ECE 484: Digital Control Applications

Course Notes: Fall 2019

John W. Simpson-Porco

`https://ece.uwaterloo.ca/~jwsimpso/`

Department of Electrical and Computer Engineering
University of Waterloo

Instructor Version

Acknowledgements

These notes are loosely based on material from many sources, including UW ECE 481 lecture notes by Christopher Nielsen and Daniel Miller, JHU ECE 454 notes by Pablo Iglesias, ETH Zürich Regelsysteme II notes by Roy Smith, Stanford EE263 notes by Stephen Boyd, and on parts of several textbooks listed as supplementary reading, most notably Åström, Åström & Wittenmark, and Åström & Murray.

Table of Contents

1. Introduction to digital control

- course mechanics
- topics & outline
- what is digital control?
- why study digital control?
- continuous/discrete/sampled-data systems
- A/D, D/A, and aliasing

Course mechanics

- syllabus is authoritative reference

`learn.uwaterloo.ca`

course requirements:

- laboratory completion (attendance and report submission)
- midterm test (outside of class, scheduling by MME)
- final exam (scheduling by Registrar)

problem sets will be posted weekly (not graded)

Prerequities

- *working* knowledge of linear algebra (*e.g.*, MATH 115)
    - vectors, matrix-vector algebra, linear independence . . .
    - matrices: rank, nullity, invertibility, eigenvalues . . .
    - solvability of linear systems of equations
- signals and systems (*e.g.*, SYDE 252)
    - differential and difference equations
    - Laplace and z-transforms
- analog control systems (*e.g.*, MTE 360)
    - details to follow
- brief mathematics review in appendix of these notes

Course notes

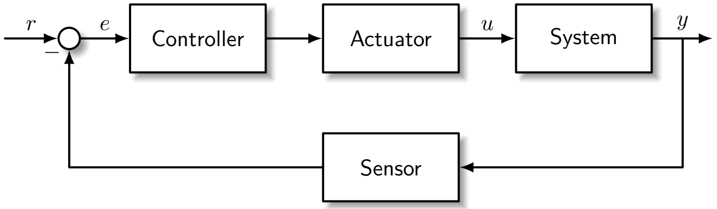
- these notes are
  - designed to be accompanied by lecture
  - full of blank spaces (we will fill these together)
- these notes are not a textbook
  - they give a bare-bones (but complete) presentation
  - supplementary reference chapters listed at the end of each chapter of these notes; these contain additional information and problems
  - blank space at end of each chapter for personal notes / digressions
  - all `MATLAB` code used in notes is on `LEARN`! **Experiment with it!**

Major topics & outline

- models of LTI systems
  - differential and difference equations (old)
  - transfer functions (old)
  - state-space models (new?)
- nonlinear continuous-time state-space models, linearization
- advanced continuous-time control design
- emulation design for sampled-data systems
- direct design for sampled-data systems

What you'll be able to do

- analyze complex models in both continuous and discrete-time
- design stable feedback systems in both continuous and discrete-time
- apply design principles to sampled-data control systems

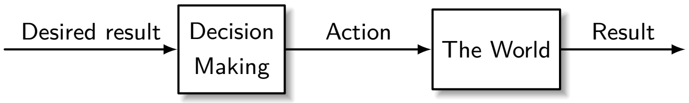


Strategies for success in ECE 484

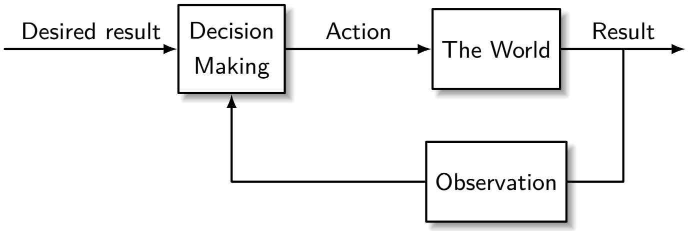
- brush up on signals & systems, linear algebra, analog control
- attend lectures to complete notes
- understand the *ideas*, including key steps in derivations
- in very broad strokes, course emphasizes
  - (i) understanding advantages/disadvantages of digital design methods
  - (ii) developing comfort with state-space models
- midterm and final will reflect this emphasis

Decision making in an uncertain world

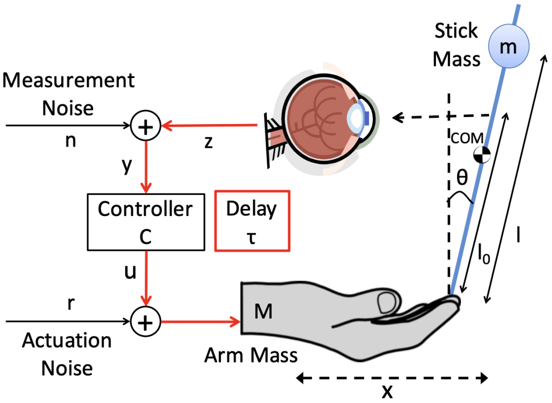
- making decisions with your eyes closed (*feedforward*)



- making decisions with your eyes open (*feedback*)



Can I please have a volunteer . . .



Y. P. Leong and J. C. Doyle, "Understanding Robust Control Theory Via Stick Balancing", in *IEEE CDC*, 2016.

What is feedback?

- **feedback** is a *scientific phenomena* where the output of a system influences the input, which again influences the output, which ...
- the broad field of **control** is concerned with
  - the mathematical study of feedback systems (control theory)
  - the application of feedback to engineering (control engineering)
- **benefits** of feedback
  - improves dynamic response of controlled variables
  - reduces or eliminates effect of disturbances on controlled variables
  - reduces sensitivity to modelling error/uncertainty
  - allows for stabilization of unstable processes

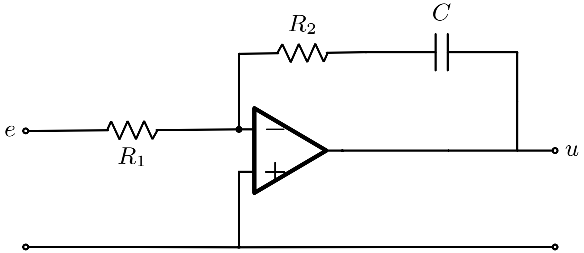
Consequences of feedback instability

Boeing illustration (Courtesy of/Boeing)

- sensor failure produced wrong angle-of-attack measurement
- MCAS control system repeatedly pitched nose downward

History of digital control

- a long time ago, in a galaxy far, far away ...
  - controllers were implemented using analog circuits (30's–60's)



exercise: $$\frac{u(s)}{e(s)} = -\frac{R_2}{R_1}\left(1 + \frac{1}{R_2Cs}\right)$$

Advantages/disadvantages of analog control

- advantages of analog controllers
  - infinite resolution
  - no computational delays
  - allows for *exact* realization of controller transfer function $C(s)$
- disadvantages of analog controllers
  - single purpose circuit
  - maintenance
  - component drift

History of digital control

- theory pioneered in 40's/50's at Columbia (Ragazzini, Franklin, . . .)
- invention of transistor (1947)
- application to paper mills and chemical plants (late 1950's)
- Apollo missions relied heavily on digital control (late 60's)
- personal computers (1980's), massive growth
- relatively cheap ICs and microcontrollers (1990's)
- ubiquitous dirt cheap sensing and computing (present)

Terminology issues

digital control is “feedback control using a computer”, including

- **sensing:** real-time signal acquisition
- **computation:** real-time control law calculation
- **actuation:** real-time control input generation

digital control *is not*:

- sequencing / digital logic (consumer electronics)
- supervisory software

Why study digital control?

all modern control systems are digital control systems

- robotics (humanoid, quad-coptors, teams, swarms . . .)
- intelligent automotive and transportation systems
- renewable energy and smart grid
- smart buildings and cities
- synthetic biology
- aerospace
- medical (*e.g.*, artificial organs, closed-loop anesthesia)
- smart materials (*e.g.*, energy harvesting)
- various evil disciplines like finance, advertising . . .

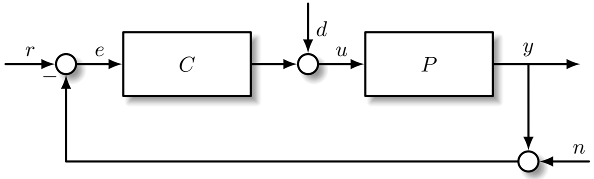
Advantages/disadvantages of digital control

- advantages of digital controllers
  - programmable and **cheap**
  - complex controllers possible
  - single-chip solutions possible
  - no component nonlinearities
  - more reliable
- disadvantages of digital controllers
  - sampling/computational issues
  - more complex to analyze/design

Issues arising in digital control

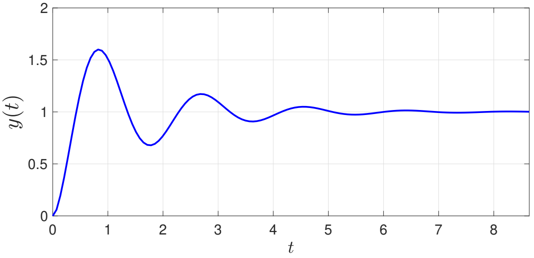
- physics is continuous-time, computers are discrete-time
- two methods for designing discrete-time controllers
  - approximate a given continuous-time controller (emulation)
  - directly design a new discrete-time controller (direct design)
- DAQ issues (sampling, conditioning, quantization)
- component selection (sensors, actuators, microprocessor)
- computing aspects (memory, computation time, priority, . . .)

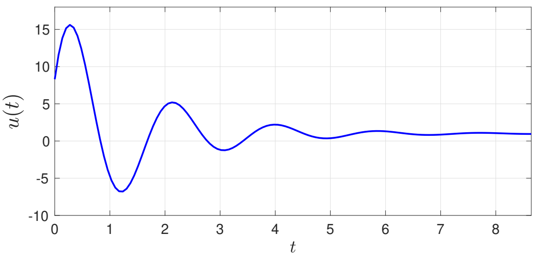
Continuous-time control systems



- standard unity-gain negative feedback configuration
- P is the *plant* or *system*, C is the *controller*
- $u(t)$ is the plant input, $y(t)$ is the plant output
- $r(t)$ is ref. signal, $d(t)$ is disturbance, $n(t)$ is noise
- $0 \leq t < \infty$ is the continuous time variable

Typical step response of continuous control system





Continuous-time control systems contd.

linear time-invariant systems can be analyzed with Laplace transforms

$$y(s) := \mathscr{L}\{y(t)\} = \int_0^\infty y(t)e^{-st}\,\mathrm{d}t\,.$$

typical control system objectives are

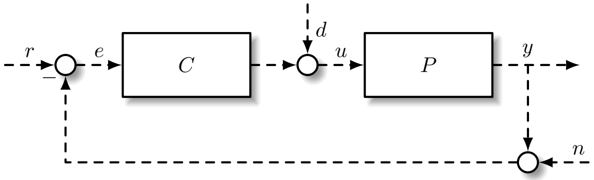
- closed-loop stability
- good transient performance for step response
- robustness to model uncertainty (property of feedback)
- attenuation (or outright rejection) of disturbances $d(t)$
- tracking $\lim_{t\to\infty} |y(t) - r(t)| = 0$

Continuous-time control systems contd.

Goal: Design controller $C(s)$ to achieve objectives. You already know tools for analyzing and designing these systems:

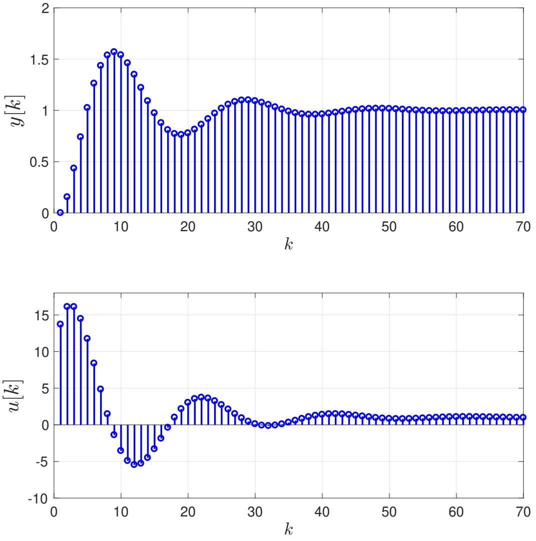
- stability analysis: poles, Routh-Hurwitz test
- stability margins: gain and phase margins
- steady-state analysis: final value theorem
- transient specs: settling time, rise time, overshoot
- frequency domain analysis: Bode plots, Nyquist plots
- controller designs: PID, possibly lead/lag as well

Discrete-time control systems



- standard unity-gain negative feedback configuration
- P is the *plant* or *system*, C is the *controller*
- $u[k]$ is the system input, $y[k]$ is the system output
- $r[k]$ is ref. signal, $d[k]$ is disturbance, $n[k]$ is noise
- $k = 0, 1, 2, 3, \dots$ is the discrete time variable

Typical step response of discrete control system



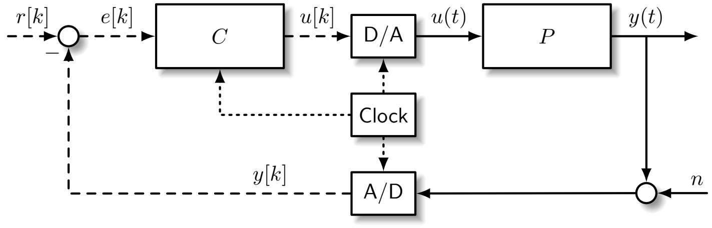
Discrete-time control systems contd.

linear time-invariant systems can be analyzed with z-transforms

$$y[z] := \mathcal{Z}\{y[k]\} = \sum_{k=0}^{\infty} y[k] z^{-k} \,.$$

- we will develop tools for
  - analyzing discrete-time control systems
  - designing discrete-time controllers
- Never *ever* <u>ever</u> mistake 'z' for 's'!

Digital control systems



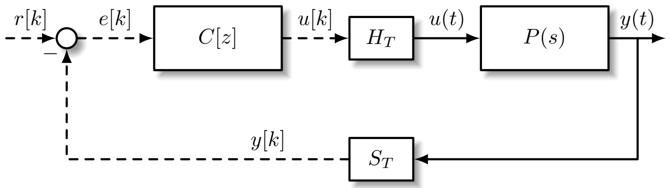
- continuous-time plant P, discrete-time controller C
- sampled (A/D) measurement $y[k]$ of plant output $y(t)$
- D/A interpolates digital control signal $u[k]$ to continuous $u(t)$
- between samples, *system runs in open-loop*!

Digital control systems contd.

some immediate issues are

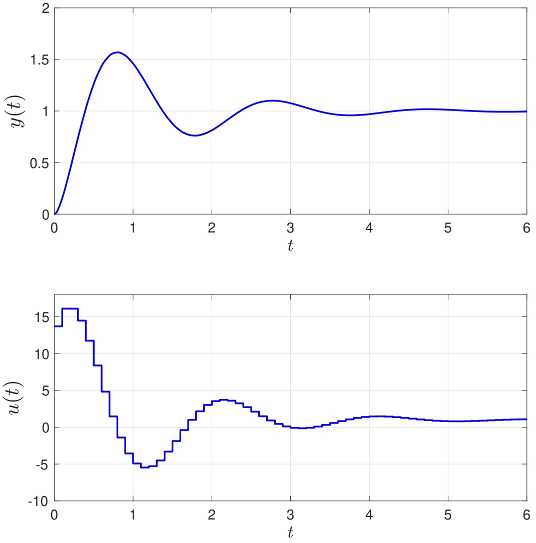
- how to model A/D and D/A
  - slew rates
  - quantization
  - synchronization errors
- how to select sampling rate (or rates)
- how to mitigate aliasing (we will define this shortly)

Sampled-data control systems



- sampled-data system is an *idealized* digital control system
- sampler S_T and hold H_T are perfectly timed, infinite precision
- sampled-data systems are **not** linear time-invariant systems!
- nonetheless, some simple analysis and design tools exist

Typical step response of sampled-data control system

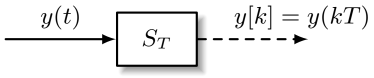


Design methods for sampled-data systems

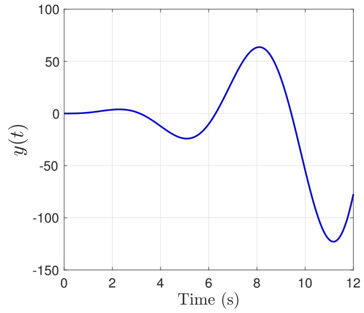
- emulation (*discrete equivalent*)
  - design $C(s)$ for $P(s)$, the *discretize* $C(s)$ to obtain $C[z]$
  - **advantage:** simple
  - **disadvantage:** requires "fast" sampling
- direct design (*discrete design*)
  - discretize $P(s)$ to obtain $P[z]$, then *directly design* $C[z]$ for $P[z]$
  - **advantage:** can handle "slow" sampling rates
  - **disadvantage:** more complicated to design and analyze

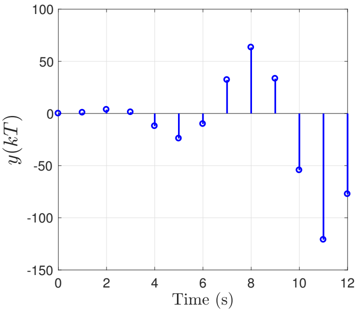
Ideal model for A/D conversion

- A/D modeled using *ideal sampler* block with period T



$$k = 0, 1, 2, 3 \ldots$$

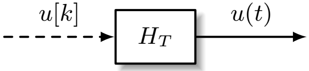




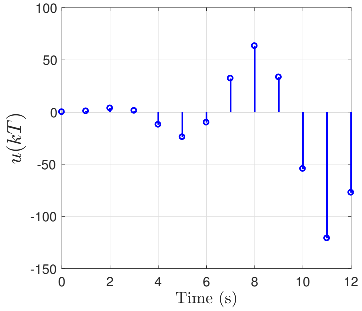
(note: $T = 1$s in figure)

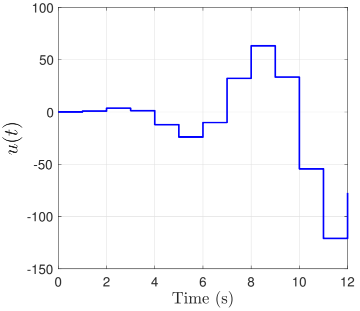
Ideal model for D/A conversion

- D/A modeled using *ideal zero-order hold* block with period T



$$u(t) = u[k] \quad \text{for} \quad kT \le t < (k+1)T$$





(note: $T = 1$s in figure)

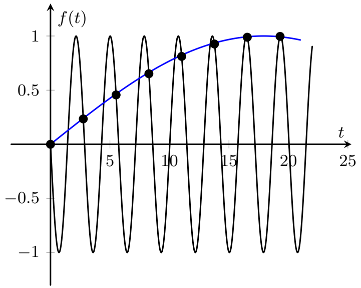
Assumptions for ideal sampler and ideal hold

- no quantization (infinite precision)
- sampling period T is constant and used by all sample/hold blocks
- all ideal samplers and ideal holds are synchronized
- no clock drift

validity of these assumptions depends on your specific hardware

Aliasing

- suppose we sample a signal $f(t)$ with sampling frequency $\omega_{\mathrm{s}} = \frac{2\pi}{T}$
- *aliasing* occurs when $f(t)$ contains frequency components higher than $\omega_{\mathrm{Nyq}} := \omega_{\mathrm{s}}/2$ (this is called the *Nyquist frequency*)



Aliasing contd.

- sampling signal $u_1(t) = \cos(\omega t)$ at times $t_k = kT$:

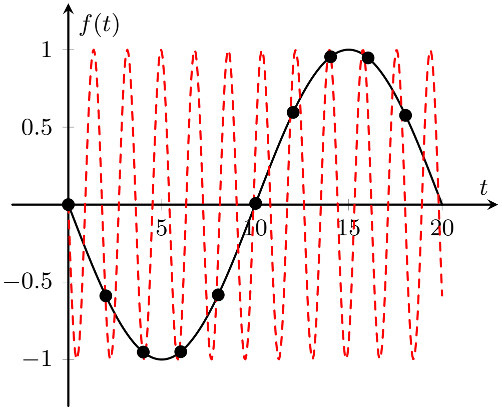
$$u_1[k] = \cos(\omega kT) = \cos\left(2\pi\frac{\omega}{\omega_\mathrm{s}}k\right)$$

- sampling signal $u_2(t) = \cos((\omega + \omega_s)t)$ yields

$$\begin{aligned} u_2[k] = \cos((\omega + \omega_\mathrm{s})kT) &= \cos\left(2\pi\frac{\omega + \omega_\mathrm{s}}{\omega_\mathrm{s}}k\right) = \cos\left(2\pi\frac{\omega}{\omega_\mathrm{s}}k + 2\pi k\right) \\ &= \cos\left(2\pi\frac{\omega}{\omega_\mathrm{s}}k\right) = u_1[k] \end{aligned}$$

- even though $u_1(t) \neq u_2(t)$, we have identical sampled signals

Aliasing contd.



Aliasing contd.

- sampling any member of the family of continuous-time signals

$$u_n(t) = \cos\left((\omega \pm n\omega_{\mathrm{s}})t\right)\,, \qquad n \in \mathbb{Z}$$

  produces *the same* discrete-time signal

$$u[k] = \cos\left(2\pi\frac{\omega}{\omega_{\mathrm{s}}}k\right)$$

- we say the frequencies $\{\omega \pm n\omega_{\mathrm{s}}\}$ are *aliases* of the base frequency ω with respect to the sampling frequency ω_{s}

Example: Data acquisition in a lab

- sampling frequency 50Hz, $T = 1/50$
- signal of interest is at frequency 4Hz
- noise from lights at 60Hz, 120Hz, 180Hz

- sampled signal will only have components less than 25Hz
- 60Hz noise will be aliased to $60 \pm n50$

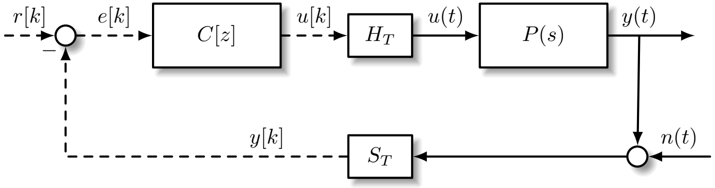
$$\{\ldots, -40, \mathbf{10}, 60, 110, \ldots\}$$

- 120Hz and 180Hz noise will be aliased to

$$\{\ldots, -30, \mathbf{20}, 70, \ldots\} \quad \text{and} \quad \{\ldots, -70, -\mathbf{20}, 30, 80, \ldots\}$$

Aliasing in control systems

- why care about aliasing for digital control?
- aliased noise enters controller, generating spurious control actions

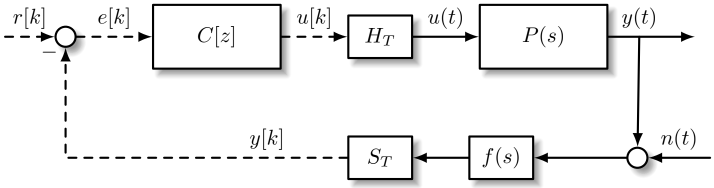


- example: 1 kHz noise in a motor control system, aliased down to 10Hz, will then generate a *real* 10Hz oscillation on the motor shaft

Strategies for avoiding aliasing

two possible strategies for avoiding aliasing

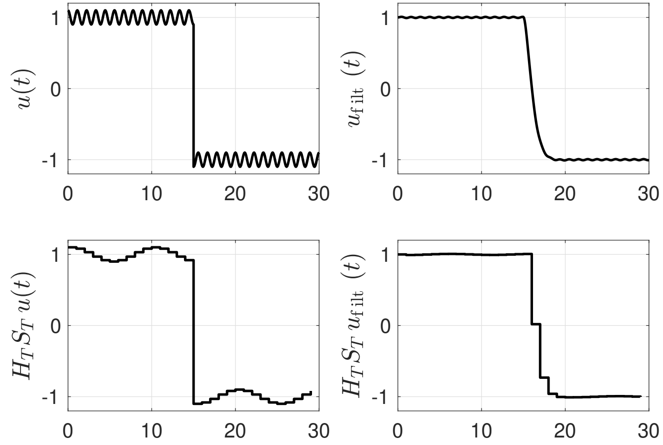
1. **sample very fast:** $\omega_{\rm s} > 2 \times$ (highest frequency in meas. noise)
    - often infeasible due to hardware limitations
2. include an analog **low-pass filter** $f(s)$ in loop with cutoff $\omega_{\rm cut}$



- added expense, inflexible, introduces *phase lag* in loop

Example: prefiltering before sampling

- square wave with 0.9Hz oscillation, 1Hz sampling



Competing constraints on sampling rates

- many competing constraints on sampling frequency ω_s

  1. faster sampling $=$ more expensive hardware
  2. in emulation approach to digital control, we *need* to sample fast, or the system will perform poorly
  3. if we sample very fast, might not be able to compute control actions between samples
  4. sampling rates often fixed by installed sensors and actuators

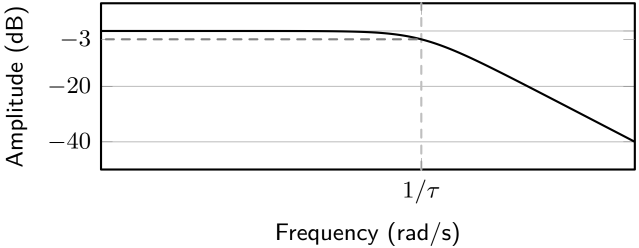
Guidelines for filter design

- achievable performance depends on frequency spectrum of noise and fixed sampling rates of system
- filter cutoff design is a compromise between
  1. introducing phase shift into the feedback loop
  2. successfully filtering out the noise
- to minimize extra phase lag at crossover, want $\omega_{\text{cut}} > \omega_{\text{bw}}$
- to filter noise above Nyquist, want $\omega_{\text{cut}} < \omega_{\text{Nyq}} = \omega_{\text{s}}/2$
- rough rule of thumb is therefore

$$\omega_{\text{bw}} < \omega_{\text{cut}} < \omega_{\text{s}}/2$$

Quick review: bandwidth of a system

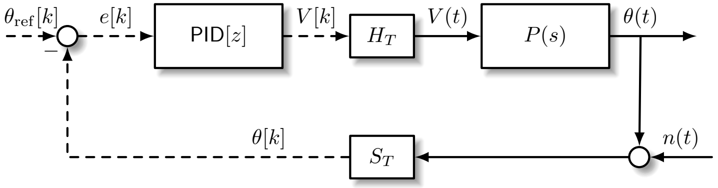
- for "low-pass" transfer function, bandwidth $\omega_{\rm bw}$ is the frequency where the magnitude drops -3 dB below low-frequency magnitude
- first order system $P(s) = \frac{1}{\tau s+1}$, $\tau > 0$



- Bandwidth is $\omega_{\rm bw} = 1/\tau$, and 90% rise time $\approx 2\tau$.

Example: DC Motor Position Control

- **goal:** fast step reference tracking for motor control
  - $\theta =$ angular position, $V =$ voltage applied
  - measurement noise of 0.02 rad at $f_{\text{noise}} = 1005$ Hz

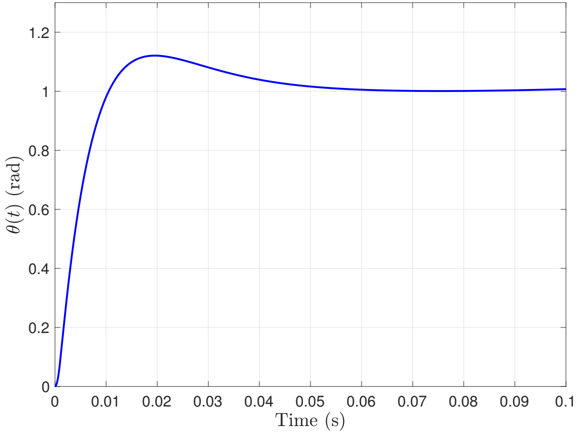


- motor model and controller given by

$$P(s) = \frac{1}{s}\frac{K}{(Js+b)(Ls+R)+K^2}\,, \quad \text{PID}[z] = K_{\text{p}} + K_{\text{d}}\frac{z-1}{z} + K_{\text{i}}\frac{z}{z-1}$$

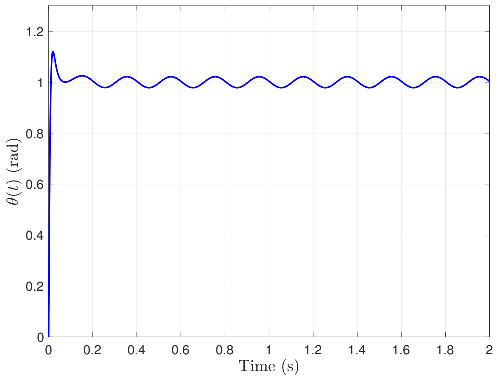
Example: DC Motor Position Control

- controller designed with sampling period $T = 0.001$s
- unit step response of sampled-data system looks great ...



Example: DC Motor Position Control

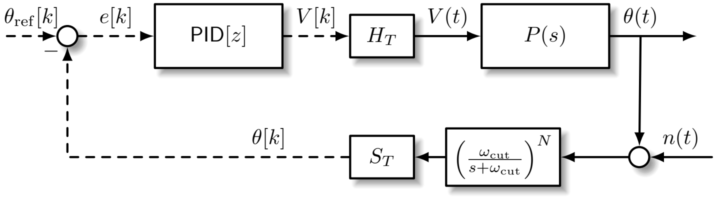
- but response contains unexpected ≈ 5 Hz oscillation



- **Why?** $f_{\rm s} = 1000$ Hz, $f_{\rm noise} = 1005$ Hz!

Example: DC Motor Position Control

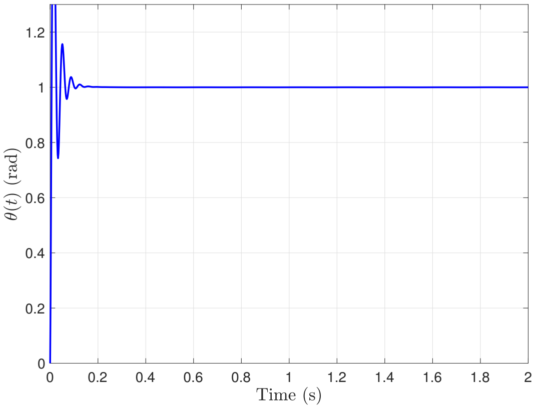
- let's add a low-pass anti-aliasing filter



- need to choose corner frequency ω_{cut} and order N of filter
  - system bandwidth: rise time ≈ 0.01 s $\Rightarrow$ $\omega_{\text{bw}} \approx 200$ rad/s
  - Nyquist frequency: $\omega_{\text{Nyq}} = (2\pi f_{\text{s}})/2 \approx 3100$ rad/s
- not a lot of room ($\approx$ one decade): use high-order for sharper cutoff

Example: DC Motor Position Control

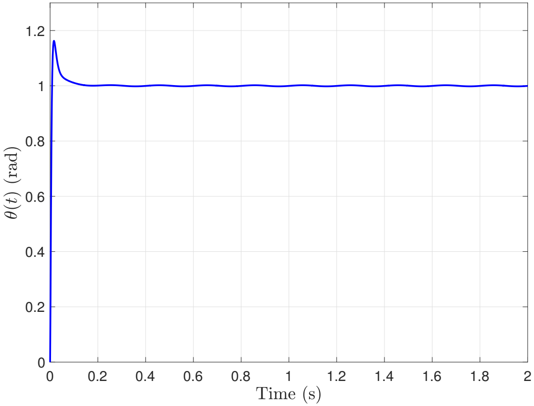
- step response with $\omega_{\text{cut}} = 400$ rad/s, $N = 2$



- cut-off too low (close to bandwidth), need to move cut-off up further

Example: DC Motor Position Control

- step response with $\omega_{\mathrm{cut}} = 3200$ rad/s, $N = 3$



- **exercise:** play with ω_{s} and N yourself in `MATLAB` code

MATLAB commands

- Plotting discrete-time signals: `stem(t,y)`
- Plotting held signals: `stairs(t,y)`
- Analog filtering signals (many different options)

```
s = tf(‘s’);
f = (2*pi*f_cut)^2 / ...
    (s^2 + 2*damping*2*pi*f_cut*s + (2*pi*f_cut)^2);
u_filt = lsim(f,u,t);
```

Additional references

- introduction to digital control
  - Nielsen, Chapter 0
  - **Franklin, Powell, & Emami-Naeini, Chapter 8**
  - Franklin, Powell, & Workman, Chapter 1 and Chapter 3
  - Phillips, Nagle, & Chakrabortty, Chapter 1
  - Astrom & Wittenmark, Chapter 1
- sampling
  - Phillips, Nagle, & Chakrabortty, Chapter 3.2
- signal reconstruction
  - Phillips, Nagle, & Chakrabortty, Chapter 3.7

Personal Notes

Personal Notes

Personal Notes

2. Classical continuous-time control systems

- modeling for controller design
- notation and Laplace transforms
- continuous-time LTI systems
- feedback stability
- time-domain analysis
- system approximation
- PID control and its variants
- static nonlinearities in control systems
- reference tracking and the internal model principle
- minor loop design

Modeling for controller design

- models can be obtained from
  - physics (Newton, Kirchhoff, Maxwell, fluid mechanics, . . .)
  - experiments on the system
  - combinations of these two
- many classes of possible models
  - finite-dimensional vs. infinite-dimensional
  - deterministic vs. random
  - linear vs. nonlinear
  - time-invariant vs. time-dependent
  - continuous-time vs. discrete-time vs. hybrid
- for control purposes, models just need to be "good enough"

Steps for successful modeling

1. model for design (simplified) vs. model for simulation (complex)
2. break system into collection of subsystems with inputs and outputs
3. model each subsystem, keeping track of assumptions, approximations, *etc.*
4. interconnect subsystems to form full model
5. validate model experimentally, repeat steps as needed

Notation

- set of real numbers $\mathbb{R}$
- $x \in S$ means x is a member of the set S
- set of complex numbers $\mathbb{C}$
- *strictly* left/right-hand complex plane $\mathbb{C}_-/\mathbb{C}_+$
- the *Laplace transform* of a signal $f(t)$ is given by

$$f(s) := \mathscr{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}\,\mathrm{d}t\,.$$

- Unit step function

$$\mathbb{1}(t) = \begin{cases} 1 & \text{if} \quad t \geq 0 \\ 0 & \text{if} \quad \text{else} \end{cases}$$

Key properties of Laplace transforms

- **linearity**

$$\mathscr{L}\{\alpha_1 f_1 + \alpha_2 f_2\} = \alpha_1 f_1(s) + \alpha_2 f_2(s)$$

- **delay**

$$\mathscr{L}\{f(t-\tau)\} = e^{-\tau s} f(s)$$

- **integral formula**

$$\mathscr{L}\left\{\int_0^t f(\tau)\,\mathrm{d}\tau\right\} = \frac{1}{s} f(s)$$

- **derivative formula**

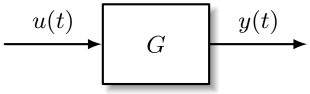
$$\mathscr{L}\left\{\frac{\mathrm{d}f}{\mathrm{d}t}\right\} = s f(s) - f(0)$$

- **convolution**

$$\mathscr{L}\{g * f\} = g(s) f(s)$$

Continuous-time causal LTI systems

- a *system* takes an input signal $u(t)$ and produces output signal $y(t)$



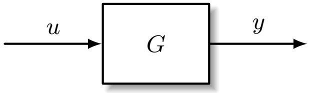
- **linearity:** if $y_1 = G(u_1)$ and $y_2 = G(u_2)$, then

$$G(\alpha_1 u_1 + \alpha_2 u_2) = \alpha_1 G(u_1) + \alpha_2 G(u_2) = \alpha_1 y_1 + \alpha_2 y_2 \,.$$

- **time-invariance:** if input $u(t)$ produces output $y(t)$, then for any delay τ, input $u(t-\tau)$ produces output $y(t-\tau)$.

- **causality:** if $u_1(t) = u_2(t)$ for all $0 \le t \le T$, then $y_1(t) = y_2(t)$ for all $0 \le t \le T$.

Representations of LTI systems



- in Laplace-domain, multiplication with *transfer function* $G(s)$

$$y(s) = G(s)u(s)$$

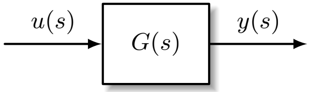
- in time-domain, convolution with *impulse response* $g(t)$

$$y(t) = (g * u)(t) = \int_0^t g(t-\tau)u(\tau)\,\mathrm{d}\tau\,.$$

- these are equivalent, can show that

$$G(s) = \mathscr{L}\{g(t)\} \quad \text{and} \quad g(t) = \mathscr{L}^{-1}\{G(s)\}$$

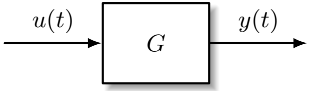
Transfer function representation contd.



- we call $G(s)$ *rational* if $G(s) = \frac{N(s)}{D(s)}$ for some polynomials $N(s)$ and $D(s)$ with real coefficients
- a *pole* $p \in \mathbb{C}$ of $G(s)$ satisfies $\lim_{s \to p} |G(s)| = \infty$
- a *zero* $z \in \mathbb{C}$ of $G(s)$ satisfies $G(z) = 0$
- the degree $\deg(D)$ of the denominator is the *order* of the system
- $G(s)$ is *proper* if $\deg(N) \leq \deg(D)$, *strictly proper* if $\deg(N) < \deg(D)$

Bounded-input bounded-output (BIBO) stability

- signal $y(t)$ is bounded if there is $M \geq 0$ s.t. $|y(t)| \leq M$ for all $t \geq 0$



- **BIBO stability:** every bounded $u(t)$ produces a bounded $y(t)$
- if the LTI system G is rational and proper, then G is BIBO stable if and only if either of the following equivalent statements hold:
  - every pole of transfer function $G(s)$ belongs to $\mathbb{C}_-$
  - the integral $\int_0^\infty |g(t)|\,\mathrm{d}t$ of the impulse response is finite.

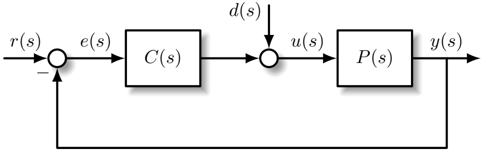
Examples

$$\frac{1}{s+1} \qquad \frac{1}{s-1} \qquad \frac{1}{s^2+2s+1}$$

$$K_\mathrm{p} + \frac{K_\mathrm{i}}{s} \qquad \frac{s-1}{s+1} \qquad \frac{1}{s^2+2s-1}$$

$$e^{-t} \qquad t^{100}e^{-t} \qquad \mathbb{1}(t) \qquad \sin(\omega_0 t)$$

Feedback stability



- r and d are *external* signals, e, u and y are *internal* signals
- **definition:** the closed-loop is *feedback stable* if every bounded pair of external inputs (r, d) leads to bounded internal responses (e, u, y)
- 2 external signals, 3 internal signals $\Longrightarrow$ 6 transfer functions

$$\frac{e(s)}{r(s)} = \frac{1}{1+PC} \qquad \frac{u(s)}{r(s)} = \frac{C}{1+PC} \qquad \frac{y(s)}{r(s)} = \frac{PC}{1+PC}$$

$$\frac{e(s)}{d(s)} = \frac{-P}{1+PC} \qquad \frac{u(s)}{d(s)} = \frac{1}{1+PC} \qquad \frac{y(s)}{d(s)} = \frac{P}{1+PC}$$

- feedback stable $\underline{\text{if and only if}}$ *all* transfer func. are BIBO stable

Feedback stability contd.

- assume $P(s)$ is rational and strictly proper: $P(s) = \frac{N_\mathrm{p}(s)}{D_\mathrm{p}(s)}$
- assume $C(s)$ is rational and proper: $C(s) = \frac{N_\mathrm{c}(s)}{D_\mathrm{c}(s)}$
- for example, we can calculate that

$$\frac{y(s)}{r(s)} = \frac{PC}{1+PC} = \frac{\frac{N_\mathrm{p}}{D_\mathrm{p}}\frac{N_\mathrm{c}}{D_\mathrm{c}}}{1+\frac{N_\mathrm{p}}{D_\mathrm{p}}\frac{N_\mathrm{c}}{D_\mathrm{c}}} = \frac{N_\mathrm{p}N_\mathrm{c}}{N_\mathrm{p}N_\mathrm{c}+D_\mathrm{p}D_\mathrm{c}}$$

- the denominator is the **characteristic polynomial**

$$\Pi(s) := N_\mathrm{p}(s)N_\mathrm{c}(s) + D_\mathrm{p}(s)D_\mathrm{c}(s)$$

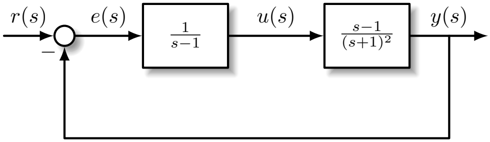
- under these assumptions, the closed-loop is feedback stable $\underline{\text{if and only if}}$ all roots of $\Pi(s)$ belong to $\mathbb{C}_-$.

Pole-zero cancellations

$$\text{plant:} \quad P(s) = \frac{N_{\mathrm{p}}(s)}{D_{\mathrm{p}}(s)} \qquad\qquad \text{controller:} \quad C(s) = \frac{N_{\mathrm{c}}(s)}{D_{\mathrm{c}}(s)}$$

- a plant P and a controller C have a *pole-zero cancellation* if either
  - (a) $N_{\mathrm{c}}(s)$ and $D_{\mathrm{p}}(s)$ have a common root, or
  - (b) $N_{\mathrm{p}}(s)$ and $D_{\mathrm{c}}(s)$ have a common root
- pole-zero cancellation is *stable* if the common root is in $\mathbb{C}_-$; otherwise, cancellation is *unstable* and must be avoided
- we only worry about pole-zero cancellations *between systems*; pole-zero cancellation *between a system and a signal* is fine!

Example of unstable pole-zero cancellation

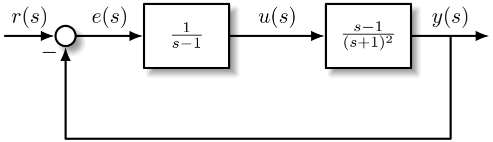


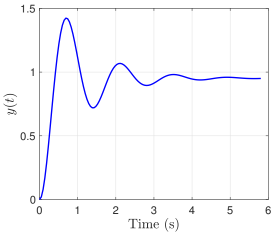
$$\Longrightarrow \qquad \begin{aligned} \Pi(s) &= (s+1)^2(s-1) + (s-1) \\ &= (s-1)(s^2+2s+2) \end{aligned}$$

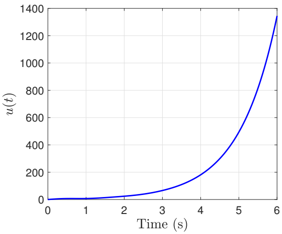
- root at $s = 1$, feedback system is unstable. Note that

$$\frac{y(s)}{r(s)} = \frac{1}{s^2+2s+2}, \qquad \frac{u(s)}{r(s)} = \frac{(s+1)^2}{(s-1)(s^2+2s+2)}$$

Example of unstable pole-zero cancellation

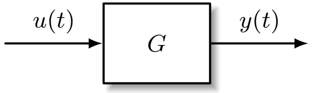


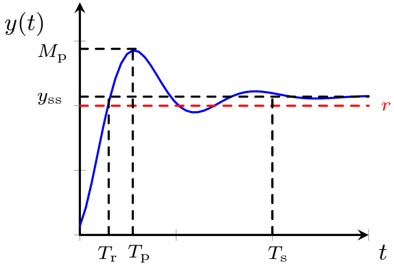




- Output $y(t)$ looks fine, but control $u(t)$ blows up!

Time-domain analysis





(i) rise time T_r, settling time T_s

(ii) peak time T_p, peak max M_p

(iii) steady state value y_ss?

- **Final value theorem:** if $f(s) = \mathscr{L}\{f(t)\}$ is rational and proper, with all poles of $sf(s)$ contained in $\mathbb{C}_-$ then $f_\mathrm{ss} = \lim_{t\to\infty} f(t)$ exists and

$$f_\mathrm{ss} = \lim_{t\to\infty} f(t) = \lim_{s\to 0} sf(s)\,.$$

The "DC gain" of $G(s)$ is $G(0)$

- suppose $G(s)$ is rational, proper, and BIBO stable T.F.
- response to step input of amplitude A, i.e., $u(s) = \frac{A}{s}$ is

$$y(s) = G(s)u(s) = G(s)\frac{A}{s}$$

- final value of response given by

$$y_{\mathrm{ss}} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} sy(s) = \lim_{s\to 0} sG(s)\frac{1}{s} \cdot A = G(0) \cdot A$$

- $G(0)$ is therefore the *steady-state gain* of the system, *i.e.*, the amplification a constant input will experience

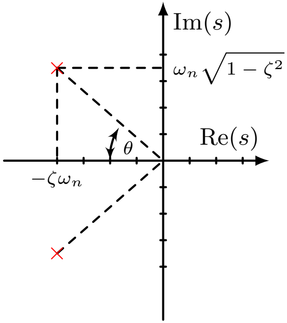
Second-order systems

$$G(s) = K\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

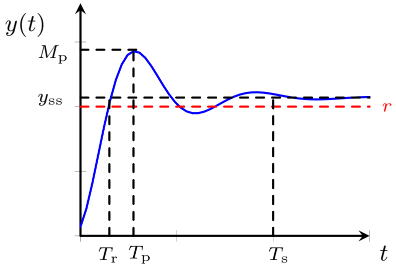
- natural frequency $\omega_n > 0$, damping ratio $\zeta > 0$, DC gain K
- many systems can be well-approximated as second-order
- overdamped $(\zeta > 1)$, critically damped $(\zeta = 1)$
- underdamped $(\zeta < 1)$

$$\begin{aligned} s &= -\zeta\omega_n \pm \mathrm{j}\omega_n\sqrt{1-\zeta^2} \\ &= \omega_n e^{\pm \mathrm{j}(\pi-\theta)} \end{aligned}$$

$$\theta = \arccos(\zeta)\,.$$



Second-order systems contd.



- closed-form results

$$T_{\mathrm{p}} = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}}, \qquad T_{\mathrm{s}} \approx \frac{4}{\zeta\omega_n}$$

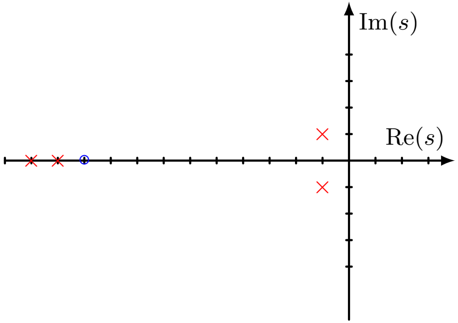
$$\%\mathsf{OS} = \frac{M_{\mathrm{p}} - K}{K} = \exp\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)$$

- to identify parameter values
  - apply step
  - measure $y_{\mathrm{ss}}, \%$OS, and T_{p}
  - solve for ω_n, ζ, K

System approximation

- approximate higher-order system with lower-order one
- example:

$$G(s) = \frac{s+10}{(s+11)(s+12)(s^2+2s+2)}$$



System approximation contd.

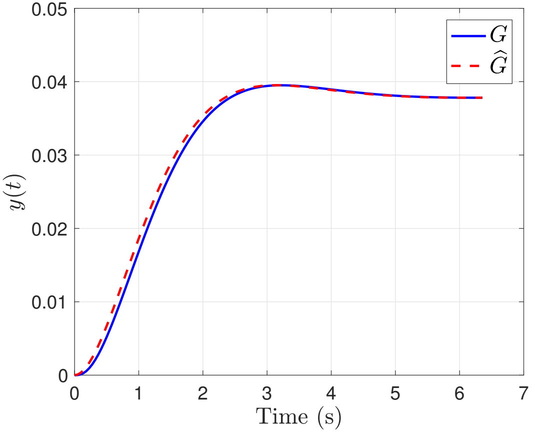
$$\begin{aligned} G(s) &= \frac{s+10}{(s+11)(s+12)(s^2+2s+2)} \\ &= \underbrace{\frac{s+10}{(s+11)(s+12)}}_{G_{\mathrm{fast}}(s)} \cdot \underbrace{\frac{1}{s^2+2s+2}}_{G_{\mathrm{slow}}(s)} \end{aligned}$$

- if $G_{\mathrm{fast}}(s)$ is BIBO stable
  $\Longrightarrow$ response due to $G_{\mathrm{fast}}(s)$ quickly reaches steady-state
- replace $G_{\mathrm{fast}}(s)$ with its steady-state value (DC gain) $G_{\mathrm{fast}}(0)$

$$\widehat{G}(s) \approx G_{\mathrm{fast}}(0) G_{\mathrm{slow}}(s) \,.$$

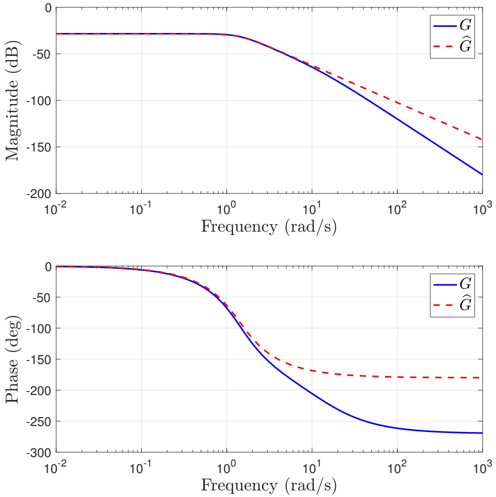
- valid if fast poles/zeros are approx. 10x faster than slow poles/zeros

Step response for example



▶ **note:** approximation very good for step inputs

Bode diagram for example



- **note:** approximation very good up to ≈ 1 rad/s

PID control

- most basic proportional-integral-derivative controller

$$u(t) = K\left(e(t) + \frac{1}{T_{\mathrm{i}}}\int_0^t e(\tau)\,\mathrm{d}\tau + T_{\mathrm{d}}\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right)$$

  where K is the proportional gain, T_{i} is the integral time constant, and T_{d} is the derivative time constant

- in transfer function form, we have

$$C(s) = K\left(1 + \frac{1}{sT_{\mathrm{i}}} + T_{\mathrm{d}}s\right)$$

- this form is called the "non-interacting" parameterization; other parameterizations are also common (*e.g.*, with gains $K_{\mathrm{p}}, K_{\mathrm{i}}, K_{\mathrm{d}}$);

PID control contd.

- the rough interpretation of the terms is
  - **present:** proportional term reacts to present error
  - **past:** integral term reacts to cumulative error in the past
  - **future:** derivative term reacts to rate of change of error (tries to *predict* where error is going)

- PID is bread-and-butter: likely 90% of all controllers in industry are PID, and 90% of those are PI controllers

- standard tuning methods available (e.g., Ziegler-Nichols, or fancier ones like `pidtool` in `MATLAB`)

- we will not review the basics of tuning PID loops here, but will instead look at some refinements and advanced topics

PID control with derivative filter

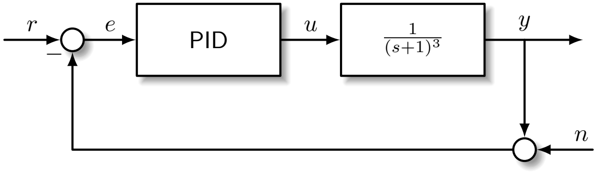
- pure derivative control is *never* implemented, as it is very sensitive to high-frequency noise (to convince yourself, plot the Bode plot)
- all implementations add a *low-pass filter* to derivative term

$$C(s) = K\left(1 + \frac{1}{sT_{\mathrm{i}}} + \frac{T_{\mathrm{d}}s}{T_{\mathrm{d}}s/N + 1}\right)$$

- time constant of low-pass filter is T_{d}/N
- N ranges between roughly 5 and 20
  - larger $N \implies$ less noise filtering, but better control performance
  - smaller $N \implies$ more noise filtering, but worse control performance

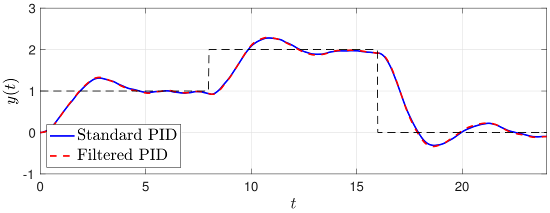
Example: PID with derivative filter

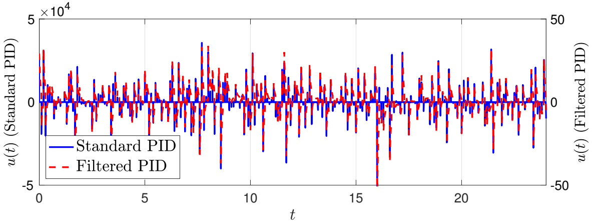
- consider PID control system with measurement noise



- PID controller gains $K = 3$, $T_\mathrm{i} = 2\,\mathrm{s}$, $T_\mathrm{d} = 0.5\,\mathrm{s}$, $N = 10$
- measurements $y(t)$ corrupted by measurement noise

Example: PID with derivative filter contd.

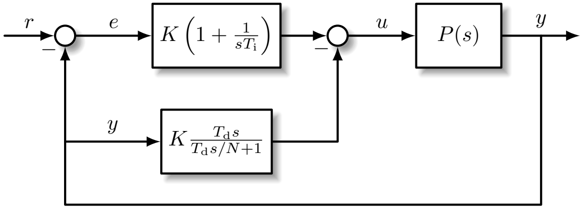




- note the two y-axes on the plots of $u(t)$; standard PID control generates *huge* control signals due to fast-varying noise

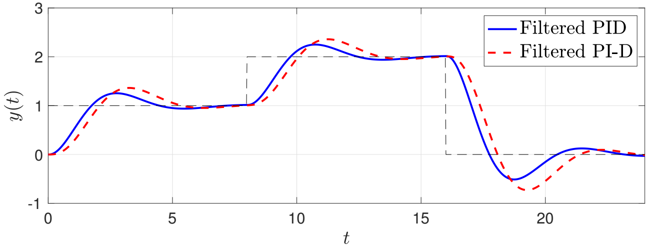
PI-D control

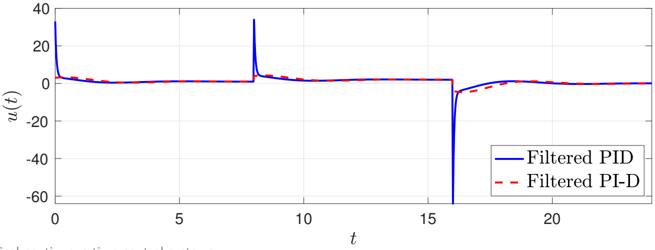
- derivative term of PID control takes derivative of error signal
- step change in reference signal $\Longrightarrow$ step change in the error signal $\Longrightarrow$ *big derivative* term $\Longrightarrow$ *big spike* in control signal
- to avoid this, apply derivative control only to y, not to e


r
e
$-$
$K\left(1+\frac{1}{sT_\mathrm{i}}\right)$
$-$
u
$P(s)$
y
y
$K\frac{T_\mathrm{d}s}{T_\mathrm{d}s/N+1}$

Example: PI-D control

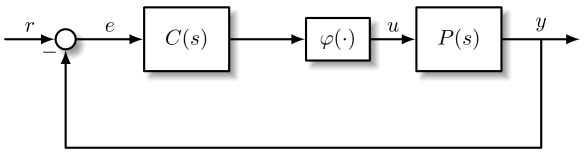
- plant and controller gains same as previous example





Static nonlinearities in control systems

- sensors and actuators are often nonlinear



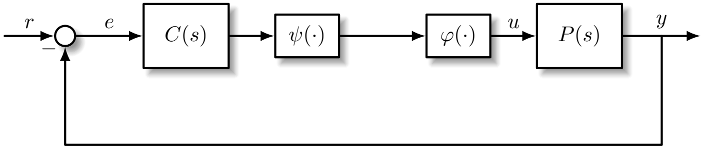
- **example:** suppose you want to shine light on a semiconductor surface by applying a voltage to a filament: irradiance $\sim$ voltage^2
- common nonlinearities: hysteresis, relay, saturation, deadzone . . .

Inverting actuator nonlinearities

- a function $\varphi(\cdot)$ is *right-invertible* if there is a function $\psi(\cdot)$ such that

$$\varphi(\psi(x)) = x$$

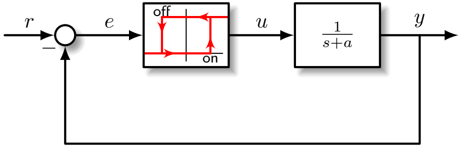
- **example:** $\psi(x) = \sqrt{x}$ is a right-inverse for $\varphi(x) = x^2$ for all $x \geq 0$
- many actuator nonlinearities can be handled by using a right-inverse



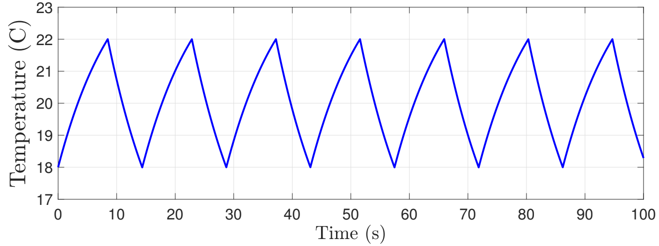
- now can design $C(s)$ for $P(s)$ like usual

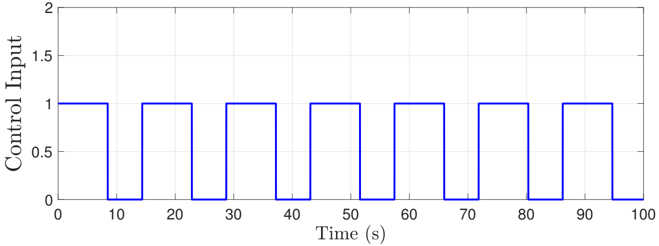
Limit cycles due to nonlinearities

- a *limit cycle* is a sustained nonlinear oscillation
- common in nature (heart beats, planetary orbits, animal populations)
- limit cycles occur in feedback systems with certain nonlinearities
- **example:** thermostat control systems



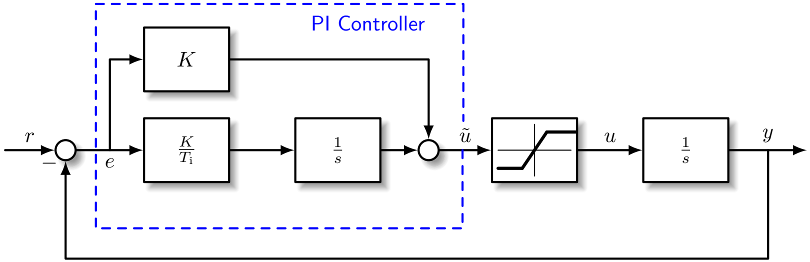
Thermostat temperature response to 20° setpoint





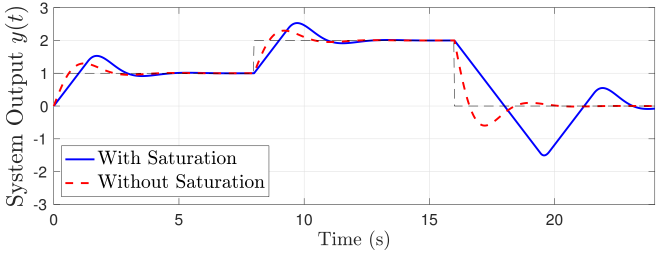
Saturation in control systems

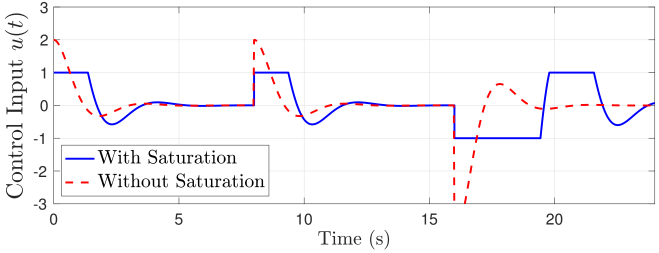
- another very common nonlinearity is *saturation*



- saturation limits output from controller
- when saturation is reached, control loop is "broken"
- can cause steady-state error, bad performance, or instability

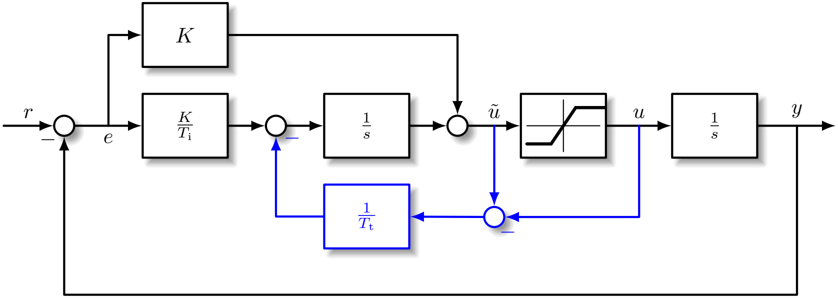
Example: step response of system with saturation





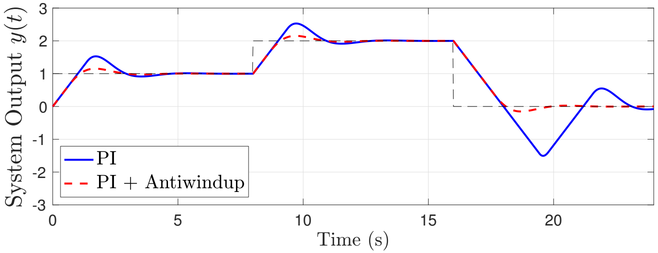
Anti-windup control

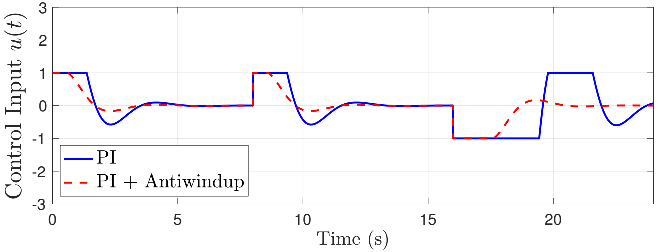
- bad performance is caused by so-called "integrator windup"; see Åström, Chapter 6 for details; solution is to feed back error between u and $\tilde{u}$ to the integrator of the PI controller



- time constant $T_\mathrm{t} > 0$ can be tuned (smaller is usually better); if you are also including derivative control, choose $T_\mathrm{d} < T_\mathrm{t} < T_\mathrm{i}$

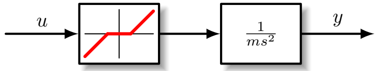
Example: performance with anti-windup ($T_\mathrm{t} = 0.1$s)

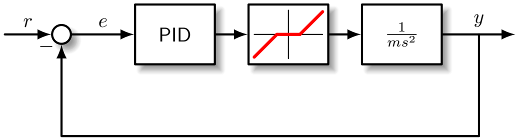




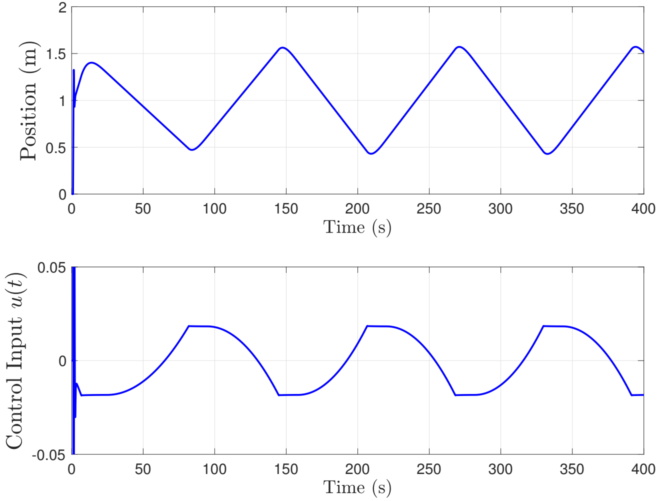
Static friction in mechanical control systems

- very common nonlinearity, difficult to model accurately
- can lead to tracking errors, oscillations, instability
- deadzone model of static friction



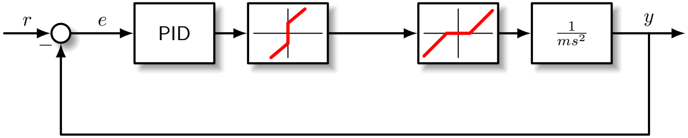


Example: step response of mass with PID + stiction



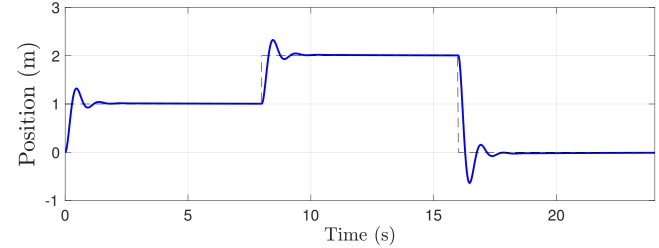
Control strategies for static friction

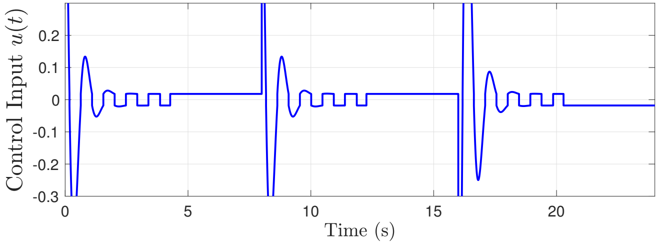
- **one approach:** invert deadzone nonlinearity in controller



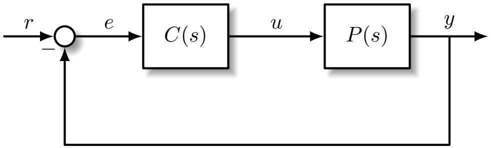
- size of inverted deadzone must be tuned
- other approaches: dither signal, dynamic friction estimator

Example: PID with friction compensation





Tracking reference signals



$$P(s) = \frac{N_{\mathrm{p}}(s)}{D_{\mathrm{p}}(s)}$$

$$C(s) = \frac{N_{\mathrm{c}}(s)}{D_{\mathrm{c}}(s)}$$

- **Perfect asymptotic tracking problem:** Given $P(s)$, design controller $C(s)$ such that closed-loop system is feedback stable and

$$\lim_{t\to\infty} (r(t) - y(t)) = 0\,.$$

- step tracking is special case when $r(t)$ is a step function
- application areas: power electronics, position control, robotics, ...

Requirements for step tracking

transfer function from $r(s)$ to $e(s) = r(s) - y(s)$ is

$$\frac{e(s)}{r(s)} = \frac{1}{1 + P(s)C(s)} = \frac{D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)}{N_{\mathrm{p}}(s)N_{\mathrm{c}}(s) + D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)} = \frac{D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)}{\Pi(s)}\,.$$

- suppose we had a step $r(t) = \mathbb{1}(t) \quad \Longrightarrow \quad r(s) = \frac{1}{s}$

$$e(s) = \frac{D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)}{\Pi(s)} \cdot \frac{1}{s}$$

- by final value theorem $e(t) \to 0$ if

  (a) $\Pi(s)$ has all roots in $\mathbb{C}_-$ (i.e., system is feedback stable)
  (b) $\lim_{s\to 0} \frac{D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)}{\Pi(s)} = 0 \qquad \Longleftrightarrow \qquad D_{\mathrm{p}}(0)D_{\mathrm{c}}(0) = 0.$

- therefore, *product* of P and C must have a pole at $s = 0$

Three cases for step-tracking

- if $P(s)$ has a zero at $s = 0$, then step-tracking is not possible (why?)
- if $P(s)$ has a pole at $s = 0$, $C(s)$ can be any stabilizing controller
- if $P(s)$ does not have a pole or zero at $s = 0$, then $C(s)$ <u>must</u> have a pole at $s = 0$
  - **design approach:** let $C(s) = \frac{1}{s}C_1(s)$, then design stabilizing $C_1(s)$
- having a pole at $s = 0$ in the controller is *integral control*

$$u(s) = \frac{1}{s}e(s) \qquad \Longrightarrow \qquad u(t) = \int_0^t e(\tau)\,\mathrm{d}\tau$$

- how does this generalize for other reference signals?

The Internal Model Principle

- **definition:** a transfer function $G(s)$ contains an *internal model* of a reference signal $r(s)$ if every pole of $r(s)$ is a pole of $G(s)$
- example:

$$G(s) = \frac{s+1}{(s^2+1)(s^2-4)} \qquad r(s) = \frac{s+2}{(s^2+1)(s+3)}$$

Internal Model Principle: *Assume $P(s)$ is strictly proper, $C(s)$ is proper, and that the closed-loop system is feedback stable. Then $\lim_{t\to\infty}(y(t)-r(t)) = 0$ if and only if $P(s)C(s)$ contains an internal model of the unstable part of $r(s)$.*

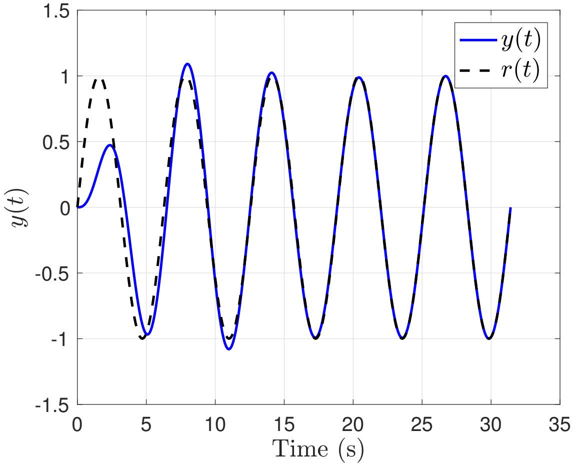
Example: sinusoidal tracking

$$P(s) = \frac{1}{s+1} \qquad r(t) = r_0 \sin(t) \qquad \Longrightarrow \qquad r(s) = \frac{r_0}{s^2+1}$$

Let's choose: $$C(s) = \frac{1}{s^2+1} C_1(s)$$

- exercise: check using Routh that $C_1(s) = \frac{s}{\tau s + 1}$ is stabilizing for sufficiently small $\tau > 0$
- exercise: check by hand using FVT that $\lim_{t \to \infty} e(t) = 0$

Example: sinusoidal tracking



Example: ramp tracking

- plant is double-integrator

$$P(s) = \frac{1}{s^2}$$

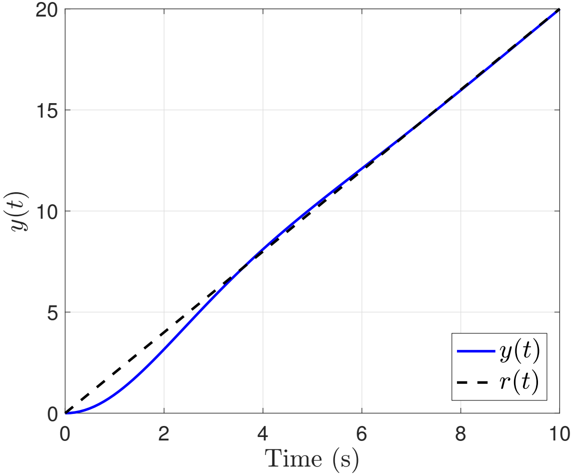
- reference signal is ramp

$$r(t) = 2t \qquad \Longrightarrow \qquad r(s) = \frac{2}{s^2}$$

- plant contains internal model, just design stabilizing $C(s)$
- for example, filtered PD controller

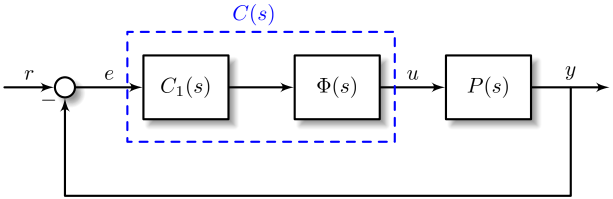
$$C(s) = K_{\mathrm{p}} + K_{\mathrm{d}} \frac{s}{\tau s + 1}$$

Example: ramp tracking



More comments on stabilization step for tracking

- our controller is $C(s) = \Phi(s)C_1(s)$, where $\Phi(s)$ includes the poles that we know we need for tracking – how do we design $C_1(s)$?

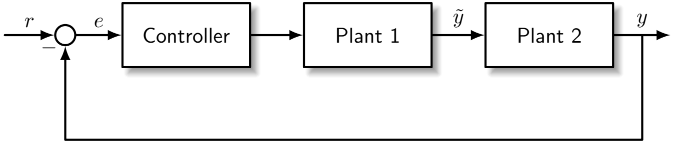


- we can equivalently group $\Phi(s)$ with $P(s)$ (same block diagram)

- use the *augmented plant* $P_{\text{aug}}(s) = \Phi(s)P(s)$, and design $C_1(s)$ using any technique you want to stabilize the system $P_{\text{aug}}(s)$

Minor loop design

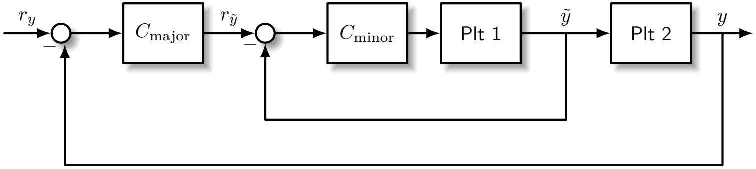
- a.k.a. *master/slave*, *primary/secondary*, or *cascade* design
- extremely common (process control, motors, power electronics . . .)
- very useful when you have an *intermediate* measurement (e.g., plant is a series combination of two subsystems)



- if we can also measure $\tilde{y}$, lets use *another* feedback loop

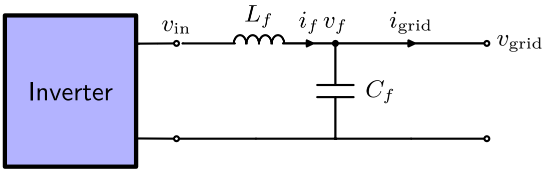
Minor loop design

- **idea:** design sequence of nested feedback loops



- *minor/inner loop* is designed first
  - control such that $\tilde{y}$ quickly tracks $r_{\tilde{y}}$ (high-bandwidth)
- *major/outer loop* is designed second
  - lets y track r_y (lower bandwidth by factor 5–10)

Example: single phase inverter



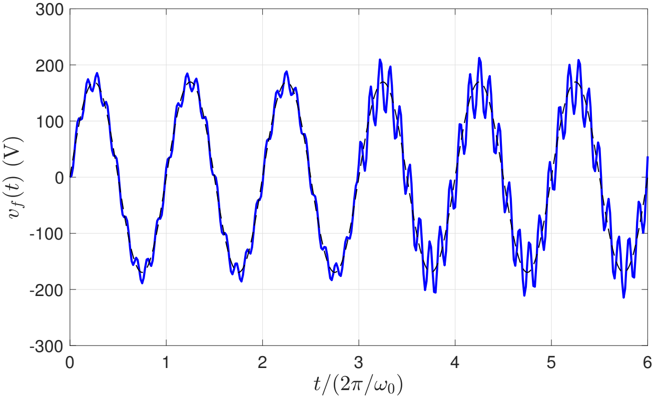
- LC filter removes harmonics from switching inside inverter
- objective: design input voltage v_{in} such that $v_f(t)$ tracks signal

$$v_{\text{ref}}(t) = \sqrt{2} \cdot 120 \cdot \cos(\omega_0 t)\,, \quad \omega_0 = 2\pi 60\,\text{rad/s}$$

- we can measure *both* current i_f and voltage v_f

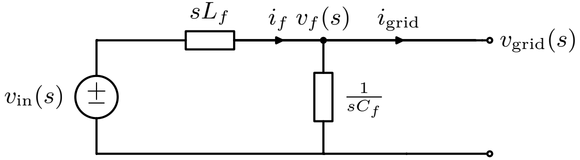
Response of single-phase inverter

- open-loop response with $v_{\mathrm{in}}(t) = v_{\mathrm{ref}}(t) = \sqrt{2} \cdot 120 \cdot \cos(\omega_0 t)$
- disturbance $i_{\mathrm{grid}}(t) = \sqrt{2} \cdot 20 \cdot \sin(\omega_0 t)$ applied after 3 cycles



Single phase inverter contd.

- Laplace-domain representation of circuit

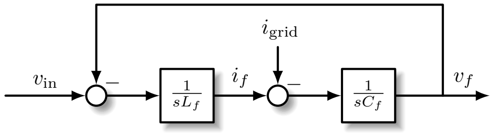


inductor current: $\quad i_f(s) = \dfrac{1}{sL_f}(v_{\text{in}}(s) - v_f(s))$

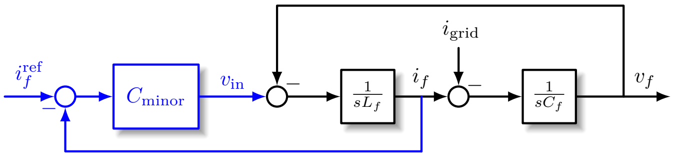
current balance: $\quad sC_f v_f(s) = i_f(s) - i_{\text{grid}}(s)$

Single phase inverter contd.

- feedback diagram for plant



- design *minor current loop* so current i_f tracks reference i_f^{ref}



Single phase inverter contd.

- typically use a PI controller

$$C_{\mathrm{minor}}(s) = K_{\mathrm{p}} \frac{1 + \tau s}{\tau s}$$

  where K_{p} is the proportional gain, τ is time constant

- **key point:** minor loop must be *fast* (K_{p} big, τ small)

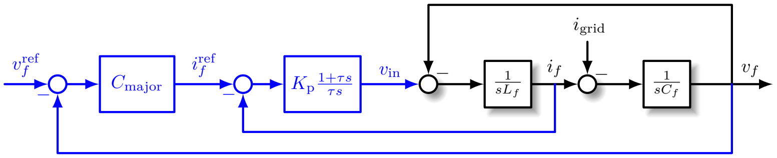
- over time-scale of minor loop, v_f is a constant disturbance

$$\Pi_{\mathrm{minor}}(s) = K_{\mathrm{p}}(1 + \tau s) + \tau s(sL_f) = \tau L_f s^2 + K_{\mathrm{p}} \tau s + K_{\mathrm{p}}$$

- for critical damping, take $K_{\mathrm{p}} = 4L_f/\tau$, then make $\tau \ll \frac{1}{\omega_0}$

Single phase inverter contd.

- design *major voltage loop* so v_f tracks v_f^{ref}



- to track sinusoidal v_f^{ref}, $C_{\text{major}}(s)$ needs *internal model*

$$\mathscr{L}\{\cos(\omega_0 t)\} = \frac{s}{s^2+\omega_0^2} \quad \Longrightarrow \quad C_{\text{major}}(s) = K_1 + K_2\frac{s}{s^2+\omega_0^2}\,.$$

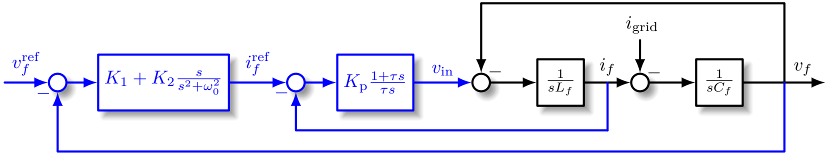
- called a *proportional-resonant* controller

Single phase inverter contd.

- now design major loop separately

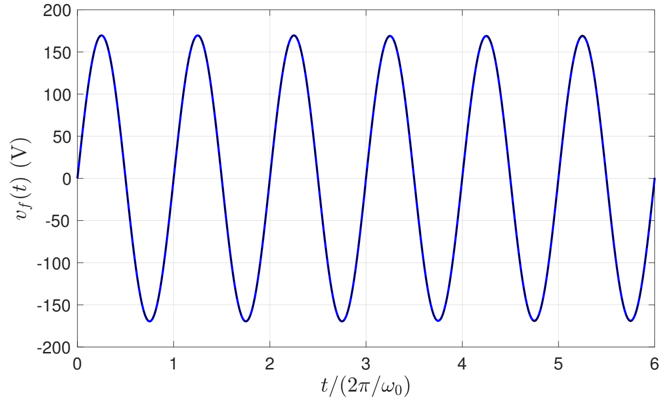
$$\Pi_{\text{major}}(s) = C_f s^3 + K_1 s^2 + (C_f\omega_0^2 + K_2)s + K_1\omega_0^2$$

- **exercise:** derive the stability conditions $K_1 > 0$, $K_2 > 0$



Single phase inverter contd.

- inverter with $C_f = 50\mu$F$, L_f = 1.5$mH, v_{ref} is 60Hz 120V RMS
- disturbance $i_{\text{grid}} = \sqrt{2} \cdot 20 \cdot \sin(\omega_0 t)$ applied after 3 cycles



MATLAB commands

- computing Laplace transform $F(s)$ of $f(t) = \sin(\omega t)$

  ```
  syms t w s; F = laplace(sin(w*t),s);
  ```

- inverse Laplace transform

  ```
  f = ilaplace(w/(s^2 + w^2),t);
  ```

- defining transfer functions

  ```
  s = tf('s'); G = (s-2)/(s^2+3);
  ```

-
  ```
  pole(G); zero(G); step(G); bode(G);
  ```

- feedback interconnection

  ```
  T = feedback(P*C,1);
  ```

Additional references

- Nielsen, Chapters 1 and 2
- Franklin, Powell, and Workman, Chapter 2
- Franklin, Powell, and Emami-Naeini, Chapter 3 and Chapter 4
- MTE 360 / ECE 380 course notes
- Åström & Murray, Chapters 8, 9, 10, 11
- Åström, Chapter 6 (pdf)

Additional references

- tracking reference signals
  - Franklin, Powell, & Emami, Chapter 4.2
  - Nielsen, Chapter 1.6
- minor loop design
  - Wikipedia, Minor loop feedback
- Smith predictor
  - Franklin, Powell, & Emami, Chapter 7.13

Personal Notes

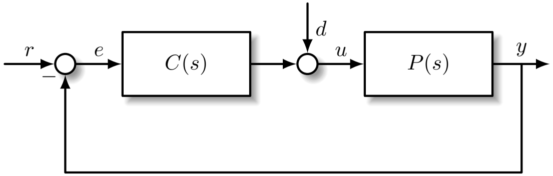
Personal Notes

Personal Notes

3. Emulation design of digital controllers

- controller emulation
- emulation techniques
- stability of discretized controllers
- emulation design procedure
- modified emulation design procedure

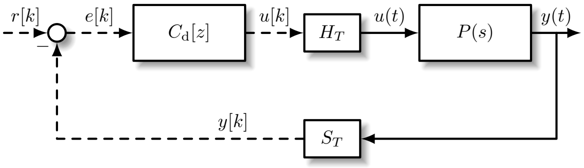
Controller emulation



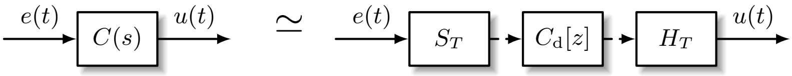
- suppose we have designed a continuous controller $C(s)$ for $P(s)$
- need to convert $C(s)$ to digital controller $C[z]$
- some immediate questions:
  - is there a unique way to do this?
  - how do we select sampling rates?

The emulation problem

- given an analog controller $C(s)$, select a sampling period $T > 0$ and a discrete controller $C_{\mathrm{d}}[z]$ such that the sampled-data control system



closely approximates the continuous-time control system



Quick review: z-transforms

- discrete signal $f[k]$ is a sequence $f[0], f[1], f[2], \ldots$
- the z-transform of a discrete-time signal $f[k]$ is

$$f[z] := \mathcal{Z}\{f[k]\} = \sum_{k=0}^{\infty} f[k]z^{-k}\,, \qquad z \in \mathbb{C}\,.$$

- **linearity:** $\mathcal{Z}\{\alpha_1 f_1 + \alpha_2 f_2\} = \alpha_1 f_1[z] + \alpha_2 f_2[z]$
- **delay formula**

$$\mathcal{Z}\left\{f[k-1]\right\} = z^{-1}f[z]$$

- **convolution**

$$\mathcal{Z}\{g * f\} = \mathcal{Z}\left\{\sum_{m=0}^{k} g[k-m]f[m]\right\} = g[z]f[z]$$

Controller emulation

- we will not do a general derivation, but motivate through an example
- integral controller $C(s) = 1/s$

$$u(s) = \frac{1}{s}e(s) \quad \Longleftrightarrow \quad \dot{u}(t) = e(t) \quad \Longleftrightarrow \quad u(t) = u(t_0) + \int_{t_0}^{t} e(\tau)\,\mathrm{d}\tau$$

- how to implement this in discrete-time? For sampling period T

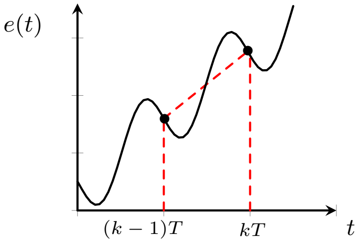
$$u(kT) = u((k-1)T) + \int_{(k-1)T}^{kT} e(\tau)\,\mathrm{d}\tau$$

$$u[k] = u[k-1] + \int_{(k-1)T}^{kT} e(\tau)\,\mathrm{d}\tau$$

- need to approximate integral over interval between samples

The bilinear discretization

- also called *trapezoidal* or *Tustin* discretization



$$\text{slope} = \frac{e(kT) - e((k-1)T)}{T}$$

$$\int_{(k-1)T}^{kT} \approx \underbrace{Te[k-1]}_{\text{rectangle}} + \underbrace{\frac{1}{2}T\left(e[k] - e[k-1]\right)}_{\text{triangle}}$$

$$= \frac{T}{2}\left(e[k-1] + e[k]\right)$$

The bilinear discretization contd.

- we therefore have the difference equation

$$u[k] = u[k-1] + \frac{T}{2}\left(e[k-1] + e[k]\right)$$

- taking z-transforms, we obtain

$$u[z] = z^{-1}u[z] + \frac{T}{2}\left(z^{-1}+1\right)e[z] \quad \implies \quad \frac{u[z]}{e[z]} = C_{\mathrm{d}}[z] = \frac{T}{2}\frac{z+1}{z-1}$$

- but our original controller was $C(s) = \frac{1}{s}$. Comparing, we find

$$C_{\mathrm{d}}[z] = C(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} \qquad \iff \qquad s = \frac{2}{T}\frac{z-1}{z+1}$$

- derivation was for integral controller, but this rule is **general**

Example: PID controller

$$C(s) = K_{\mathrm{p}} + K_{\mathrm{d}} s + \frac{1}{s} K_{\mathrm{i}} \quad \Longrightarrow \quad C_{\mathrm{d}}[z] = K_{\mathrm{p}} + K_{\mathrm{d}} \frac{2}{T} \frac{z-1}{z+1} + K_{\mathrm{i}} \frac{T}{2} \frac{z+1}{z-1}$$

$$C_{\mathrm{d}}[z] = \frac{2TK_{\mathrm{p}}(z+1)(z-1) + 4K_{\mathrm{d}}(z-1)^2 + T^2 K_{\mathrm{i}}(z+1)^2}{2T(z+1)(z-1)}$$

therefore, we obtain $\qquad C_{\mathrm{d}}[z] = \dfrac{\beta_0 z^2 + \beta_1 z + \beta_2}{z^2 - 1}$

$$\beta_0 = (2TK_{\mathrm{p}} + 4K_{\mathrm{d}} + T^2 K_{\mathrm{i}})/(2T)\,, \quad \beta_1 = (2T^2 K_{\mathrm{i}} - 8K_{\mathrm{d}})/(2T)$$
$$\beta_2 = (4K_{\mathrm{d}} + T^2 K_{\mathrm{i}} - 2TK_{\mathrm{p}})/(2T)$$

From transfer function to difference equation

- after simplifying $C_{\mathrm{d}}[z]$, we *always* get a rational TF (why?)

$$C_{\mathrm{d}}[z] = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n}$$

- for implementation need a difference equation

1. divide top and bottom through by z^n

$$C[z] = \frac{u[z]}{e[z]} = \frac{\beta_0 + \beta_1 z^{-1} + \cdots + \beta_n z^{-n}}{1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n}}$$

2. rearrange

$$(1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n})u[z] = (\beta_0 + \beta_1 z^{-1} + \cdots + \beta_n z^{-n})e[z]$$

3. inverse z-transform both sides

$$u[k] + \alpha_1 u[k-1] + \cdots + \alpha_n u[k-n] \quad = \quad \beta_0 e[k] + \beta_1 e[k-1] + \cdots$$

Implementing difference equation

- suppose we want to implement controller $u[k] = u[k-1] + e[k]$
- loop the following code

```
read y from A/D
     e = r - y;
     u = u_old + e;
output u from D/A
     u_old = u;
return to ‘read’ after T seconds from last ‘read’
```

- **note:** we always output new value for $u[k]$ as soon as it is available

Example: PID controller

$$C_{\mathrm{d}}[z] = \frac{u[z]}{e[z]} = \frac{\beta_0 z^2 + \beta_1 z + \beta_2}{z^2 - 1}$$

divide top and bottom by z^2

$$\frac{u[z]}{e[z]} = \frac{\beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2}}{1 - z^{-2}}$$

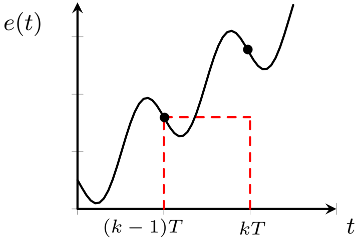
multiply through and invert term-by-term

$$u[k] - u[k-2] = \beta_0 e[k] + \beta_1 e[k-1] + \beta_2 e[k-2]$$

or

$$u[k] = u[k-2] + \beta_0 e[k] + \beta_1 e[k-1] + \beta_2 e[k-2]$$

The left-side discretization

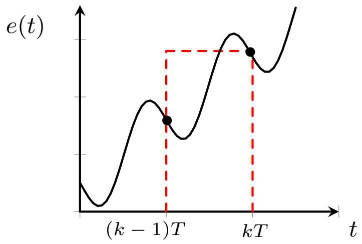


$$\int_{(k-1)T}^{kT} e(\tau)\,\mathrm{d}\tau \approx Te[k-1]$$

- proceeding as before we get the discretization rule **(exercise)**

$$s = \frac{z-1}{T}$$

The right-side discretization



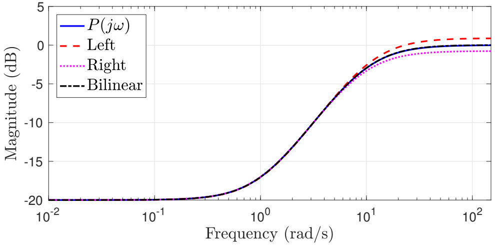
$$\int_{(k-1)T}^{kT} e(\tau)\,\mathrm{d}\tau \approx Te[k]$$

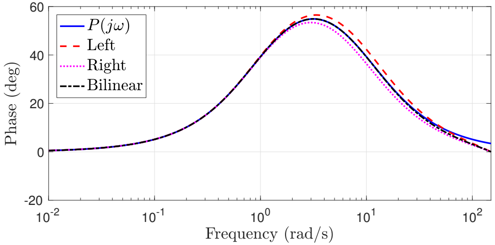
- proceeding as before we get the discretization rule **(exercise)**

$$s = \frac{z-1}{Tz}$$

Example: comparison of controller discretizations

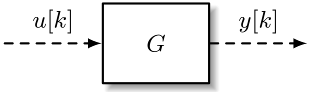
- discretization of lead compensator $(s+1)/(s+10)$, $\omega_s = 300$ rad/s





Aside: discrete-time stability

- a discrete-time signal $y[k]$ is bounded if there exists $M \geq 0$ such that $|y[k]| \leq M$ for all $k = 0, 1, 2, \ldots$



- **BIBO stability:** every bounded $u[k]$ produces a bounded $y[k]$

- G is BIBO stable if and only if every pole of transfer function $G[z]$ belongs to

$$\mathbb{D} = \{z \in \mathbb{C} \mid |z| < 1\} \qquad \text{“unit disk in z-plane”}$$

i.e., every pole has *magnitude* less than one

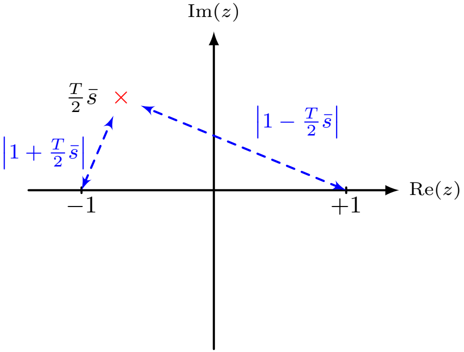
Stability of discretized controllers

- any good discretization should be such that as $T \to 0$, the approximation becomes perfect; bilinear, left-side, and right-side all satisfy this
- how else should we compare/contrast discretization schemes?
- can ask: does discretization preserves stability *of the controller*?
- of our three schemes, only bilinear does so!
- proof:

$$s = \frac{2}{T}\frac{z-1}{z+1} \qquad \Longleftrightarrow \qquad z = \frac{1+\frac{T}{2}s}{1-\frac{T}{2}s}$$

Bilinear discretization preserves BIBO stability

- suppose $C(s)$ has a pole at $\bar{s} \in \mathbb{C}_-$
- then $C_{\mathrm{d}}[z]$ must have a pole at $\bar{z} = \frac{1+\frac{T}{2}\bar{s}}{1-\frac{T}{2}\bar{s}}$

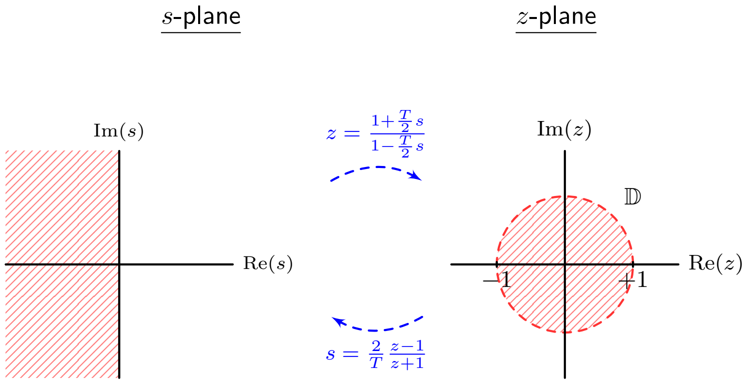


$\mathrm{Im}(z)$
$\frac{T}{2}\bar{s}$
$\left|1 + \frac{T}{2}\bar{s}\right|$
$\left|1 - \frac{T}{2}\bar{s}\right|$
-1
$+1$
$\mathrm{Re}(z)$

$$|\bar{z}| = \frac{\left|1 + \frac{T}{2}\bar{s}\right|}{\left|1 - \frac{T}{2}\bar{s}\right|} < 1$$

therefore, $\bar{z} \in \mathbb{D}$
(converse also true)

Summary of bilinear discretization



$C(s)$ BIBO stable $\quad\Longleftrightarrow\quad$ $C_\mathrm{d}[z]$ BIBO stable

Final comments on stability

- the previous stability results refer to stability of the *controller*, and **not feedback stability of the sampled-data system**
- feedback stability of sampled-data control system determined by the ratio of the sampling frequency ω_{s} to the *bandwidth of the closed-loop continuous-time control system* ω_{bw}
- rule of thumb: for best performance, choose $\omega_{\mathrm{s}} = \frac{2\pi}{T}$ to be **25 times** the bandwidth of the closed-loop continuous-time system
- for sample rates slower than 20 times the closed-loop bandwidth, consider instead using direct digital design.

Design procedure for controller emulation

1. design – *by any method* – a controller $C(s)$ for $P(s)$ to meet design specifications

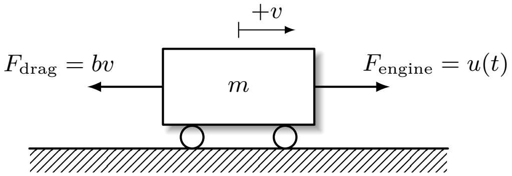
2. select a sampling period and emulate $C(s)$ to obtain $C_{\rm d}[z]$

3. simulate the closed-loop **sampled data-system**, and adjust design or sampling period as needed

4. implement $C_{\rm d}[z]$ as a difference equation

- once again, rule of thumb: $\omega_{\rm s} = \frac{2\pi}{T} \geq 25\omega_{\rm bw}$

Design example: cruise control

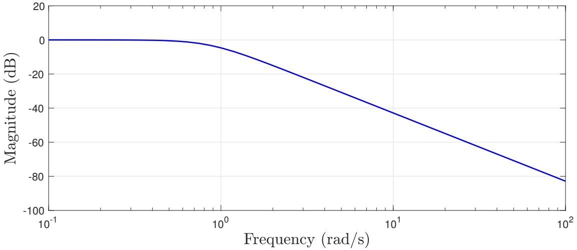


- objective: track constant reference velocity commands
- design specs: 5% overshoot, 7s settling time for 0 to 100 km/h
- $m = 1200\,\text{kg}$, $b = 10000\,\text{Ns/m}$

$$P(s) = \frac{(8.33 \times 10^{-7})}{s + (8.33 \times 10^{-3})}\,, \qquad C(s) = \frac{1}{s}\frac{(8.6 \times 10^{5})s + (5.2 \times 10^{3})}{s + 1.2}$$

Design example: cruise control contd.

- check bandwidth of closed-loop transfer function $T(s) = y(s)/r(s)$



$$\omega_{\text{bw}} \approx 1.2\,\text{rad/s} \qquad \Longrightarrow \qquad \omega_{\text{s}} \geq 25(1.2) = 30\,\text{rad/s}$$

$$\Longrightarrow \qquad T \leq \frac{2\pi}{30} \approx 0.2\,\text{s}$$

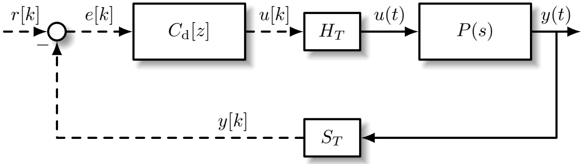
Design example: cruise control contd.

- discretize controller using bilinear transform with $T = 0.2$s

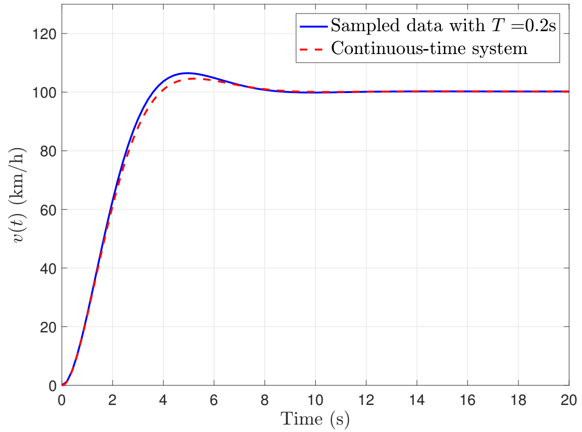
```
T = 0.2;
C_d = c2d(C,T,‘tustin’)
```

$$C_{\mathrm{d}}[z] = \frac{(7.742 \times 10^4)z^2 + (154.3)z - (7.727 \times 10^4)}{z^2 - 1.785z + 0.7854}$$

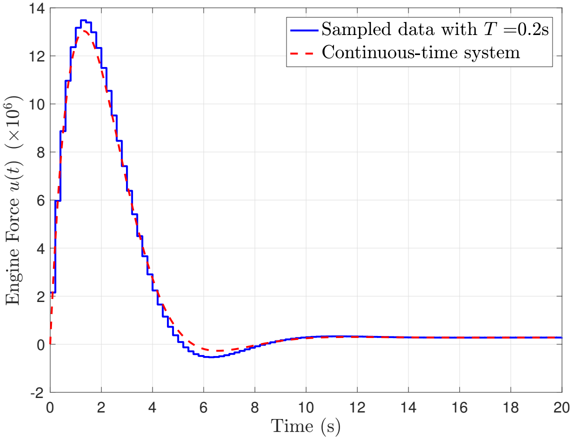
- simulate **sampled-data system**



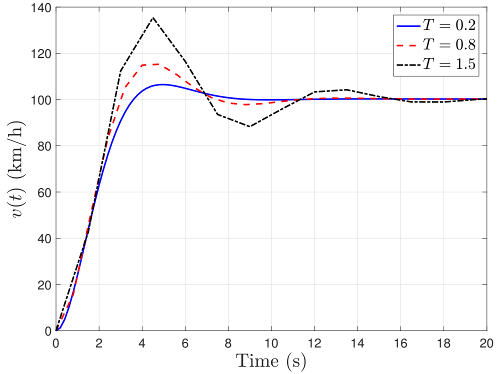
Design example: cruise control contd.



Design example: cruise control contd.



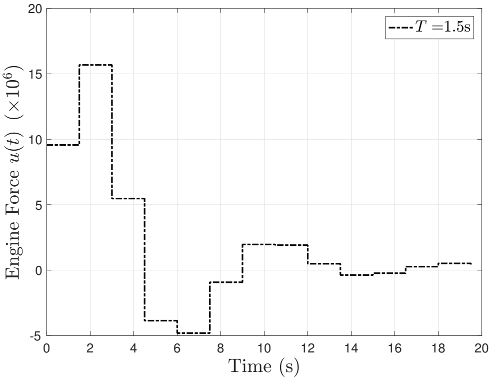
Design example: cruise control contd.



- system becomes **unstable** for $T \geq 4$
- performance getting worse with slower sampling rates – why?

Design example: cruise control contd.

- control input for $T = 1.5$s sampling rate

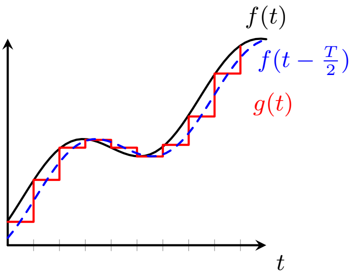


Effect of sample-and-hold blocks

- consider the sample-hold combination

$$\begin{tikzcd} {} \arrow[r, "f(t)"] & \boxed{S_T} \arrow[r, dashed] & \boxed{H_T} \arrow[r, "g(t)"] & {} \end{tikzcd}$$

- if we ignore the effects of sample-and-hold, then $g(t) \approx f(t)$



better approximation is

$$g(t) \approx f\left(t - \frac{T}{2}\right) \quad \text{(time delay!)}$$

"Proof" of $T/2$ delay

- **exercise:** sample-and-hold $H_T S_T$ has impulse response

$$g(t) = \begin{cases} \frac{1}{T} & \text{if } 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases}$$

- **exercise:** compute the transfer function $G(s)$

$$\begin{aligned} G(s) = \mathscr{L}\{g(t)\} &= \frac{1 - e^{-sT}}{sT} \\ &= \frac{-\sum_{k=1}^{\infty} (-sT)^k / k!}{sT} \\ &= 1 - sT/2 + \cdots \\ &\approx e^{-\frac{T}{2}s} \end{aligned}$$

Approximating sample-and-hold with delay

- to account for sample-and-hold effects, we can lump in this time delay with the plant, and design for the augmented plant

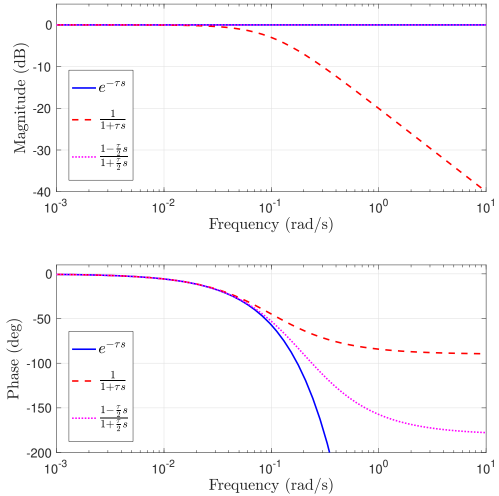
$$P_{\mathrm{aug}}(s) = e^{-\frac{T}{2}s} P(s)\,.$$

- if computational delay T_{comp} is significant, we can also include that in the augmented plant as $P_{\mathrm{aug}}(s) = e^{-\left(\frac{T}{2}+T_{\mathrm{comp}}\right)s} P(s)$

- to obtain rational $P_{\mathrm{aug}}(s)$, formulas for approximating delay:

$$e^{-\frac{T}{2}s} \approx \frac{1}{1+\frac{T}{2}s}\,, \qquad e^{-\frac{T}{2}s} \approx \frac{1-\frac{T}{4}s}{1+\frac{T}{4}s}\,, \qquad e^{-\frac{T}{2}s} \approx \frac{1}{\left(1+\frac{1}{n}\frac{T}{2}s\right)^n}$$

Comparison of delay approximations for $\tau = 10$ s



- both approximations are pretty good up to $\omega \approx 1/\tau$

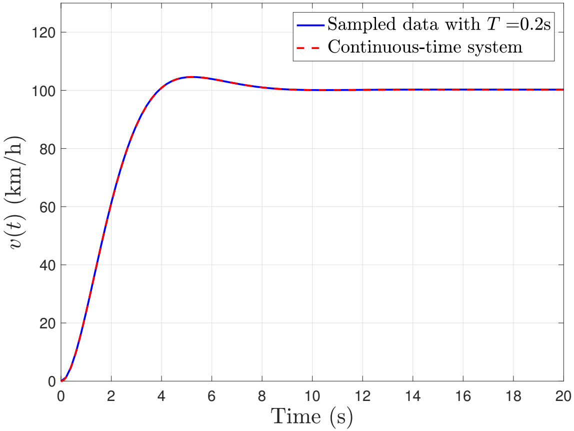
Modified design procedure for emulation

1. select sampling period T and use rational approximation of $e^{-\frac{T}{2}s}$
2. build augmented plant model $P_{\text{aug}}(s)$
3. design a controller $C(s)$ for augmented plant $P_{\text{aug}}(s)$ to meet design specifications
4. emulate $C(s)$ to obtain $C_{\text{d}}[z]$ with sampling period T
5. check performance by simulating the **sampled data-system**
6. implement $C_{\text{d}}[z]$ as a difference equation

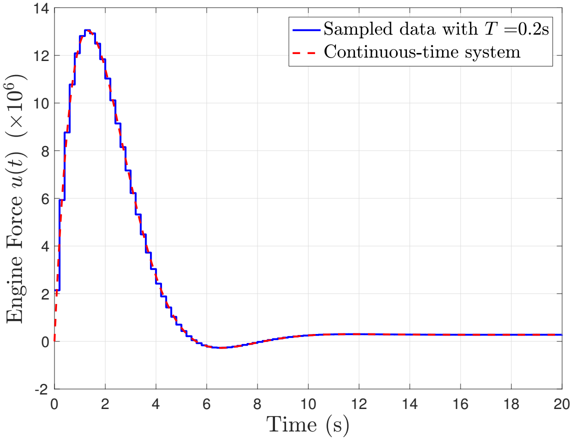
Design example: cruise control contd.

- with $T = 0.2$ s, apply modified design procedure to augmented plant

$$P_{\text{aug}}(s) = \frac{(8.33 \times 10^{-7})}{s + (8.33 \times 10^{-3})} \frac{1}{1 + \frac{T}{2}s}$$

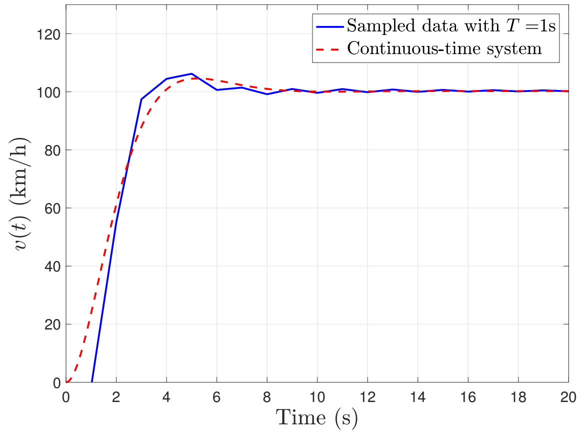


Design example: cruise control contd.

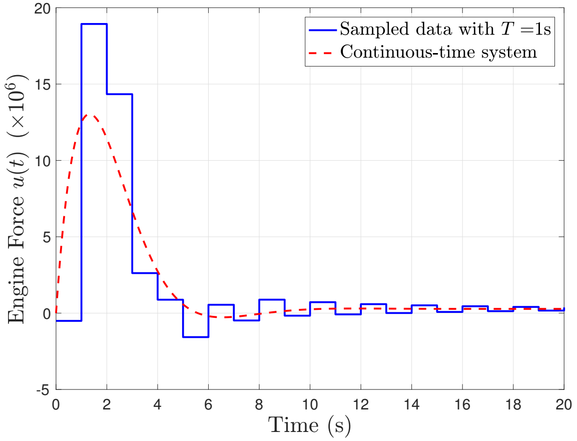


Design example: cruise control contd.

- can use modified design procedure to get decent performance with larger sampling periods



Design example: cruise control contd.



MATLAB commands

- discretizing a continuous-time controller

```
s = tf('s');
C = (s+2)/(s+1);
T = 0.01;
C_d = c2d(C,T,'tustin')
```

- `Simulink` code for sampled-data system on LEARN

Additional references

- Nielsen, Chapter 6
- Åström & Wittenmark, Chapter 8
- Franklin, Powell, & Emami-Naeini, Chapter 8.3
- Franklin, Powell, & Workman, Chapter 6

Personal Notes

Personal Notes

Personal Notes

4. Pole placement for continuous-time systems

- pole placement
- pole placement and reference tracking

Motivation for pole placement designs

- suppose we need to stabilize the following plant

$$P(s) = \frac{s^2 - 1}{s^4 - s^2 - 1}$$

- how would you design a stabilizing controller? let's try PID

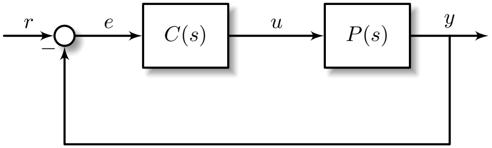
$$C(s) = K_{\mathrm{p}} + K_{\mathrm{d}} s + \frac{1}{s} K_{\mathrm{i}} = \frac{K_{\mathrm{d}} s^2 + K_{\mathrm{p}} s + K_{\mathrm{i}}}{s}$$

- characteristic polynomial is

$$\Pi(s) = s^5 + K_{\mathrm{d}} s^4 + (K_{\mathrm{p}} - 1)s^3 + (K_{\mathrm{i}} - K_{\mathrm{d}})s^2 - (1 + K_{\mathrm{p}})s - K_{\mathrm{i}}$$

- there is no choice of gains which makes all the coefficients positive
- system *cannot* be stabilized by PID!

Pole placement



$$P(s) = \frac{N_{\mathrm{p}}(s)}{D_{\mathrm{p}}(s)}$$

$$C(s) = \frac{N_{\mathrm{c}}(s)}{D_{\mathrm{c}}(s)}$$

- feedback stability of closed-loop system determined by roots of

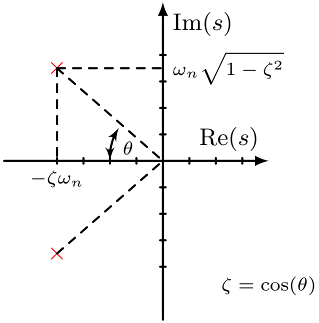
$$\Pi(s) = N_{\mathrm{p}}(s)N_{\mathrm{c}}(s) + D_{\mathrm{p}}(s)D_{\mathrm{c}}(s)$$

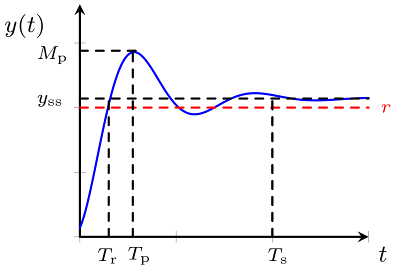
- usually have step response specs on T_{settling}, %OS, tracking, ...
- **idea:** convert performance specs into "good region" of $\mathbb{C}_-$, then design $C(s)$ so that all closed-loop poles are in this region

Converting performance specs into pole locations

- **first step**: relate pole locations to response for second-order system

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \qquad 0 < \zeta < 1$$

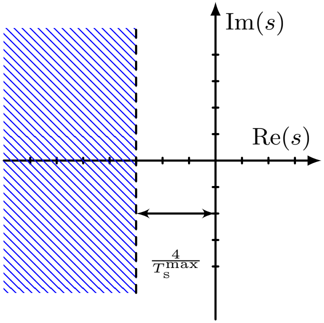




Settling time spec

- 2% settling time for second-order system is approximately

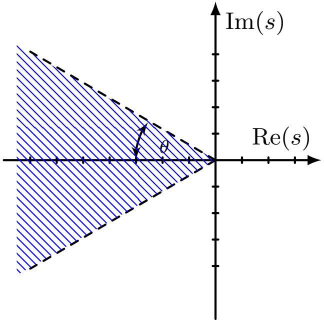
$$T_{\mathrm{s}} \simeq \frac{4}{\zeta\omega_n} \qquad \Longrightarrow \qquad T_{\mathrm{s}} \leq T_{\mathrm{s}}^{\max} \quad \text{if} \quad \zeta\omega_n \geq \frac{4}{T_{\mathrm{s}}^{\max}}$$



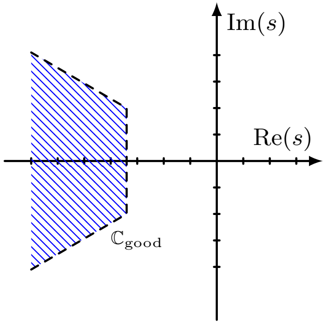
Overshoot spec

- percentage overshoot for a second-order system is

$$\%\mathsf{OS} = \exp\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right) \leq \%\mathsf{OS}^{\max} \quad \Longrightarrow \quad \zeta \geq \frac{-\ln(\%\mathsf{OS}^{\max})}{\sqrt{\pi^2 + \ln(\%\mathsf{OS}^{\max})^2}}$$



Settling time and overshoot specs



- think of $\mathbb{C}_{\text{good}}$ as a <u>*very rough*</u> *first guess* for where to place poles
- we will see shortly that this guess often needs adjustment

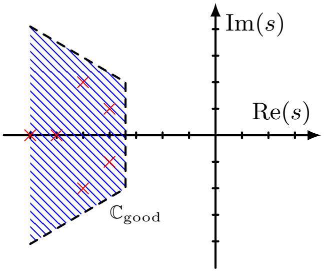
The pole-placement design problem

- suppose we have a strictly proper plant transfer function

$$P(s) = \frac{N_{\mathrm{p}}(s)}{D_{\mathrm{p}}(s)} = \frac{b_m s^m + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

and $k > 0$ desired closed-loop poles, *symmetric* w.r.t. the real axis

$$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\} \subset \mathbb{C}_{\mathrm{good}}$$



The pole-placement design problem contd.

- the *pole placement problem (P.P.P.)* is to find a controller $C(s)$ such that the roots of the closed-loop characteristic polynomial are exactly the poles specified by $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$
- **fact:** if $N_{\rm p}(s)$ and $D_{\rm p}(s)$ are coprime, the P.P.P. is solvable
  - for proof details, look up Sylvester matrix and diophantine equations
- **question:** how complicated does our controller need to be to freely place k poles?

The pole-placement design problem contd.

- if $C(s)$ is chosen to have order $n-1$, P.P.P. has *unique solution*

$$C(s) = \frac{g_{n-1}s^{n-1} + \cdots + g_1 s + g_0}{f_{n-1}s^{n-1} + \cdots f_1 s + f_0}$$

- with this choice, $\Pi(s)$ is a polynomial of order $2n-1$
- need to choose $2n-1$ poles for set Λ, obtain *desired polynomial*

$$\Pi_{\mathrm{des}}(s) = (s-\lambda_1)(s-\lambda_2)\cdots(s-\lambda_{2n-1})$$

- **note:** due to symmetry of pole choices Λ, coefficients of $\Pi_{\mathrm{des}}(s)$ will be real, as poles with non-zero imaginary part will appear as complex conjugate pairs

Example: first-order plant

$$P(s) = \frac{1}{s-1}$$

- since $P(s)$ has order $n = 1$, take $C(s)$ of order zero

$$C(s) = g_0$$

- choose $2n - 1 = 1$ poles based on specs, compute desired polynomial

$$\Lambda = \{\lambda_1\}\,, \qquad \Pi_{\mathrm{des}}(s) = (s - \lambda_1)$$

- characteristic polynomial of closed-loop system

$$\Pi(s) = s - 1 + g_0\,.$$

- set $\Pi(s) = \Pi_{\mathrm{des}}(s)$ and equate powers of $s \implies g_0 = 1 - \lambda_1$

Example: second-order plant

$$P(s) = \frac{s+1}{s(s-1)}$$

- plant is second order, so take

$$C(s) = \frac{g_1 s + g_0}{f_1 s + f_0}$$

- we need $2n - 1 = 3$ poles. for simplicity here, let's choose

$$\Lambda = \{-3, -4, -5\} \quad \Longrightarrow \quad \Pi_{\text{des}}(s) = s^3 + 12s^2 + 47s + 60$$

- characteristic polynomial is

$$\begin{aligned}\Pi(s) &= (s+1)(g_1 s + g_0) + s(s-1)(f_1 s + f_0) \\ &= f_1 s^3 + (f_0 - f_1 + g_1)s^2 + (-f_0 + g_1 + g_0)s + g_0\end{aligned}$$

Example: second-order plant contd

- set $\Pi(s) = \Pi_{\text{des}}(s)$ and equate powers of s

$$\begin{aligned} f_1 &= 1 \\ f_0 - f_1 + g_1 &= 12 \\ -f_0 + g_1 + g_0 &= 47 \\ g_0 &= 60 \end{aligned} \quad \Longrightarrow \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_0 \\ g_1 \\ g_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 12 \\ 47 \\ 60 \end{bmatrix}$$

and solve for solution

$$\begin{bmatrix} f_1 \\ f_0 \\ g_1 \\ g_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 13 \\ 0 \\ 60 \end{bmatrix} \qquad \Longrightarrow \qquad C(s) = \frac{60}{s + 13}$$

Example: general second-order plant

- for a general plant of second order

$$P(s) = \frac{b_2 s^2 + b_1 s + b_0}{a_2 s^2 + a_1 s + a_0}\,, \quad C(s) = \frac{g_1 s + g_0}{f_1 s + f_0}$$

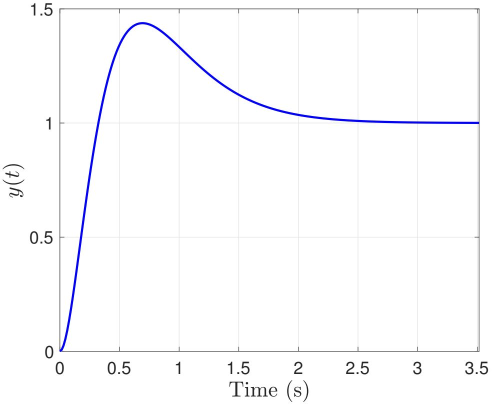
and a desired characteristic polynomial

$$\Pi_{\mathrm{des}}(s) = s^3 + c_2 s^2 + c_1 s + c_0$$

the same procedure yields (**exercise**)

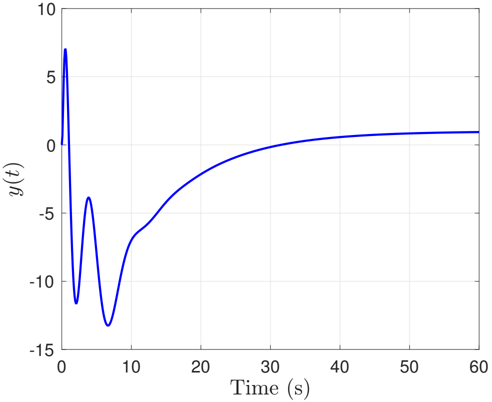
$$\begin{bmatrix} a_2 & 0 & b_2 & 0 \\ a_1 & a_2 & b_1 & b_2 \\ a_0 & a_1 & b_0 & b_1 \\ 0 & a_0 & 0 & b_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_0 \\ g_1 \\ g_0 \end{bmatrix} = \begin{bmatrix} 1 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}$$

Example: second-order plant contd.



(note: response is not very good, would want to tune further)

Example: fourth-order plant from earlier



(note: response is unacceptably bad! What is going on?)

Pole placement and closed-loop zeros

- recall closed-loop transfer function from $r(s)$ to $y(s)$

$$\frac{y(s)}{r(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} = \frac{N_{\mathrm{p}}N_{\mathrm{c}}}{N_{\mathrm{p}}N_{\mathrm{c}} + D_{\mathrm{p}}D_{\mathrm{c}}}$$

- closed-loop TF shares zeros with plant **and** controller
- we specify poles (roots of denominator); cannot directly specify zeros (roots of numerator)!
- slow or unstable zeros lead to bad closed-loop performance

Tips for tuning pole placement designs

- no definitive recipe (mix of "art and science")
- don't choose *all* closed-loop poles far away from plant poles
  - **intuition:** moving poles requires effort!
  - **example:** if plant has poles at $-a$ and $-b$, start by trying to place poles at $-1.1a$, $-1.1b$, and $\approx -20a$, then iterate
- sometimes useful to *cancel* stable zeros by forcing controller to have appropriate poles
  - **example:**

$$P(s) = \frac{s+1}{(s+10)^3}\,, \qquad C(s) = \frac{g_2 s^2 + g_1 s + g_0}{(s+1)(f_1 s + f_0)}$$

Comments on pole placement

- easy to implement – just need to solve a linear equation
- MATLAB function $\mathtt{C(s)} = \mathtt{pp(P(s)}, \Lambda)$ on course website
- **limitation:** cannot specify *zeros* of closed-loop transfer functions, can lead to poor bandwidth or high sensitivity to disturbances
- **always** simulate pole-placement designs, then adjust pole locations to obtain a good response
- **common exam mistake:** do not conflate pole placement with the emulation approach; these are independent concepts

Pole placement and reference tracking

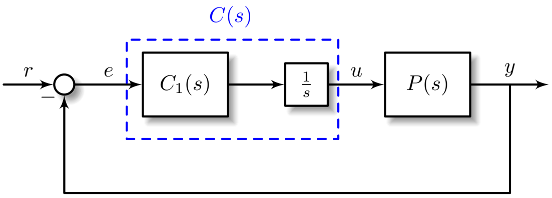
- want to track step reference with zero error (integral control)

- from previous discussion on tracking, there are three cases

  (i) if $P(s)$ has a zero at $s = 0$, step tracking is not possible

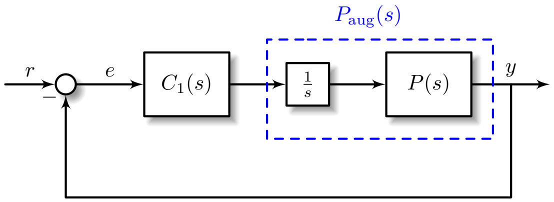
  (ii) if $P(s)$ has a pole at $s = 0$, we just need to stabilize the feedback loop (e.g., use pole placement as above)

  (iii) otherwise, need to include pole at $s = 0$ in controller: for example, let $C(s) = \frac{1}{s}C_1(s)$



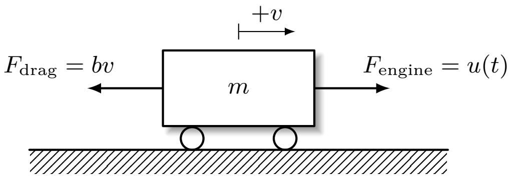
Pole placement and reference tracking

- for design, we can group the $\frac{1}{s}$ block with $P(s)$



- now just design $C_1(s)$ for $P_{\text{aug}}(s)$ using normal pole placement
- final controller given by $C(s) = \frac{1}{s}C_1(s)$ – order of controller?
- similar procedure for other ref. signals (internal model principle)

Example: cruise control



- objective: track constant reference velocity commands
- design specs: $\leq 1\%$ overshoot, 7s settling time for 0 to 100 km/h
- $m = 1200$ kg, $b = 10000$ Ns/m

$$m\dot{v} = -bv + u \qquad \Longrightarrow \qquad P(s) = \frac{1/m}{s + b/m}$$

- single pole at $s = -0.0083$ rad/s (i.e., $\tau \approx 120$s)

Example: cruise control contd.

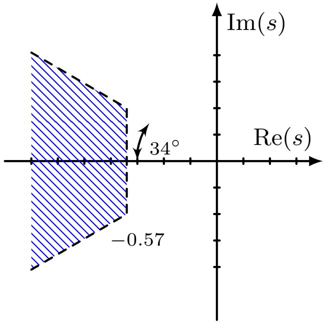
- augment plant with integrator

$$P_{\mathrm{aug}}(s) = \frac{1}{s}\frac{1/m}{s + b/m}$$

- overshoot and settling time specs yield $\mathbb{C}_{\mathrm{good}}$

$$\cos(\theta) \geq \frac{-\ln(0.05)}{\sqrt{\pi^2 + \ln(0.05)^2}} = 0.82$$

$$\xi\omega_n \geq \frac{4}{7\mathsf{s}} = 0.57\,\mathsf{s}^{-1}$$



Example: cruise control contd.

- must choose $n + (n-1) = (2) + (1) = 3$ poles; first attempt

$$\Lambda_{\text{first}} = \{-0.009, -0.7, -0.7\}$$

to obtain corresponding desired polynomial

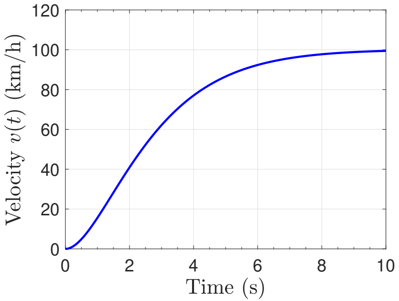
$$\Pi_{\text{des}}(s) = (s + 0.009)(s + 0.7)(s + 0.7) = s^3 + c_2 s^2 + c_1 s + c_0$$

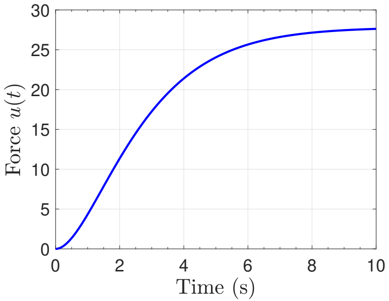
- solve pole placement equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ b/m & 1 & 0 & 0 \\ 0 & b/m & 1/m & 0 \\ 0 & 0 & 0 & 1/m \end{bmatrix} \begin{bmatrix} f_1 \\ f_0 \\ g_1 \\ g_0 \end{bmatrix} = \begin{bmatrix} 1 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}$$

Example: cruise control contd.

$$\Lambda_{\text{first}} = \{-0.009, -0.7, -0.7\}$$



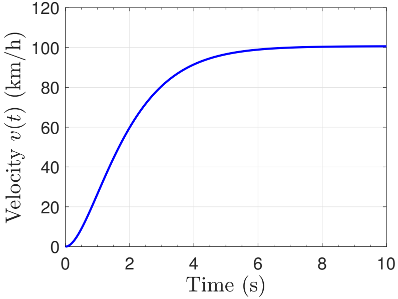


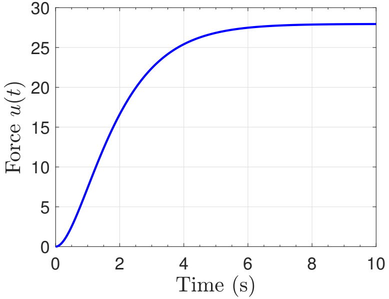
- almost meeting specs, would like response a bit faster

Example: cruise control contd.

- try slightly modified poles

$$\Lambda_{\text{second}} = \{-0.012, -1, -1\}$$

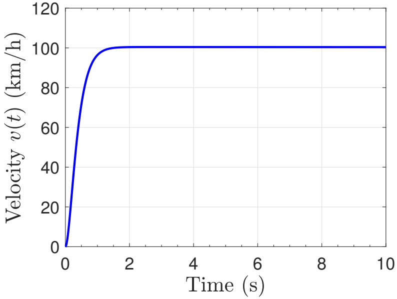


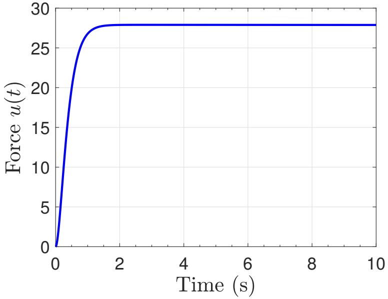


- looks great, but just for fun, can we go even faster?

Example: cruise control contd.

- try even faster poles $\Lambda_{\text{third}} = \{-0.02, -5, -5\}$





- **note**: driver would pull about 3 g's in this car :)

- **question:** pole at $s = -0.02$ should lead to a response with a time constant of $\tau \approx 50$s . . . why don't we see a slower response then? To find the answer, go explore the `MATLAB` code . . .

MATLAB commands

- simulate system response

```
[r,t] = gensig('sin',2*pi);
y = lsim(G,r,t);
```

- calculate pole placement controller (code on LEARN)

```
P = (s+1)/(s^2+3*s+2);
poles = [-3,-4,-5];
C = pp(P,poles);
```

- connecting systems with named inputs/outputs

```
C = pid(2,1);  C.u = 'e';   C.y = 'u';
P.u = 'u'; P.y = 'y';
Sum = sumblk('e = r - y');
T = connect(G,C,Sum,'r','y');
```

Additional references

- pole placement
  - Nielsen, Chapter 4
  - Åström & Wittenmark, Chapter 10
  - Iglesias, John's Hopkins ECE 484, Chapter 4

Personal Notes

Personal Notes

Personal Notes

5. Continuous-time LTI control systems

- models of continuous-time LTI systems
- LTI state-space models
- solutions and stability of state-space models
- transfer functions from state models
- nonlinear systems and linearization

equivalent models of SISO LTI systems

- linear, constant-coefficient differential equations

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \omega_n^2 u$$

- transfer functions

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

- impulse response (for $0 < \zeta < 1$)

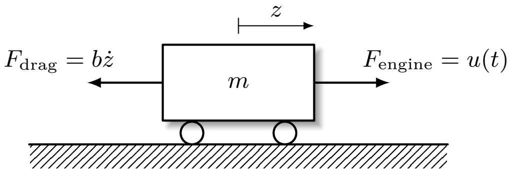
$$g(t) = \frac{\omega_n}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin\left(\sqrt{1-\zeta^2}\omega_n t\right) \mathbb{1}(t)$$

- **state-space models**

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

$$y = Cx + Du$$

Example: cart with air resistance



differential equation: $m\ddot{z} = -b\dot{z} + u$

transfer function from $u(s)$ to $z(s)$: $$G(s) = \frac{1}{ms^2 + bs}$$

impulse response: $$g(t) = \mathscr{L}^{-1}\{G(s)\} = \frac{1}{b}\left(1 - e^{-bt/m}\right)\mathbb{1}(t)$$

Example: cart with air resistance contd.

- differential equation: $m\ddot{z} = -b\dot{z} + u$

- for state model, introduce two "states"

$$x_1 = z\,, \qquad x_2 = \dot{z} \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- take derivatives to find "state equations"

$$\begin{aligned} \dot{x}_1 &= \dot{z} \\ &= x_2 \end{aligned} \qquad \begin{aligned} \dot{x}_2 = \ddot{z} &= -\frac{b}{m}\dot{z} + \frac{1}{m}u \\ &= -\frac{b}{m}x_2 + \frac{1}{m}u \end{aligned}$$

- write down "output/measurement equation"

$$y = x_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Example: cart with air resistance contd.

In matrix form, the equations are

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1\\ 0 & -b/m\end{bmatrix}}_{=A}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ 1/m\end{bmatrix}}_{=B}u$$

$$y = \underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{=C}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \underbrace{[0]}_{=D}u$$

- LTI state model completely specified by (A, B, C, D)

LTI state-space models

a continuous time LTI state-space model has the form

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

- $x(t) \in \mathbb{R}^n$ is the *state vector*
- $n \in \mathbb{N}$ is the *order*
- $u(t) \in \mathbb{R}^m$ is the *input vector*
- $y(t) \in \mathbb{R}^p$ is the *output vector*

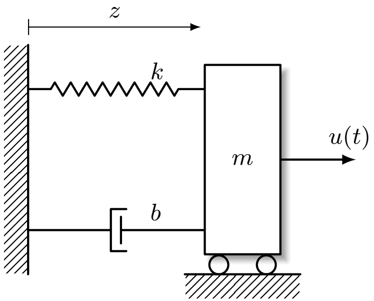
- $A \in \mathbb{R}^{n\times n}$
- $B \in \mathbb{R}^{n\times m}$
- $C \in \mathbb{R}^{p\times n}$
- $D \in \mathbb{R}^{p\times m}$

We focus on *single-input single-output* (SISO) systems: $m = p = 1$

Why study state-space models?

- abstraction allows for systematic analysis and design
- extends to multi-input multi-output systems
- extends to nonlinear systems, stochastic systems, PDEs . . .
- (almost) all advanced control design is based on state models
  - linear quadratic control (1960s–)
  - robust control (1980s–)
  - receding horizon / model predictive control (1990s–)

Example: mass spring damper



$$m\ddot{z} + b\dot{z} + kz = u$$

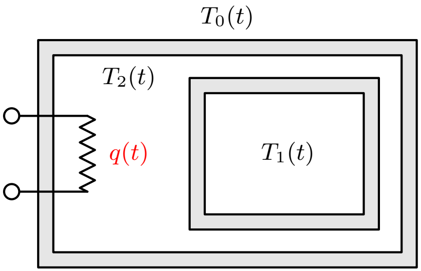
choose states

$$x_1 = z\,, \quad x_2 = \dot{z}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

Example: thermal control system



T_0	external temperature
T_1, T_2	temperatures of volumes
q	heat flux into outer volume
m_1, m_2	masses of volumes
c_1, c_2	heat capacities
g_{ij}	thermal conductances

$$m_1 c_1 \dot{T}_1 = g_{12}\,(T_2 - T_1)$$
$$m_2 c_2 \dot{T}_2 = -g_{12}\,(T_2 - T_1) - g_{20}\,(T_2 - T_0) + q(t)$$

Example: thermal control system

- states: $x = (x_1, x_2) = (T_1, T_2)$
- output: $y = T_1$
- inputs: $u = (T_0, q)$ (**note:** example of two input system)

$$
\begin{aligned}
m_1 c_1 \dot{T}_1 &= -g_{12} T_1 + g_{12} T_2 \\
m_2 c_2 \dot{T}_2 &= -(g_{12} + g_{20}) T_2 + g_{12} T_1 + g_{20} T_0 + q(t)
\end{aligned}
$$

$$
\begin{aligned}
\begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \end{bmatrix} &= \begin{bmatrix} -\frac{g_{12}}{m_1 c_1} & \frac{g_{12}}{m_1 c_1} \\ \frac{g_{12}}{m_2 c_2} & -\frac{g_{12}+g_{20}}{m_2 c_2} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{g_{20}}{m_2 c_2} & \frac{1}{m_2 c_2} \end{bmatrix} \begin{bmatrix} T_0 \\ q \end{bmatrix} \\
y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} T_0 \\ q \end{bmatrix}
\end{aligned}
$$

Example: PI controller

- proportional-integral controller:

$$\frac{u(s)}{e(s)} = K_{\mathrm{p}} + \frac{K_{\mathrm{i}}}{s} \quad \Longleftrightarrow \quad u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,\mathrm{d}\tau$$

- define controller state variable $x_{\mathrm{c}} \in \mathbb{R}$ to be integral of error

$$\dot{x}_{\mathrm{c}}(t) = e(t)$$

- state-space model is therefore

$$\begin{aligned} \dot{x}_{\mathrm{c}} &= [0]x + [1]e \\ u &= [K_{\mathrm{i}}]x_{\mathrm{c}} + [K_{\mathrm{p}}]e \end{aligned}$$

- "A" matrix equal to zero, "D" matrix equal to proportional gain

Example: high-order ODEs

$$y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_1 y^{(1)} + a_0 y = u$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y \\ y^{(1)} \\ \vdots \\ y^{(n-1)} \end{bmatrix}$$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= -a_{n-1}x_n - \cdots - a_0 x_1 + u \end{aligned}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix} x + [0]u$$

Example: high-order ODEs w/ input derivatives ($m < n$)

$$\begin{aligned} y^{(n)} + a_{n-1}y^{(n-1)} &+ \cdots + a_1 y^{(1)} + a_0 y \\ &= b_m u^{(m)} + b_{m-1}u^{(m-1)} + \cdots + b_0 u \end{aligned}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} b_0 & \cdots & b_m & 0 & \cdots & 0 \end{bmatrix} x + [0]u$$

(details for this example in a handout on LEARN)

Example: generalized mechanical systems

mechanical systems with k degrees of freedom undergoing small motions

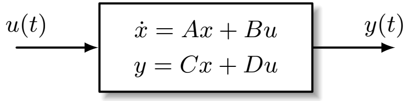
$$M\ddot{q} + D\dot{q} + Kq = \tau$$

- $q \in \mathbb{R}^k$ is the vector of generalized coordinates (positions, angles)
- $M, D, K \in \mathbb{R}^{k\times k}$ are mass, damping, stiffness **matrices**
- with state vector $x = (x_1, x_2) = (q, \dot{q})$, output $y = \dot{q}$

$$\begin{aligned}
\dot{x} &= \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \mathbb{0} & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} \mathbb{0} \\ M^{-1} \end{bmatrix} \tau \\
y &= \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix}
\end{aligned}$$

Other comments on state models

- state vector x is *internal* to the system, acts as intermediary



- for an nth order differential equation, you need n states
- there is no unique choice of state variables, but usually
  - mechanical systems: positions and velocities
  - analog circuits: capacitor voltages and inductor currents
- the condition for *equilibrium* is $\dot{x} = 0 = Ax + Bu$
- x, y, u are often *deviations* from some desired values

Solving state-space models without inputs

- consider zero input $u(t) = 0$ with initial condition $x(0) = x_0$

$$\dot{x} = Ax\,, \qquad x(0) = x_0$$

- take Laplace transforms of both sides with $x(s) = \mathscr{L}\{x(t)\}$

$$sx(s) - x_0 = Ax(s)\,, \qquad x(s) = \begin{bmatrix} x_1(s) \\ \vdots \\ x_n(s) \end{bmatrix}$$

- solving, we have that

$$x(s) = (sI - A)^{-1} x_0$$

- how do we take the inverse Laplace transform of this?

Diagonalization and powers of matrices

- diagonalization $A = V\Lambda V^{-1}$ provides a method to compute A^k
- if A is diagonalizable, then

$$\begin{aligned} A^k &= (V\Lambda V^{-1})^k \\ &= (V\Lambda V^{-1})(V\Lambda V^{-1})\cdots(V\Lambda V^{-1}) \\ &= V\Lambda^k V^{-1} \end{aligned}$$

where

$$\Lambda^k = \begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_n^k \end{bmatrix}$$

The matrix exponential

- for a scalar variable $x \in \mathbb{R}$, regular exponential

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

- for a square matrix $A \in \mathbb{R}^{n \times n}$, *matrix exponential*

$$e^A = I_n + A + \frac{A^2}{2} + \frac{A^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{A^k}{k!} \in \mathbb{R}^{n \times n}$$

- if A is diagonalizable

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \sum_{k=0}^{\infty} \frac{V \Lambda^k V^{-1}}{k!} = V \left(\sum_{k=0}^{\infty} \frac{\Lambda^k}{k!} \right) V^{-1} = V e^{\Lambda} V^{-1}$$

Laplace transform of the matrix exponential

Let t be a time variable, and consider the signal e^{At} for $t \geq 0$

$$e^{At} = Ve^{\Lambda t}V^{-1} = V \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix} V^{-1}$$

Take Laplace-transform element-by-element

$$\begin{aligned} \mathscr{L}\{e^{At}\} &= \mathscr{L}\{Ve^{\Lambda t}V^{-1}\} \\ &= V\,\mathscr{L}\{e^{\Lambda t}\}V^{-1} && \text{(by linearity)} \\ &= V(sI-\Lambda)^{-1}V^{-1} && \text{(by L.T. that } \mathscr{L}\{e^{\lambda t}\} = \tfrac{1}{s-\lambda}\text{)} \\ &= (sVV^{-1} - V\Lambda V^{-1})^{-1} \\ &= (sI-A)^{-1} \end{aligned}$$

Solution of state-space model contd.

- the Laplace domain solution was

$$x(s) = (sI - A)^{-1}x_0$$

- taking inverse Laplace transforms, we have the explicit solution

$$x(t) = \begin{cases} e^{At}x_0 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

Internal stability of state models

- state-space system $\dot{x} = Ax$ is *internally asymptotically stable* if $x(t) \to 0$ as $t \to \infty$ from any initial condition $x(0)$

- **interpretation:** with no control input, system "dissipates energy"

- suppose A is diagonalizable with $V^{-1}AV = \Lambda$
  - change of state variables $z = V^{-1}x \quad \implies \quad z(0) = V^{-1}x(0)$

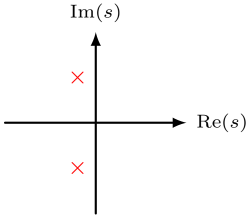
$$z(t) = V^{-1}x(t) = V^{-1}e^{At}x(0) = V^{-1}e^{At}Vz(0) = e^{\Lambda t}z(0)$$

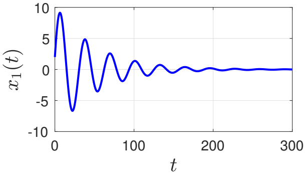
this says that $\qquad z_i(t) = e^{\lambda_i t}z_i(0)$

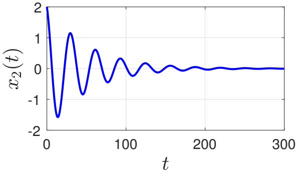
A continuous-time LTI state model is internally asymptotically stable if and only if $\lambda_i \in \mathbb{C}_-$ for all $\lambda_i \in \mathrm{eig}(A)$, i.e., all eigenvalues of A have negative real part

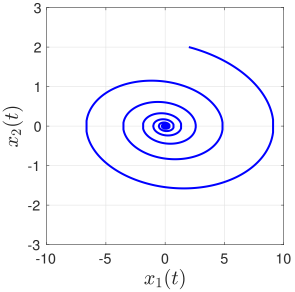
Example: second-order system with damping

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} x\,, \quad x(0) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$



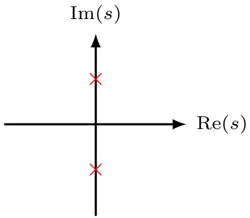


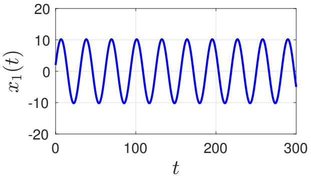


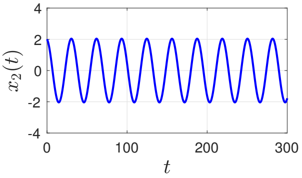


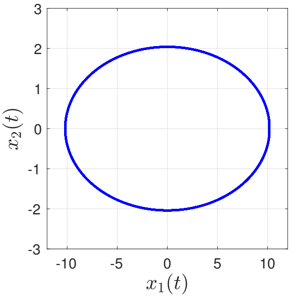
Example: second-order system with no damping

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & 0 \end{bmatrix} x\,, \quad x(0) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$



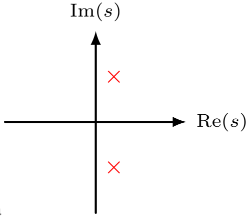


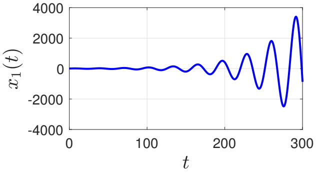


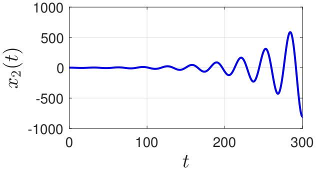


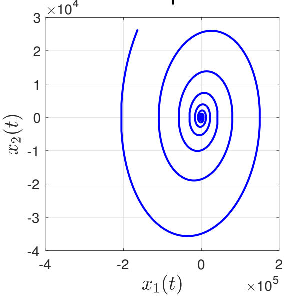
Example: second-order system with negative damping

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & 2\zeta\omega_n \end{bmatrix} x\,, \quad x(0) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$









Solution of state-space model with input

- back to our general model with inputs

$$\dot{x}(t) = Ax(t) + Bu(t)\,, \qquad x(0) = x_0$$

- take L.T. of both sides with $x(s) = \mathscr{L}\{x(t)\}$, $u(s) = \mathscr{L}\{u(t)\}$

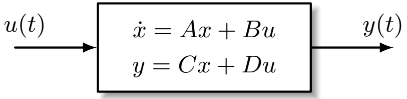
$$\begin{aligned} sx(s) - x(0) &= Ax(s) + Bu(s) \\ \implies \qquad x(s) &= (sI - A)^{-1}x(0) + (sI - A)^{-1}Bu(s) \end{aligned}$$

- therefore

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau\,, \quad t \geq 0$$

- we will use this formula for direct design of digital controllers

Transfer function from state models



- taking Laplace transforms with zero initial condition $x(0) = 0$

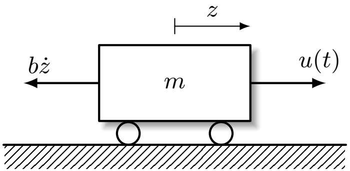
$$\begin{aligned} sx(s) &= Ax(s) + Bu(s) \\ y(s) &= Cx(s) + Du(s) \end{aligned} \quad \Longrightarrow \quad \begin{aligned} x(s) &= (sI - A)^{-1}Bu(s) \\ y(s) &= Cx(s) + Du(s) \end{aligned}$$

- eliminate $x(s)$ to obtain

$$\frac{y(s)}{u(s)} = P(s) = C(sI - A)^{-1}B + D$$

- state-space model *uniquely* determines $P(s)$

Example: cart with air resistance contd.



$$m\ddot{z} = -b\dot{z} + u$$
$$P(s) = \frac{1}{ms^2 + bs}$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -b/m\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1/m\end{bmatrix}u$$
$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$\begin{aligned}(sI - A)^{-1} &= \begin{bmatrix}s & -1\\ 0 & s + b/m\end{bmatrix}^{-1}\\ &= \frac{1}{s(s+b/m)}\begin{bmatrix}s + b/m & 1\\ 0 & s\end{bmatrix}\end{aligned}$$

$$P(s) = C(sI - A)^{-1}B = \frac{1/m}{s(s+b/m)} = \frac{1}{ms^2 + bs}$$

Internal asymptotic stability vs. BIBO stability

- we have two stability concepts: internal stability and BIBO stability
- are they related? yes.

$$\begin{aligned} P(s) &= C(sI-A)^{-1}B + D \\ &= C\,\frac{\mathrm{adj}(sI-A)}{\det(sI-A)}\,B + D \\ &= \frac{C\,\mathrm{adj}(sI-A)B + D\det(sI-A)}{\det(sI-A)} = \frac{\text{(some polynomial)}}{\Pi_A(s)} \end{aligned}$$

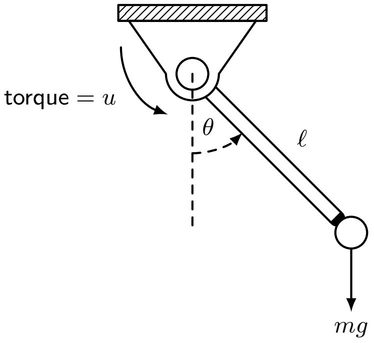
- all *poles* of $P(s)$ come from *eigenvalues* of matrix A

- if state-space system internally stable, then $P(s)$ is BIBO stable

Nonlinear systems

- All real systems are nonlinear
  - robot manipulator arms (Lagrangian dynamics)
  - drones, UAV's (aerodynamics)
  - chemical reactors (mass-action kinetics)
  - power systems (AC power flow)
  - population dynamics (Lotka-Volterra models)
  - ...
- nonlinear systems **do not** have transfer functions
- nonlinear systems **do** have state models, but they **do not** look like $\dot{x}(t) = Ax(t) + Bu(t)$

Example: state model for simple pendulum



$$m\ell^2\ddot{\theta} = -mg\ell\sin\theta - b\dot{\theta} + u$$

choose state variables

$$x_1 = \theta \qquad x_2 = \dot{\theta}$$

- state space model is given by

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{\ell}\sin x_1 - \frac{b}{m\ell^2}x_2 + \frac{1}{m\ell^2}u\end{bmatrix} \qquad \Longleftrightarrow \qquad \dot{x} = f(x,u)\,.$$

Nonlinear state models

a nonlinear state-space model has the form

$$\begin{aligned}\dot{x} &= f(x,u)\\ y &= h(x,u)\end{aligned}$$

- $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}$ is the input, $y \in \mathbb{R}$ is the output
- $f(x,u)$ is a nonlinear function which describes the dynamics
- $h(x,u)$ is a nonlinear function which describes the measurement
- if f and h are both linear in (x,u), then we have

$$\begin{aligned}\dot{x} &= Ax + Bu\\ y &= Cx + Du\end{aligned}$$

Equilibrium configurations

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

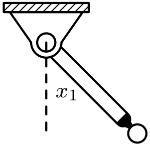
- an *equilibrium configuration* is any state/input pair $(\bar{x}, \bar{u})$ such that

$$f(\bar{x}, \bar{u}) = 0$$

- at an equilibrium configuration, $\dot{x} = 0 \quad \implies \quad x(t) = \bar{x}$ for all t
- the output is then fixed at $\bar{y} = h(\bar{x}, \bar{u})$

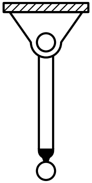
Example: simple pendulum contd.

general case



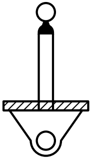
$$0 = \bar{x}_2$$
$$0 = -\frac{g}{\ell}\sin\bar{x}_1 + \frac{1}{m\ell^2}\bar{u}$$

hanging down



$$(\bar{x},\bar{u}) = \left(\begin{bmatrix}0\\0\end{bmatrix}, 0\right)$$

balancing up

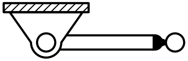


$$(\bar{x},\bar{u}) = \left(\begin{bmatrix}\pi\\0\end{bmatrix}, 0\right)$$

Example: simple pendulum contd.

- what if we wanted the angle to stay at 90°?

$$0 = -\frac{g}{\ell}\sin\frac{\pi}{2} + \frac{1}{m\ell^2}\bar{u} \qquad \Longleftrightarrow \qquad \bar{u} = mg\ell$$



$$(\bar{x}, \bar{u}) = \left(\begin{bmatrix}\pi/2 \\ 0\end{bmatrix}, mg\ell\right)$$

- torque balances gravity to create equilibrium

Linearization of nonlinear systems

- nonlinear model is usually too difficult to design with
- *linearization* around an equilibrium yields approximate linear model
- **Idea:** for small changes around equilibrium $(\bar{x}, \bar{u})$, functions $f(x, u)$ and $h(x, u)$ are well-approximated by linear functions
- linearization will be dynamic model which uses *deviation variables*

$$\begin{aligned}\delta x &= x - \bar{x} \\ \delta u &= u - \bar{u} \\ \delta y &= y - \bar{y}\end{aligned}$$

Linearization and derivative matrices

- from vector calculus, Taylor expand $f(x, u)$ around $(\bar{x}, \bar{u})$

$$f(x,u) \approx f(\bar{x},\bar{u}) + \frac{\partial f}{\partial x}(\bar{x},\bar{u}) \cdot (x - \bar{x}) + \frac{\partial f}{\partial u}(\bar{x},\bar{u}) \cdot (u - \bar{u})$$

- matrices of partial derivatives, *evaluated at equilibrium*

$$A := \frac{\partial f}{\partial x}(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad B := \frac{\partial f}{\partial u}(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \vdots \\ \frac{\partial f_n}{\partial u} \end{bmatrix}$$

- $\delta x = x - \bar{x}$ *approximately* satisfies the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}(\delta x) = \dot{x} = f(x,u) \approx f(\bar{x},\bar{u}) + A\delta x + B\delta u$$

Linearization and derivative matrices contd.

- for output $y = h(x,u)$, Taylor expand $h(x,u)$ around $(\bar{x},\bar{u})$

$$h(x,u) \approx h(\bar{x},\bar{u}) + \frac{\partial h}{\partial x}(\bar{x},\bar{u}) \cdot (x - \bar{x}) + \frac{\partial h}{\partial u}(\bar{x},\bar{u}) \cdot (u - \bar{u})$$

- matrices of partial derivatives, *evaluated at equilibrium*

$$C := \frac{\partial h}{\partial x}(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \cdots & \frac{\partial h}{\partial x_n} \end{bmatrix} \qquad D := \frac{\partial h}{\partial u}(\bar{x},\bar{u})$$

- output deviation $\delta y = y - \bar{y}$ therefore satisfies

$$\begin{aligned} \delta y = y - \bar{y} &= h(x,u) - \bar{y} \\ &\approx (h(\bar{x},\bar{u}) + C\delta x + D\delta u) - \bar{y} \\ &= C\delta x + D\delta u \end{aligned}$$

Linearization contd.

around equilibrium configuration $(\bar{x}, \bar{u})$, we have the LTI model

$$\dot{\delta x} = A\delta x + B\delta u$$
$$\delta y = C\delta x + D\delta u$$

- will be accurate as long as (x, u) stays close to $(\bar{x}, \bar{u})$
- works unbelievably well in practice (why?)
- computation of (A, B, C, D) easily automated using symbolic tools

```
syms x1 x2 u k b real
f = [x2;-k*sin(x1) - b*x2 + u];
A = subs(jacobian(f,[x1;x2]),[x1,x2,u],[0,0,0]);
```

Controlling around an equilibrium via linearization

1. use science to find nonlinear model $\dot{x} = f(x,u)$, $y = h(x,u)$

2. for your desired output $\bar{y}$, find appropriate equilibrium $(\bar{x}, \bar{u})$

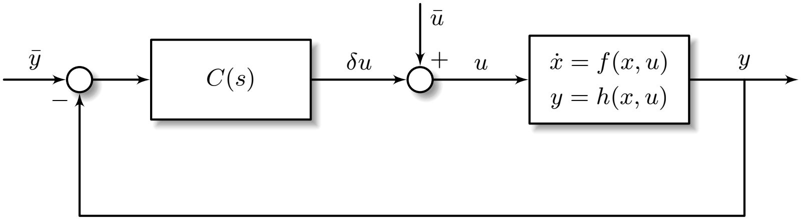
3. linearize the nonlinear model around $(\bar{x}, \bar{u})$ yielding

$$\begin{aligned} \dot{\delta x} &= A\delta x + B\delta u \\ \delta y &= C\delta x + D\delta u \end{aligned} \quad \Longrightarrow \quad P(s) = \frac{\delta y(s)}{\delta u(s)} = C(sI_n - A)^{-1}B + D$$

4. design a controller $C(s)$ for the linearized system

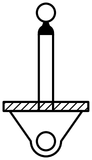
5. check performance by simulating the **nonlinear closed-loop system**

Control design based on linearization



Note: this toy example is quite important . . .

Example: inverted pendulum



$$(\bar{x}, \bar{u}) = \left(\begin{bmatrix} \pi \\ 0 \end{bmatrix}, 0 \right) \qquad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{\ell}\sin x_1 - \frac{b}{m\ell^2}x_2 + \frac{1}{m\ell^2}u \end{bmatrix}$$

$$y = x_1$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell}\cos x_1 & -\frac{b}{m\ell^2} \end{bmatrix} \quad \Longrightarrow \quad A = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{b}{m\ell^2} \end{bmatrix}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix} \quad \Longrightarrow \quad B = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$$

Example: inverted pendulum contd.

$$\frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \Longrightarrow \quad C = \frac{\partial h}{\partial x}(\bar{x}, \bar{u}) = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\frac{\partial h}{\partial u} = 0 \quad \Longrightarrow \quad D = \frac{\partial h}{\partial u}(\bar{x}, \bar{u}) = 0$$

- linearized model is therefore

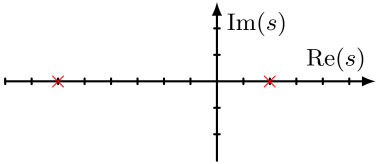
$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{b}{m\ell^2} \end{bmatrix}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}\delta u$$

$$\delta y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + [0]\delta u$$

Example: inverted pendulum contd.

- intuitively, this equilibrium is unstable. check internal stability

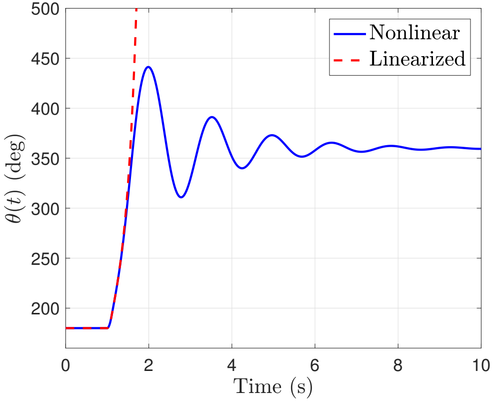
$$\det(sI - A) = \det \begin{bmatrix} s & -1 \\ -\frac{g}{\ell} & s + \frac{b}{mg\ell} \end{bmatrix} = s\left(s + \frac{b}{m\ell^2}\right) - \frac{g}{\ell}$$
$$= s^2 + \frac{b}{m\ell^2}s - \frac{g}{\ell}$$



- linearized system is internally unstable

Example inverted pendulum contd.

- we hit the pendulum with torque disturbance
- open-loop response



Example inverted pendulum contd.

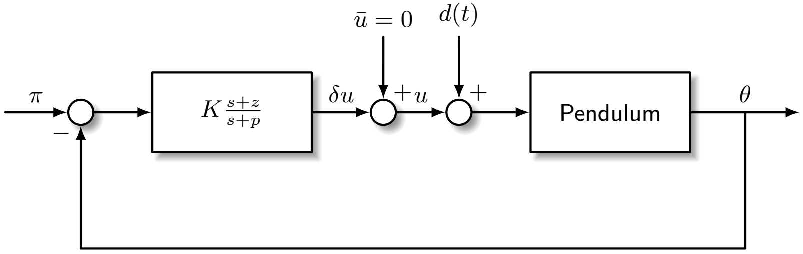
- we now design a controller to stabilize upright equilibrium point
- for now, use transfer function methods

$$P(s) = C(sI - A)^{-1}B + D = \frac{\frac{1}{m\ell^2}}{s^2 + \frac{b}{m\ell^2}s - \frac{g}{\ell}}$$

- **exercise:** design a stabilizing controller, *e.g.*,

$$C_{\mathrm{pd}}(s) = K_{\mathrm{p}} + K_{\mathrm{d}}\frac{s}{\tau s + 1} \qquad \text{or} \qquad C_{\mathrm{lead}}(s) = K\frac{s+z}{s+p}$$

Control design for inverted pendulum

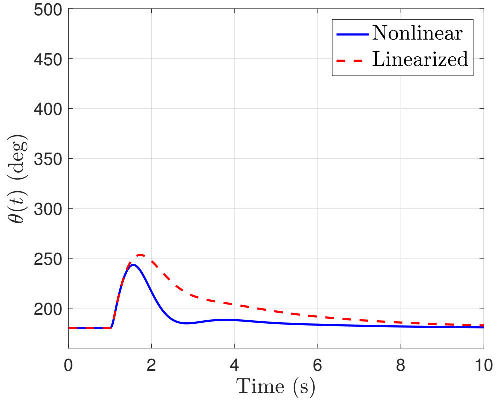


- let's simulate two different closed-loop systems

1. feedback connection of controller and linearized pendulum model
   - this is the model we used for design
2. feedback connection of controller and nonlinear pendulum model
   - this is the "real" system

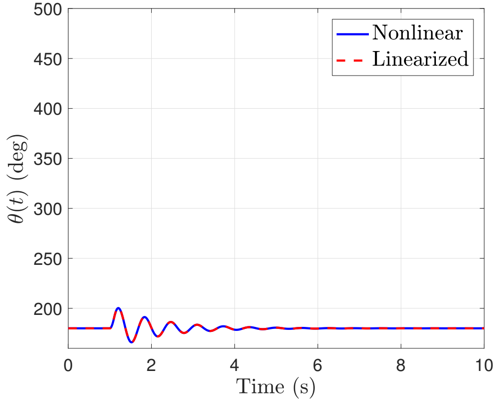
Example inverted pendulum contd.

- lead controller $C(s) = K\frac{s+2}{s+2.5}$
- pendulum subject to square pulse disturbance
- $K = 0.65$



Example inverted pendulum contd.

- lead controller $C(s) = K\frac{s+2}{s+2.5}$
- pendulum subject to square pulse disturbance
- $K = 3$



Final remarks on linearization-based control

- if you change equilibrium configurations, you must recompute the matrices (A, B, C, D) which define the LTI model
- if the (A, B, C, D) matrices are not constant, then something is wrong in your derivation; the matrices should not depend on x or u
- linearization-based control works very well if (x, u) stays close to equilibrium configuration $(\bar{x}, \bar{u})$ – how 'close' you must stay is application dependent

MATLAB commands

- compute eigenpairs of matrix

```
A = [1,0,0;0,2,3;5,-5,1];
[V,Lambda] = eig(A);
```

- define state-space model

```
my_ss = ss(A,B,C,D);
```

- you can operate on state-space models like TF models, *i.e.*

```
bode(my_ss);
step(my_ss);
```

- if $G(s)$ is a TF model, `MATLAB` will construct state-space realization

```
G = tf([1,0,1],[2,0,4]);
my_ss = ss(G);
```

Additional references

- LTI continuous-time state-space models
  - Nielsen, Chapter 2
  - Åström & Murray, Chapter 5
  - Franklin, Powell, & Emami-Naeini, Chapter 7
  - Hespanha, Topics in Undergraduate Control System Design, Chap. 8
  - Boyd & Lall, EE263 Course Notes, Chapters 24, 25, 28
- nonlinear systems and linearization
  - Nielsen, Chapter 2
  - Åström & Wittenmark, Chapters 5 and 9
  - Franklin, Powell, & Emami-Naeini, Chapter 9
  - Franklin, Powell, & Workman, Chapter 13

Personal Notes

Personal Notes

Personal Notes

6. Discrete-time LTI control systems

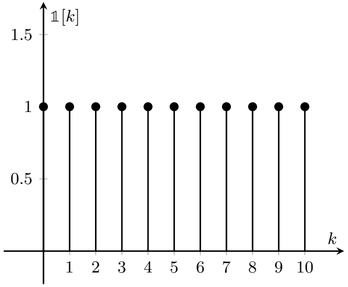
- discrete-time signals and z-transforms
- discrete-time LTI systems
- feedback stability
- time-domain analysis
- discrete-time frequency response
- state-space models: solutions, stability, transfer functions
- discrete-time stability analysis: Routh & Jury conditions

Notation

- set of integers $\mathbb{Z}$
- interior of the *unit disk* is $\mathbb{D} = \{z \in \mathbb{C} \mid |z| < 1\}$
- Unit step function

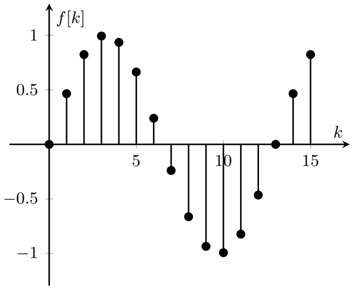
$$\mathbb{1}[k] = \begin{cases} 1 & \text{if} \quad k = 0, 1, 2, \ldots \\ 0 & \text{if} \quad \text{else} \end{cases}$$

$$\delta[k] = \begin{cases} 1 & \text{if} \quad k = 0 \\ 0 & \text{if} \quad \text{else} \end{cases}$$



Discrete-time signals

- a discrete-time signal is a sequence of numbers $f[0], f[1], f[2], \ldots$



- *only* defined at discrete points, not inbetween
- *may* be a sampled signal, with associated sampling period
- in this chapter, we effectively assume sampling period $T = 1$

The z-transform

- discrete equivalent of Laplace transform
- the (*unilateral* or *one-sided*) z-transform of a signal $f[k]$ is

$$F[z] := \mathcal{Z}\{f[k]\} = \sum_{k=0}^{\infty} f[k] z^{-k}\,, \qquad z \in \mathbb{C}\,.$$

- if signal $f[k]$ does not grow too fast, i.e., if

$$|f[k]| \leq M\rho^k$$

for some $M, \rho > 0$, then z-transform sum converges for all values of $z \in \mathbb{C}$ satisfying $|z| > \rho$

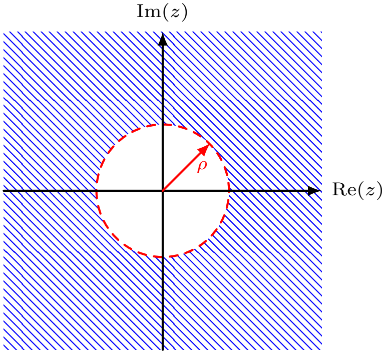
Convergence of z-transform

- suppose $f[k]$ satisfies $|f[k]| \leq M\rho^k$, and write $z = re^{j\theta}$. Then

$$\begin{aligned}
|F[z]| &= \left|\sum_{k=0}^{\infty} f[k]z^{-k}\right| = \left|\sum_{k=0}^{\infty} f[k](re^{j\theta})^{-k}\right| \leq \sum_{k=0}^{\infty} \left|f[k]r^{-k}e^{-jk\theta}\right| \\
&\leq \sum_{k=0}^{\infty} |f[k]| \cdot |r^{-k}| \cdot \underbrace{|e^{-jk\theta}|}_{=1} \\
&= \sum_{k=0}^{\infty} |f[k]| \cdot r^{-k} \\
&\leq \sum_{k=0}^{\infty} M\rho^k r^{-k} = M\sum_{k=0}^{\infty} \left(\frac{\rho}{r}\right)^k \qquad \text{(geometric series sum)} \\
&= M\frac{1}{1-\rho/r} \qquad \text{if} \qquad r > \rho
\end{aligned}$$

Convergence of z-transform contd.

- $F[z]$ is well-defined for all z *outside* a disk of radius ρ



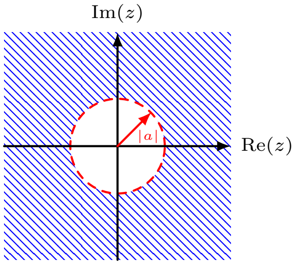
Region of Convergence

- in general, ρ equals magnitude of largest *pole* of $F[z]$

Example: exponential function

$$f[k] = a^k\,, \quad a \in \mathbb{R}$$

$$F[z] = \sum_{k=0}^{\infty} a^k z^{-k} = \sum_{k=0}^{\infty} \left(\frac{a}{z}\right)^k = \frac{1}{1-a/z} = \frac{z}{z-a}\,, \quad \text{for} \quad |z| > |a|$$



Region of Convergence

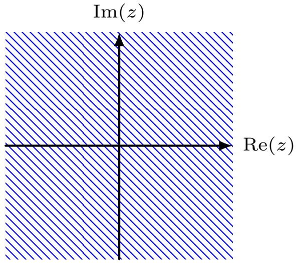
- note that $f[k]$ satisfies bound $|f[k]| \leq M\rho^k$ with

$$M = 1 \qquad \rho = |a|$$

Example: impulse function

$$f[k] = \delta[k] = \begin{cases} 1 & \text{if} \quad k = 0 \\ 0 & \text{if} \quad \text{else} \end{cases}$$

$$F[z] = \sum_{k=0}^{\infty} \delta[k] z^{-k} = z^{-0} = 1\,, \quad \text{for all } z \in \mathbb{C}$$



Region of Convergence

- note that $f[k]$ satisfies bound $|f[k]| \leq M\rho^k$ with

$$M = 1 \qquad \rho = 0$$

Key properties of z-transform

- **linearity**

$$\mathcal{Z}\{\alpha_1 f_1 + \alpha_2 f_2\} = \alpha_1 F_1[z] + \alpha_2 F_2[z]$$

- **accumulation formula**

$$\mathcal{Z}\left\{\sum_{\ell=0}^{k} f[\ell]\right\} = \frac{z}{z-1}F[z]$$

- **time-shifting formula** (with zero initial conditions)

$$\mathcal{Z}\left\{f[k-m]\right\} = z^{-m}F[z]$$

- **convolution**

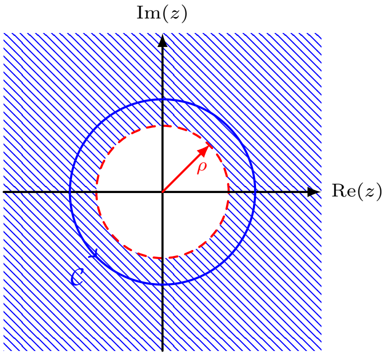
$$\mathcal{Z}\{g * f\} = \mathcal{Z}\left\{\sum_{\ell=0}^{k} g[k-\ell]f[\ell]\right\} = G[z]F[z]$$

Inverse z-transform

- given $F[z]$, signal $f[k]$ can be reconstructed through *contour integral*

$$f[k] = \frac{1}{2\pi j}\oint_{\mathcal{C}} F[z] z^{k-1}\,\mathrm{d}z\,, \qquad k \geq 0$$

where $\mathcal{C}$ is a closed counter-clockwise contour in the R.O.C.



Region of Convergence

Inverse z-transforms contd.

- in practice, we do partial fraction expansion, often on $\frac{F[z]}{z}$
- example

$$F[z] = \frac{8z-19}{(z-2)(z-3)} \qquad \Longrightarrow \qquad \frac{F[z]}{z} = \frac{8z-19}{z(z-2)(z-3)}$$

- partial fractions gives (**exercise: find $\boldsymbol{a_0, a_1, a_2}$**)

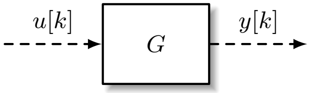
$$\begin{aligned} \frac{F[z]}{z} &= \frac{a_0}{z} + \frac{a_1}{z-2} + \frac{a_2}{z-3} \\ \Longrightarrow \quad F[z] &= a_0 + a_1\frac{z}{z-2} + a_2\frac{z}{z-3} \end{aligned}$$

- therefore

$$f[k] = a_0\delta[k] + a_1 2^k + a_2 3^k\,, \qquad k \geq 0$$

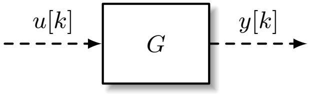
Discrete-time LTI systems

- system G takes input signal $u[k]$, produces output signal $y[k]$



- linearity, time-invariance, and causality defined exactly as they were for continuous-time systems

Representations of LTI systems



- in z-domain, *transfer function* $G[z]$

$$y[z] = G[z]u[z]$$

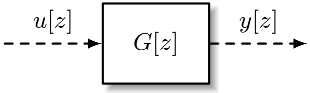
- in time-domain, *impulse response* $g[k]$

$$y[k] = g * u := \sum_{\ell=0}^{k} g[k-\ell]u[\ell]\,.$$

- these are equivalent, can show that

$$G[z] = \mathcal{Z}\{g[k]\} \qquad \text{and} \qquad g[k] = \mathcal{Z}^{-1}\{G[z]\}$$

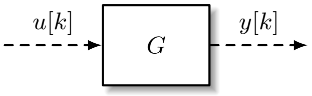
Transfer function representation contd.



- we call $G[z]$ *rational* if $G[z] = \frac{N[z]}{D[z]}$ for some polynomials $N[z]$ and $D[z]$ with real coefficients
- a *pole* p of $G[z]$ satisfies $\lim_{z\to p} |G[z]| = \infty$
- a *zero* ζ of $G[z]$ satisfies $G[\zeta] = 0$
- the degree $\deg(D)$ of the denominator is the *order* of the system
- $G[z]$ is *proper* if $\deg(N) \leq \deg(D)$, and is *strictly proper* if $\deg(N) < \deg(D)$
- *DC gain* of $G[z]$ is $G[1]$ (not $G[0]$!)

Bounded-input bounded-output stability

- a signal $y[k]$ is bounded if $|y[k]| \leq C$ for all $k \geq 0$



- **BIBO stability:** every bounded $u[k]$ produces a bounded $y[k]$
- if the LTI system G is rational and proper, then G is BIBO stable if and only if either of the following equivalent statements hold:
  - every pole of the transfer function $G[z]$ belongs to $\mathbb{D}$
  - the sum $\sum_{k=0}^{\infty} |g[k]|$ of the impulse response is finite.

Examples

$$\frac{1}{z} \qquad \frac{1}{z+1} \qquad \frac{(z+3)}{z^2+3z+2}$$

$$2^{-k} \qquad k^{100}2^{-k} \qquad \delta[k] \qquad \mathbb{1}[k]$$

Linear constant-coefficient difference equations

- discrete-time equivalent of differential equations ($n < m$)

$$y[k] + a_1 y[k-1] + \cdots + a_n y[k-n] = b_0 u[k] + \cdots + b_m u[k-m]$$

- initial conditions $\{y[-1], \ldots, y[-n]\}$ and input sequence $\{u[-m], \ldots, u[0], \ldots\}$ uniquely determine output sequence $\{y[0], y[1], \ldots\}$ (for example, by recursion)

- examples:
  - delay system: $y[k] = u[k-1]$
  - averaging system: $y[k] = \frac{1}{2}\left(u[k] + u[k-1]\right)$

- *difference equations and state-space models are how digital controllers are implemented*

Transfer functions from difference equations

- for difference equation

$$y[k] + a_1 y[k-1] + \cdots + a_n y[k-n] = b_0 u[k] + \cdots + b_m u[k-m]$$

- take z-transforms with zero initial conditions:

$$\begin{aligned} Y[z] + a_1 z^{-1} Y[z] + \cdots + a_n z^{-n} Y[z] &= b_0 U[z] + \cdots + b_m z^{-m} U[z] \\ (1 + a_1 z^{-1} + \cdots + a_n z^{-n}) Y[z] &= \left(b_0 + \cdots + b_m z^{-m}\right) U[z] \end{aligned}$$

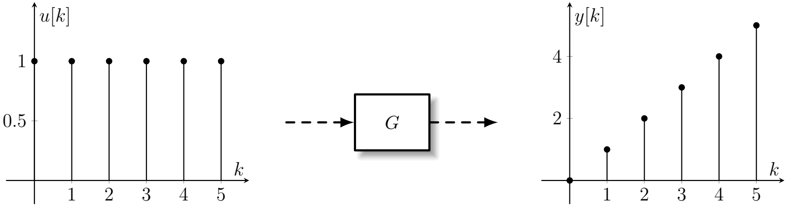
$$\begin{aligned} G[z] = \frac{Y[z]}{U[z]} &= \frac{b_0 + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \\ &= \frac{b_0 z^n + \cdots + b_m z^{n-m}}{z^n + a_1 z^{n-1} + \cdots + a_n} \end{aligned}$$

Example: summing system

- output is a running sum of inputs: $y[k] = y[k-1] + u[k-1]$

$$Y[z] = z^{-1}Y[z] + z^{-1}U[z] \qquad \Longrightarrow \qquad G[z] = \frac{Y[z]}{U[z]} = \frac{1}{z-1}$$

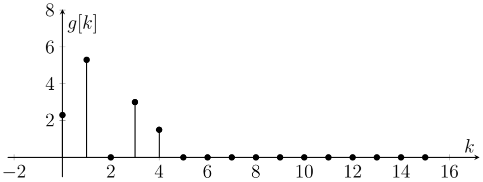
- system is not BIBO stable due to pole at $z = 1$



Example: finite impulse response (FIR) filters

- the impulse response of an FIR filter is non-zero for only a finite number of time steps

$$g[k] = b_0\delta[k] + b_1\delta[k-1] + \cdots + b_m\delta[k-m]$$

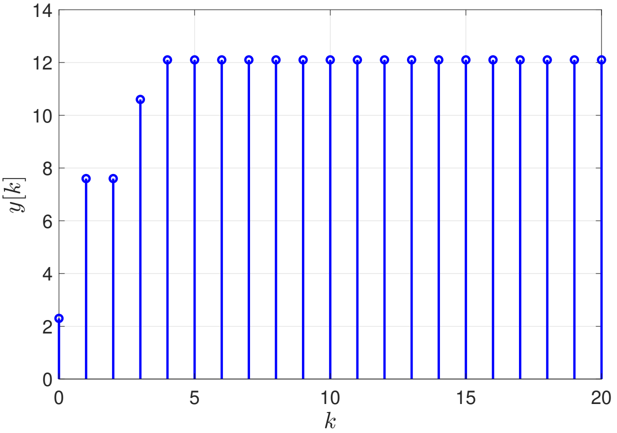


- transfer function $G[z] = \mathcal{Z}\{g[k]\}$ is

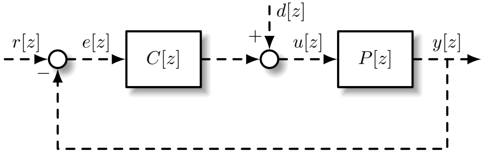
$$G[z] = b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}$$

and has *all of its poles* at $z = 0$

Example: unit step response of FIR filter



Feedback stability of discrete-time control systems



- r and d are *external* signals, e, u and y are *internal* signals
- the closed-loop is *feedback stable* if every bounded (r, d) leads to bounded (e, u, y)
- 2 inputs, 3 outputs $\Longrightarrow$ 6 transfer functions

$$\frac{e[z]}{r[z]} = \frac{1}{1+PC} \qquad \frac{u[z]}{r[z]} = \frac{C}{1+PC} \qquad \frac{y[z]}{r[z]} = \frac{PC}{1+PC}$$
$$\frac{e[z]}{d[z]} = \frac{-P}{1+PC} \qquad \frac{u[z]}{d[z]} = \frac{1}{1+PC} \qquad \frac{y[z]}{r[z]} = \frac{P}{1+PC}$$

Feedback stability contd.

- assume $P[z]$ is rational and strictly proper: $P[z] = \frac{N_\mathrm{p}[z]}{D_\mathrm{p}[z]}$
- assume $C[z]$ is rational and proper: $C[z] = \frac{N_\mathrm{c}[z]}{D_\mathrm{c}[z]}$
- we calculate that

$$\frac{y[z]}{r[z]} = \frac{PC}{1+PC} = \frac{\frac{N_\mathrm{p}}{D_\mathrm{p}}\frac{N_\mathrm{c}}{D_\mathrm{c}}}{1+\frac{N_\mathrm{p}}{D_\mathrm{p}}\frac{N_\mathrm{c}}{D_\mathrm{c}}} = \frac{N_\mathrm{p}N_\mathrm{c}}{N_\mathrm{p}N_\mathrm{c}+D_\mathrm{p}D_\mathrm{c}}$$

- *characteristic polynomial:* $\Pi[z] := N_\mathrm{p}[z]N_\mathrm{c}[z] + D_\mathrm{p}[z]D_\mathrm{c}[z]$
- the closed-loop is feedback stable <u>if and only if</u> all roots of $\Pi[z]$ belong to $\mathbb{D}$

Example: feedback stability

- plant: $y[k] = y[k-1] + u[k-1]$, or T.F. $P[z] = \frac{1}{z-1}$
- controller: $u[k] = \frac{1}{2}\left(e[k] + e[k-1]\right)$, or T.F. $C[z] = \frac{z+1}{2z}$
- compute characteristic polynomial

$$\Pi[z] = (z+1) + 2z(z-1) = 2\left(z - \frac{1}{4} + j\frac{\sqrt{7}}{4}\right)\left(z - \frac{1}{4} - j\frac{\sqrt{7}}{4}\right)$$

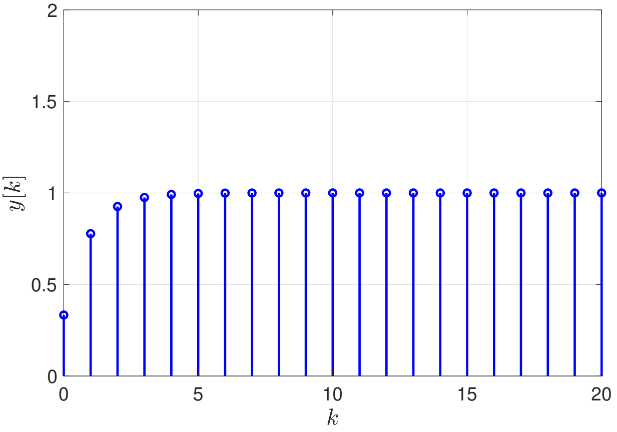
- magnitude of pole(s) is

$$\left[\left(\frac{1}{4}\right)^2 + \left(\frac{\sqrt{7}}{4}\right)^2\right]^{1/2} = \frac{1}{\sqrt{2}} < 1$$

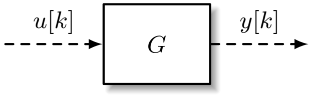
- closed-loop system is feedback stable

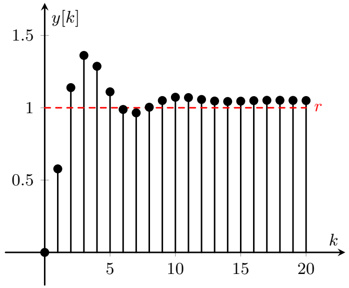
Example: feedback stability contd.

- unit step response of unity-gain feedback system



Time-domain analysis





(i) rise time T_r, settling time T_p

(ii) peak time T_p, peak max M_p

(iii) steady state value y_ss?

- **Final value theorem:** if $F[z] = \mathcal{Z}\{f[k]\}$ is rational and proper, with all poles of $(z-1)F[z]$ contained in $\mathbb{D}$, then $f_\mathrm{ss} = \lim_{k\to\infty} f[k]$ exists and

$$f_\mathrm{ss} = \lim_{k\to\infty} f[k] = \lim_{z\to 1} (z-1)F[z]\,.$$

Example: second-order system

for a step $u[k] = \mathbb{1}[k]$, find steady-state value for $y[k]$

$$G[z] = \frac{z+2}{z(z-0.5)}$$

$$Y[z] = G[z]U[z] = \frac{z+2}{z(z-0.5)}\frac{z}{z-1}$$

$$(z-1)Y[z] = \frac{z+2}{(z-0.5)}$$

this has all poles inside the unit circle, therefore

$$\lim_{k\to\infty} y[k] = \lim_{z\to 1}(z-1)Y[z] = 6 = G[1]$$

- ▶ DC gain $G[1]$ gives steady-state value of step response

First-order discrete-time systems

- prototypical first-order system is described by

$$y[k] = -a_1 y[k-1] + b_1 u[k-1] \qquad \Longrightarrow \qquad G[z] = \frac{b_1}{z + a_1}$$

- pole at $z = -a_1$, system is BIBO stable if $|a_1| < 1$

- to normalize DC gain such that $G[1] = 1$, set $b_1 = 1 + a_1$

$$G[z] = \frac{1 + a_1}{z + a_1}$$

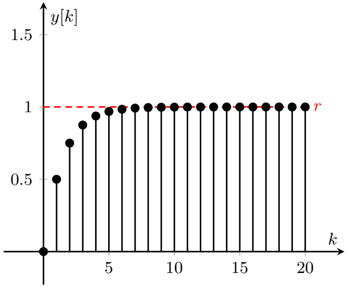
- unit step response

$$Y[z] = \frac{1 + a_1}{z + a_1} \frac{z}{z - 1} \qquad \xmapsto{\mathcal{Z}^{-1}} \qquad y[k] = 1 - (-a_1)^k$$

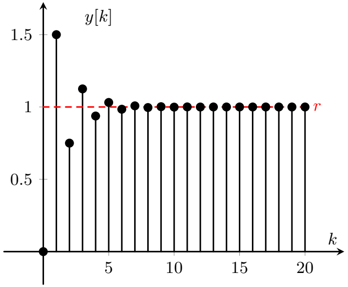
Step response of first-order system

$$y[k] = 1 - (-a_1)^k\,, \qquad k \geq 0$$

for $-1 < a_1 < 0$

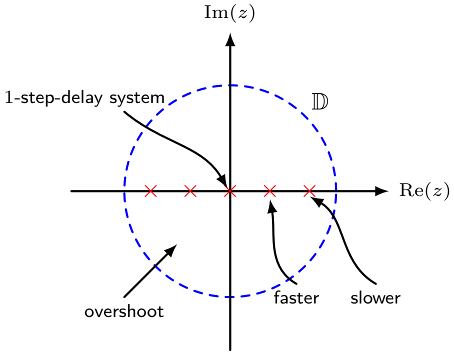


for $0 < a_1 < 1$



Step response of first-order system contd.

- in discrete-time, first-order systems can have overshoot
- magnitude of a_1 controls settling time



Second-order discrete-time systems

- prototypical second-order system is described by

$$y[k] + a_1 y[k-1] + a_2 y[k-2] = b_2 u[k-2]$$

- transfer function

$$G[z] = \frac{b_2}{z^2 + a_1 z + a_2}$$

- to normalized DC gain $G[1] = 1$, set $b_2 = 1 + a_1 + a_2$
- **three cases of interest:** $a_2 > 0$, $a_2 < 0$, and $a_2 = 0$
- if $a_2 > 0$, two conjugate poles $z_\pm = re^{\pm j\theta}$, determined by

$$a_2 = r^2 \qquad a_1 = -2r\cos(\theta)$$

- system BIBO stable if $r < 1 \quad \Longleftrightarrow \quad a_2 < 1$

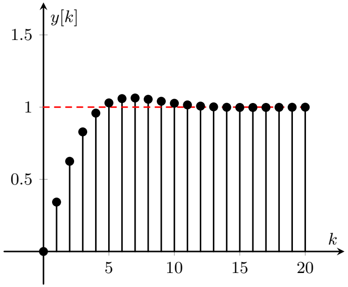
Step response of second-order systems

- messy inverse z-transform calculation gives formula

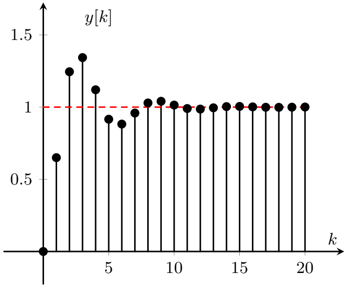
$$y[k] = 1 - \alpha r^k \cos(\theta k + \beta)\,, \qquad k \geq 0$$

where α, β are constants; θ determines the amount of oscillation

<u>for $r = 0.7, \theta = 20$</u>



<u>for $r = 0.7, \theta = 60$</u>

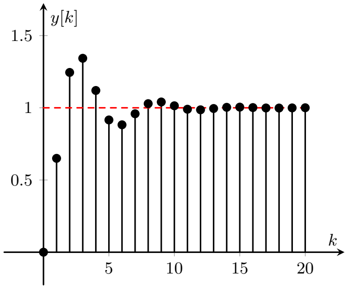


Step response of second-order system contd.

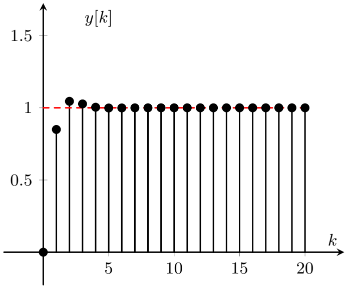
$$y[k] = 1 - \alpha r^k \cos(\theta k + \beta)$$

- the value of r determines the settling time

<u>for $r = 0.7, \theta = 60$</u>

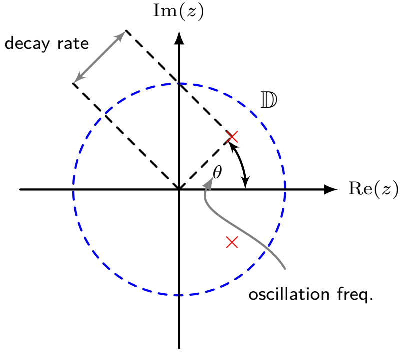


<u>for $r = 0.3, \theta = 60$</u>



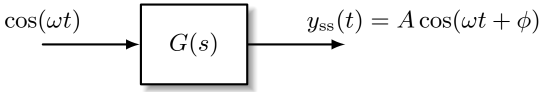
Step-response of second-order systems contd.

- responses very similar to continuous-time second-order systems
- pole location meanings are very different



Frequency response of discrete-time systems

- for a continuous-time LTI system which is BIBO stable



where $A = |G(j\omega)|$ and $\phi = \angle G(j\omega)$

- is something similar true for discrete-time systems?
- assume $G[z]$ is rational, proper, and BIBO stable
- output $y[k]$ of system is given by convolution

$$y[k] = \sum_{m=-\infty}^{\infty} g[m]u[k-m]$$

Frequency response contd.

- let's try the complex exponential input $u[k] = e^{j\omega k}$
- *annoying technical note*: we will start applying input at $k = -\infty$, so that system can reach a nice steady-state by time $k = 0$.

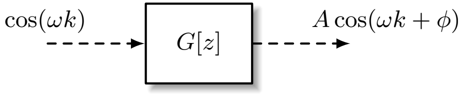
$$
\begin{aligned}
y[k] &= \sum_{m=-\infty}^{\infty} g[m] e^{j\omega(k-m)} \\
&= e^{j\omega k} \sum_{m=-\infty}^{\infty} g[m] e^{-j\omega m} \\
&= e^{j\omega k} G[e^{j\omega}] \qquad \text{(discrete Fourier transform)}
\end{aligned}
$$

$$\begin{tikzcd} {} \arrow[r, dashed, "e^{j\omega k}"] & \boxed{G[z]} \arrow[r, dashed, "{y_{\text{ss}}[k] = G[e^{j\omega}]e^{j\omega k}}"] & {} \end{tikzcd}$$

Frequency response contd.

- we may now take the real part of each signal

$$\mathrm{Re}(e^{j\omega k}) = \cos(\omega k)$$
$$\mathrm{Re}(G[e^{j\omega}]e^{j\omega k}) = \left|G[e^{j\omega}]\right| \cos(\omega k + \angle G[e^{j\omega}])$$



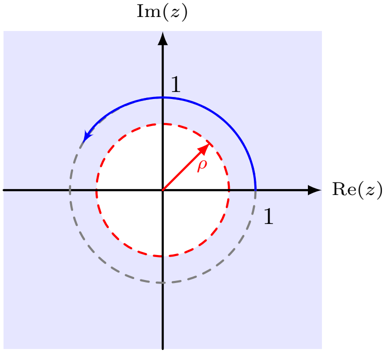
where

$$A = \left|G[e^{j\omega}]\right| \qquad \phi = \angle G[e^{j\omega}]$$

- $G[e^{j\omega}]$ is the frequency response of the system
- cosine applied at input yields shifted and scaled cosine at the output

Frequency response contd.

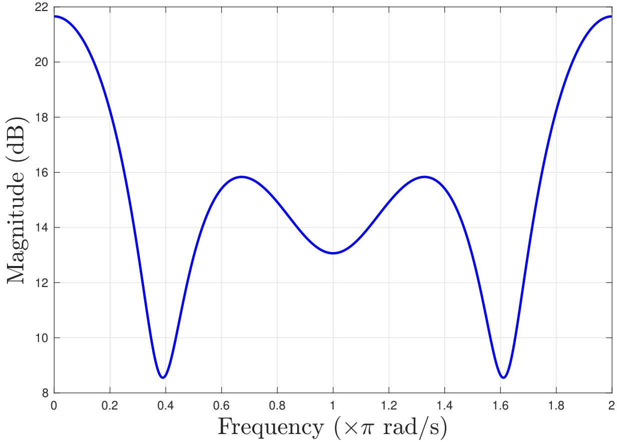
- to get frequency response, evaluate $G[z]$ at $z = e^{j\omega}$
- what do things look like in the complex plane?



- $G[e^{j\omega}]$ is a periodic function of ω

Example: frequency response of FIR filter

- Bode magnitude plot



- **note**: typically frequency plotted from $[0, \pi]$ on **log** scale

Example: frequency response

$$G[z] = \frac{3}{z - 0.5} \qquad \text{with input} \qquad u[k] = \cos(3k)$$

- system is BIBO stable, so steady-state exists, and is of the form $A\cos(3k + \phi)$

$$A = \left|G[e^{j\omega}]\right| = \frac{3}{|e^{j3} - 0.5|} \simeq 2.005$$

$$\begin{aligned} \phi = \angle G[e^{j\omega}] &= \angle 3 - \angle(e^{j3} - 0.5) \\ &= 0 - \angle(\cos(3) - 0.5 + j\sin(3)) \\ &= -\arctan\left(\frac{\sin(3)}{\cos(3) - 0.5}\right) \simeq -174^\circ \end{aligned}$$

- steady-state output is $y_{\mathrm{ss}}[k] = 2.005\cos(3k - 174^\circ)$

Discrete-time LTI state-space models

- a discrete-time LTI state-space model has the form

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] \\ y[k] &= Cx[k] + Du[k] \end{aligned}$$

- $x[k] \in \mathbb{R}^n$ is the *state vector*
- $n \in \mathbb{N}$ is the *order*
- $u[k] \in \mathbb{R}^m$ is the *input vector*
- $y[k] \in \mathbb{R}^p$ is the *output vector*

- $A \in \mathbb{R}^{n \times n}$
- $B \in \mathbb{R}^{n \times m}$
- $C \in \mathbb{R}^{p \times n}$
- $D \in \mathbb{R}^{p \times m}$

often, we write x^+ for $x[k+1]$ and simply x for $x[k]$

Example: second-order system

- second-order system: $f[k] + a_1 f[k-1] + a_2 f[k-2] = b_0 u[k]$
- let $x_1[k] = f[k-2]$ and $x_2[k] = f[k-1]$

$$\begin{aligned} x_1[k+1] &= f[k-1] \\ &= x_2[k] \end{aligned} \qquad \begin{aligned} x_2[k+1] &= f[k] \\ &= -a_1 f[k-1] - a_2 f[k-2] + b_0 u[k] \\ &= -a_1 x_2[k] - a_2 x_1[k] + b_0 u[k] \end{aligned}$$

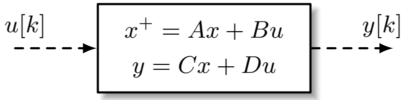
- state model with output $x_1[k] = f[k-2]$ is therefore

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^+ = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

Other comments on state models

- state vector x is *internal* to the system, acts as intermediary



- for an nth order difference equation, you need n states

- there is no unique choice of state variables

- the condition for *equilibrium* is

$$x^+ = x \qquad \Longleftrightarrow \qquad x = Ax + Bu$$

- x, y, u are often *deviations* from desired values

Solving state-space models without inputs

- consider zero input $u[k] = 0$ with initial condition $x[0] = x_0 \in \mathbb{R}^n$

$$x[k+1] = Ax[k]\,, \qquad x[0] = x_0$$

- just by iterating, we find that

$$\begin{aligned} x[1] &= Ax_0 \\ x[2] &= A(Ax_0) = A^2 x_0 \\ &\vdots \\ x[k] &= A^k x_0 \end{aligned}$$

- so with no input, the solution is

$$x[k] = A^k x_0\,, \qquad k \geq 0$$

Computing A^k via z-transforms

- we know solution to $x^+ = Ax$ is $x[k] = A^k x[0]$
- we could also z-transform (element-by-element) both sides to find

$$\begin{aligned} zX[z] - zx[0] = AX[z] \quad &\iff \quad (zI - A)X[z] = zx[0] \\ &\iff \quad (I - z^{-1}A)X[z] = x[0] \\ &\iff \quad X[z] = (I - z^{-1}A)^{-1}x[0] \\ &\iff \quad x[k] = \mathcal{Z}^{-1}\{(I - z^{-1}A)^{-1}\}x[0] \end{aligned}$$

- comparing the solutions, we find that

$$A^k = \mathcal{Z}^{-1}\{(I - z^{-1}A)^{-1}\}$$

Example: computing A^k

$$x[k+1] = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} x[k]\,, \qquad x[0] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

compute that

$$(I - z^{-1}A)^{-1} = z(zI - A)^{-1} = z\begin{bmatrix} z & -1 \\ 2 & z+3 \end{bmatrix}^{-1} \quad = \frac{z}{z^2+3z+2}\begin{bmatrix} z+3 & 1 \\ -2 & z \end{bmatrix}$$

therefore

$$A^k = \mathcal{Z}^{-1}\left\{\begin{bmatrix} \frac{z(z+3)}{(z+1)(z+2)} & \frac{z}{(z+1)(z+2)} \\ \frac{-2z}{(z+1)(z+2)} & \frac{z^2}{(z+1)(z+2)} \end{bmatrix}\right\}$$

$$= \begin{bmatrix} 2(-1)^k - (-2)^k & (-1)^k - (-2)^k \\ 2(-2)^k - 2(-1)^k & 2(-2)^k - (-1)^k \end{bmatrix}$$

Internal stability of state models

- state-space system $x^+ = Ax$ is *internally asymptotically stable* if $x[k] \to 0$ as $k \to \infty$ from any initial condition $x[0]$
- with no external inputs, the state goes to zero
- suppose A is diagonalizable with $V^{-1}AV = \Lambda$
  - change of state variables $z = V^{-1}x \quad \Longrightarrow \quad z[0] = V^{-1}x[0]$

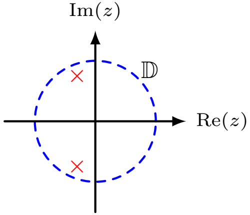
$$z[k] = V^{-1}x[k] = V^{-1}A^k x[0] = V^{-1}A^k V z[0] = \Lambda^k z[0]$$

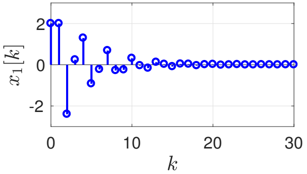
  - this says that $z_i[k] = \lambda_i^k z_i[0]$

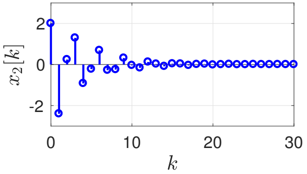
A discrete-time LTI state model is internally asymptotically stable if and only if $\lambda_i \in \mathbb{D}$ for all $\lambda_i \in \mathrm{eig}(A)$, i.e., all eigenvalues of A have magnitude less than one

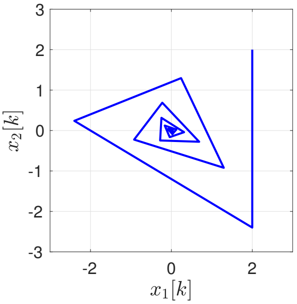
Example: internally stable second-order system

$$x^+ = \begin{bmatrix} 0 & 1 \\ -0.6 & -0.6 \end{bmatrix} x\,, \quad x[0] = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$



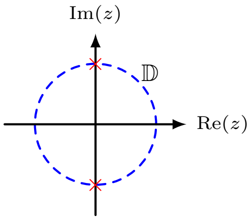


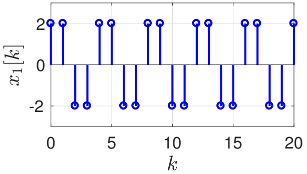


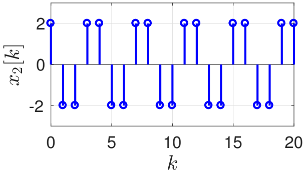


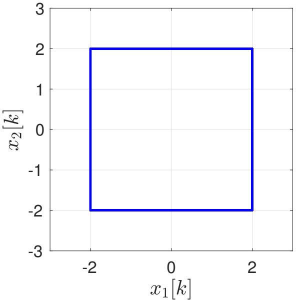
Example: unstable second-order system

$$x^+ = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x\,, \quad x[0] = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$



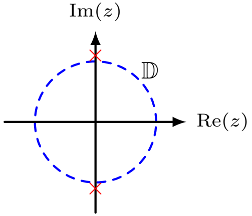


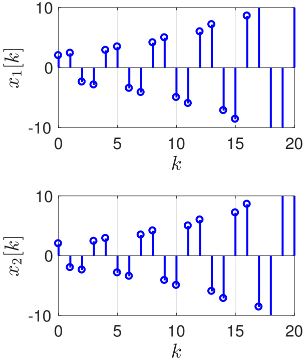


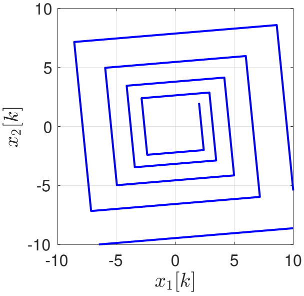


Example: unstable second-order system

$$x^+ = \begin{bmatrix} 0 & 1.2 \\ -1 & 0 \end{bmatrix} x\,, \quad x[0] = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$







Solution of state-space model with input

back to our general model with inputs

$$x[k+1] = Ax[k] + Bu[k]\,, \qquad x[0] \in \mathbb{R}^n$$

we can just iterate to find the solution

$$\begin{aligned}
x[1] &= Ax[0] + Bu[0] \\
x[2] &= A^2x[0] + ABu[0] + Bu[1] \\
x[3] &= A^3x[0] + A^2Bu[0] + ABu[1] + Bu[2] \\
&\;\vdots \\
x[k] &= A^kx[0] + \sum_{j=0}^{k-1} A^{k-j-1}Bu[j]
\end{aligned}$$

- combination of *natural* response and *forced* response

Transfer function from state models

$$\begin{tikzcd} {} \arrow[r, dashed, "{u[k]}"] & \boxed{\begin{array}{c} x^+ = Ax + Bu \\ y = Cx + Du \end{array}} \arrow[r, dashed, "{y[k]}"] & {} \end{tikzcd}$$

taking z-transforms with zero initial conditions $x[0] = \mathbb{0}$

$$\begin{aligned} zX[z] &= AX[z] + BU[z] \\ Y[z] &= CX[z] + DU[z] \end{aligned} \quad \Longrightarrow \quad \begin{aligned} X[z] &= (zI - A)^{-1}BU[z] \\ Y[z] &= CX[z] + DU[z] \end{aligned}$$

$$\boxed{\frac{Y[z]}{U[z]} = P[z] = C(zI - A)^{-1}B + D}$$

- state-space model *uniquely* determines $P[z]$
- transfer functions from state-space models are always proper

Internal asymptotic stability vs. BIBO stability

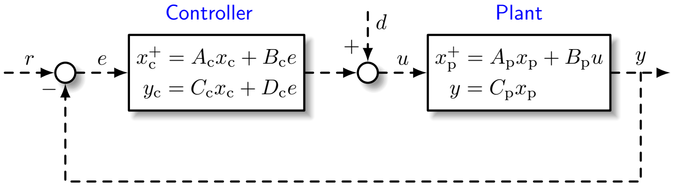
- internal stability and BIBO stability are related

$$
\begin{aligned}
P[z] &= C(zI - A)^{-1}B + D \\
&= C\,\frac{\operatorname{adj}(zI - A)}{\det(zI - A)}\,B + D \\
&= \frac{C\operatorname{adj}(zI - A)B + D\det(zI - A)}{\det(zI - A)} = \frac{\text{(some polynomial)}}{\Pi_A[z]}
\end{aligned}
$$

- every *pole* of $P[z]$ is an *eigenvalue* of A
- state model internally asymptotically stable $\Longrightarrow$ TF is BIBO stable

Feedback stability of state-space models

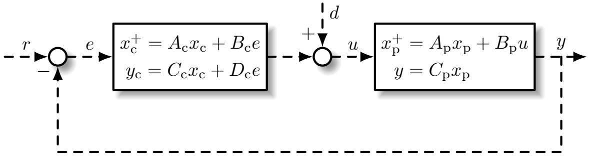
- we can understand feedback stability using state models



- **idea:** eliminate e, y_c, u and find write state model with states $x = (x_p, x_c)$, inputs (r, d) and output y

- **note:** in this case, our state-space model will have two inputs

Feedback stability using state-space models



r
$-$
e
$x_{\mathrm{c}}^+ = A_{\mathrm{c}}x_{\mathrm{c}} + B_{\mathrm{c}}e$
$y_{\mathrm{c}} = C_{\mathrm{c}}x_{\mathrm{c}} + D_{\mathrm{c}}e$
d
$+$
u
$x_{\mathrm{p}}^+ = A_{\mathrm{p}}x_{\mathrm{p}} + B_{\mathrm{p}}u$
$y = C_{\mathrm{p}}x_{\mathrm{p}}$
y

$$x_{\mathrm{p}}^+ = A_{\mathrm{p}}x_{\mathrm{p}} + B_{\mathrm{p}}(d + y_{\mathrm{c}}) \qquad x_{\mathrm{c}}^+ = A_{\mathrm{c}}x_{\mathrm{c}} + B_{\mathrm{c}}(r - y)$$

$$y = C_{\mathrm{p}}x_{\mathrm{p}} \qquad y_{\mathrm{c}} = C_{\mathrm{c}}x_{\mathrm{c}} + D_{\mathrm{c}}(r - y)$$

combine:
$$\begin{bmatrix} x_{\mathrm{p}} \\ x_{\mathrm{c}} \end{bmatrix}^+ = \underbrace{\begin{bmatrix} A_{\mathrm{p}} - B_{\mathrm{p}}D_{\mathrm{c}}C_{\mathrm{p}} & B_{\mathrm{p}}C_{\mathrm{c}} \\ -B_{\mathrm{c}}C_{\mathrm{p}} & A_{\mathrm{c}} \end{bmatrix}}_{:=A_{\mathrm{cl}}} \underbrace{\begin{bmatrix} x_{\mathrm{p}} \\ x_{\mathrm{c}} \end{bmatrix}}_{:=x_{\mathrm{cl}}} + \underbrace{\begin{bmatrix} B_{\mathrm{p}}D_{\mathrm{c}} & B_{\mathrm{p}} \\ B_{\mathrm{c}} & 0 \end{bmatrix}}_{:=B_{\mathrm{cl}}} \begin{bmatrix} r \\ d \end{bmatrix}$$

$$y = \underbrace{\begin{bmatrix} C_{\mathrm{p}} & 0 \end{bmatrix}}_{C_{\mathrm{cl}}} \begin{bmatrix} x_{\mathrm{p}} \\ x_{\mathrm{c}} \end{bmatrix}$$

Feedback stability using state-space models

- the closed-loop state model has the form

$$x_{\mathrm{cl}}^{+} = A_{\mathrm{cl}} x_{\mathrm{cl}} + B_{\mathrm{cl}} \begin{bmatrix} r \\ d \end{bmatrix}, \qquad y = C_{\mathrm{cl}} x_{\mathrm{cl}}$$

- **from before:** internal asymptotic stability of state model $\Longrightarrow$ BIBO stability of transfer function

- similarly now, internal asymptotic stability of closed-loop model $\Longrightarrow$ BIBO stability of any closed-loop transfer function (i.e., feedback stability)

$$\begin{array}{c} \lambda_i \in \mathbb{D} \quad \text{for all} \\ \lambda_i \in \mathrm{eig}(A_{\mathrm{cl}}) \end{array} \quad \Longrightarrow \quad \begin{array}{c} \text{all transfer func.} \\ \text{are BIBO stable} \end{array} \quad \Longleftrightarrow \quad \begin{array}{c} \text{feedback stability of} \\ \text{feedback system} \end{array}$$

Discrete-time stability analysis

- we have seen several discrete-time stability concepts

  (i) BIBO stability of a transfer function $G[z]$

  (ii) feedback stability with plant $P[z]$ and controller $C[z]$

  (iii) internal asymptotic stability of state-space models

  (iv) feedback stability with state-space plant and controller

- in all cases, we must ask whether all roots of some polynomial

$$\Pi[z] = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

  are contained in $\mathbb{D}$

- we now develop a tool for actually *checking* stability

Review: Routh-Hurwitz for continuous-time systems

- **goal**: determine whether all roots of polynomial $\Pi(s)$ are in $\mathbb{C}_-$

$$\Pi(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0$$

- algorithmic test through the *Routh array*

row n	a_n	a_{n-2}	a_{n-4}	$\cdots$
row n-1	a_{n-1}	a_{n-3}	a_{n-5}	$\cdots$
row n-2	b_1	b_2	b_3	$\cdots$
row n-3	c_1	c_2	c_3	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
row 2	$*$	$*$		
row 1	$*$			
row 0	$*$			

$$b_i = \frac{a_{n-1}a_{n-2i} - a_n a_{n-2i-1}}{a_{n-1}}$$

$$c_i = \frac{b_1 a_{n-2i-1} - a_{n-1} b_{i+1}}{b_1}$$

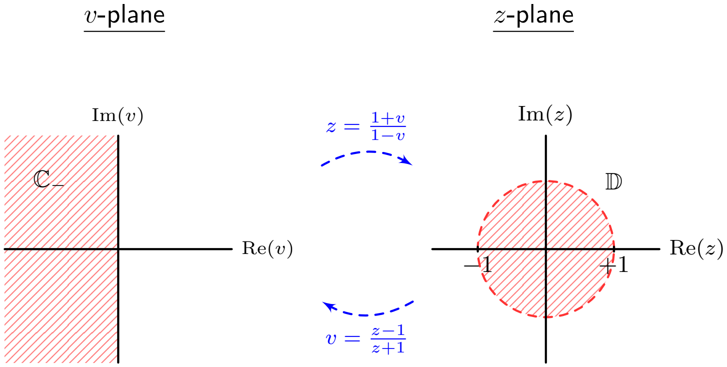
$$d_i = \cdots$$

Routh-Hurwitz contd.

- **necessary conditions** for all roots in $\mathbb{C}_-$
  - the coefficients a_i must all be non-zero
  - the coefficients a_i must all be positive
- **sufficient conditions** for all roots in $\mathbb{C}_-$
  - if all entries in the first column are positive, then all roots are in $\mathbb{C}_-$
  - there is one pole in $\mathbb{C}_+$ for every sign change in the first column
- useful for tuning controller gains

Routh-Hurwitz for discrete-time systems

- Routh-Hurwitz addresses roots in $\mathbb{C}_-$, not in $\mathbb{D}$
- **key trick:** use the transformation $z = \frac{1+v}{1-v}$ between $\mathbb{C}_-$ and $\mathbb{D}$



Routh-Hurwitz for discrete-time systems contd.

given a polynomial

$$\Pi[z] = z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

(i) evaluate $\Pi[z]$ at $z = \frac{1+v}{1-v}$, that is, form $\Pi\left[\frac{1+v}{1-v}\right]$

(ii) multiply through by $(1-v)^n$ to obtain a new polynomial $\widehat{\Pi}(v)$

(iii) apply standard Routh-Hurwitz test to $\widehat{\Pi}(v)$

The polynomial $\Pi[z]$ has all roots in $\mathbb{D}$ if and only if the polynomial $\widehat{\Pi}(v)$ has all roots in $\mathbb{C}_-$

Example: second-order system

$$\Pi[z] = z^2 + a_1 z + a_2$$

- form $\widehat{\Pi}(v)$

$$\Pi\left[\frac{1+v}{1-v}\right] = \left(\frac{1+v}{1-v}\right)^2 + a_1\left(\frac{1+v}{1-v}\right) + a_2$$

$$\begin{aligned}\widehat{\Pi}(v) &= (1-v)^2\Pi\left[\frac{1+v}{1-v}\right] \\ &= (1+v)^2 + a_1(1+v)(1-v) + a_2(1-v)^2 \\ &= (1+a_2-a_1)v^2 + 2(1-a_2)v + (1+a_2+a_1)\end{aligned}$$

Example: second-order system contd.

$$\widehat{\Pi}(v) = (1 + a_2 - a_1)v^2 + 2(1 - a_2)v + (1 + a_2 + a_1)$$

row 2	$1 + a_2 - a_1$	$1 + a_2 + a_1$
row 1	$2(1 - a_2)$	0
row 0	$1 + a_2 + a_1$	0

- we need all first column entries to be positive

$$1 + a_2 - a_1 > 0 \quad \text{and} \quad 1 - a_2 > 0 \quad \text{and} \quad 1 + a_2 + a_1 > 0$$

Example: second-order system contd.

- we have three inequalities

$$1 + a_2 - a_1 > 0 \qquad 1 - a_2 > 0 \qquad 1 + a_2 + a_1 > 0$$

$$1 + a_2 > a_1 \qquad a_2 < 1 \qquad 1 + a_2 > -a_1$$

- if $a_2 \leq -1$, first and third inequalities are impossible, so $a_2 > -1$
- conditions are therefore

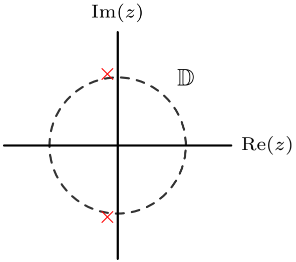
$$\boxed{\begin{aligned} &|a_2| < 1 \\ &|a_1| < 1 + a_2 \end{aligned}}$$

Example: second-order system contd.

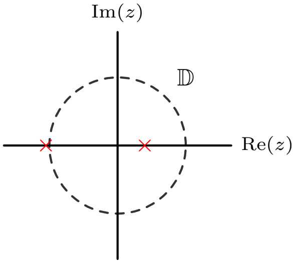
$$\Pi[z] = z^2 + a_1 z + a_2$$

$$|a_2| < 1$$
$$|a_1| < |1 + a_2|$$

$\underline{a_2 = 1.1 \;\; a_1 = 0.3}$



$\underline{a_2 = -0.4 \;\; a_1 = 0.7}$



Jury's necessary conditions

- alternative set of tests for all roots of polynomial

$$\Pi[z] = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

  to be contained in $\mathbb{D}$

- Jury's **necessary** stability conditions are

  (i) $\Pi[1] > 0$

  (ii) $(-1)^n \Pi[-1] > 0$

  (iii) $|a_n| < |a_0|$

- equivalently, these conditions are *sufficient* for *instability*

Example: testing feedback stability

- first-order plant with "PI" controller

$$P[z] = \frac{z}{z+a}\,, \qquad C[z] = K_{\mathrm{p}} + K_{\mathrm{i}}\frac{z}{z-1}\,, \qquad \text{where } a > 0$$

- characteristic polynomial

$$\begin{aligned}\Pi[z] &= (K_{\mathrm{p}} + K_{\mathrm{i}})z^2 - K_{\mathrm{p}}z + (z+a)(z-1)\\ &= (1 + K_{\mathrm{p}} + K_{\mathrm{i}})z^2 + (a - 1 - K_{\mathrm{p}})z - a\end{aligned}$$

- Jury's necessary conditions tell us that — at least — we need

  (i) $K_i > 0$

  (ii) $1 + K_{\mathrm{p}} + \frac{1}{2}K_{\mathrm{i}} > a$

  (iii) $|1 + K_{\mathrm{p}} + K_{\mathrm{i}}| > a$

Example: testing feedback stability contd.

- the necessary & sufficient conditions from Routh-Hurwtiz are

$$\left|\frac{-a}{1+K_{\mathrm{p}}+K_{\mathrm{i}}}\right| < 1\,, \qquad \left|\frac{a-1-K_{\mathrm{p}}}{1+K_{\mathrm{p}}+K_{\mathrm{i}}}\right| < \left|1-\frac{a}{1+K_{\mathrm{p}}+K_{\mathrm{i}}}\right|$$

- first condition says $\boxed{|1+K_{\mathrm{p}}+K_{\mathrm{i}}| > a}$
- second condition says that

$$|1+K_{\mathrm{p}}-a| < 1+K_{\mathrm{p}}+K_{\mathrm{i}}-a$$

  - if LHS is positive, this says $\boxed{K_{\mathrm{i}} > 0}$
  - if LHS is negative, this says $\boxed{1+K_{\mathrm{p}}+\frac{1}{2}K_{\mathrm{i}} > a}$

MATLAB commands

- specifying discrete-time model

```
T = 1; z = tf(‘z’,T); G = z/(z-1);
```

- computing z-transform

```
syms k z
f = k^2*sin(2*pi*k); F = ztrans(f,k,z);
f_again = iztrans(F,z,k);
```

- specifying state models

```
my_ss = ss(A,B,C,D,T);
```

- pole, zero, feedback, connect, bode, pzplot, ...
- custom command `routh` (on LEARN)

Additional references

- discrete time systems and control systems
  - Nielsen, Chapter 5
  - Phillips, Nagle, & Chakrabortty, Chapter 2
  - Franklin, Powell, & Workman, Chapter 4
  - Franklin, Powell, & Emami-Naeini, Chapter 5
- discrete-time stability tests
  - Nielsen, Chapter 8
  - Phillips, Nagle, & Chakrabortty, Chapter 7

Personal Notes

Personal Notes

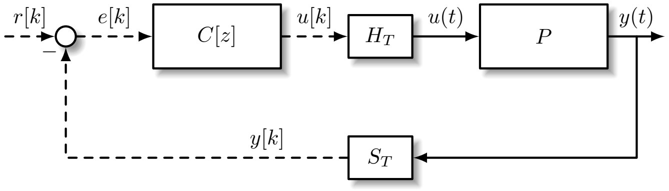
Personal Notes

7. Discretizing plants for direct design

- step-invariant discretization
- stability of sampled-data systems
- frequency response of discretized model

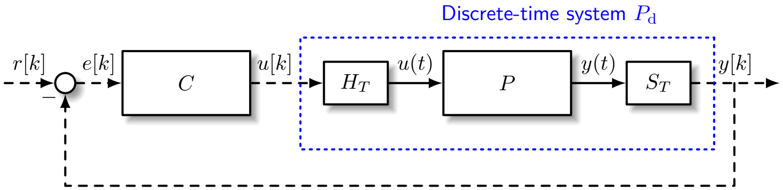
Back to sampled-data systems

- we have been doing emulation design: design $C(s)$ then discretize it
- alternative approach: take plant P, discretize it, then design discrete-time controller for the discrete-time plant



- lets slightly redraw this block diagram

Step-invariant discretization



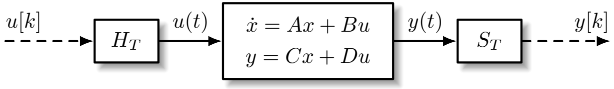
- the system inside brackets is a discrete-time system: we call it

$$P_{\mathrm{d}} = \mathrm{c2d}(P)$$

- remarkably, we will show P_{d} is an LTI system
- therefore, controller "sees" a discrete-time LTI system!

Step-invariant discretization

- we will represent P by a state-space model (other choices possible)



- if $x(0)$ is the initial condition at time $t_0 = 0$, then

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau\,, \qquad t \geq 0$$

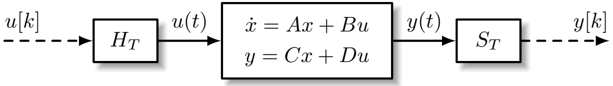
- more generally, if $x(t_0)$ is the initial condition at time t_0, then

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^t e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau\,, \qquad t \geq t_0$$

Step-invariant discretization contd.

- let $t_k = kT$ be the sampling times, $T > 0$ sampling period
- set initial time $t_0 = t_k$, solution at time $t = t_{k+1}$ is

$$x(t_{k+1}) = e^{A(t_{k+1}-t_k)}x(t_k) + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bu(\tau)\,\mathrm{d}\tau$$



- $u(t)$ comes from hold block $\Longrightarrow u(t) = u(t_k) =$ const. over $[t_k, t_{k+1}]$

$$x(t_{k+1}) = e^{A(t_{k+1}-t_k)}x(t_k) + \left(\int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}B\,\mathrm{d}\tau\right)u(t_k)$$

Step-invariant discretization contd.

- change of variables $\sigma = t_{k+1} - \tau$ inside integral

$$x(t_{k+1}) = e^{AT}x(t_k) + \left(\int_0^T e^{A\sigma}B\,\mathrm{d}\sigma\right)u(t_k)$$

- with $x[k] = x(t_k)$ and $u[k] = u(t_k)$, get **discrete-time system**

$$x[k+1] = A_\mathrm{d}x[k] + B_\mathrm{d}u[k] \qquad \begin{aligned} A_\mathrm{d} &= e^{AT} \\ B_\mathrm{d} &= \int_0^T e^{A\sigma}B\,\mathrm{d}\sigma \end{aligned}$$

- output equation is simply $y[k] = Cx[k] + Du[k]$, therefore

$$C_\mathrm{d} = C\,, \qquad D_\mathrm{d} = D$$

Comments on step-invariant discretization

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned} \quad \xmapsto{\text{“c2d”}} \quad \begin{aligned}x[k+1] &= A_{\mathrm{d}}x[k] + B_{\mathrm{d}}u[k]\\ y[k] &= C_{\mathrm{d}}x[k] + D_{\mathrm{d}}u[k]\end{aligned}$$

- notation: $P_{\mathrm{d}} = \texttt{c2d}(P)$
- we made **no approximations** – P_{d} is an *exact* description of P *at the sampling instants* (analogy: stroboscope)
- also called “zero-order hold” discretization
- if A is invertible, then a simplified formula for B_{d} is (**exercise**)

$$B_{\mathrm{d}} = A^{-1}(e^{AT} - I_n)B$$

Transfer function formula for step-invariant trans.

- what if we have a transfer function instead of state model?
- can also show that step-invariant transform given by

$$P_{\mathrm{d}}[z] = \frac{z-1}{z}\mathcal{Z}\left\{S_T\left(\mathscr{L}^{-1}\left\{\frac{P(s)}{s}\right\}\right)\right\}$$

1. compute inverse L.T. of $P(s)\frac{1}{s}$ (continuous-time step applied to P)
2. sample the resulting signal
3. take the z-transform of the resulting sequence
4. divide by $z/(z-1)$ (divide by discrete-time step)

Equivalent roads step-invariant transform

$$\begin{tikzcd}
\boxed{\begin{array}{r} \dot{x} = Ax + Bu \\ y = Cx + Du \end{array}} \arrow[r, bend left, dashed] \arrow[d, dashed] & \boxed{\begin{array}{r} x^{+} = A_{\mathrm{d}}x + B_{\mathrm{d}}u \\ y = C_{\mathrm{d}}x + D_{\mathrm{d}}u \end{array}} \arrow[d, dashed] \\
\boxed{P(s)} \arrow[r, bend right, dashed] & \boxed{P_{\mathrm{d}}[z]}
\end{tikzcd}$$

Example: first-order system

$$P(s) = \frac{\alpha}{s+\alpha} \quad \Longrightarrow \quad \begin{aligned} \dot{x} &= -\alpha x + u \\ y &= \alpha x \end{aligned} \quad \Longrightarrow \quad \begin{aligned} A &= -\alpha\,, \quad & B &= 1 \\ C &= \alpha\,, & D &= 0 \end{aligned}$$

compute discretization

$$\begin{aligned} A_{\mathrm{d}} &= e^{-\alpha T} & B_{\mathrm{d}} &= A^{-1}(e^{AT} - 1)B = \frac{1}{\alpha}\left(1 - e^{-\alpha T}\right) \\ C_{\mathrm{d}} &= \alpha & D_{\mathrm{d}} &= 0 \end{aligned}$$

therefore

$$\begin{aligned} x[k+1] &= e^{-\alpha T} x[k] + \frac{1 - e^{-\alpha T}}{\alpha} u[k] \\ y[k+1] &= \alpha x[k] \end{aligned} \qquad P_{\mathrm{d}}[z] = \frac{1 - e^{-\alpha T}}{z - e^{-\alpha T}}$$

Example: satellite attitude control model

$$J\ddot{\theta} = \tau$$

states: $x = (\theta, \dot{\theta})$

input: $u = \tau$

output: $y = \theta$

$$\Longrightarrow \quad \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

- need to compute e^{AT}

$$(sI - A)^{-1} = \begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s^2} \\ 0 & \frac{1}{s} \end{bmatrix}$$

$$e^{At} = \mathscr{L}^{-1}\{(sI - A)^{-1}\} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad e^{AT} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$

Example: satellite model contd.

$$B_\mathrm{d} = \int_0^T e^{A\sigma} B \,\mathrm{d}\sigma = \int_0^T \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} \mathrm{d}\sigma = \quad \cdots \quad = \begin{bmatrix} \frac{T^2}{2J} \\ \frac{T}{J} \end{bmatrix}$$

- therefore, we find the discrete-time model

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned} \quad \xmapsto{\texttt{c2d}} \quad \begin{aligned} x[k+1] &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} \frac{T^2}{2J} \\ \frac{T}{J} \end{bmatrix} u[k] \\ y[k] &= \begin{bmatrix} 1 & 0 \end{bmatrix} x[k] \end{aligned}$$

- exercise:

$$P_\mathrm{d}[z] = \frac{T^2}{2J} \frac{z+1}{(z-1)^2}$$

Stability of step-invariant discretization

- from before, eigenvalues of matrix e^A are $\{e^{(\text{eigenvalues of } A)}\}$
- said differently, if $\lambda \in \operatorname{eig}(A)$, then $e^\lambda \in \operatorname{eig}(e^A)$
- **claim:** P_d is internally stable if and only if P is internally stable
- *proof*: suppose $s = \alpha + \mathrm{j}\beta$ is an eigenvalue of A. Then $z = e^{sT} = e^{(\alpha+\mathrm{j}\beta)T} = e^{\alpha T}e^{\mathrm{j}\beta T}$ is an eigenvalue of $A_\mathrm{d} = e^{AT}$, and

$$|z| = |e^{\alpha T}| \times |e^{\mathrm{j}\beta T}| = e^{\alpha T}$$

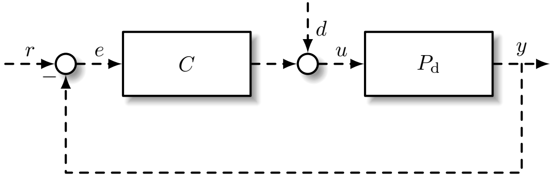
- since $T > 0$, we therefore have that

$$e^{\alpha T} < 1 \qquad \Longleftrightarrow \qquad \alpha < 0$$

- **moral:** step-invariant discretization preserves plant stability

Feedback stability of sampled-data systems

- the discrete-time system



represents the sampled-data system *at the sampling instants*.

- **question:** if the discrete-time system is feedback stable, can we say the original sampled-data system is also feedback stable?

- **answer:** yes (modulo a detail we will now discuss)

Pathological sampling

- particular sampling rates can cause an issue with `c2d`
- example

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad \Longrightarrow \qquad P(s) = \frac{1}{s^2+1}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

- compute `c2d` with sampling period $T = 2\pi$

$$A_{\mathrm{d}} = e^{A(2\pi)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad B_{\mathrm{d}} = A^{-1}(e^{A(2\pi)} - I_2)B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- therefore

$$P_{\mathrm{d}}[z] = C(zI_2 - A_{\mathrm{d}})^{-1}B_{\mathrm{d}} = 0 \quad (!!!)$$

Pathological sampling contd.

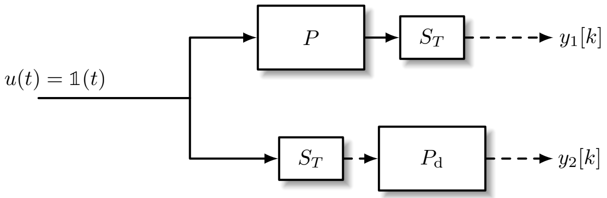
- we say a sampling period T is *pathological* if the number of poles of $P_{\mathrm{d}}[z]$ (counting multiplicities) is less than the number of poles of $P(s)$ (counting multiplicities)
- problem is "resonance" of sampling period with complex poles
- in practice, pathological sampling never happens (need very finely tuned sampling rate)

If the sampling period is not pathological, then the sampled-data system is feedback stable if and only if the discrete-time system is feedback stable

- for the rest of the course, all statements are implicitly prefixed with "assuming the sampling rate is not pathological"

What's with the name 'step-invariant'?

- let $P_{\mathrm{d}} = \texttt{c2d}(P)$, and consider the following experiment



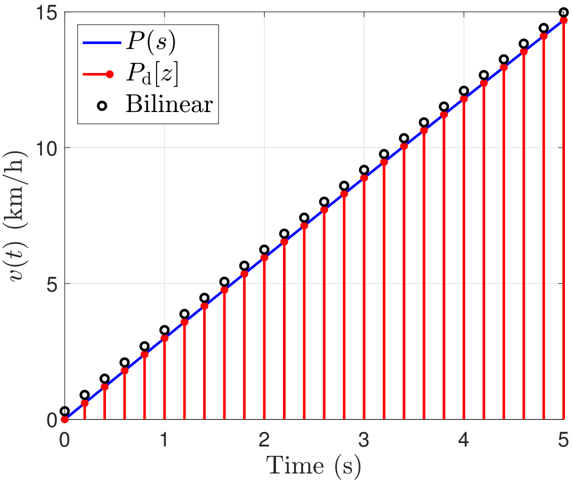
$$\begin{aligned} y_1[k] &= S_T\,(P\,\mathbb{1}(t)) \\ &= (S_T\,P\,H_T)\,\mathbb{1}[k] \\ &= P_{\mathrm{d}}\mathbb{1}[k] \end{aligned}$$

$$\begin{aligned} y_2[k] &= P_{\mathrm{d}}\,(S_T\,\mathbb{1}(t)) \\ &= P_{\mathrm{d}}\,\mathbb{1}[k] \end{aligned}$$

- step response of P_{d} is the same as sampled step response of P

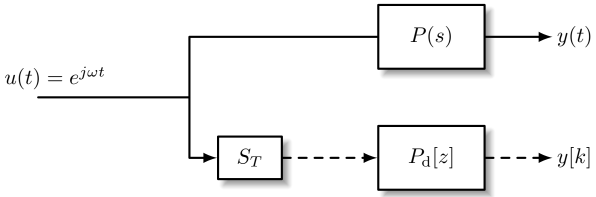
Example: step response of cruise control model

$(T = 0.2\text{s})$



Frequency response of discretized plant

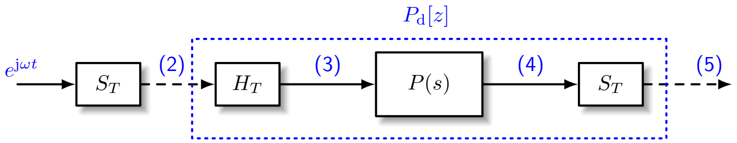
- we know the frequency response $P(j\omega)$ of the plant is very useful
- the discretized plant $P_{\mathrm{d}} = \texttt{c2d}(P)$ has frequency response $P_{\mathrm{d}}[e^{j\omega T}]$
- what is the relationship between these two? thought experiment:



- how do $y(t)$ and $y[k]$ compare in steady-state?

Frequency response of discretized plant contd.

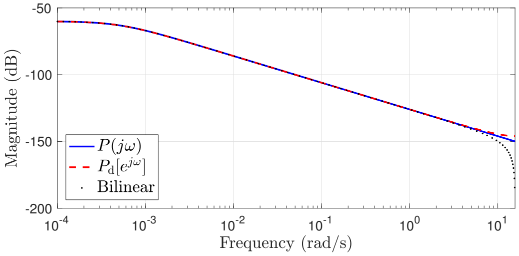
- top branch of diagram gives $y(t) = P(j\omega)e^{j\omega t}$, bottom branch gives



- signal (2) is $e^{j\omega kT}$ $\implies$ signal (5) is $P_\mathrm{d}[e^{j\omega T}] \cdot e^{j\omega kT}$
- at low frequencies where $\omega \ll \omega_\mathrm{Nyquist} = \omega_\mathrm{s}/2$, there is no aliasing
  - signal (3) is close to signal (1)
  - signal (4) is close to $P(j\omega)e^{j\omega t}$
  - signal (5) is close to $P(j\omega)e^{j\omega kT}$

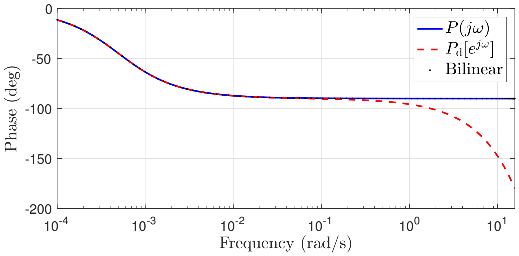
$$\implies \qquad P_\mathrm{d}[e^{j\omega T}] \simeq P(j\omega) \quad \text{at low frequencies}$$

Example: cruise control model ($T = 0.2$s)



$$P(s) = \frac{1/m}{s + b/m}$$

$$P_{\mathrm{d}}[z] = \frac{1}{b}\,\frac{1 - e^{-Tb/m}}{z - e^{-Tb/m}}$$



MATLAB commands

- computing step-invariant transformation (explicitly)

```
Ad = expm(A*T);
Bd = inv(A)*(Ad-eye(n));
```

- computing step-invariant transformation (built-in)

```
P = ss(A,B,C,D);
Pd = c2d(P,T,‘zoh’);
```

Additional references

- step-invariant discretization (ZOH)
    - Nielsen, Chapter 7
    - Franklin, Powell, & Workman, Chapter 4.3.3
    - Åström & Wittenmark, Chapter 3

Personal Notes

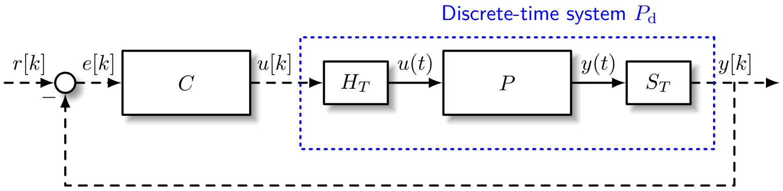
Personal Notes

Personal Notes

8. Direct design of digital controllers

- pole placement for discrete-time systems
- tracking reference signals (transfer functions)
- state-space control design
- state feedback, controllability, and pole placement
- observers, observability, and output feedback
- preview of optimal control

Sampled-data control problem



- plant and controller described by LTI transfer function or LTI state-space models

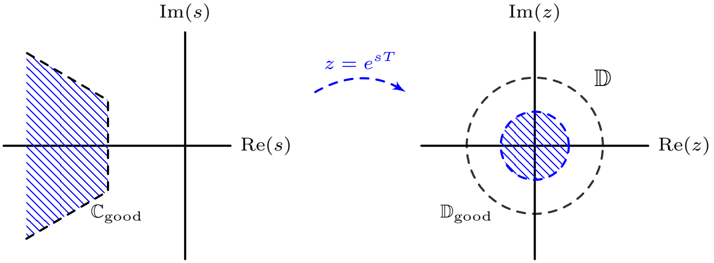
  Sampled-data design problem: *given plant P and a set of design specifications, design a discrete-time controller C such that the closed-loop sampled-data system meets the design specifications.*

Big picture for direct design

1. choose an initial sampling period T
   - possibly dictated by hardware, informed by design specs
2. discretize plant P to obtain P_{d}
   - *note:* P_{d} is implicitly a function of T
3. design — *by any method* — a discrete-time controller C for P_{d} to meet your design specs
   - *note:* C is also a function of T, since it was tuned for the plant P_{d}
4. simulate **sampled-data** system; if performance specs are not met, adjust controller design and/or decrease T

Pole placement for discrete-time systems

- in continuous-time, we selected a subset $\mathbb{C}_{\text{good}}$ of $\mathbb{C}_-$ in which to place the poles of the closed-loop system
- from step-invariant transformation, we saw that if $s \in \mathbb{C}$ is a eigenvalue/pole of P, then $z = e^{sT}$ is an eigenvalue/pole of P_{d}
- **idea:** we can map $\mathbb{C}_{\text{good}} \subset \mathbb{C}_-$ to a subset $\mathbb{D}_{\text{good}} \subset \mathbb{D}$



Pole placement procedure in discrete-time

1. use design specs to find $\mathbb{C}_{\text{good}}$

2. choose a set of symmetric pole locations

$$\Lambda = \{\lambda_1, \ldots, \lambda_{2n-1}\} \subset \mathbb{C}_{\text{good}}$$

3. map pole locations from $\mathbb{C}_{\text{good}}$ to $\mathbb{D}_{\text{good}}$

$$E = \{e^{\lambda_1 T}, \ldots, e^{\lambda_{2n-1} T}\}$$

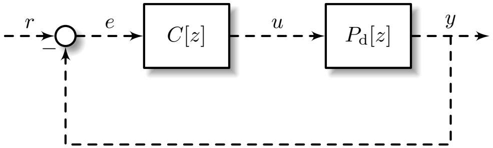
$$\Pi_{\text{des}}[z] = (z - e^{\lambda_1 T}) \cdots (z - e^{\lambda_{2n-1} T})$$

4. solve discrete-time pole placement problem for controller

$$P_{\text{d}}[z] = \frac{N_{\text{p}}[z]}{D_{\text{p}}[z]}, \qquad C[z] = \frac{N_{\text{c}}[z]}{D_{\text{c}}[z]} = \frac{g_{n-1}z^{n-1} + \cdots + g_1 z + g_0}{f_{n-1}z^{n-1} + \cdots + f_1 z + f_0}$$

$$\Pi[z] = N_{\text{p}} N_{\text{c}} + D_{\text{p}} D_{\text{c}}$$

Tracking reference signals



Tracking problem: Given a plant $P_\mathrm{d}[z]$, design a controller $C[z]$ such that the closed-loop system is feedback stable and

$$\lim_{k\to\infty}(r[k]-y[k])=0\,.$$

Internal Model Principle: Assume $P_\mathrm{d}[z]$ is strictly proper, $C[z]$ is proper, and that the closed-loop system is feedback stable. Then $\lim_{k\to\infty}(r[k]-y[k])=0$ if and only if $P_\mathrm{d}[z]C[z]$ contains an internal model of the unstable part of $r[z]$.

Example: step tracking

- for a step $r[k] = \mathbb{1}[k]$, $r[z] = \frac{z}{z-1}$
- there are three cases

  (i) if $P_{\mathrm{d}}[z]$ has a zero at $z = 1$, then step-tracking is not possible

  (ii) if $P_{\mathrm{d}}[z]$ has a pole at $z = 1$, $C[z]$ can be any stabilizing controller

  (iii) if $P_{\mathrm{d}}[z]$ does not have a pole or zero at $z = 1$, then $C[z]$ must have a pole at $z = 1$

- pole at $z = 1$ is "integral control" (running sum of tracking error)

$$u[k] = \mathcal{Z}^{-1}\{u[z]\} = \mathcal{Z}^{-1}\left\{\frac{z}{z-1}e[z]\right\} = \sum_{\ell=0}^{k} e[\ell]$$

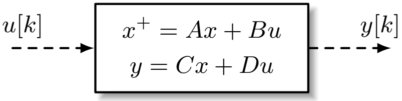
Control design for LTI state-space models

- all MIMO (multi-input multi-output) control techniques are based on state-space models
- allows for use of powerful computational techniques, optimization
- we will keep things as simple as possible, introduce the basic ideas for single-input single-output discrete-time models

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] \\ y[k] &= Cx[k] + Du[k] \end{aligned}$$

- nearly identical theory for continuous-time state models

State-space regulation problem



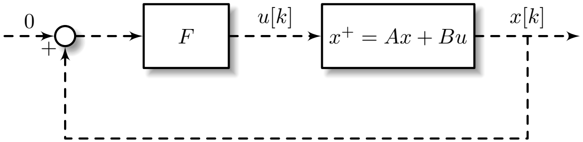
- **regulation problem:** find a controller such that $x[k] \to 0$ for any choice of initial condition $x[0]$
- in other words: closed-loop system is *internally asymptotically stable* (recall that this implies feedback stability of the closed-loop system)
- designing a regulating controller ("regulator") can be extended to reference tracking

Regulation by state feedback

- simplest case: assume we can measure the *entire state* $x[k]$, and we try to design a *state-feedback controller*

$$u[k] = Fx[k]\,, \qquad F = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}$$

"state-feedback gain"



State feedback and controllability

- closed-loop system is

$$x^+ = Ax + Bu = Ax + BFx = \underbrace{(A + BF)}_{A_{\mathrm{cl}}} x$$

- to achieve internal asymptotic stability, need to find F such that

$$\mathrm{eig}(A + BF) \subset \mathbb{D}$$

- when can we find such an F? Need new idea of **controllability**

- **definition:** a state-space system is *controllable* if from every initial state $x[0] \in \mathbb{R}^n$, there is a sequence of control inputs $\{u[0], u[1], \ldots, u[n-1]\}$ such that $x[n] = 0$

- **idea:** can choose inputs to "deliver" the state to the origin

Controllability contd.

- let's think about how system state changes over time

$$
\begin{aligned}
x[1] &= Ax[0] + Bu[0] \\
x[2] &= Ax[1] + Bu[1] = A^2x[0] + Bu[1] + ABu[0] \\
&\vdots \\
x[n] &= A^n x[0] + Bu[n-1] + ABu[n-2] + \cdots + A^{n-1}Bu[0]
\end{aligned}
$$

- since we want $x[n] = 0$, we can also write this as

$$
-A^n x[0] = \underbrace{\begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}}_{\text{Controllability matrix } W_c} \underbrace{\begin{bmatrix} u[n-1] \\ u[n-2] \\ \vdots \\ u[0] \end{bmatrix}}_{:= u_c}
$$

Controllability contd.

- required input sequence is determined by linear equation

$$-A^n x[0] = W_c u_c$$

- if $\mathrm{rank}(W_c) = n$, we can solve for u_c for any $x[0]$

  conclusion: *a state-space system is controllable if the controllability matrix*

$$W_c = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \in \mathbb{R}^{n\times n}$$

  has rank n.

- vectors $\{B, AB, A^2B, \ldots, A^{n-1}B\}$ tell us about the directions in state-space that we can "push" the system using our input

State-feedback control and pole placement

- controllability lets us do "pole" placement for state-space systems
- suppose we have a desired (symmetric w.r.t. real axis) set of closed-loop eigenvalues

$$\{z_1, \ldots, z_n\} \subset \mathbb{D}$$

for closed-loop system $x^+ = (A + BF)x$

Pole-placement theorem for state feedback: *there exists a state-feedback matrix $F \in \mathbb{R}^{1\times n}$ such that $A + BF$ has the desired eigenvalues if and only if the system is controllable*

Calculating state-feedback gain F

- to calculate $F = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}$, there are two approaches
  1. let $\Pi_{\text{des}}[z] = (z - z_1) \cdots (z - z_n)$ be the desired characteristic polynomial, and match coefficients of z from the equation

     $$\Pi[z] = \det(zI - (A + BF)) = \Pi_{\text{des}}[z]$$

  2. use *Ackerman's Formula*

     $$F = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} W_{\text{c}}^{-1} \Pi_{\text{des}}[A]$$

     where

     $$\Pi_{\text{des}}[A] = (A - z_1 I) \cdots (A - z_n I)$$

- both implemented in `MATLAB` as `place` and `acker`

Example: satellite attitude control

- discretized model of satellite $J\ddot{\theta} = \tau$

$$x[k+1] = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} \frac{T^2}{2J} \\ \frac{T}{J} \end{bmatrix} u[k]$$
$$y[k] = \begin{bmatrix} 1 & 0 \end{bmatrix} x[k]$$

- system is internally unstable (repeated eigenvalue at $z = 1$)
- **objective:** stabilize $x = 0$ using state-feedback
- system is controllable for all sampling periods $T > 0$

$$W_{\mathrm{c}} = \begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} \frac{T^2}{2J} & \frac{3T^2}{2J} \\ \frac{T}{J} & \frac{T}{J} \end{bmatrix}, \quad \det(W_{\mathrm{c}}) = -\frac{T^3}{J^2} \neq 0$$

Example: satellite attitude control contd.

- we must select two eigenvalues for $A + BF$, let them be z_1, z_2

$$\Pi_{\text{des}}[z] = (z - z_1)(z - z_2) = z^2 + (-z_1 - z_2)z + z_1 z_2$$

- we can form the closed-loop system matrix

$$A + BF = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2J} \\ \frac{T}{J} \end{bmatrix} \begin{bmatrix} f_1 & f_2 \end{bmatrix} = \begin{bmatrix} 1 + \frac{f_1 T^2}{2J} & T + \frac{f_2 T^2}{2J} \\ \frac{f_1 T}{J} & 1 + \frac{f_2 T}{J} \end{bmatrix}$$

- characteristic polynomial $\Pi[z] = \det(zI - (A + BF))$

$$\Pi[z] = z^2 + \left(\frac{f_1 T^2 + 2 f_2 T - 4J}{2J} \right) z + \left(\frac{f_1 T^2 - 2 f_2 T + 2J}{2J} \right)$$

Example: satellite attitude control contd.

- comparing coefficients, we get two simultaneous equations

$$-(z_1+z_2) = \frac{f_1T^2 + 2f_2T - 4J}{2J} \qquad z_1z_2 = \frac{f_1T^2 - 2f_2T + 2J}{2J}$$

  which we can solve to find f_1 and f_2:

$$f_1 = -\frac{J}{T}(1 - z_1 - z_2 + z_1z_2)\,, \qquad f_2 = -\frac{J}{2T}(3 - z_1 - z_2 - z_1z_2)$$

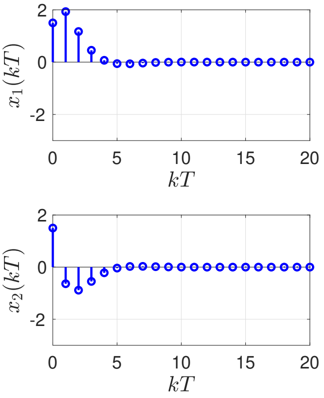
  and our controller is then

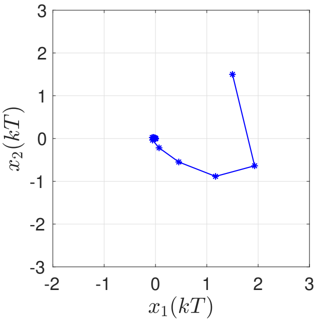
$$u[k] = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix}$$

- since states x_1 and x_2 are position and velocity, f_1 is the feedback gain on position, while f_2 is the gain on velocity (PD control)

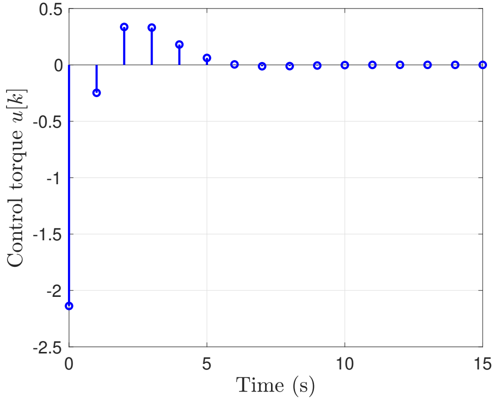
Example: satellite attitude control contd.

- $z_{1,2} = 0.4 \pm 0.3\mathrm{j}$, $x[0] = (1.5, 1.5)$, $J = 1\,\mathrm{kg}\cdot\mathrm{m}^2$, $T = 1\,\mathrm{s}$





Example: satellite attitude control contd.



Deadbeat control

- deadbeat design is fastest (most aggressive) design possible
- we place all eigenvalues of $A + BF$ at the origin $z = 0$ (*nilpotency*)

$$\det(zI - (A + BF)) = z^n$$

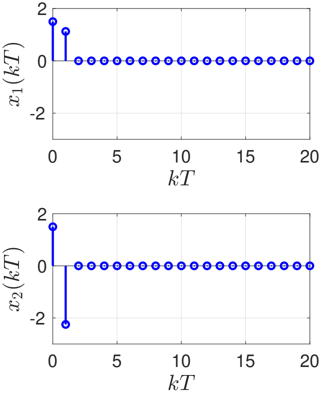
- **fact:** if $M \in \mathbb{R}^{n\times n}$ is nilpotent, there exists $p \leq n$ s.t. $M^p = 0$
- **recall:** solution to state model from initial condition $x[0]$ is

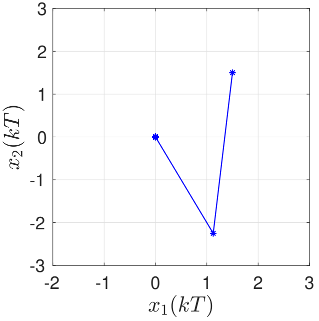
$$x[k+1] = (A + BF)x[k] \qquad \Longrightarrow \qquad x[k] = (A + BF)^k x[0]$$

- therefore, $x[p] = 0$; state goes to the origin after p time steps!

Example: satellite attitude control (deadbeat tuning)

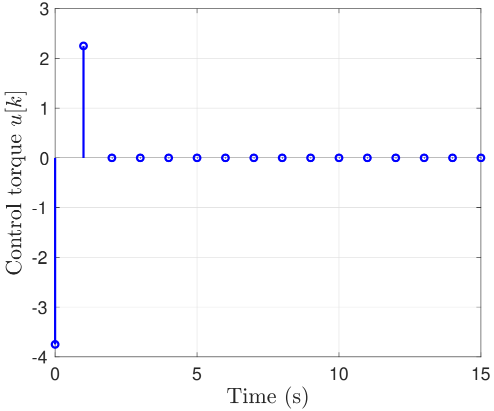
- $z_{1,2} = 0$, $x[0] = (1.5, 1.5)$, $J = 1\,\mathsf{kg}\cdot\mathsf{m}^2$, $T = 1\,\mathsf{s}$





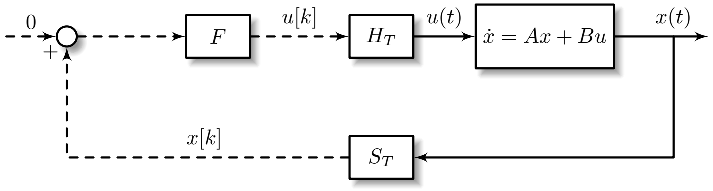
- deadbeat tuning results in convergence in 2 time steps, but with a larger swing in velocity (x_2) compared to previous tuning

Example: satellite attitude control (deadbeat tuning)



Intersample ripple

- suppose we have designed a state-feedback F for discretized plant
- what happens when implemented in a sampled-data system?

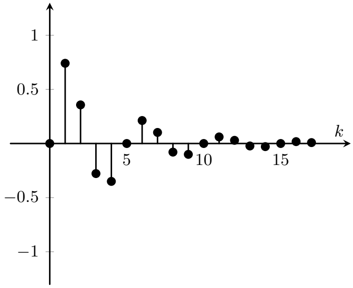


- we know that $x[k] = x(kT) \to 0$
- **question:** what is happening *between* samples?

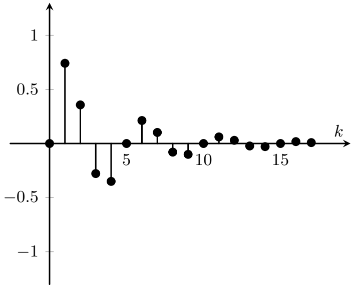
Intersample ripple contd.

- there are two basic possibilities

$\underline{x(t)\text{ stable}}$

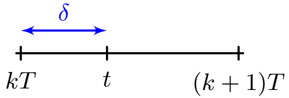


$\underline{x(t)\text{ unstable}}$



Intersample ripple contd.

- **claim:** $x(t)$ converges to zero (no sustained oscillations)
- **proof:** consider a sampling interval $[kT, (k+1)T]$



solution of state-model at time t is

$$\begin{aligned} x(t) &= e^{A(t-kT)}x(kT) + \int_{kT}^{t} e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\tau \\ &= e^{A(t-kT)}x(kT) + \int_{kT}^{t} e^{A(t-\tau)}B\mathrm{d}\tau\, u(kT) \end{aligned}$$

Intersample ripple contd.

- change variables $\sigma = t - \tau$

$$x(t) = e^{A(t-kT)}x(kT) + \int_{t-kT}^{0} e^{A\sigma} B(-\mathrm{d}\sigma)\, u(kT)$$
$$= e^{A\delta}x(kT) + \int_{0}^{\delta} e^{A\sigma} B\, \mathrm{d}\sigma\, u(kT)$$

- but $u(kT) = Fx(kT)$, therefore

$$x(t) = \left(e^{A\delta} + \int_{0}^{\delta} e^{A\sigma}\, \mathrm{d}\sigma\, BF \right) x(kT) = M_\delta x(kT)$$

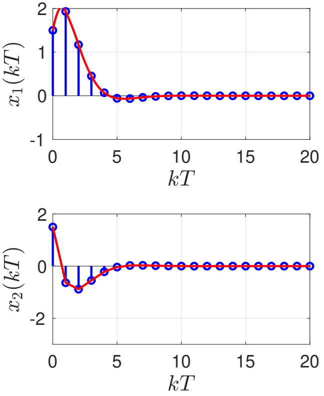
  or simply $x(kT + \delta) = M_\delta x(kT)$.

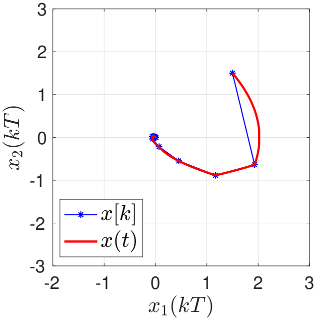
- $x(kT) \to 0$, and therefore $x(kT + \delta) \to 0$

- but δ was arbitrary, therefore $x(t) \to 0$ as $t \to \infty$

Example: satellite attitude control contd.

- $z_{1,2} = 0.4 \pm 0.3\mathrm{j}$, $x[0] = (1.5, 1.5)$, $J = 1\,\mathrm{kg}\cdot\mathrm{m}^2$, $T = 1\,\mathrm{s}$

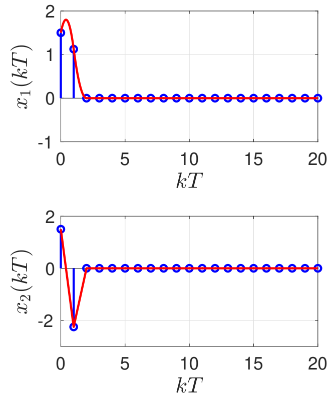


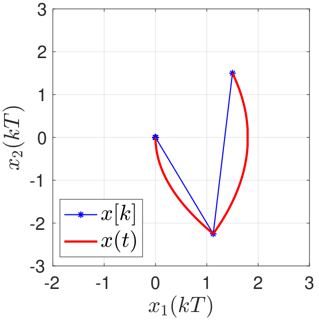


- **takeaway:** everything is fine between the samples

Example: satellite attitude control contd.

- $z_{1,2} = 0$, $x[0] = (1.5, 1.5)$, $J = 1\,\mathsf{kg}\cdot\mathsf{m}^2$, $T = 1\,\mathsf{s}$





- **takeaway:** everything is fine between the samples

Output feedback for state-space systems

- usually, we cannot measure the entire state $x[k] \in \mathbb{R}^n$
  - some variables in model not easy to measure
  - might require too many sensors
- when we design transfer function controllers, we only use the measured output $y[k]$, so state feedback looks quite restrictive
- the solution to this problem is to use an *observer*, which takes the output $y[k]$ and produces an *estimate* $\hat{x}[k]$ of the state $x[k]$
- we can then *use the estimated state* for state-feedback control

$$u[k] = F\hat{x}[k]$$

Observers for state-space systems

- naively, knowing $y[k]$ and $u[k]$, we could try to solve equation

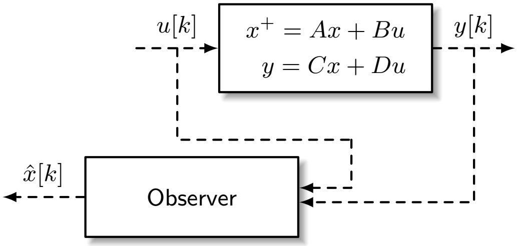
$$y[k] = Cx[k] + Du[k]$$

  for current state $x[k]$ at each time k (not a great idea, why?)

- if we apply an input and look at the corresponding *time series* of the output, we will be able to infer something more information about the state

- **key idea:** use the *sequence* of inputs and measurements over time

The Luenberger observer



- the observer is a discrete-time LTI system with state $\hat{x}[k]$

$$
\begin{aligned}
\hat{x}^+ &= A\hat{x} + Bu + L(\hat{y} - y) \\
\hat{y} &= C\hat{x} + Du
\end{aligned}
\qquad\qquad
L = \begin{bmatrix} l_1 \\ \vdots \\ l_n \end{bmatrix} \in \mathbb{R}^{n\times 1}
$$

"observer gain"

Analysis of Luenberger observer

$$\hat{x}^+ = \underbrace{A\hat{x} + Bu}_{\text{prediction}} + \underbrace{L(\hat{y} - y)}_{\text{correction}}$$

$$\hat{y} = \underbrace{C\hat{x} + Du}_{\text{prediction}}$$

$$L = \begin{bmatrix} l_1 \\ \vdots \\ l_n \end{bmatrix} \in \mathbb{R}^{n\times 1}$$

- let $\varepsilon[k] = x[k] - \hat{x}[k]$ be the estimation error

$$\begin{aligned} \varepsilon^+ &= (Ax + Bu) - (A\hat{x} + Bu + L(\hat{y} - y)) \\ &= A(x - \hat{x}) - L(C\hat{x} + Du - Cx - Du) \\ &= A(x - \hat{x}) + LC(x - \hat{x}) \end{aligned}$$

$$\varepsilon^+ = (A + LC)\varepsilon$$

- error will satisfy $\varepsilon[k] \to 0$ from every I.C. $\varepsilon[0]$ if we choose L s.t.

$$\mathrm{eig}(A + LC) \subset \mathbb{D}$$

Observability

- need new idea of **observability**
- **definition:** a system is *observable* if from knowledge of the inputs $\{u[0], u[1], \ldots, u[n-1]\}$ and outputs $\{y[0], y[1], \ldots, y[n-1]\}$ up to time $n-1$, we can uniquely determine the state $x[n]$ at time n
- idea is that output contains "enough" information about the state

$$\begin{aligned}
y[0] &= Cx[0] + Du[0] \\
y[1] &= Cx[1] + Du[1] = \cdots = CAx[0] + CBu[0] + Du[1] \\
y[2] &= Cx[2] + Du[2] = \cdots = CA^2x[0] + CABu[0] + CBu[1] + Du[2] \\
&\vdots \\
y[n-1] &= CA^{n-1}x[0] + CA^{n-2}Bu[0] + \cdots + CBu[n-2] + Du[n-1]
\end{aligned}$$

Observability contd.

- we can write this as a linear system of equations for $x[0]$

$$\underbrace{\begin{bmatrix} y[0] \\ y[1] \\ \cdots \\ y[n-1] \end{bmatrix}}_{:=y_\mathrm{o}} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{:=W_\mathrm{o}} x[0] + \underbrace{\begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{n-2}B & \cdots & CB & D \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix}}_{:=\xi}$$

  or simply

$$y_\mathrm{o} - \xi = W_\mathrm{o}\, x[0]$$

- if we could solve for $x[0] \in \mathbb{R}^n$, then we can immediately find $x[n]$ as

$$x[n] = A^n x[0] + \sum_{j=0}^{n-1} A^{n-j-1} B u[j]$$

Observability contd.

- if $\mathrm{rank}(W_\mathrm{o}) = n$, we can uniquely solve for the linear equation

$$y_\mathrm{o} - \xi = W_\mathrm{o}\, x[0]$$

  for $x[0]$ and then uniquely determine $x[n]$!

- **conclusion:** a state-space system is observable if the *observability matrix*

$$W_\mathrm{o} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \in \mathbb{R}^{n\times n}$$

  has rank n.

Observability and observer design

- observability guarantees we can design an observer
- suppose we have a desired (symmetric w.r.t. real axis) set of eigenvalues

$$\{\zeta_1, \ldots, \zeta_n\} \subset \mathbb{D}$$

  for estimation error dynamics $\varepsilon^+ = (A + LC)\varepsilon$

Pole-placement theorem for observer design: *There exists an observer gain matrix $L \in \mathbb{R}^{n\times 1}$ such that $A + LC$ has the desired eigenvalues if and only if the system is observable*

Calculating the observer gain L

- to calculate $L = \begin{bmatrix} l_1 & l_2 & \cdots & l_n \end{bmatrix}^\mathsf{T}$, there are two approaches

  1. let $\Pi_{\text{des}}[z] = (z - \zeta_1)\cdots(z - \zeta_n)$ be the desired characteristic polynomial, and match coefficients of z from the equation

     $$\Pi[z] = \det(zI - (A + LC)) = \Pi_{\text{des}}[z]$$

  2. use *Ackerman's Formula*

     $$L = -\Pi_{\text{des}}[A]\, W_{\text{o}}^{-1} \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}^\mathsf{T}$$

     where

     $$\Pi_{\text{des}}[A] = (A - \zeta_1 I)\cdots(A - \zeta_n I)$$

- both implemented in `MATLAB` as `place` and `acker`

Example: observer for satellite system

- discretized model of satellite $J\ddot{\theta} = \tau$

$$\begin{aligned} x[k+1] &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} \frac{T^2}{2J} \\ \frac{T}{J} \end{bmatrix} u[k] \\ y[k] &= \begin{bmatrix} 1 & 0 \end{bmatrix} x[k] \end{aligned}$$

- **objective:** design observer to estimate $x[k]$ from $y[k]$

- system is observable for all sampling periods T:

$$W_{\mathrm{o}} = \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & T \end{bmatrix}, \quad \det(W_{\mathrm{o}}) = T \neq 0$$

Example: observer for satellite system contd.

- we must select two eigenvalues for $A + LC$, let them be ζ_1, ζ_2

$$\Pi_{\text{des}}[z] = (z - \zeta_1)(z - \zeta_2) = z^2 + (-\zeta_1 - \zeta_2)z + \zeta_1\zeta_2$$

- we can form the system matrix for the estimation error

$$A + LC = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 + l_1 & T \\ l_2 & 1 \end{bmatrix}$$

- characteristic polynomial $\Pi[z] = \det(zI - (A + LC))$

$$\Pi[z] = z^2 + (-2 - l_1)z + (1 + l_1 - l_2 T)$$

Example: observer for satellite system contd.

- comparing coefficients, we get two simultaneous equations

$$-(\zeta_1 + \zeta_2) = -2 - l_1 \qquad \zeta_1\zeta_2 = 1 + l_1 - l_2 T$$

  which we can solve to find l_1 and l_2:

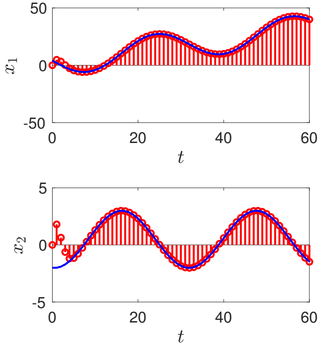
$$l_1 = \zeta_1 + \zeta_2 - 2\,, \qquad l_2 = \frac{1}{T}(\zeta_1 + \zeta_2 - \zeta_1\zeta_2 - 1)$$

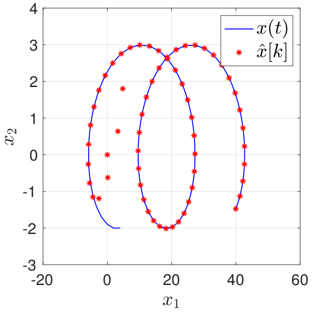
- our observer is therefore

$$\begin{aligned}\hat{x}^+ &= \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}\hat{x} + \begin{bmatrix}\frac{T^2}{2J}\\ \frac{T}{J}\end{bmatrix}u + \begin{bmatrix}l_1\\ l_2\end{bmatrix}(\hat{y} - y)\\ \hat{y} &= \begin{bmatrix}1 & 0\end{bmatrix}\hat{x}\end{aligned}$$

Example: observer for satellite system contd.

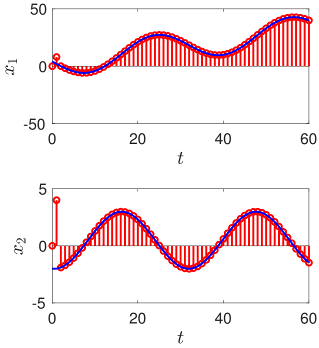
- satellite system subject to sinusoidal force input, $\zeta_{1,2} = 0.4 \pm 0.3\mathrm{j}$
- initial conditions $x[0] = (4, -2)$, $\hat{x}[0] = (0, 0)$

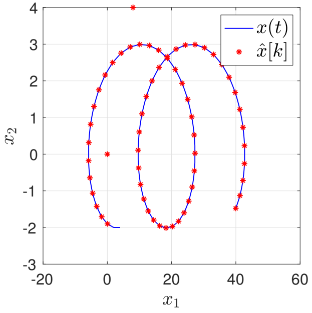




Example: observer for satellite system contd.

- satellite system subject to sinusoidal toruqe input, $\zeta_{1,2} = 0$
- initial conditions $x[0] = (4, -2)$, $\hat{x}[0] = (0, 0)$





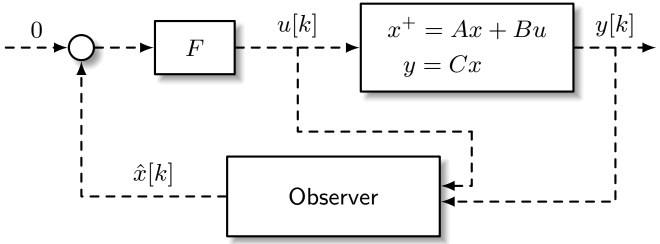
- **takeaway:** estimates converge in 2 steps with deadbeat tuning, but with large error at time step $k = 1$ compared to previous tuning

Comments on state feedback / observer design

- for state-feedback controllers, aggressive pole placements (i.e., closer to the origin) will give very stable system, but
  - control effort will be very large
  - state variables may have poor transient response (overshoot)
- for observers, aggressive pole placements (i.e., closer to the origin) will make estimation error go to zero quickly, but
  - measurement noise will be amplified
  - estimation error may have poor transient response (overshoot)
- usually, want to select *least* aggressive pole placements which still give a response meeting specs; this is a trial-and-error procedure

Combining state-feedback and observers

- combination of observer and state-feedback controller can be used to solve the state-space regulation problem



- control input produced using *estimated state*

$$u[k] = F\hat{x}[k]$$

Output feedback control

- observer and state-feedback matrix F form dynamic controller

$$\begin{aligned}\hat{x}^+ &= A\hat{x} + Bu + L(\hat{y} - y)\\ u &= F\hat{x}\end{aligned}$$

  where $\mathrm{eig}(A + BF) \subset \mathbb{D}$ and $\mathrm{eig}(A + LC) \subset \mathbb{D}$

- substituting for $\hat{y} = C\hat{x}$ and $u = F\hat{x}$, controller has state model

$$\begin{aligned}\hat{x}^+ &= (A + BF + LC)\hat{x} - Ly\\ u &= F\hat{x}\end{aligned}$$

  with input y and output u

Output feedback control cpntd.

- observer + state feedback controller:

$$\begin{aligned}\hat{x}^{+} &= (A + BF + LC)\hat{x} - Ly \\ u &= F\hat{x}\end{aligned}$$

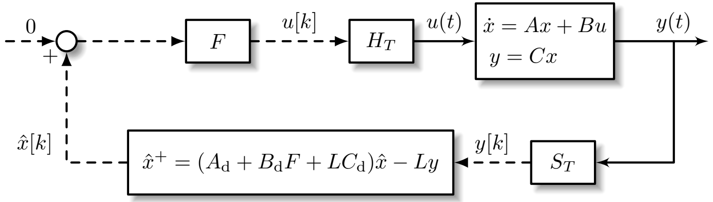
  with input y and output u

- controller has same order as plant

- if you prefer transfer functions, controller has the TF

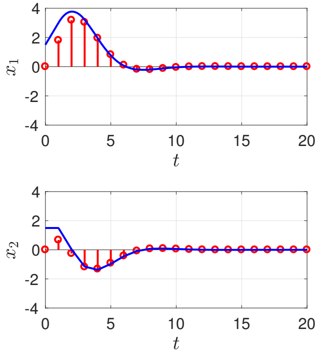
$$\frac{u[z]}{y[z]} = -F(zI_n - (A + BF + LC))^{-1}L$$

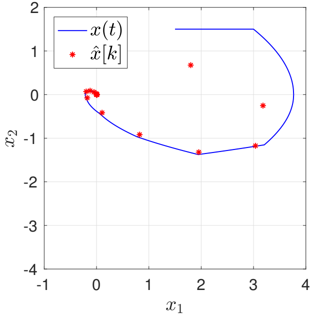
Sampled-data implementation



Example: output feedback for satellite system contd.

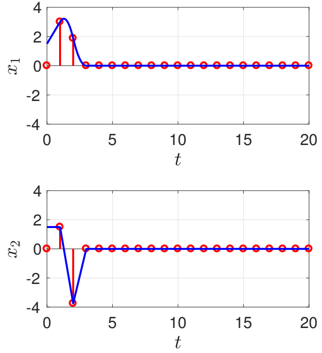
- observer and feedback gain with poles $z_{1,2} = 0.4 \pm 0.3\mathrm{j}$

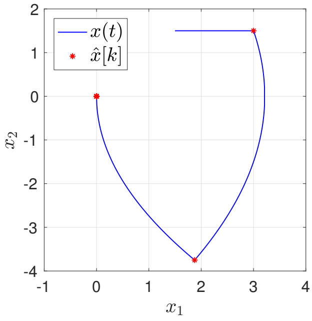




Example: output feedback for satellite system contd.

- observer and feedback gain with deadbeat poles $z_{1,2} = 0$





- same story as before: deadbeat tuning is more "aggressive": it gives faster convergence, but with worse transients

Incorporating reference signals

- output reference r included by slight twist on previous controller
- if we want $y = r$ in steady-state, need to find $\bar{x}$ and $\bar{u}$ such that

$$\begin{aligned} \bar{x} &= A\bar{x} + B\bar{u} \\ r &= C\bar{x} + D\bar{u} \end{aligned} \qquad \Longleftrightarrow \qquad \begin{bmatrix} A - I_n & B \\ C & D \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} = \begin{bmatrix} \mathbb{0} \\ r \end{bmatrix}$$

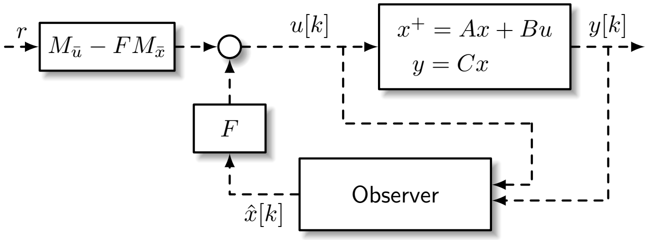
  therefore

$$\begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} = \begin{bmatrix} A - I_n & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{0} \\ 1 \end{bmatrix} r := \begin{bmatrix} M_{\bar{x}} \\ M_{\bar{u}} \end{bmatrix} r$$

- now use modified control law

$$u[k] = \bar{u} + F(\hat{x}[k] - \bar{x}) = F\hat{x} + (M_{\bar{u}} - FM_{\bar{x}})r$$

Incorporating reference signals contd.

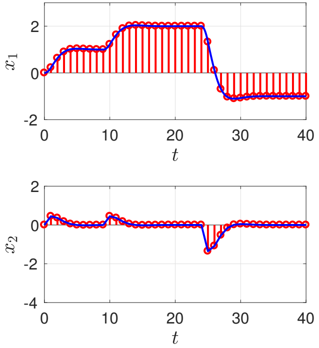


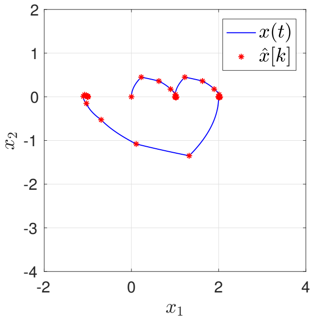
- this solution can work quite well; a more sophisticated approach would introduce an integral state (see problem set for details)

$$x_I[k+1] = x_I[k] + \underbrace{(Cx[k] - r[k])}_{\text{tracking error}}$$

Example: reference tracking for satellite system contd.

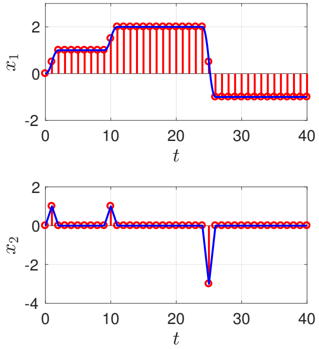
- observer and feedback gain with poles $z_{1,2} = 0.4 \pm 0.3\text{j}$

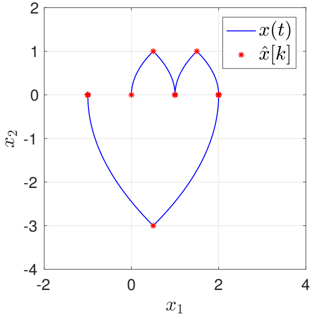




Example: reference tracking for satellite system contd.

- observer and feedback gain with deadbeat poles $z_{1,2} = 0$





- as expected, deadbeat tuning gives fast response at the cost of worse transient behaviour

Preview of optimal control

- picking locations for poles/eigenvalues is hard work, and for large control problems is simply not a realistic solution
- in *optimal control*, we specify a "cost" that we want to minimize, and find the controller that minimizes that cost
- problem of selecting poles is transformed into the problem of designing a cost; what should our cost include?
  - for regulation, we want $x[k]$ and $y[k]$ to stay small
  - to minimize control effort, $u[k]$ should stay small also
- therefore, cost should penalize *both* $u[k]$ and $y[k]$

The linear quadratic regulator

- the linear quadratic regulator (LQR) cost is

$$\text{Cost/Loss Function} = J_{\mathrm{LQR}} := \sum_{k=0}^{\infty} \left(y[k]^2 + \rho u[k]^2\right)$$

  where $\rho > 0$ is a tuning parameter

- if J_{LQR} is finite, then $y[k]$ and $u[k]$ converge to zero (duh)

- cost captures transient of output and transient of control input
  - if ρ is big, we are saying that control effort is expensive
  - if ρ is small, we are saying that control effort is inexpensive

- **goal:** find control sequence $\{u[0], u[1], \ldots\}$ that minimizes J_{LQR}

Solution of LQR problem

- if state-space system is controllable and observable, LQR problem can be solved, and solution is a *state feedback*

$$u[k] = F_{\mathrm{LQR}} x[k]$$

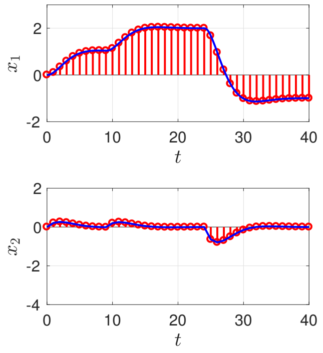
- feedback matrix $F_{\mathrm{LQR}} \in \mathbb{R}^{1\times n}$ computed in `MATLAB` as

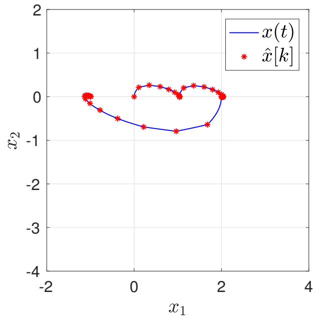
```
F = -lqr(sys, C'*C, rho, zeros(n,1));
```

- control design problem reduced to picking a single constant ρ

Example: LQR feedback for satellite

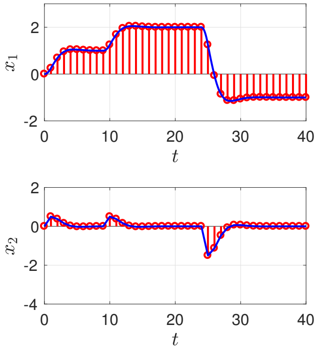
- deadbeat observer with LQR feedback gain, $\rho = 10$

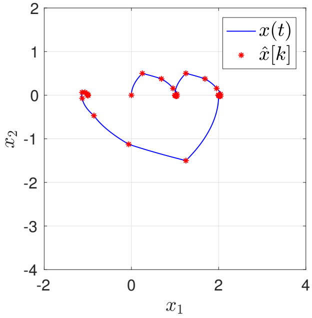




Example: LQR feedback for satellite

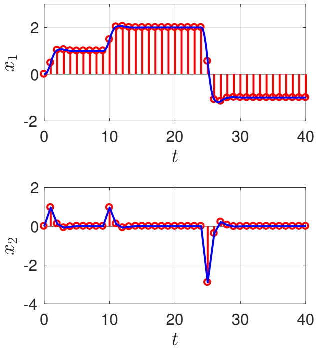
- deadbeat observer with LQR feedback gain, $\rho = 1$

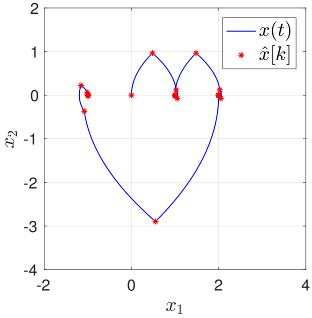




Example: LQR feedback for satellite

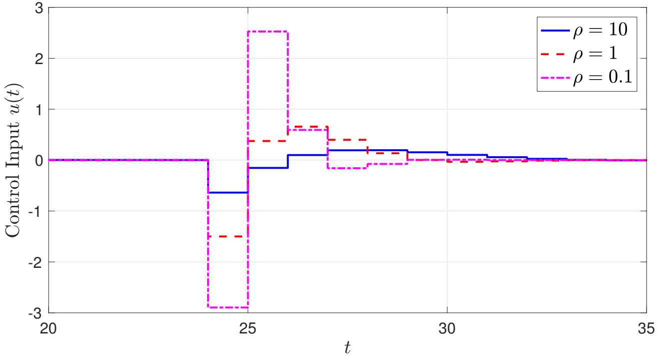
- deadbeat observer with LQR feedback gain, $\rho = 0.1$



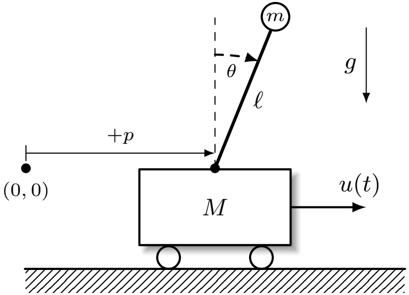


Example: LQR feedback for satellite

- comparison of control efforts during third step change



Modelling of simple pendulum on cart



- $p =$ position of cart
- $r_{\text{tip}} =$ pendulum tip

$$r_{\text{tip}} = \begin{bmatrix} p + \ell \sin\theta \\ \ell\cos\theta \end{bmatrix}$$

$$\dot{r}_{\text{tip}} = \begin{bmatrix} \dot{p} + \ell\dot{\theta}\cos\theta \\ -\ell\dot{\theta}\sin\theta \end{bmatrix}$$

kinetic energy:
$$\begin{aligned} T &= \frac{1}{2}M\dot{p}^2 + \frac{1}{2}m\|\dot{r}_{\text{tip}}\|_2^2 \\ &= \frac{1}{2}(M+m)\dot{p}^2 + m\ell\dot{p}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 \end{aligned}$$

potential energy:
$$U = U_{\text{tip}} = -mg\ell(1-\cos\theta)$$

Modelling of simple pendulum on cart

- the *Lagrangian* L is kinetic minus potential

$$L = \frac{1}{2}(M+m)\dot{p}^2 + m\ell\dot{p}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 + mg\ell(1-\cos\theta)$$

- equations of motion given by two *Lagrange equations*

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{p}}\right) - \frac{\partial L}{\partial p} = u\,, \qquad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$$

which yield

$$\boxed{\begin{aligned}(M+m)\ddot{p} + m\ell\cos(\theta)\ddot{\theta} - m\ell\dot{\theta}^2\sin(\theta) &= u\\ m\ell^2\ddot{\theta} + m\ell\cos(\theta)\ddot{p} - mg\ell\sin(\theta) &= 0\end{aligned}}$$

Design example: pendulum on cart

- equations can be transformed into

$$\begin{bmatrix}\ddot{p}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}M+m & m\ell\cos(\theta)\\ m\ell\cos(\theta) & m\ell^2\end{bmatrix}^{-1}\begin{bmatrix}m\ell\dot{\theta}^2\sin(\theta)+u\\ mg\ell\sin(\theta)\end{bmatrix}$$

  can can subsequently be put into state-space form

$$\dot{x} = f(x,u)\,, \quad y = h(x,u)$$

  with $x = (p, \dot{p}, \theta, \dot{\theta})^{\mathsf{T}}$, $y =$ position of cart

- **control objectives:**

  1. stabilize upright position $\theta = 0$ with constant position $p = 0$
  2. track constant position references

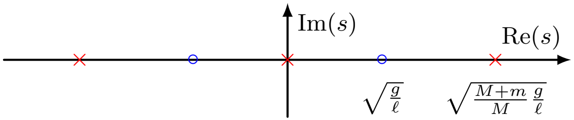
Design example: pendulum on cart

- **exercise:** linearized model around $(p, \dot{p}, \theta, \dot{\theta}) = (0, 0, 0, 0)$ is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{m}{M}g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{M+m}{M}\frac{g}{\ell} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{M\ell} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x$$

- poles and zeros of transfer function



Design example: pendulum on cart

- parameters are $m = 1\,\text{kg}$, $M = 0.5\,\text{kg}$, $\ell = 1\,\text{m}$
- take $T = 0.05$s, and build discrete-time model

```
pend_ct = ss(A,B,C,D);
pend_dt = c2d(pend_ct,T);
```

- can check system is controllable and observable

```
W_c = ctrb(Ad,Bd);
W_o = obsv(Ad,Cd);
```

- design controller and observer gains

```
rho=0.5;
F = -lqr(pend_dt, Cd'*Cd, rho, zeros(n,1));
L = acker(Ad', -Cd', [0;0;0;0])';
```

Design example: pendulum on cart

- control signal $u[k]$ is given by

$$u[k] = F\hat{x}[k] + (M_{\bar{u}} - FM_{\bar{x}})r[k]$$

- observer state-space model is therefore

$$\begin{aligned}\hat{x}^+ &= A_\mathrm{d}\hat{x} + B_\mathrm{d}u + L(C_\mathrm{d}\hat{x} - y)\\ &= (A_\mathrm{d} + B_\mathrm{d}F + LC_\mathrm{d})\hat{x} + (M_{\bar{u}} - FM_{\bar{x}})B_\mathrm{d}r - Ly\end{aligned}$$

- **note:** observer has *two* inputs, first is reference $r[k]$, second is plant output $y[k]$

Design example: pendulum on cart

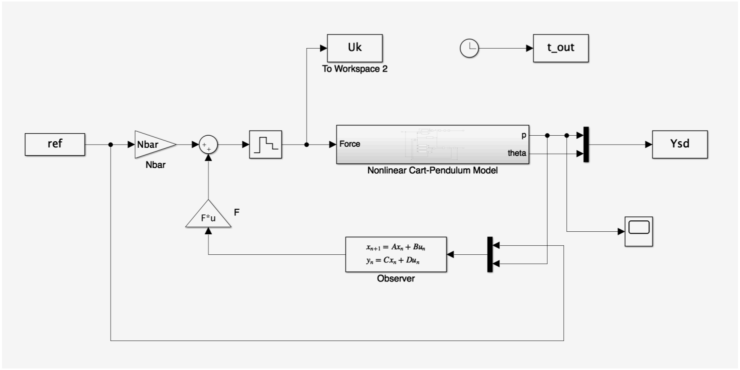
- compute scaling for reference input

```
Temp = [Ad-eye(n),Bd;Cd,0]\[zeros(n,1);1];
Mu = Temp(n+1:end); Mx = Temp(1:n);
Nbar = Mu-F*Mx;
```

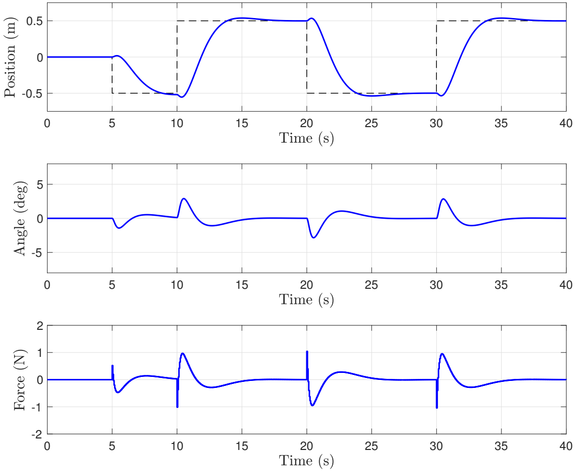
- define observer matrices for `Simulink`

```
Ao = Ad+Bd*F+L*Cd;
Bo = [Bd*Nbar,-L];
Co = eye(n); Do = zeros(n,2);
```

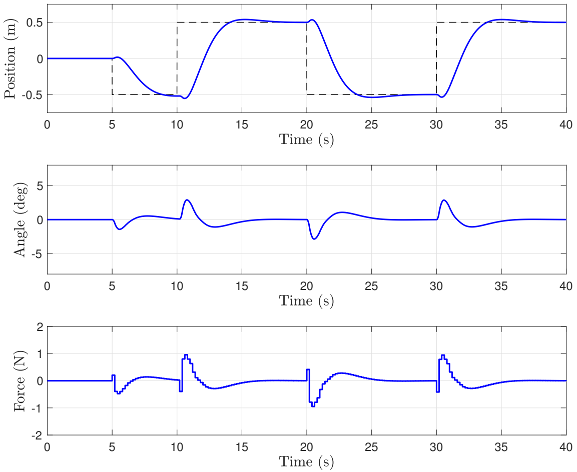
Design example: pendulum on cart



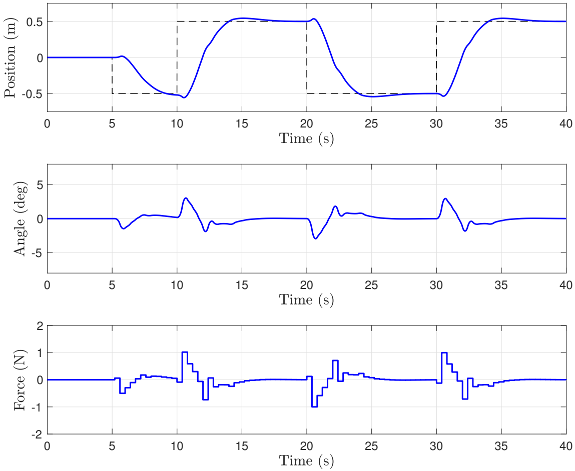
State-space control of pendulum on cart $T = 0.05$s



State-space control of pendulum on cart $T = 0.2$s



State-space control of pendulum on cart $T = 0.4$s



- can maintain performance with quite large sampling period!

MATLAB commands

- controllability and observability matrices

```
sys_dt = ss(Ad,Bd,Cd,Dd,T);
sys_dt = c2d(sys,T);
W_o = obsv(sys);
W_c = ctrb(sys);
```

- pole-placement using Ackerman

```
F = -acker(Ad, Bd, pole_vec);
L = -acker(Ad', Cd', pole_vec)';
```

- LQR controller

```
F = -lqr(sys_dt, Cd'*Cd, rho, zeros(n,1));
```

Additional references

- Nielsen, Chapter 9
- Åström & Wittenmark, Chapters 5 and 9
- Phillips, Nagle, & Chakrabortty, Chapter 9
- Franklin, Powell, & Workman, Chapter 8
- Hespanha, Topics in Undergraduate Control System Design, Chapter 8, 9, 11 (topics are discussed for continuous-time models)

Personal Notes

Personal Notes

Personal Notes

9. Introduction to system identification

- identification of functions
- identification of dynamic systems

Motivation for system identification

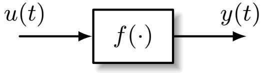
- nearly all of the control design we have done is *model-based*, in that our controller is designed based on a model of the plant
- in principle, feedback gives us robustness against model uncertainty; in practice, we still want/need to have a decent model
- sometimes it is impossible to model a system from first principles
  - may be unclear how to model some components
  - functional forms of nonlinearities may be unknown
- other times, an accurate first-principles model may be unnecessarily complex for control purposes

System identification

- **system identification** is the process of determining a *dynamic model* from measured input/output data
- identification can be
  - *unstructured*, where we assume a general class of models (*e.g.*, LTI)
  - *structured*, where we fix a model structure and estimate parameters
- system identification aims for simple (*e.g.*, LTI) models that are useful for actually doing control

Identification of functions

- suppose we have a static input/output relationship



- **example:** u is voltage applied to a load, y is the current drawn
- the precise function f is unknown (and probably *unknowable*)
- **idea:** apply inputs, measure outputs, and fit a model $\hat{f}$ for f

$$\hat{f}(u) = \theta_1\varphi_1(u) + \theta_2\varphi_2(u) + \cdots + \theta_n\varphi_n(u)$$

where $\{\theta_1, \ldots, \theta_n\}$ are unknown parameters and $\{\varphi_i(\cdot)\}$ are *basis functions* that we must choose

Identification of functions contd.

- **example:** if we have a nonlinear resistive load, we may choose

$$\hat{f}(u) = \theta_1 + \theta_2 u + \theta_3 u^2 + \theta_4 u^3$$

- we now run $N > 0$ *experiments*, where we apply an input u_i and record the output y_i, generating N pairs of data points

$$(u_i, y_i)\,, \qquad i \in \{1, \ldots, N\}.$$

- for each input u_i, we can estimate the output using our model $\hat{f}$

$$\hat{y}_i = \hat{f}(u_i) = \theta_1 \varphi_1(u_i) + \cdots + \theta_n \varphi_n(u_i)\,, \qquad i \in \{1, \ldots, N\}$$

Least squares

- stacking all these equations, we obtain

$$\underbrace{\begin{bmatrix}\hat{y}_1\\ \hat{y}_2\\ \vdots\\ \hat{y}_N\end{bmatrix}}_{:=\hat{y}} = \underbrace{\begin{bmatrix}\varphi_1(u_1) & \varphi_2(u_1) & \cdots & \varphi_n(u_1)\\ \varphi_2(u_2) & \varphi_2(u_2) & \cdots & \varphi_n(u_2)\\ \vdots & \vdots & \ddots & \vdots\\ \varphi_1(u_N) & \varphi_2(u_N) & \cdots & \varphi_n(u_N)\end{bmatrix}}_{:=\Phi}\underbrace{\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_n\end{bmatrix}}_{:=\theta}$$

- *least squares identification* minimizes measurement and prediction mismatch in mean-square sense

$$\underset{\theta\in\mathbb{R}^n}{\text{minimize}} \quad J_{\text{ls}}(\theta) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = (y-\hat{y})^{\mathsf{T}}(y-\hat{y})\,.$$

- any solution $\theta^* \in \mathbb{R}^n$ to this problem is a *least squares minimizer*

Least squares contd.

- least squares cost is

$$\begin{aligned} J_{\rm ls}(\theta) &= (y-\hat{y})^{\mathsf{T}}(y-\hat{y}) = y^{\mathsf{T}}y + \hat{y}^{\mathsf{T}}\hat{y} - 2y^{\mathsf{T}}\hat{y} \\ &= y^{\mathsf{T}}y + \theta^{\mathsf{T}}\Phi^{\mathsf{T}}\Phi\theta - 2y^{\mathsf{T}}\Phi\theta \end{aligned}$$

- take gradient of cost w.r.t. θ and set to zero

$$\nabla_\theta J_{\rm ls}(\theta) = 2\Phi^{\mathsf{T}}\Phi\theta - 2\Phi^{\mathsf{T}}y = \mathbb{0} \qquad \Longrightarrow \qquad \theta^* = (\Phi^{\mathsf{T}}\Phi)^{-1}\Phi^{\mathsf{T}}y$$

- inverse will exist as long as $N \geq n$ and $\{u_i\}$ are sufficiently diverse

- quality of fit expressed as

$$\frac{J_{\rm ls}(\theta^*)}{y^{\mathsf{T}}y} = 1 - \frac{y^{\mathsf{T}}\Phi\theta^*}{y^{\mathsf{T}}y} \qquad \text{(ideally } \ll 1\text{)}$$

Least squares and measurement noise

- measurements typically corrupted by noise $y_i \to y_i + v_i$
- assume noise is zero-mean $\mathbb{E}[v_i] = 0$
- instead of least-square solution θ^*, we have least squares *estimate*

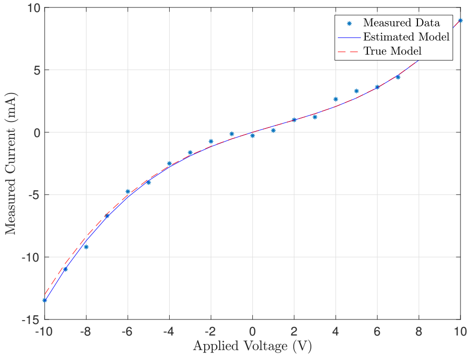
$$\hat{\theta} = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}(y+v)$$

- least squares is an *unbiased* estimator of θ^*

$$\begin{aligned}\mathbb{E}[\hat{\theta}] &= \mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}y] + \mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}v] \\ &= \mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}y] + (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}\underbrace{\mathbb{E}[v]}_{=\mathbb{0}} \\ &= \theta^*\end{aligned}$$

Example: nonlinear resistor

- data generated from $I = (0.5)V - (0.02)V^2 + (0.006)V^3 + v$, where v is Gaussian with standard deviation 0.4
- $\varphi_1(V) = V$, $\varphi_2(V) = V^2$, $\varphi_3(V) = V^3$



$$\hat{\theta} = \begin{bmatrix} 0.5066 \\ -0.0227 \\ 0.0062 \end{bmatrix}$$

Identification of dynamic systems

- we will examine structured identification of discrete-time LTI systems

$$G[z] = \frac{y[z]}{u[z]} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} = \frac{B[z]}{A[z]}$$

  of fixed order n, with corresponding difference equation

$$\begin{aligned} y[k] + a_1 y[k-1] + \cdots + a_n y[k-n] & \\ &= b_0 u[k] + \cdots + b_m u[k-m] \end{aligned}$$

- **Identification problem:** given experimental input/output data for u and y, find (the best) estimates for $\{a_i\}$ and $\{b_i\}$

Identification of dynamic systems contd.

- idea is to apply known input sequence $\{u[1], \ldots, u[N]\}$ and record output sequence $\{y[1], y[2], \ldots, y[N]\}$
- generate prediction from estimated model

$$\begin{aligned}\hat{y}[k] = -\hat{a}_1 \hat{y}[k-1] - \cdots - \hat{a}_n \hat{y}[k-n] \\ + \hat{b}_0 u[k] + \cdots + \hat{b}_m u[k-m]\end{aligned}$$

  where $\{\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_0, \ldots, \hat{b}_m\}$ are our parameters to be estimated

- since $y[k]$ depends on past values of y, the first prediction we can make is $y[n+1]$ at time $n+1$; we will therefore have $N-n$ values to compare between measurement and prediction

Identification of dynamic systems contd.

- lay out all equations for estimates

$$\underbrace{\begin{bmatrix} \hat{y}[n+1] \\ \hat{y}[n+2] \\ \vdots \\ \hat{y}[N] \end{bmatrix}}_{\hat{y}} = \underbrace{\begin{bmatrix} y[n] & \cdots & y[1] & u[n+1] & \cdots & u[n+1-m] \\ y[n+1] & \cdots & y[2] & u[n+2] & \cdots & u[n+2-m] \\ \vdots & & \vdots & \vdots & & \vdots \\ y[N-1] & \cdots & y[N-n] & u[N] & \cdots & u[N-m] \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} -\hat{a}_1 \\ \vdots \\ -\hat{a}_n \\ \hat{b}_0 \\ \vdots \\ \hat{b}_m \end{bmatrix}}_{\theta}$$

- like before, form least squares problem

$$\underset{\theta \in \mathbb{R}^{n+m+1}}{\text{minimize}} \quad J_{\text{ls}}(\theta) = \sum_{i=n+1}^{N} (y[i] - \hat{y}[i])^2 = (y - \hat{y})^{\mathsf{T}}(y - \hat{y})\,.$$

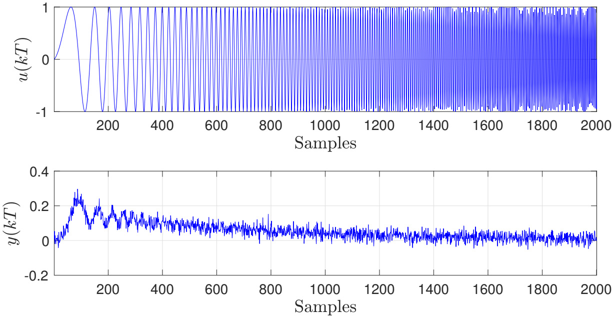
with solution

$$\theta^* = (\Phi^{\mathsf{T}}\Phi)^{-1}\Phi^{\mathsf{T}} y$$

Example: mass with friction

- we wish to identify mass and damping for model $(m = 2, b = 0.3)$

$$m\dot{v} = -bv + u \qquad \Longrightarrow \qquad \dot{v} = -\frac{b}{m}v + \frac{1}{m}u$$



Example: mass with friction

- applying the `c2d` transformation, we obtain discrete model

$$v[k+1] = \underbrace{e^{-bT/m}}_{-a_1} v[k] + \underbrace{\frac{1}{b}\left(1 - e^{-bT/m}\right)}_{b_1} u[k]$$

  or in standard difference equation form with $y[k] := v[k]$

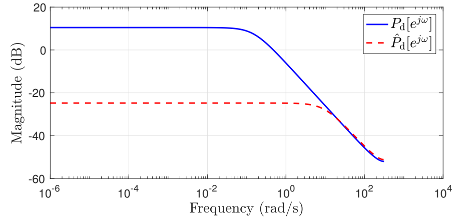
$$y[k] + a_1 y[k-1] = b_1 u[k-1]$$

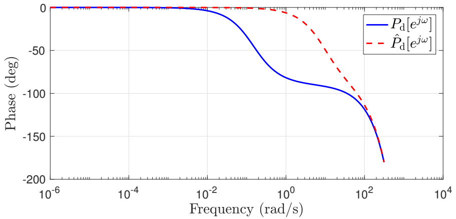
- therefore, we will fit a model $\hat{P}_{\mathrm{d}}$ of the form

$$\hat{y}[k] = -\hat{a}_1 \hat{y}[k-1] + \hat{b}_1 u[k-1]$$

Example: mass with friction

- $m = 2$, $b = 0.3$, $T = 0.01$, $N = 2000$ samples





Noise and dynamic system identification

- if least squares is unbiased, why are our results so poor?
- as before, the expected value of our estimate is

$$\mathbb{E}[\hat{\theta}] = \mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T} y] + \mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T} v]\,.$$

  but Φ depends on y, which depend on noise v, therefore

$$\mathbb{E}[(\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T} v] \neq (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}\mathbb{E}[v] = \mathbb{0}$$

- our estimate $\hat{\theta}$ is **biased**
- need a methodology for removing this bias

Noise and dynamic system identification contd.

- our underlying system is described by the model

$$\begin{aligned} x[k] + a_1 x[k-1] + \cdots + a_n x[k-n] \\ = b_0 u[k] + \cdots + b_m u[k-m] \end{aligned}$$

  from which we take noisy measurements $y[k] = x[k] + v[k]$

- take z-transforms with zero initial conditions:

$$\begin{aligned} (1 + a_1 z^{-1} + \cdots + a_n z^{-n}) X[z] &= (b_0 + \cdots + b_m z^{-m}) U[z] \\ Y[z] &= X[z] + V[z] \end{aligned}$$

  or, eliminating $X[z]$,

$$A[z]Y[z] \quad = \quad B[z]U[z] \qquad + \qquad \underbrace{A[z]V[z]}_{\text{noise filtered by system!}}$$

Noise and dynamic system identification contd.

- dividing by $A[z]$, we obtain

$$A[z]\underbrace{\left(\frac{Y[z]}{A[z]}\right)}_{:=\bar{Y}[z]} = B[z]\underbrace{\left(\frac{U[z]}{A[z]}\right)}_{:=\bar{U}[z]} + V[z]$$

- we have "unmixed" the noise and the system
- TF from $\bar{U}$ to $\bar{Y}$ is $G[z]$, *i.e.*, the same as TF from U to Y
- **therefore:** to estimate the transfer function $G[z]$, it makes no difference where we use the original measured data $(u[k], y[k])$, or the *filtered data* $(\bar{u}[k], \bar{y}[k])$!

Noise and dynamic system identification contd.

- since $\bar{Y}[z] = Y[z]/A[z]$, we have $A[z]\bar{Y}[z] = Y[z]$, and the signal $\bar{y}[k]$ may be constructed as

$$\bar{y}[k] = -a_1\bar{y}[k-1] - \cdots - a_n\bar{y}[k-n] + y[k]$$

- similarly, can construct signal $\bar{u}[k]$ via

$$\bar{u}[k] = -a_1\bar{u}[k-1] - \cdots - a_n\bar{u}[k-n] + u[k]$$

- **problem:** we don't know the coefficients $\{a_1, \ldots, a_n\}$
- **idea:** instead use *current estimates* $\{\hat{a}_1, \ldots, \hat{a}_n\}$

Iterative least-squares for dynamic system id

1. run least squares with unfiltered data $u[k]$ and $y[k]$, obtaining initial estimates $\{\hat{a}_1, \ldots, \hat{a}_n\}$ and $\{\hat{b}_1, \ldots, \hat{b}_m\}$

2. build the approximate system denominator

$$\hat{A}[z] = 1 + \hat{a}_1 z^{-1} + \cdots + \hat{a}_n z^{-n}$$

   and filter the input/output data $(u[k], y[k])$ to obtain $(\bar{u}[k], \bar{y}[k])$

3. using the filtered data $(\bar{u}[k], \bar{y}[k])$, recompute the least squares solution to find updated estimates $\{\hat{a}_1, \ldots, \hat{a}_n\}$ and $\{\hat{b}_1, \ldots, \hat{b}_m\}$

4. repeat steps (2)–(4) until coefficients stop changing

Example: mass with friction

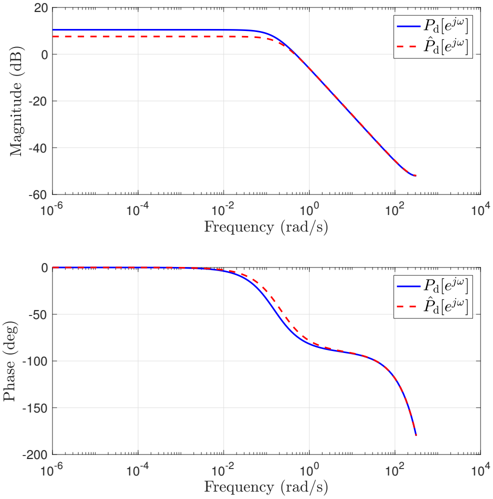
- filter data with filter $\hat{A}[z] = 1 + \hat{a}_1 z^{-1}$

  ```
  Ahat = [1,ahat_1];
  u_filt = filter(1,Ahat,u);
  y_filt = filter(1,AHat,y);
  ```

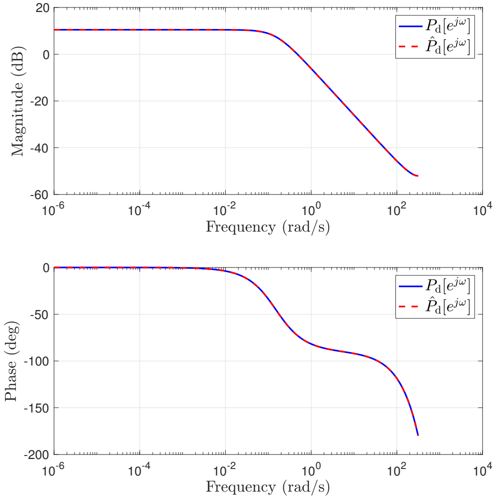
- resolve least-squares problem

  ```
  Phi = [y_filt(n:N-1),u_filt(n:N-1)];
  theta = Phi\y_filt(n+1:N);
  ```

Example: mass with friction (data filtered once)



Example: mass with friction (data filtered twice)



Final comments on system identification

- many more non-trivial issues such as
  - selecting appropriate test inputs
  - numerical conditioning of data for least squares
  - choice of sampling frequency in discretization of c.t. model
  - combining multiple experiments
  - identifying systems in closed-loop
  - recursive (online) system ID
  - other choices for objective function
  - ...

MATLAB commands

- solve non-square linear system

```
x = A\b;
```

- filter vector x using filter $F[z] = \frac{1+8z^{-2}}{1+z^{-1}+2z^{-2}}$

```
x_filt = filter([1,0,8],[1,1,2],x);
```

- extract coefficients from transfer function

```
[Bd,Ad]=tfdata(Pd);
Bd=Bd{1}; Ad=Ad{1};
```

Additional references

- Nielsen Chapter 3
- Åström & Wittenmark, Chapter 13
- Swevers, Introduction to System Identification (slides)
- Keesman (*System Identification*) Chapter 6
- Franklin, Powell, & Workman, Chapter 12
- Phillips, Nagle, & Chakrabortty, Chapter 10
- Hespanha, Topics in Undergraduate Control System Design, Chapters 2–4

Parting thoughts on ECE 484 . . .

- broadly speaking, control design is about managing dynamics and uncertainty in real-time using feedback (and feedforward)
- state-of-the-art control design looks like fancy versions of the LQR problem, but with
  - many sensors and actuators, distributed architectures
  - more emphasis on optimization theory
  - stochastic, robust, nonlinear, large-scale . . .
- multivariable optimal control in ECE 488 (planes, robots, *etc.*)
- *the really good stuff* is in graduate school

Personal Notes

Personal Notes

Personal Notes

10. Appendix: mathematics review

Important Laplace transforms

Name	$f(t)$	$F(s) = \mathscr{L}\{f(t)\}$
Delta	$\delta(t)$	1
Step	$\mathbb{1}(t)$	$1/s$
Ramp	t	$1/s^2$
Monomial	t^n	$n!/t^{n+1}$
Sine	$\sin(\omega_0 t)$	$\omega_0/(s^2+\omega_0^2)$
Cosine	$\cos(\omega_0 t)$	$s/(s^2+\omega_0^2)$
Exponential	$e^{-\alpha t}$	$1/(s+a)$
Exp/Sin	$e^{-\alpha t}\sin(\omega_0 t)$	$\omega_0/[(s+\alpha)^2+\omega_0^2]$
Exp/Cos	$e^{-\alpha t}\cos(\omega_0 t)$	$(s+\alpha)/[(s+\alpha)^2+\omega_0^2]$

(note: all signals assumed to be zero for $t < 0$)

Properties of the Laplace transform

Name	$f(t)$	$\mathscr{L}\{f(t)\}$
Superposition	$\alpha f_1(t) + \beta f_2(t)$	$\alpha F_1(s) + \beta F_2(s)$
Delay	$f(t-\tau)$	$e^{-\tau s}F(s)$
Derivative rule	$\dot{f}(t)$	$sF(s) - f(0)$
Integral rule	$\int_0^t f(\tau)\,\mathrm{d}\tau$	$\frac{1}{s}F(s)$
Convolution	$f_1(t) * f_2(t)$	$F_1(s)F_2(s)$
Initial value theorem	$f(0^+)$	$\lim_{s\to\infty} sF(s)$
Final value theorem	$\lim_{t\to\infty} f(t)$	$\lim_{s\to 0} sF(s)$

(note: all signals assumed to be zero for $t < 0$)

Important z-transforms

Name	$f[k]$	$F[z] = \mathcal{Z}\{f[k]\}$
Delta	$\delta[k]$	1
Step	$\mathbb{1}[k]$	$z/(z-1)$
Ramp	k	$z/(z-1)^2$
Exponential	a^k	$z/(z-a)$
Sine	$\sin(\omega_0 k)$	$z\sin(\omega_0)/(z^2 - 2z\cos(\omega_0) + 1)$
Cosine	$\cos(\omega_0 k)$	$z(z - \cos(\omega_0))/(z^2 - 2z\cos(\omega_0) + 1)$

(note: all signals assumed to be zero for $k < 0$)

Properties of the z-transform

Name	$f[k]$	$\mathcal{Z}\{f[k]\}$
Superposition	$\alpha f_1[k] + \beta f_2[k]$	$\alpha F_1[z] + \beta F_2[z]$
Delay-by-n	$f[k-n]$	$z^{-n}F[z]$
Advance-by-1	$f[k+1]$	$zF[z] - zf[0]$
Sum rule	$\sum_{\ell=0}^{k} f[\ell]$	$\frac{z}{z-1}F[z]$
Convolution	$f_1[k] * f_2[k]$	$F_1[z]F_2[z]$
Initial value theorem	$f[0^+]$	$\lim_{z\to\infty} F[z]$
Final value theorem	$\lim_{k\to\infty} f[k]$	$\lim_{z\to 1}(z-1)F[z]$

(note: all signals assumed to be zero for $k < 0$)

Linear algebra and matrices

- vector spaces and subspaces
- matrix definitions
- invertibility
- eigenvalues/eigenvectors
- diagonalization

Vector spaces and subspaces

- a space of objects which can be added and scaled by constants
- most important vector space for our purposes is $\mathbb{R}^n$ with elements

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad x_i \in \mathbb{R},$$

where addition and scalar multiplication are defined by

$$x + y = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}, \qquad \alpha x = \begin{bmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{bmatrix}, \quad \alpha \in \mathbb{R}.$$

Vector spaces and subspaces contd.

- a set $X = \{x_1, x_2, \dots, x_k\}$ of vectors is *linearly independent* if

$$\alpha_1 x_1 + \cdots + \alpha_n x_n = 0 \qquad \Longrightarrow \qquad \alpha_1 = \cdots = \alpha_n = 0\,.$$

- the *span* of X is the set of all possible linear combinations

$$\mathrm{span}(X) = \{\alpha_1 x_1 + \cdots + \alpha_n x_n \mid \alpha_1, \dots, \alpha_n \in \mathbb{R}\}$$

- a *subspace* S of $\mathbb{R}^n$ is a subset which is *also* a vector space
  - all subspaces of $\mathbb{R}^n$ are hyperplanes passing through the origin

- a *basis* for a subspace S is a set $X = \{x_1, x_2, \dots, x_k\}$ of linearly independent vectors such that $\mathrm{span}(X) = S$, and the *dimension* $\dim(S)$ of the subspace is the smallest number of vectors required in such a basis (in this case, k vectors)

Matrices

A square $n \times n$ matrix is a collection of n^2 real numbers

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

- $A \in \mathbb{R}^{n \times n}$ means that A is an $n \times n$ matrix with real entires
- A defines a *linear transformation* from $\mathbb{R}^n$ to $\mathbb{R}^n$ via matrix-vector multiplication

$$y = Ax \qquad \Longleftrightarrow \qquad y_i = \sum_{j=1}^{n} A_{ij} x_j, \quad i = 1, \ldots, n.$$

Range, nullspace, and the rank-nullity theorem

For a matrix $A \in \mathbb{R}^{n \times n}$

- *nullspace:* $\mathrm{null}(A) := \{x \in \mathbb{R}^n \mid Ax = \mathbb{0}\}$
  - the subspace of input vectors that get “zeroed out” by A
  - $\mathrm{nullity}(A) := \dim(\mathrm{null}(A))$

- *range:* $\mathrm{range}(A) := \{Ax \mid x \in \mathbb{R}^n\}$
  - the subspace of linear combinations of columns of A
  - $\mathrm{rank}(A) = \dim(\mathrm{range}(A))$

Rank-Nullity Theorem: $\mathrm{rank}(A) + \mathrm{nullity}(A) = n$

Invertibility of matrices

- square matrix $A \in \mathbb{R}^{n \times n}$ is *invertible* or *nonsingular* if there exists another matrix $B \in \mathbb{R}^{n \times n}$ such that $AB = BA = I_n$
- the inverse is always unique and we denote it by A^{-1}
- the following statements are all equivalent:
  - A is invertible
  - $\mathrm{rank}(A) = n$
  - $\mathrm{nullity}(A) = 0$
  - the columns of A form a basis for $\mathbb{R}^n$
  - $0 \notin \mathrm{eig}(A)$
  - $\det(A) \neq 0$
- if A, B, C are all invertible, then $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$

Solvability of linear systems of equations

For $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, consider the equation

$$Ax = b$$

in the unknown vector $x \in \mathbb{R}^n$.

- the equation is solvable if and only if $b \in \mathrm{range}(A)$
- in this case, all solutions can be written as $x = p + v$, where p is a *particular* solution satisfying $Ap = b$ and $v \in \mathrm{null}(A)$ is any *homogeneous* solution, i.e., a solution to $Ax = 0$.
- for every $b \in \mathbb{R}^n$ there exists a unique x solving $Ax = b$ if and only if A is invertible.

Constructing inverses

For $A \in \mathbb{R}^{2\times 2}$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad\qquad A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

assuming $\det(A) = ad - bc \neq 0$.

- For $A \in \mathbb{R}^{n\times n}$ there is a formula similar to the above

$$A^{-1} = \frac{1}{\det(A)}\mathrm{adj}(A)\,.$$

- $\mathrm{adj}(A)$ is the *adjugate matrix*
  - $[\mathrm{adj}(A)]_{ij}$ is formed from (j,i)th *minor* of A
  - important for relating state-space and transfer function models

Eigenvalues and eigenvectors

- **Eigenvalue equation:** $Ax = \lambda x$
  - $\lambda \in \mathbb{C}$ is the eigenvalue
  - $x \in \mathbb{C}^n$ is the (right) eigenvector
- eigenvalues obtained from *characteristic polynomial* $\Pi_A(s)$

$$\Pi_A(s) := \det(sI_n - A)$$

$$\text{set of eigenvalues} = \operatorname{eig}(A) := \{\lambda \in \mathbb{C} \mid \Pi_A(\lambda) = 0\}$$

- given eigenvalue λ_i, eigenvector x_i obtained by solving

$$(A - \lambda_i I_n)x_i = \mathbb{0}$$

- eigenvalues are unique, eigenvectors are not

Diagonalization

- $A \in \mathbb{R}^{n \times n}$, with eigenvalues λ_i and eigenvectors $v_i \in \mathbb{R}^n$

$$Av_i = \lambda_i v_i$$

form matrix of eigenvectors: $$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{n \times n}$$

- if $\{v_1, v_2, \ldots, v_n\}$ are linearly independent then V is invertible

$$V^{-1}AV = \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

- similar ideas (Jordan form) when diagonalization is not possible

Example

$$A = \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix} \qquad \lambda I_2 - A = \begin{bmatrix} \lambda + 1 & -1 \\ -2 & \lambda + 2 \end{bmatrix}$$

$$\Pi_A(\lambda) = \det(\lambda I_2 - A) = \lambda^2 + 3\lambda + 2 - 2 = \lambda(\lambda + 3)$$

$$\mathrm{eig}(A) = \{\lambda_1, \lambda_2\} = \{0, -3\} \qquad \text{with} \qquad v_{\lambda_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad v_{\lambda_2} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\Lambda = V^{-1} A V = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -3 \end{bmatrix}$$

Personal Notes

Personal Notes

Personal Notes