CS 287: Advanced Robotics
Fall 2009

Lecture 2: Control 1: Feedforward, feedback, PID, Lyapunov direct method

Pieter Abbeel
UC Berkeley EECS

Announcements

- Office hours: Thursdays 2-3pm + by email arrangement, 746 SDH
  - SDH 7th floor should be unlocked during office hours on Thursdays
- Questions about last lecture?

CS 287 Advanced Robotics

- **Control**
- Estimation
- Manipulation/Grasping
- Reinforcement Learning
- Misc. Topics
- Case Studies

Control in CS287

- Overarching goal:
  - Understand what makes control problems hard
  - What techniques do we have available to tackle the hard (and the easy) problems
- Any applicability of control outside robotics? Yes, many!
  - Process industry, feedback in nature, networks and computing systems, economics, …
    - [See, e.g., Chapter 1 of Astrom and Murray, http://www.cds.caltech.edu/~murray/amwiki/Main_Page, for more details---_optional_ reading. Fwiw: Astrom and Murray is a great read on mostly classical feedback control and is freely available at above link.]
  - We will not have time to study these application areas within CS287 [except for perhaps in your final project!]

Today's lecture

- Feedforward vs. feedback
- PID (Proportional Integral Derivative)
- Lyapunov direct method --- a method that can be helpful in proving guarantees about controllers
- Reading materials:
  - Astrom and Murray, 10.3
  - Tedrake, 1.2
  - Optional: Slotine and Li, Example 3.21.

Based on a survey of over eleven thousand controllers in the refining, chemicals and pulp and paper industries, 97% of regulatory controllers utilize PID feedback. L. Desborough and R. Miller, 2002 [DM02]. [Quote from Astrom and Murray, 2009]

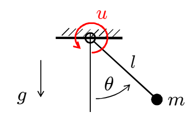
Today's lecture

- Practical result: can build a trajectory controller for a fully actuated robot arm

- Our abstraction: torque control input to motor, read out angle [in practice: voltages and encoder values]

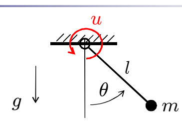
Intermezzo: Unconventional (?) robot arm use

Single link manipulator (aka the simple pendulum)



$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t) \qquad I = ml^2$$

Single link manipulator



$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

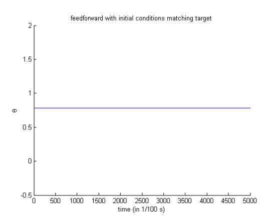
How to hold arm at θ = 45 degrees?

The Matlab code that generated all discussed simulations will be posted on www.

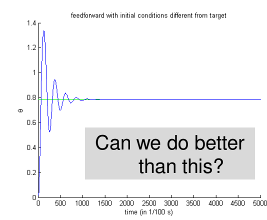
Single link manipulator

Simulation results:

$$I\ddot{\theta}(t) + c\dot{\theta}(t) + mgl\sin\theta(t) = u(t), \quad u = mgl\sin\tfrac{\pi}{4}$$



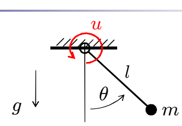
$\theta(0) = \frac{\pi}{4}, \dot{\theta}(0) = 0$



Can we do better than this?

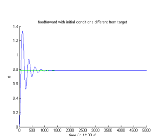
$\theta(0) = 0, \dot{\theta}(0) = 0$

Feedforward control



$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

How to make arm follow a trajectory $\theta^*(t)$?



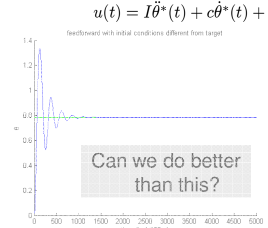
$\theta(0) = 0, \dot{\theta}(0) = 0$

$$u(t) = I\ddot{\theta}^*(t) + c\dot{\theta}^*(t) + mgl\sin\theta^*(t)$$

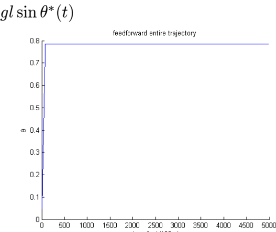
Feedforward control

Simulation results:

$$I\ddot{\theta}(t) + c\dot{\theta}(t) + mgl\sin\theta(t) = u(t) \qquad \theta(0) = 0, \dot{\theta}(0) = 0$$
$$u(t) = I\ddot{\theta}^*(t) + c\dot{\theta}^*(t) + mgl\sin\theta^*(t)$$

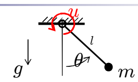


Can we do better than this?



n DOF (degrees of freedom) manipulator?

- Thus far:



$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

- n DOF manipulator: standard manipulator equations

$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$

  - H : "inertial matrix," full rank
  - B : identity matrix if every joint is actuated
- → Given trajectory $q(t)$, can readily solve for feedforward controls $u(t)$ for all times t

Fully-Actuated vs. Underactuated

- A system is fully actuated when in a certain state $(q,\text{\dot{q}},t)$ if, when in that state, it can be controlled to instantaneously accelerate in any direction.
- Many systems of interest are of the form:

$$\ddot{q} = f_1(q,\dot{q},t) + f_2(q,\dot{q},t)u \qquad (1)$$

- Defn. **Fully actuated**: A control system described by Eqn. (1) is fully-actuated in state (q,\dot{q},t) if it is able to command an instantaneous acceleration in an arbitrary direction in q:

$$\text{rank} f_2(q,\dot{q},t) = \dim q$$

- Defn. **Underactuated**: A control system described by Eqn. (1) is underactuated in configuration (q,\dot{q},t) if it is not able to command an instantaneous acceleration in an arbitrary direction in q:

$$\text{rank} f_2(q,\dot{q},t) < \dim q$$

- [See also, Tedrake, Section 1.2.]

Fully-Actuated vs. Underactuated

$\ddot{q} = f_1(q,\dot{q},t) + f_2(q,\dot{q},t)u$ fully actuated in $(q,\dot{q},t)$ iff $\text{rank} f_2(q,\dot{q},t) = \dim q$.

- Hence, for any fully actuated system, we can follow a trajectory by simply solving for $u(t)$:

$$u(t) = f_2^{-1}(q,\dot{q},t)\,(\ddot{q} - f_1(q,\dot{q},t))$$

- [We can also transform it into a linear system through a change of variables from u to v:

$$\ddot{q}(t) = v(t)$$
$$u(t) = f_2^{-1}(q,\dot{q},t)\,(v(t) - f_1(q,\dot{q},t))$$

The literature on control for linear systems is very extensive and hence this can be useful. This is an example of feedback linearization. More on this in future lectures.]

Fully-Actuated vs. Underactuated

$\ddot{q} = f_1(q,\dot{q},t) + f_2(q,\dot{q},t)u$ fully actuated in $(q,\dot{q},t)$ iff $\text{rank} f_2(q,\dot{q},t) = \dim q$.

- **n DOF manipulator**

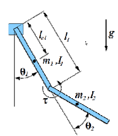
$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$
$$f_2 = H^{-1}B,\ H \text{ full rank},\ B = I, \text{ hence } \text{rank}(H^{-1}B) = \text{rank}(B)$$

  - All joints actuated → rank(B) = n → fully actuated
  - Only $p < n$ joints actuated → rank(B) = p → underactuated

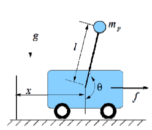
Example underactuated systems

- Car

- Acrobot

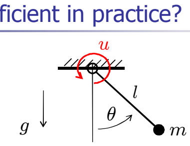


- Cart-pole



- Helicopter

Fully actuated systems: is our feedforward control solution sufficient in practice?

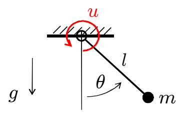


$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

Task: hold arm at 45 degrees.

- What if parameters off? --- by 5%, 10%, 20%, …
- What is the effect of perturbations?

Fully actuated systems: is our feedforward control solution sufficient in practice?

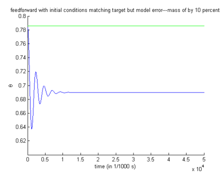


$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

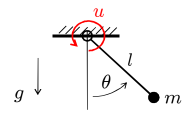
Task: hold arm at 45 degrees.

- Mass off by 10%:

→ steady-state error



Fully actuated systems: is our feedforward control solution sufficient in practice?

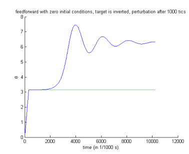


$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

Task: swing arm up to **180** degrees and hold there

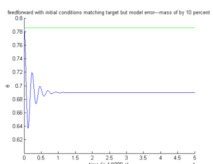
- Perturbation after 1sec:
  - Does **not** recover

[θ = 180 is an “unstable” equilibrium point]

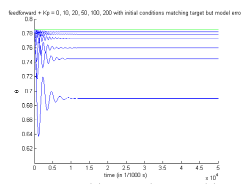


Proportional control

Task: hold arm at 45 degrees



$$u(t) = u_{\text{feedforward}}(t)$$

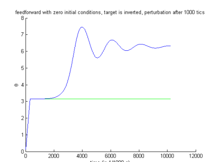


$$u(t) = u_{\text{feedforward}}(t) + K_p(q_{\text{desired}}(t) - q(t))$$

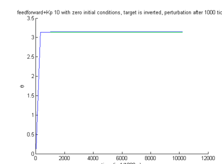
- Feedback can provide
  - Robustness to model errors
- However, still:
  - Overshoot issues --- ignoring momentum/velocity!
  - Steady-state error --- simply crank up the gain?

Proportional control

Task: swing arm up to **180** degrees and hold there



$$u(t) = u_{\text{feedforward}}(t)$$



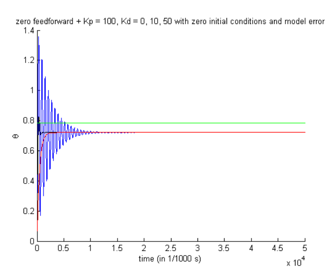
$$u(t) = u_{\text{feedforward}}(t) + K_p(q_{\text{desired}}(t) - q(t))$$

Current status

- Feedback can provide
  - *Robustness to model errors*
  - *Stabilization around states which are unstable in open-loop*
- Overshoot issues --- ignoring momentum/velocity!
- Steady-state error --- simply crank up the gain?

PD control

$$u(t) = K_p(q_{\text{desired}}(t) - q(t)) + K_d(\dot{q}_{\text{desired}} - \dot{q}(t))$$

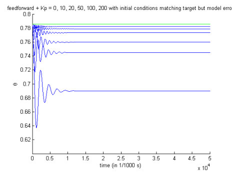


Eliminate steady state error by cranking up Kp ?

Task: hold arm at 45 degrees

$$I\ddot{\theta}(t) + b\dot{\theta}(t) + mgl\sin\theta(t) = u(t)$$

$$u(t) = u_{\text{feedforward}}(t) + K_p(q_{\text{desired}}(t) - q(t))$$



In steady-state, $\ddot{q} = \dot{q} = 0$ and we get:

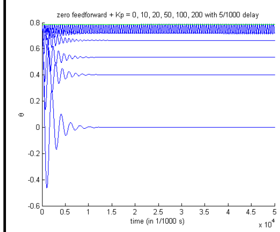
$$mgl\sin\theta = u_{\text{feedforward}} + K_p(\theta^* - \theta)$$

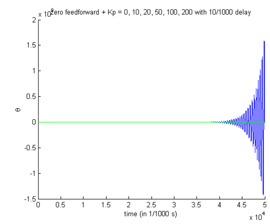
Using some trigoniometry and assuming θ is close to θ^* we get:

$$\theta - \theta^* = \frac{u_{\text{feedforward}} - mgl\sin\theta^*}{K_p + mgl\cos\theta^*}$$

Eliminate steady state error by cranking up Kp ?

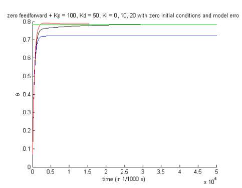
$$u(t + \delta t) = K_p(q_{\text{desired}}(t) - q(t))$$





PID

$$u(t) = K_p(q_{\text{desired}}(t) - q(t)) + K_d(\dot{q}_{\text{desired}} - \dot{q}(t)) + K_i\int_0^t (q_{\text{desired}}(\tau) - q(\tau))d\tau$$



- Zero error in steady-state: [assumes steady-state is achieved!]

$\ddot{q} = \dot{q} = 0$, $\dot{u} = 0$, hence, taking derivatives of above:

$$\begin{aligned} \dot{u} &= K_p(\dot{q}_{\text{desired}} - \dot{q}(t)) + K_d(\ddot{q}_{\text{desired}} - \ddot{q}(t)) + K_i(q_{\text{desired}}(t) - q(t)) \\ 0 &= K_k(q_{\text{desired}}(t) - q(t)) \end{aligned}$$

Recap so far

- Given a ***fully actuated system*** and a (smooth) target trajectory
  - Can solve dynamics equations for required control inputs = "feedforward controls"
  - Feedforward control is insufficient in presence of
    - Model inaccuracy
    - Perturbations + instability
  - Proportional feedback control can alleviate some of the above issues.
    - Steady state error reduced by (roughly) factor K_p, but large K_p can be problematic in presence of delay → Add integral term
    - Ignores momentum → Add derivative term
- **Remaining questions:**
  - How to choose PID constants? Aka "tuning"
  - Any guarantees?

PID tuning

- Typically done by hand (3 numbers to play with) [policy search should be able to automate this in many settings]
- Ziegler-Nichols method (1940s) provides recipe for starting point
  - Frequency response method
  - Step response method
- Recipe results from
  - (a) Extensively hand-tuning controllers for many settings
  - (b) Fitting a function that maps from easy to measure parameters to the three gains

[See also Astrom and Murray Section 10.3]

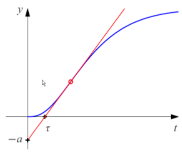
PID tuning: Ziegler-Nichols frequency domain method

- Set derivative and integral gain to zero
- Drive up the proportional gain until steady oscillation occurs, record the corresponding gain k_c and period T_c
- Use the following table to set the three gains:

Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.4k_c$	$0.8T_c$	
PID	$0.6k_c$	$0.5T_c$	$0.125T_c$

Notation: $K_I = \frac{k_p}{T_i}$, $K_D = k_pT_d$

PID tuning: Ziegler-Nichols step response method

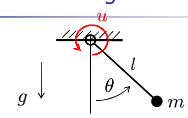


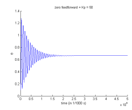
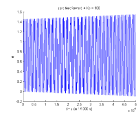
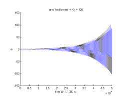
Type	k_p	T_i	T_d
P	1/a		
PI	0.9/a	3τ	
PID	1.2/a	2τ	0.5τ

1. Record open-loop step-response characteristics

2. Read gains out from above table

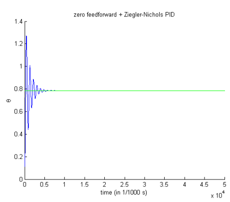
Frequency domain Ziegler-Nichols for single link

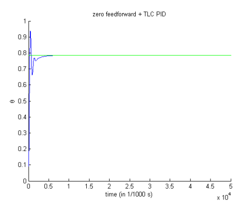


- Kc = 100;
- Tc = 0.63s;

ZN and TLC results





Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.4k_c$	$0.8T_c$	
PID	$0.6k_c$	$0.5T_c$	$0.125T_c$

Tyreus-Luyben tuning chart:
$Kp = k_c/2.2, T_i = 2.2T_c, T_d = T_c/6.3$
Tends to:
increase robustness,
decrease oscillation.

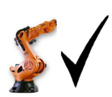
Aside: Integrator wind-up

- Recipe: Stop integrating error when the controls saturate
- Reason: Otherwise it will take a long time to react in the opposite direction in the future.
- Matters in practice!

[See also Astrom and Murray, Section 10.4]

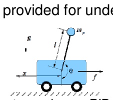
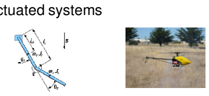
Recap of main points

- To control a fully actuated system:
  - Compute feedforward by solving for $u(t)$
  - However, feedforward is insufficient when:
    - Model is imperfect (i.e., always when dealing with real systems)
    - System is unstable
  - Feedback can address these issues
    - Standard feedback: PID
  - In practice, often even only feedback (i.e., without feedforward) can already provide ok results → in these settings, no model needed, which can be very convenient



- In this lecture no solution provided for underactuated systems

?

- Note: many underactuated systems do use PID type controllers in their core (e.g., helicopter governor, gyro)