http://elec3004.com

Intro to Digital Controls

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

Dr. Paul Pounds

Lecture 7

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

April 15, 2014

Lecture Schedule:

1	4-Mar	Introduction & Systems Overview
	6-Mar	[Linear Dynamical Systems]
2	11-Mar	Signals as Vectors & Systems as Maps
	13-Mar	[Signals]
3	18-Mar	Sampling & Data Acquisition & Antialiasing Filters
	20-Mar	[Sampling]
4	25-Mar	System Analysis & Convolution
	27-Mar	[Convolution & FT]
5	1-Apr	Frequency Response & Filter Analysis
	3-Apr	[Filters]
6	8-Apr	Discrete Systems & Z-Transforms
	10-Apr	[Z-Transforms]
7	15-Apr	Introduction to Control
	17-Apr	[Feedback]
8	29-Apr	Digital Filters
	1-May	[Digital Filters]
9	6-May	Introduction to Digital Control
	8-May	[Digitial Control]
10	13-May	Stability of Digital Systems
	15-May	[Stability]
11	20-May	State-Space
	22-May	Controllability & Observability
12	27-May	PID Control & System Identification
	29-May	Digitial Control System Hardware
13	3-Jun	Applications in Industry & Information Theory & Communications
	5-Jun	Summary and Course Review

z Transforms

(Digital Systems Made eZ)

The *z*-transform

- The discrete equivalent is the *z*-Transform†:

$$\mathcal{Z}\{f(k)\} = \sum_{k=0}^{\infty} f(k) z^{-k} = F(z)$$

and

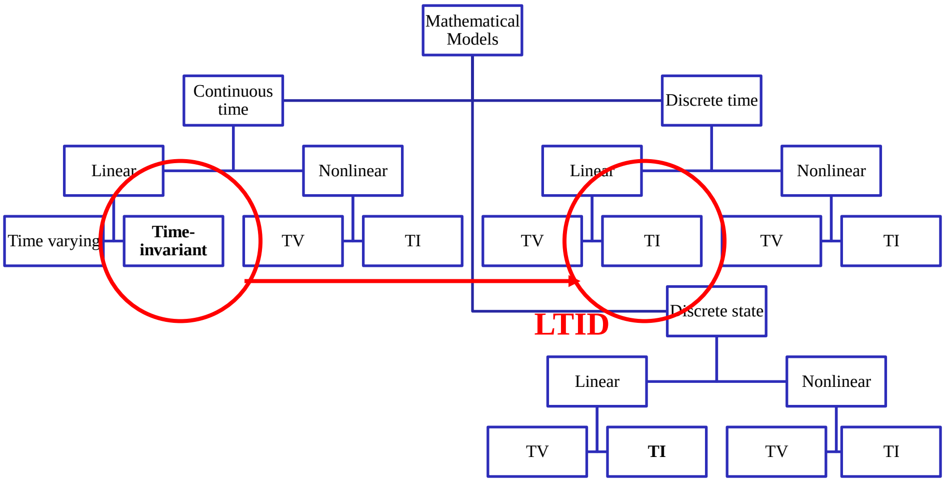
$$\mathcal{Z}\{f(k-1)\} = z^{-1} F(z)$$

$$x(k) \longrightarrow \boxed{F(z)} \longrightarrow y(k)$$

Convenient!

†This is not an approximation, but approximations are easier to derive

What about the Discrete Domain? [Lecture 4-Slide 10]



The z-Transform

- It is defined by:

$$z = re^{j\omega}$$

Or in the Laplace domain:

$$z = e^{sT}$$

- Thus: $Y(z) = \sum_{k=-\infty}^{\infty} h[k] z^{-k}$ or $y[n] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z)$

- I.E., It's a discrete version of the Laplace:

$$f(kT) = e^{-akT} \Rightarrow \mathcal{Z}\{f(k)\} = \frac{z}{z - e^{-aT}}$$

Introduction to Digital Control

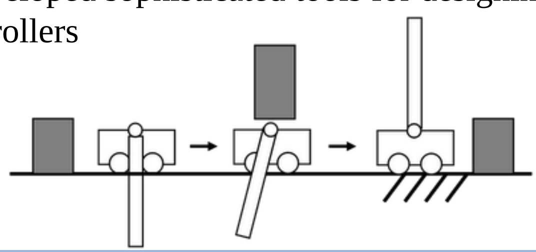
By Paul Pounds
(Mostly ☺)

Digital control

Once upon a time…

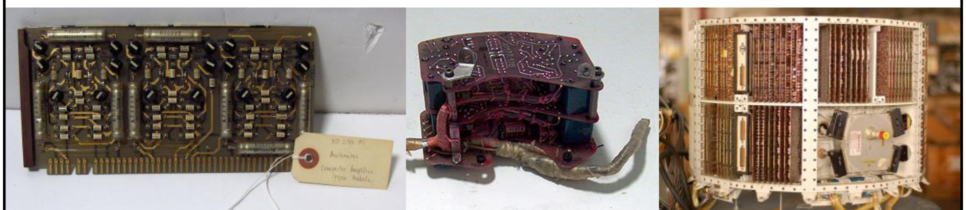
- Electromechanical systems were controlled by electromechanical compensators
  - Mechanical flywheel governors, capacitors, inductors, resistors, relays, valves, solenoids (fun!)
  - But also complex and sensitive!

- Humans developed sophisticated tools for designing reliable analog controllers



Computer revolution

- In the 1950s and 60s very smart people developed computerised controllers
- Digital processor implements the control algorithm numerically, rather than in discrete hardware



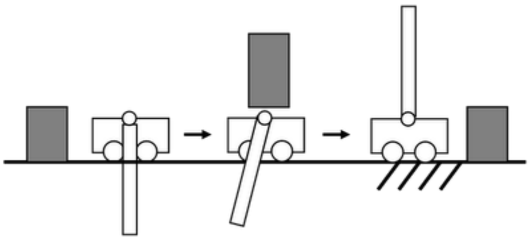
Minuteman ICBM guidance computer components [CHM]

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**

Ok, so how do we do this?

We already know about control, right?



What you already know*

- Signals can be represented by transfer functions in the s-domain
- Roots of a transfer function's denominator (poles) indicate the stability of the system
- Poles move around under feedback control
  - Feedback can stabilise an unstable system

*If you have no idea what I'm talking about, now is the time to mention it.

The good news

Digital control is just like that!

Thanks for coming, see you at the exam!

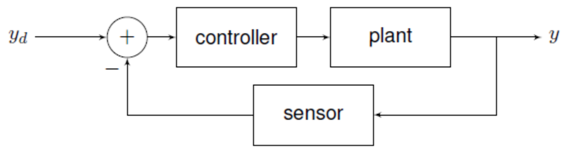
Not so fast

- While there are discrete analogues for every part of continuous systems theory, there are unique and important differences you must be familiar with

> Virtually every control system you will ever use will be a computerised digital controller

Feedback Control

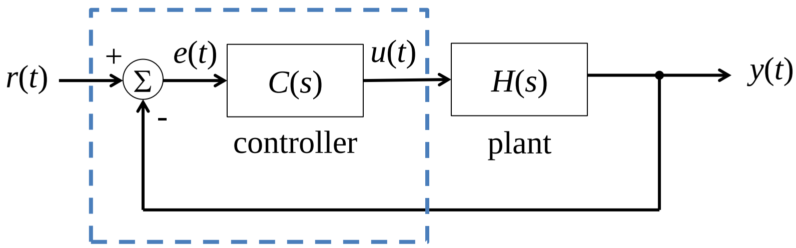
(Simple) control systems have three parts:



- 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.

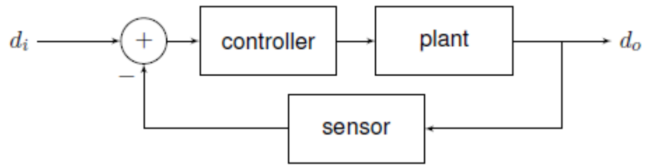
Archetypical control system

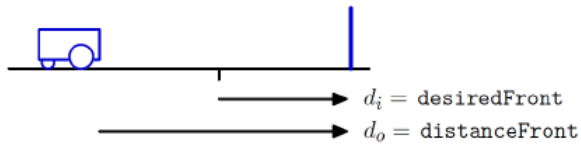
- Consider a continuous control system:



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

Simple Controller Goes Digital





plant: $y[n] = y[n-1] - Tu[n-1]$

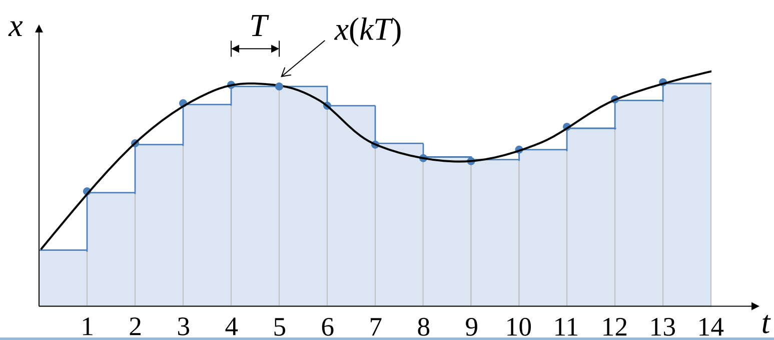
sensor: $y[n] = u[n-1]$

controller: $y[n] = Ku[n]$

Complex system behaviors, depending on K

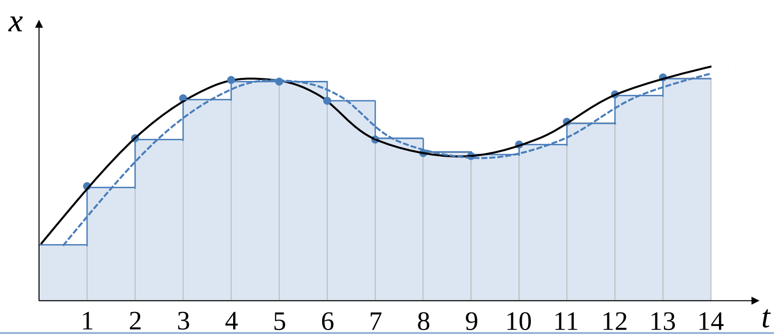
Return to the discrete domain

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



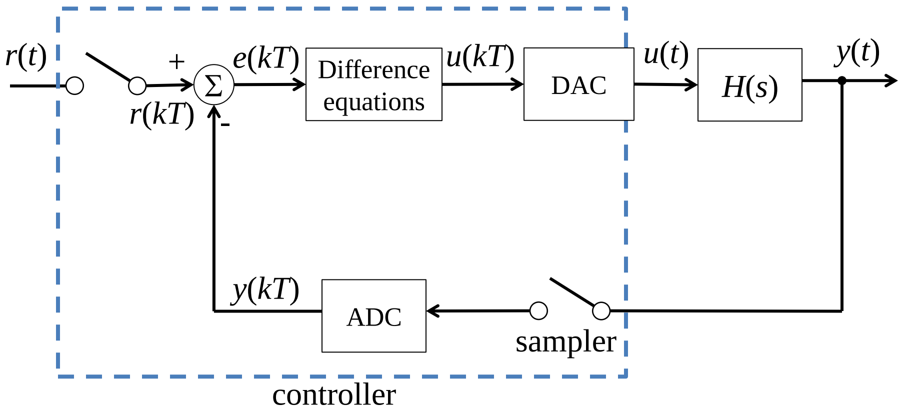
Zero Order Hold

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



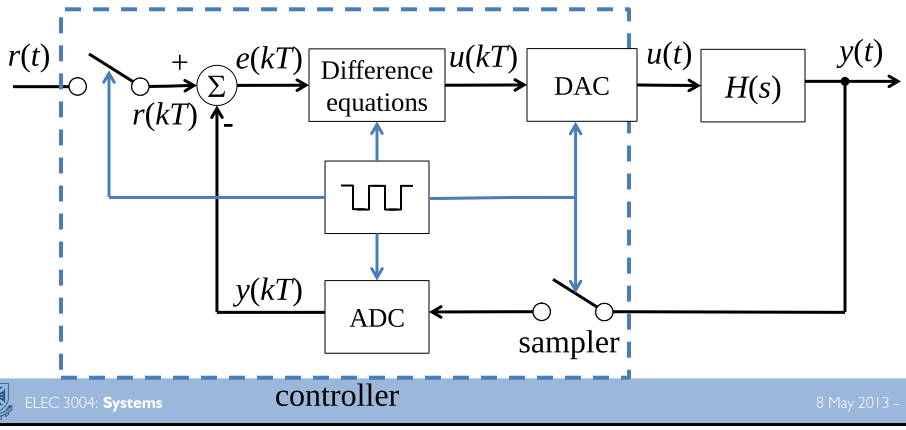
Digitisation

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



Digitisation

- Continuous signals sampled with period T
- kth 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:

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

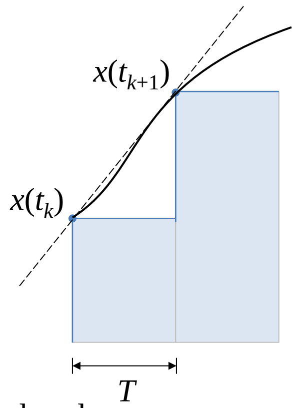
  – We can think of x and y as parameterised in k

Useful shorthand: $x(t_{k+i}) \equiv x(k+i)$

Euler's method*

- Dynamic systems can be approximated† by recognising that:

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



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

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

An example!

Convert the system $\frac{\mathrm{Y(s)}}{X(s)} = \frac{s+2}{s+1}$ 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

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

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

(this requires very advanced technology!)

future value current values

- 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 maths bit)

Great!

We already know how to design compensators, and now we can implement them in a computer.

That means we're done, right?

Not quite

There are unanswered questions:

- How do we analyse more elaborate systems of these difference equation things?
- What happens as you change *T*?
  - How would you even choose the right *T*?
- What about Nyquist? Or noise?
- How good are those approximations anyway?

Coping with complexity

- Transfer functions help control complexity
  - Recall the Laplace transform:

$$\mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt = F(s)$$

where

$$\mathcal{L}\{\dot{f}(t)\} = sF(s)$$

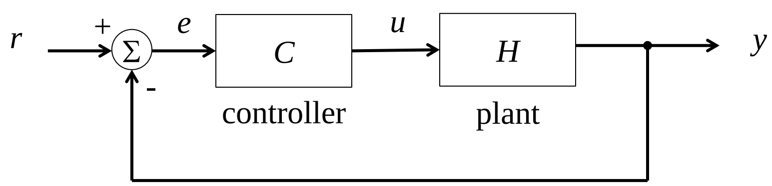
$$x(t) \longrightarrow \boxed{H(s)} \longrightarrow y(t)$$

Is there a something similar for sampled systems?

Quick Background:
Pole-Zero Diagrams & The Root Locus

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

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



Quick Background:
Pole-Zero Diagrams & The Root Locus

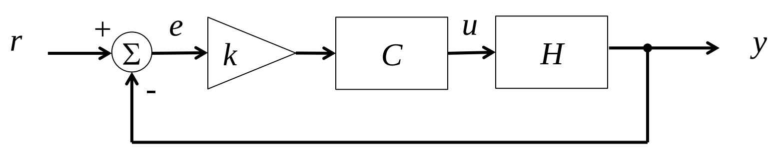
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

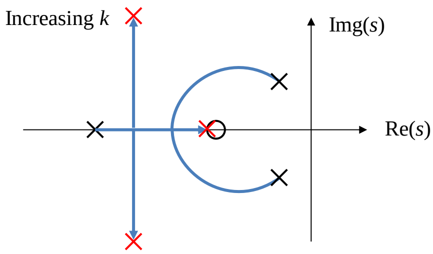
$$\frac{y}{r} = \frac{kC_nH_n}{C_dH_d + kCnHn}$$

characteristic polynomial



Quick Background:
Pole-Zero Diagrams & The Root Locus

- Pole positions change with increasing gain
  - The trajectory of **closed-loop 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)

The z-transform

- The discrete equivalent is the z-Transform†:

$$\mathcal{Z}\{f(k)\} = \sum_{k=0}^{\infty} f(k) z^{-k} = F(z)$$

and

$$\mathcal{Z}\{f(k-1)\} = z^{-1} F(z)$$

$$x(k) \longrightarrow \boxed{F(z)} \longrightarrow y(k)$$

Convenient!

†This is not an approximation, but approximations are easier to derive

The *z*-transform

- Some useful properties
  - **Delay by n samples**: $\mathcal{Z}\{f(k-n)\} = z^{-n}F(z)$
  - **Linear**: $\mathcal{Z}\{af(k) + bg(k)\} = aF(z) + bG(z)$
  - **Convolution**: $\mathcal{Z}\{f(k) * g(k)\} = F(z)G(z)$

So, all those block diagram manipulation tools you know and love will work just the same!

The *z*-transform

- In practice, you'll use look-up tables or computer tools (ie. Matlab) to find the *z*-transform of your functions

$F(s)$	$F(kt)$	$F(z)$
$\frac{1}{s}$	1	$\frac{z}{z-1}$
$\frac{1}{s^2}$	kT	$\frac{Tz}{(z-1)^2}$
$\frac{1}{s+a}$	e^{-akT}	$\frac{z}{z-e^{-aT}}$
$\frac{1}{(s+a)^2}$	kTe^{-akT}	$\frac{zTe^{-aT}}{(z-e^{-aT})^2}$
$\frac{1}{s^2+a^2}$	$\sin(akT)$	$\frac{z\sin aT}{z^2-(2\cos aT)z+1}$

Why z-Transform

- Makes it easy to analyse feedback systems governed by difference equations
- For any complex number $z = re^{j\omega}$ $y[n] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z)$
- Forward Analysis: $Y(z) = \sum_{k=-\infty}^{\infty} h[k] z^{-k}$
- Backward Synthesis
  (for any fixed r > 0 on which the Z-transform converges):

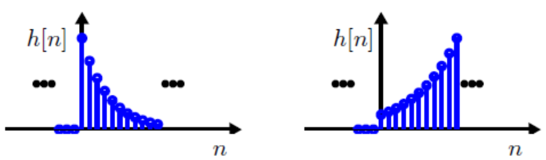
$$y[n] = \frac{1}{2\pi} \int_{2\pi} Y(re^{j\omega})(re^{j\omega})^n d\omega$$

z-Transforms for Difference Equations

- First-order linear constant coefficient difference equation:

First-order linear constant coefficient difference equation:

$$y[n] = ay[n-1] + bu[n]$$



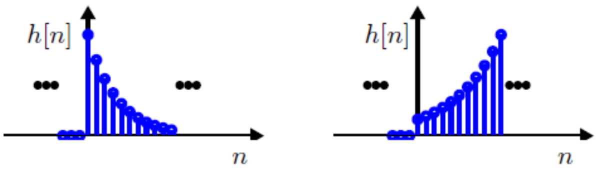
$$h[n] = \begin{cases} ba^n & n \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

$$H(z) = \sum_{k=0}^{\infty} ba^k z^{-k} = b \sum_{k=0}^{\infty} \left(\frac{a}{z}\right)^k = \frac{b}{1 - az^{-1}}, \quad \text{when } |z| > |a|.$$

z-Transforms for Difference Equations

First-order linear constant coefficient difference equation:

$$y[n] = ay[n-1] + bu[n]$$

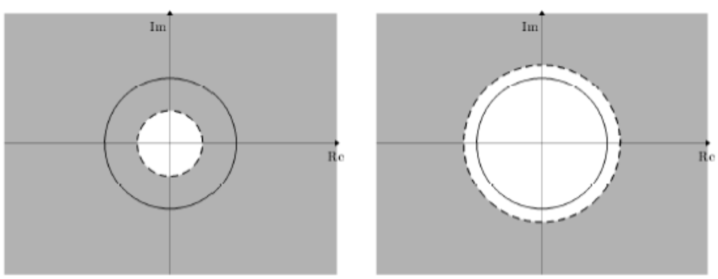


$$y[n] - ay[n-1] = bu[n]$$
$$\updownarrow$$
$$Y(z) - az^{-1}Y(z) = bU(z)$$

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b}{1-az^{-1}},$$ when does it converge?

Region of Convergence (ROC) Plots

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b}{1-az^{-1}}, \quad |z| > |a|$$



$a = .5$ $\qquad$ $a = 1.2$

Properties of the ROC

- ➔The ROC is always defined by circles centered around the origin.

  $h[k]r^{-k}$ is absolutely summable, where $r = |z|$.

- ➔Right-sided signals have "outsided" ROCs.

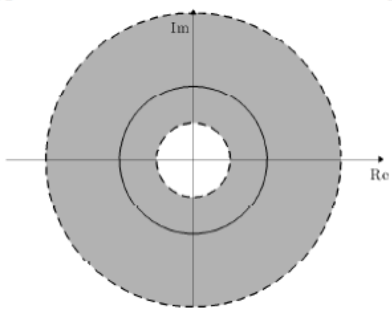
  if $\exists n_0$ such that $h[n] = 0\ \forall n < n_0$, then if $r_0 \in$ ROC, then $\forall r$ with $r_0 < r < \infty$ are also in the ROC.

- ➔Left-sided signals have "insided" ROCs. (with ∀r within $0<r<r_0$)

Combinations of Signals

$$y_1[n] = \begin{cases} ba^n & n \geq 0 \\ 0 & n < 0 \end{cases} \qquad y_2[n] = \begin{cases} 0 & n \geq 0 \\ -ba^n & n < 0 \end{cases}$$

$a = .5$ $\qquad a = 2$



ROC for $\alpha_1 y_1[n] + \alpha_2 y_2[n]$

Higher-order difference equations

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + a_3 y[n-3] + b_0 u[n] + b_1 u[n-1] + \dots$$

Easy to take the Z-transform

$$Y(z) = a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) + a_3 z^{-3} Y(z) + b_0 U(z) + \dots$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-1} + \dots}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} + \dots}$$

Final value theorem

- An important question: what is the steady-state output a stable system at $t = \infty$?
  - For continuous systems, this is found by:
    $$\lim_{t \to \infty} x(t) = \lim_{s \to 0} sX(s)$$
  - The discrete equivalent is:
    $$\lim_{k \to \infty} x(k) = \lim_{z \to 1} (1 - z^{-1}) X(z)$$

(Provided the system is stable)

An example!

- Back to our difference equation:

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

becomes

$$Y(z) = X(z) + Az^{-1}X(z) - Bz^{-1}Y(z)$$
$$(z+B)Y(z) = (z+A)X(z)$$

which yields the transfer function:

$$\frac{Y(z)}{X(z)} = \frac{z+A}{z+B}$$

Note: It is also not uncommon to see systems expressed as polynomials in z^{-n}

This looks familiar…

- Compare:

$$\frac{\mathrm{Y(s)}}{X(s)} = \frac{s+2}{s+1} \text{ vs } \frac{Y(z)}{X(z)} = \frac{z+A}{z+B}$$

How are the Laplace and *z* domain representations related?

Consider the simplest system

- Take a first-order response:

$$f(t) = e^{-at} \Rightarrow \mathcal{L}\{f(t)\} = \frac{1}{s+a}$$

- The discrete version is:

$$f(kT) = e^{-akT} \Rightarrow \mathcal{Z}\{f(k)\} = \frac{z}{z - e^{-aT}}$$

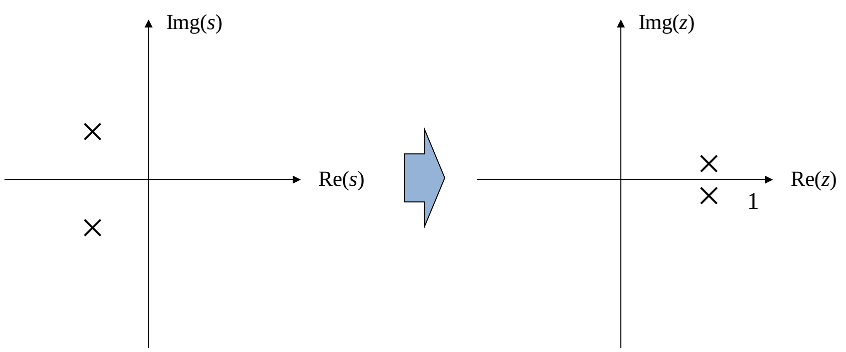
The equivalent system poles are related by

$$z = e^{sT}$$

That sounds somewhat profound… but what does it mean?

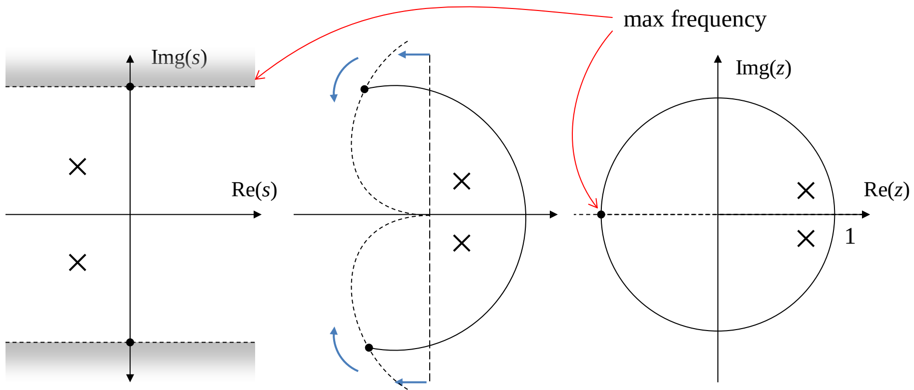
The *z*-Plane

- *z*-domain poles and zeros can be plotted just like *s*-domain poles and zeros:



Deep insight #1

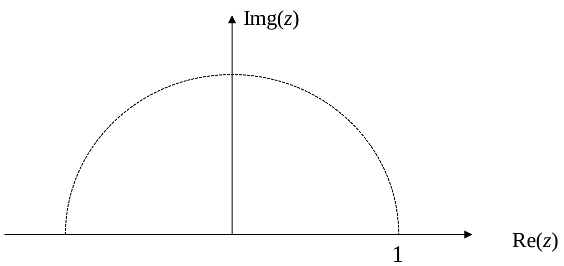
The mapping between continuous and discrete poles and zeros acts like a distortion of the plane



The z-plane

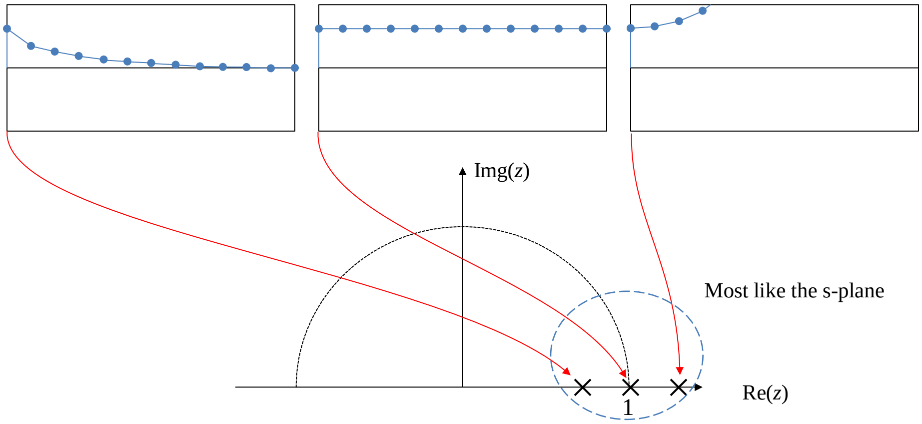
- We can understand system response by pole location in the *z*-plane

[Adapted from Franklin, Powell and Emami-Naeini]



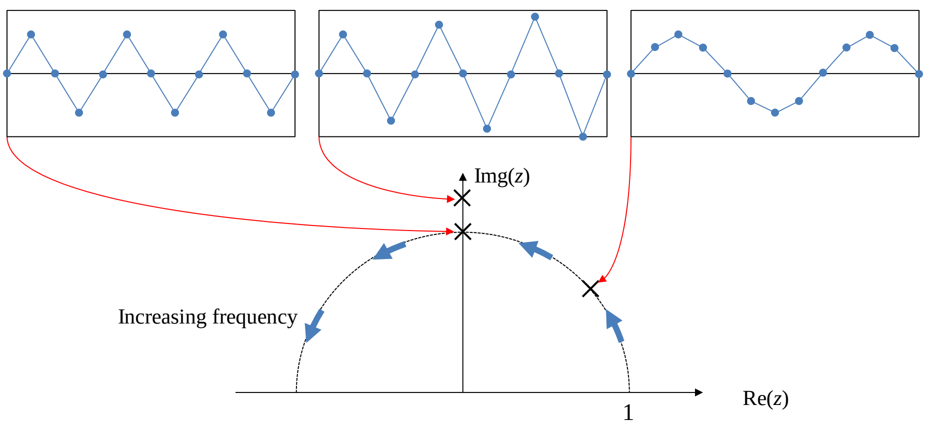
Effect of pole positions

- We can understand system response by pole location in the *z*-plane



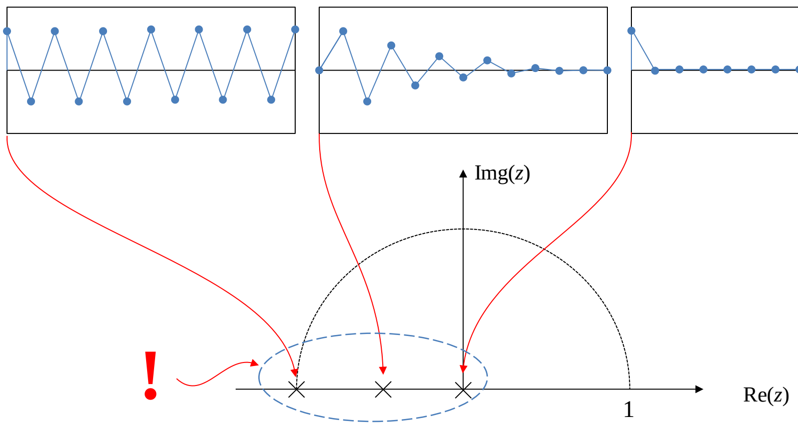
Effect of pole positions

- We can understand system response by pole location in the *z*-plane



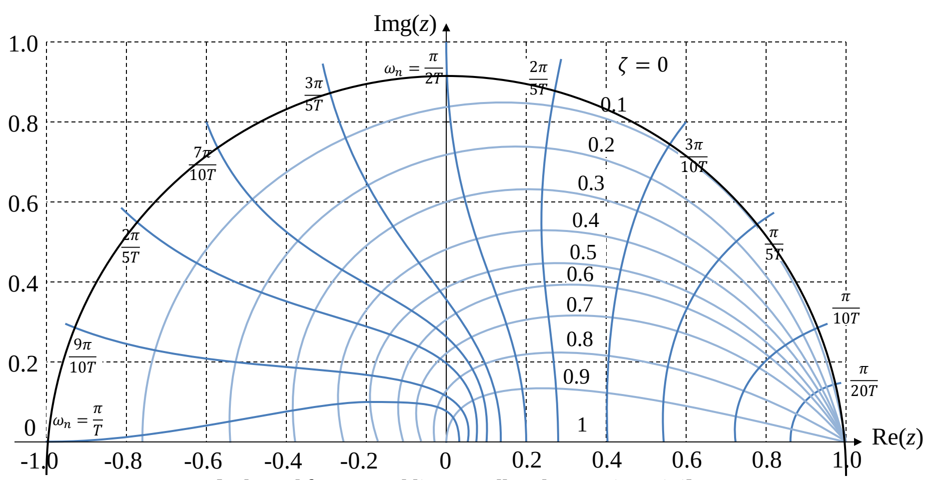
Effect of pole positions

- We can understand system response by pole location in the z-plane



Damping and natural frequency

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

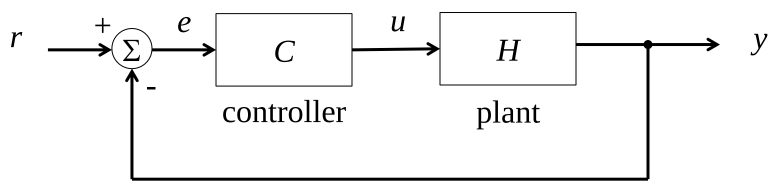


[Adapted from Franklin, Powell and Emami-Naeini]

Quick refresher: the root locus

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

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



Quick refresher: the root locus

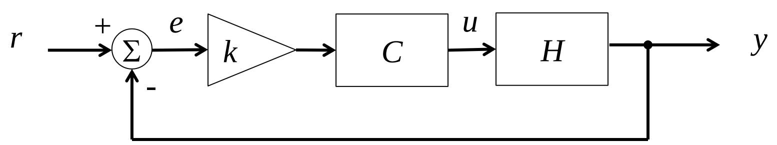
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

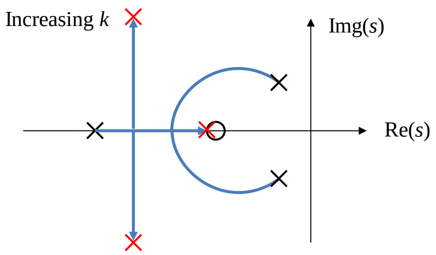
$$\frac{y}{r} = \frac{kC_nH_n}{C_dH_d + kCnHn}$$

characteristic polynomial



Quick refresher: the root locus

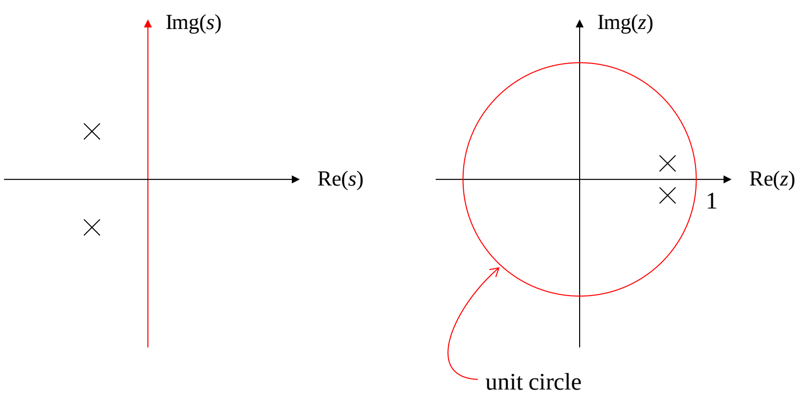
- Pole positions change with increasing gain
  - The trajectory of poles on the pole-zero plot with changing *k* is called the "root locus"
  - This is sometimes quite complex



(In practice you'd plot these with computers)

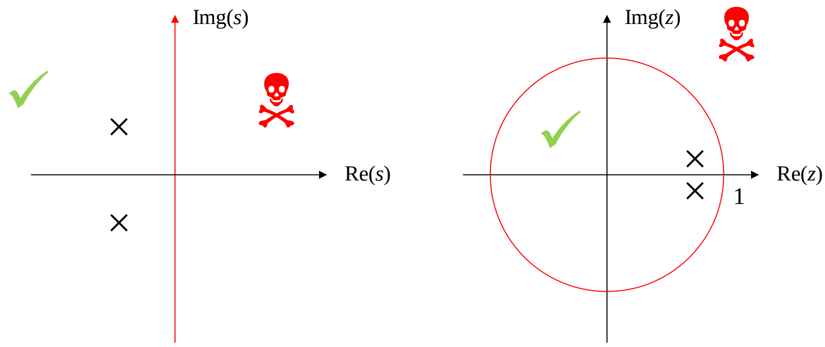
z-plane stability

- In the *z*-domain, the unit circle is the system stability bound



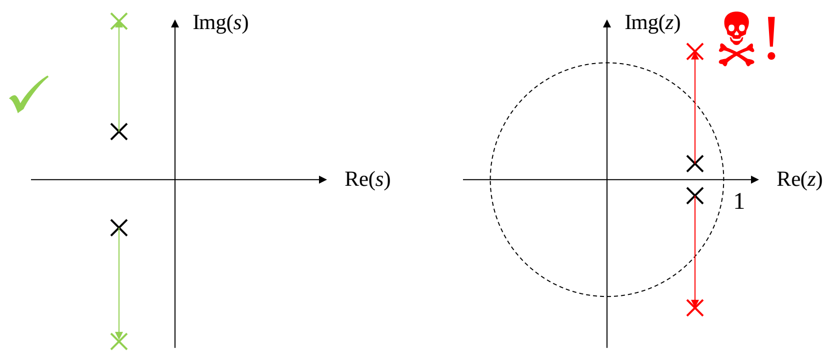
z-plane stability

- In the z-domain, the unit circle is the system stability bound



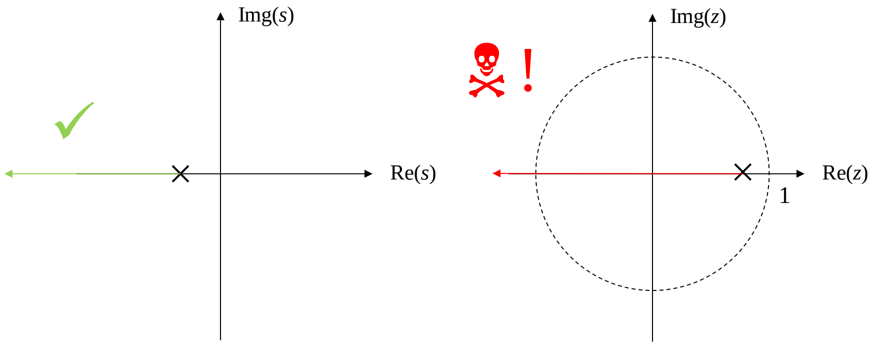
z-plane stability

- The z-plane root-locus in closed loop feedback behaves just like the s-plane:



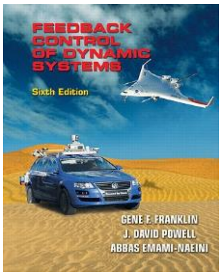
Deep insight #2

Gains that stabilise continuous systems can actually *destabilise* digital systems!



Quick plug*

- Most of this is based on Chapter 8 of "**Feedback Control of Dynamic Systems**" by Franklin, Powell and Emami-Naeini.



* No, they're not paying me – it's just a really good book!