ME 599/699 Robot Modeling & Control
Feedback Control

Hasan Poonawala

You've Got A Plan. Now What?

So far: How to compute a trajectory $q(t)$ that gets you from start q_0 to goal q_g, while avoiding obstacles.

You've Got A Plan. Now What?

So far: How to compute a trajectory $q(t)$ that gets you from start q_0 to goal q_g, while avoiding obstacles.

This framework captures many robot (sub)tasks: driving, fetching soda from a fridge, robotic surgery, etc.

You've Got A Plan. Now What?

So far: How to compute a trajectory $q(t)$ that gets you from start q_0 to goal q_g, while avoiding obstacles.

This framework captures many robot (sub)tasks: driving, fetching soda from a fridge, robotic surgery, etc.

Your particular trajectory planning algorithm either

- Also provides a way to execute that trajectory
- Expects you to figure it out

You've Got A Plan. Now What?

So far: How to compute a trajectory $q(t)$ that gets you from start q_0 to goal q_g, while avoiding obstacles.

This framework captures many robot (sub)tasks: driving, fetching soda from a fridge, robotic surgery, etc.

Your particular trajectory planning algorithm either

- Also provides a way to execute that trajectory
- Expects you to figure it out

Our trajectory reflects how we want the configuration to change with time.

You've Got A Plan. Now What?

So far: How to compute a trajectory $q(t)$ that gets you from start q_0 to goal q_g, while avoiding obstacles.

This framework captures many robot (sub)tasks: driving, fetching soda from a fridge, robotic surgery, etc.

Your particular trajectory planning algorithm either

- Also provides a way to execute that trajectory
- Expects you to figure it out

Our trajectory reflects how we want the configuration to change with time.

The task of motion control is to achieve that change

Ideal Behavior

If you're lucky, when your robot says 'I want to go to there', it goes to there.

Ideal Behavior

If you're lucky, when your robot says 'I want to go to there', it goes to there.

Ideally, $q_d(t) = q(t)$ at all times

Reality

That's not how we get motion out of mechanical systems.

Reality

That's not how we get motion out of mechanical systems.

We need to identify the inputs for a system:

Control Input

A physical variable whose value we can instantly set to any value

Reality

That's not how we get motion out of mechanical systems.

We need to identify the inputs for a system:

Control Input

A physical variable whose value we can instantly set to any value

For a robot, the position is not something we can instantly set.

Reality

That's not how we get motion out of mechanical systems.

We need to identify the inputs for a system:

Control Input

A physical variable whose value we can instantly set to any value

For a robot, the position is not something we can instantly set.

What can we usually set instantaneously?

Reality

That's not how we get motion out of mechanical systems.

We need to identify the inputs for a system:

Control Input

A physical variable whose value we can instantly set to any value

For a robot, the position is not something we can instantly set.

What can we usually set instantaneously?

We can apply forces, that become accelerations (Newton's Second Law).

Reality

That's not how we get motion out of mechanical systems.

We need to identify the inputs for a system:

Control Input

A physical variable whose value we can instantly set to any value

For a robot, the position is not something we can instantly set.

What can we usually set instantaneously?

We can apply forces, that become accelerations (Newton's Second Law).

If we can apply large enough forces, we may assume that we can practically instantaneously set velocities. Eg: low inertia wheels.

A Point-Mass Robot

Consider a point-mass robot whose configuration q is 1D, its position on the real number line $\mathbb{R}$.

A Point-Mass Robot

Consider a point-mass robot whose configuration q is 1D, its position on the real number line $\mathbb{R}$.

Assume we can apply any force $F \in \mathbb{R}$ at time $t \in R$.

A Point-Mass Robot

Consider a point-mass robot whose configuration q is 1D, its position on the real number line $\mathbb{R}$.

Assume we can apply any force $F \in \mathbb{R}$ at time $t \in R$.

Newton's Second Law says that $\ddot{q}(t) = F(t)$.

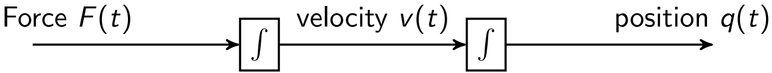
A Point-Mass Robot

Consider a point-mass robot whose configuration q is 1D, its position on the real number line $\mathbb{R}$.

Assume we can apply any force $F \in \mathbb{R}$ at time $t \in R$.

Newton's Second Law says that $\ddot{q}(t) = F(t)$.

In other words, $q(t) = \int_0^t \int_0^t F(t) dv \ dq$, where $v(t) = \dot{q}(t)$.



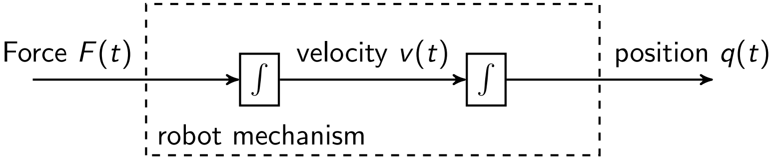
A Point-Mass Robot

Consider a point-mass robot whose configuration q is 1D, its position on the real number line $\mathbb{R}$.

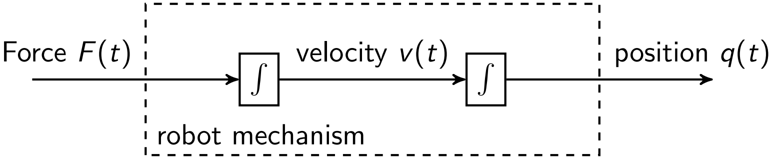
Assume we can apply any force $F \in \mathbb{R}$ at time $t \in R$.

Newton's Second Law says that $\ddot{q}(t) = F(t)$.

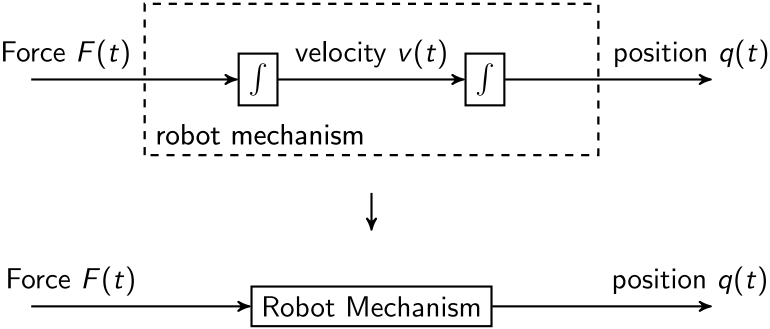
In other words, $q(t) = \int_0^t \int_0^t F(t) dv \; dq$, where $v(t) = \dot{q}(t)$.



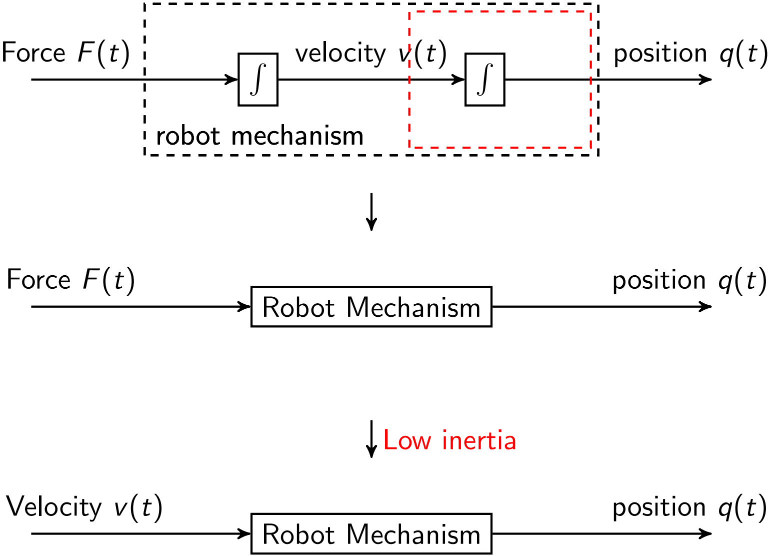
A Point-Mass Robot



A Point-Mass Robot



A Point-Mass Robot



Control

We want $q(t)$ to be $q_d(t)$

Control

We want $q(t)$ to be $q_d(t)$

We can't set $q(t)$ instantaneously, but we can set $F(t)$ instantaneously

Control

We want $q(t)$ to be $q_d(t)$

We can't set $q(t)$ instantaneously, but we can set $F(t)$ instantaneously

So, our goal is to choose $F(t)$ in a way that makes $q(t)$ become equal to $q_d(t)$. That is, we want

$$q_d(t) \rightarrow q(t).$$

Control

We want $q(t)$ to be $q_d(t)$

We can't set $q(t)$ instantaneously, but we can set $F(t)$ instantaneously

So, our goal is to choose $F(t)$ in a way that makes $q(t)$ become equal to $q_d(t)$. That is, we want

$$q_d(t) \to q(t).$$

Model Inversion

One approach: Ensure that $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$ and make $F(t) = \ddot{q}_d(t)$ (model inversion).

Model Inversion

One approach: Ensure that $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$ and make $F(t) = \ddot{q}_d(t)$ (model inversion).

Then,

$$\ddot{q}(t) = F(t) = \ddot{q}_d(t) = \frac{d}{dt}\frac{d}{dt}q_d(t),$$

so that upon integration, $q_d(t) \equiv q(t)$.

Model Inversion

One approach: Ensure that $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$ and make $F(t) = \ddot{q}_d(t)$ (model inversion).

Then,

$$\ddot{q}(t) = F(t) = \ddot{q}_d(t) = \frac{d}{dt}\frac{d}{dt}q_d(t),$$

so that upon integration, $q_d(t) \equiv q(t)$. Visually,

$$q_d(t) \longrightarrow \boxed{\frac{d}{dt}} \longrightarrow \boxed{\frac{d}{dt}} \xrightarrow{\ddot{q}_d(t) = F(t)} \boxed{\int} \xrightarrow{v(t)} \boxed{\int} \xrightarrow{q(t)}$$

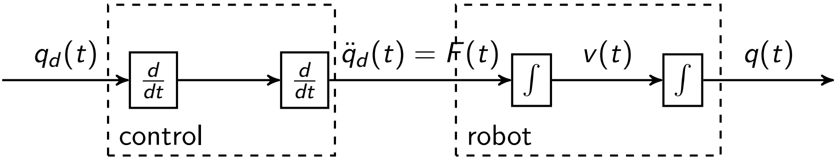
Model Inversion

One approach: Ensure that $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$ and make $F(t) = \ddot{q}_d(t)$ (model inversion).

Then,

$$\ddot{q}(t) = F(t) = \ddot{q}_d(t) = \frac{d}{dt}\frac{d}{dt}q_d(t),$$

so that upon integration, $q_d(t) \equiv q(t)$. Visually,



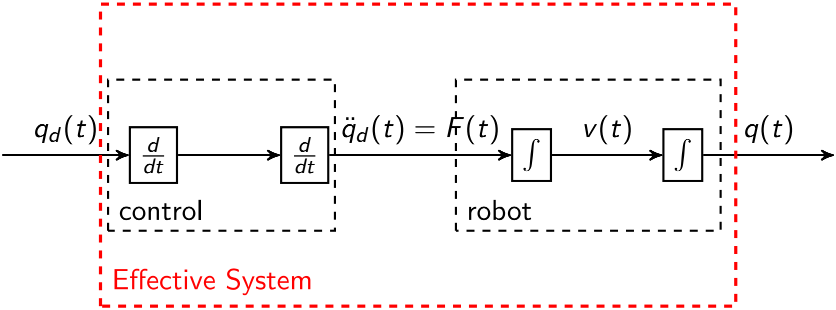
Model Inversion

One approach: Ensure that $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$ and make $F(t) = \ddot{q}_d(t)$ (model inversion).

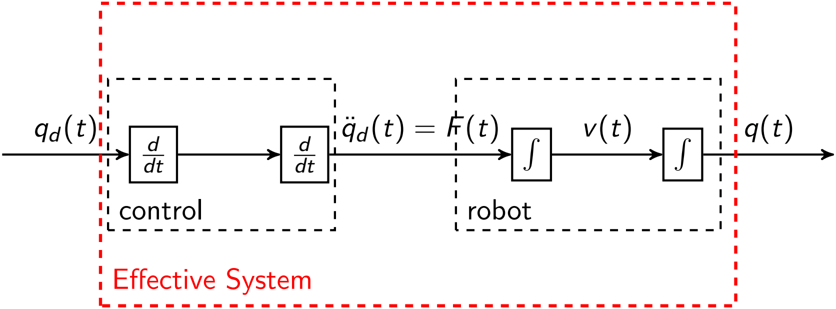
Then,

$$\ddot{q}(t) = F(t) = \ddot{q}_d(t) = \frac{d}{dt}\frac{d}{dt} q_d(t),$$

so that upon integration, $q_d(t) \equiv q(t)$. Visually,



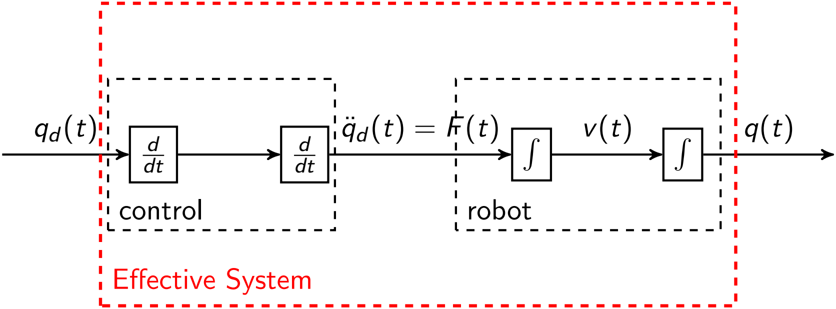
Model Inversion: Open-Loop



We've managed to make our robot react to desired position instantly, by

- Setting the initial condition
- Knowing the 'input' $q_d(t)$ perfectly

Model Inversion: Open-Loop



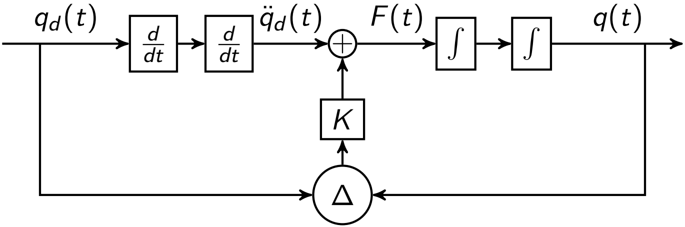
Issues:

- What if desired position is not known ahead of time?
- What if a disturbance force $f_d(t)$ acts, so that input is $F(t) + f_d(t) \neq \ddot{q}_d(t)$?

Closed-Loop Control

One way to account for these issues is to also react to errors in position:

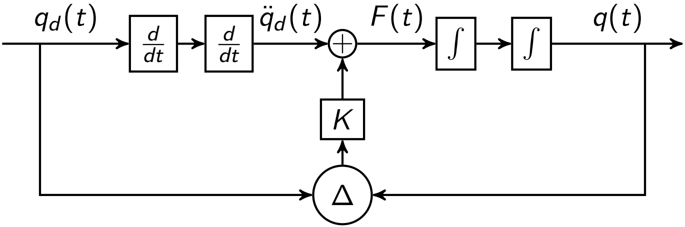
$$F(t) = \ddot{q}_d(t) + K\left(q_d(t) - q(t)\right)$$



Closed-Loop Control

One way to account for these issues is to also react to errors in position:

$$F(t) = \ddot{q}_d(t) + K\left(q_d(t) - q(t)\right)$$

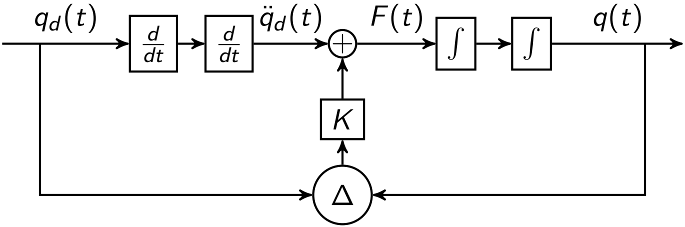


Issue: How do we choose K? Why will a good choice be possible?

Closed-Loop Control

One way to account for these issues is to also react to errors in position:

$$F(t) = \ddot{q}_d(t) + K\left(q_d(t) - q(t)\right)$$



Issue: How do we choose K? Why will a good choice be possible?
Another issue: How do you know what $q(t)$ is?

Recap

We studied a simple point-mass system.

Recap

We studied a simple point-mass system.

Configuration is $q(t)$, input is a force $F(t)$

Recap

We studied a simple point-mass system.

Configuration is $q(t)$, input is a force $F(t)$

We then related force to change in $q(t)$ using an ordinary differential equation:

$$\frac{d^2}{dt^2}q(t) = F(t).$$

This ODE is a forward dynamics model: how the state and input affect the change in state.

State

State

The state is a quantity that allows prediction of the change in state given the inputs.

The state summarizes the history of a system, since the current state and (future) input dictates the future state.

State

State

The state is a quantity that allows prediction of the change in state given the inputs.

The state summarizes the history of a system, since the current state and (future) input dictates the future state.

State is typically denoted by x, and state at time t is $x(t)$.

State

State

The state is a quantity that allows prediction of the change in state given the inputs.

The state summarizes the history of a system, since the current state and (future) input dictates the future state.

State is typically denoted by x, and state at time t is $x(t)$.

For physical robots, state consists of configuration and velocity, or

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

State

State

The state is a quantity that allows prediction of the change in state given the inputs.

The state summarizes the history of a system, since the current state and (future) input dictates the future state.

State is typically denoted by x, and state at time t is $x(t)$.

For physical robots, state consists of configuration and velocity, or

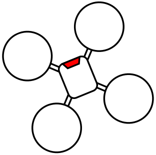
$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

For point-mass:

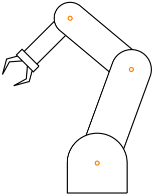
$$\dot{x} = \begin{bmatrix} 0 \\ F(t) \end{bmatrix} = f(x(t), F(t)).$$

Dynamics

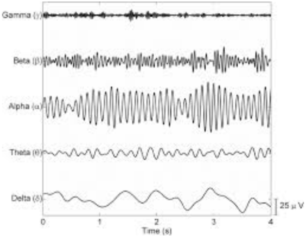
- The system has a state ***x***
- The dynamics are captured by how ***x*** changes with time



Quadrotor

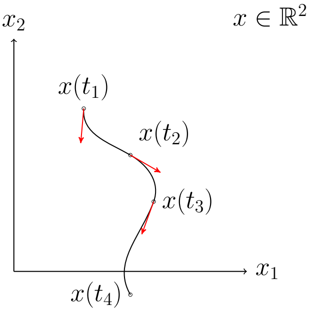


Robot Arm

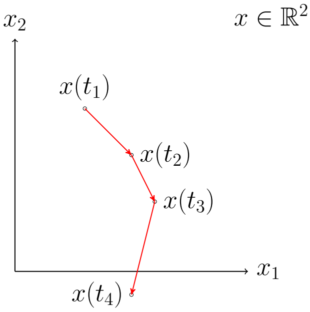


EEG

Dynamics



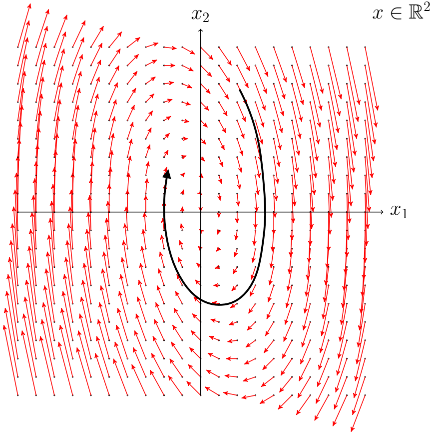
Continuous-time

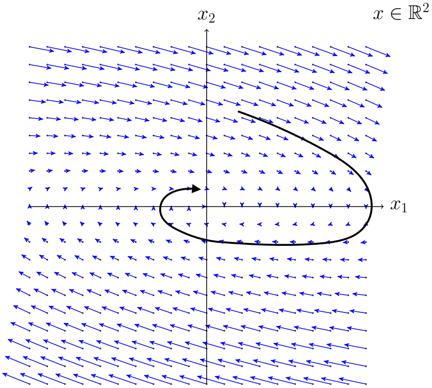


Discrete-time

Think of vectors assigned to each point *x*
(A vector field)

Dynamics





Two continuous vector fields

Forward Dynamics

Continuous Time:

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

where the independent variable is time $t \in \mathbb{R}$
Discrete Time:

$$x_{t+1} = f_t(x_t, u_t),$$

where the independent variable is $t \in \mathbb{N}$

Inverse Dynamics

Idea: given some desired value $\dot{x}_d(t)$ for $\dot{x}(t)$, what $u(t)$ will achieve it?

Inverse Dynamics

Idea: given some desired value $\dot{x}_d(t)$ for $\dot{x}(t)$, what $u(t)$ will achieve it?

For example, for the point mass, we said that we want $\ddot{q}(t)$ to equal $\ddot{q}_d(t)$, so we choose

$$F(t) = \ddot{q}_d(t)$$

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Many control algorithms compute a control input $u(t)$ given our state $x(t)$.

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Many control algorithms compute a control input $u(t)$ given our state $x(t)$.

Where does $x(t)$ come from?

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Many control algorithms compute a control input $u(t)$ given our state $x(t)$.

Where does $x(t)$ come from?

We assume we have a sensor map $h \colon X \to Y$ where X is our state space and Y is our space of measurements.

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Many control algorithms compute a control input $u(t)$ given our state $x(t)$.

Where does $x(t)$ come from?

We assume we have a sensor map $h\colon X \to Y$ where X is our state space and Y is our space of measurements.

Challenge

How do we extract $x(t)$ from our measurements?

Sensor Model

Consider a dynamical system with state $x \in \mathbb{R}^n$.

Many control algorithms compute a control input $u(t)$ given our state $x(t)$.

Where does $x(t)$ come from?

We assume we have a sensor map $h \colon X \to Y$ where X is our state space and Y is our space of measurements.

Challenge

How do we extract $x(t)$ from our measurements?