



<http://elec3004.com>

# Introduction to Feedback Control

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

Lecture 15  
(with material from Lathi)

[elec3004@itee.uq.edu.au](mailto:elec3004@itee.uq.edu.au)

April 28, 2016

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

© 2016 School of Information Technology and Electrical Engineering at The University of Queensland

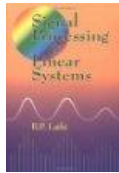


## Lecture Schedule:

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	Holiday
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
	25-Apr	Holiday
<b>8</b>	<b>28-Apr</b>	<b>Introduction to Feedback Control</b>
9	3-May	Holiday
	5-May	Servoregulation/PID
10	9-May	Introduction to (Digital) Control
	12-May	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

- Chapter 6  
 (Continuous-Time System Analysis  
 Using the Laplace Transform)
  - § 6.3 Solution of Differential and  
 Integro-Differential Equations
  - § 6.5 Block Diagrams
  - → § 6.4 KVL | KCL made easy!

- FPW
  - Chapter 3: Sampled-Data Systems
  - Chapter 2: Linear, Discrete,  
 Dynamic-Systems Analysis

Next Time



## Announcements

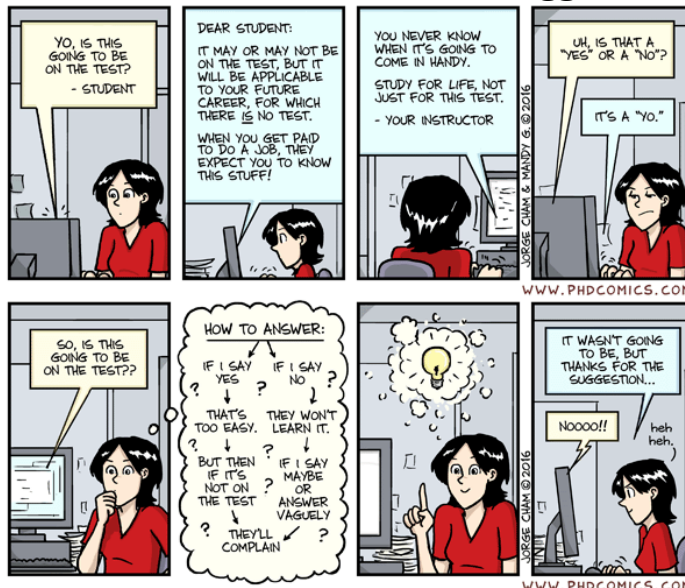
- Please peer review!
  - PS 1:  
 679 of 3359 peer reviews  
 completed.
  - ☹
- PS 1 Grades Out!
- To Median Or Not To ....

	MEDIAN>Tutor	Median<TUTOR
Q1	83	24
Q2	63	16
Q3	88	13
Q4	60	2
Q5	52	8

- Why?
  - Giving is the best reward
  - Review is the best learning!



## Announcements: “Thanks for the Suggestion” 😊

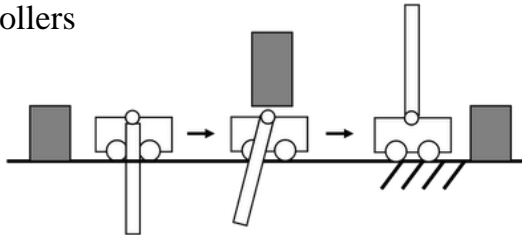


## Introducing Feedback

## Control

Once upon a time...

- Electromechanical systems were controlled by electromechanical compensators
  - Mechanical flywheel governors, capacitors, inductors, resistors, relays, valves, solenoids (fun!)
  - But also complex and sensitive!
- Humans developed sophisticated tools for designing reliable analog controllers



## Control

Once upon a time...

- Electromechanical systems were controlled by electromechanical compensators
  - Mechanical flywheel governors, capacitors, inductors, resistors, relays, valves, solenoids (fun!)
  - But also complex and sensitive!
- ➔ Idea: Digital computers in real-time control
  - Transform approach (classical control)
    - Root-locus methods (pretty much the same as METR 3200)
    - Bode's frequency response methods (these change compared to METR 3200)
  - State-space approach (modern control)
- ➔ Model Making: Control of frequency response as well as Least Squares Parameter Estimation



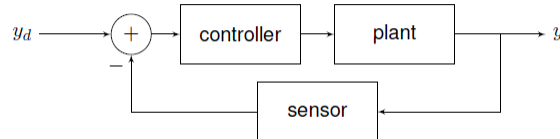
## Many advantages

- Practical improvement over analog control:
  - **Flexible**; reprogrammable to implement different control laws for different systems
  - **Adaptable**; control algorithms can be changed on-line, during operation
  - **Insensitive** to environmental conditions; (heat, EMI, vibration, etc.)
  - **Compact**; handful of components on a PCB
  - **Cheap**



## Feedback Control

(Simple) control systems have three parts:

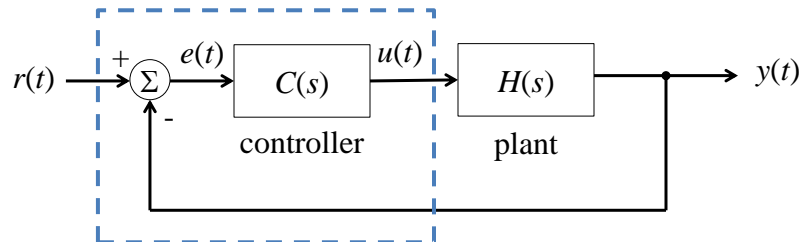


- The plant is the system to be controlled (e.g. the robot).
- The sensor measures the output of the plant.
- The controller sends an input command to the plant based on the difference from the actual output and the desired output.



## Archetypical control system

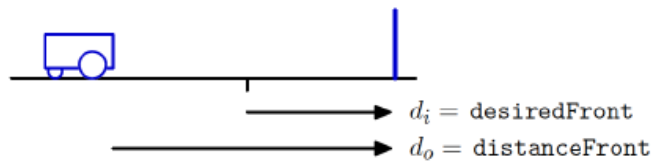
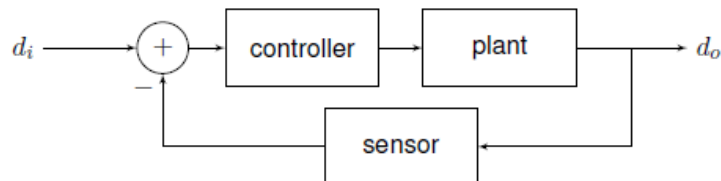
- Consider a continuous control system:



- The functions of the controller can be entirely represented by a discretised computer system



## Simple Controller Goes Digital



$$\text{plant: } y[n] = y[n - 1] - Tu[n - 1]$$

$$\text{sensor: } y[n] = u[n - 1]$$

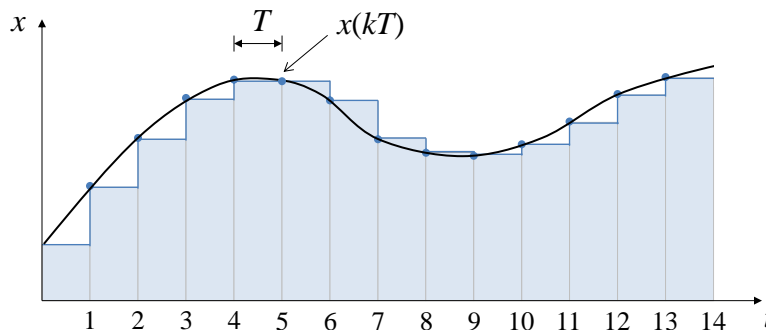
$$\text{controller: } y[n] = Ku[n]$$

Complex system behaviors, depending on  $K$



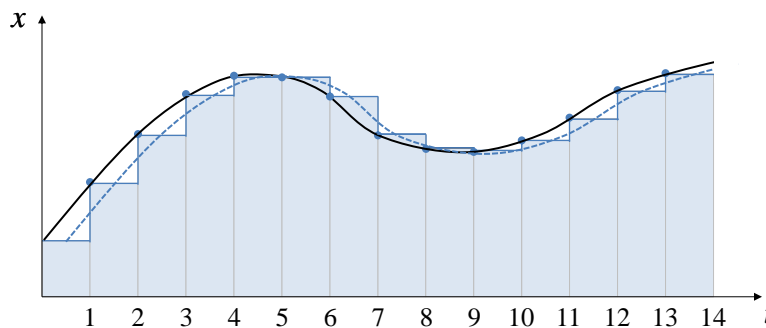
## Return to the discrete domain

- Recall that continuous signals can be represented by a series of samples with period  $T$



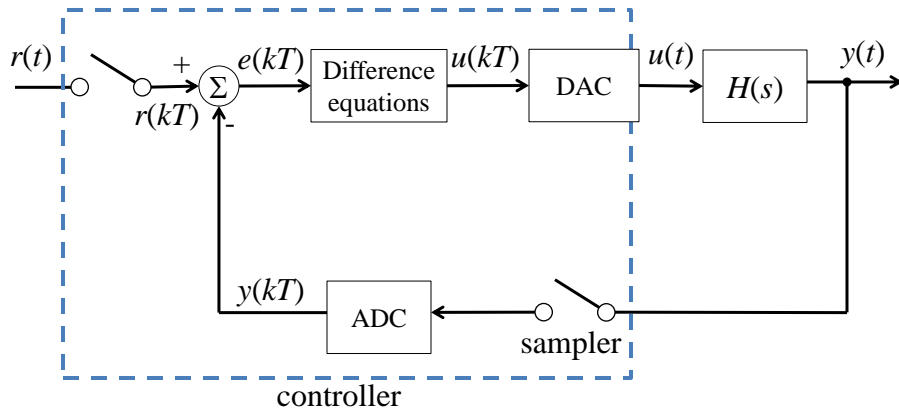
## Zero Order Hold

- An output value of a synthesised signal is held constant until the next value is ready
  - This introduces an effective delay of  $T/2$



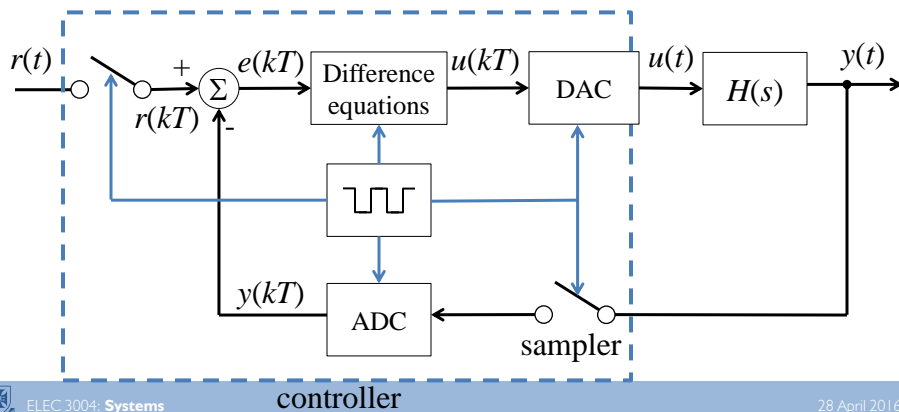
## Digitisation

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



## Digitisation

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



## Difference equations

- How to represent differential equations in a computer?  
Difference equations!
- The output of a difference equation system is a function of current and previous values of the input and output:

$$y(t_k) = D(x(t_k), x(t_{k-1}), \dots, x(t_{k-n}), y(t_{k-1}), \dots, y(t_{k-n}))$$

- We can think of  $x$  and  $y$  as parameterised in  $k$   
Useful shorthand:  $x(t_{k+i}) \equiv x(k+i)$



## → Discrete-time transfer function

take  $\mathcal{Z}$ -transform of system equations

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t)$$

yields

$$zX(z) - zx(0) = AX(z) + BU(z), \quad Y(z) = CX(z) + DU(z)$$

solve for  $X(z)$  to get

$$X(z) = (zI - A)^{-1}zx(0) + (zI - A)^{-1}BU(z)$$

(note extra  $z$  in first term!)

hence

$$Y(z) = H(z)U(z) + C(zI - A)^{-1}zx(0)$$

where  $H(z) = C(zI - A)^{-1}B + D$  is the *discrete-time transfer function*

note power series expansion of resolvent:

$$(zI - A)^{-1} = z^{-1}I + z^{-2}A + z^{-3}A^2 + \dots$$



# How to Design? Back to Analog !

## Two cases for control design

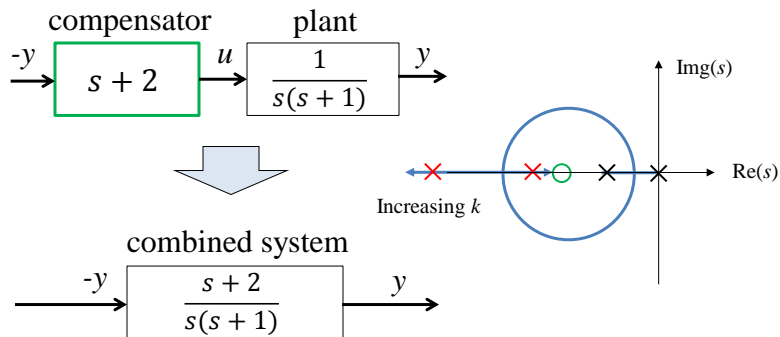
The system...

- Isn't fast enough
- Isn't damped enough
- Overshoots too much
- Requires too much control action  
(“Performance”)
  
- Attempts to spontaneously disassemble itself  
(“Stability”)



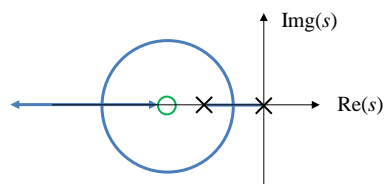
## Dynamic compensation

- 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)



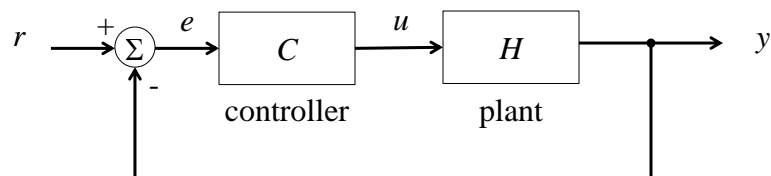
## The Root Locus (Quickly)

- 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)

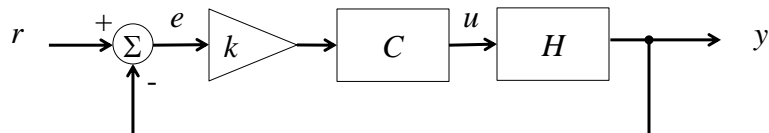
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

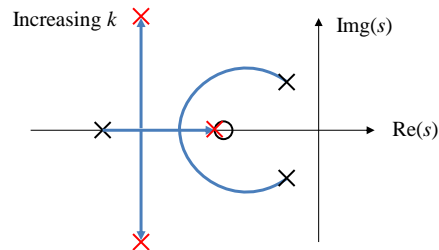
$$\frac{y}{r} = \frac{kC_n H_n}{C_d H_d + kC_n H_n}$$

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

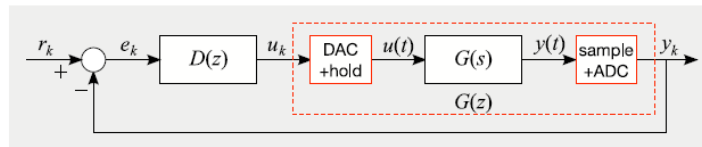


(In practice you'd plot these with computers)

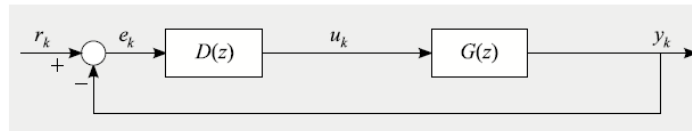


## Designing in the Purely Discrete...

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



by considering the purely discrete time system:



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

- How do the closed loop poles relate to
- stability?
  - performance?



## Now in discrete

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

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

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

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  $t_1 \leq t_2 \leq \dots$ : 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)



## 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  $0 < t_1 < t_2 < \dots$  (sometimes called jump linear system)

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

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

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

$i$ th 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 $\mathbf{x}(t)$ [2]

first assume eigenvalues  $\lambda_i$  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  $\beta_{ij}$  depend on  $x(0)$  (linearly)

eigenvalues determine (possible) qualitative behavior of  $x$ :

- eigenvalues give exponents that can occur in exponentials
- real eigenvalue  $\lambda$  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 $\mathbf{x}(t)$ [3]

first assume eigenvalues  $\lambda_i$  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  $\beta_{ij}$  depend on  $x(0)$  (linearly)

eigenvalues determine (possible) qualitative behavior of  $x$ :

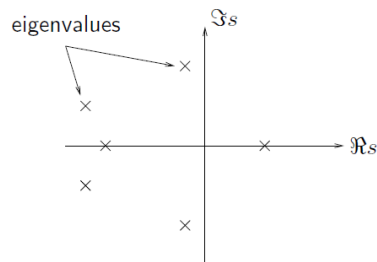
- eigenvalues give exponents that can occur in exponentials
- real eigenvalue  $\lambda$  corresponds to an exponentially decaying or growing term  $e^{\lambda t}$  in solution
- complex eigenvalue  $\lambda = \sigma + j\omega$  corresponds to decaying or growing sinusoidal term  $e^{\sigma t} \cos(\omega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

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



## Qualitative behaviour of $\mathbf{x}(t)$ [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 $\mathbf{x}(t)$ [5]

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

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

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:

$$\text{eg. } s^2 + 2s + 1 = (s + 1)^2$$

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

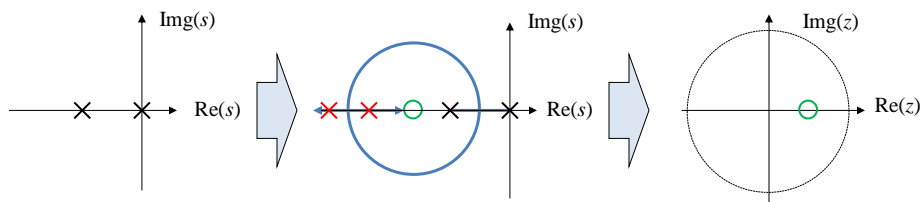
Root loci behave the same on both planes!

- 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

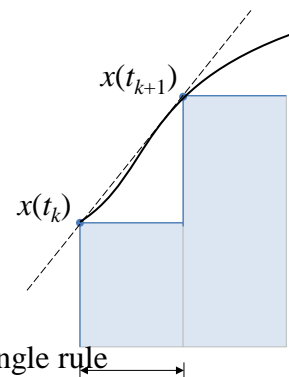


## Euler's method\*

- Dynamic systems can be approximated<sup>†</sup> 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  $T$



## 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)



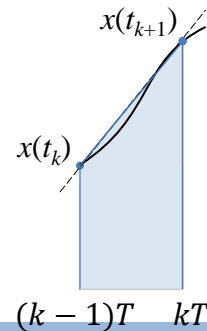
## Tustin's method

- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:
$$u(kT) = \frac{T}{2} [x(k-1) + x(k)]$$

Taking the derivative, then z-transform yields:

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

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 numerator, multiply the numerator by  $(z + 1)$  until they are of equal order\*

\* This introduces an averaging effect like Tustin's method



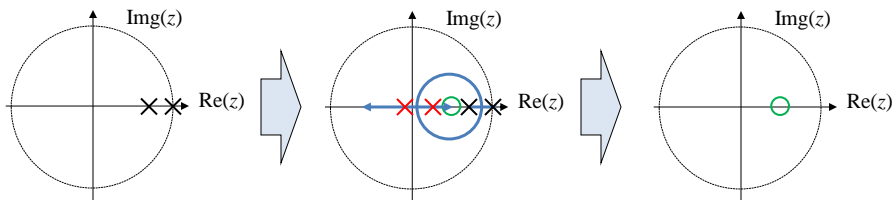
## Modified matched pole-zero

- We 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 $\times$  for “stability”
    - 20 to 30  $\times$  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:

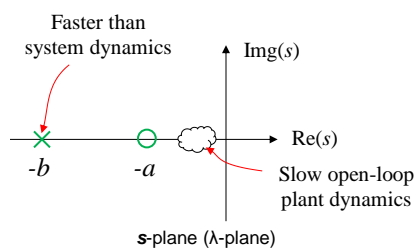
$$D(s) = \frac{s + a}{s + b}$$

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.



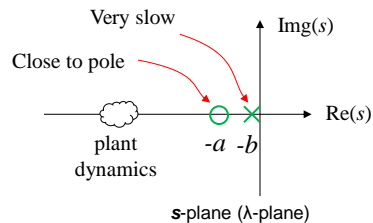
## Lead compensation: $a < b$



- Acts to decrease rise-time and overshoot
  - Zero draws poles to the left; adds phase-lead
  - Pole decreases noise
- Set  $a$  near desired  $\omega_n$ ; set  $b$  at  $\sim 3$  to  $20 \times a$



## 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  $\sim 3$  to  $10 \times b$



## PID – the Good Stuff

- 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 – the Good Stuff

- PID control performance is driven by three parameters:
  - $k$ : system gain
  - $\tau_i$ : integral time-constant
  - $\tau_d$ : 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  $t \rightarrow \infty, s \rightarrow 0$ )

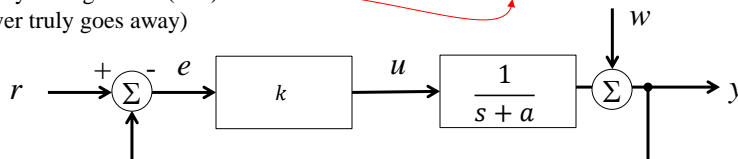
$$y = k \frac{1}{s+a} (r-y) + w$$

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

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

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

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



## Now with added integral action

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

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

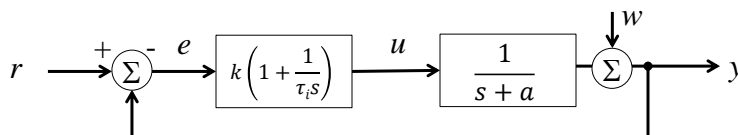
Same dynamics

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

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

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

Must go to zero for constant  $w$ !



## Derivative

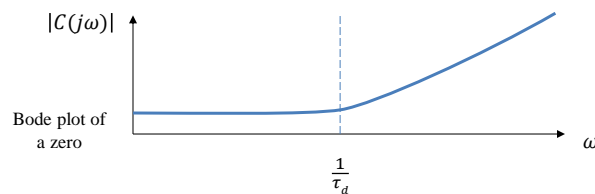
- Derivative uses the rate of change of the error signal to anticipate control action
  - Increases system damping (when done right)
  - Can be thought of as ‘leading’ the output error, applying correction predictively
  - Almost always found with P control\*

*\*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  $k \rightarrow \infty$
- 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



Ex: 2<sup>nd</sup> Order Responses!

## Direct Design: Second Order Digital Systems

Consider the z-transform of a decaying exponential signal:

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

★ sample:  $y(kT) = r^k \cos(k\theta) \mathcal{U}(kT)$  with  $r = e^{-aT}$  &  $\theta = bT$

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

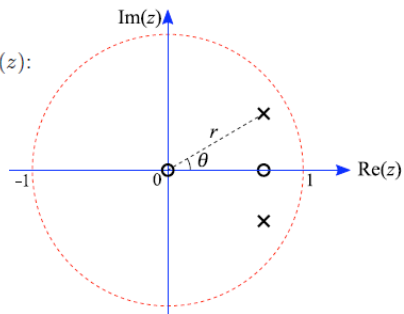
$$= \frac{z(z - r \cos \theta)}{(z - re^{j\theta})(z - re^{-j\theta})}$$

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

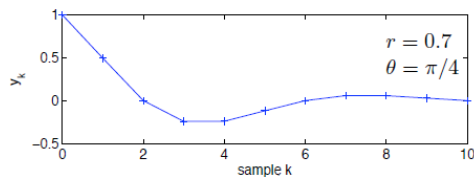
zeros:  $\begin{cases} z = 0 \\ z = r \cos \theta \end{cases}$



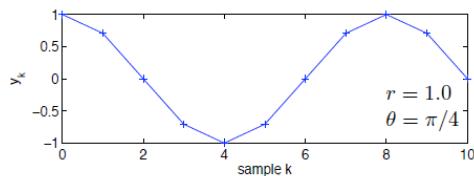
## Response of 2nd order system [1/3]

Responses for varying  $r$ :

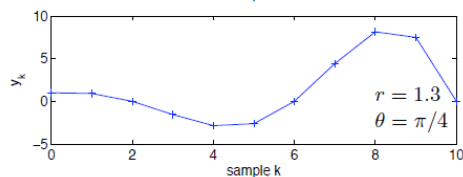
▷  $r < 1$   
↓  
exponentially decaying envelope



▷  $r = 1$   
↓  
sinusoidal response with  $2\pi/\theta$  samples per period



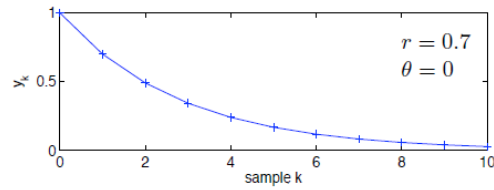
▷  $r > 1$   
↓  
exponentially increasing envelope



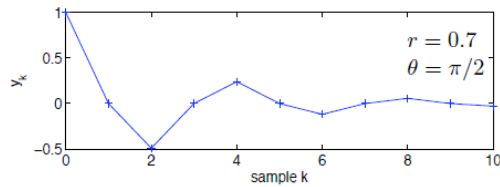
## Response of 2nd order system [2/3]

Responses for varying  $\theta$ :

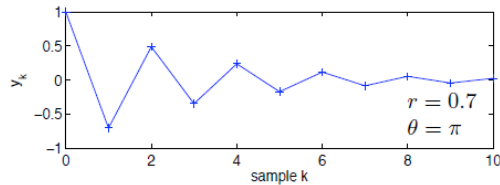
▷  $\theta = 0$   
 ↓  
 decaying exponential



▷  $\theta = \pi/2$   
 ↓  
 $2\pi/\theta = 4$  samples  
 per period

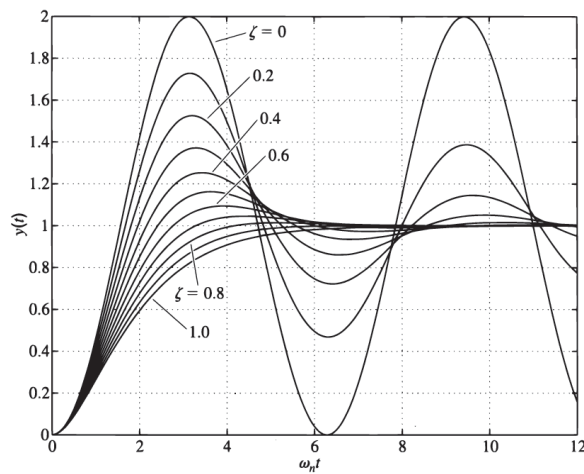


▷  $\theta = \pi$   
 ↓  
 2 samples per period



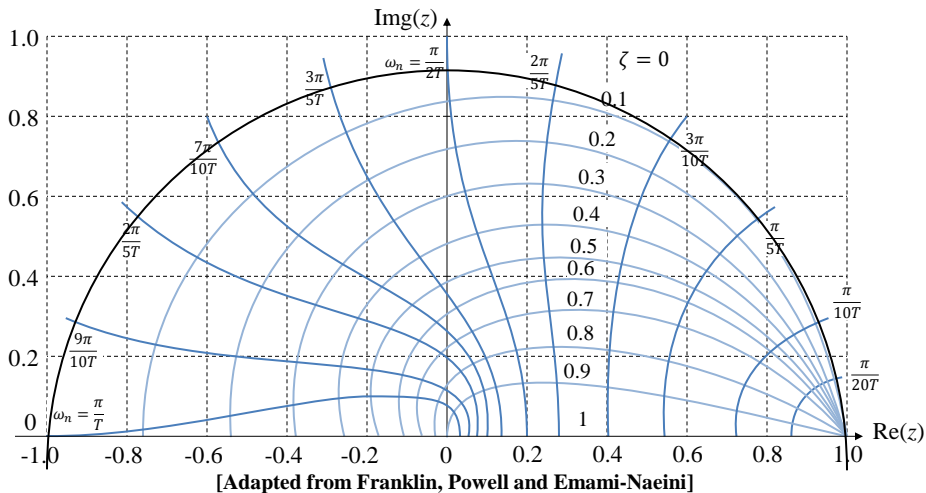
## 2<sup>nd</sup> Order System Response

- Response of a 2<sup>nd</sup> order system to increasing levels of damping:



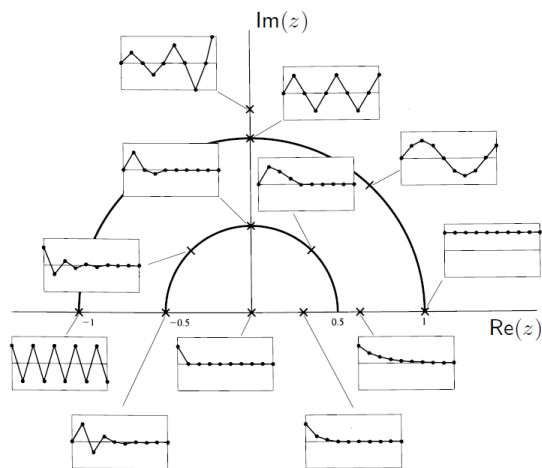
## Damping and natural frequency

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



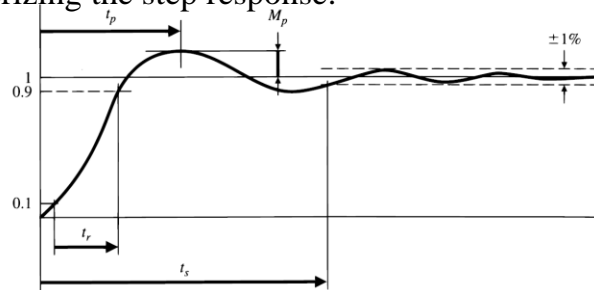
## Pole positions in the z-plane

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



## 2<sup>nd</sup> Order System Specifications

Characterizing the step response:

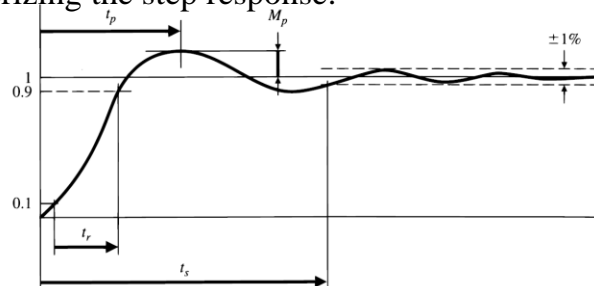


- Rise time (10%  $\rightarrow$  90%):  $t_r \approx \frac{1.8}{\omega_0}$
- Overshoot:  $M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1-\zeta^2}}$
- Settling time (to 1%):  $t_s = \frac{4.6}{\zeta\omega_0}$
- Steady state error to unit step:  $e_{ss}$
- Phase margin:  $\phi_{PM} \approx 100\zeta$



## 2<sup>nd</sup> Order System Specifications

Characterizing the step response:



- Rise time (10%  $\rightarrow$  90%) & Overshoot:  
 $t_r, M_p \rightarrow \zeta, \omega_0$  : Locations of dominant poles
- Settling time (to 1%):  
 $t_s \rightarrow$  radius of poles:  $|z| < 0.01^{1/T}$
- Steady state error to unit step:  
 $e_{ss} \rightarrow$  final value theorem  $e_{ss} = \lim_{z \rightarrow 1} \{(z-1)F(z)\}$



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

Design a controller for a system with:

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

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

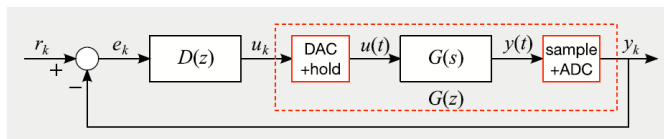
The closed loop system is required to have:

- $M_p < 16\%$
- $t_s < 10$  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:

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

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

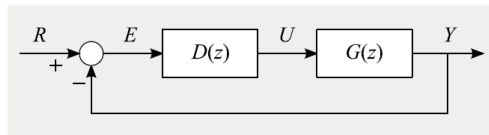
$$\frac{U(z)}{E(z)} = D(z) = 13 \frac{(1-0.88z^{-1})}{(1+0.5z^{-1})} = 13 \frac{(z-0.88)}{(z+0.5)}$$



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

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

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

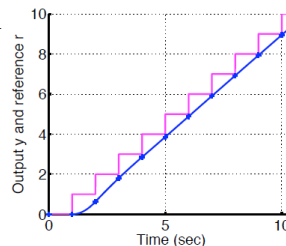


$$\frac{E(z)}{R(z)} = \frac{1}{1 + D(z)G(z)}$$

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

$$\begin{aligned} \text{so } e_{ss} &= \lim_{z \rightarrow 1} \left\{ (z-1) \frac{Tz}{(z-1)^2} \frac{1}{1 + D(z)G(z)} \right\} = \lim_{z \rightarrow 1} \frac{T}{(z-1)D(z)G(z)} \\ &= \lim_{z \rightarrow 1} \frac{T}{(z-1) \frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)} D(1)} \\ &= \frac{1-0.9048}{0.0484(1+0.9672)D(1)} = 0.96 \end{aligned}$$

$$\Rightarrow e_{ss} < 1 \quad (\text{as required})$$



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

3. Step response: overshoot  $M_p < 16\% \Rightarrow \zeta > 0.5$   
 settling time  $t_s < 10 \Rightarrow |z| < 0.01^{1/10} = 0.63$

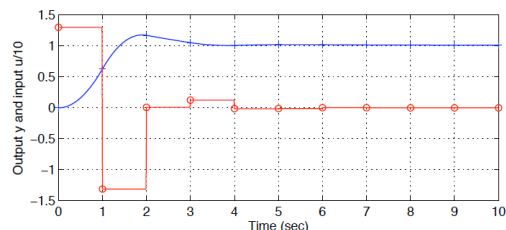
The closed loop poles are the roots of  $1 + D(z)G(z) = 0$ , i.e.

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

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

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

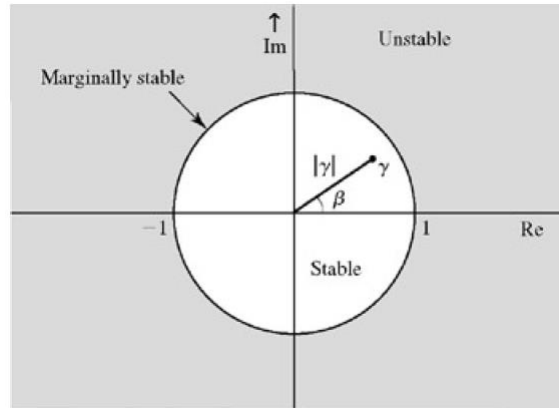
$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \Rightarrow \begin{cases} r = 0.31, \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$



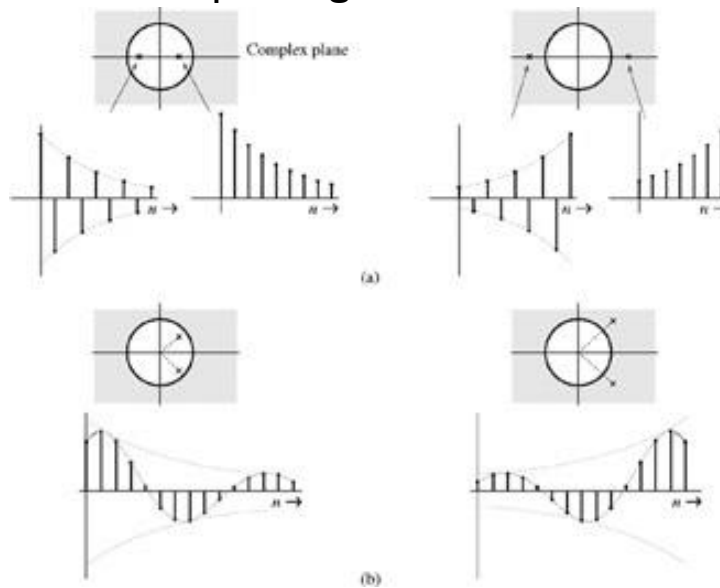
all specs satisfied!



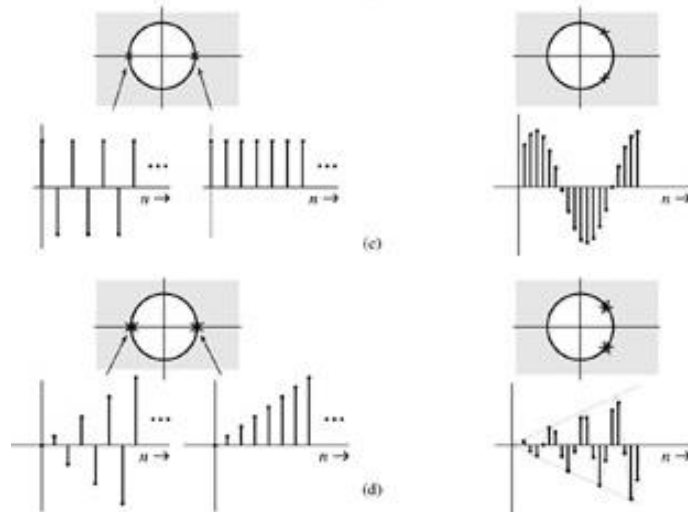
## LTID Stability



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



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



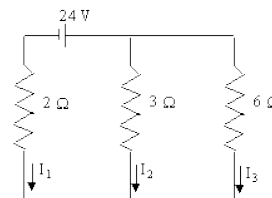
## Break!: Fun Application: Linear Algebra & KVL!

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>



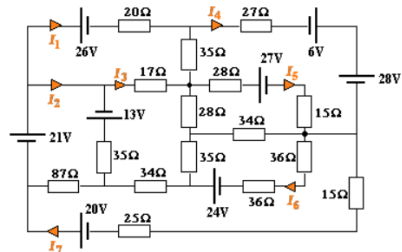
## Break!: Fun Application: Linear Algebra & KCL!

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](http://www.wolframalpha.com) is a free alternative.

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

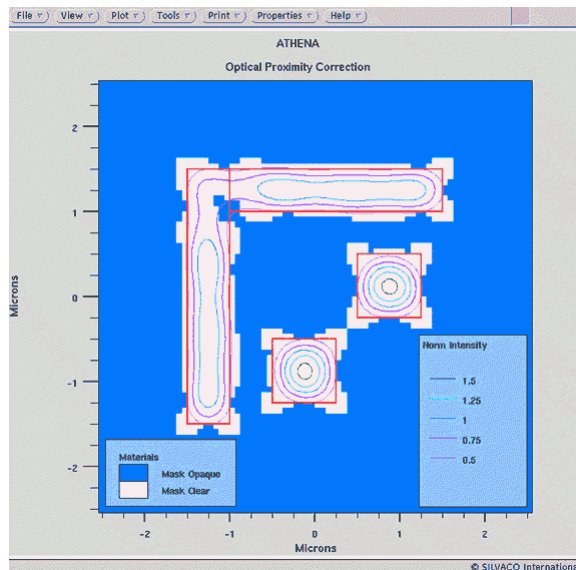
$$= \begin{bmatrix} -0.46801 \\ 0.42932 \\ 5.193 \times 10^{-3} \\ -0.22243 \\ -0.27848 \\ 0.21115 \\ 0.20914 \end{bmatrix}$$



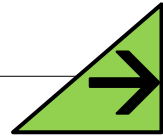
Source: <http://www.intmath.com/matrices-determinants/6-matrices-linear-equations.php>



## Break!: Fun Application: Optical Proximity Correction



## Next Time...



- **Digital Feedback Control**
- Review:
  - Chapter 2 of FPW
- More Pondering??

