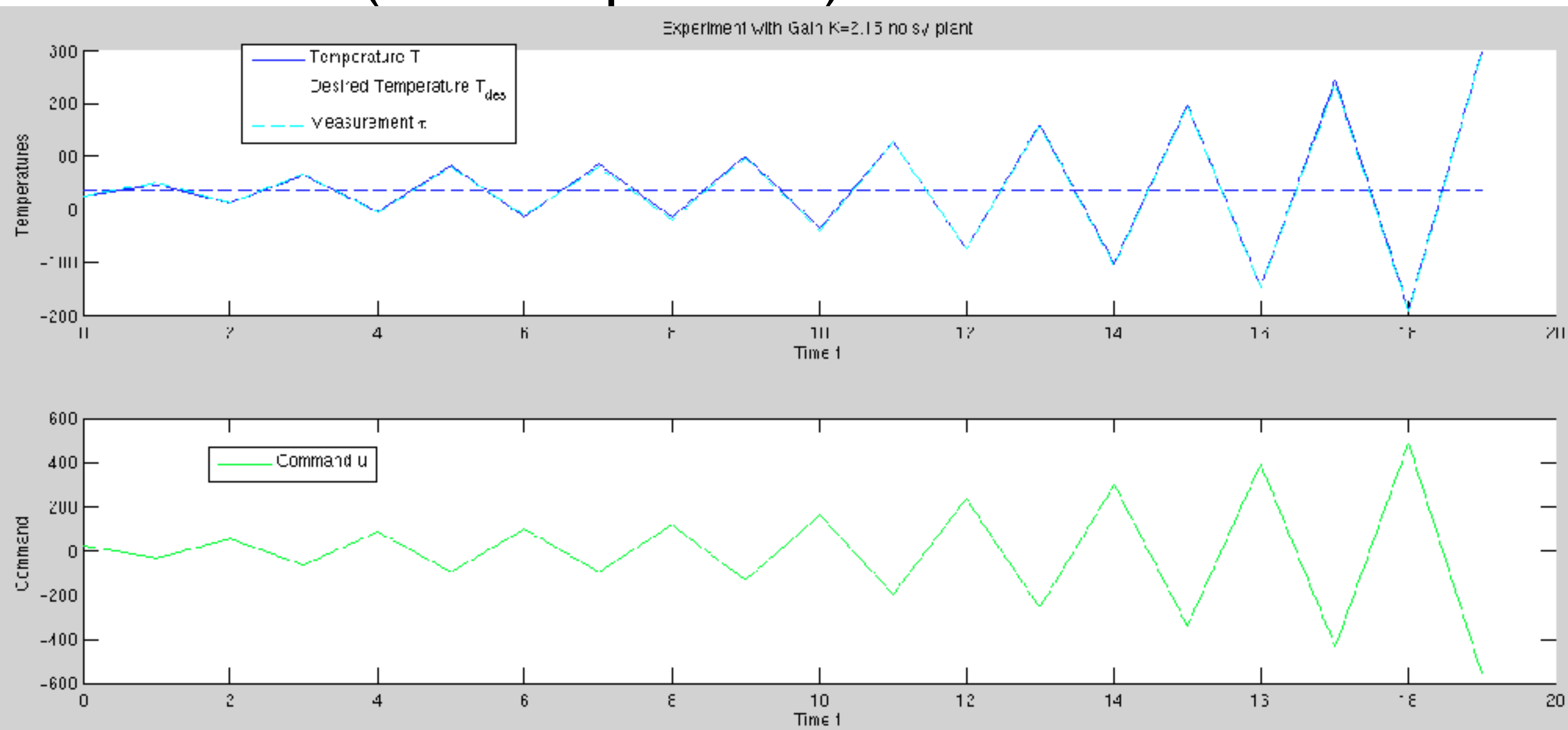
- With measurement noise:



Simple Example

$$A = 1, B = 1, K = -2.15 \rightarrow A + BK = -1.15$$

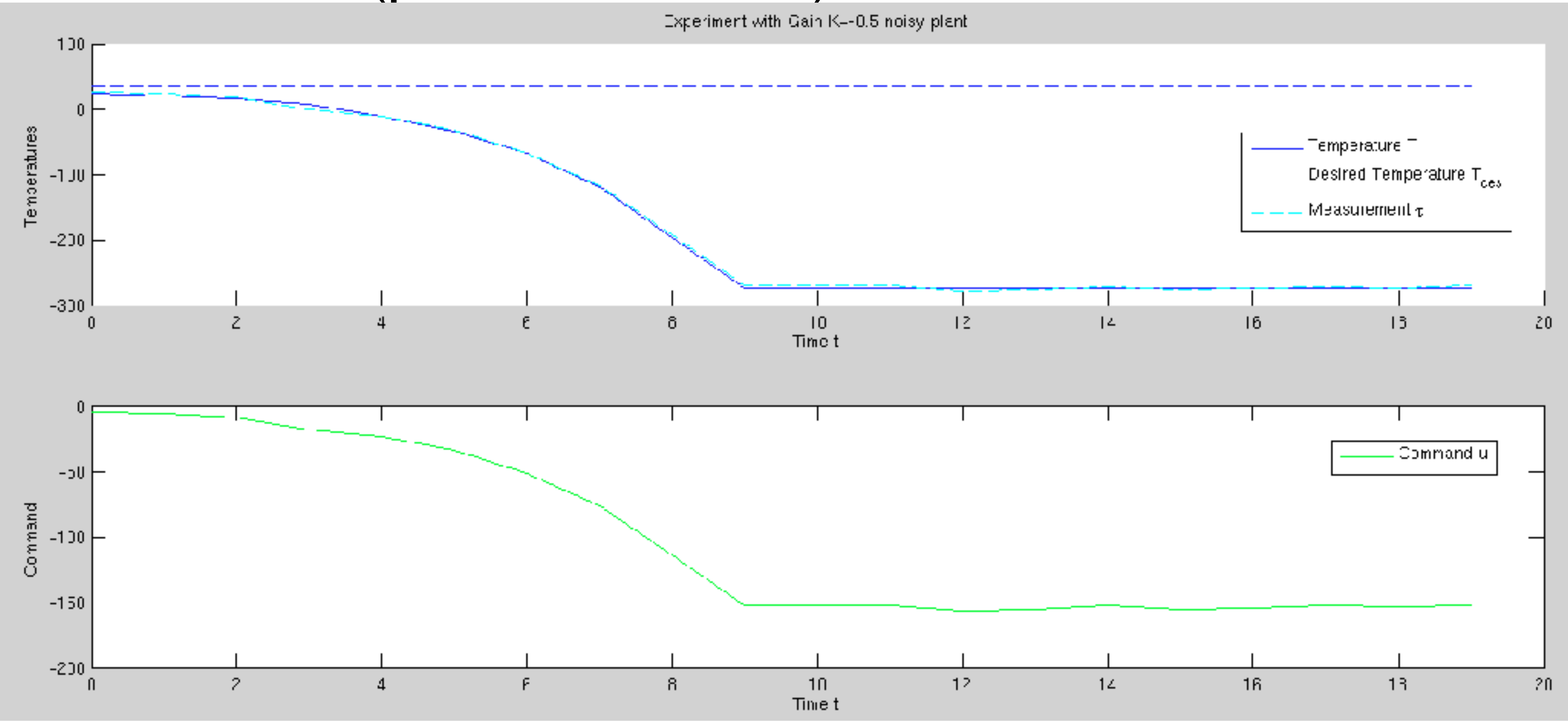
- Unstable (overcompensates)



Simple Example

$$A = 1, B = 1, K = 0.5 \rightarrow A + BK = 1.5$$

- Unstable (positive feedback)



Continuous Time Systems

- Continuous time linear systems:

$$\dot{x} = Ax + Bu = Ax + BKx$$

$$\dot{x} = (A + BK)x$$

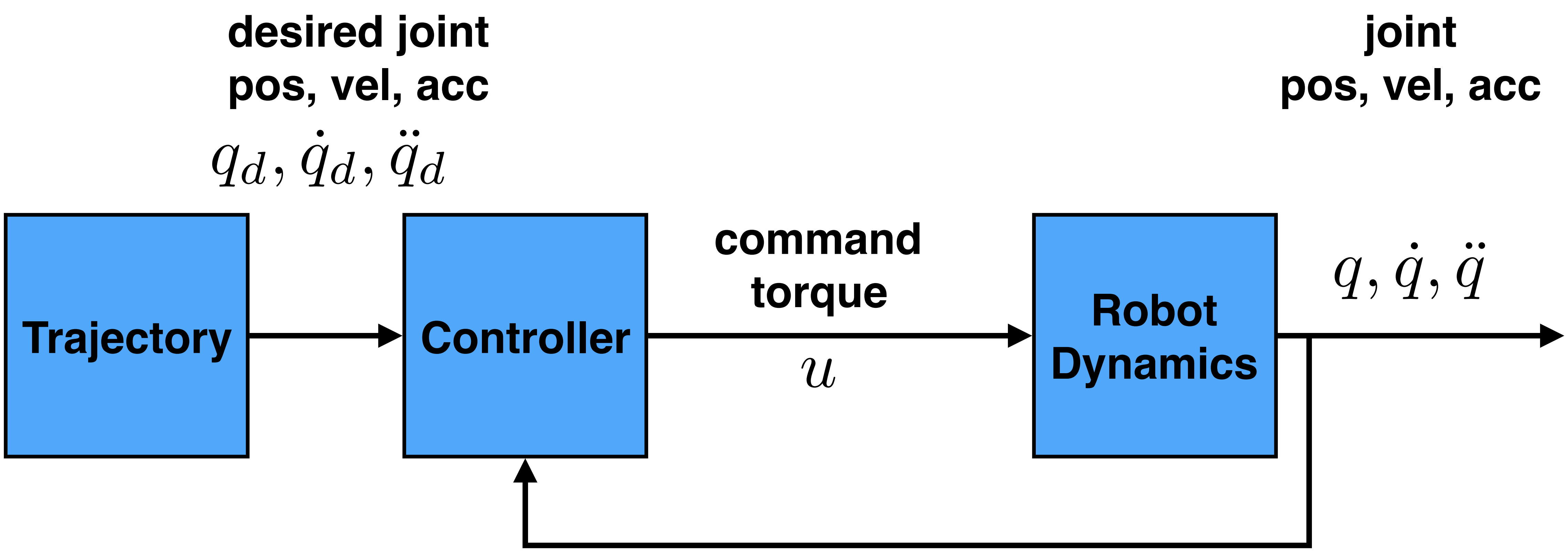
$$x(t) = \exp^{(A+BK)t} x(0)$$

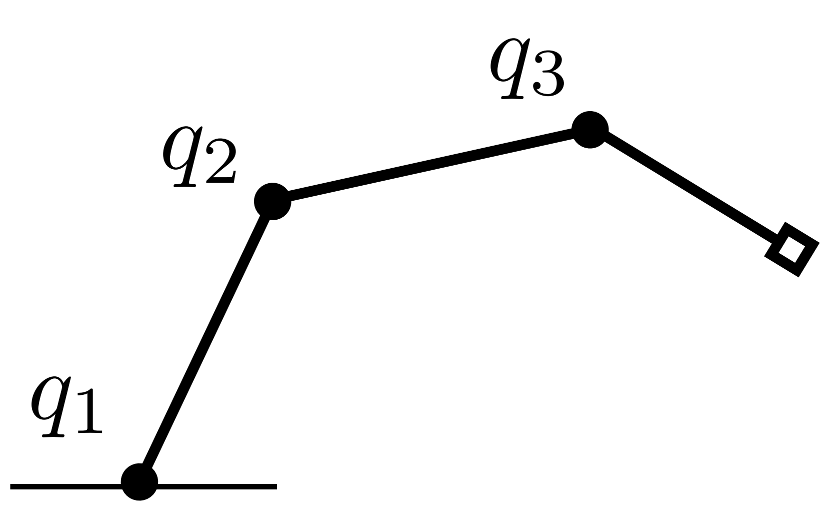
- We want state to tend to zero as n tends to infinity
- Real part of eigenvalues need to be negative

$$\text{Real}(\text{eig}(A + BK)) < 0$$

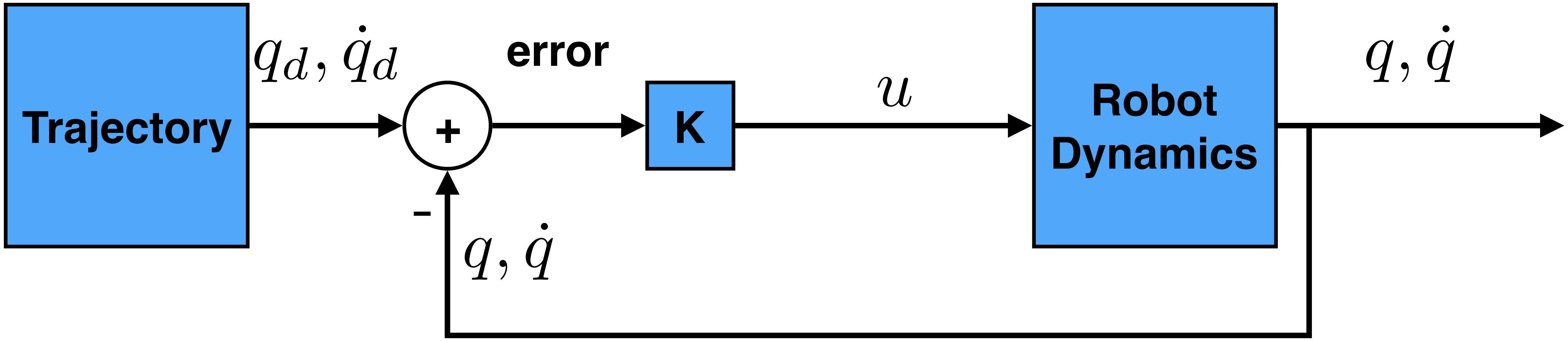
PID Control

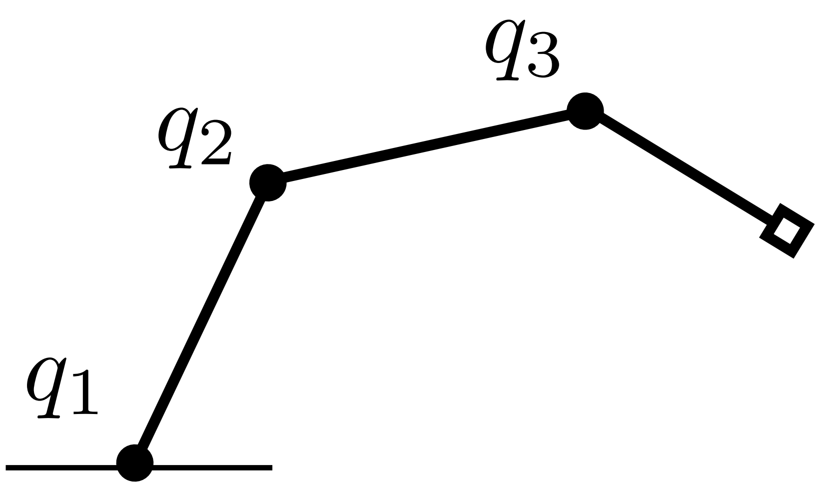
Linear Feedback Control in Robotics





Linear Feedback Control in Robotics





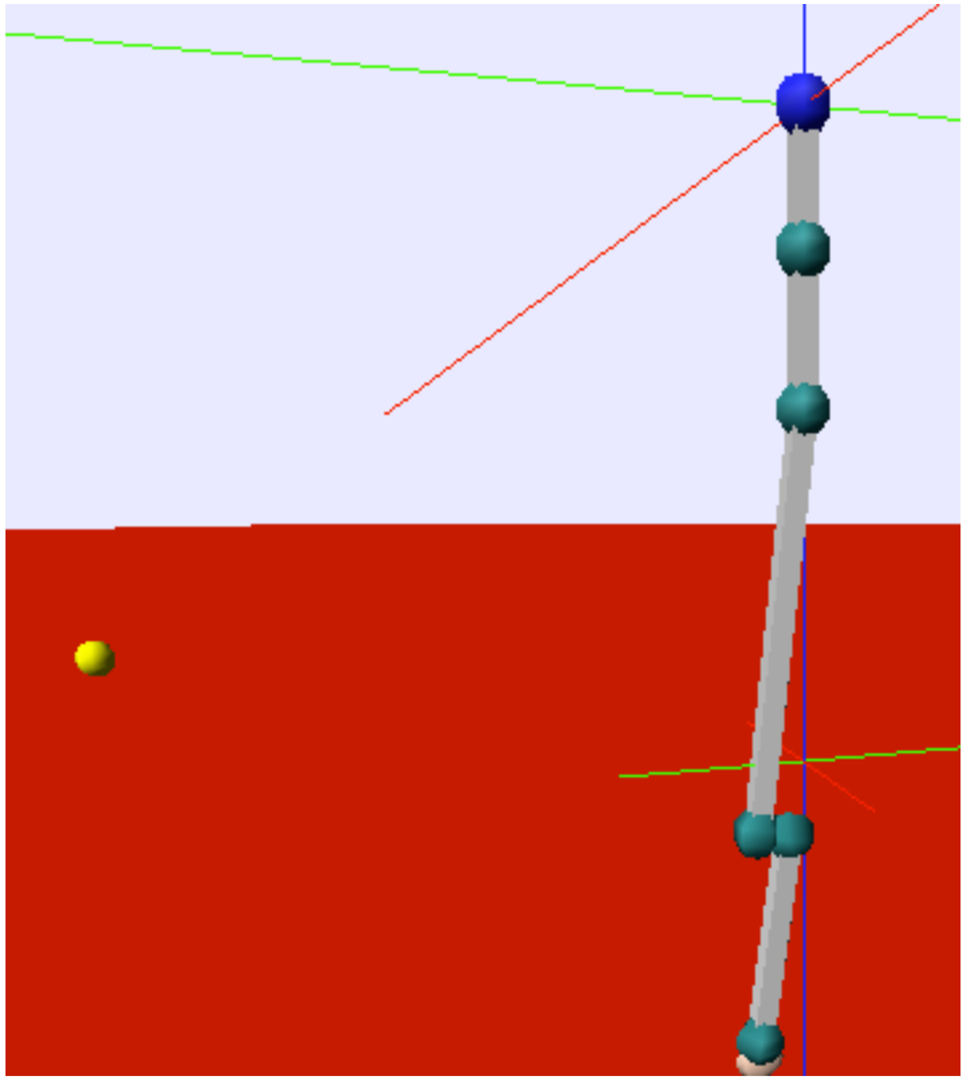
Proportional Feedback

- Compute torque based on position error

$$u = K_p(q_d - q)$$

$$\mathbf{q}_d = \begin{bmatrix} 0 \\ 0.9 \\ 0 \\ 0.9 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{\mathbf{q}}_d = 0$$



- Stable, but very underdamped leading to poor tracking

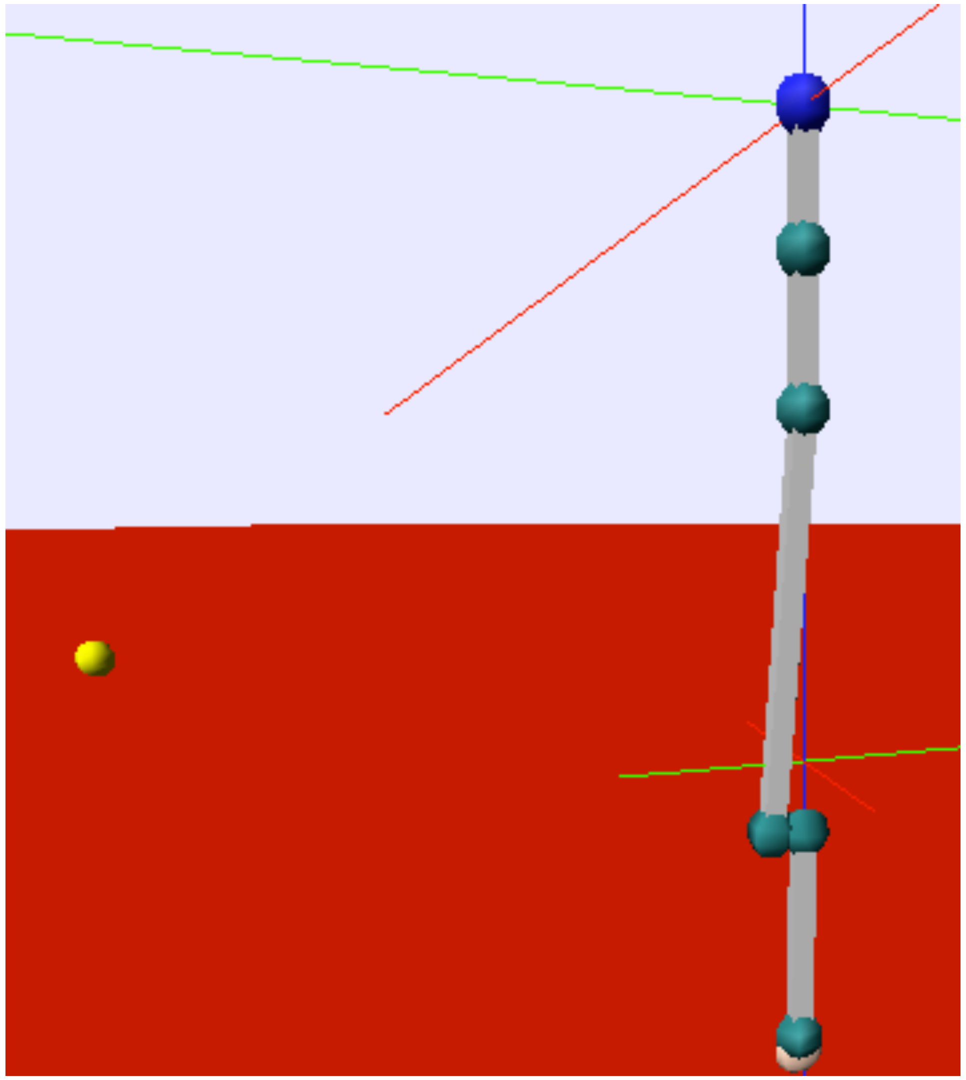
Derivative Feedback

- Compute torque based on position and velocity errors

$$u = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$$

$$\mathbf{q}_d = \begin{bmatrix} 0 \\ 0.9 \\ 0 \\ 0.9 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

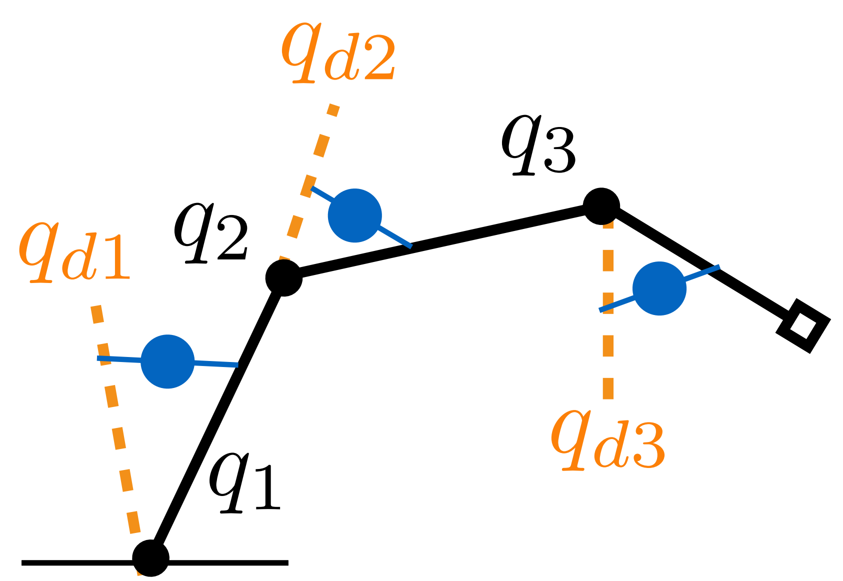
$$\dot{\mathbf{q}}_d = 0$$

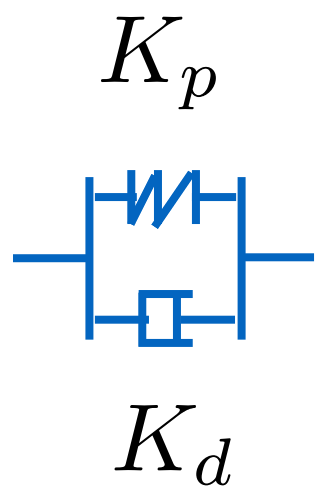


- Stable, but a bit underdamped
- Increase d gain to remove overshoot (overdamp)

Physical Interpretation

- Controller simulates virtual spring and damper

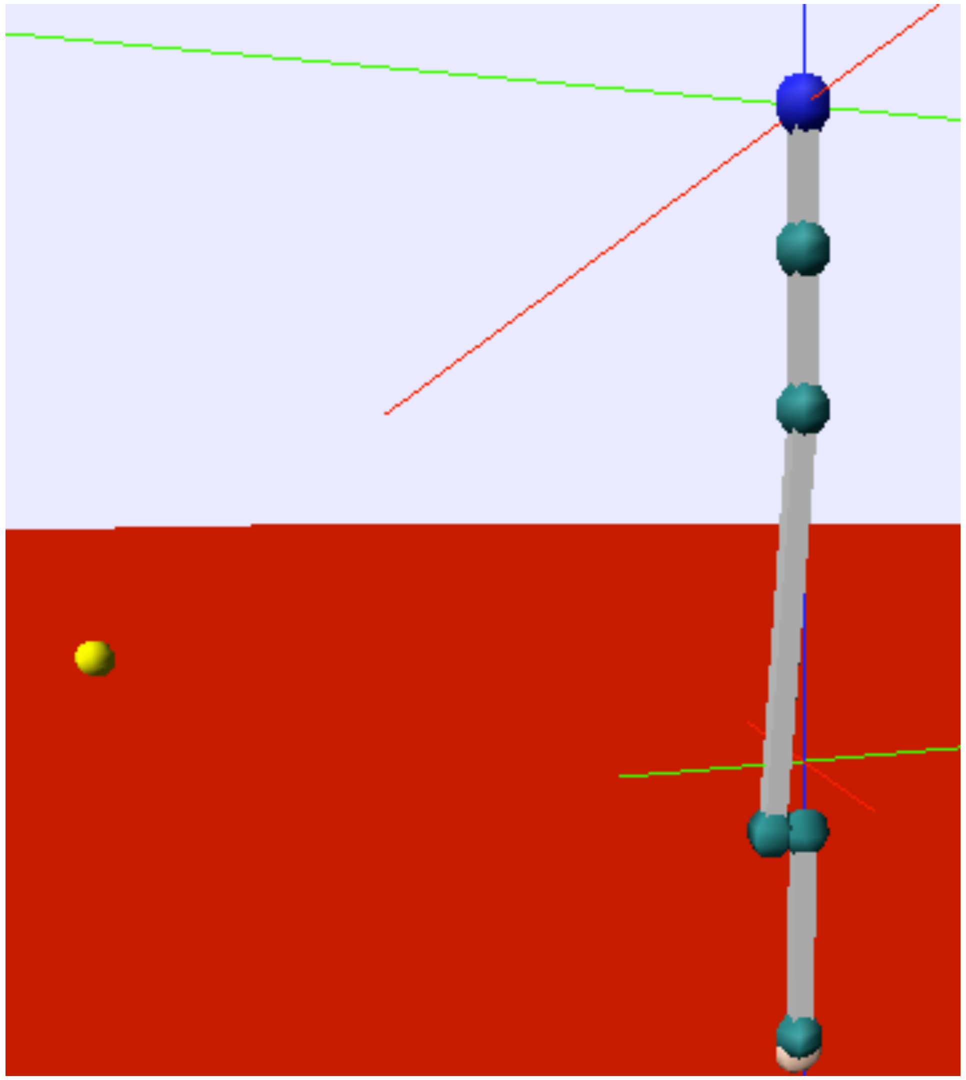




  - Known as simple joint-space impedance control
  - Passive elements- spring and damper do not create energy
  - Passive systems are inherently stable
    Lyapunov stability criterion:
    - Energy in system is minimum at desired pose
    - Energy is constantly decreasing over time

Improving Performance

- What about the offset at the end?



- How can we reduce this error?

Integral Feedback

- Could include an integral term to create PID controller

$$u = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_i \int_{-\infty}^{t} (q_d - q)\mathrm{d}\tau$$

- Integral term ensures error will be removed over time

- Why may an integral term be undesirable?

Integral Feedback

- Could include an integral term to create PID controller

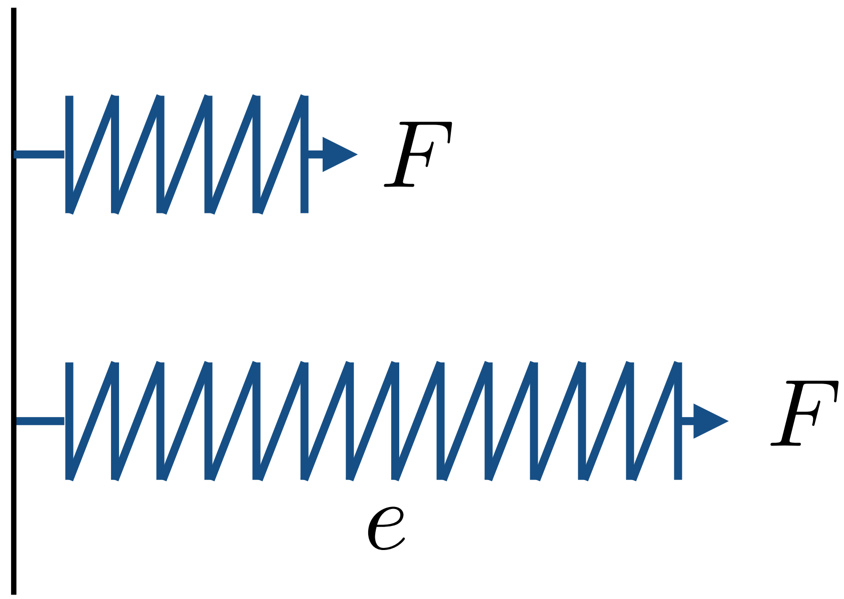
$$u = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_i \int_{-\infty}^{t} (q_d - q)\mathrm{d}\tau$$

- Integral term ensures error will be removed over time

- Integral is useful for constant desired joint angle
  - Usually use PD for tracking dynamic movements
  - Integral adds memory and wind-up
  - Consider moving arm to straight down afterwards:
    arm will initially have an additional offset due to integral

High Gains

- Error can be reduced by increasing the (P) gains
  - Stiffer springs result in smaller offsets

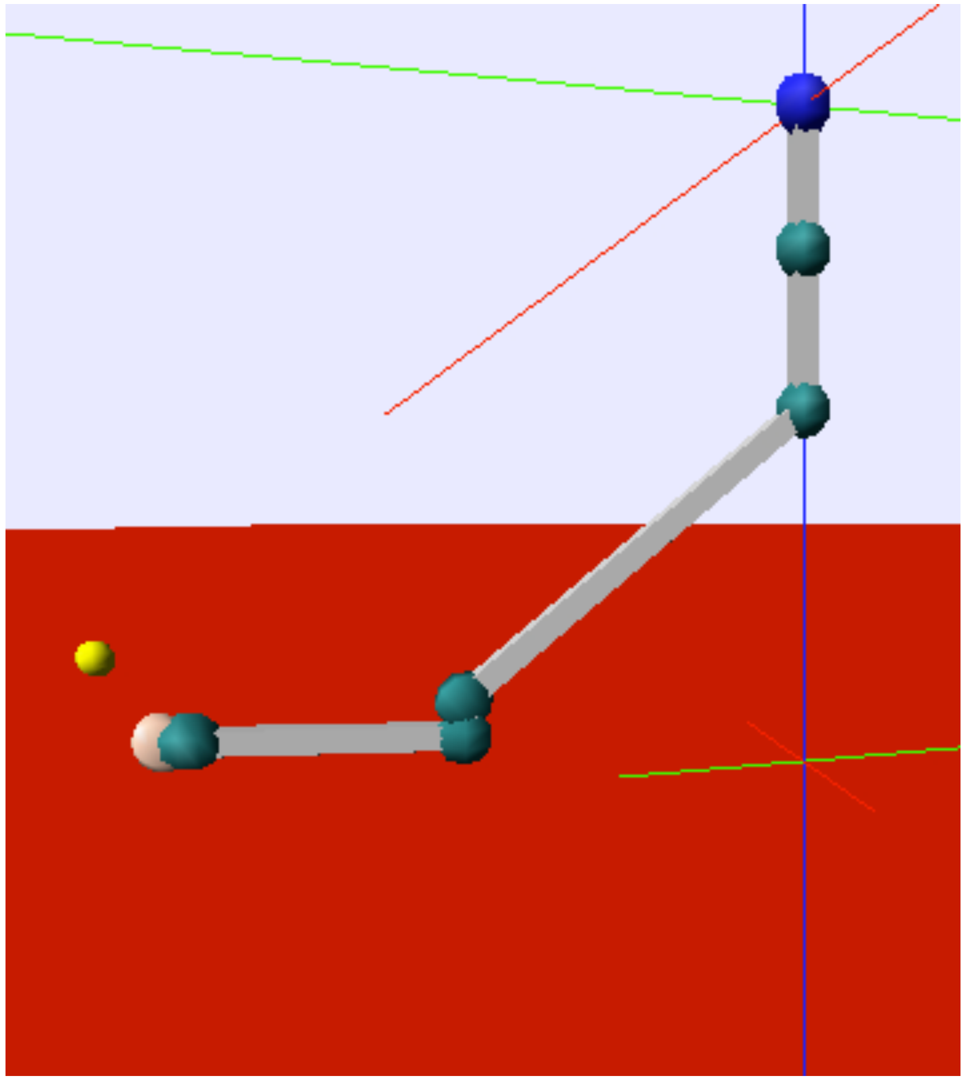


$$e = F/K_p$$

- Require large torques for executing dynamic motions
- Unsafe for humans and unstructured environments
  - Want robot to give way to perturbations from humans

Error

- What is causing the offset in the first place?



Gravity- weight of the robot's arm

- Can directly compensate for gravity given robot model

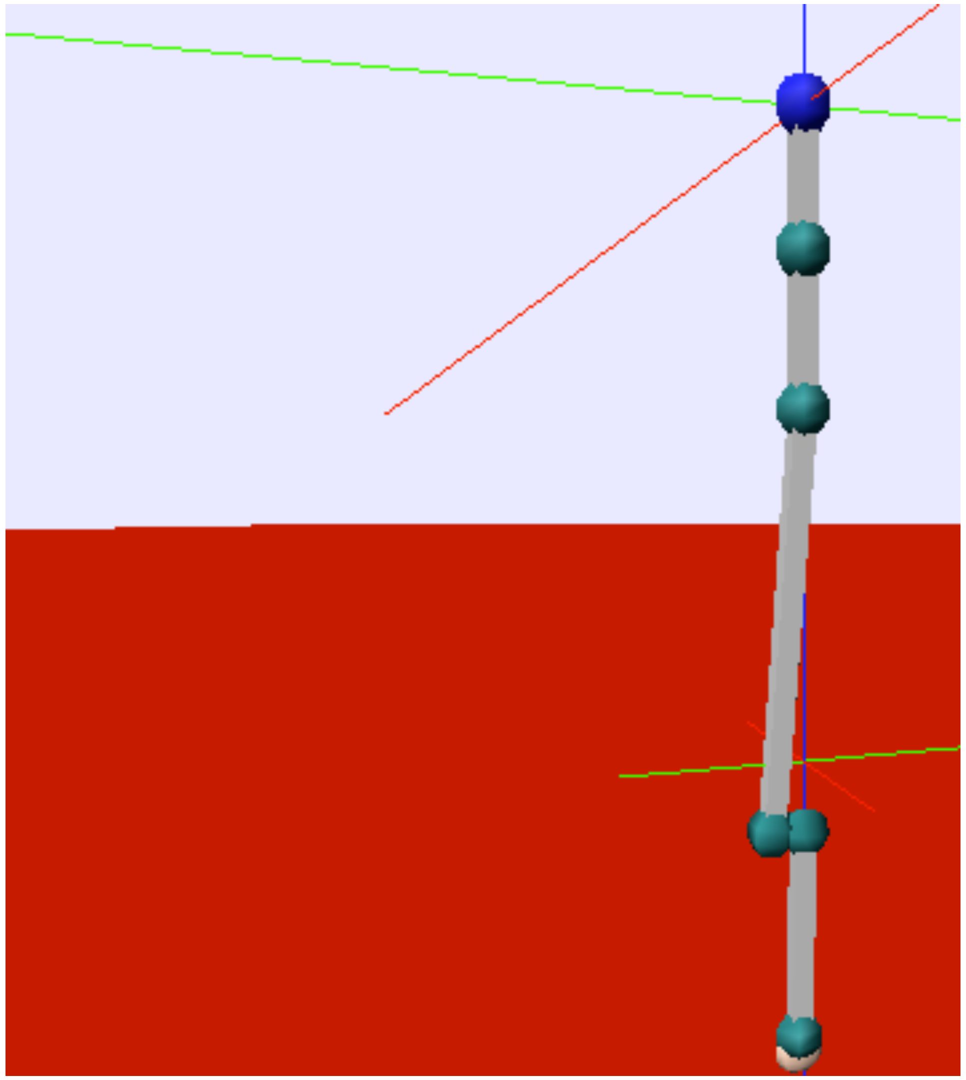
Gravity Compensation

- Offset is caused by the weight of the robot arm

$$u = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + g(q)$$

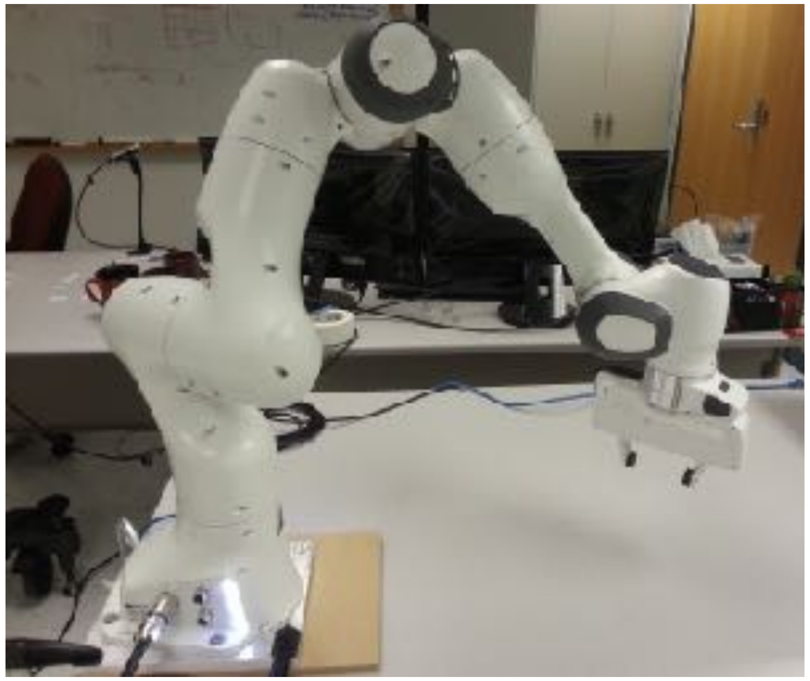
$$\mathbf{q}_d = \begin{bmatrix} 0 \\ 0.9 \\ 0 \\ 0.9 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{\mathbf{q}}_d = 0$$



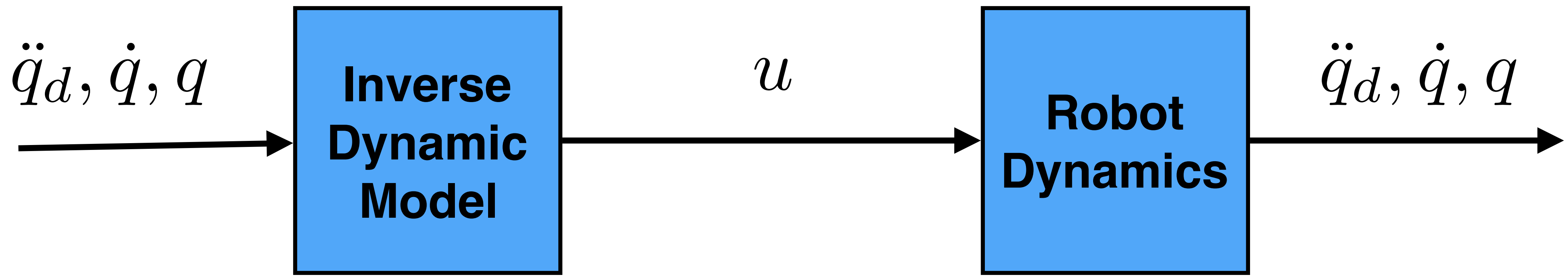
- Stable and good tracking (given large jump in desired)
- What about other dynamic effects than just gravity?

Model-Based Control



Model-based Control

- Why not compensate for full dynamics?



Mass Matrix

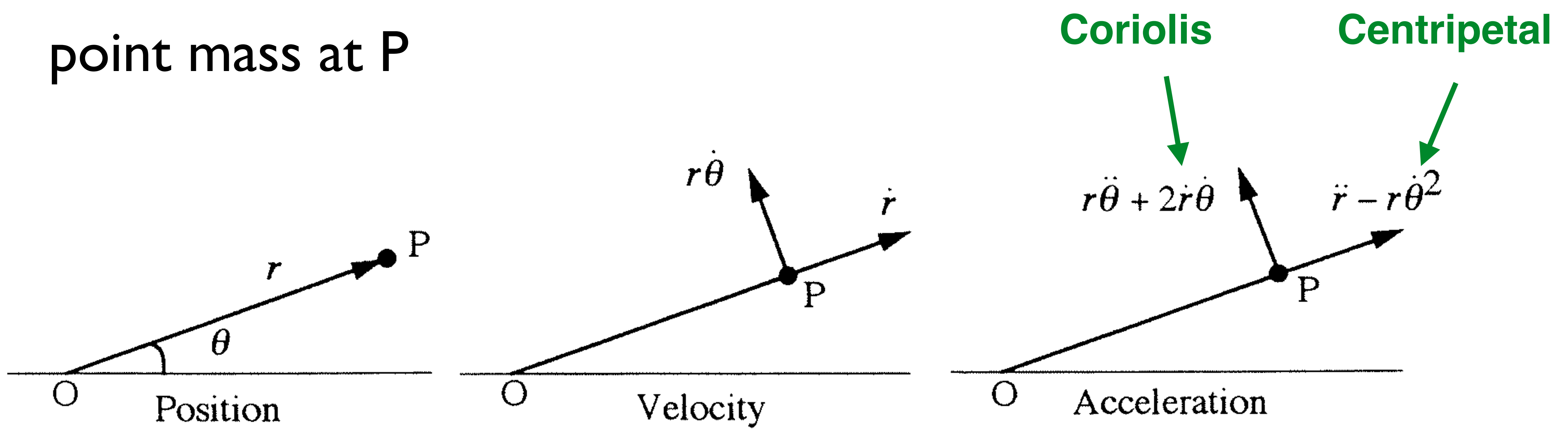
Coriolis + Centripetal Forces

Gravity

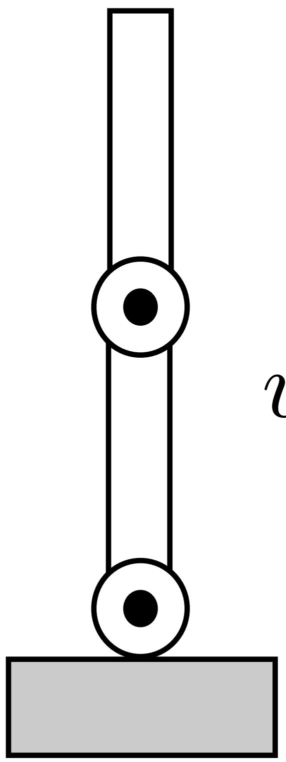
- Forward Dynamics: $$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\mathbf{u} - \mathbf{c}(\dot{\mathbf{q}}, \mathbf{q}) - \mathbf{g}(\mathbf{q}))$$
- Inverse Dynamics: $$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_\mathbf{d} + \mathbf{c}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{g}(\mathbf{q})$$
- When combined: $$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_\mathbf{d}$$
- Need to compensate for model errors and perturbations

Coriolis and Centripetal Forces

- Consider a horizontal robot with:
  - rotational joint angle θ
  - prismatic joint extension r
  - point mass at P



Example Dynamics



$$
\begin{aligned}
u_1 \quad = \quad & [m_1 l_{g1}^2 + J_1 + m_2(l_1^2 + l_{g2}^2 + 2 l_1 l_{g2} \cos\theta_2) + J_2]\ddot{\theta}_1 \\
+ \quad & [m_2(l_{g2}^2 + l_1 l_2 \cos\theta_2) + J_2]\ddot{\theta}_2 \qquad \text{Inertial Forces} \\
- \quad & 2 m_2 l_1 l_{g2} \dot{\theta}_1 \dot{\theta}_2 \sin\theta_2 \qquad \text{Coriolis Forces} \\
- \quad & 2 m_2 l_1 l_{g2} \dot{\theta}_1^2 \sin\theta_2 \qquad \text{Centripetal Forces} \\
+ \quad & m_1 g l_{g1} \cos\theta_1 + m_2 g (l_1 \cos\theta_1 + l_{g2} \cos(\theta_1 + \theta_2)) \qquad \text{Gravity} \\
u_2 \quad = \quad & [m_2(l_{g2}^2 + l_1 l_{g2} \cos\theta_2) + J_2]\ddot{\theta}_1 \\
+ \quad & (m_2 l_{g2}^2 + J_2)\ddot{\theta}_2 \qquad \text{Inertial Forces} \\
- \quad & m_2 l_1 l_{g2} \dot{\theta}_1^2 \sin\theta_2 \qquad \text{Centripetal Forces} \\
+ \quad & m_2 g l_{g2} \cos(\theta_1 + \theta_2) \qquad \text{Gravity}
\end{aligned}
$$

Note: J are the links' moments of inertia, not Jacobians

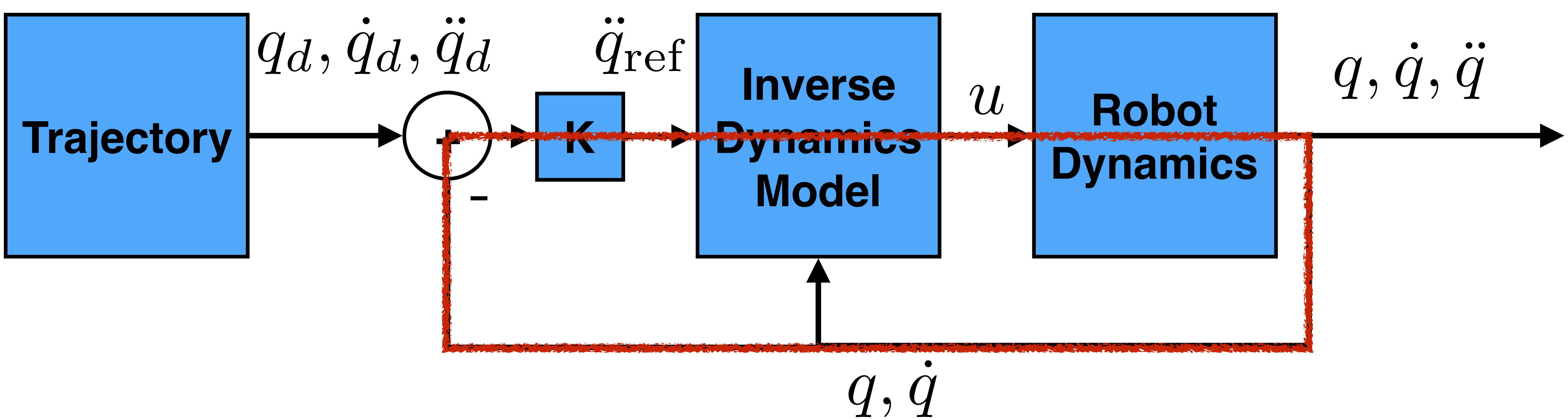
Model-Based Feedback Control

- For errors, adapt only our reference trajectory

$$\ddot{\mathbf{q}}_{\text{ref}} = \ddot{\mathbf{q}}_{\mathbf{d}} + \mathbf{K}_D(\dot{\mathbf{q}}_{\text{des}} - \dot{\mathbf{q}}) + \mathbf{K}_P(\mathbf{q}_{\text{des}} - \mathbf{q})$$

- Insert ref acceleration into inverse dynamics model

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_{\text{ref}} + \mathbf{c}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{g}(\mathbf{q})$$



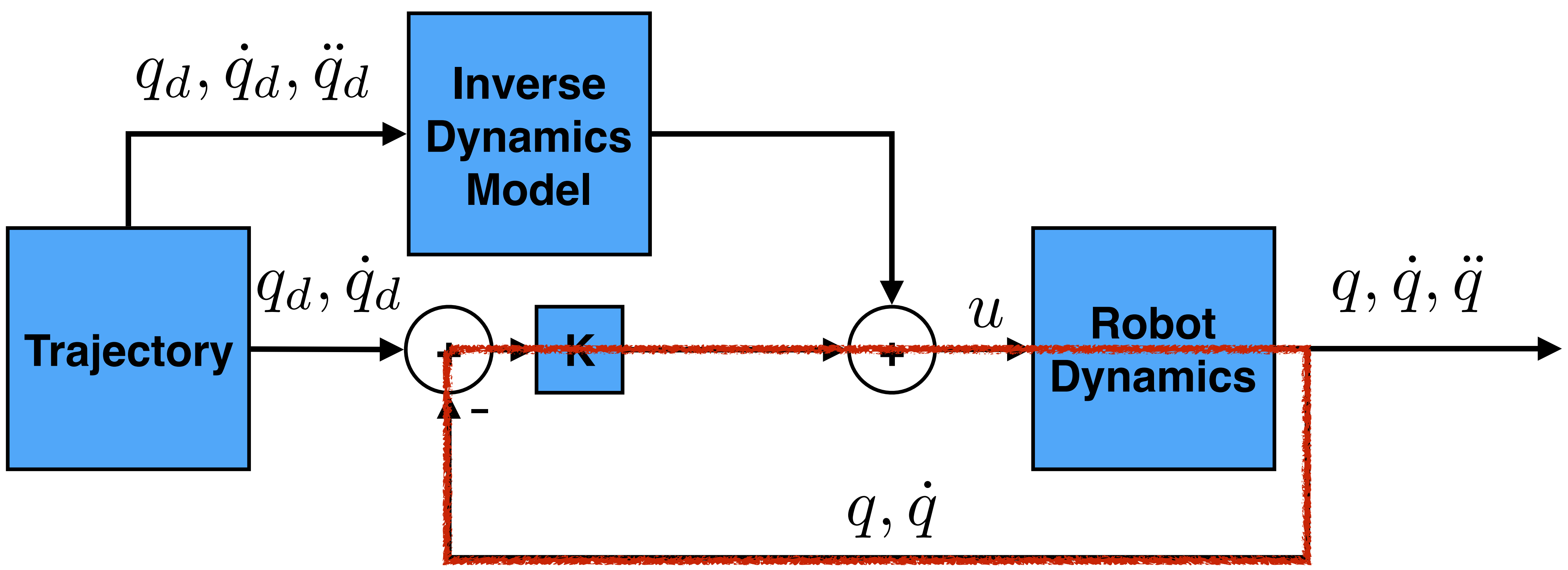
- Good error compensation given good inverse dynamics

Model-Based Feedforward Control

- Compute feedforward term based only on desired trajectory

$$u_{ff} = M(q_d)\ddot{q}_d + c(\dot{q}_d, q_d) + g(q_d)$$

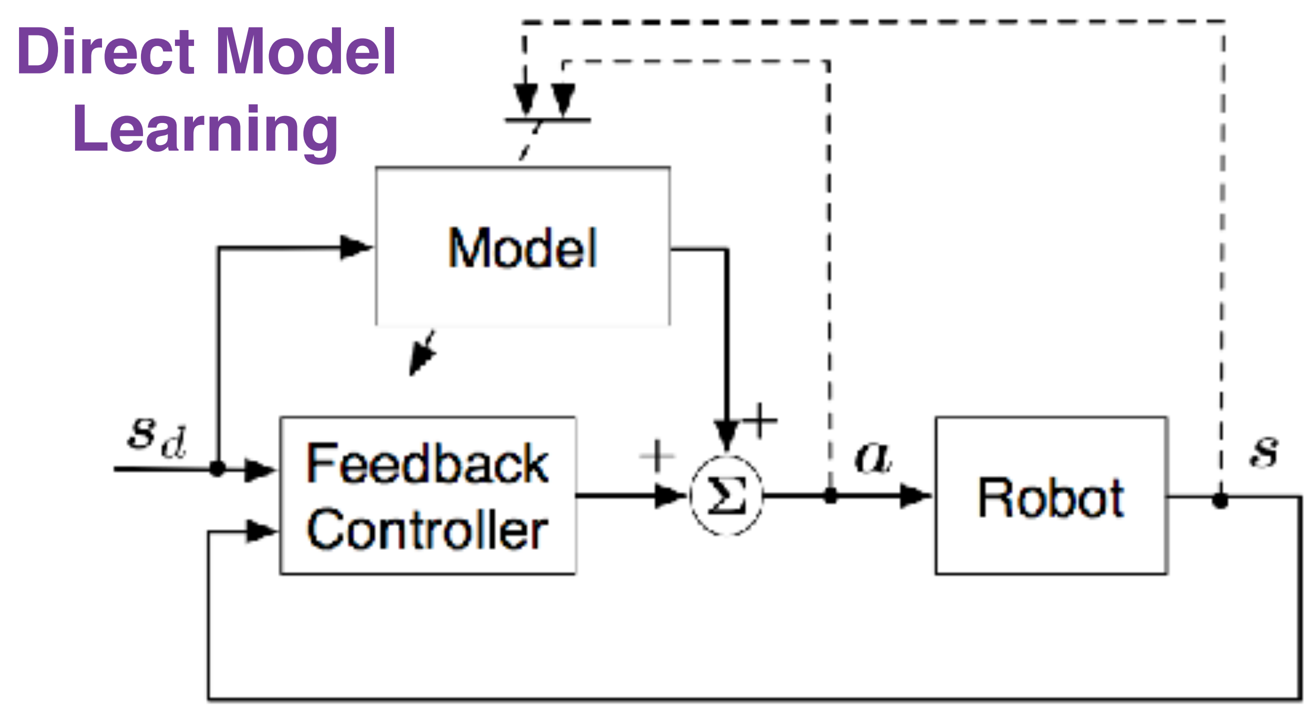
- Add feedback on top of feedforward term

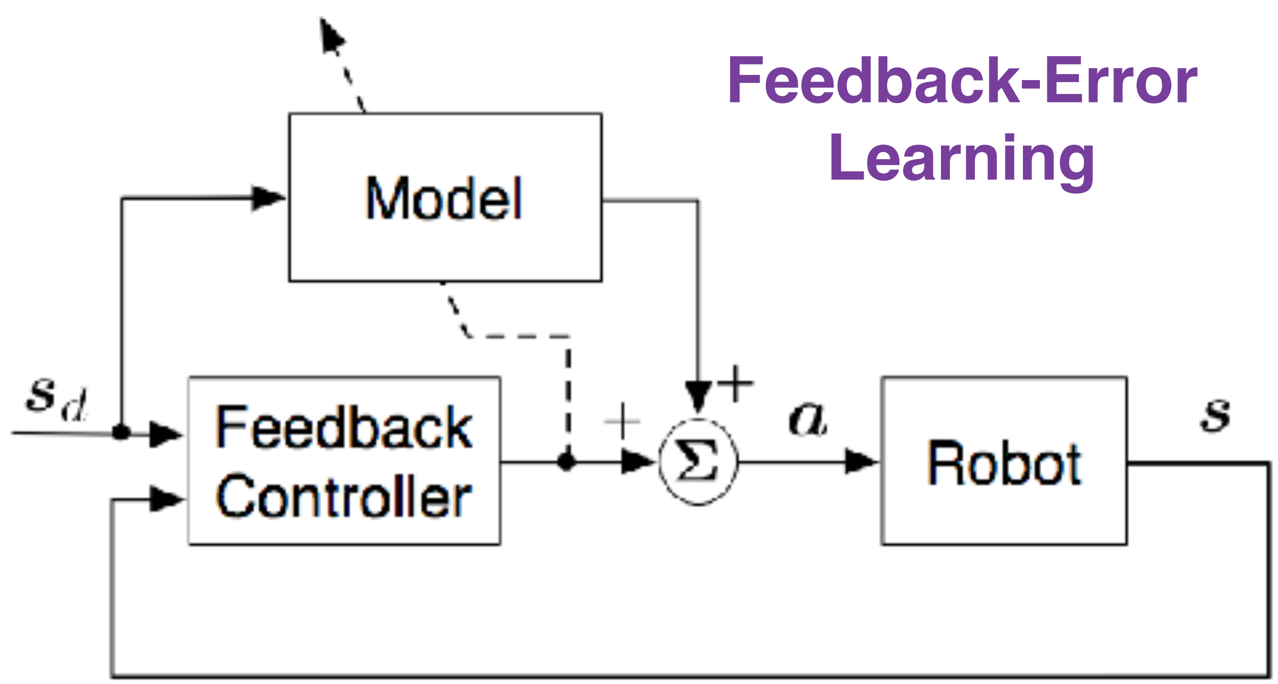


- Assumes robot is near desired trajectory
- Allows feedback to compensate more for model errors

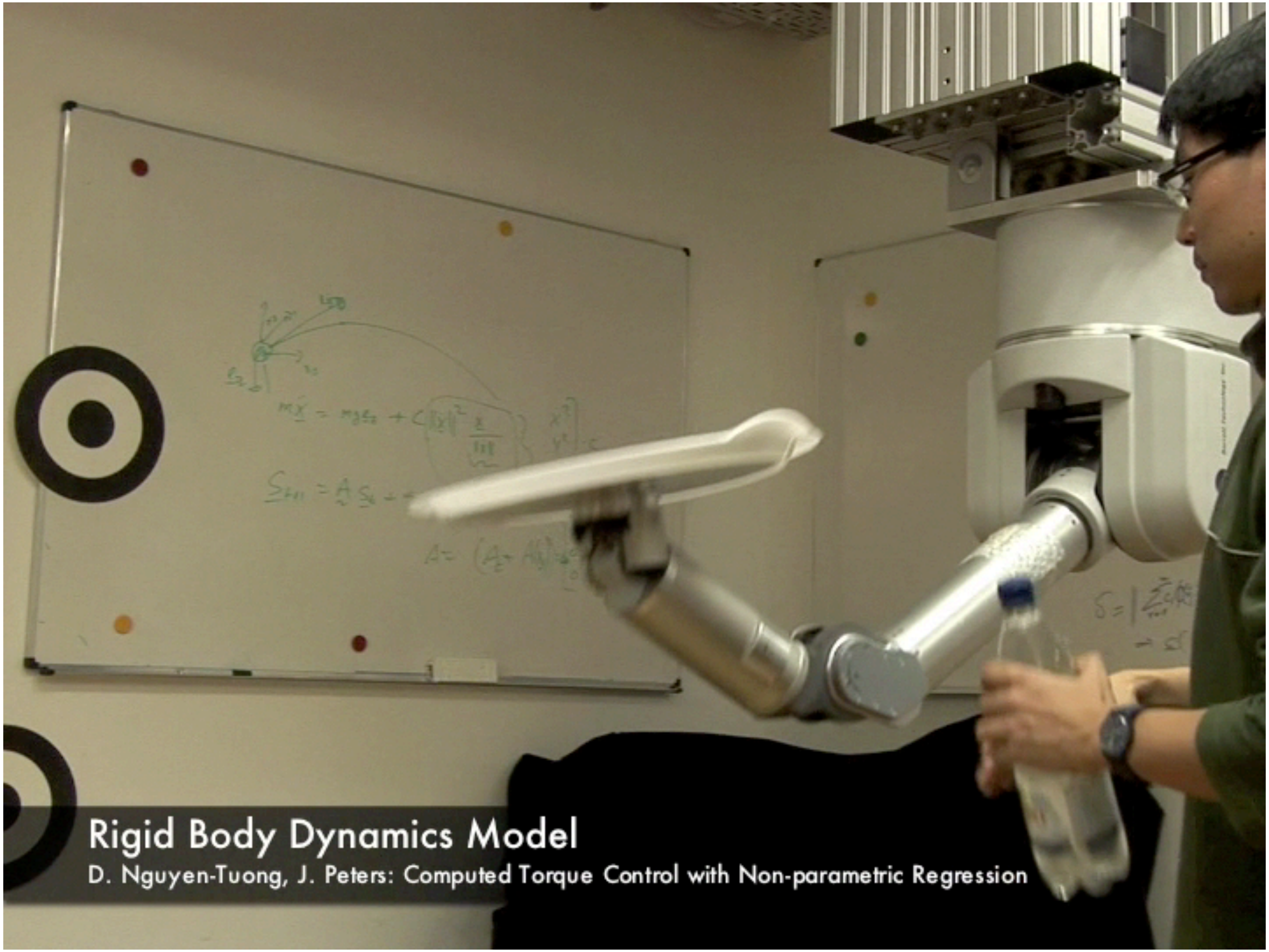
Model Learning

- What to do if the model is inaccurate? Learn a model





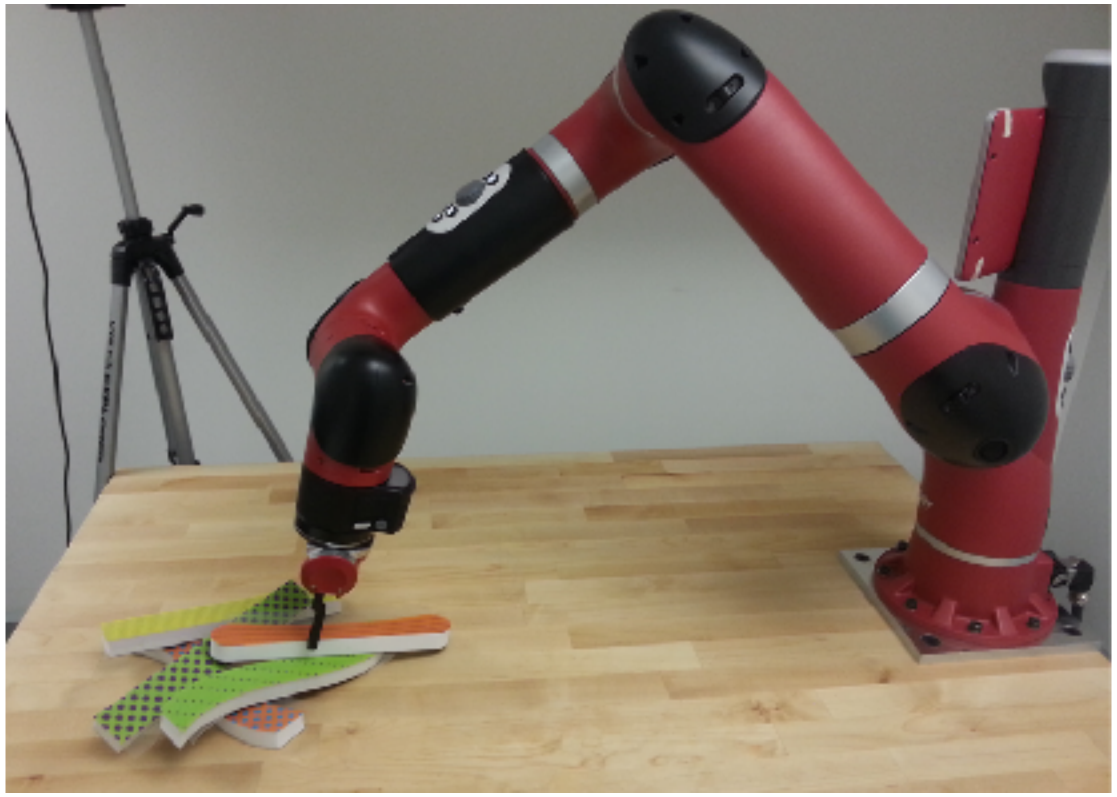
Model Learning



Interaction Control

Interaction Control

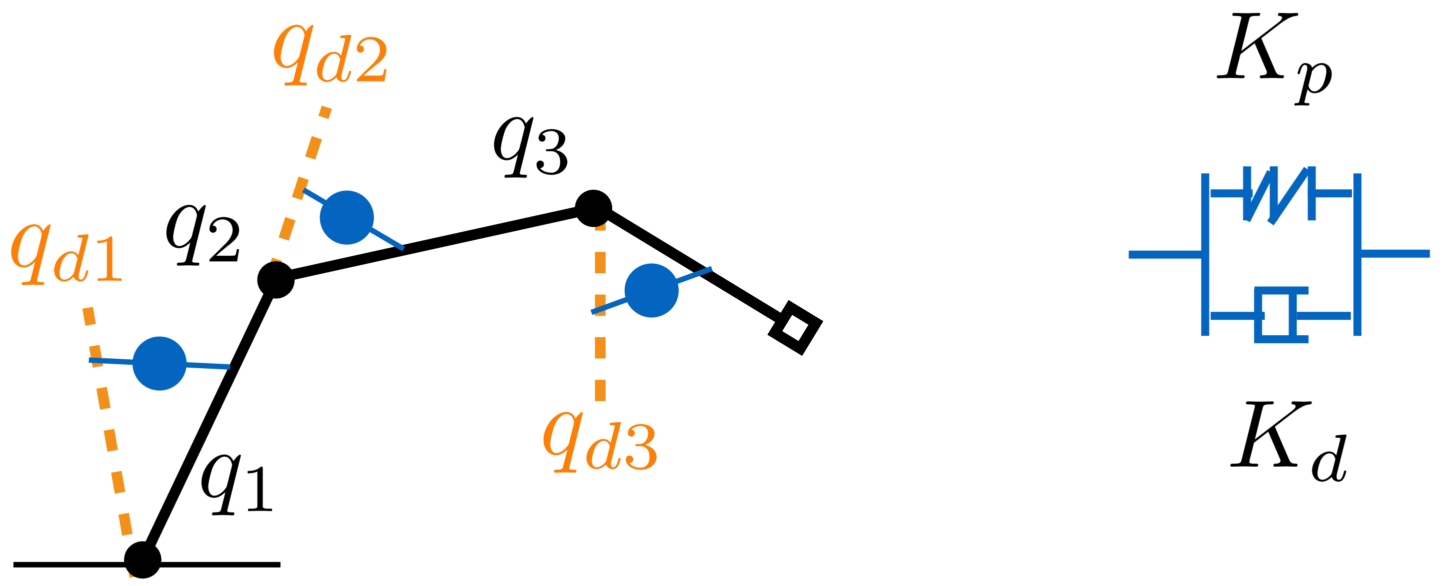
- Mainly looked at following a trajectory so far
- What about when the robot is in contact with objects?



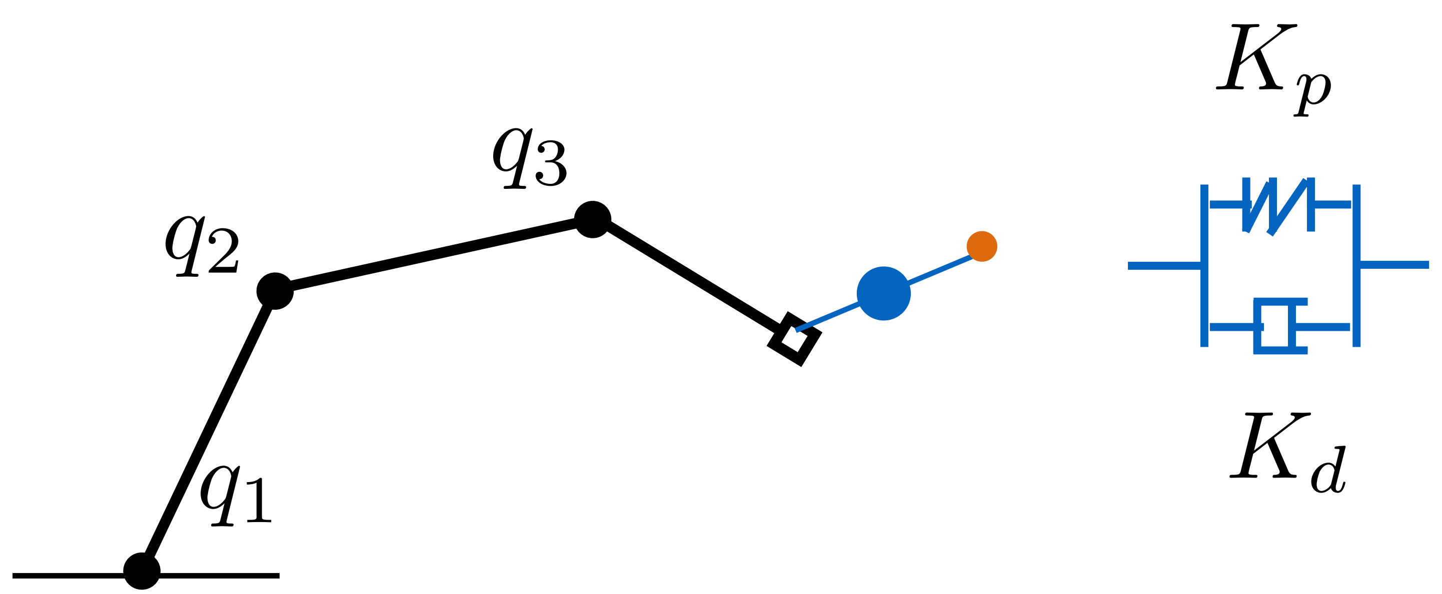
- Need to control interaction forces with environment
- If objects/environment act as masses (admittence), why not control with spring and damper (impedance)?

Impedance Control

- Joint space impedance control



- Task space impedance control



Simple Task-Space Impedance Control

- Simple task-space impedance controller

$$u = J(q)^T \left(K_p(x_d - x) + K_d(\dot{x}_d - \dot{x})\right)$$

Jacobian maps EE forces and torques to joint torques

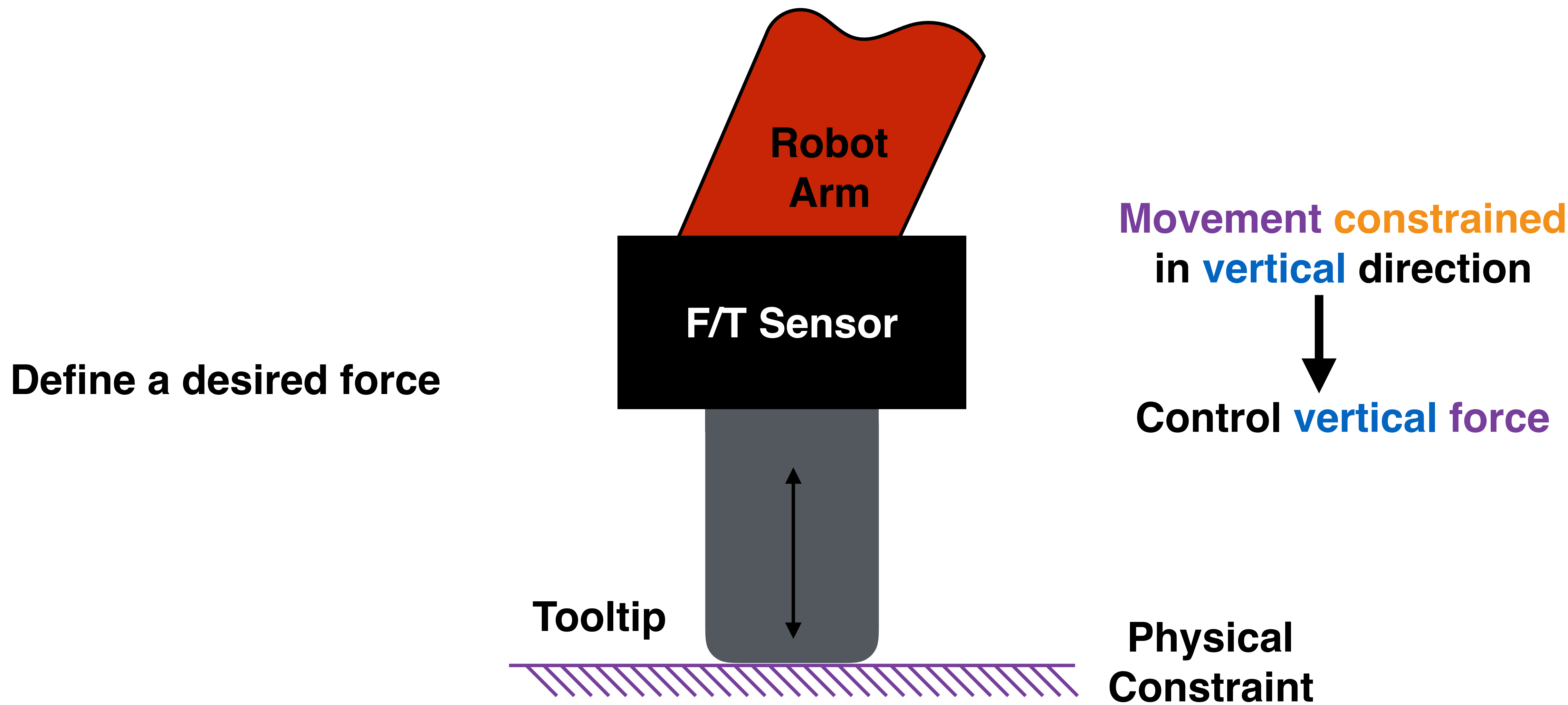
$f(q)$

$J(q)\dot{q}$

- Well-defined throughout workspace
- Does not compensate for the robot's own dynamics
  - Use with low friction and low inertia robots
- Indirect sensing - no force torque sensor required
- Interaction force created when desired x is in an object

Direct Force Control

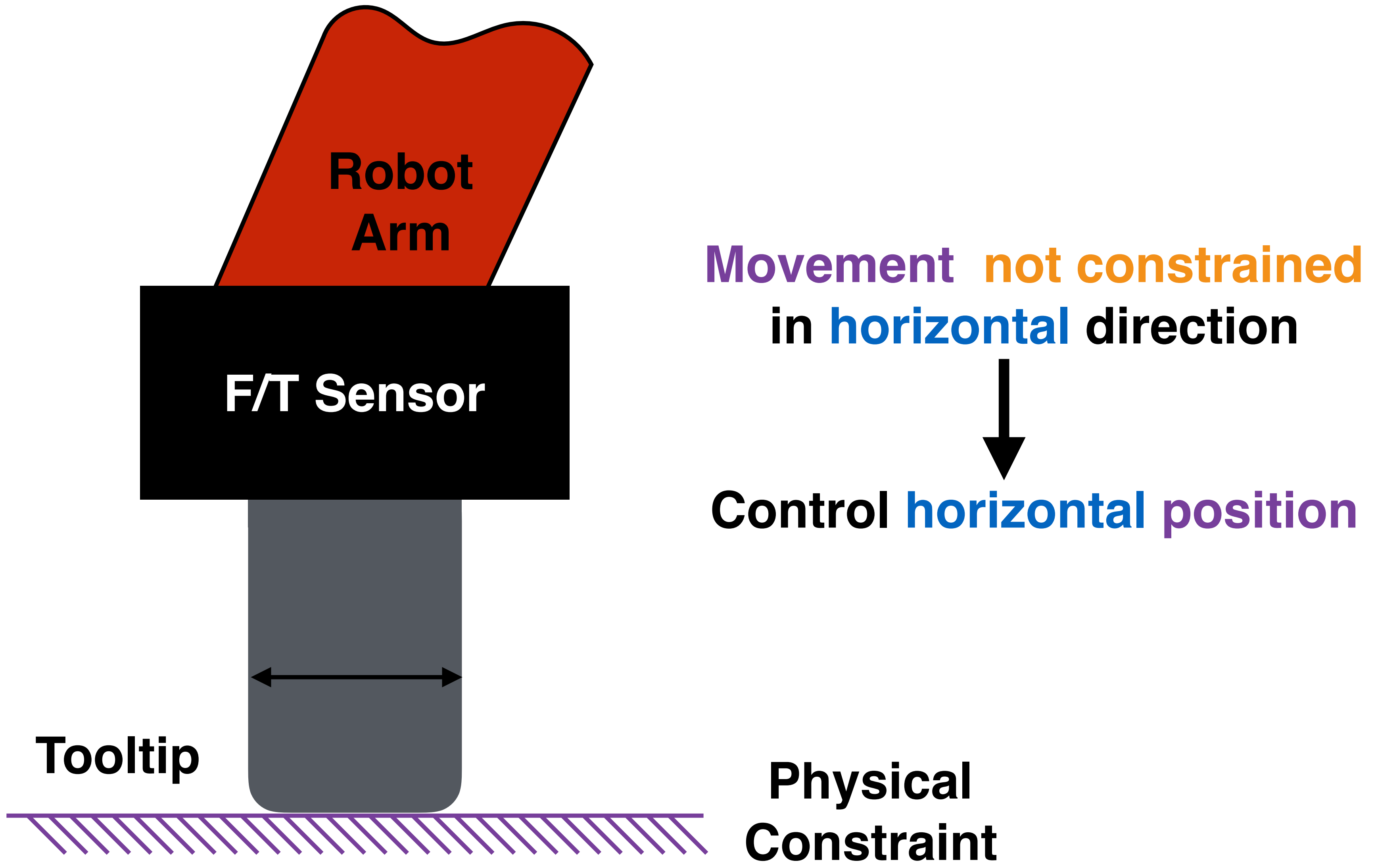
- Can directly sense forces using force-torque sensors



- Use Jacobian transpose to map forces to joints
- Use PI(D) to control torques to achieve desired force

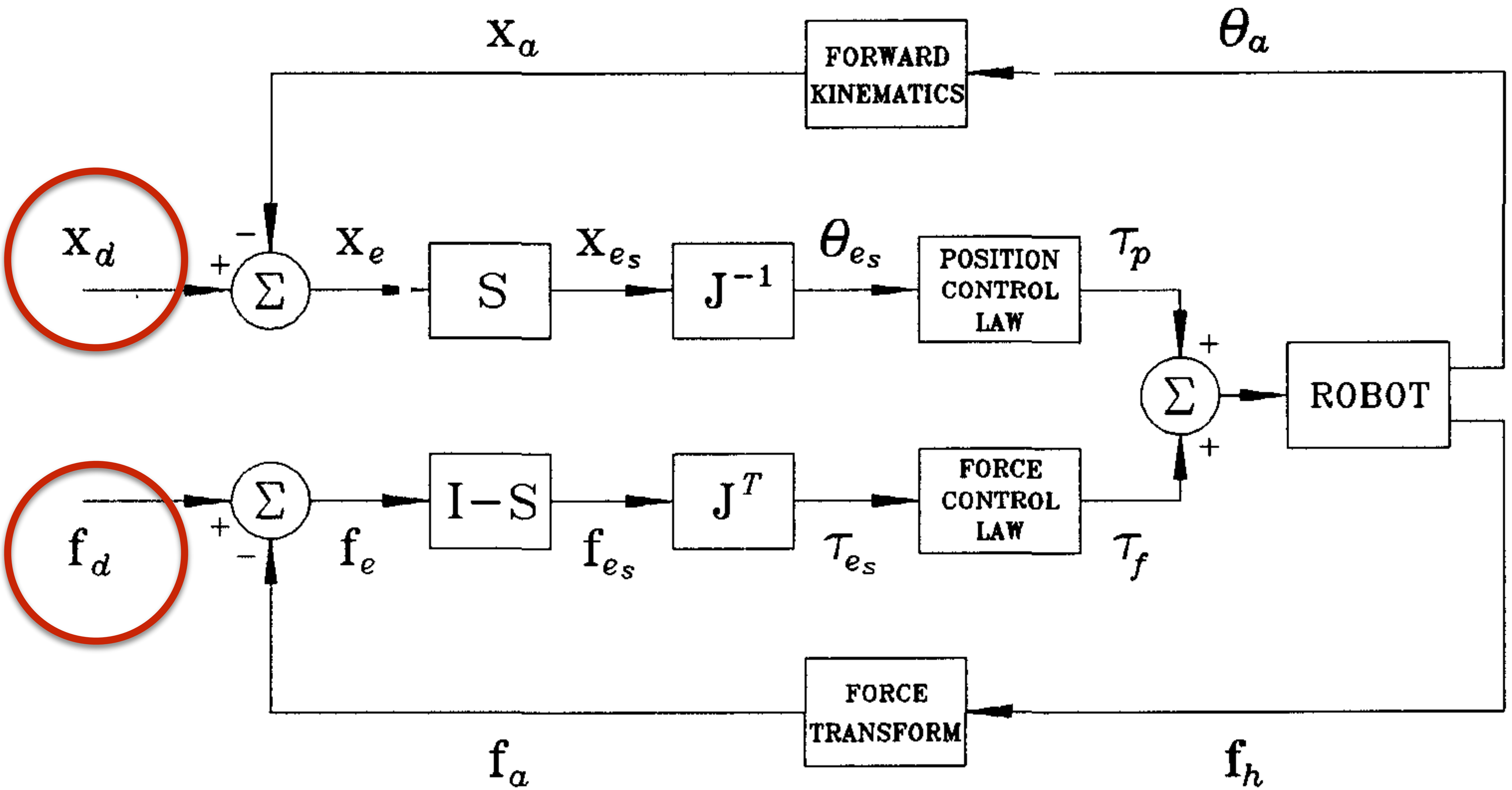
Direct Force Control

- What about the horizontal direction?



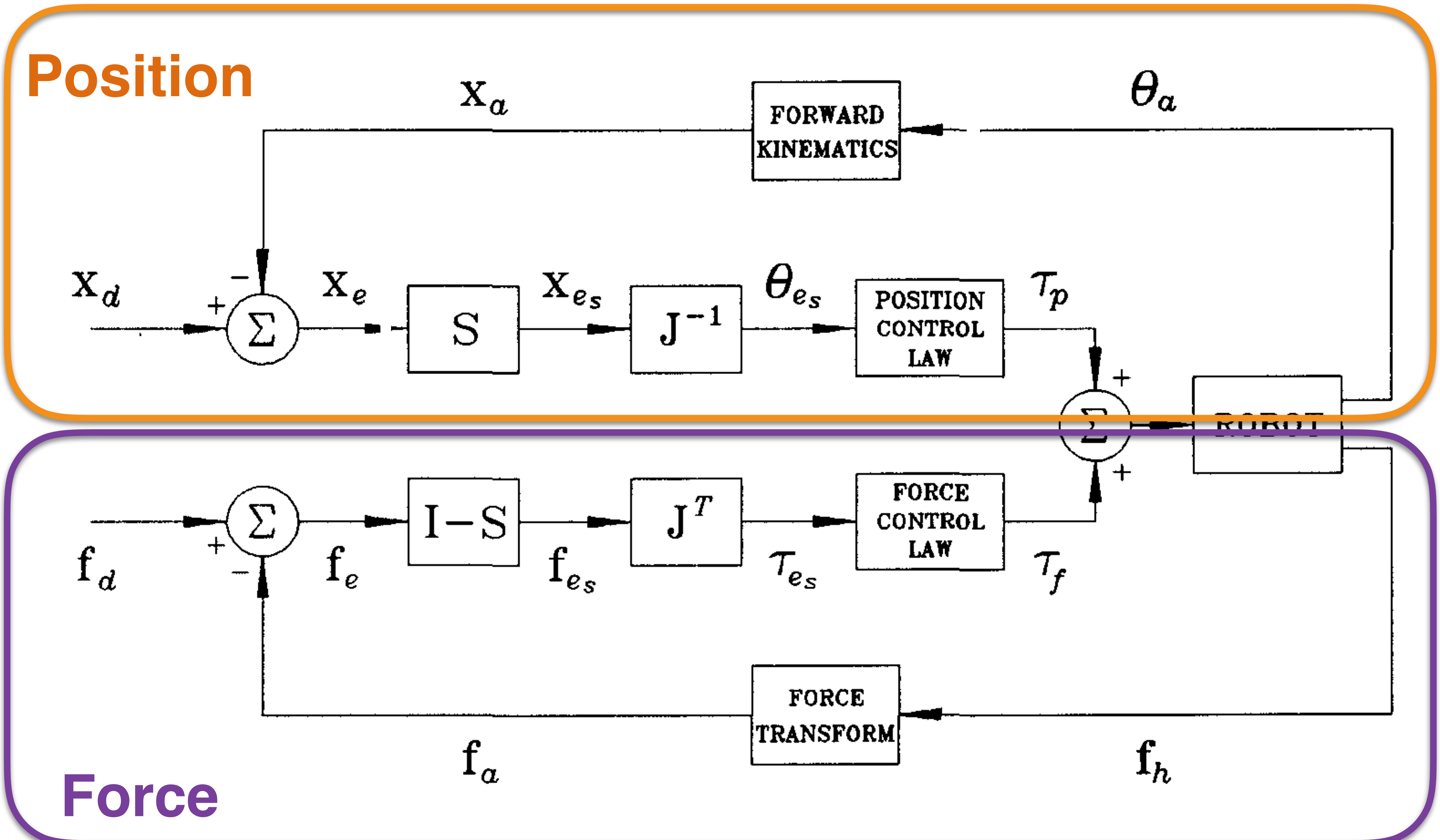
- No object to push against to control force (force constraint)
- Want to use position control for free space motions

Hybrid Force-Position Control



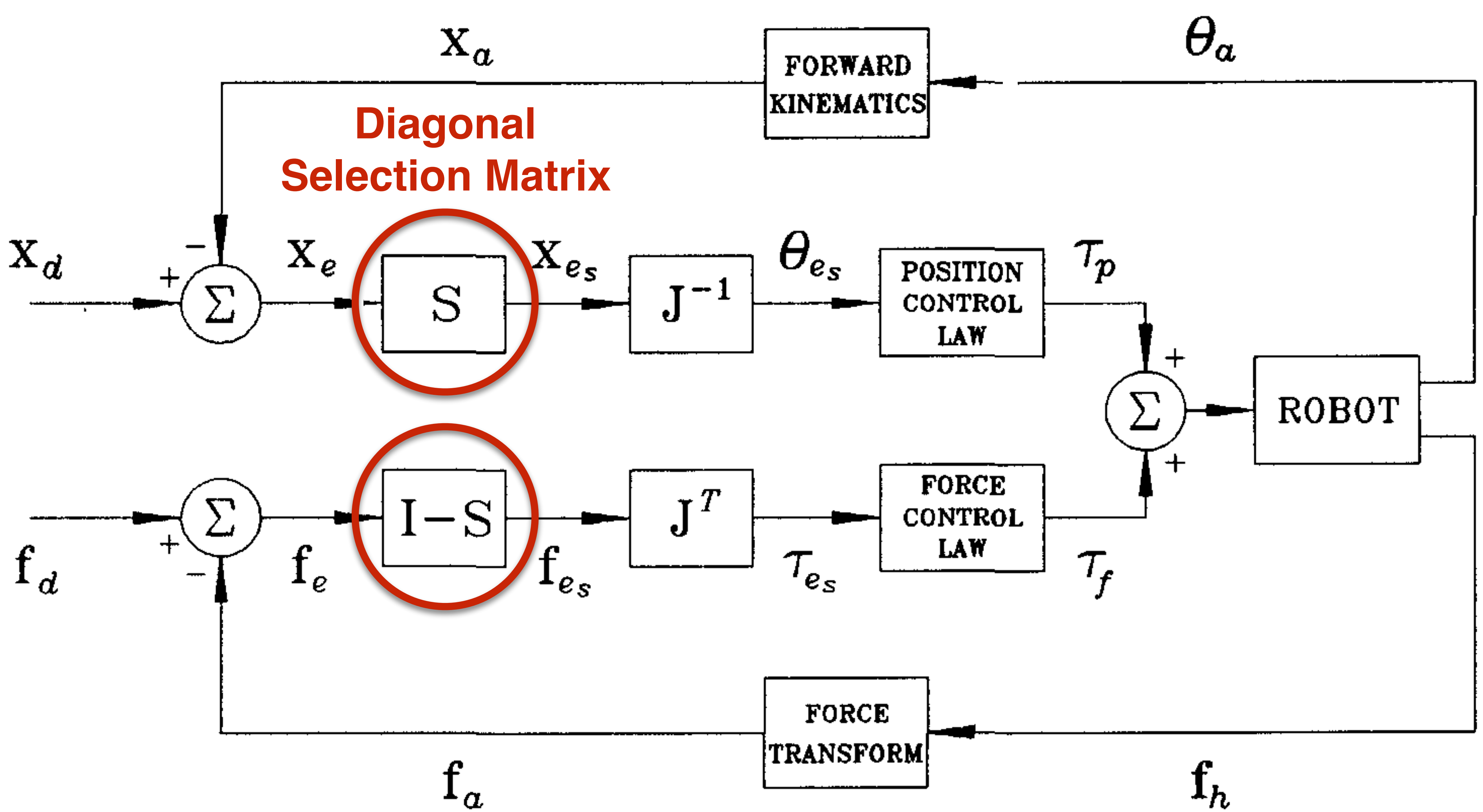
Raibert and Craig

Hybrid Force-Position Control



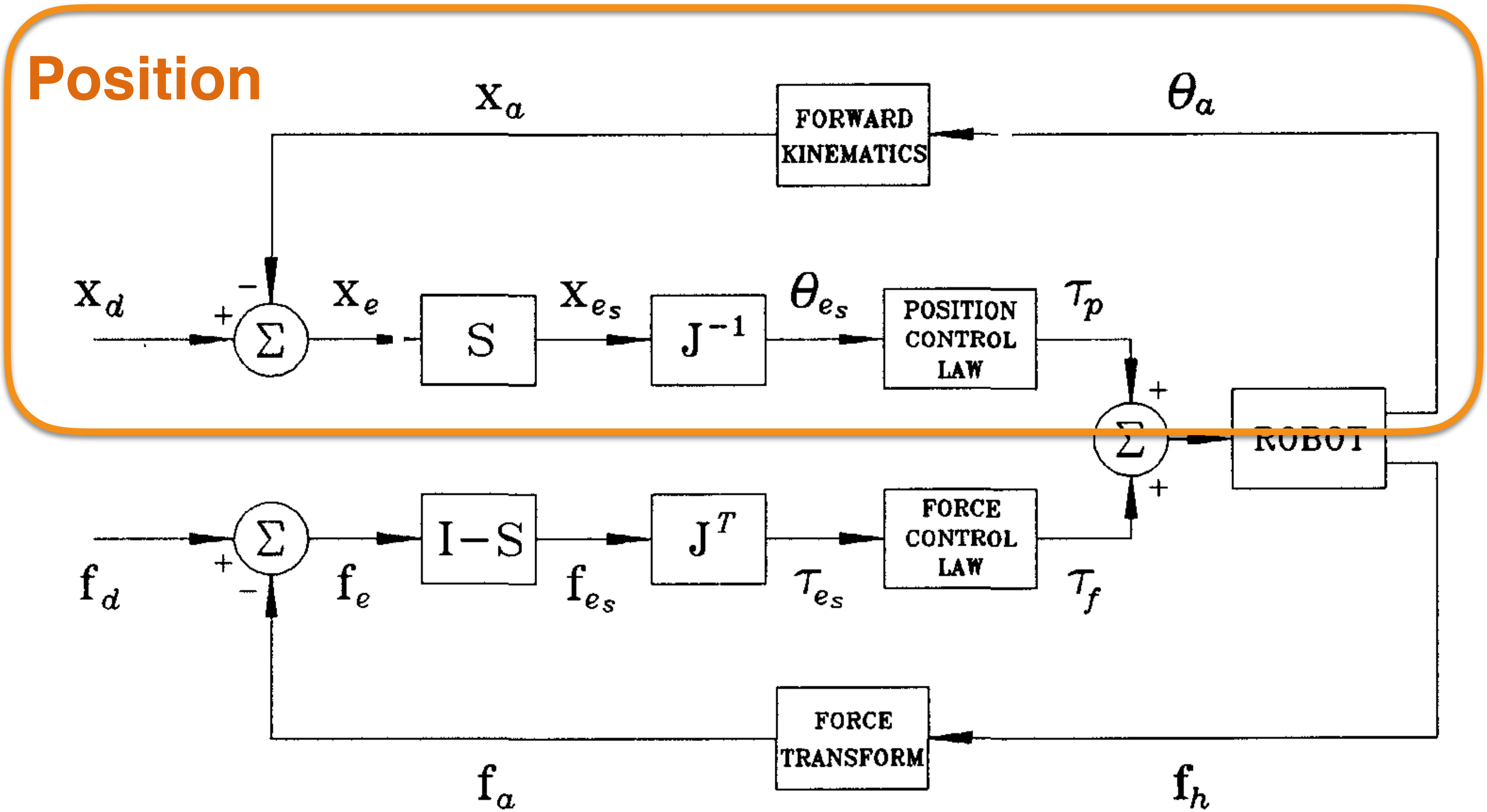
Raibert and Craig

Hybrid Force-Position Control



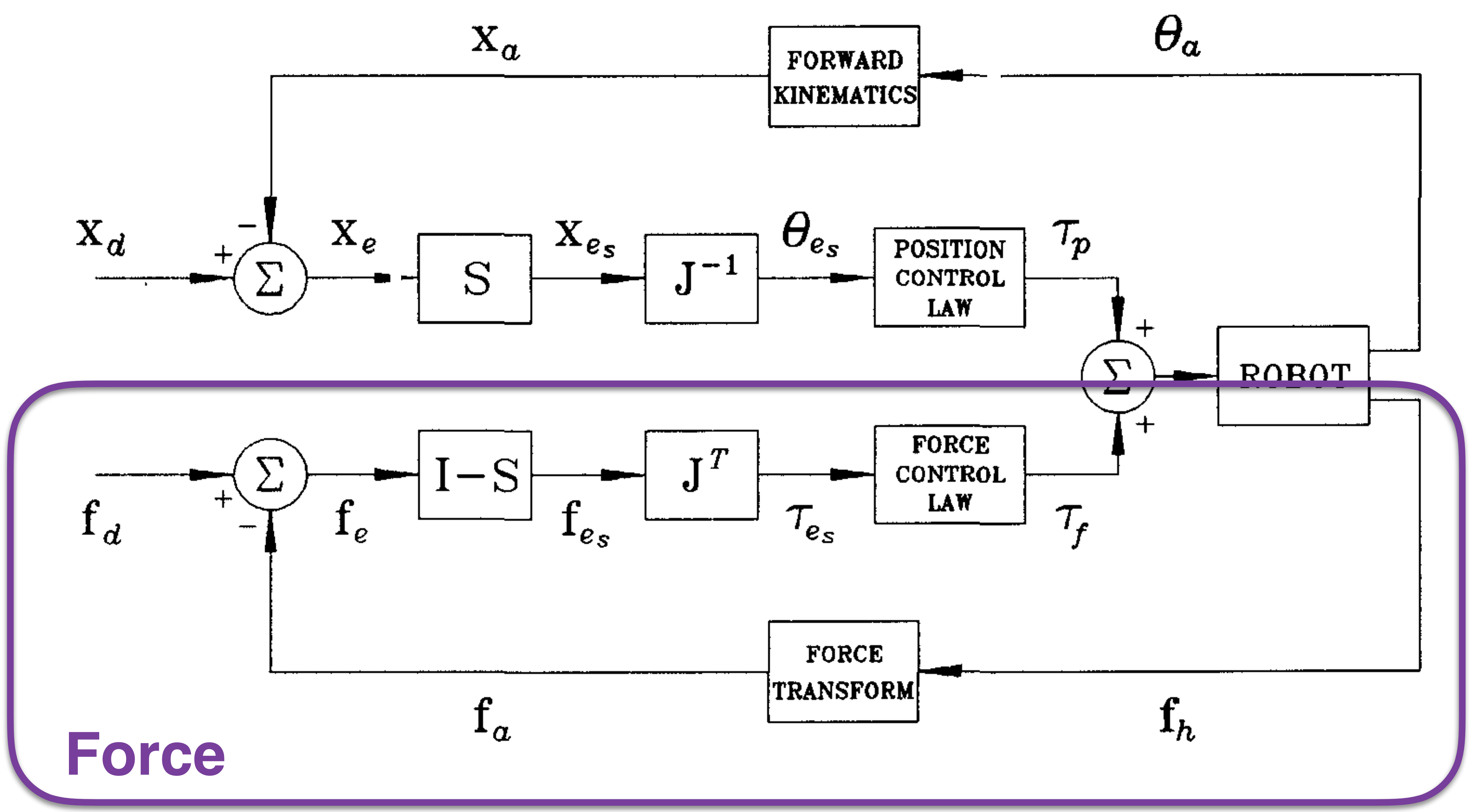
Raibert and Craig

Hybrid Force-Position Control



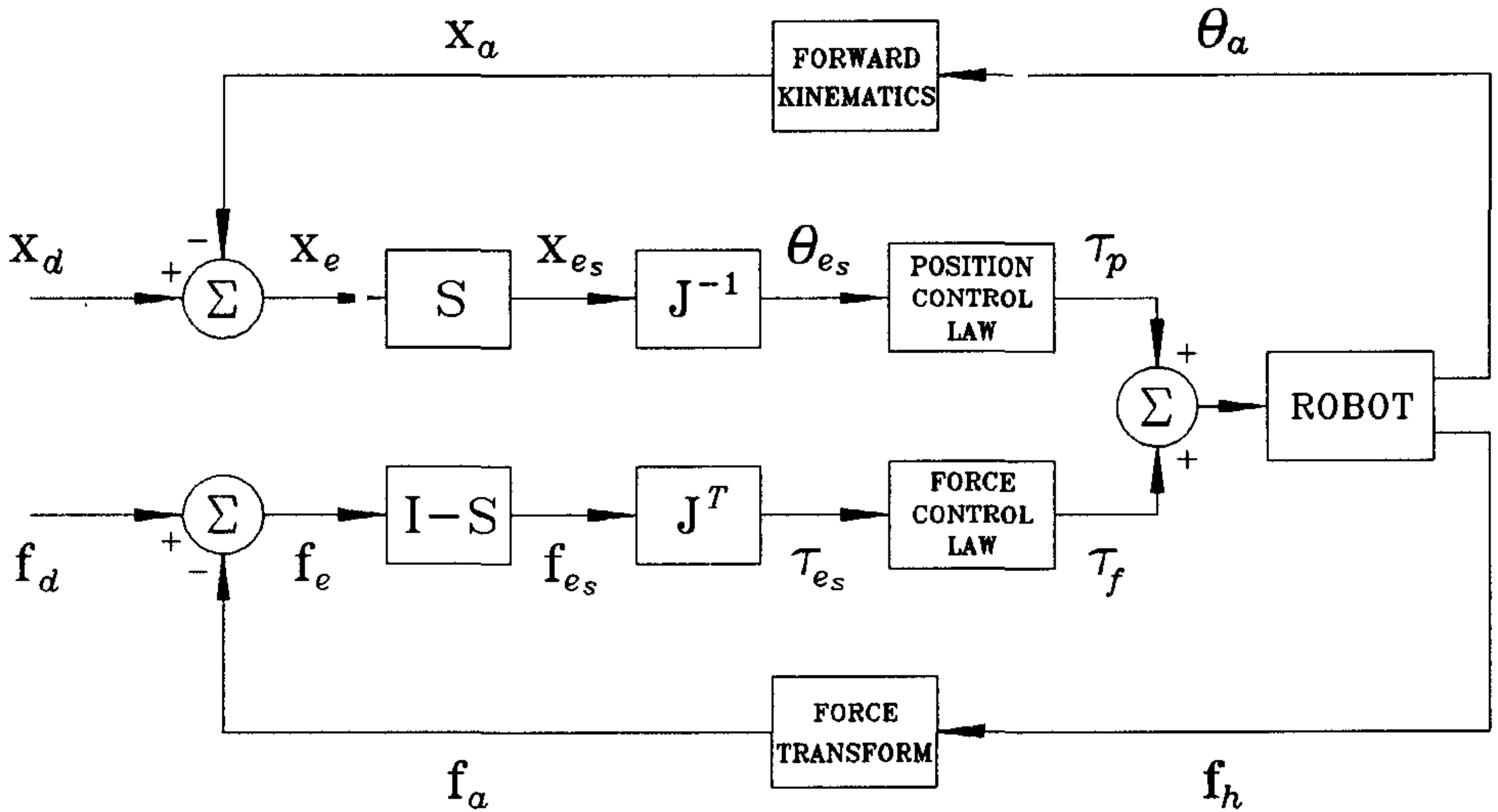
Raibert and Craig

Hybrid Force-Position Control



Raibert and Craig

Hybrid Force-Position Control



Raibert and Craig

Summer School on Impedance Control

Looking for more (binge-watchable) information?

http://summerschool.stiff-project.org/keynotes/index.html

Questions?