http://elec3004.com

Feedback & Regulation

ELEC 3004: **Systems**: Signals & Controls

Dr. Surya Singh

Lecture 16

(with material from Lathi, FPW and Albertos+Mareels)

elec3004@itee.uq.edu.au

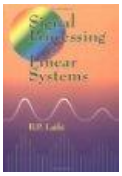
http://robotics.itee.uq.edu.au/~elec3004/

May 5, 2016

Lecture Schedule:

Week	Date	Lecture Title
1	29-Feb	Introduction
	3-Mar	Systems Overview
2	7-Mar	Systems as Maps & Signals as Vectors
	10-Mar	Data Acquisition & Sampling
3	14-Mar	Sampling Theory
	17-Mar	Antialiasing Filters
4	21-Mar	Discrete System Analysis
	24-Mar	Convolution Review
	28-Mar	Holiday
	31-Mar	
5	4-Apr	Frequency Response & Filter Analysis
	7-Apr	Filters
6	11-Apr	Digital Filters
	14-Apr	Digital Filters
7	18-Apr	Digital Windows
	21-Apr	FFT
8	25-Apr	Holiday
	28-Apr	Introduction to Feedback Control
9	3-May	Holiday
	5-May	**Feedback Control & Regulation**
10	9-May	Servoregulation/PID
	12-May	Introduction to (Digital) Control
11	16-May	Digital Control Design
	19-May	Stability
12	23-May	Digital Control Systems: Shaping the Dynamic Response & Estimation
	26-May	Applications in Industry
13	30-May	System Identification & Information Theory
	2-Jun	Summary and Course Review

Follow Along Reading:



B. P. Lathi
Signal processing and linear systems
1998
[TK5102.9.L38 1998]

G. Franklin,
J. Powell,
M. Workman
Digital Control of Dynamic Systems
1990

[TJ216.F72 1990]
[Available as UQ Ebook]

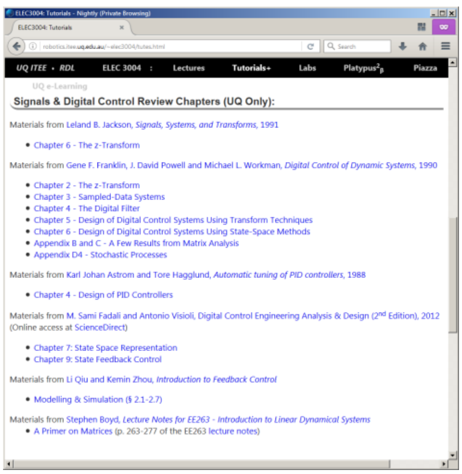
Today

- Lathi
  Continuous-Time System Analysis Using the Laplace Transform
- FPW
  Chapter 2: Linear, Discrete, Dynamic-Systems Analysis

→ **P - I - D**
- FPW
  - Chapter 4: Discrete Equivalents to Continuous
  - Transfer Functions: The Digital Filter
  - Chapter 5: Design of Digital Control Systems Using Transform Techniques

Next Time

Announcements: Controls Resources


ELEC3004: Tutorials - Nightly (Private Browsing)
ELEC3004: Tutorials
robotics.itee.uq.edu.au/~elec3004/tutes.html
UQ ITEE • RDL ELEC 3004 : Lectures Tutorials+ Labs Platypus$^2_\beta$ Piazza
UQ e-Learning

Signals & Digital Control Review Chapters (UQ Only):

Materials from Leland B. Jackson, *Signals, Systems, and Transforms*, 1991

- Chapter 6 - The z-Transform

Materials from Gene F. Franklin, J. David Powell and Michael L. Workman, *Digital Control of Dynamic Systems*, 1990

- Chapter 2 - The z-Transform
- Chapter 3 - Sampled-Data Systems
- Chapter 4 - The Digital Filter
- Chapter 5 - Design of Digital Control Systems Using Transform Techniques
- Chapter 6 - Design of Digital Control Systems Using State-Space Methods
- Appendix B and C - A Few Results from Matrix Analysis
- Appendix D4 - Stochastic Processes

Materials from Karl Johan Astrom and Tore Hagglund, *Automatic tuning of PID controllers*, 1988

- Chapter 4 - Design of PID Controllers

Materials from M. Sami Fadali and Antonio Visioli, Digital Control Engineering Analysis & Design (2nd Edition), 2012 (Online access at ScienceDirect)

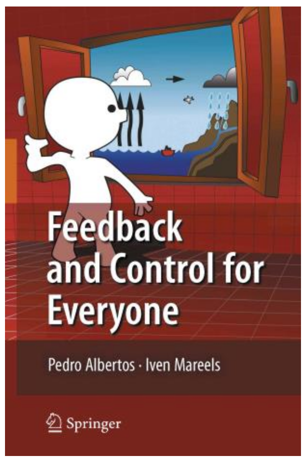
- Chapter 7: State Space Representation
- Chapter 9: State Feedback Control

Materials from Li Qiu and Kemin Zhou, *Introduction to Feedback Control*

- Modelling & Simulation (§ 2.1-2.7)

Materials from Stephen Boyd, *Lecture Notes for EE263 - Introduction to Linear Dynamical Systems*

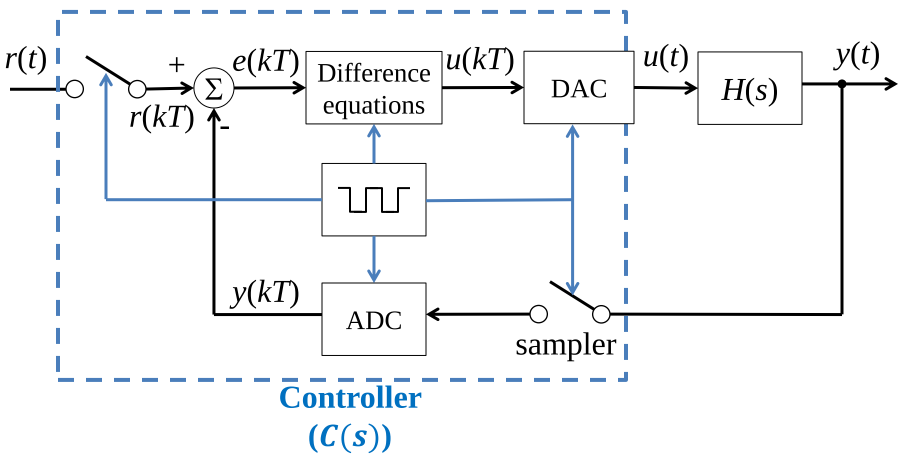
- A Primer on Matrices (p. 263-277 of the EE263 lecture notes)



- Free e-Book [via UQ]:
  https://library.uq.edu.au/record=b2508512~S7

Modelling

(Digital) **Feedback Control**



- Continuous signals sampled with period T
- kth control value computed at $t_k = kT$

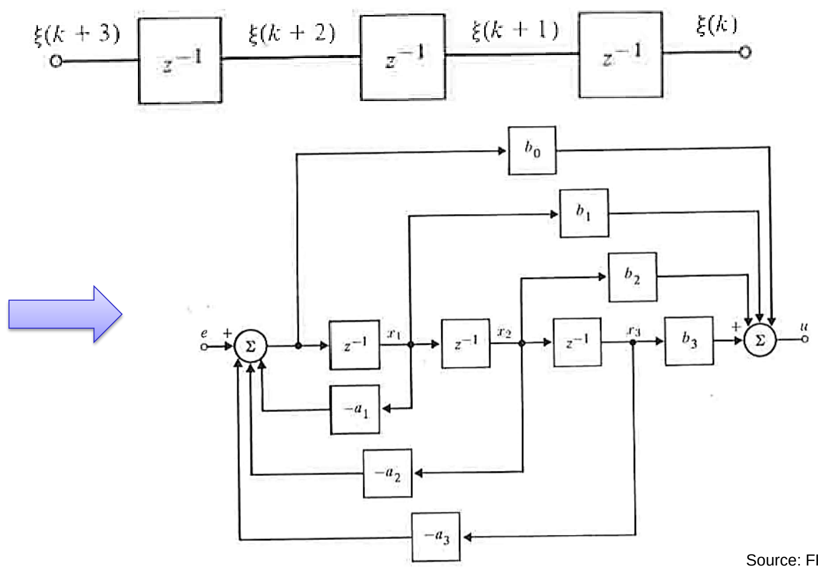
C(s): PID = Control for the PID-dly minded

- Proportional-Integral-Derivative control is the control engineer's hammer*
  - For P,PI,PD, etc. just remove one or more terms

$$C(s) = k\left(\underbrace{1}_{\text{Proportional}} + \underbrace{\frac{1}{\tau i s}}_{\text{Integral}} + \underbrace{\tau d s}_{\text{Derivative}}\right)$$

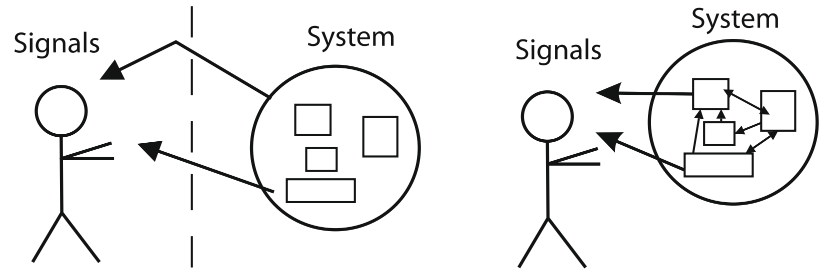
*Everything is a nail. That's why it's called "Bang-Bang" Control ☺

Feedback Control: Tuning Nightmare!



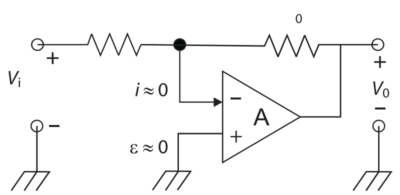
Source: FPW, Fig. 2.8

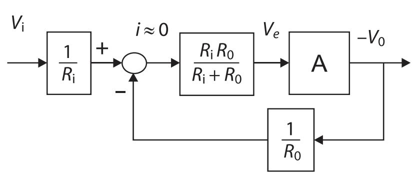
Signals and Systems: Modelling Tools!



- Signals: Often came from a system
- Now we want feedback tools so as to understand the structure of the systems and how they interact so as to get desired signals

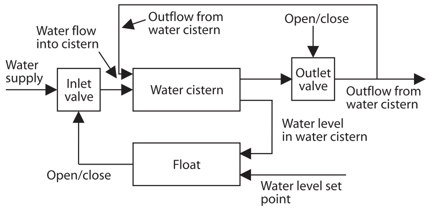
Feedback Control + Models: Everywhere

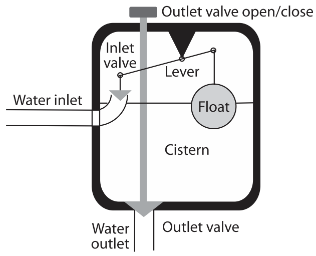


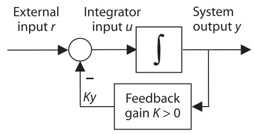


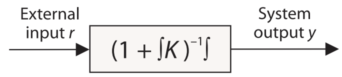
- $$\frac{V_0}{V_1} = -\frac{R_0}{R_1}\frac{1}{1+\frac{1}{A}\left(1+\frac{R_0}{R_1}\right)}$$

Feedback Control + Models: Everywhere







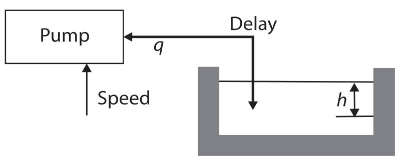


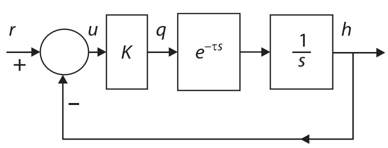
- Integrator:

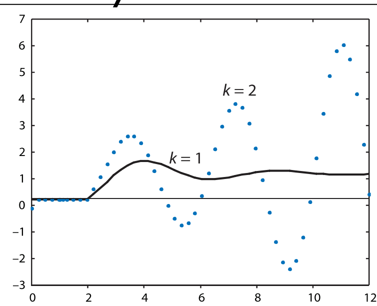
$$y(t) = y_0 + \int_0^t u(\tau)d\tau$$

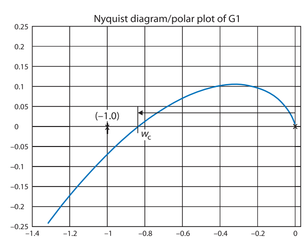
$$y(t) = y_0 + \frac{A\sin(\omega t)}{\omega}$$

Feedback Control + Models: Everywhere



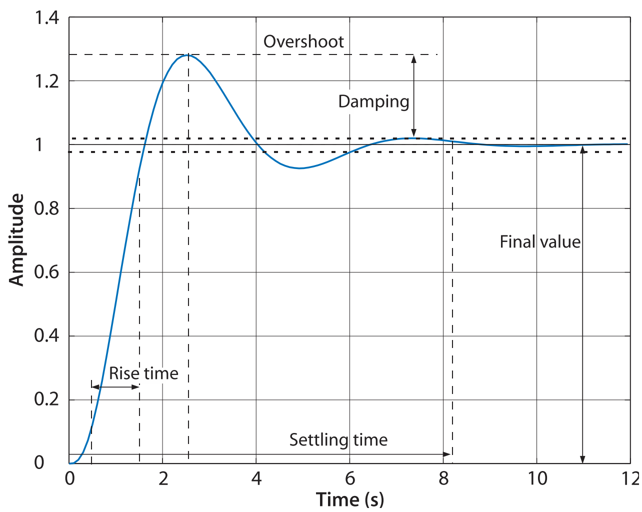




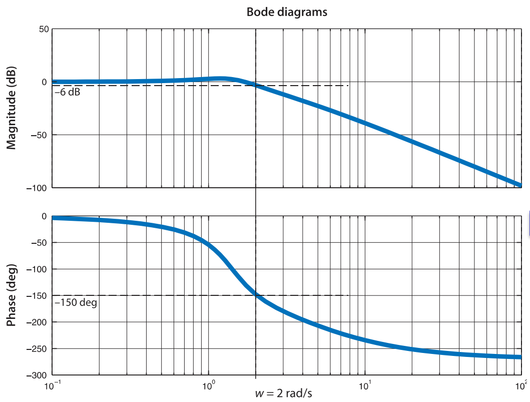


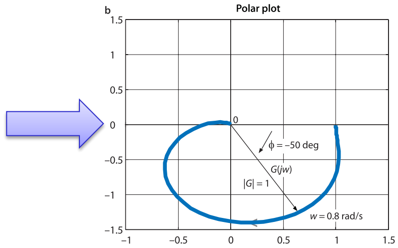
Feedback as a Filter

Time Response



Frequency Domain Analysis



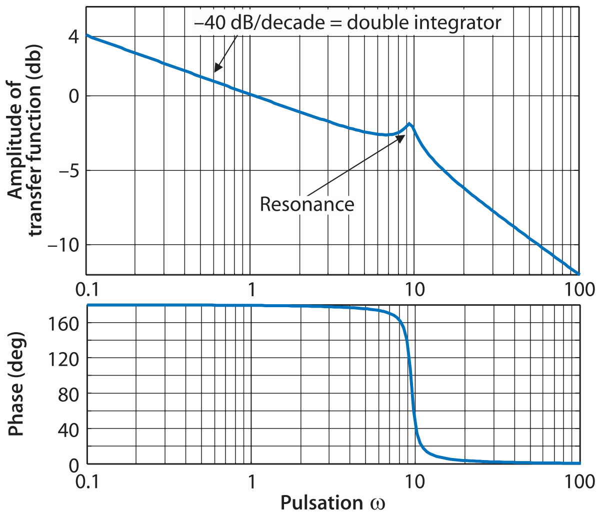


- Bode
(Magnitude + Phase Plots)

- Nyquist Plot
(Polar)

In This Way Feedback May Be Seen as a Filter

- Ex: Lightly Damped Robot Arm



Ex: 2nd Order Responses!

Review: Direct Design: Second Order Digital Systems

Consider the z-transform of a decaying exponential signal:

$$y(t) = e^{-at}\cos(bt)\,\mathcal{U}(t) \qquad (\mathcal{U}(t) = \text{unit step})$$

- sample: $y(kT) = r^k \cos(k\theta)\,\mathcal{U}(kT)$ with $r = e^{-aT}$ & $\theta = bT$
- transform: $Y(z) = \frac{1}{2}\frac{z}{(z - re^{j\theta})} + \frac{1}{2}\frac{z}{(z - re^{-j\theta})}$

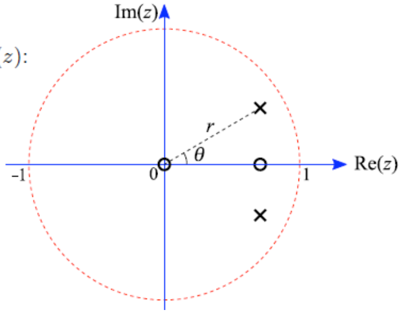
$$= \frac{z(z - r\cos\theta)}{(z - re^{j\theta})(z - re^{-j\theta})}$$

- e.g. y_k is the pulse response of $G(z)$:

$$G(z) = \frac{z(z - r\cos\theta)}{(z - re^{j\theta})(z - re^{-j\theta})}$$

poles: $\begin{cases} z = re^{j\theta} \\ z = re^{-j\theta} \end{cases}$

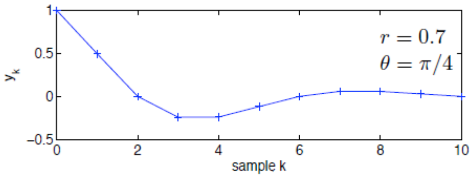
zeros: $\begin{cases} z = 0 \\ z = r\cos\theta \end{cases}$



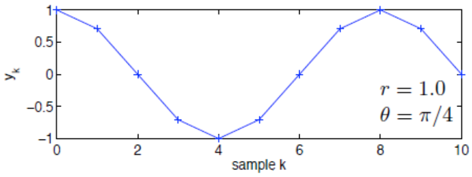
Response of 2nd order system [1/3]

Responses for varying r:

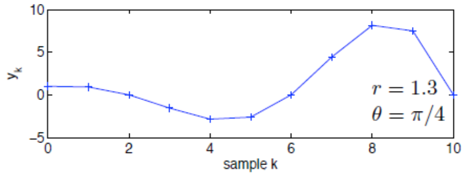
- $r < 1$
  $\Downarrow$
  exponentially decaying envelope



- $r = 1$
  $\Downarrow$
  sinusoidal response with $2\pi/\theta$ samples per period



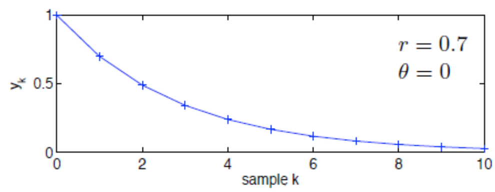
- $r > 1$
  $\Downarrow$
  exponentially increasing envelope



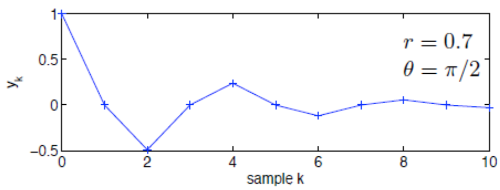
Response of 2nd order system [2/3]

Responses for varying θ:

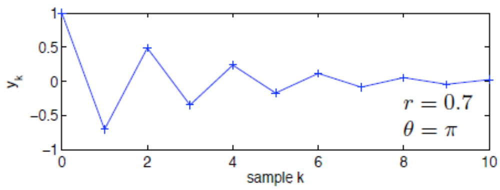
- $\theta = 0$
  $\Downarrow$
  decaying exponential



- $\theta = \pi/2$
  $\Downarrow$
  $2\pi/\theta = 4$ samples per period



- $\theta = \pi$
  $\Downarrow$
  2 samples per period



Response of 2nd order system [3/3]

Some special cases:

- for $\theta = 0$, $Y(z)$ simplifies to:

$$Y(z) = \frac{z}{z - r}$$

$\Longrightarrow$ exponentially decaying response

- when $\theta = 0$ and $r = 1$:

$$Y(z) = \frac{z}{z - 1}$$

$\Longrightarrow$ unit step

- when $r = 0$:

$$Y(z) = 1$$

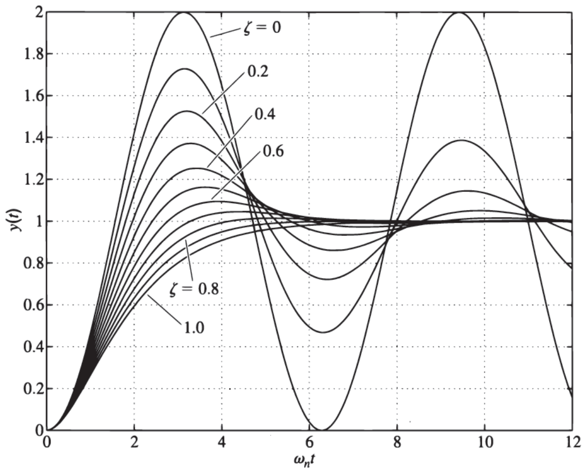
$\Longrightarrow$ unit pulse

- when $\theta = 0$ and $-1 < r < 0$:

samples of alternating signs

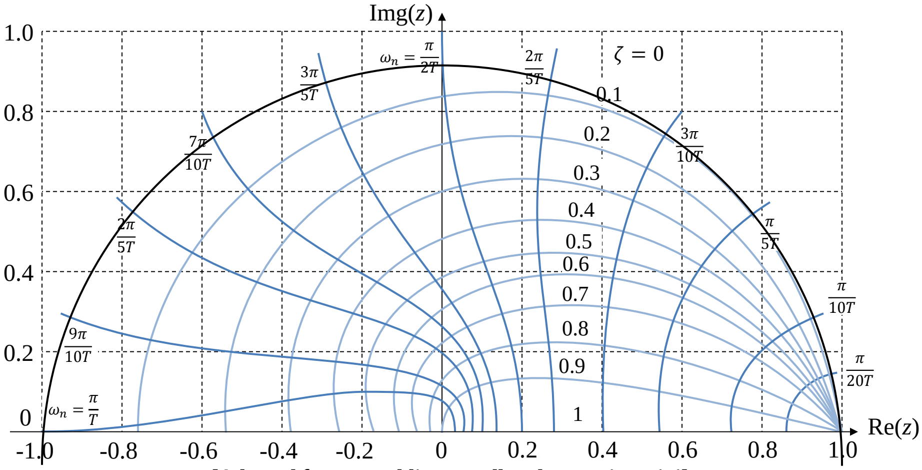
2nd Order System Response

- Response of a 2nd order system to increasing levels of damping:



Damping and natural frequency

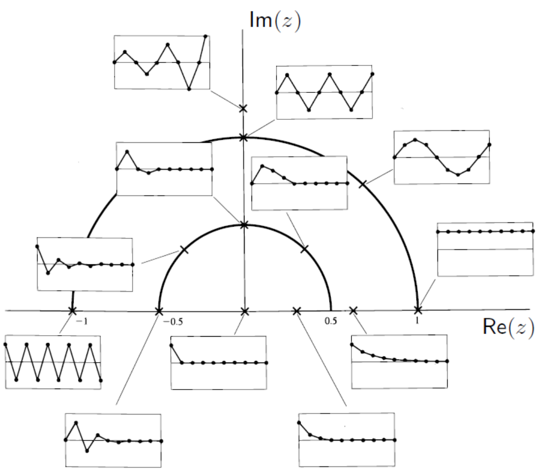
$$z = e^{sT} \text{ where } s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$



[Adapted from Franklin, Powell and Emami-Naeini]

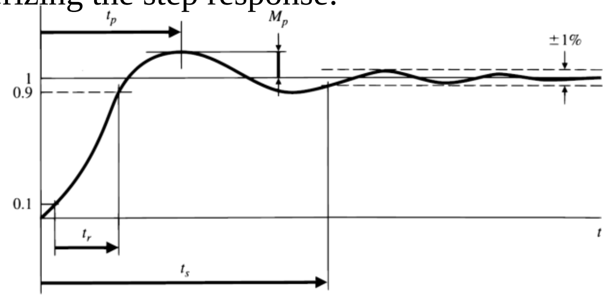
Pole positions in the z-plane

- Poles inside the unit circle are **stable**
- Poles outside the unit circle **unstable**
- Poles on the unit circle are oscillatory
- Real poles at 0 < z < 1 give exponential response
- Higher frequency of oscillation for larger
- Lower apparent damping for larer and r



2^{nd} Order System Specifications

Characterizing the step response:



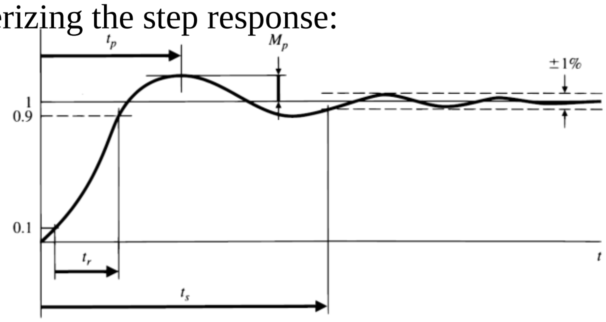
- Rise time (10% → 90%): $t_r \approx \frac{1.8}{\omega_0}$
- Overshoot: $M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1-\zeta^2}}$
- Settling time (**<u>to 1%</u>**): $t_s = \frac{4.6}{\zeta\omega_0}$
- Steady state error to unit step: e_{ss}
- Phase margin:

$$\phi_{PM} \approx 100\zeta$$

Why 4.6? It's **-ln(1%)**
$\rightarrow e^{-\zeta\omega_0} = 0.01 \rightarrow \zeta\omega_0 = 4.6 \rightarrow t_s = \frac{4.6}{\zeta\omega_0}$

2^{nd} Order System Specifications

Characterizing the step response:



- Rise time (10% → 90%) & Overshoot:
  t_r, M_p → ζ, ω_0 : Locations of dominant poles
- Settling time (to 1%):
  t_s → radius of poles: $|z| < 0.01^{\frac{T}{t_s}}$
- Steady state error to unit step:
  e_{ss} → final value theorem $e_{ss} = \lim_{z\to 1}\{(z-1)F(z)\}$

Ex: System Specifications → Control Design [1/4]

Design a controller for a system with:

- A continuous transfer function: $G(s) = \dfrac{0.1}{s(s+0.1)}$
- A discrete ZOH sampler
- Sampling time (T_s): $T_s = 1$s
- Controller:

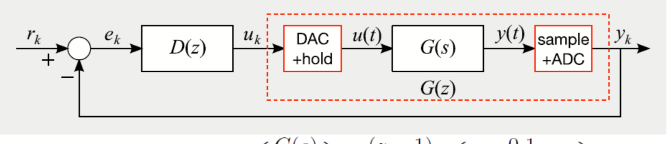
$$u_k = -0.5u_{k-1} + 13(e_k - 0.88e_{k-1})$$

The closed loop system is required to have:

- $M_p < 16\%$
- $t_s < 10$ s
- $e_{ss} < 1$

Ex: System Specifications → Control Design [2/4]

1. (a) Find the pulse transfer function of $G(s)$ plus the ZOH



$$G(z) = (1 - z^{-1})\mathcal{Z}\left\{\frac{G(s)}{s}\right\} = \frac{(z-1)}{z}\mathcal{Z}\left\{\frac{0.1}{s^2(s+0.1)}\right\}$$

e.g. look up $\mathcal{Z}\{a/s^2(s+a)\}$ in tables:

$$G(z) = \frac{(z-1)}{z}\,\frac{z\left((0.1-1+e^{-0.1})z + (1-e^{-0.1}-0.1e^{-0.1})\right)}{0.1(z-1)^2(z-e^{-0.1})}$$
$$= \frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}$$

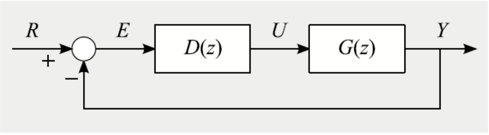
(b) Find the controller transfer function (using z = shift operator):

$$\frac{U(z)}{E(z)} = D(z) = 13\frac{(1-0.88z^{-1})}{(1+0.5z^{-1})} = 13\frac{(z-0.88)}{(z+0.5)}$$

Ex: System Specifications → Control Design [3/4]

2. Check the steady state error e_{ss} when $r_k =$ unit ramp

$$e_{ss} = \lim_{k\to\infty} e_k = \lim_{z\to 1}(z-1)E(z)$$



$$\frac{E(z)}{R(z)} = \frac{1}{1+D(z)G(z)}$$

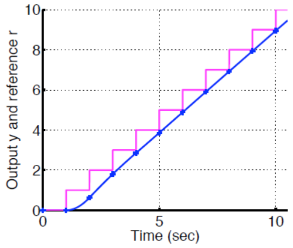
$$R(z) = \frac{Tz}{(z-1)^2}$$

so $$e_{ss} = \lim_{z\to 1}\left\{(z-1)\frac{Tz}{(z-1)^2}\frac{1}{1+D(z)G(z)}\right\} = \lim_{z\to 1}\frac{T}{(z-1)D(z)G(z)}$$

$$= \lim_{z\to 1}\frac{T}{(z-1)\dfrac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}D(1)}$$

$$= \frac{1-0.9048}{0.0484(1+0.9672)D(1)} = 0.96$$

$\implies e_{ss} < 1$ (as required)



Ex: System Specifications → Control Design [4/4]

3. Step response: overshoot $M_p < 16\% \implies \zeta > 0.5$

settling time $t_s < 10 \implies |z| < 0.01^{1/10} = 0.63$

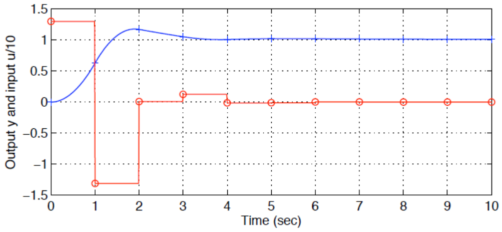
The closed loop poles are the roots of $1 + D(z)G(z) = 0$, i.e.

$$1 + 13\frac{(z-0.88)}{(z+0.5)}\frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)} = 0$$

$$\implies z = 0.88,\ -0.050 \pm j0.304$$

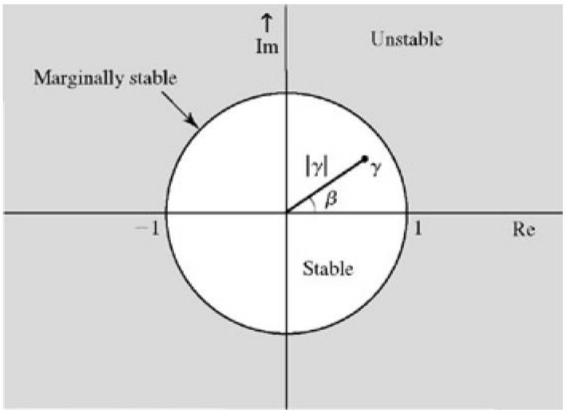
But the pole at $z = 0.88$ is cancelled by controller zero at $z = 0.88$, and

$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \implies \begin{cases} r = 0.31,\ \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$

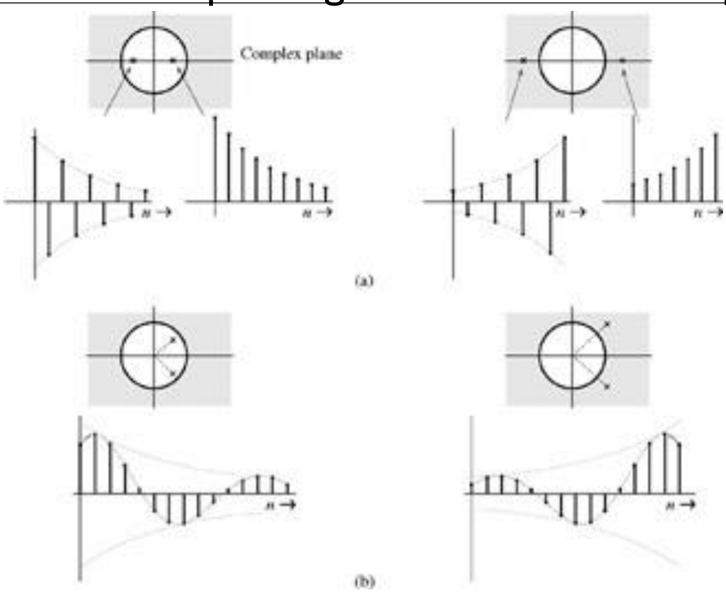


all specs satisfied!

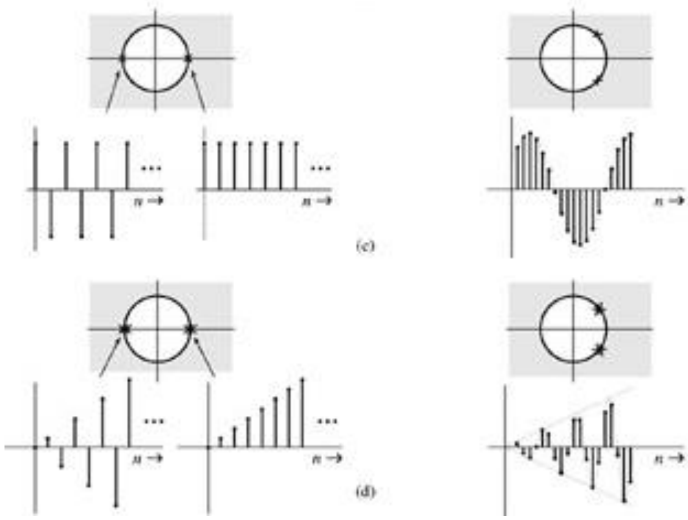
LTID Stability



Characteristic roots location and the corresponding characteristic modes [1/2]



Characteristic roots location and the corresponding characteristic modes [2/2]



How to Design? Back to Analog !

Two cases for control design

The system…
- Isn't fast enough
- Isn't damped enough
- Overshoots too much
- Requires too much control action

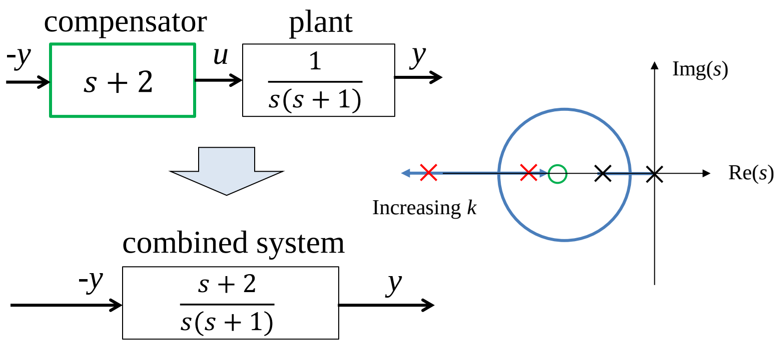
("Performance")

- Attempts to spontaneously disassemble itself

("Stability")

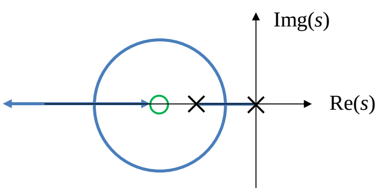
Dynamic compensation

- We can do more than just apply gain!
  - We can add dynamics into the controller that alter the open-loop response



But what dynamics to add?

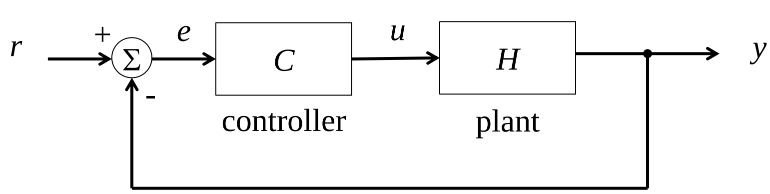
- Recognise the following:
  - A root locus starts at poles, terminates at zeros
  - "Holes eat poles"
  - Closely matched pole and zero dynamics cancel
  - The locus is on the real axis to the left of an odd number of poles (treat zeros as 'negative' poles)



The Root Locus (Quickly)

- The transfer function for a closed-loop system can be easily calculated:

$$y = CH(r - y)$$
$$y + CHy = CHr$$
$$\therefore \frac{y}{r} = \frac{CH}{1 + CH}$$



The Root Locus (Quickly)

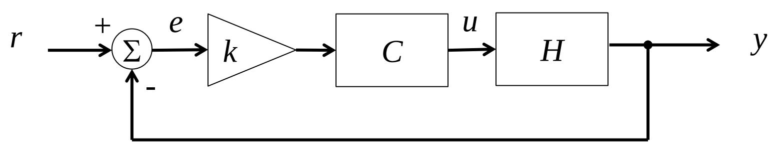
- We often care about the effect of increasing gain of a control compensator design:

$$\frac{y}{r} = \frac{kCH}{1 + kCH}$$

Multiplying by denominator:

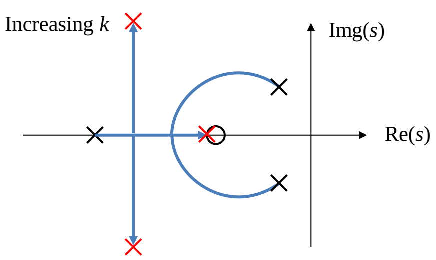
characteristic polynomial

$$\frac{y}{r} = \frac{kC_nH_n}{C_dH_d + kCnHn}$$



The Root Locus (Quickly)

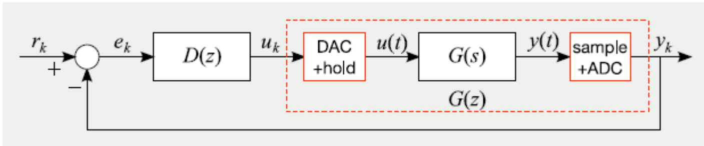
- Pole positions change with increasing gain
  - The trajectory of poles on the pole-zero plot with changing *k* is called the "root locus"
  - This is sometimes quite complex



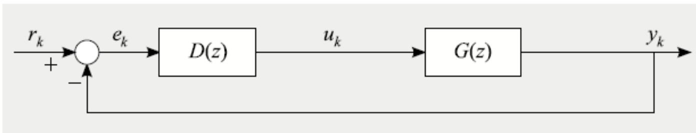
(In practice you'd plot these with computers)

Designing in the Purely Discrete...

Analyse/design a discrete controller $D(z)$:



by considering the purely discrete time system:



Closed loop system tranfer function: $\dfrac{Y(z)}{R(z)} = \dfrac{G(z)D(z)}{1 + G(z)D(z)}$

How do the closed loop poles relate to $\rightarrow$ stability?

$\rightarrow$ performance?

Now in discrete

- Naturally, there are discrete analogs for each of these controller types:

Lead/lag: $\dfrac{1-\alpha z^{-1}}{1-\beta z^{-1}}$

PID: $k\left(1 + \frac{1}{\tau_i(1-z^{-1})} + \tau_d(1 - z^{-1})\right)$

But, where do we get the control design parameters from?
The s-domain?

Sampling a continuous-time system

suppose $\dot{x} = Ax$

sample x at times $t_1 \leq t_2 \leq \cdots$: define $z(k) = x(t_k)$

then $z(k+1) = e^{(t_{k+1}-t_k)A} z(k)$

for uniform sampling $t_{k+1} - t_k = h$, so

$$z(k+1) = e^{hA} z(k),$$

a discrete-time LDS (called *discretized version* of continuous-time system)

Source: Boyd, Lecture Notes for EE263, 10-22

Piecewise constant system

consider *time-varying* LDS $\dot{x} = A(t)x$, with

$$A(t) = \begin{cases} A_0 & 0 \leq t < t_1 \\ A_1 & t_1 \leq t < t_2 \\ \vdots & \end{cases}$$

where $0 < t_1 < t_2 < \cdots$ (sometimes called jump linear system)

for $t \in [t_i, t_{i+1}]$ we have

$$x(t) = e^{(t-t_i)A_i} \cdots e^{(t_3-t_2)A_2} e^{(t_2-t_1)A_1} e^{t_1 A_0} x(0)$$

(matrix on righthand side is called state transition matrix for system, and denoted $\Phi(t)$)

Source: Boyd, Lecture Notes for EE263, 10-23

Qualitative behaviour of *x(t)*

suppose $\dot{x} = Ax$, $x(t) \in \mathbf{R}^n$

then $x(t) = e^{tA}x(0)$; $X(s) = (sI - A)^{-1}x(0)$

ith component $X_i(s)$ has form

$$X_i(s) = \frac{a_i(s)}{\mathcal{X}(s)}$$

where a_i is a polynomial of degree $< n$

thus the poles of X_i are all eigenvalues of A (but not necessarily the other way around)

Source: Boyd, Lecture Notes for EE263, 10-24

Qualitative behaviour of *x(t)* [2]

first assume eigenvalues λ_i are distinct, so $X_i(s)$ cannot have repeated poles

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{n} \beta_{ij} e^{\lambda_j t}$$

where β_{ij} depend on $x(0)$ (linearly)

eigenvalues determine (possible) qualitative behavior of x:

- eigenvalues give exponents that can occur in exponentials
- real eigenvalue λ corresponds to an exponentially decaying or growing term $e^{\lambda t}$ in solution
- complex eigenvalue $\lambda = \sigma + j\omega$ corresponds to decaying or growing sinusoidal term $e^{\sigma t}\cos(\omega t + \phi)$ in solution

Source: Boyd, Lecture Notes for EE263, 10-25

Qualitative behaviour of *x(t)* [3]

first assume eigenvalues λ_i are distinct, so $X_i(s)$ cannot have repeated poles

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{n} \beta_{ij} e^{\lambda_j t}$$

where β_{ij} depend on $x(0)$ (linearly)

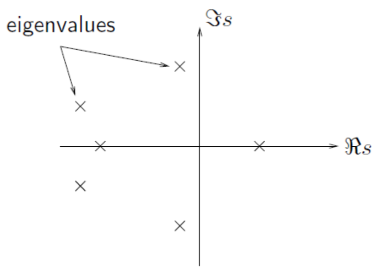
eigenvalues determine (possible) qualitative behavior of x:

- eigenvalues give exponents that can occur in exponentials
- real eigenvalue λ corresponds to an exponentially decaying or growing term $e^{\lambda t}$ in solution
- complex eigenvalue $\lambda = \sigma + j\omega$ corresponds to decaying or growing sinusoidal term $e^{\sigma t}\cos(\omega t + \phi)$ in solution

Source: Boyd, Lecture Notes for EE263, 10-26

Qualitative behaviour of *x(t)* [4]

- $\Re\lambda_j$ gives exponential growth rate (if > 0), or exponential decay rate (if < 0) of term
- $\Im\lambda_j$ gives frequency of oscillatory term (if $\neq 0$)



Source: Boyd, Lecture Notes for EE263, 10-27

Qualitative behaviour of *x(t)* [5]

now suppose A has repeated eigenvalues, so X_i can have repeated poles

express eigenvalues as $\lambda_1, \ldots, \lambda_r$ (distinct) with multiplicities $n_1, \ldots, n_r$, respectively $(n_1 + \cdots + n_r = n)$

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{r} p_{ij}(t) e^{\lambda_j t}$$

where $p_{ij}(t)$ is a polynomial of degree $< n_j$ (that depends linearly on $x(0)$)

Source: Boyd, Lecture Notes for EE263, 10-28

Emulation vs Discrete Design

- Remember: polynomial algebra is the same, whatever symbol you are manipulating:

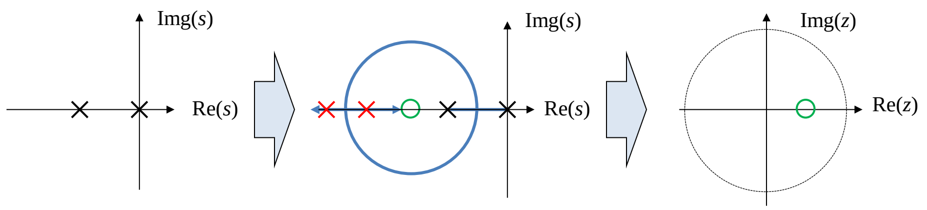
  eg. $s^2 + 2s + 1 = (s + 1)^2$

  $z^2 + 2z + 1 = (z + 1)^2$

  Root loci behave the same on both planes!
- Therefore, we have two choices:
  - Design in the s-domain and digitise (emulation)
  - Design only in the z-domain (discrete design)

Emulation design process

1. Derive the dynamic system model ODE
2. Convert it to a continuous transfer function
3. Design a continuous controller
4. Convert the controller to the z-domain
5. Implement difference equations in software



Emulation design process

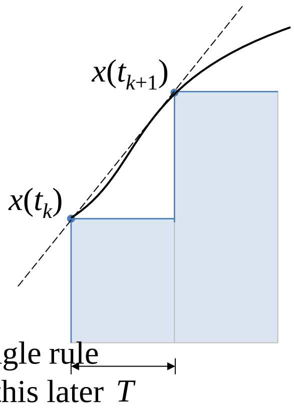
- Handy rules of thumb:
  - Use a sampling period of 20 to 30 times faster than the closed-loop system bandwidth
  - Remember that the sampling ZOH induces an effective $T/2$ delay
  - There are several approximation techniques:
    - Euler's method
    - Tustin's method
    - Matched pole-zero
    - Modified matched pole-zero

Euler's method*

- Dynamic systems can be approximated† by recognising that:

$$\dot{x} \cong \frac{x(k+1) - x(k)}{T}$$

- As $T \to 0$, approximation error approaches 0



*Also known as the forward rectangle rule
†Just an approximation – more on this later

Back to the future

A quick note on causality:

- Calculating the "(k+1)th" value of a signal using

$$y(k+1) = \underbrace{x(k+1)}_{\text{future value}} + \underbrace{Ax(k) - By(k)}_{\text{current values}}$$

relies on also knowing the next (future) value of $x(t)$.

(this requires very advanced technology!)

- Real systems always run with a delay:

$$y(k) = x(k) + Ax(k-1) - By(k-1)$$

Back to the example!

```
T = 0.02;  //period of 50 Hz, a number pulled from thin air
A = 2*T-1; //pre-calculated control constants
B = T-1;

...

while(1)
   {
      if(interrupt_flag)          //this triggers every 20 ms
      {
         x0 = x;                  //save previous values
         y0 = y;
         x = update_input();      //get latest x value
         y = x + A*x0 - B*y0;     //do the difference equations
         update_output(y);        //write out current value
      }
   }
```

(The actual calculation)

Tustin's method

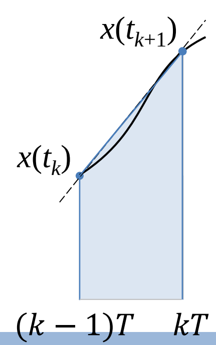
- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:

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

Taking the derivative, then z-transform yields:

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

which can be substituted into continuous models



Matched pole-zero

- If $z = e^{sT}$, why can't we just make a direct substitution and go home?

$$\frac{Y(s)}{X(s)} = \frac{s+a}{s+b} \Rightarrow \frac{Y(z)}{X(z)} = \frac{z-e^{-aT}}{z-e^{-bT}}$$

- Kind of!
  - Still an approximation
  - Produces quasi-causal system (hard to compute)
  - Fortunately, also very easy to calculate.

Matched pole-zero

The process:

1. Replace continuous poles and zeros with discrete equivalents:

$$(s+a) \Rightarrow (z - e^{-aT})$$

2. Scale the discrete system DC gain to match the continuous system DC gain

3. If the order of the denominator is higher than the enumerator, multiply the numerator by $(z+1)$ until they are of equal order*

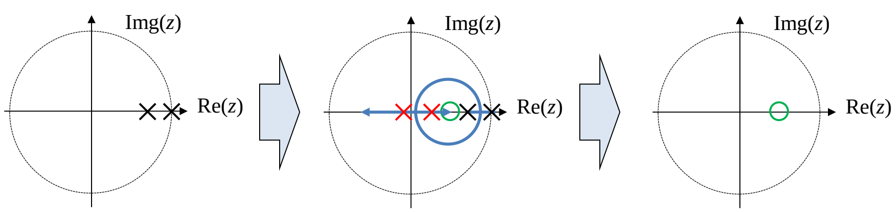
* This introduces an averaging effect like Tustin's method

Modified matched pole-zero

- We're prefer it if we didn't require instant calculations to produce timely outputs
- Modify step 2 to leave the dynamic order of the numerator one less than the denominator
  - Can work with slower sample times, and at higher frequencies

Discrete design process

1. Derive the dynamic system model ODE
2. Convert it to a discrete transfer function
3. Design a digital compensator
4. Implement difference equations in software
5. Platypus Is Divine!



Discrete design process

- Handy rules of thumb:
  - Sample rates can be as low as twice the system bandwidth
    - but 5 to 10× for "stability"
    - 20 to 30 × for better performance
  - A zero at $z = -1$ makes the discrete root locus pole behaviour more closely match the s-plane
  - Beware "dirty derivatives"
    - dy/dt terms derived from sequential digital values are called 'dirty derivatives' – these are especially sensitive to noise!
    - Employ actual velocity measurements when possible

Lead/Lag

Some standard approaches

- Control engineers have developed time-tested strategies for building compensators
- Three classical control structures:
  - Lead
  - Lag
  - Proportional-Integral-Derivative (PID)
    (and its variations: P, I, PI, PD)

How do they work?

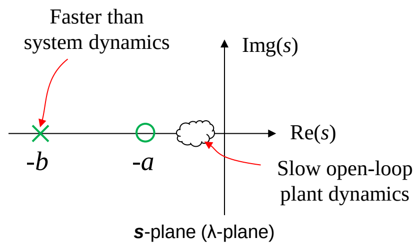
Lead/lag compensation

- Serve different purposes, but have a similar dynamic structure:

$$D(s) = \frac{s+a}{s+b}$$

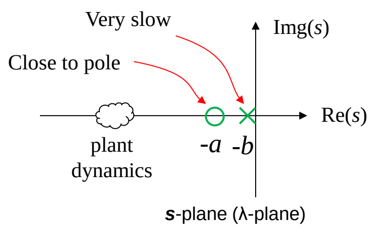
Note:
Lead-lag compensators come from the days when control engineers cared about constructing controllers from networks of op amps using frequency-phase methods. These days pretty much everybody uses PID, but you should at least know what the heck they are in case someone asks.

Lead compensation: $a < b$



- Acts to decrease rise-time and overshoot
  - Zero draws poles to the left; adds phase-lead
  - Pole decreases noise
- Set *a* near desired ω_n; set *b* at ~3 to 20x *a*

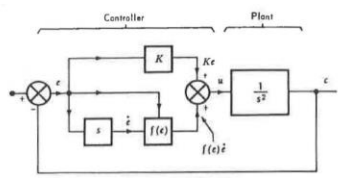
Lag compensation: $a > b$



- Improves steady-state tracking
  - Near pole-zero cancellation; adds phase-lag
  - Doesn't break dynamic response (too much)
- Set *b* near origin; set *a* at ~3 to 10x *b*

Poles are Eigenvalues: Some Implications

Stability of a 2^{nd} order regulator



$$u = Ke + f(e)\dot{e}$$

state equations let $e = x_1$ and $\dot{e} = x_2$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -Kx_1 - f(x_1)\, x_2$$

assume for simplicity that $K = 1$.

$$0 = x_2^0$$

$$0 = -x_1^0 - f(x_1^0)x_2^0$$

The Jacobian matrix is

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -f(0) \end{bmatrix}$$

- The linear behavior of the system in the close neighborhood of the origin is described by

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -x_1 - f(0)x_2$$

- AND, the characteristic equation is:

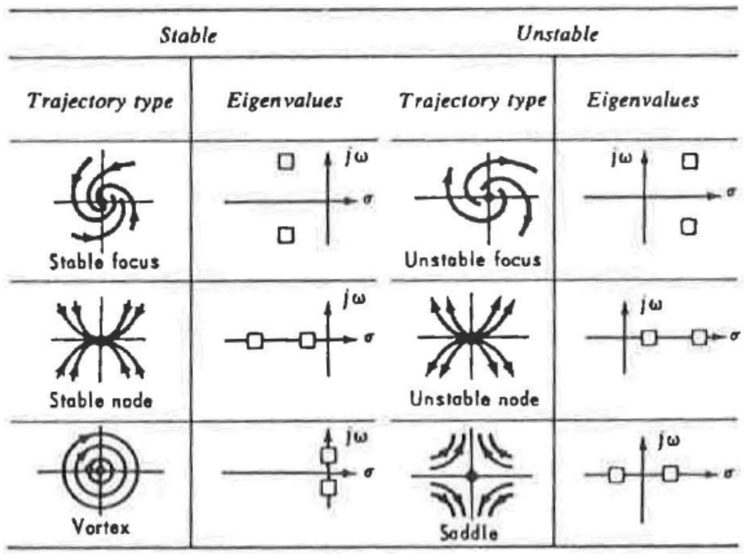
$$s[s + f(0)] + 1 = 0$$

with the eigenvalues

$$\lambda_1 = -\frac{1}{2}f(0) + \sqrt{\frac{1}{4}f^2(0) - 1}$$

$$\lambda_2 = -\frac{1}{2}f(0) - \sqrt{\frac{1}{4}f^2(0) - 1}$$

Various Types of Singularities (2nd order systems)



PID
(Intro)

PID – Control for the PID-dly minded

- Proportional-Integral-Derivative control is the control engineer's hammer*
  - For P,PI,PD, etc. just remove one or more terms

$$C(s) = k\left(\underbrace{1}_{\text{Proportional}} + \underbrace{\frac{1}{\tau i s}}_{\text{Integral}} + \underbrace{\tau d s}_{\text{Derivative}}\right)$$

*Everything is a nail. That's why it's called "Bang-Bang" Control ☺

PID – the Good Stuff

- PID control performance is driven by three parameters:
  - k: system gain
  - τ_i: integral time-constant
  - τ_d: derivative time-constant

You're already familiar with the effect of gain.
What about the other two?

Integral

- Integral applies control action based on accumulated output error
  - Almost always found with P control
- Increase dynamic order of signal tracking
  - Step disturbance steady-state error goes to zero
  - Ramp disturbance steady-state error goes to a constant offset

Let's try it!

Integral: P Control only

- Consider a first order system with a constant load disturbance, w; (recall as $t \to \infty$, $s \to 0$)

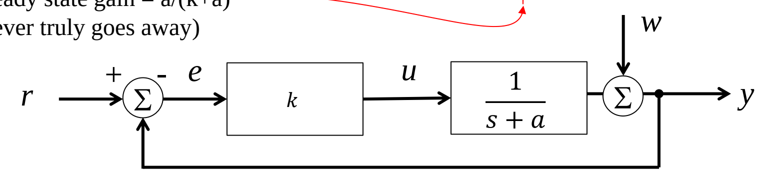
$$y = k\frac{1}{s+a}(r-y) + w$$

$$(s+a)y = k\,(r-y) + (s+a)w$$

$$(s+k+a)y = kr + (s+a)w$$

$$y = \frac{k}{s+k+a}r + \frac{(s+a)}{s+k+a}w$$

Steady state gain = a/(k+a)
(never truly goes away)



Now with added integral action

$$y = k\left(1 + \frac{1}{\tau_i s}\right)\frac{1}{s+a}(r - y) + w$$

$$y = k\frac{s + \tau_i^{-1}}{s}\frac{1}{s+a}(r - y) + w$$

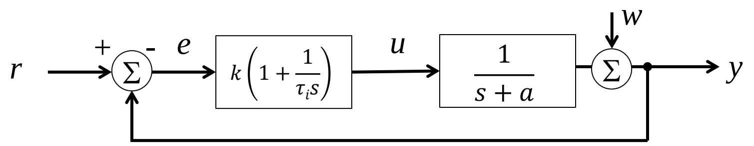
Same dynamics

$$s(s+a)y = k(s + \tau_i^{-1})(r - y) + s(s+a)w$$

$$(s^2 + (k+a)s + \tau_i^{-1})y = k(s + \tau_i^{-1})r + s(s+a)w$$

$$y = \frac{k(s + \tau_i^{-1})}{(s^2 + (k+a)s + \tau_i^{-1})}r + \frac{s(s+a)}{k(s + \tau_i^{-1})}w$$

Must go to zero for constant *w*!



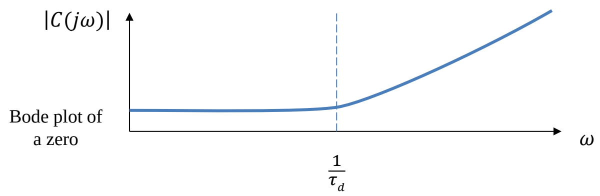
Derivative

- Derivative uses the rate of change of the error signal to anticipate control action
  - Increases system damping (when done right)
  - Can be thought of as 'leading' the output error, applying correction predictively
  - Almost always found with P control*

**What kind of system do you have if you use D, but don't care about position?* Is it the same as P control in velocity space?

Derivative

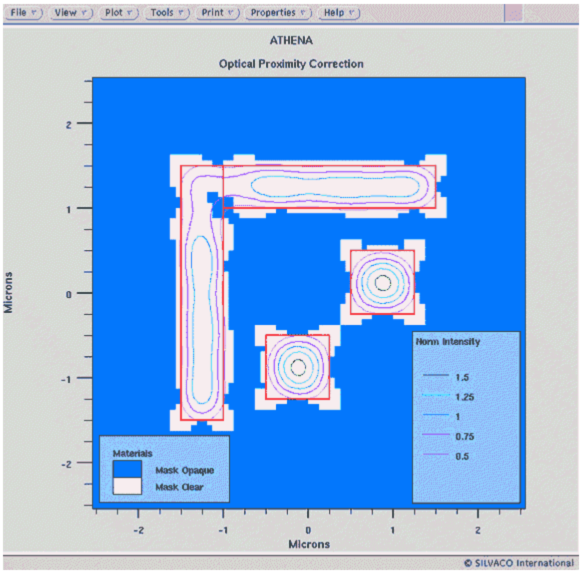
- It is easy to see that PD control simply adds a zero at $s = -\frac{1}{\tau_d}$ with expected results
  - Decreases dynamic order of the system by 1
  - Absorbs a pole as $k \to \infty$
- Not all roses, though: derivative operators are sensitive to high-frequency noise



PID

- Collectively, PID provides two zeros plus a pole at the origin
  - Zeros provide phase lead
  - Pole provides steady-state tracking
  - Easy to implement in microprocessors
- Many tools exist for optimally tuning PID
  - Zeigler-Nichols
  - Cohen-Coon
  - Automatic software processes

Break!: Fun Application: Optical Proximity Correction



Next Time…

- **PID!**

- Review:
  - PID notes online
  - Chapter 5 of FPW

- Deep Pondering??