Feedforward and Feedback Control

Spring 2020

Hasan Poonawala

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

Robots are dynamical systems with inputs that are forces and/or torques.

The position is – roughly speaking – the double integral of the history of applied forces.

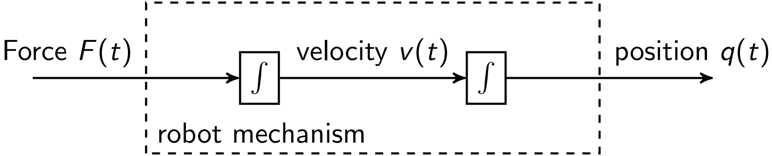
A Point-Mass Robot

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

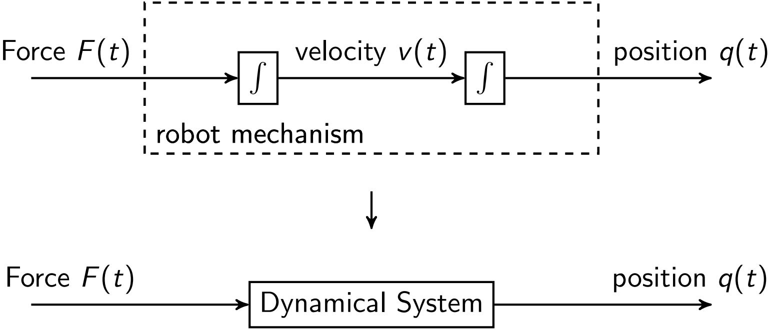
Assume we can apply any force $F \in \mathbb{R}$ at time $t \in R$.

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



Control

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

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

We must choose $F(t)$ so that the resulting solution $q(t)$ has the property that $q(t)$ gets closer to and maybe equal to $q_d(t)$ as time proceeds.

Mathematically, we want

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

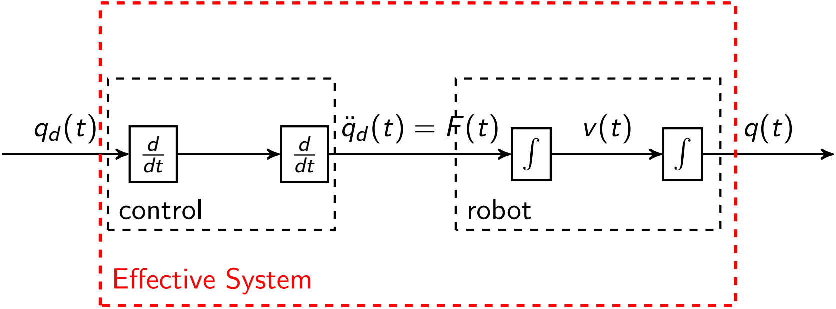
Model Inversion

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).

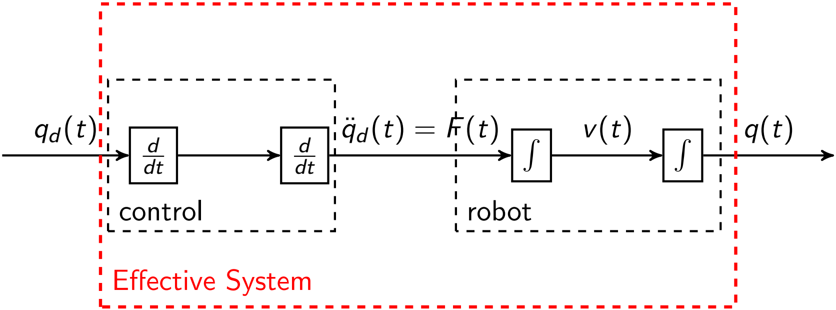
Then,

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

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



Model Inversion: Feed-forward Open-Loop

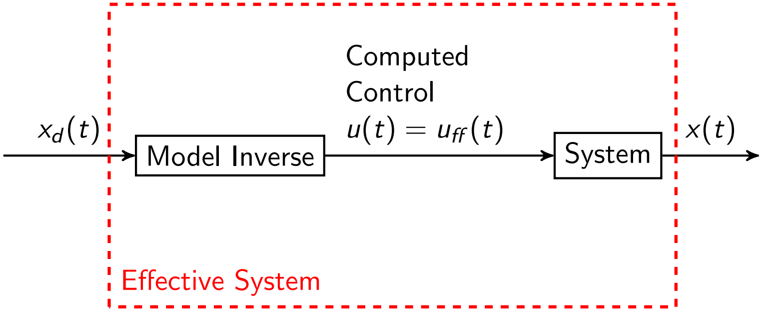


We've managed to make our robot react to desired position instantly, by

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

Model Inversion: Feed-forward Open-Loop

In general,

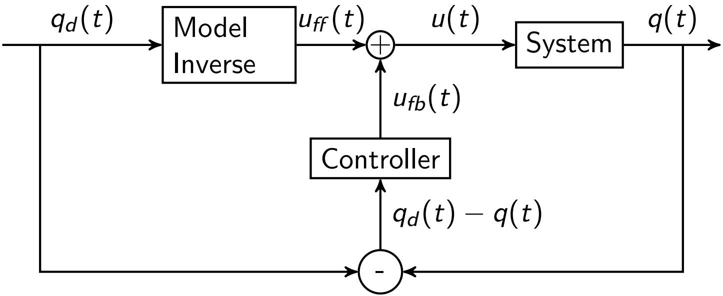


Issues:

- What if desired position $x_d(t)$ is not known ahead of time?
- What if a disturbance force $w(t)$ acts, so that input is $u(t) + w(t)$?

Closed-Loop Control

If the model inversion is not perfect, $q_d(t) \neq q(t)$.
We add a correction to account for this difference, which is called feedback.



A good choice for the controller helps overcome modeling errors and disturbances

However, a bad choice can damage the system

Summary

We've seen that a model of the system we are trying to control can help us design a feedforward control

To account for imperfections, a feedback controller looks at the error between our goal and reality, and focuses on correcting this error

Coming Up

When the system is a robotic manipulator, what types of feedforward and feedback terms should we use? How do we design them for a specific robot?

We'll consider two cases:

1. Each joint is viewed independently, and servo control approaches are used to track a joint trajectory $q_d(t)$. Analysis is based on linear control theory.
2. We use the coupled multi-link model to design a controller. Lyapunov theory-based analysis dominates here.