http://elec3004.com

Servoregulation: Lead/Lag & PID

ELEC 3004: **Systems**: Signals & Controls
Dr. Surya Singh

Lecture 16

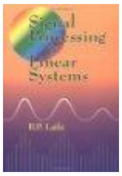
elec3004@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~elec3004/

May 4, 2017

Lecture Schedule:

Week	Date	Lecture Title
1	28-Feb	Introduction
	2-Mar	Systems Overview
2	7-Mar	Systems as Maps & Signals as Vectors
	9-Mar	Systems: Linear Differential Systems
3	14-Mar	Sampling Theory & Data Acquisition
	16-Mar	Aliasing & Antialiasing
4	21-Mar	Discrete Time Analysis & Z-Transform
	23-Mar	Second Order LTID (& Convolution Review)
5	28-Mar	Frequency Response
	30-Mar	Filter Analysis
6	4-Apr	Digital Filters (IIR) & Filter Analysis
	6-Apr	Digital Filter (FIR)
7	11-Apr	Digital Windows
	13-Apr	FFT
	18-Apr	Holiday
	20-Apr	
	25-Apr	
8	27-Apr	Active Filters & Estimation
9	2-May	Introduction to Feedback Control
	4-May	Servoregulation/PID
10	9-May	Introduction to State-Space
	11-May	(Digitial) State-Space Control
11	16-May	Digital Control Design
	18-May	Stability
12	23-May	Digital Control Systems: Shaping the Dynamic Response
	25-May	Applications in Industry
13	30-May	System Identification & Information Theory
	1-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

➔ **P - I - D**

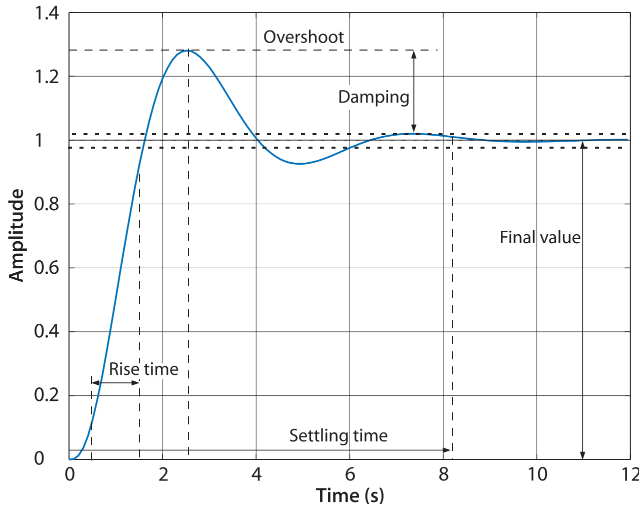
- FPW
  - Chapter 4:
    Discrete Equivalents to Continuous Transfer Functions: The Digital Filter

- FPW
  - Chapter 5: Design of Digital Control Systems Using Transform Techniques

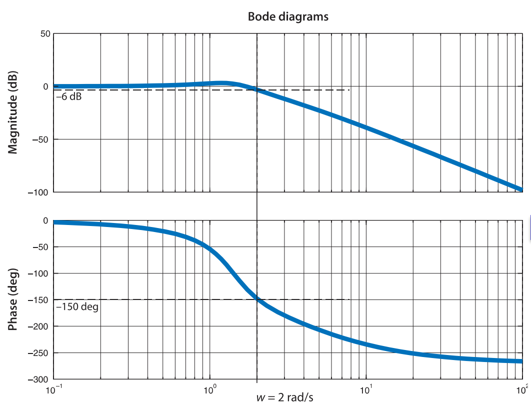
Next Time

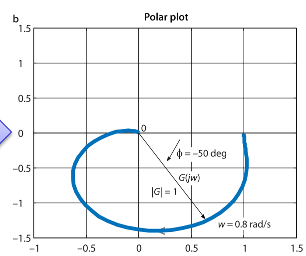
Feedback as a Filter

Time Response



Frequency Domain Analysis

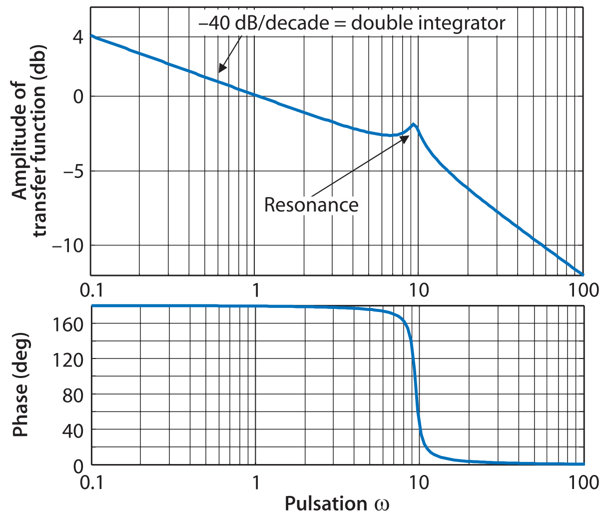




- Bode
(Magnitude + Phase Plots)
- Nyquist Plot
(Polar)

In This Way Feedback May Be Seen as a Filter

- Ex: Lightly Damped Robot Arm



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?

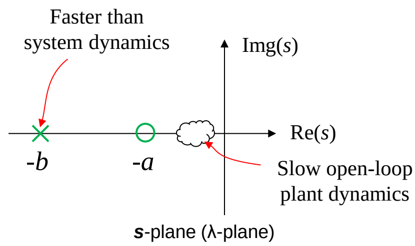
Lead/lag compensation

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

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

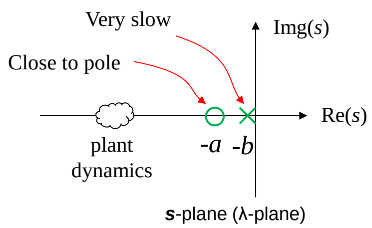
Note:
Lead-lag compensators come from the days when control engineers cared about constructing controllers from networks of op amps using frequency-phase methods. These days pretty much everybody uses PID, but you should at least know what the heck they are in case someone asks.

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

BREAK

PID
(Intro)

Proportional Control

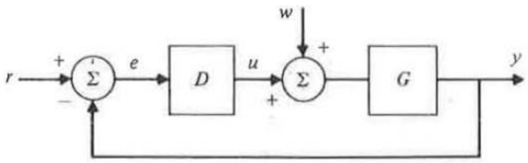
A discrete implementation of proportional control is identical to continuous; that is, where the continuous is

$$u(t) = K_p e(t) \quad \Rightarrow \quad D(s) = K_p,$$

the discrete is

$$u(k) = K_p e(k) \quad \Rightarrow \quad \boxed{D(z) = K_p}$$

where $e(t)$ is the error signal as shown in Fig 5.2.



Derivative Control

For continuous systems, derivative or rate control has the form

$$u(t) = K_p T_D \dot{e}(t) \quad \Rightarrow \quad D(s) = K_p T_D s$$

where T_D is called the *derivative time*. Differentiation can be approximated in the discrete domain as the first difference, that is,

$$u(k) = K_p T_D \frac{(e(k) - e(k-1))}{T} \quad \Rightarrow \quad \boxed{D(z) = K_p T_D \frac{1 - z^{-1}}{T} = K_p T_D \frac{z-1}{Tz.}}$$

In many designs, the compensation is a sum of proportional and derivative control (or PD control). In this case, we have

$$D(z) = K_p \left(1 + \frac{T_D(z-1)}{Tz}\right).$$

or, equivalently,

$$\boxed{D(z) = K \frac{z - \alpha}{z}}$$

Derivative Control [2]

- Similar to the lead compensators
  - The difference is that the pole is at $z = 0$

    [Whereas the pole has been placed at various locations along the z-plane real axis for the previous designs.]

- In the continuous case:
  - pure derivative control represents the ideal situation in that there is no destabilizing phase lag from the differentiation
  - the pole is at $s = -\infty$

- In the discrete case:
  - $z=0$
  - However this has phase lag because of the necessity to wait for one cycle in order to compute the first difference

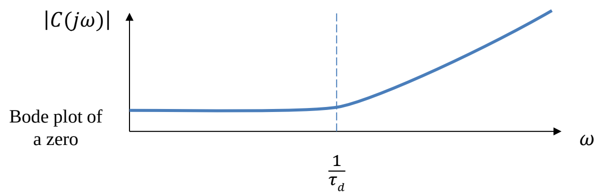
Derivative

- 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*

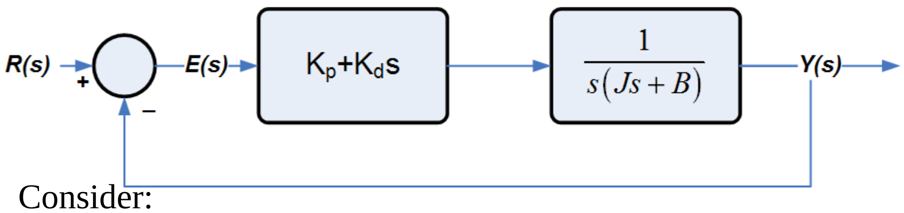
**What kind of system do you have if you use D, but don't care about position?* Is it the same as P control in velocity space?

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



PD for 2nd Order Systems



- Consider:

$$\frac{Y(s)}{R(s)} = \frac{(K_P + K_D s)}{Js^2 + (B + K_D)s + K_P}$$

- Steady-state error: $e_{ss} = \frac{B}{K_P}$
- Characteristic equation: $Js^2 + (B + K_D)s + K_P = 0$
- Damping Ratio: $\zeta = \frac{B+K_D}{2\sqrt{K_P J}}$

➔It is possible to make e_{ss} and overshoot small (↓) by making B small (↓), K_P large ↑, K_D such that ζ:between [0.4 – 0.7]

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 Control

For continuous systems, we integrate the error to arrive at the control,

$$u(t) = \frac{K_p}{T_I} \int_{t_o}^{t} e(t)dt \;\Rightarrow\; D(s) = \frac{K_p}{T_I s},$$

where T_I is called the *integral*, or *reset time*. The discrete equivalent is to sum all previous errors, yielding

$$u(k) = u(k-1) + \frac{K_p T}{T_I} e(k) \;\Rightarrow\; \boxed{D(z) = \frac{K_p T}{T_I(1 - z^{-1})} = \frac{K_p T z}{T_I(z-1)}.} \quad (5.60)$$

Just as for continuous systems, the primary reason for integral control is to reduce or eliminate steady-state errors, but this typically occurs at the cost of reduced stability.

Integral: P Control only

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

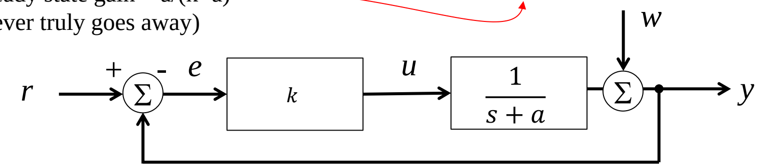
$$y = k\frac{1}{s+a}(r-y)+w$$

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

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

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

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



Now with added integral action

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

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

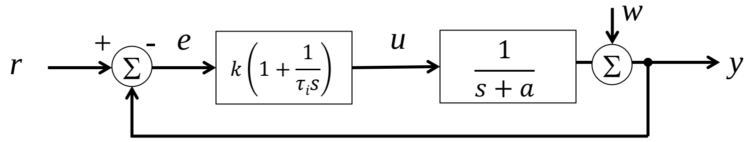
Same dynamics

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

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

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

Must go to zero
for constant w!



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

$$C(s) = k\left(1 + \frac{1}{\tau i s} + \tau d s\right)$$

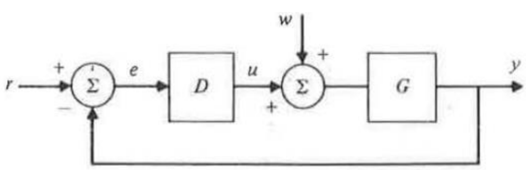
Proportional
Integral
Derivative

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

PID

- Three basic types of control:
  - Proportional
  - Integral, and
  - Derivative

- The next step up from lead compensation
  - Essentially a combination of proportional and derivative control



PID Control

$$D(z) = K_p\left(1 + \frac{Tz}{T_I(z-1)} + \frac{T_D(z-1)}{Tz}\right).$$

The user simply has to determine the best values of

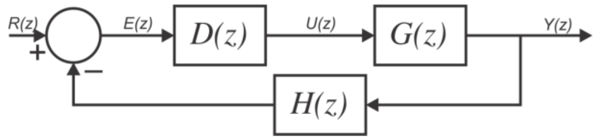
- K_p
- T_D and
- T_I

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

PID as Difference Equation

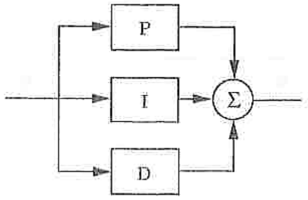


$$\frac{U(z)}{E(z)} = D(z) = K_p + K_i\left(\frac{Tz}{z-1}\right) + K_d\left(\frac{z-1}{Tz}\right)$$

$$u(k) = \left[K_p + K_i T + \left(\frac{K_d}{T}\right)\right] \cdot e(k) - [K_d T] \cdot e(k-1) + [K_i] \cdot u(k-1)$$

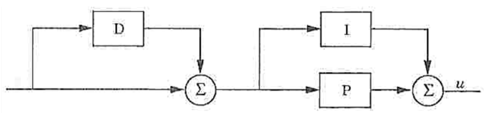
PID Implementation

- Non-Interacting



$$C(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right)$$

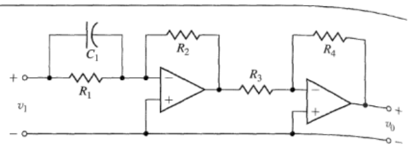
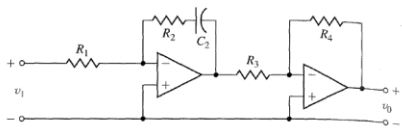
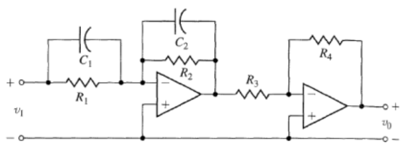
- Interacting Form



$$C'(s) = K\left(1 + \frac{1}{sT_i}\right)(1 + sT_d)$$

- Note: Different K,T_i and T_d

Operational Amplifier Circuits for Compensators

Type of Controller	$G_c(s) = \frac{V_0(s)}{V_1(s)}$	
PD	$G_c = \frac{R_4 R_2}{R_3 R_1}(R_1 C_1 s + 1)$	
PI	$G_c = \frac{R_4 R_2 (R_2 C_2 s + 1)}{R_3 R_1 (R_2 C_2 s)}$	
Lead or lag Lead if $R_1C_1 > R_2C_2$ Lag if $R_1C_1 < R_2C_2$	$G_c = \frac{R_4 R_2 (R_1 C_1 s + 1)}{R_3 R_1 (R_2 C_2 s + 1)}$	

- (Yet Another Way to See PID)

Source: Dorf & Bishop, *Modern Control Systems*, p. 828

PID Algorithm (in various domains):

FPW § 5.8.4 [p.224]

- PID Algorithm (in Z-Domain):

$$D(z) = K_p\left(1 + \frac{Tz}{T_I(z-1)} + \frac{T_D(z-1)}{Tz}\right)$$

- As Difference equation:

$$u(t_k) = u(t_{k-1}) + K_p\left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t}\right)e(t_k) + \left(-1 - \frac{2T_d}{\Delta t}\right)e(t_{k-1}) + \frac{T_d}{\Delta t}e(t_{k-2})\right]$$

- Pseudocode [Source: Wikipedia]:

```
previous_error = 0, integral = 0
start:
  error = setpoint - measured_value
  integral = integral + error*dt
  derivative = (error - previous_error)/dt
  output = Kp*error + Ki*integral + Kd*derivative
  previous_error = error
  wait(dt)
  goto start
```

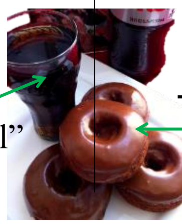
Another way to see P I|D

- Derivative

  D provides:
  - High sensitivity
  - Responds to change
  - Adds "damping" & ∴ permits larger K_P
  - Noise sensitive
  - Not used alone
    (∵ its on rate change of error – by itself it wouldn't get there)

→ "**D**iet Coke of control"

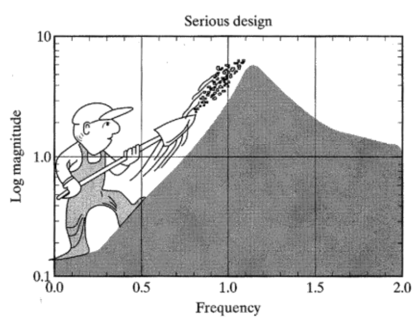
- Integral
  - Eliminates offsets (makes regulation ☺)
  - Leads to Oscillatory behaviour
  - Adds an "order" but instability
    (Makes a 2nd order system 3rd order)



→ "**I**nteresting cake of control"

Seeing PID – No Free Lunch

- The energy (and sensitivity) moves around (in this case in "frequency")



- Sensitivity reduction at low frequency unavoidably leads to sensitivity increase at higher frequencies.

Source: Gunter Stein's interpretation of the water bed effect – G. Stein, *IEEE Control Systems Magazine*, 2003.

PID Intuition & Tuning

- Tuning – How to get the “magic” values:
  - Dominant Pole Design
  - Ziegler Nichols Methods
  - Pole Placement
  - Auto Tuning

- Although PID is common it is often poorly tuned
  - The derivative action is frequently switched off!
  (Why ∵ it’s sensitive to noise)
  - Also lots of “I” will make the system more transitory & leads to integrator wind-up.

PID Intuition

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int e(s)\,ds + T_d\frac{de(t)}{dt}\right]$$

- P:
  - Control action is proportional to control error
  - It is necessary to have an error to have a non-zero control signal

- I:
  - The main function of the integral action is to make sure that the process output agrees with the set point in steady state

PID Intuition

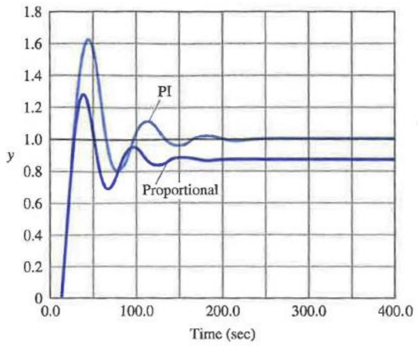
$$u(t) = K\left[e(t) + \frac{1}{T_i}\int e(s)\,ds + T_d\frac{de(t)}{dt}\right]$$

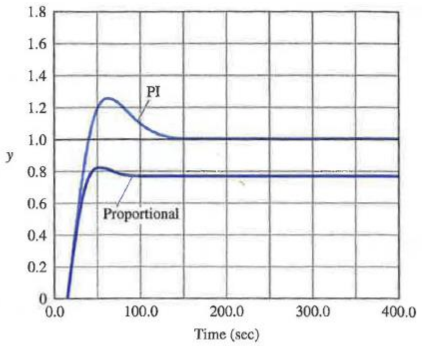
- P:
- I:
- D:
  - The purpose of the derivative action is to improve the closed loop stability.
  - The instability "mechanism" "controlled" here is that because of the process dynamics it will take some time before a change in the control variable is noticeable in the process output.
  - The action of a controller with proportional and derivative action may e interpreted as if the control is made proportional to the *predicted* process output, where the prediction is made by extrapolating the error by the tangent to the error curve.

PID Intuition

Effects of increasing a parameter independently					
Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	↓	⇑	Minimal change	↓	↓
K_I	↓	⇑	⇑	Eliminate	↓
K_D	Minor change	↓	↓	No effect / minimal change	Improve (if K_D small)

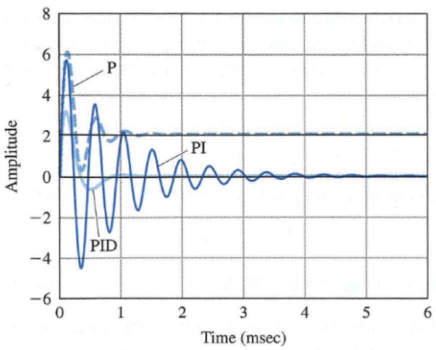
PID Intuition: P and PI



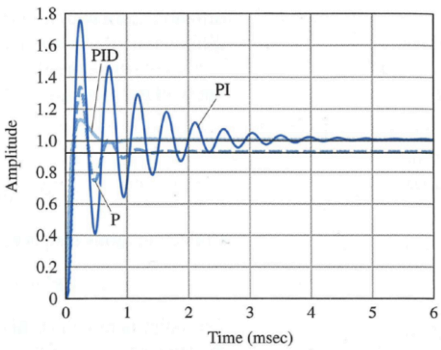


PID Intuition: P and PI and PID

- Responses of P, PI, and PID control to



(a) step disturbance input



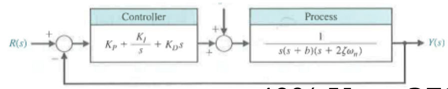
(b) step reference input

PID Example

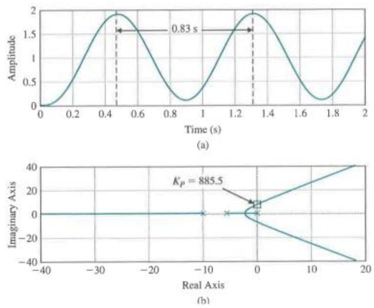
- A 3rd order plant: b=10, ζ=0.707, ω_n=4

$$G(s) = \frac{1}{s(s+b)(s+2\zeta\omega_n)}$$

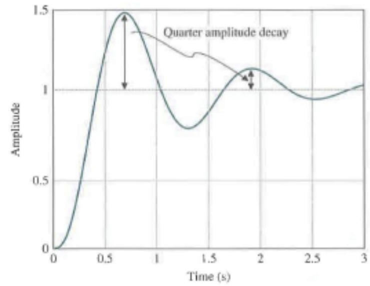
- PID:



- Kp=855:



- 40% Kp = 370



Ziegler-Nichols Tuning – Reaction Rate

FPW § 5.8.5 [p.224]

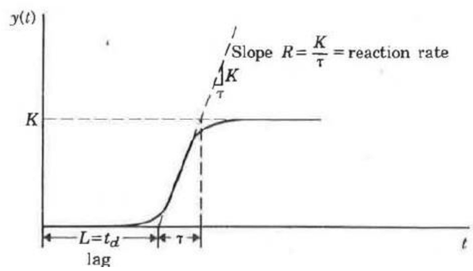
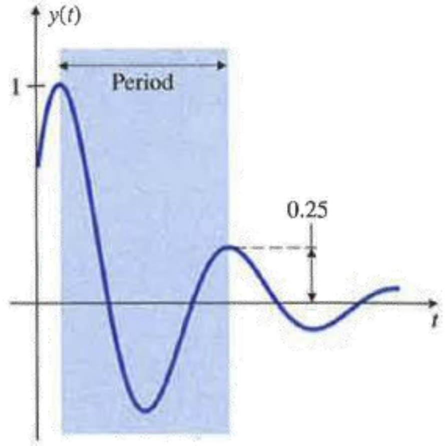


Table 5.2 Ziegler-Nichols tuning parameters using transient response.

	K_p	T_I	T_D
P	$1/RL$		
PI	$0.9/RL$	$3L$	
PID	$1.2/RL$	$2L$	$0.5L$

Quarter decay ratio



Ziegler-Nichols Tuning – Stability Limit Method

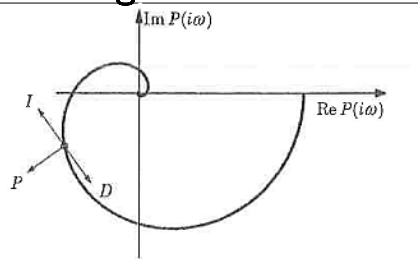
FPW § 5.8.5 [p.226]

- Increase K_P until the system has continuous oscillations
  - ≡ K_U : Oscillation Gain for "Ultimate stability"
  - ≡ P_U : Oscillation Period for "Ultimate stability"

Table 5.3 Ziegler-Nichols tuning parameters using stability limit.

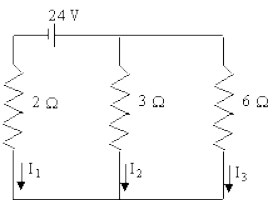
	K_p	T_I	T_D
P	$0.5K_u$		
PI	$0.45K_u$	$P_u/1.2$	
PID	$0.6K_u$	$P_u/2$	$P_u/8$

Ziegler-Nichols Tuning



$$C(i\omega_u) = K\left(1 + i\left(\omega_u T_d - \frac{1}{\omega_u T_i}\right)\right) \approx 0.6K_u(1 + 0.467i)$$

Break!: Fun Application: Linear Algebra & KVL!



We can write this as:

$$\begin{pmatrix} 1 & 1 & 1 \\ -2 & 3 & 0 \\ 0 & -3 & 6 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 24 \\ 0 \end{pmatrix}$$

So we have:

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -2 & 3 & 0 \\ 0 & -3 & 6 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 24 \\ 0 \end{pmatrix}$$

Using a computer algebra system to perform the inverse and multiply by the constant matrix, we get:

$$I_1 = -6 \text{ A}$$

$$I_2 = 4 \text{ A}$$

$$I_3 = 2 \text{ A}$$

We observe that I_1 is negative, as expected from the circuit diagram.

Source: http://www.intmath.com/matrices-determinants/6-matrices-linear-equations.php

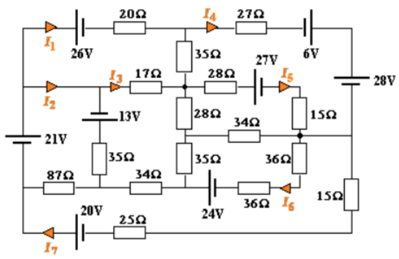
Break!: Fun Application: Linear Algebra & KCL!

We solve this using a computer as follows. We just write the coefficient matrix on the left, find the inverse (raise the matrix to the power -1) and multiply the result by the constant matrix.

You can use Matlab, Mathcad or similar math software to do this. Wolfram|Alpha is a free alternative.

$$X = \begin{bmatrix} 72 & 0 & -17 & -35 & 0 & 0 & 0 \\ 0 & 122 & -35 & 0 & 0 & 0 & -87 \\ 0 & -87 & -34 & 0 & 0 & -72 & 233 \\ -17 & -35 & 149 & 0 & -28 & -35 & -34 \\ 0 & 0 & -28 & -43 & 105 & -34 & 0 \\ 0 & 0 & -35 & 0 & -34 & 141 & -72 \\ -35 & 0 & 0 & 105 & -43 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -26 \\ 34 \\ -4 \\ -13 \\ -27 \\ 24 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} -0.46801 \\ 0.42932 \\ 5.193 \times 10^{-3} \\ -0.22243 \\ -0.27848 \\ 0.21115 \\ 0.20914 \end{bmatrix}$$



Source: http://www.intmath.com/matrices-determinants/6-matrices-linear-equations.php

Next Time…

- **Digital Feedback Control**

- Review:
  - Chapter 2 of FPW

- More Pondering??

Extension!
Design by Emulation

Two cases for control design

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

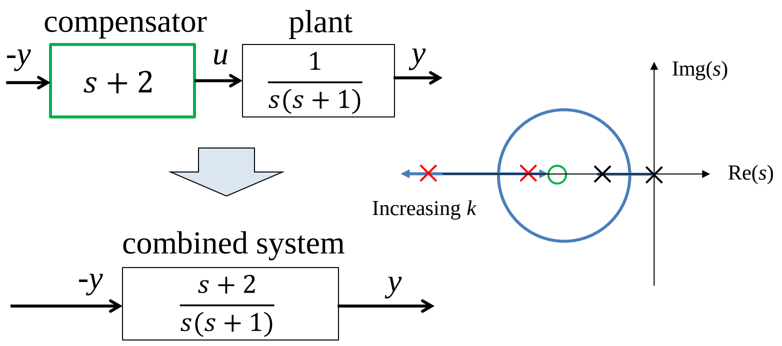
("Performance")

- Attempts to spontaneously disassemble itself

("Stability")

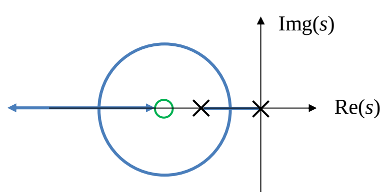
Dynamic compensation

- 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?

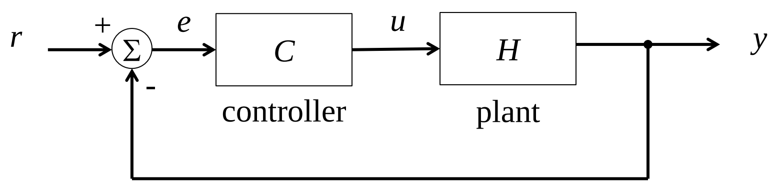
- Recognise the following:
  - A root locus starts at poles, terminates at zeros
  - "Holes eat poles"
  - Closely matched pole and zero dynamics cancel
  - The locus is on the real axis to the left of an odd number of poles (treat zeros as 'negative' poles)



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)

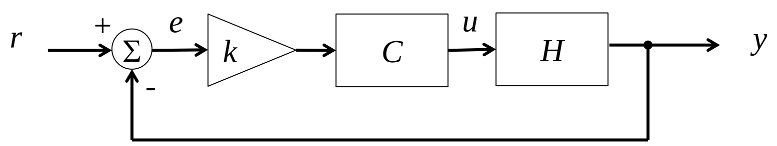
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

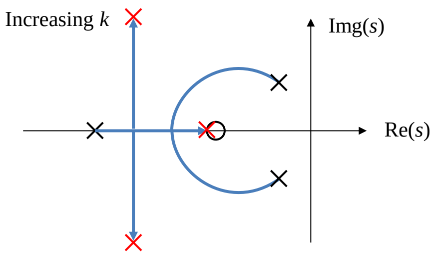
$$\frac{y}{r} = \frac{kC_n H_n}{C_d H_d + kC_n H_n}$$

characteristic polynomial



The Root Locus (Quickly)

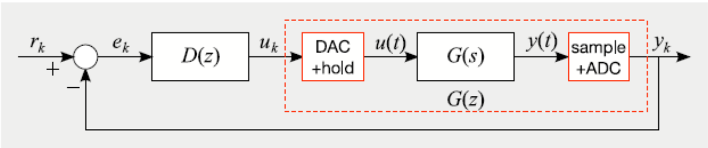
- Pole positions change with increasing gain
  - The trajectory of poles on the pole-zero plot with changing k is called the "root locus"
  - This is sometimes quite complex



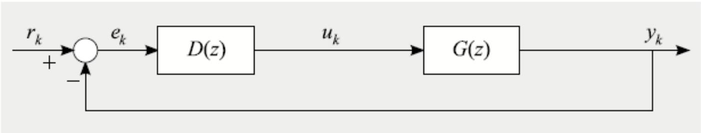
(In practice you'd plot these with computers)

Designing in the Purely Discrete…

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



by considering the purely discrete time system:



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

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/law: $\frac{1-\alpha z^{-1}}{1-\beta z^{-1}}$

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 $t_1 \le t_2 \le \cdots$: 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)

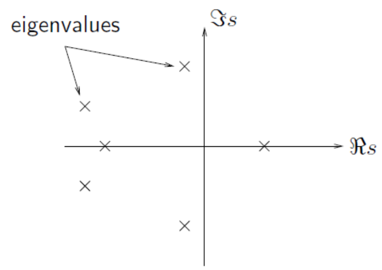
eigenvalues determine (possible) qualitative behavior of x:

- 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:

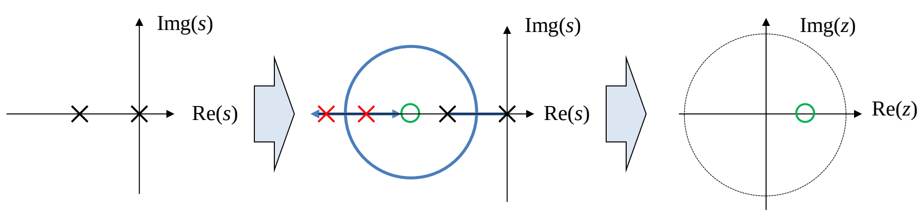
  eg. $s^2 + 2s + 1 = (s + 1)^2$

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

  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

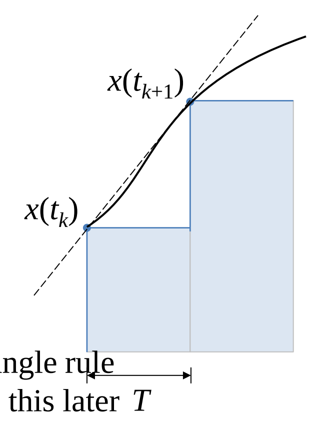
- Handy rules of thumb:
  - Use a sampling period of 20 to 30 times faster than the closed-loop system bandwidth
  - Remember that the sampling ZOH induces an effective $T/2$ delay
  - There are several approximation techniques:
    - Euler's method
    - Tustin's method
    - Matched pole-zero
    - 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 $T \rightarrow 0$, approximation error approaches 0



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

An example!

Convert the system $\frac{Y(s)}{X(s)} = \frac{s+2}{s+1}$ into a difference equation with period T, using Euler's method.

1. Rewrite the function as a dynamic system:

$$sY(s) + Y(s) = sX(s) + 2X(s)$$

Apply inverse Laplace transform:

$$\dot{y}(t) + y(t) = \dot{x}(t) + 2x(t)$$

2. Replace continuous signals with their sampled domain equivalents, and differentials with the approximating function

$$\frac{y(k+1) - y(k)}{T} + y(k) = \frac{x(k+1) - x(k)}{T} + 2x(k)$$

An example!

Simplify:

$$y(k+1) - y(k) + Ty(k) = x(k+1) - x(k) + 2Tx(k)$$
$$y(k+1) + (T-1)y(k) = x(k+1) + (2T-1)x(k)$$

$$y(k+1) = x(k+1) + (2T-1)x(k) - (T-1)y(k)$$

We can implement this in a computer.

Cool, let's try it!

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

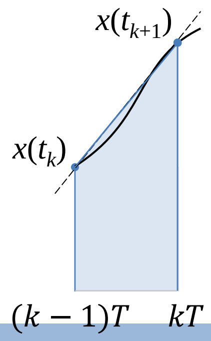
- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:

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

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*

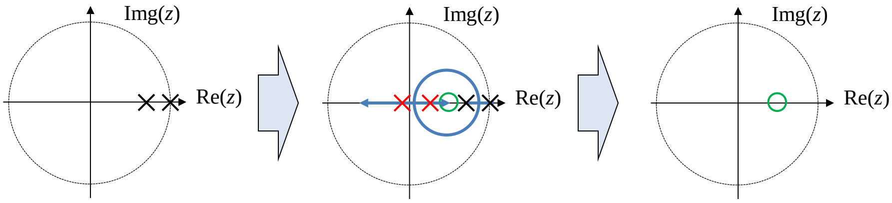
   * This introduces an averaging effect like Tustin's method

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

Extension!

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: $Y(z) = \frac{1}{2}\frac{z}{(z - re^{j\theta})} + \frac{1}{2}\frac{z}{(z - re^{-j\theta})}$

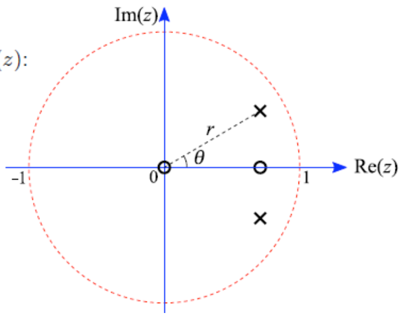
$$= \frac{z(z - r\cos\theta)}{(z - re^{j\theta})(z - re^{-j\theta})}$$

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

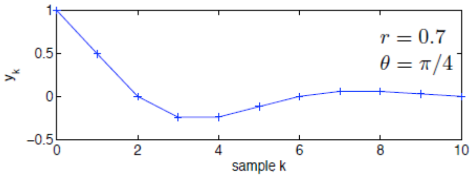
zeros: $\begin{cases} z = 0 \\ z = r\cos\theta \end{cases}$



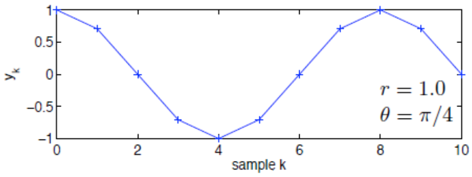
Response of 2nd order system [1/3]

Responses for varying r:

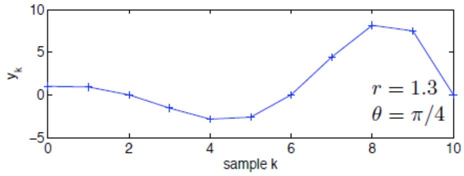
- $r < 1$
  $\Downarrow$
  exponentially decaying envelope



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



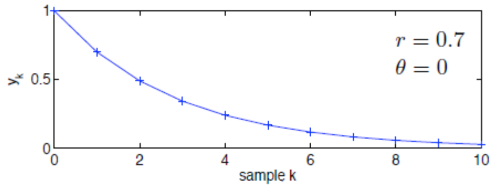
- $r > 1$
  $\Downarrow$
  exponentially increasing envelope



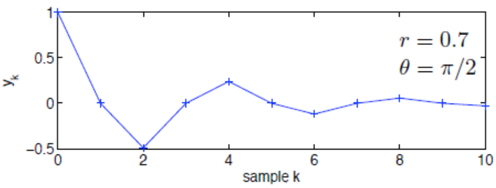
Response of 2nd order system [2/3]

Responses for varying θ:

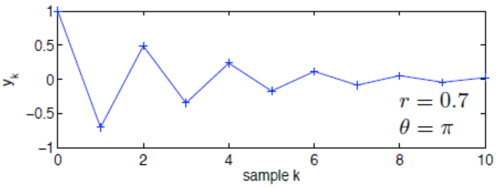
- $\theta = 0$
  $\Downarrow$
  decaying exponential



- $\theta = \pi/2$
  $\Downarrow$
  $2\pi/\theta = 4$ 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 $r = 0$:

$$Y(z) = 1$$

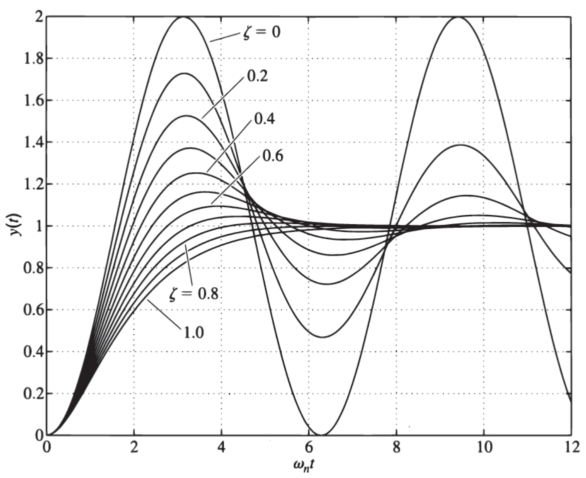
$\Longrightarrow$ unit pulse

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

samples of alternating signs

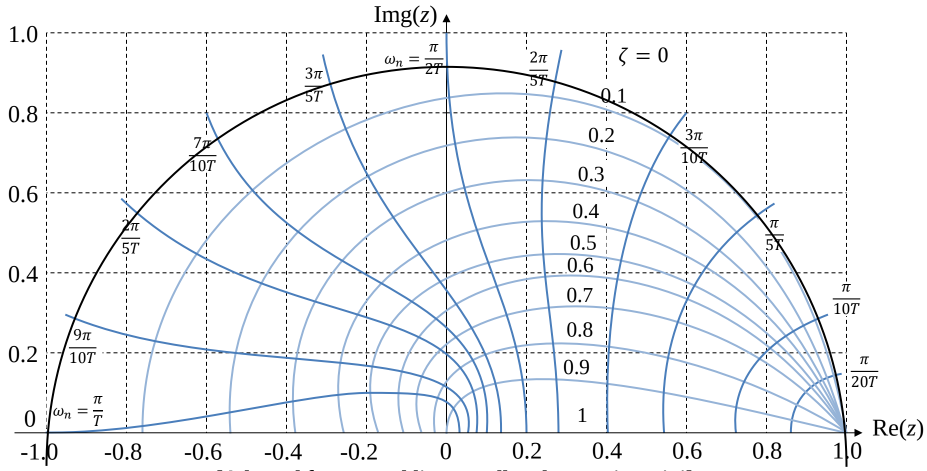
2nd Order System Response

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



Damping and natural frequency

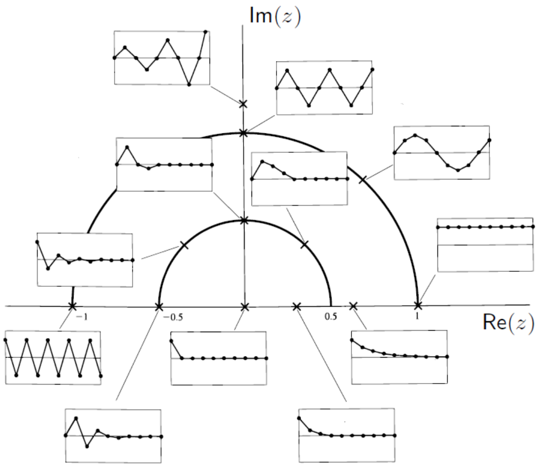
$$z = e^{sT} \text{ where } s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$



[Adapted from Franklin, Powell and Emami-Naeini]

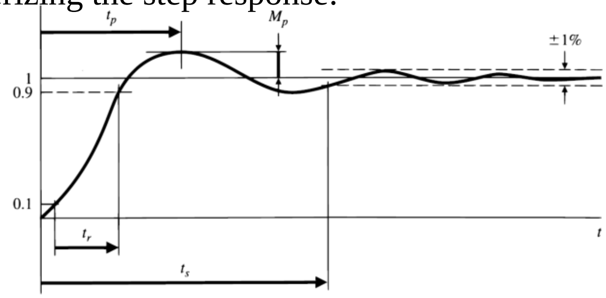
Pole positions in the z-plane

- 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:



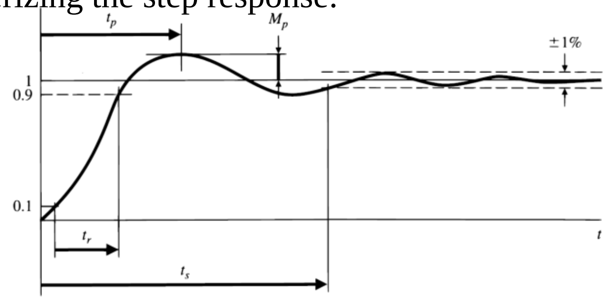
- Rise time (10% → 90%): $t_r \approx \frac{1.8}{\omega_0}$
- Overshoot: $M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1-\zeta^2}}$
- Settling time (**<u>to 1%</u>**): $t_s = \frac{4.6}{\zeta\omega_0}$
- Steady state error to unit step: e_{ss}
- Phase margin:

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

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= 1s
- Controller:

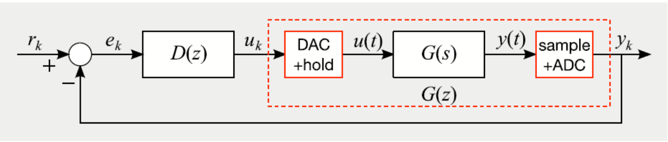
$$u_k = -0.5u_{k-1} + 13\left(e_k - 0.88e_{k-1}\right)$$

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:

$$G(z) = \frac{(z-1)}{z}\,\frac{z\Big((0.1 - 1 + e^{-0.1})z + (1 - e^{-0.1} - 0.1e^{-0.1})\Big)}{0.1(z-1)^2(z - e^{-0.1})}$$

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

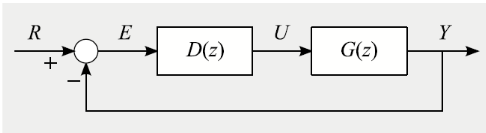
(b) Find the controller transfer function (using z = shift operator):

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

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

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

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



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

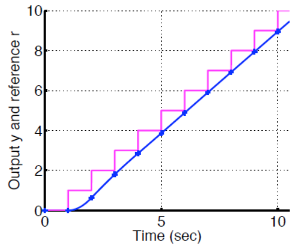
$$R(z) = \frac{Tz}{(z-1)^2}$$

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

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

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

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



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

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

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

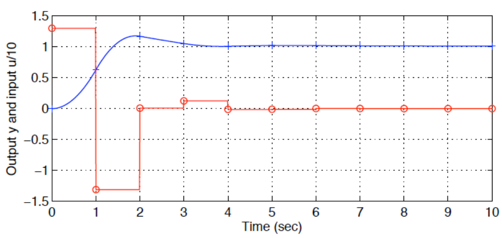
The closed loop poles are the roots of $1 + D(z)G(z) = 0$, i.e.

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

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

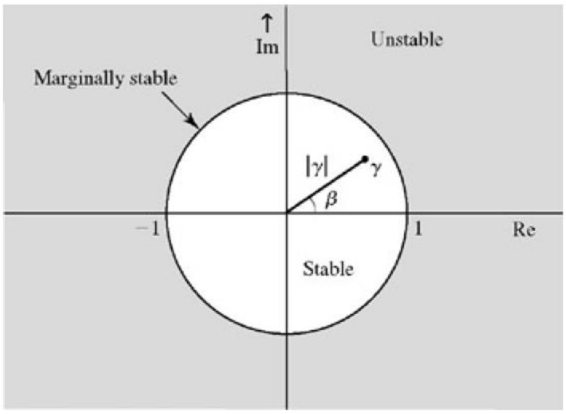
But the pole at $z = 0.88$ is cancelled by controller zero at $z = 0.88$, and

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

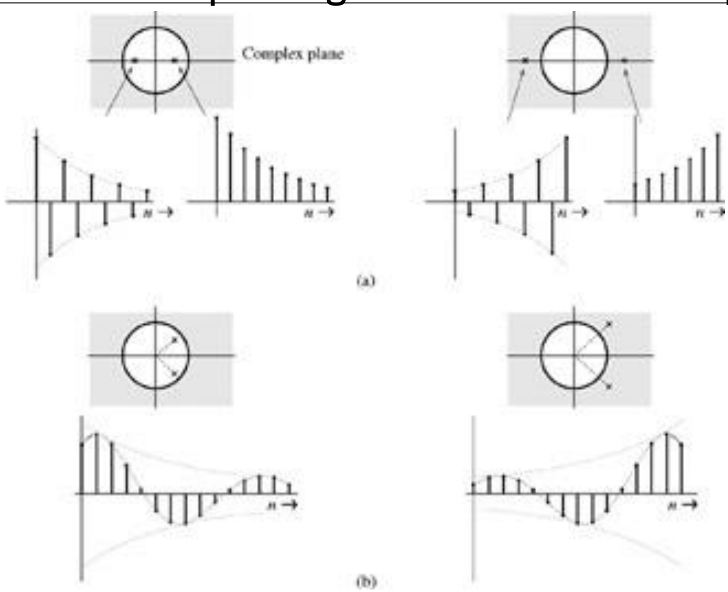


all specs satisfied!

LTID Stability



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



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