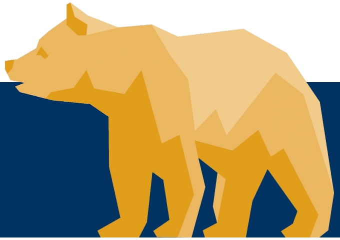
EECS 106B/206B
Robotic Manipulation and Interaction

Valmik Prabhu



Announcements

- First paper presentations in lab this week. “First-pass” the following:
  - Feedback Linearization for Unknown Systems via Reinforcement Learning, Westenbroek, Fridovich-Keil, Mazumdar, et al
- Project 1 Part B has been released
  - An introduction to it will be presented in lab section this week to help you get started.
- Homework 2 released
  - It’s much easier

Goals of this lecture

Introduce a number of practical tools that we can use on real hardware

- Cascaded Control
- PID Control
- Gain Scheduling

PRACTICAL CONTROL THEORY

Content draws from many sources, including ME 132 (Sp 16) and ME 136 (Fa 17)

What’s the goal of control?

- Tracking
- Stability
- Robustness to Noise and Disturbances
- Speed

Tuning a Linear Controller

$$\dot{x} = Ax + Bu \qquad u = -Kx$$

- Pole Placement (pick the eigenvalues)
- LQR
- ** Randomly picking things until it works **

Controller Tuning: Trade-off between Disturbance Rejection and Noise

$$\begin{bmatrix} Y \\ U \end{bmatrix} = \begin{bmatrix} \frac{PC}{1+PC} & \frac{P}{1+PC} & \frac{-PC}{1+PC} \\ \frac{C}{1+PC} & \frac{-PC}{1+PC} & \frac{-C}{1+PC} \end{bmatrix} \begin{bmatrix} R \\ D \\ N \end{bmatrix}$$

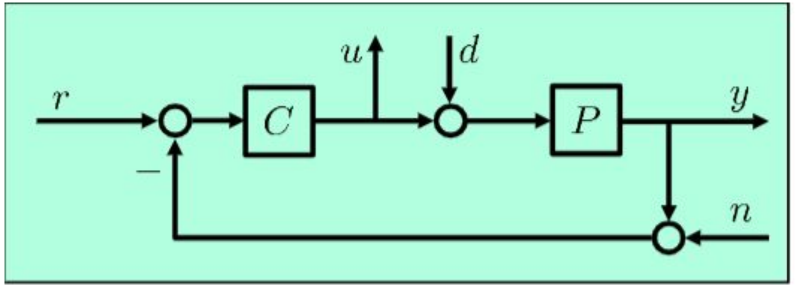
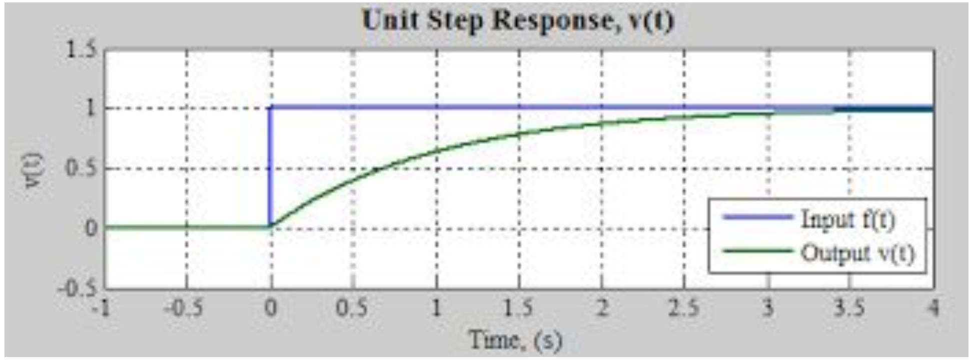


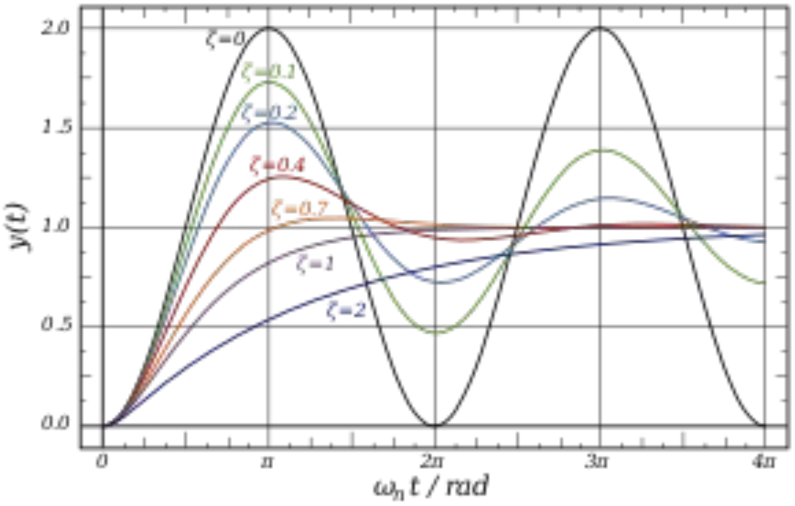
Figure from: Andy Packards ME 231A slides, Fall 2018

Two Common Linear Systems

First order: $\tau \dot{x} + x = k_{dc} u$

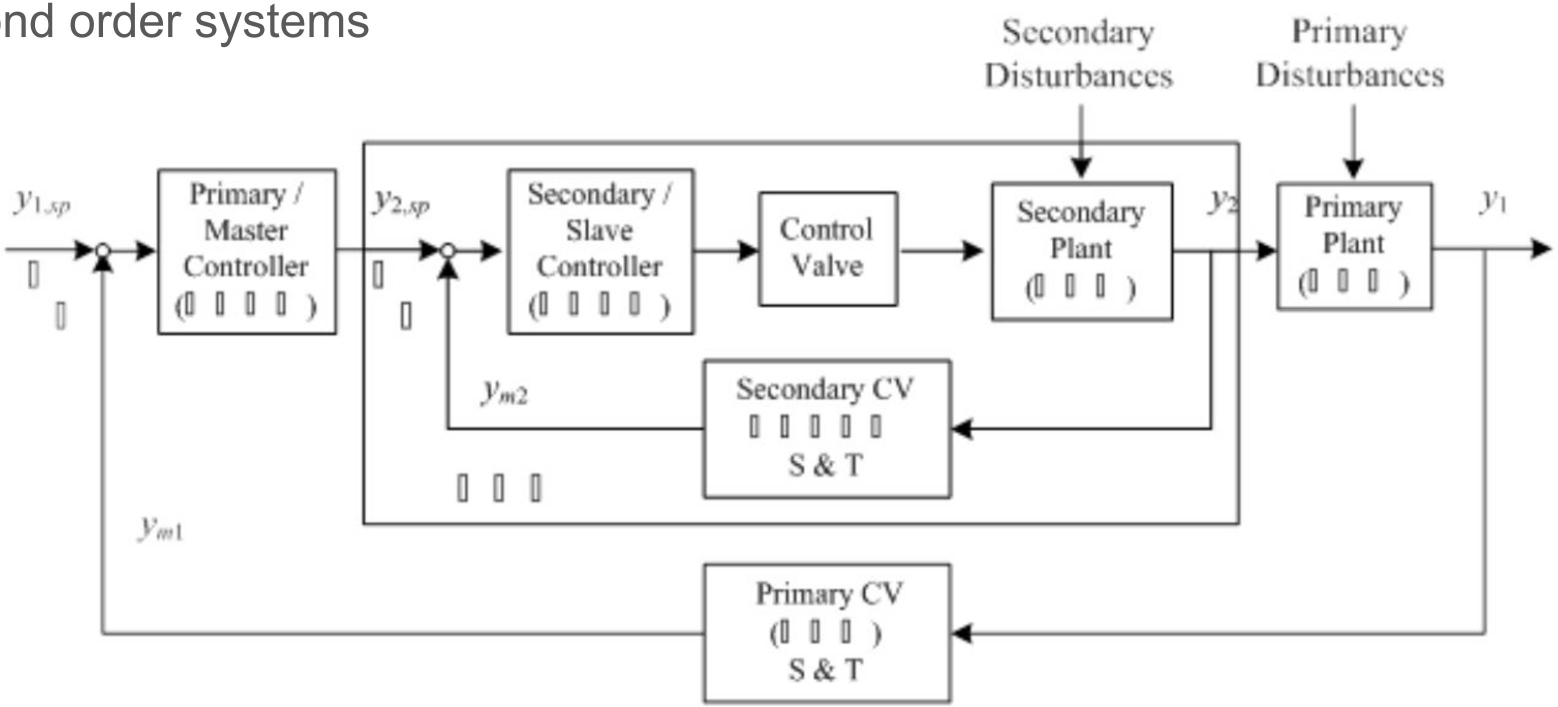
Second Order: $\ddot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = \omega_n^2 k_{dc} u$





Cascaded Control

Take a system with a lot of states and break it up into a “cascade” of first and second order systems



Cascaded Control Requirements

Each stage needs to be “faster” than the previous one

- For first order systems, this is just the time constant
- For second order systems, this is the natural frequency

Cascaded Control Example: Quadcopter

Desired: Position Control

- Position Controller
- Velocity Controller
- Acceleration Controller
- Attitude Controller
- Angular Velocity Controller
- Angular Acceleration Controller
- Motor Controller

Each is its own controller and its own set of gains

Cascaded Control Example: Velocity Control on Baxter

Baxters actual motors take a current input

- You design a controller that uses velocity to control position
- An internal controller uses torque to control velocity
- Each motor has a motor controller which uses current to control torque

I expect that most of you will have better performance on your velocity controllers than your torque controllers. Why? When would you expect your torque controllers to perform better?

PID: The most intuitive control method

- Proportional
- Derivative
- Integral

$$u = K_p e + K_d \dot{e} + K_I \int_0^t e dt$$

Nonlinear Control Techniques we have Covered

- Linearization
- Control Lyapunov Functions
- Feedback Linearization
- MPC*

None of this works on hardware

- Every nonlinear control technique assumes perfect model knowledge, which we never have

Augment Nonlinear Control with PID

The way to make nonlinear control work is to take the output of your nonlinear control, and add some PID to it.

$$u = u_{NL} + u_{PID}$$

The additional error due to model mismatch will be an (unmodelable) nonlinear system, but we know that any nonlinear system can be approximated by a linear system close to an equilibrium point. Thus you can use a hand-tuned PID controller to bring that error to zero.

Augmented Torque Control Law

This is the control law you'll probably be using in your project

$$\tau = M\ddot{q}_d + C\dot{q} + G - K_p e - K_d \dot{e}$$

Gain Scheduling

Have different gains in different parts of your system

- Be very careful at the boundaries. Interpolating control laws at the boundaries can be quite difficult.

PID Control

- Proportional
  - Work horse term
  - Causes oscillation
- Derivative
  - Stabilizing influence
  - Decrease overshoot and ringing, but slows response
  - Problem: sensor noise
- Integral
  - Eliminates constant disturbances, but decreases stability
  - Problem: integrator windup

The Proportional Term

A virtual spring, pulling the system state towards the desired state

$$\ddot{e} = -K_p e$$

Increasing Kp will:

- Decrease rise time (get you to the goal faster)
- Increase system oscillation

The Derivative Term

A virtual damper, stopping the system from changing too quickly

$$\ddot{e} = -K_d \dot{e}$$

Increasing the damping term will

- Stabilize your system response
- Decrease Overshoot
- Increase rise time (but likely decrease settling time)
- Destabilize your system???

The Derivative Term and Noise

Most of the time, you can't actually sense the derivative of your system state, and instead have to use a finite difference model

$$\dot{e}(t) \approx \frac{e(t) - e(t - \delta t)}{\delta t}$$

Unfortunately, this is also a high pass filter, which will amplify any sensor noise. To get around this issue, you usually have to put a low-pass filter over your derivative measurements, but this decreases the effectiveness of your derivative term.

The Integral Term

The integral term is a bit more complex. Essentially, you're controlling the derivative of your control input.

$$\ddot{e} = K_I \int_0^t e dt \implies \dddot{e} = K_i e$$

Increasing the Integral term will:

- Reject constant disturbances
- Destabilize your system

The Integral Term and Windup

If your control input saturates, ie hits a limit, you'll get windup, which can easily destabilize your system.

Fixes:

- Finite horizon integrator
- Weighted horizon integrator
- Bounds on the integral term
- Turn off the integrator when error is high

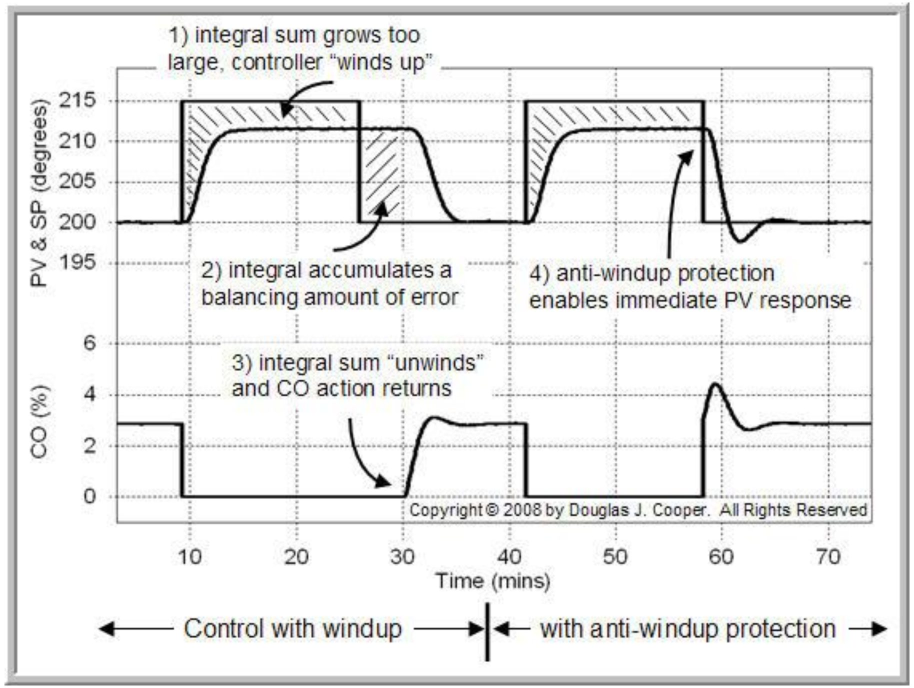
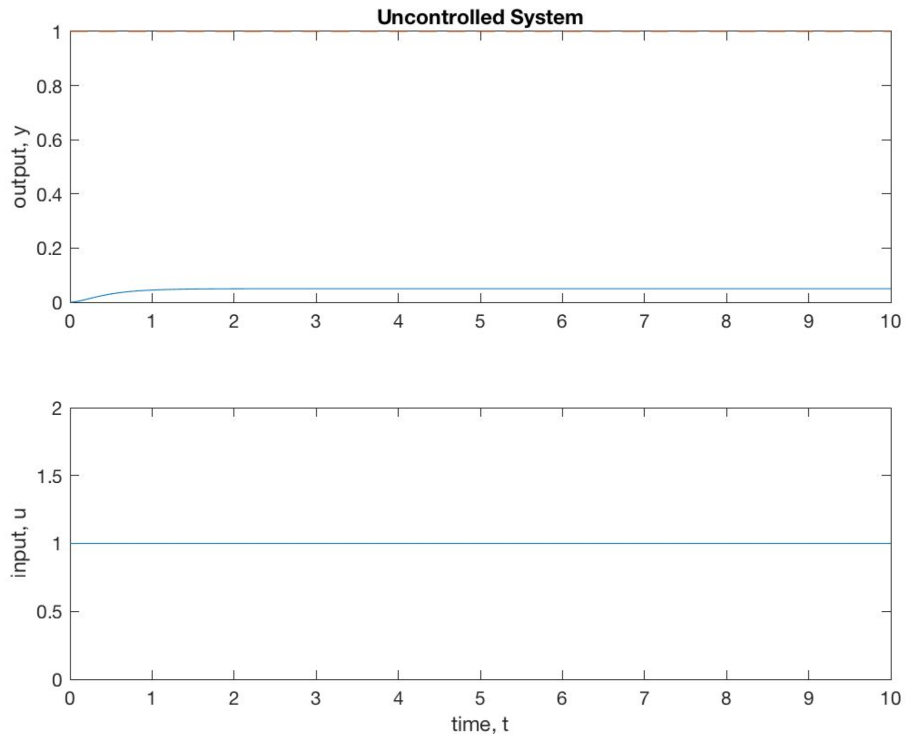


Figure from: https://controlguru.com/integral-reset-windup-jacketing-logic-and-the-velocity-pi-form/

Uncontrolled System

What order system is this?

- Can’t tell (looks like first)
- It is second order
- Overdamped



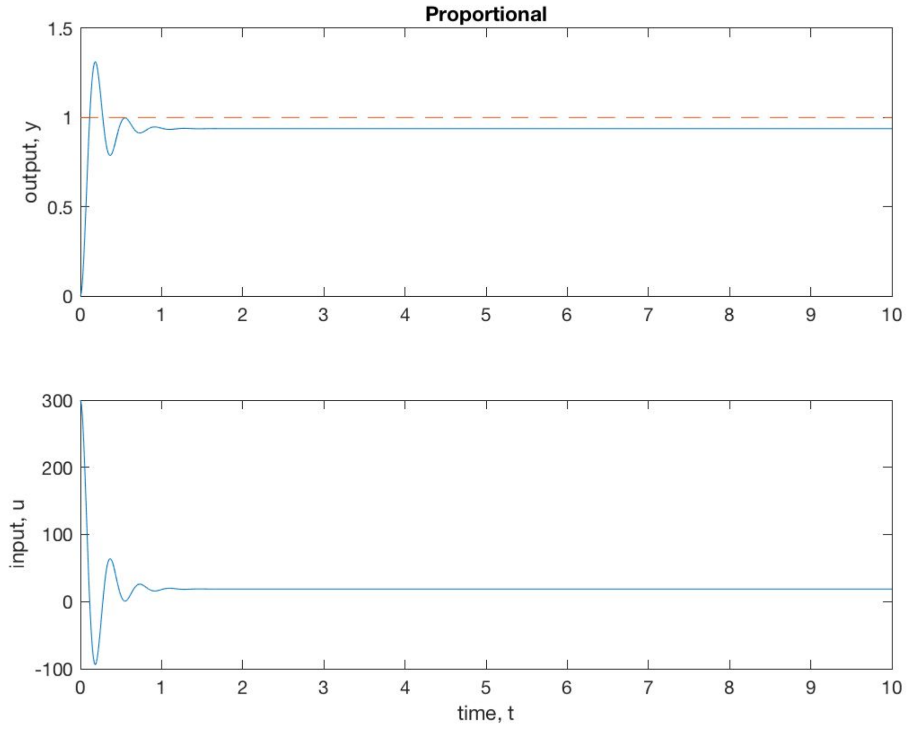
Proportional

What's the percent overshoot?

- Around 30%

Why does the system settle?

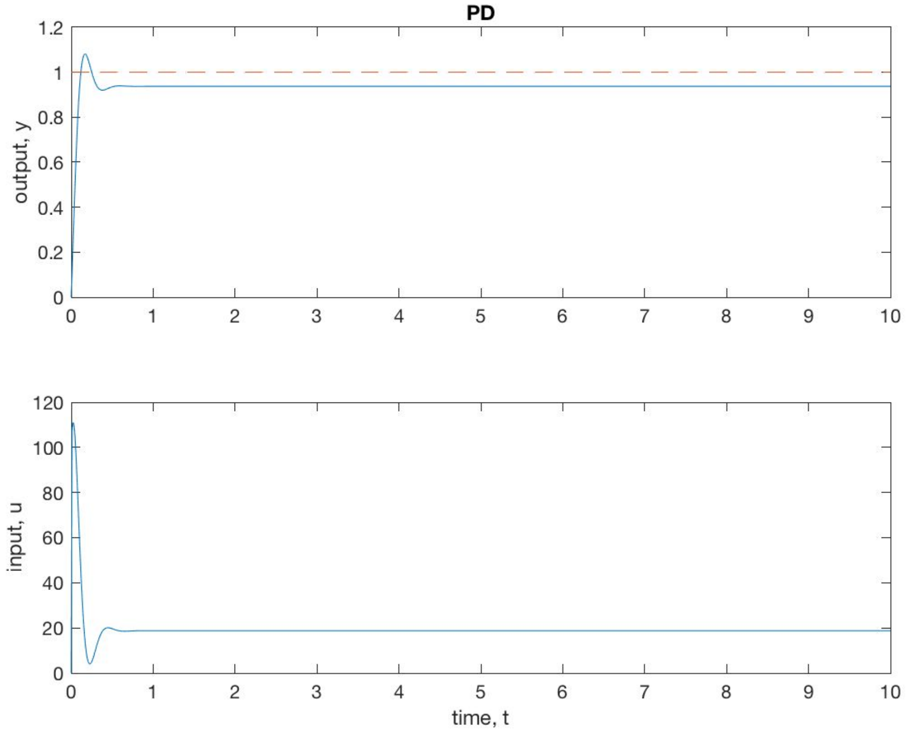
- The system has inherent damping



PD

What is the derivative doing?

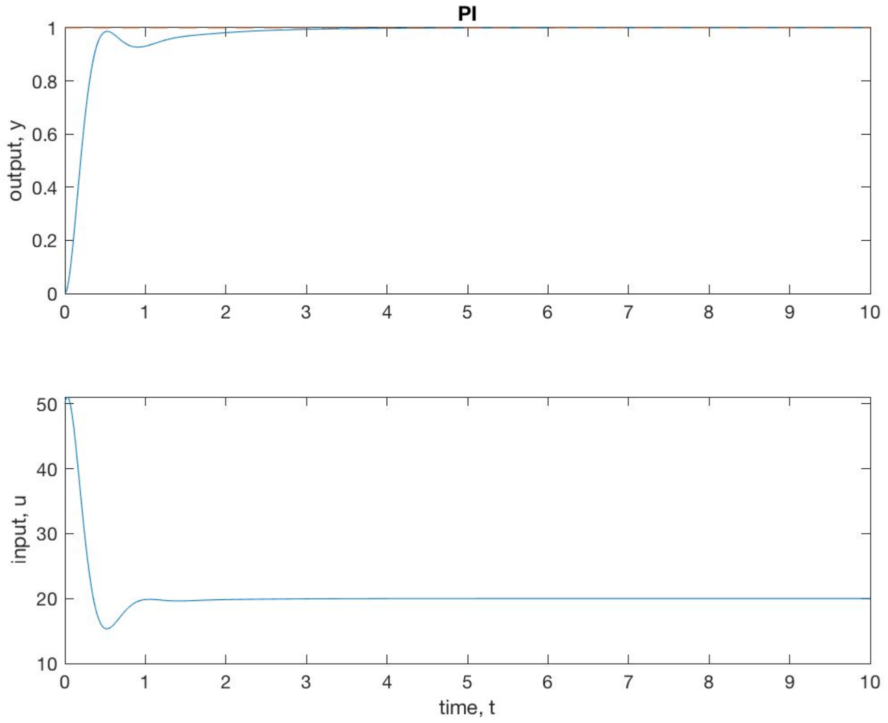
- Reduces overshoot
- Reduces overall energy



PI

What is the integrator doing?

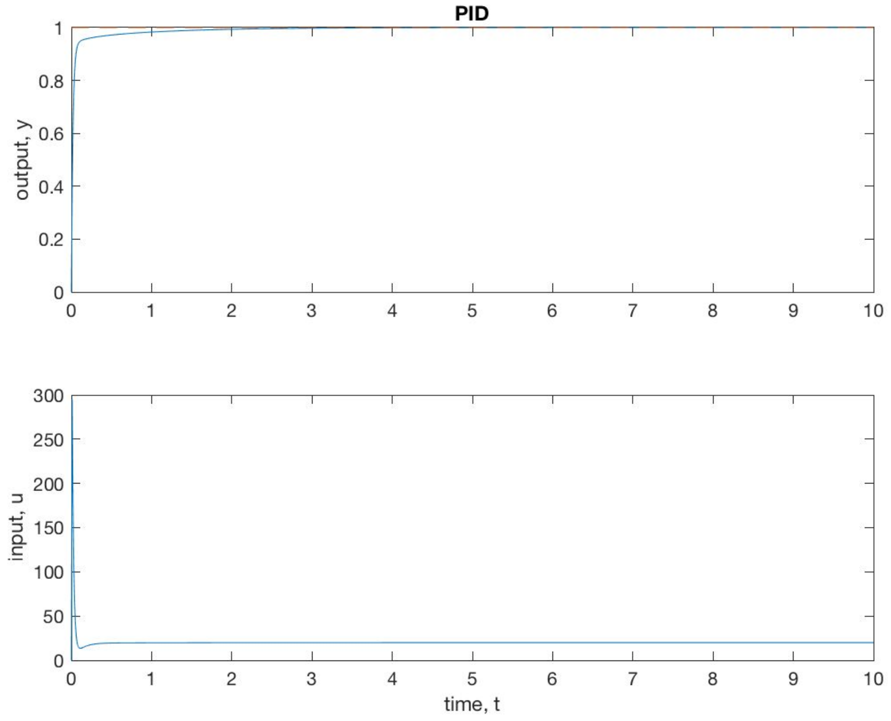
- Reducing static error
- Integrator delay



PID

Benefits of PID (here):

- Fast rise time
- Low settling time
- No overshoot



Feed-forward control

Feed-forward or “ballistic term” does not depend upon error

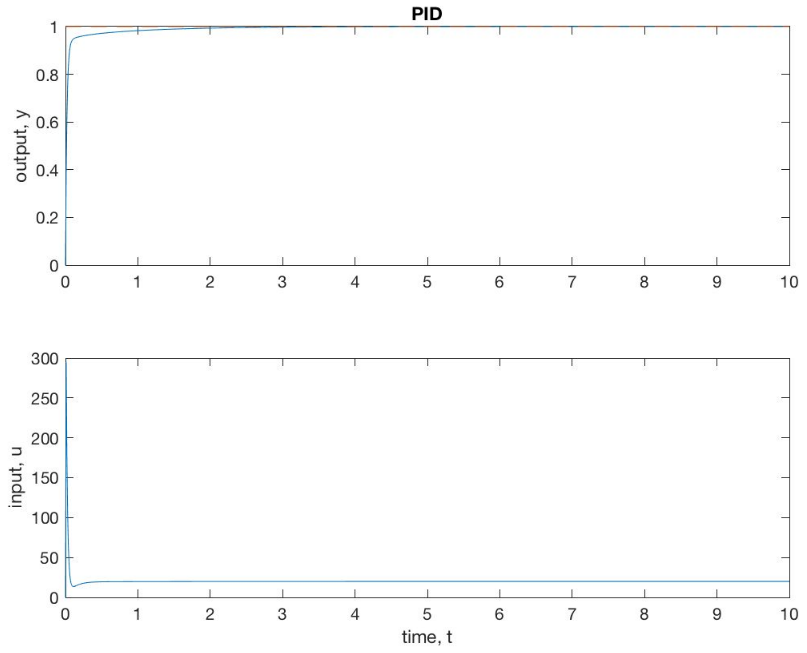
$$u = K_{ff} + K_{fb}e$$

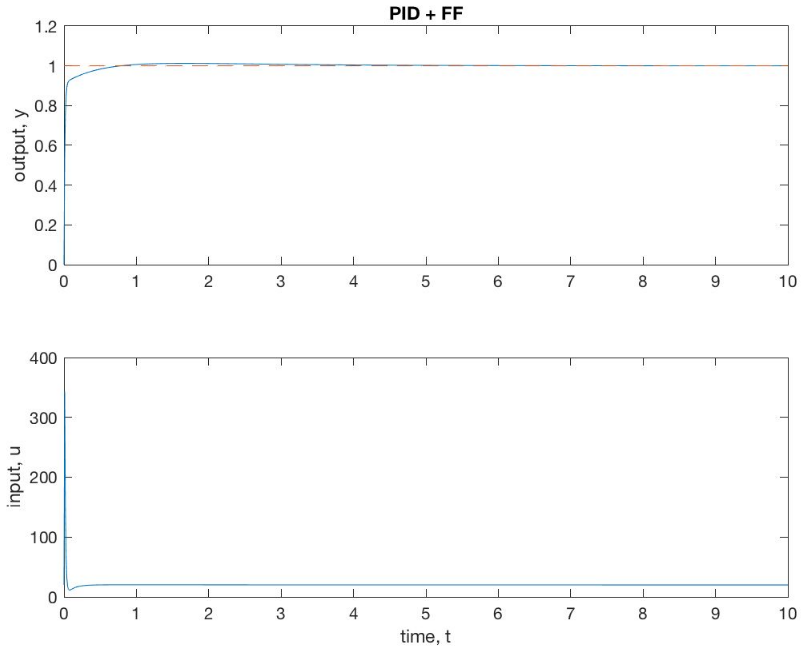
Feed-forward term should do a bulk of your work, with the feedback doing small corrections.

Feed-forward control doesn’t oscillate, and has no stability problems, so it improves system response without decreasing stability.

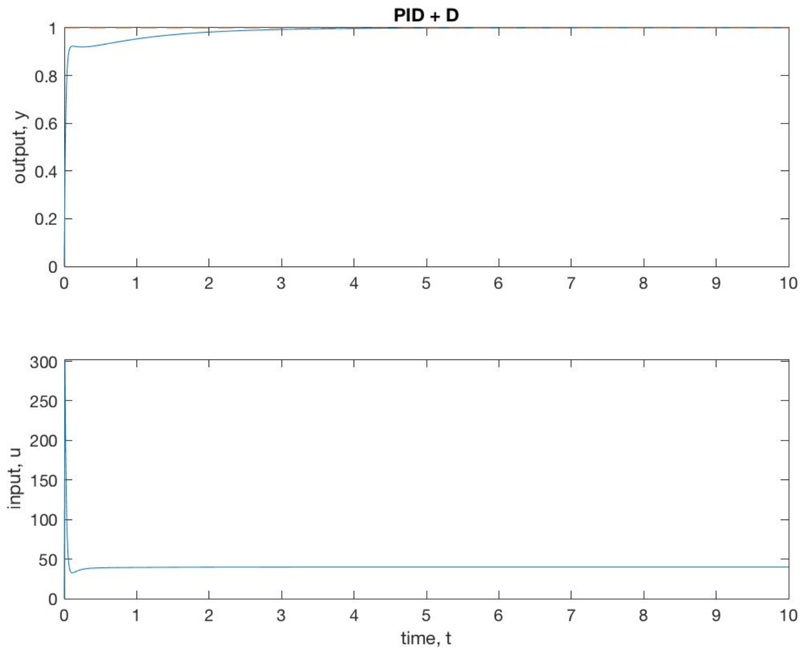
This feedforward gain should include your model-based/nonlinear controller

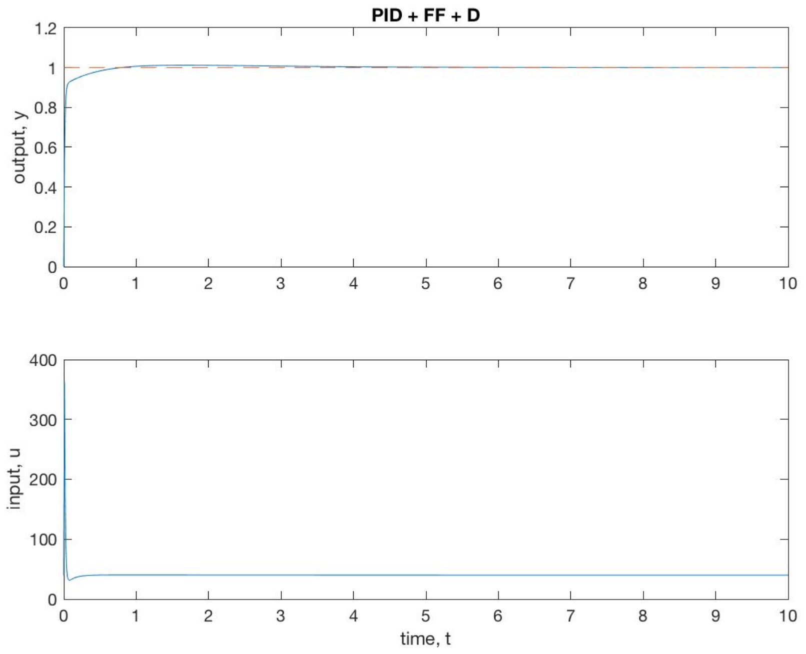
Feed-forward Control Example





Feed-forward can Counteract Known Disturbance





Sources

MLS 4.5
EE 222 Notes by Frank Chiu, Valmik Prabhu, David Fridovich-Keil and Sarah Fridovich-Keil. Course taught by Koushil Sreenath
Nonlinear Systems: Analysis, Stability, and Control Ch 9, Sastry
ME 230A. Course taught by Francesco Borrelli