



<http://elec3004.com>

Introduction to Digital Control

ELEC 3004: Systems: Signals & Controls

Dr. Surya Singh

Lecture 18

(with material from FPW + Åström and Hägglund)

elec3004@itee.uq.edu.au

May 12, 2016

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

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

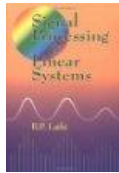


Lecture Schedule:

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
	21-Mar	Discrete System Analysis
4	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
	9-May	Servoregulation/PID
10	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

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

Next Time

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



Final Exam Information announcement

- Date:
Saturday, June/18
(remember buses on Saturday Schedule)

- Time:
9:30 am

- Location:
TBA
(Maybe Heath Room?)



- UQ Exams are now “ID Verified”
➔ Please remember your ID! ←



PID in Perspective

- Deceivingly effective
 - ∴ a little control goes a long way
- Lots of legacy
 - Shifts happen in abruptly
- Ex: Temperature Control
 - On|Off: Lots of oscillations
 - High gains + Inertia
 - Derivative action useful **here**
- PID: “Low level”
 - Computer as “high-level” setting set-point

Case Studies :

- DVD drive: 3 PIDs
 - Rotation speed
 - Focus
 - Track following
- Canadian Paper mill PID Audit
 - 2000 Loops
 - 97%: PI
 - 20%: Worked well ($\downarrow\sigma$)
 - 30%: Bad Tuning
 - 30%: Broken kit
 - 20%: non anti-aliasing, poor sampling rates, etc.
- Old-timers unwilling to learn new technology
 - Lower γ [Learning Rate]
- $\nabla+$: Not change for change sake
 - Auto-tuning
 - Distributed operation
 - Fast sampling and data-processing



PID Algorithm (in various domains):

- 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
```



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)



Tuning

Ziegler-Nichols

- (Only) moderately good tuning only in restricted situations.

Step Response Method

Controller	aK	T_i/L	T_d/L	T_p/L
P	1			4
PI	0.9	3		5.7
PID	1.2	2	$L/2$	3.4

Frequency Response Method

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

Chien, Hrones, and Reswick

$$P(s) = \frac{K_p}{1 + sT} e^{-sL}$$

Load disturbance response method

Controller	No overshoot			20% overshoot		
	aK	T_i/L	T_d/L	aK	T_i/L	T_d/L
P	0.3			0.7		
PI	0.6	4.0		0.7	2.3	
PID	0.95	2.4	0.42	1.2	2.0	0.42

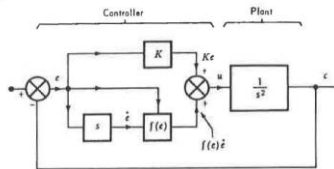
set-point response method

Controller	No overshoot			20% overshoot		
	aK	T_i/L	T_d/L	aK	T_i/L	T_d/L
P	0.3			0.7		
PI	0.35	1.2		0.6	1.0	
PID	0.6	1.0	0.5	0.95	1.4	0.47



Poles are Eigenvalues: Some Implications

Stability of a 2nd 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:

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


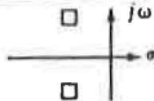

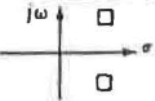

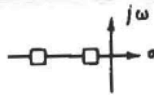

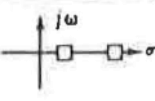

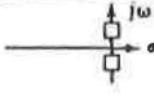

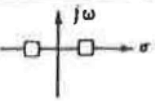
with the eigenvalues

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

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



Various Types of Singularities (2nd order systems)

<i>Stable</i>		<i>Unstable</i>	
<i>Trajectory type</i>	<i>Eigenvalues</i>	<i>Trajectory type</i>	<i>Eigenvalues</i>
 <p>Stable focus</p>		 <p>Unstable focus</p>	
 <p>Stable node</p>		 <p>Unstable node</p>	
 <p>Vortex</p>		 <p>Saddle</p>	



How to Design?
Back to Analog !
(Design By Emulation)

Two cases for control design

The system...

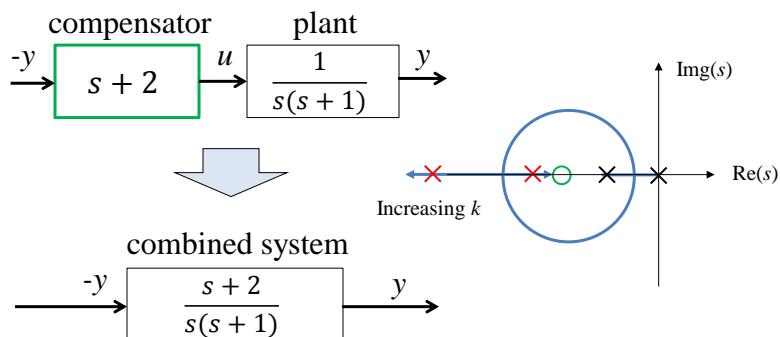
- Isn't fast enough
- Isn't damped enough
- Overshoots too much
- Requires too much control action
(“Performance”)

- Attempts to spontaneously disassemble itself
(“Stability”)



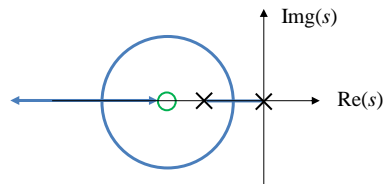
Dynamic compensation

- 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?

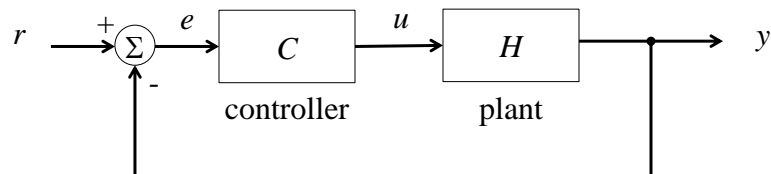
- Recognise the following:
 - A root locus starts at poles, terminates at zeros
 - “Holes eat poles”
 - Closely matched pole and zero dynamics cancel
 - The locus is on the real axis to the left of an odd number of poles (treat zeros as ‘negative’ poles)



The Root Locus (Quickly)

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

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



The Root Locus (Quickly)

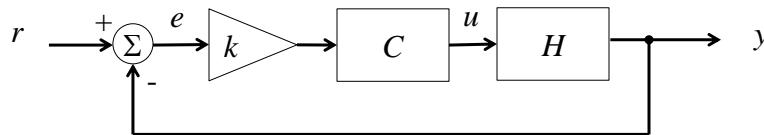
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

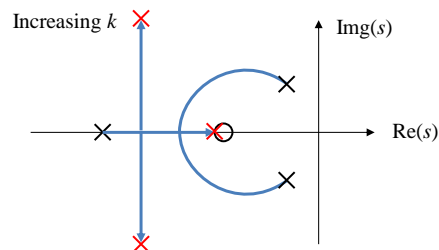
$$\frac{y}{r} = \frac{kC_n H_n}{C_d H_d + kC_n H_n}$$

characteristic polynomial



The Root Locus (Quickly)

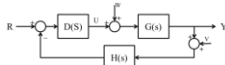
- Pole positions change with increasing gain
 - The trajectory of poles on the pole-zero plot with changing k is called the “root locus”
 - This is sometimes quite complex



(In practice you'd plot these with computers)

Root Locus Design: “Evan’s Method”

- Imagine “the basic feedback system”:



$$TF(s) = \frac{Y(s)}{R(s)} = \frac{D(s)G(s)}{1 + D(s)G(s)H(s)} = \frac{DG}{1 + DGH}$$

$$\rightarrow \text{Characteristic Equation: } 1 + DGH = 0$$

- Put the characteristic equation in Root Locus form:

$$1 + K \cdot L(s) = 0$$

- **If** we define $L(s) = \frac{b(s)}{a(s)}$

- **then** $a(s) + K \cdot b(s) = 0$ and $L(s) = -\frac{1}{K}$

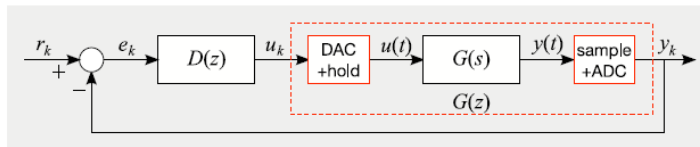
- Thus a Root Locus is a graph of all possible roots of $1 + K \cdot L(s) = 0$ and K as the variable parameter

- ∴ This is the solution to the roots of the closed-loop system characteristic equation and thus the closed-loop poles of the system. The root-locus graph may be viewed as a method for interring the **dynamic properties of a system as K changes**.

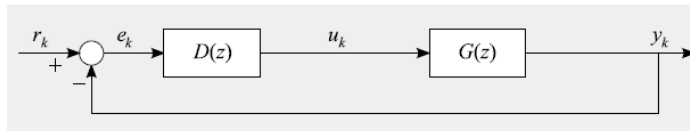


Designing in the Purely Discrete...

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 → 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?



Emulation vs Discrete Design

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

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

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

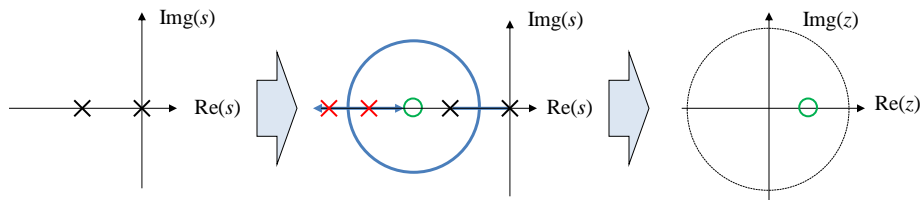
Root loci behave the same on both planes!

- 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

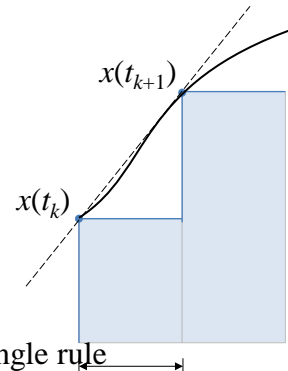


Euler's method*

- Dynamic systems can be approximated[†] 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



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.

- Rewrite the function as a dynamic system:

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

Apply inverse Laplace transform:

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

- 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:

$$\begin{aligned}y(k+1) - y(k) + Ty(k) &= x(k+1) - x(k) + 2Tx(k) \\y(k+1) + (T-1)y(k) &= x(k+1) + (2T-1)x(k)\end{aligned}$$

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

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

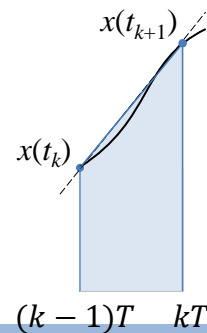
- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:

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

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



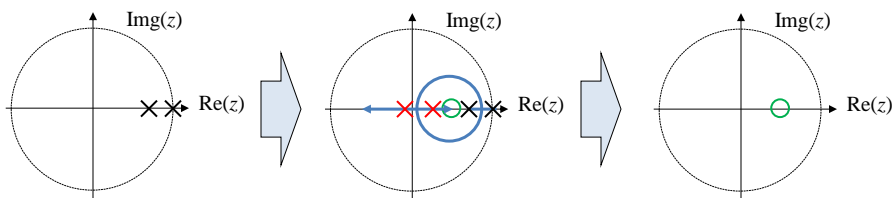
Modified matched pole-zero

- 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× 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



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 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 $\mathbf{x}(t)$ [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 :

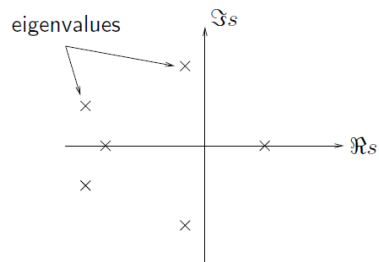
- eigenvalues give exponents that can occur in exponentials
- real eigenvalue λ corresponds to an exponentially decaying or growing term $e^{\lambda t}$ in solution
- complex eigenvalue $\lambda = \sigma + j\omega$ corresponds to decaying or growing sinusoidal term $e^{\sigma t} \cos(\omega t + \phi)$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

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



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



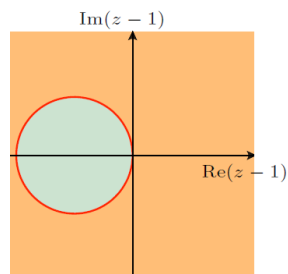
Stability

Fast sampling revisited

- For small T:

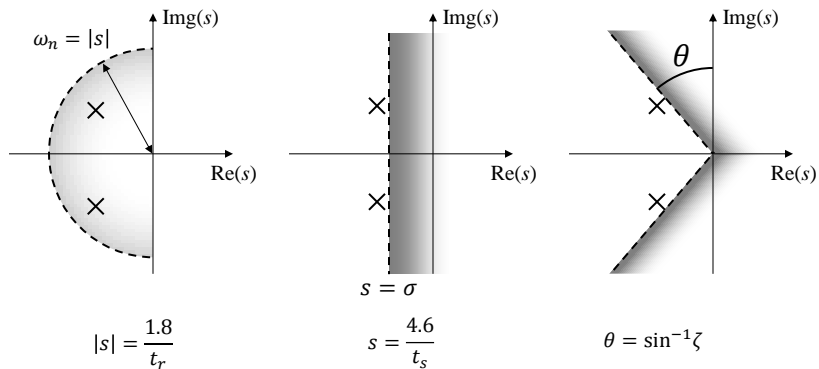
$$z = e^{sT} = 1 + sT + \frac{(sT)^2}{2} + \dots \approx 1 + sT$$
$$\rightarrow z \approx 1 + sT \rightarrow s = \frac{z - 1}{T}$$

- Hence, the unit circle under the map from z to s-plane becomes:



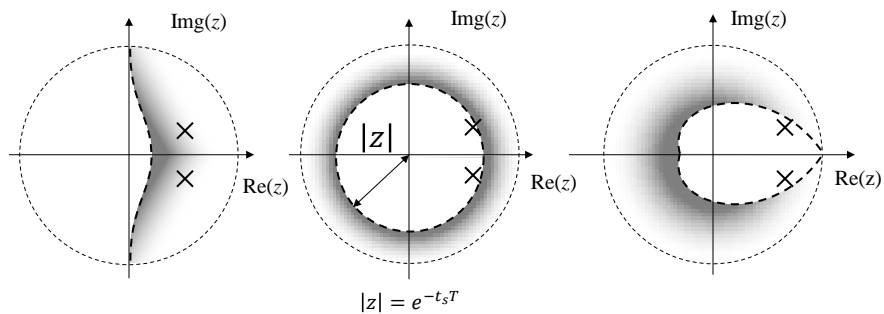
Specification bounds

- Recall in the continuous domain, response performance metrics map to the s-plane:



Discrete bounds

- These map to the discrete domain:



In practice, you'd use Matlab to plot these, and check that the spec is satisfied



Example Code:

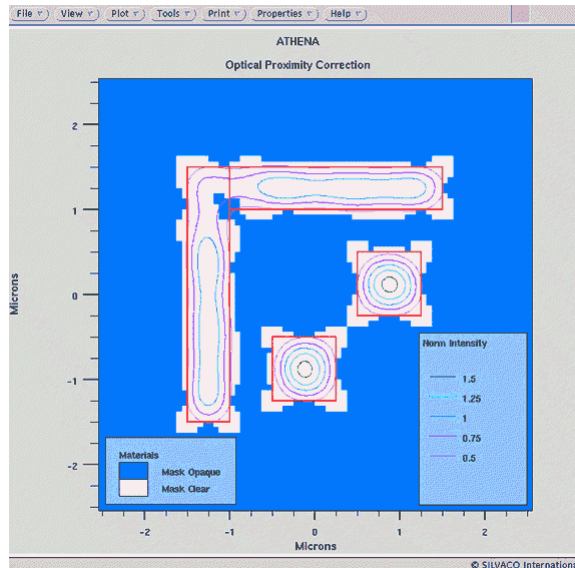
```
%% Input System Model G
numg=5; deng=[1 20 0]; sysg=tf(numg, deng);

%% Approximate the ZOH (1-e^-sT)/(s)
[nd, dd]=pade(1,2); %pade gives us the "hold" or -e^-sT of a ZOH
syp=tf(nd, dd); sysi=tf([1],[1,0]); %Now we need the "1/s" portion
sysl=series(1-syp, sysi); % Approximation as a series

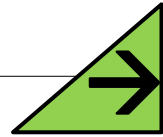
%% Open loop response
syso=series(sysl, sysg); % computer the open loop G with the ZOH
sys=feedback(syso,1); % Computer the unity feedback response
step(sys) % Display the step response
```



Break!: Fun Application: Optical Proximity Correction



Next Time...



- **PID!**
- Review:
 - PID notes online
 - Chapter 5 of FPW
- Deep Pondering??

