Digital Control Super Review

or

Everything in one place

Paul Pounds

14 June 2012

University of Queensland

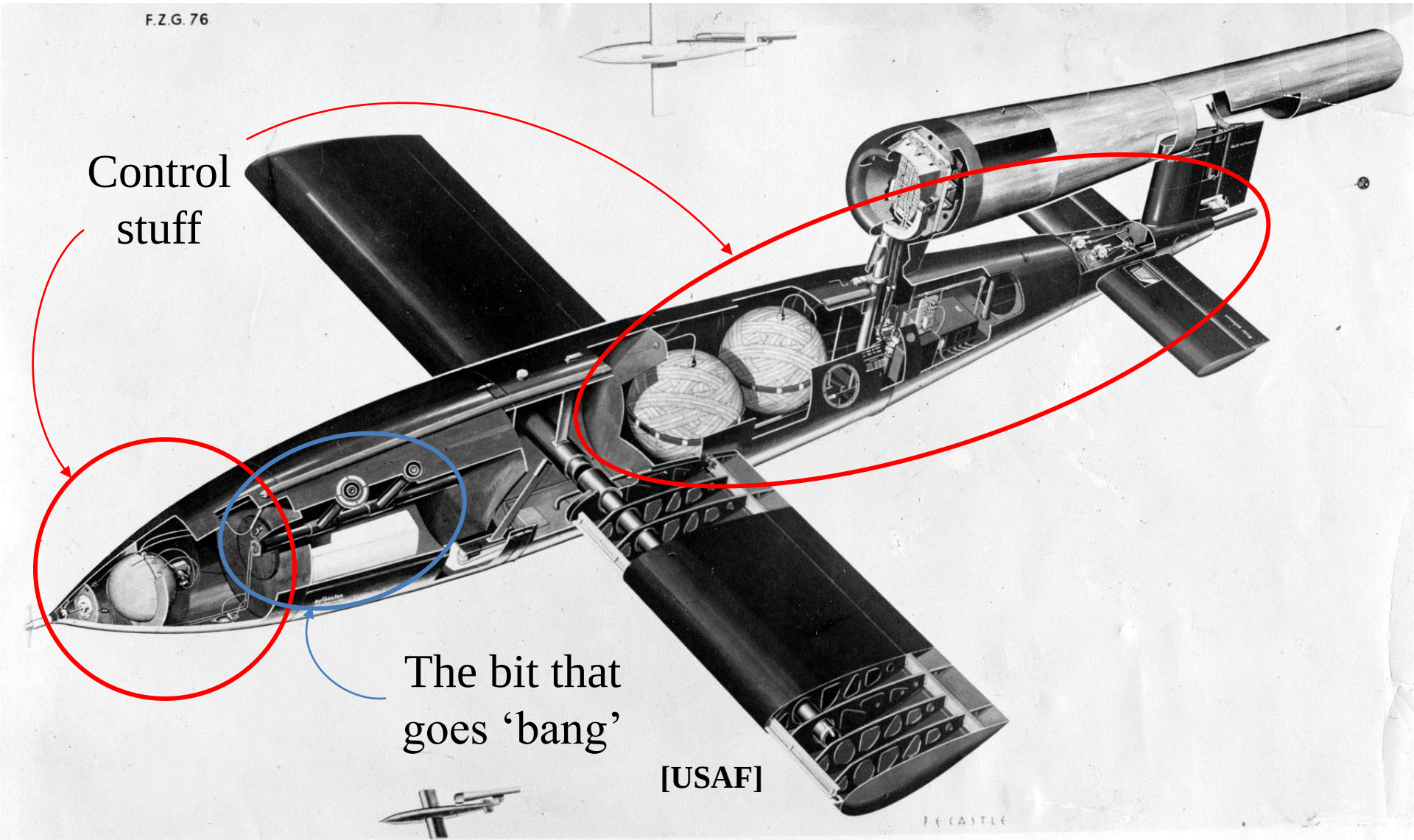
Introduction to Digital Control

Digital control lolwut?

Once upon a time…

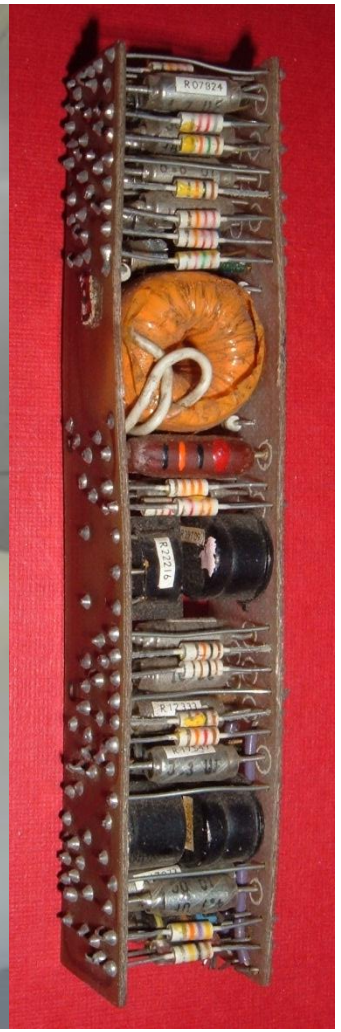
- Electromechanical systems were controlled by electromechanical compensators
  - Mechanical flywheel governors, capacitors, inductors, resistors, relays, valves, solenoids (fun!)
  - But also complex and sensitive!
- Humans developed sophisticated tools for designing reliable analog controllers

Eg. Early UAV flight control



Eg. missile guidance

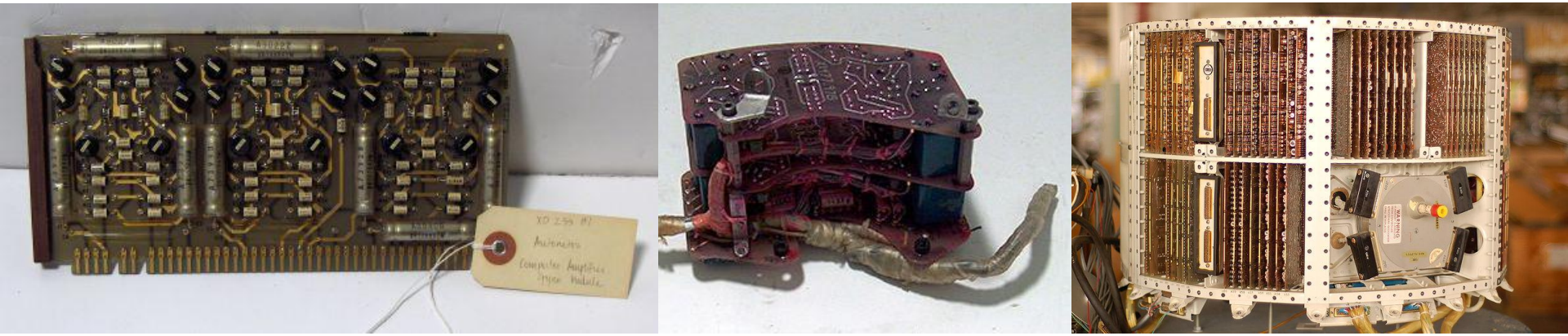
[Computer History Museum]



[Arnold Reinhold]

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**

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 point

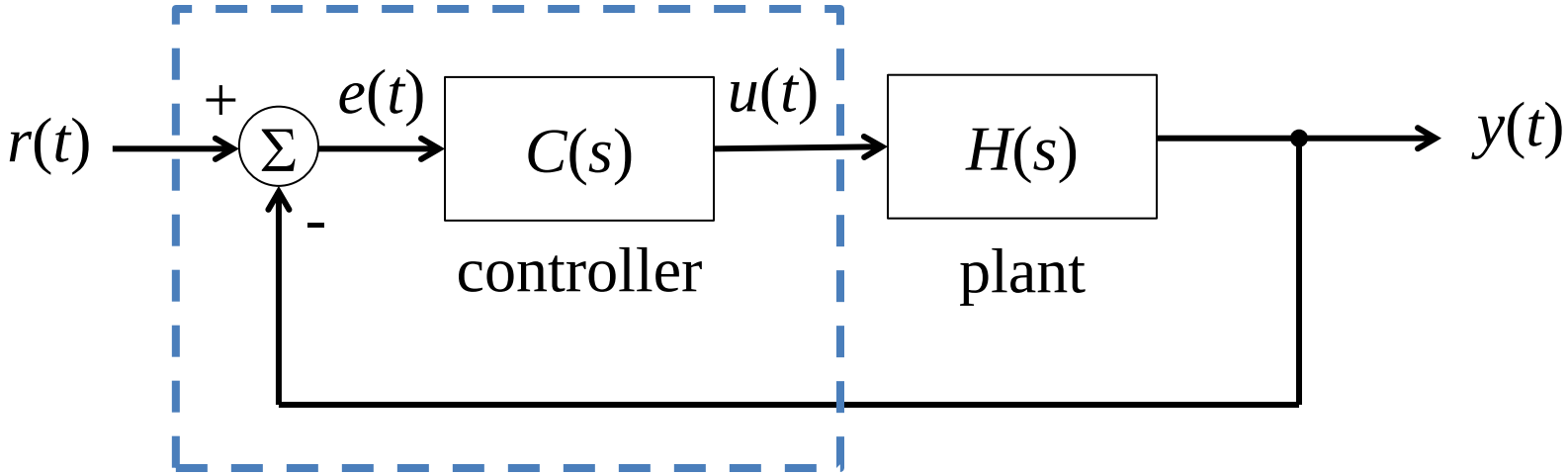
- While there are discrete analogues for every part of continuous systems theory, there are unique and important differences you must be familiar with

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

Digital Control Basics

Archetypical control system

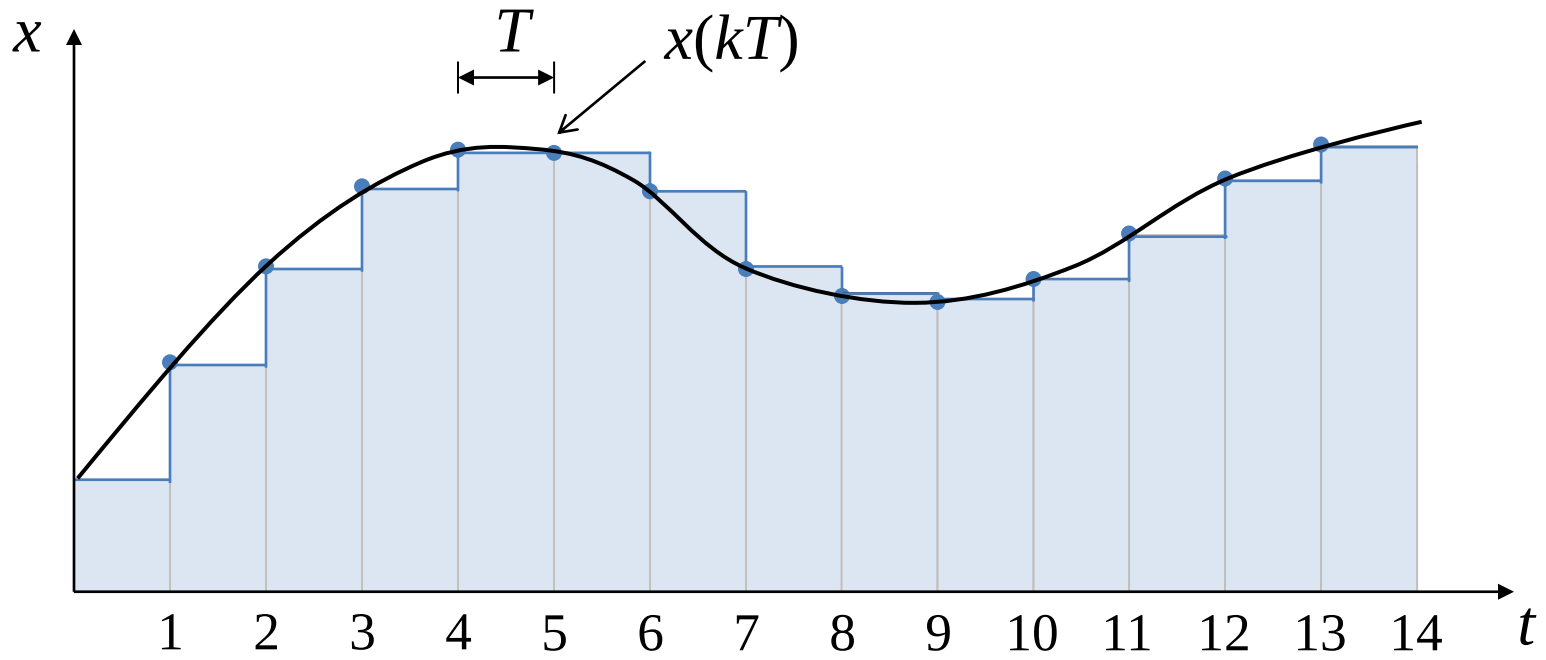
- Consider a continuous control system:



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

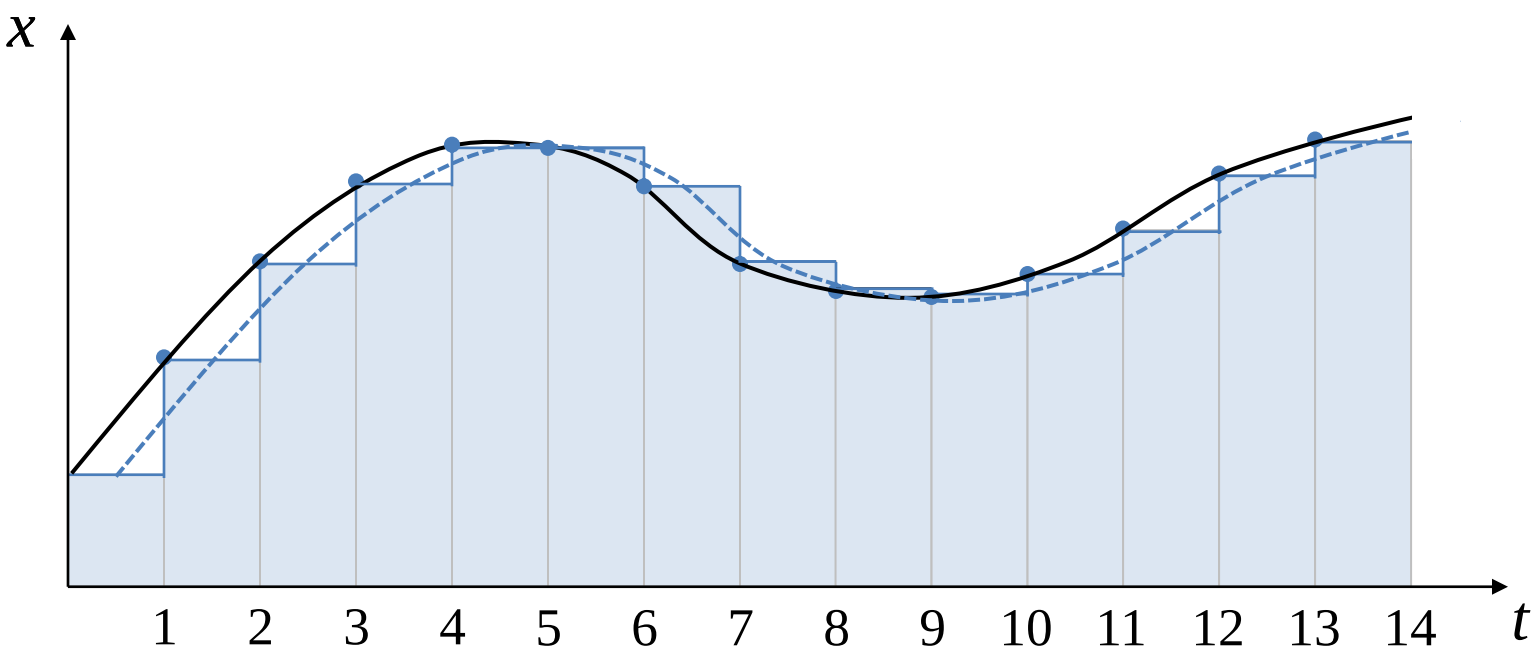
Return to the discrete domain

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



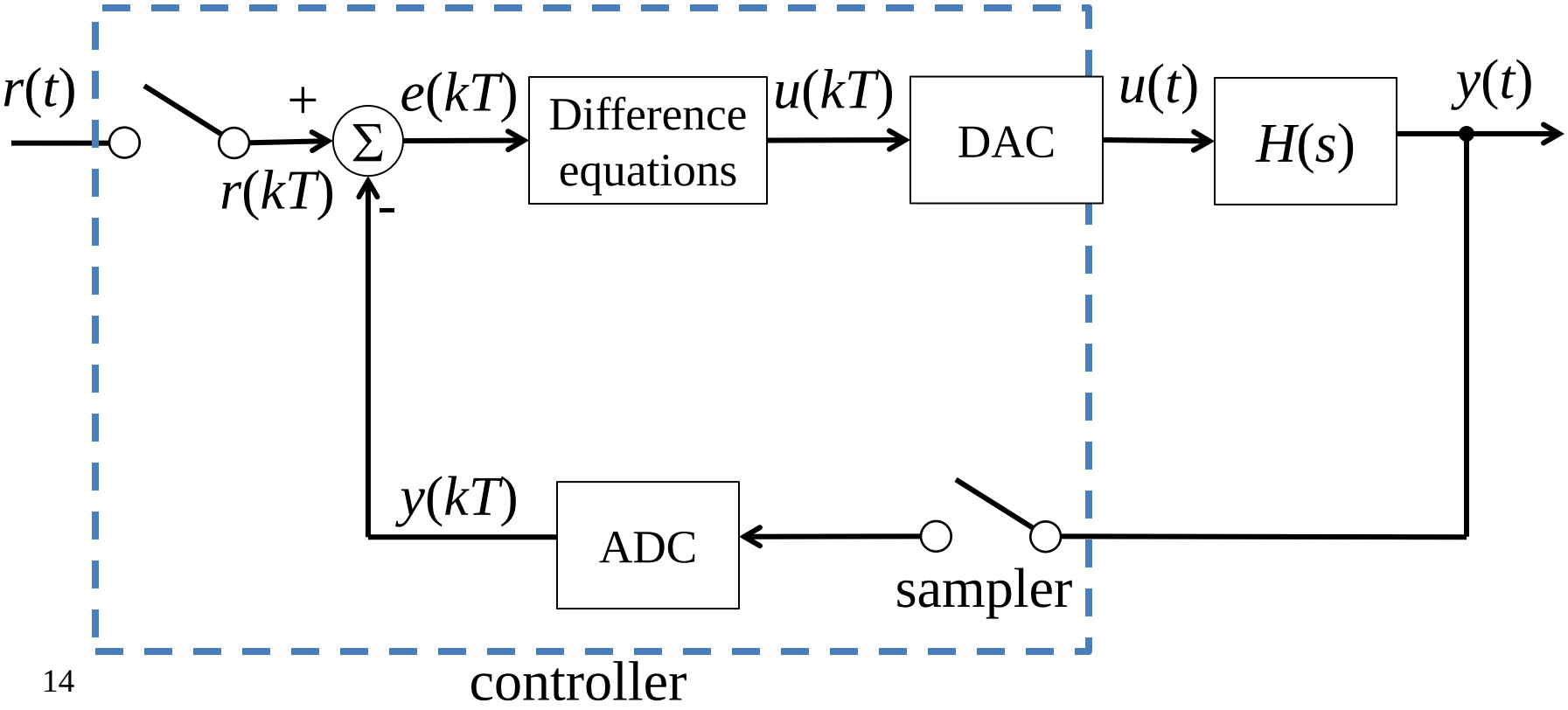
Zero Order Hold

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



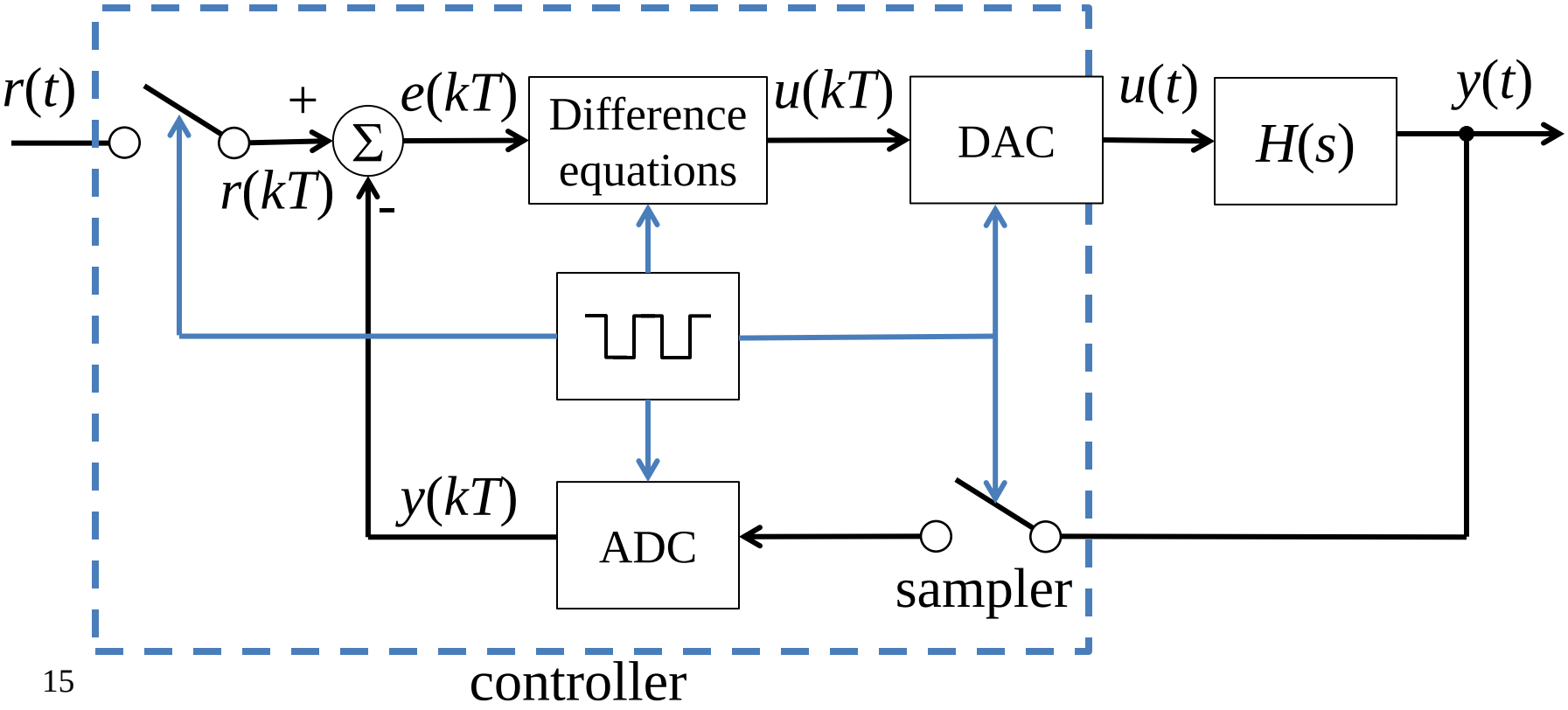
Digitisation

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

The Root Locus

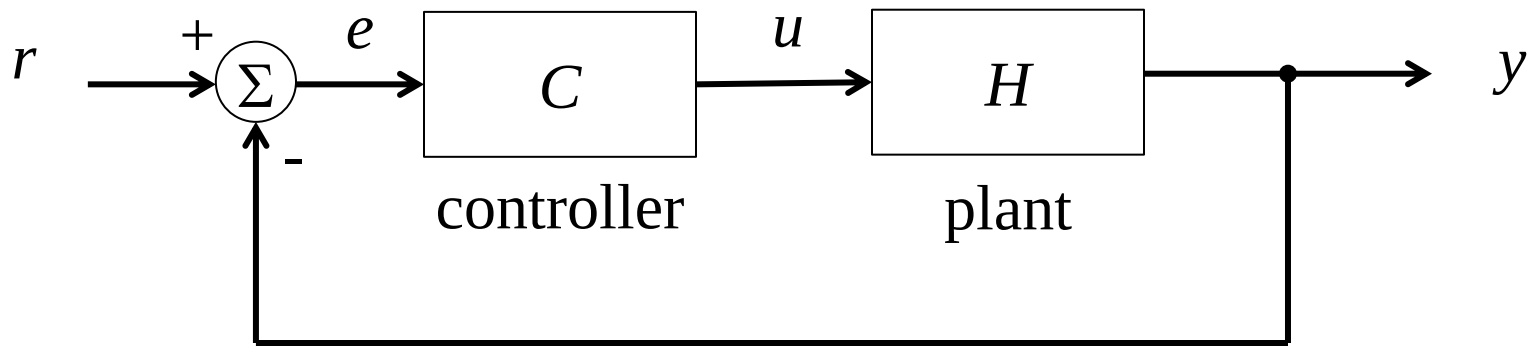
Quick refresher: the root locus

- 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

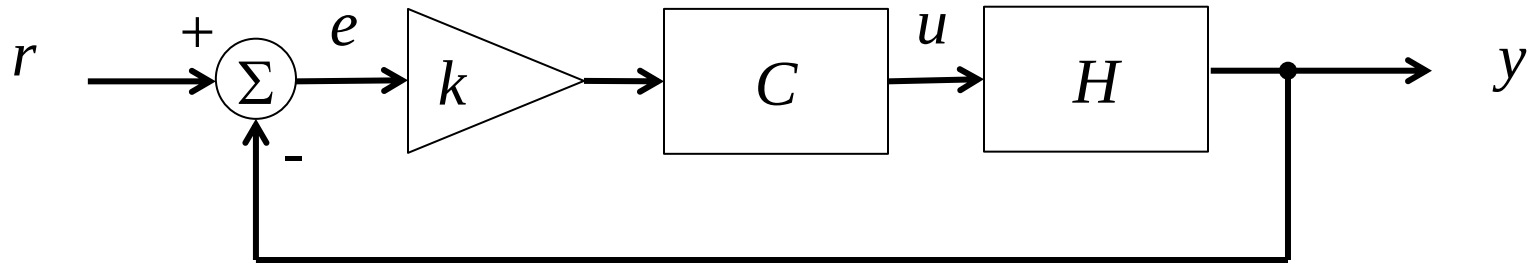
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

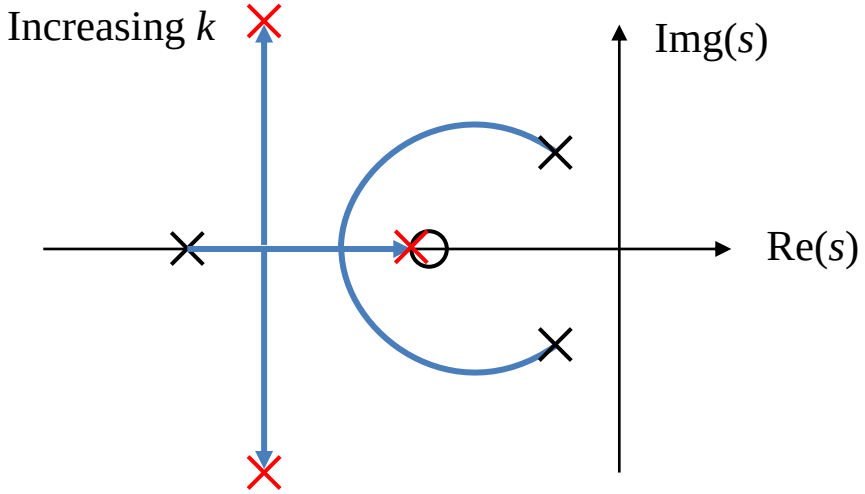
$$\frac{y}{r} = \frac{kC_nH_n}{C_dH_d + kCnHn}$$

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



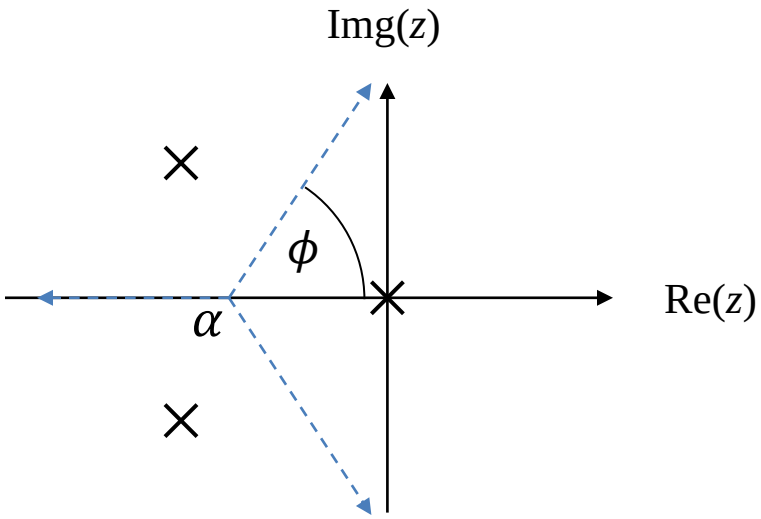
(In practice you'd plot these with computers)

Quick ‘n dirty guide to root loci

- Pole departure asymptotes as $k \to \infty$ are determined by pole-zero excess $(n - m)$:

$$\alpha = \frac{\sum p_i - \sum z_i}{n-m}, \phi_l = \frac{\pi + 2\pi(l-1)}{n-m}$$

$$\text{for } l = 1, 2, \ldots (n-m)$$



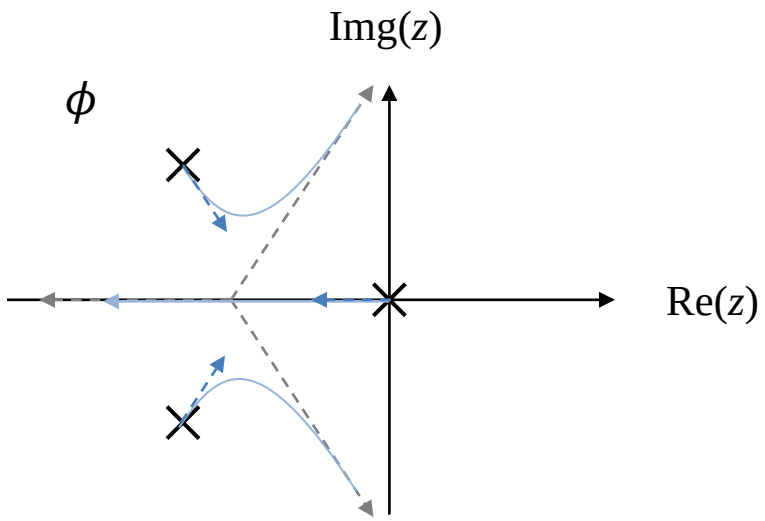
Quick ‘n dirty guide to root loci

- Pole departure/zero arrival angles given by:

$$q\phi_{\text{dep}} = \sum\psi_i - \sum\phi_i - \pi - 2\pi l$$

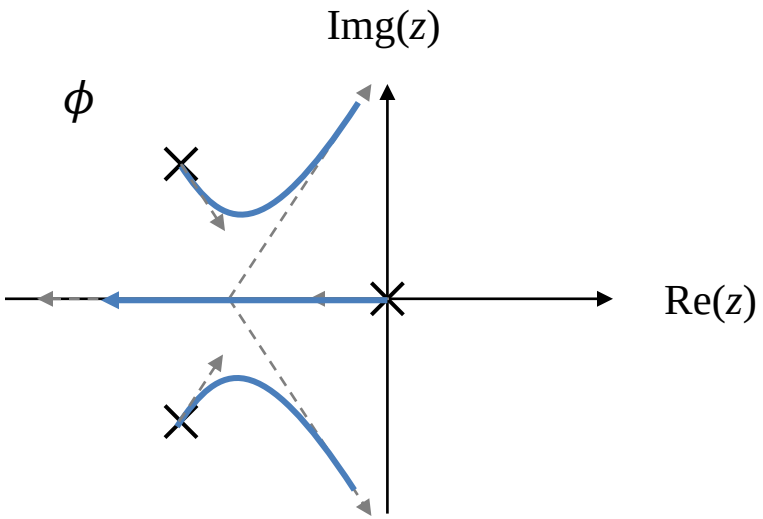
$$q\psi_{\text{arr}} = \sum\phi_i - \sum\psi_i + \pi + 2\pi l$$

for q^{th} poles or zeros, and for $l = 1, 2, \ldots q$



Quick ‘n dirty guide to root loci

- Draw the root locus on the real axis to the left of an odd number of poles
  - Zeros count as ‘negative’ poles
- Now draw the locus to observe the angle constraints

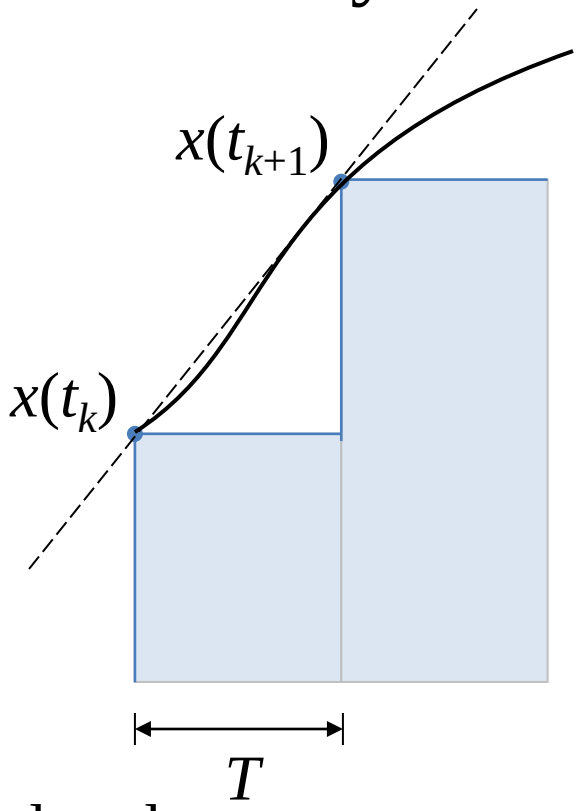


Approximation Methods

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

Euler's method

- Euler approximation can produce a system z-transform directly
- Use the substitution:

$$s^n = \left(\frac{z-1}{T}\right)^n$$

Tustin's method

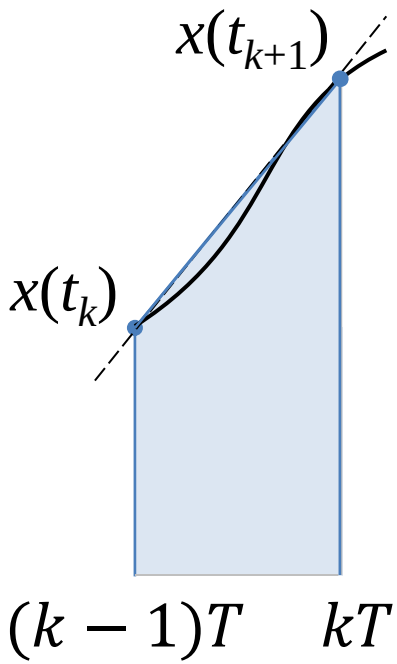
- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:

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

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 $(1 + z^{-1})$ 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

Approximation comparison

- We've been making a lot of approximations
  - Just how goods are these approximations?
  - As you might expect, it depends on how closely T matches the bandwidth of the system
  - Also varies by order of the approximation

Let's consider the system

$$H(s) = \frac{10}{(s+2)(s+10)}$$

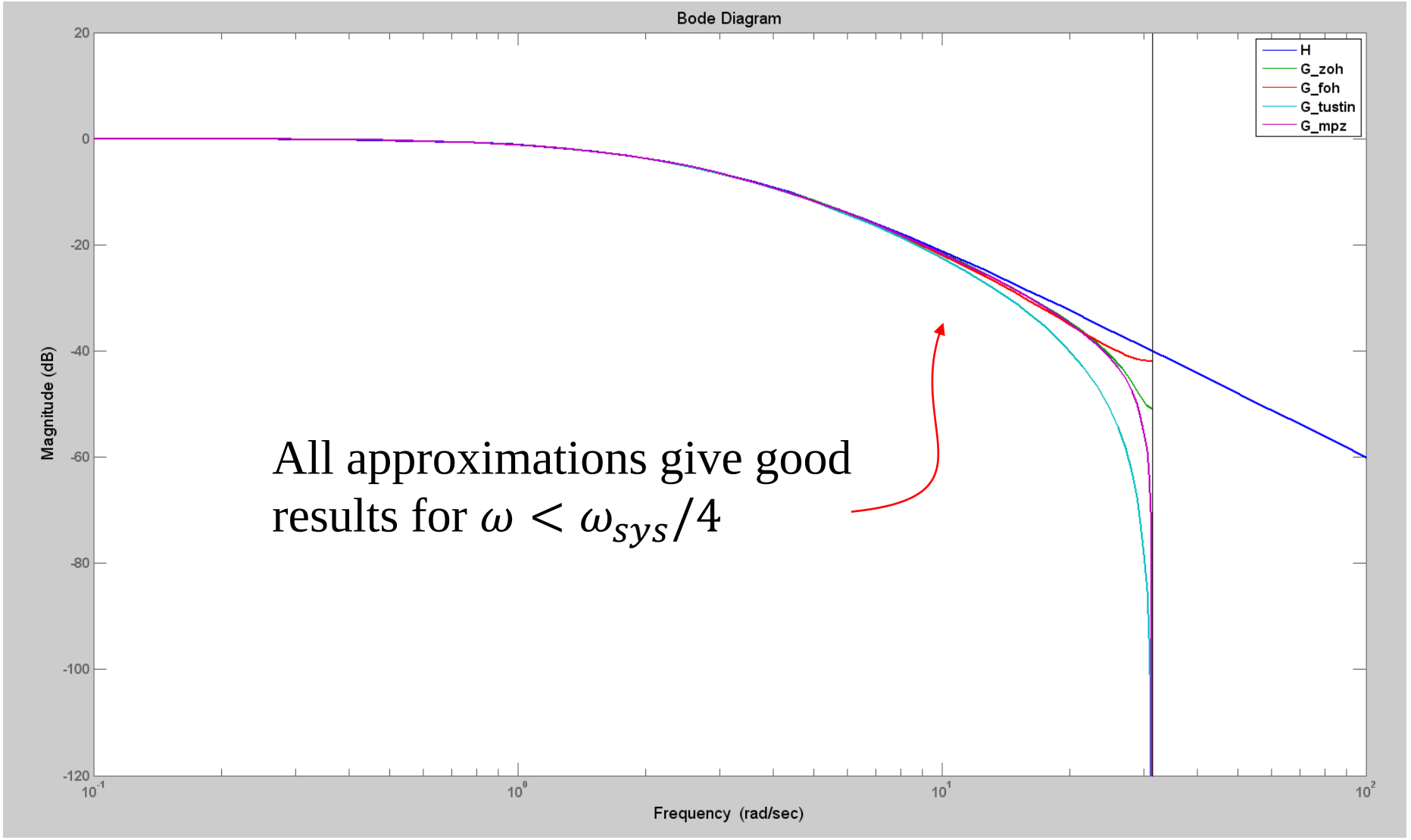
sampled at 10 Hz

Approximation comparison

- $G_{ZTF}(z) = \dfrac{0.07132z}{(z-0.8187)\,(z-0.6065)}$
- $G_{ZOH}(z) = \dfrac{0.039803\,(z+0.7919)}{(z-0.8187)\,(z-0.6065)}$
- $G_{FOH}(z) = \dfrac{0.014049\,(z+3.148)\,(z+0.2239)}{(z-0.8187)\,(z-0.6065)}$
- $G_{Tustin}(z) = \dfrac{0.018182\,(z+1)^2}{(z-0.8182)\,(z-0.6)}$
- $G_{MPZ}(z) = \dfrac{0.035662\,(z+1)}{(z-0.8187)\,(z-0.6065)}$

*FOH: First Order Hold ‘triangle approximation’

Approximation comparison



Effect of ZOH delay

- Recall the intrinsic time-delay associated with ZOH? What about that?
- This is the discrete domain; we can model the delay exactly:

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

This can be thought of a as a step input, followed by an immediate negative step one sample time later

The z-Transform

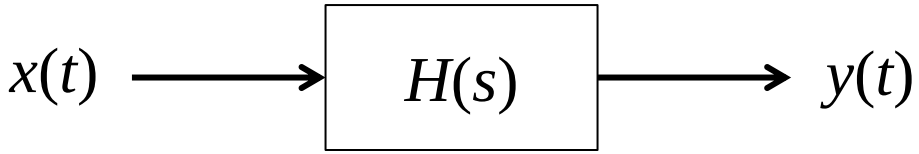
Coping with complexity

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

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

where

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



Is there a something similar for sampled systems?

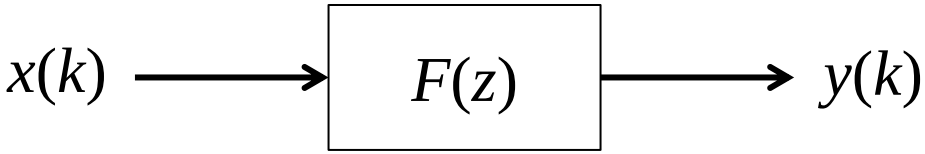
The z-transform

- The discrete equivalent is the z-Transform†:

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

and

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



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**: $\mathcal{Z}\{af(k) + bg(k)\} = \mathrm{a}F(z) + bG(z)$
  - **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}	$\frac{zTe^{-aT}}{(z-e^{-aT})^2}$
$\frac{1}{s^2+a^2}$	$\sin(akT)$	$\frac{z\sin aT}{z^2-(2\cos aT)z+1}$

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

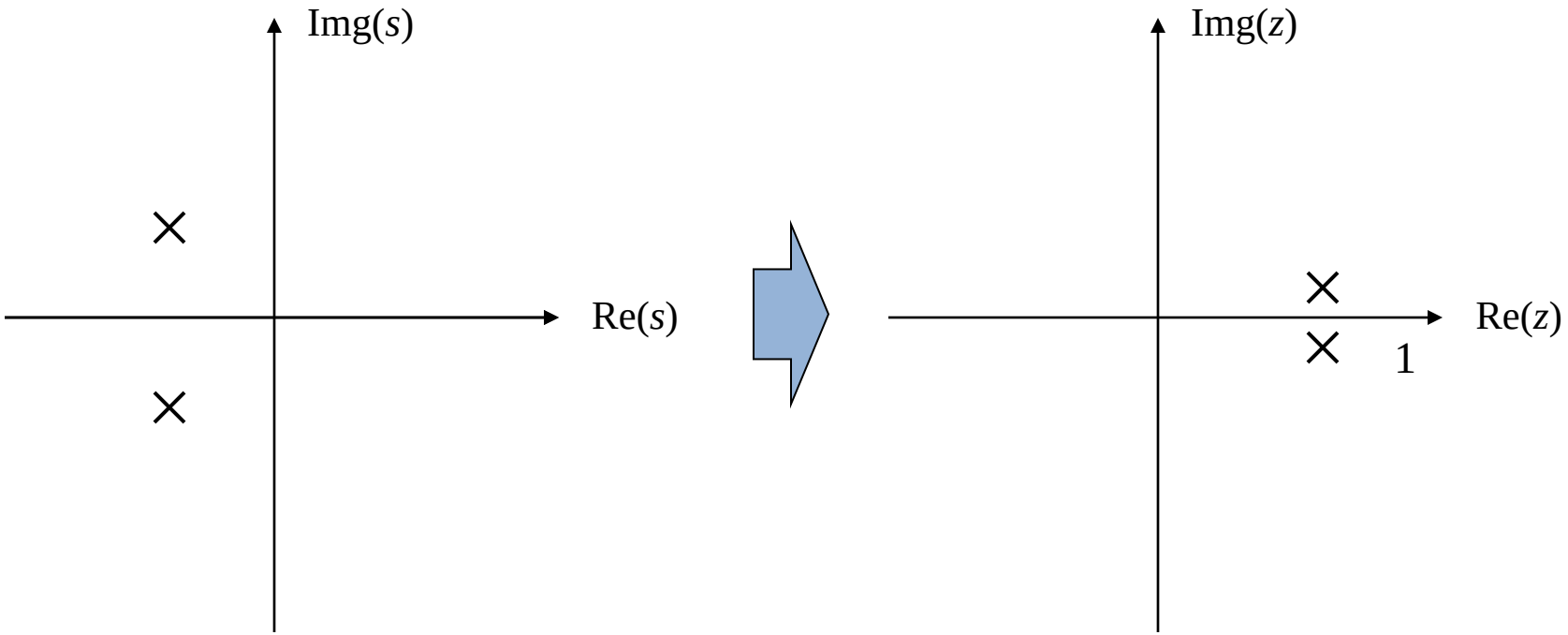
$$z = e^{sT}$$

That sounds somewhat profound… but what does it mean?

The z-Plane and Stability

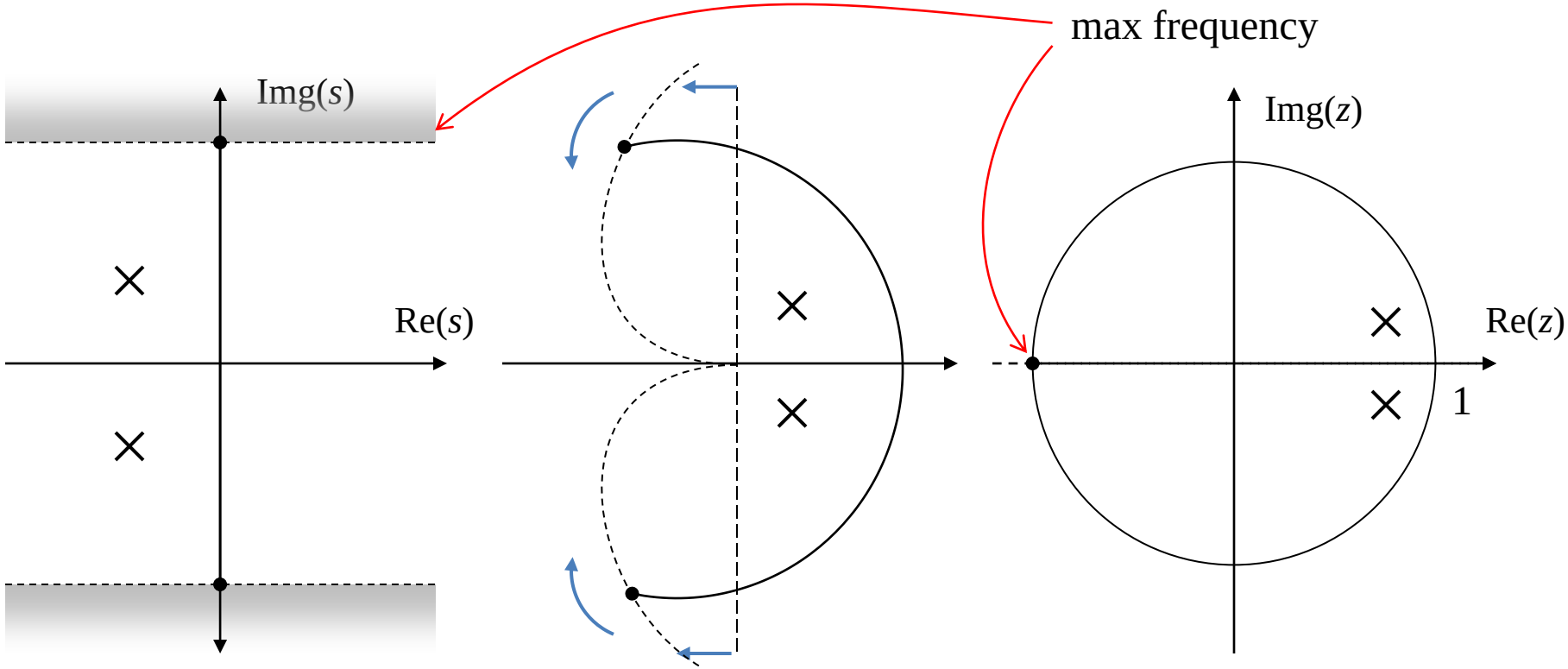
The *z*-Plane

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



Deep insight #1

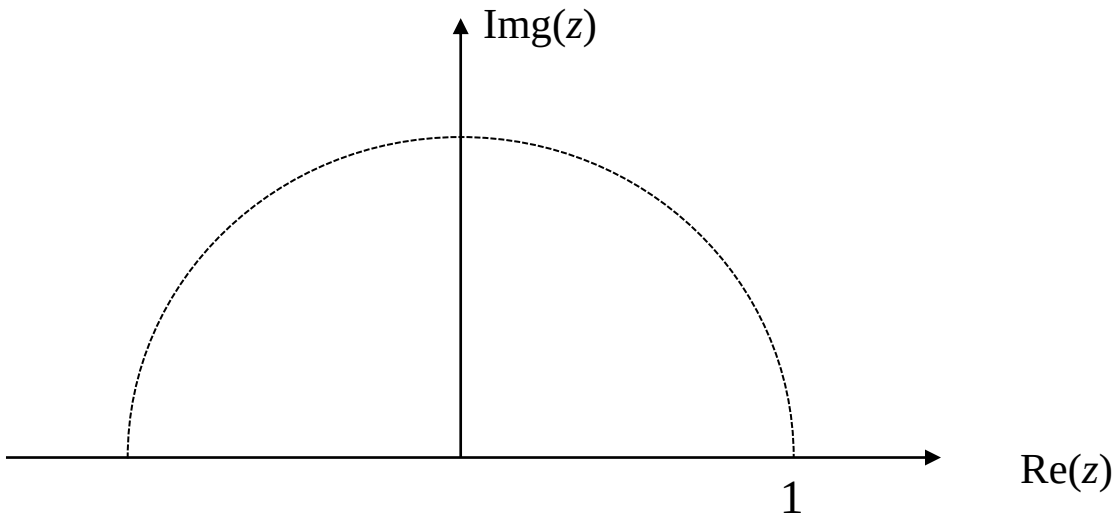
The mapping between continuous and discrete poles and zeros acts like a distortion of the plane



The *z*-Plane

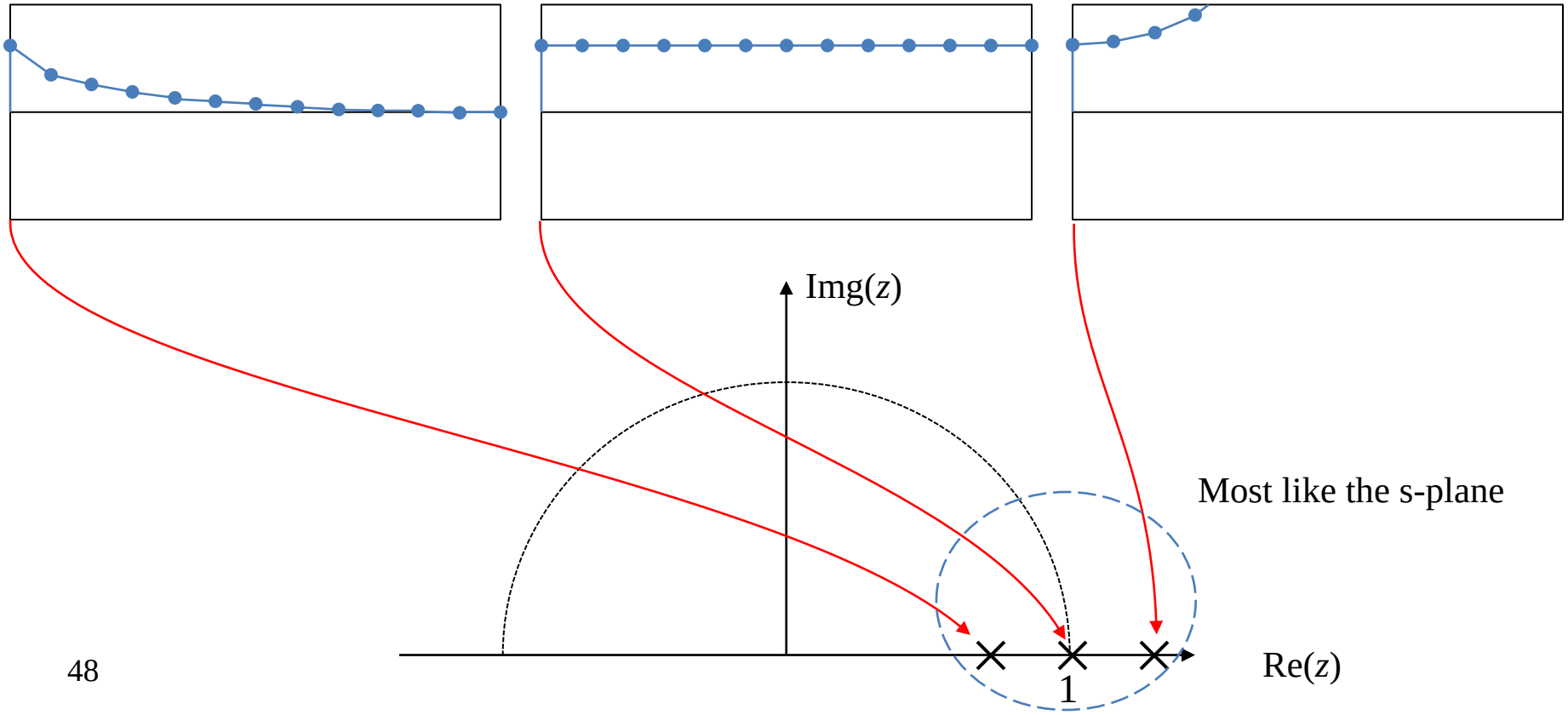
- We can understand system response by pole location in the *z*-plane

[Adapted from Franklin, Powell and Emami-Naeini]



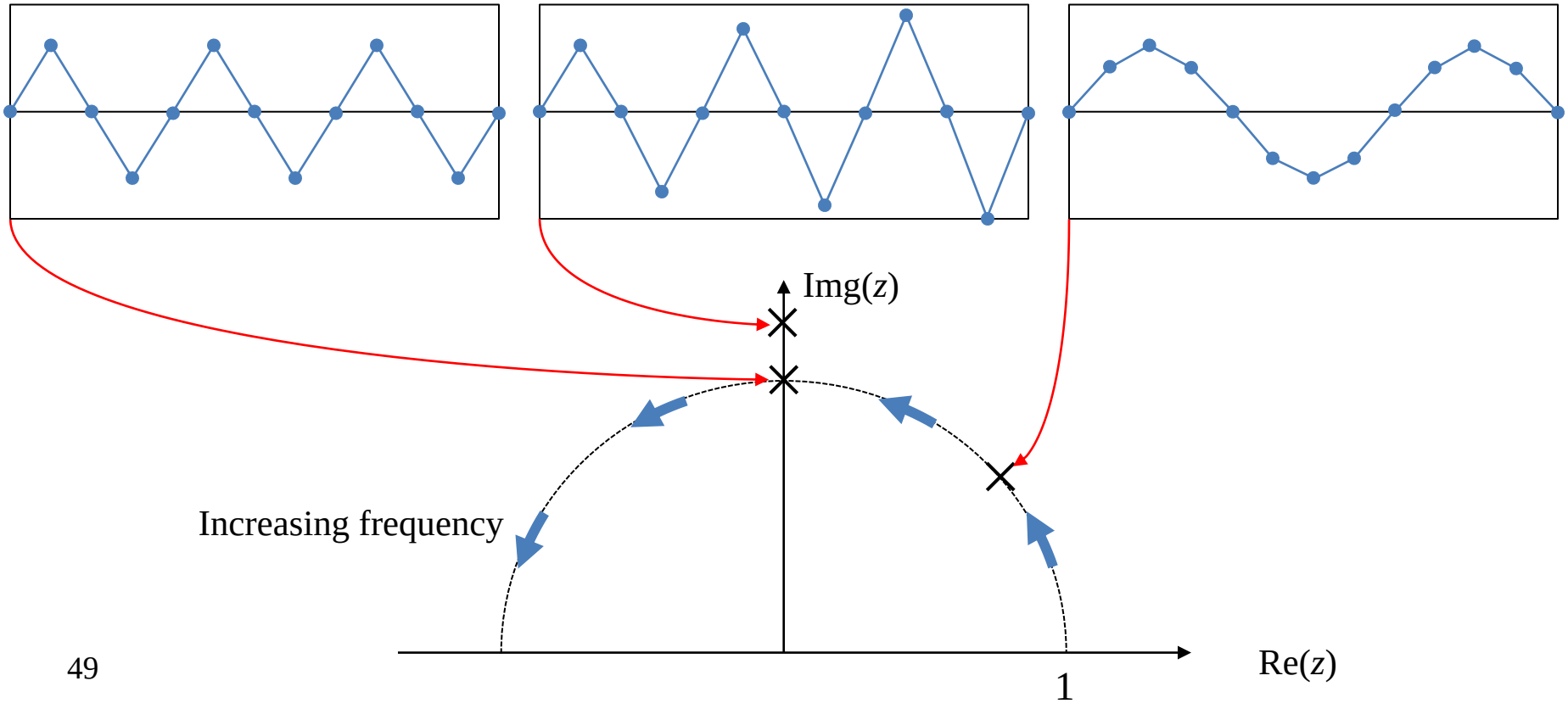
Effect of pole positions

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



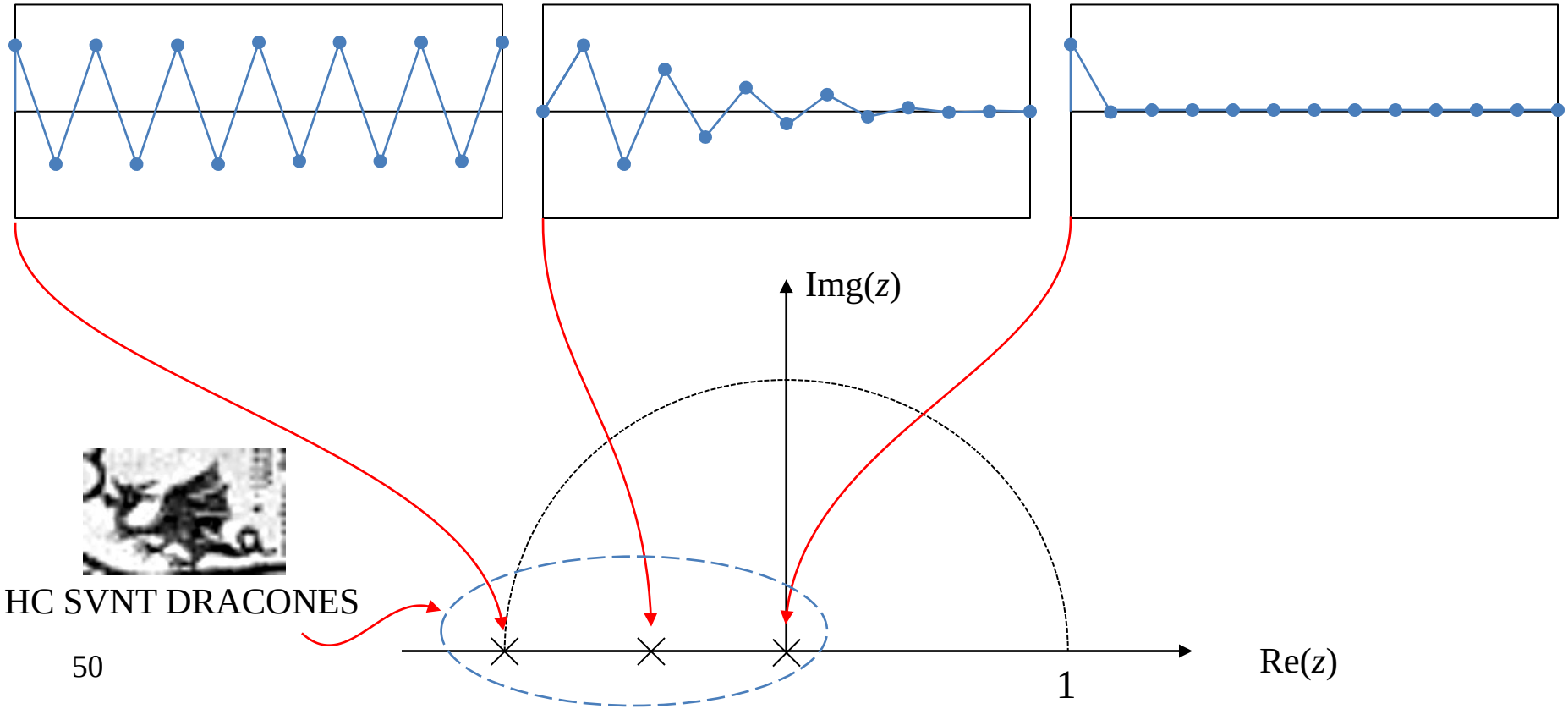
Effect of pole positions

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



Effect of pole positions

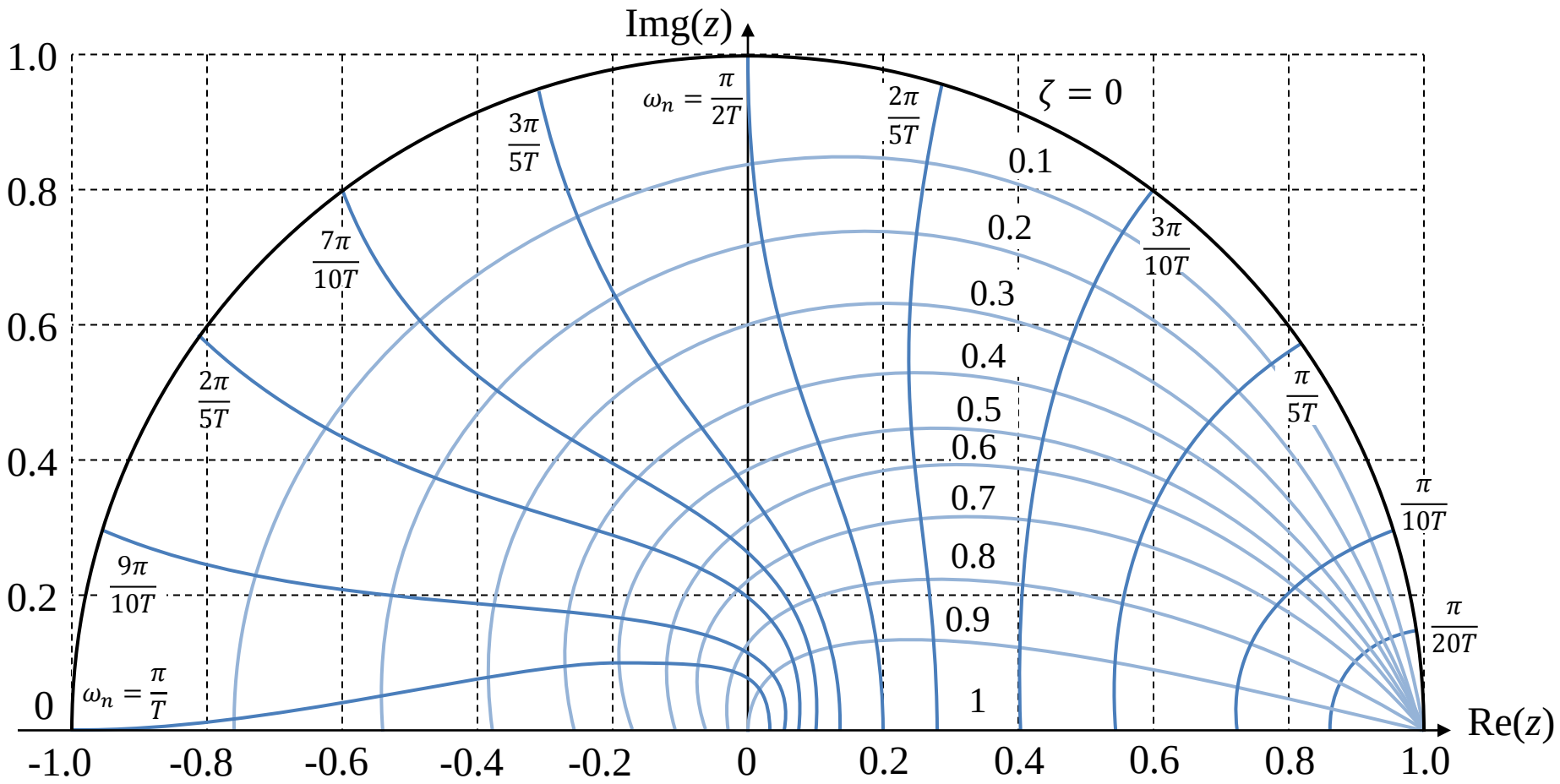
- We can understand system response by pole location in the *z*-plane



HC SVNT DRACONES

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]

Root loci in the z-plane

- The mathematics of polynomial algebra are the same in both the s-Plane and the z-Plane:

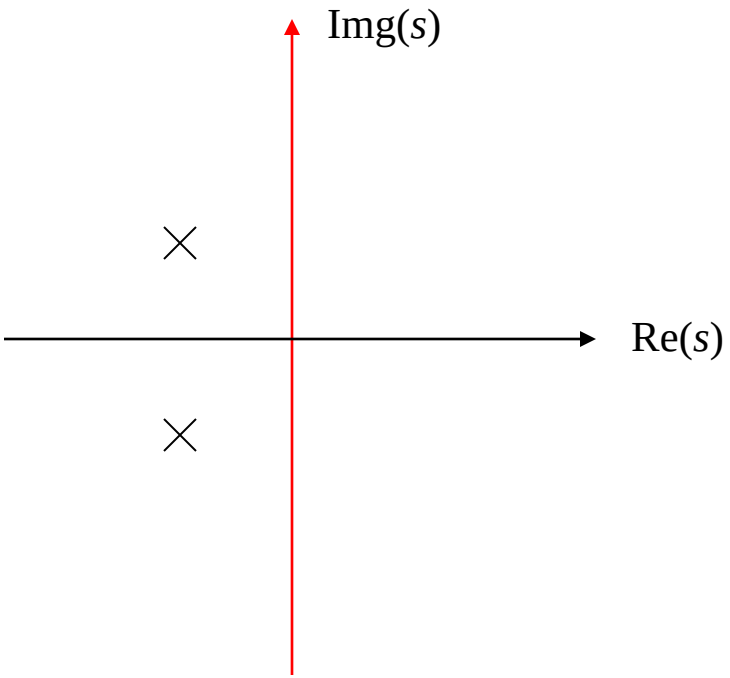
$$\text{eg. } s^2 + 2s + 1 = (s+1)^2$$

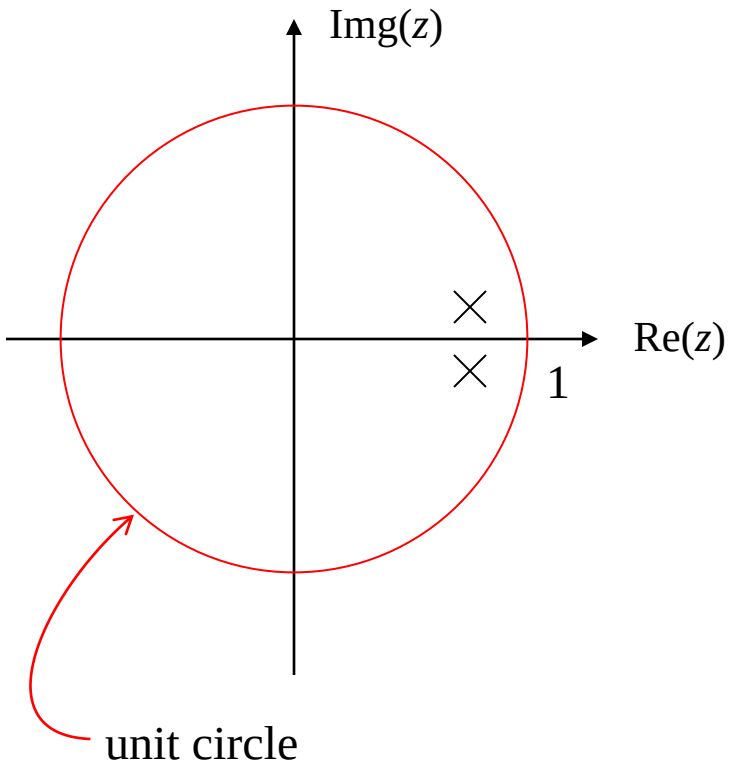
$$z^2 + 2z + 1 = (z+1)^2$$

  - Roots in a locus move in identical ways for identical polynomials
  - What changes is the physical interpretation of what the location of each pole represents in terms of response behaviour

z-plane stability

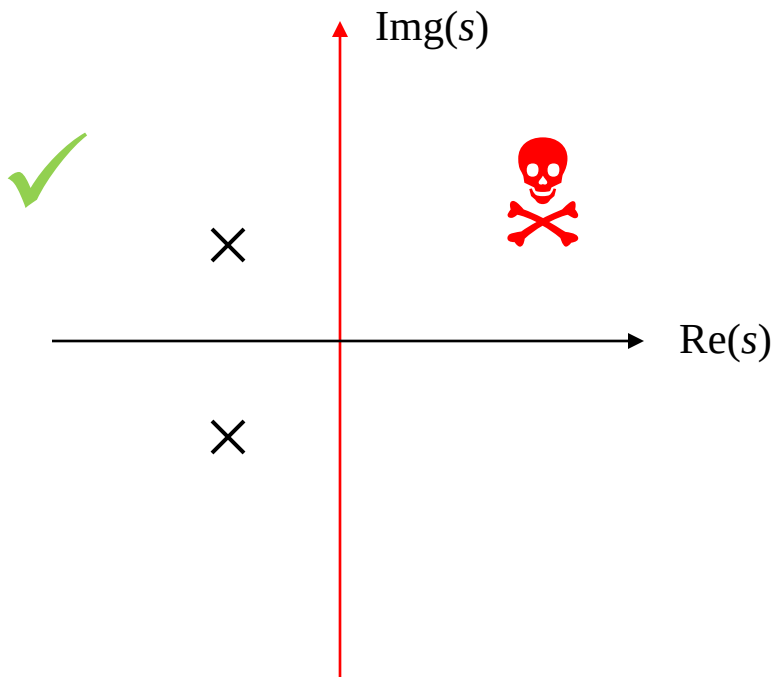
- In the *z*-domain, the unit circle is the system stability bound

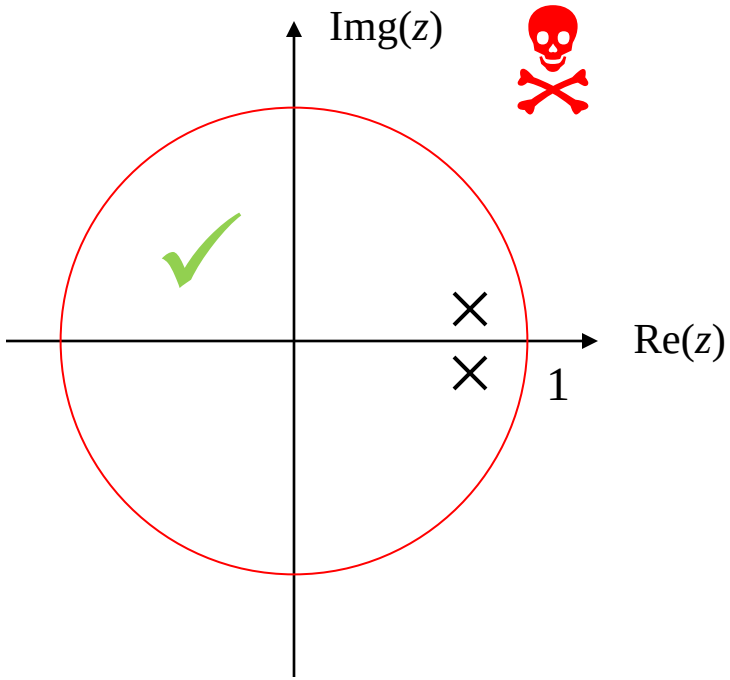




z-plane stability

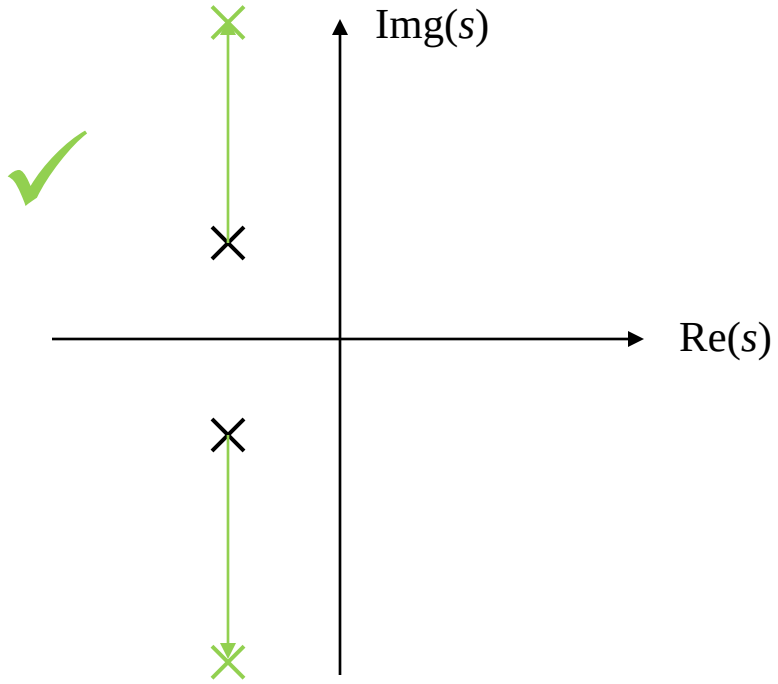
- In the *z*-domain, the unit circle is the system stability bound

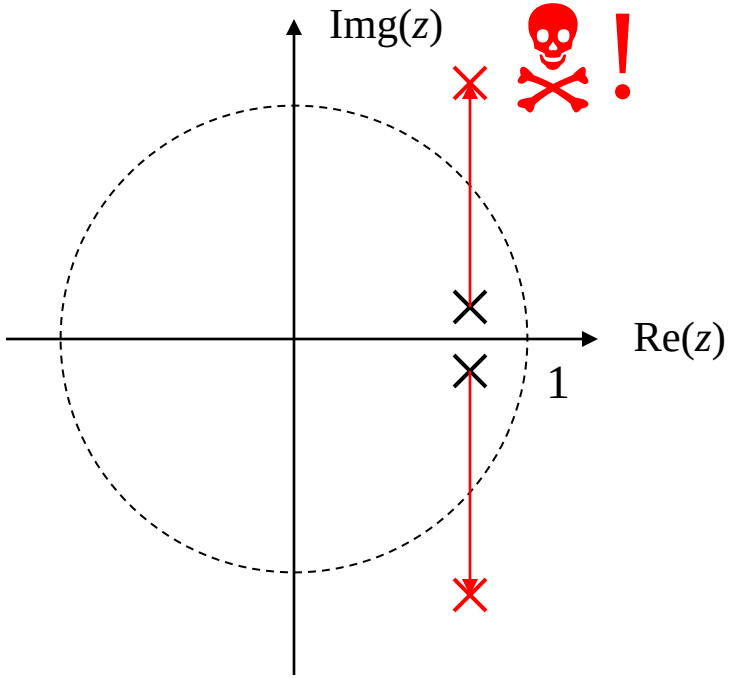




z-plane stability

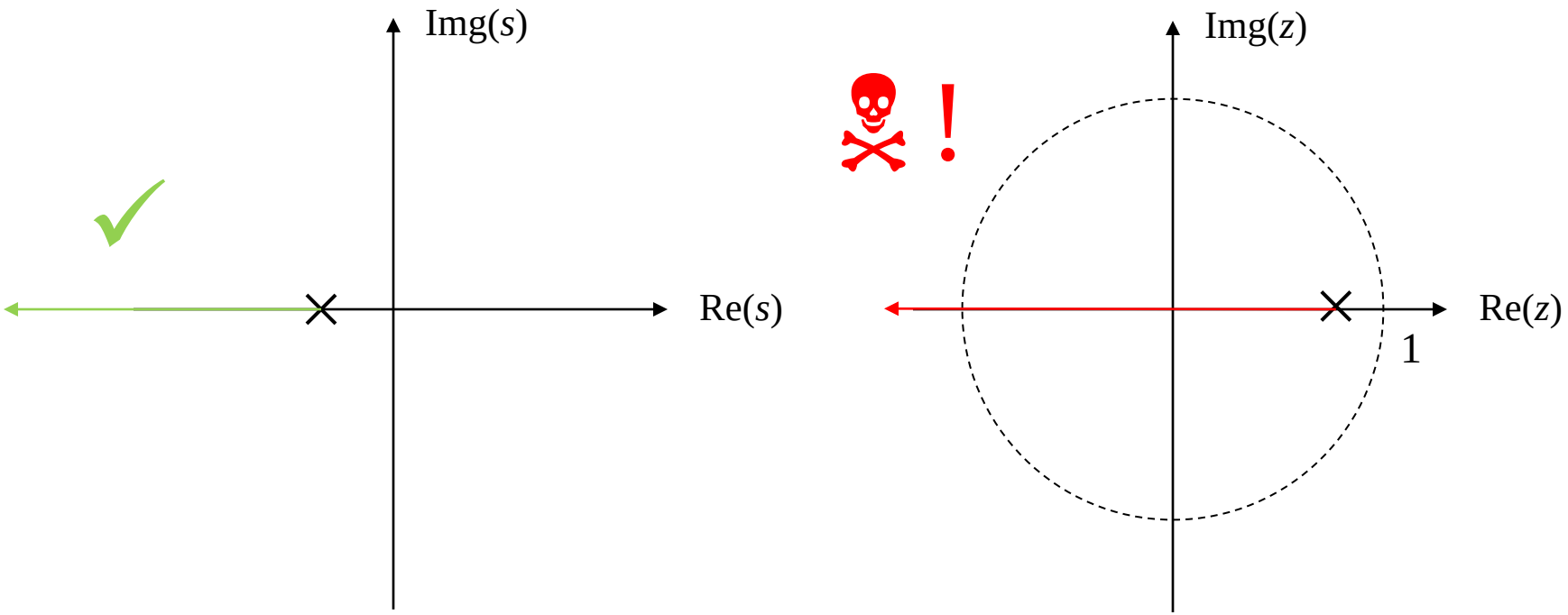
- The z-plane root-locus in closed loop feedback behaves just like the s-plane:





Deep insight #2

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



Digital Control Design

Two classes of control design

The system…

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

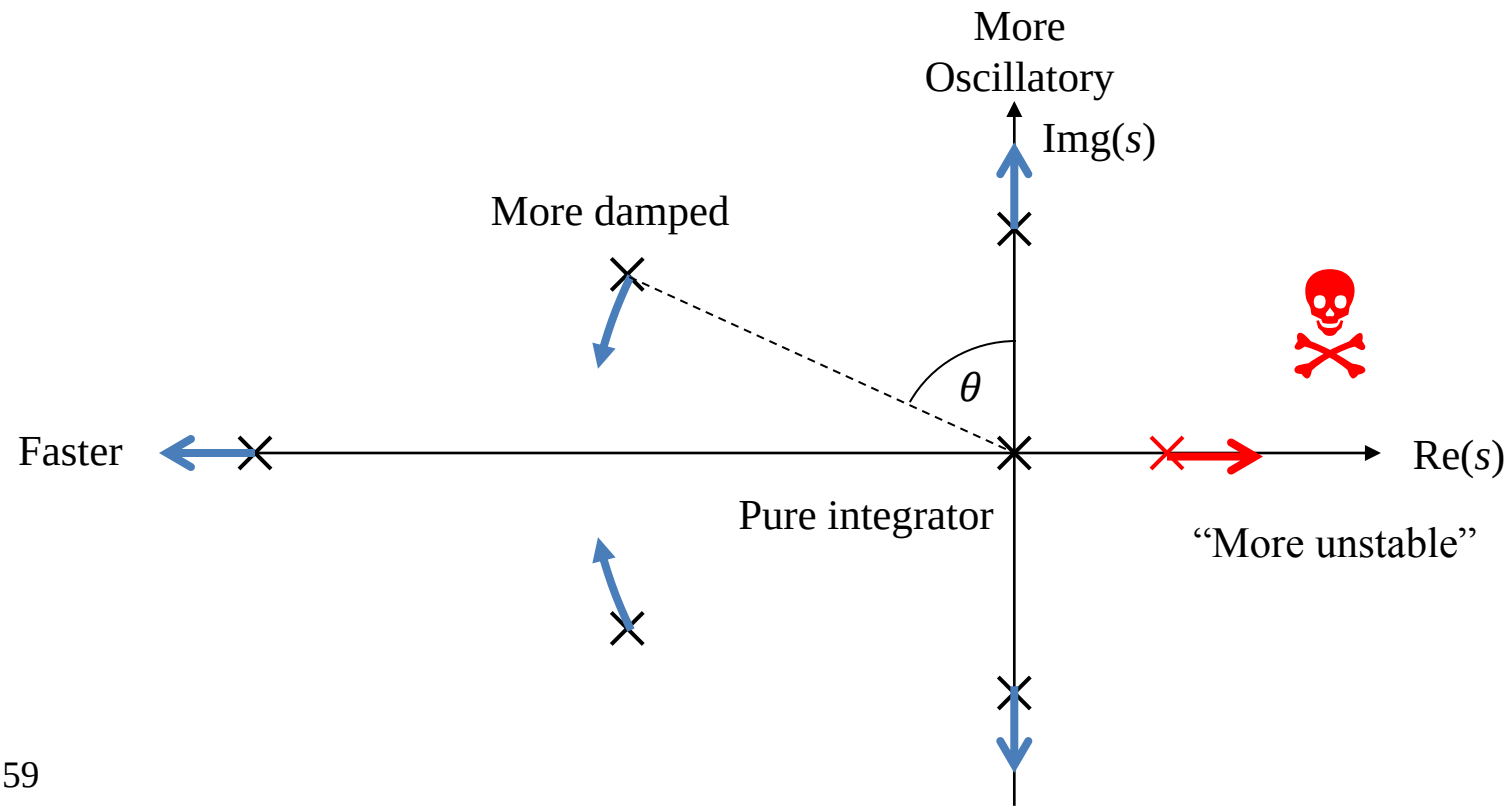
("Performance")

- Attempts to spontaneously disassemble itself

("Stability")

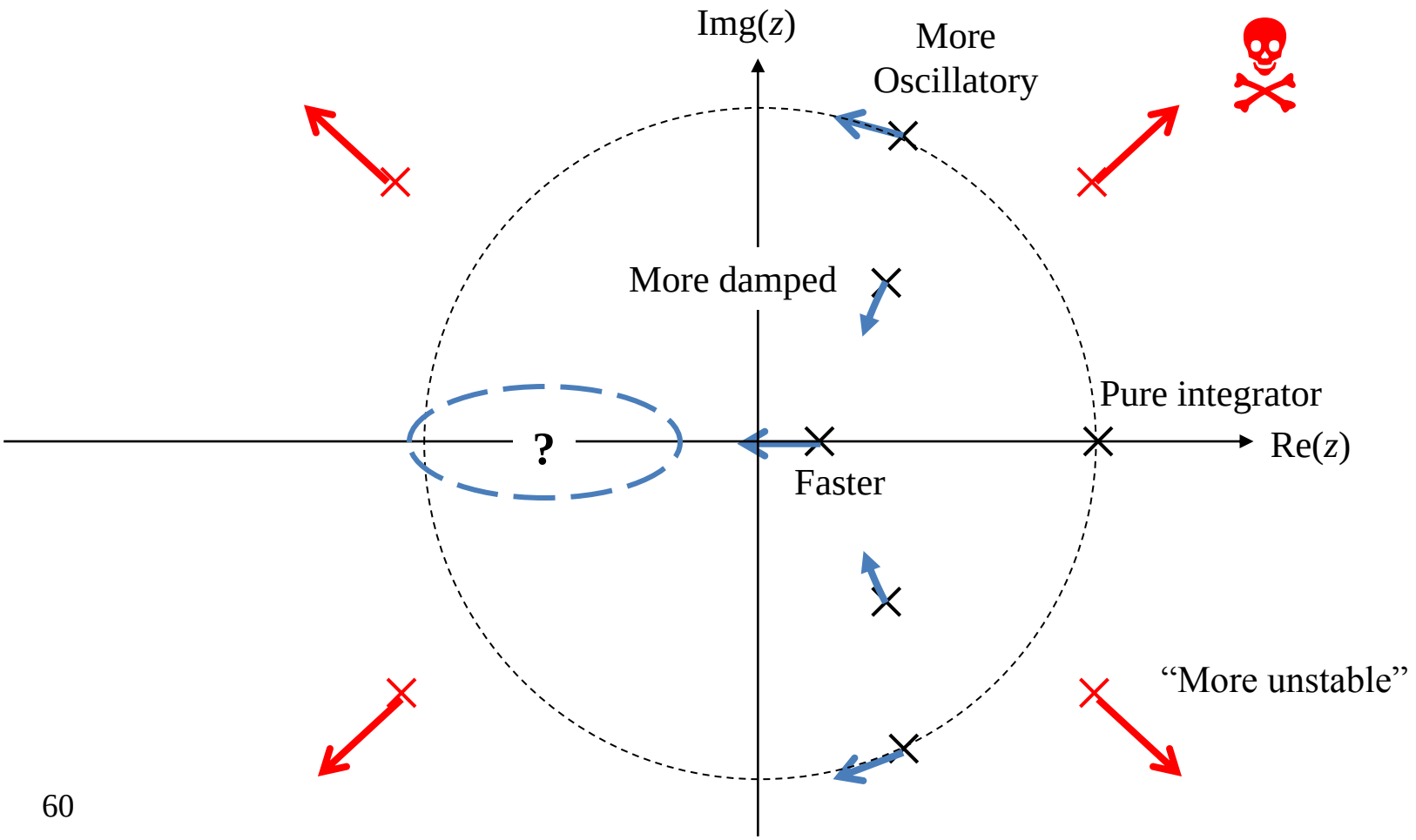
Recall dynamic responses

- Moving pole positions change system response characteristics



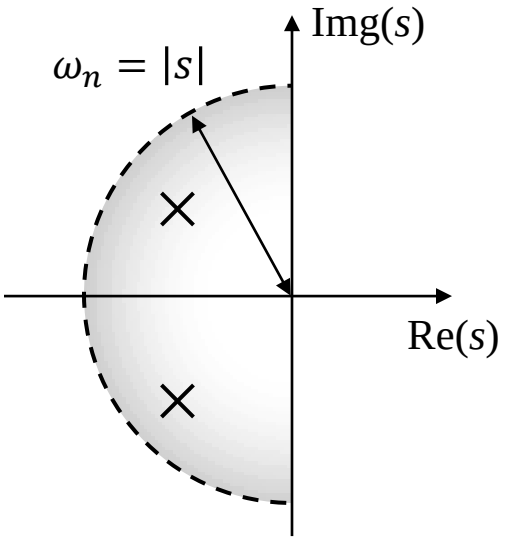
Recall dynamic responses

- Ditto the z-plane:

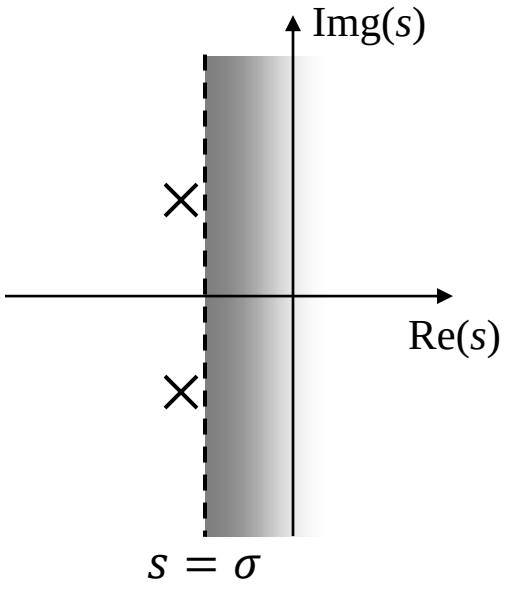


Specification bounds

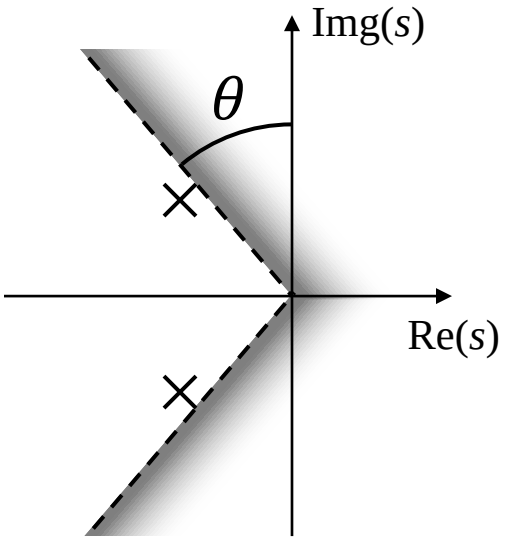
- Recall in the continuous domain, response performance metrics map to the s-plane:



$$|s| = \frac{1.8}{t_r}$$



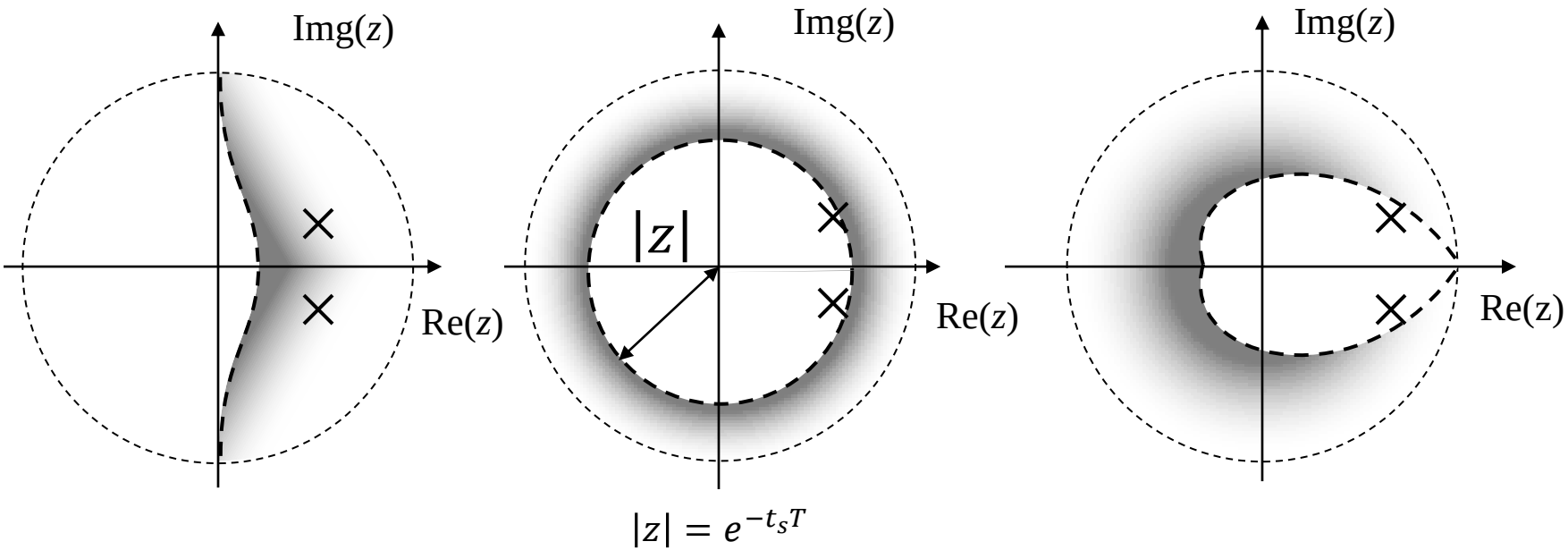
$$s = \frac{4.6}{t_s}$$



$$\theta = \sin^{-1}\zeta$$

Discrete bounds

- These map to the discrete domain:



$$|z| = e^{-t_s T}$$

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

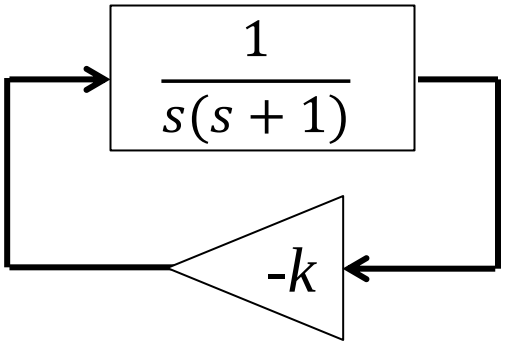
The fundamental control problem

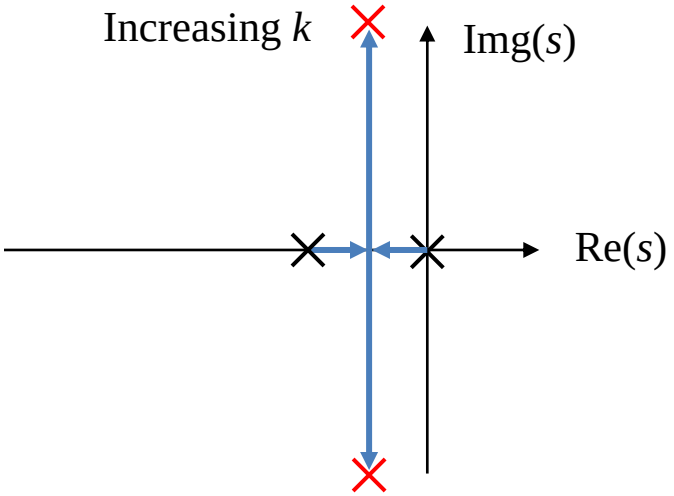
The poles are in the wrong place

How do we get them where we want them to be?

Recall the root locus

- We know that under feedback gain, the poles of the closed-loop system move
  - The root locus tells us where they go!
  - We can solve for this analytically*

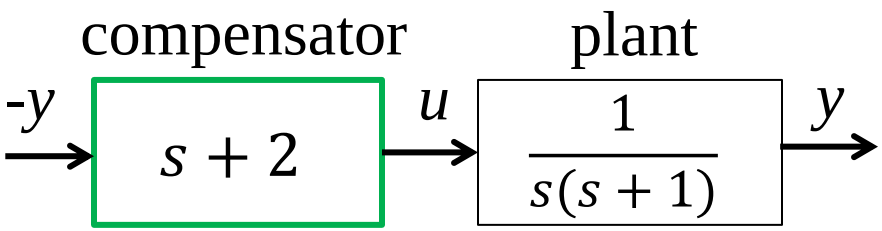


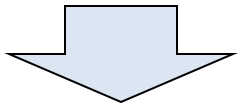


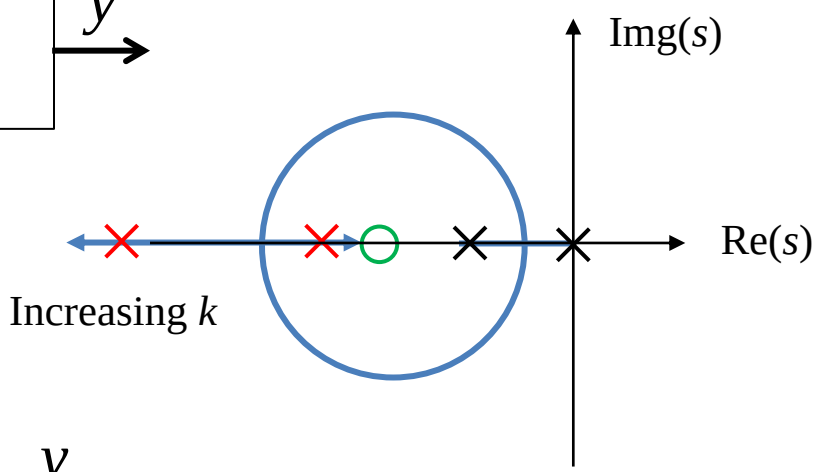
Root loci can be plotted for all sorts of parameters, not just gain!

Dynamic compensation

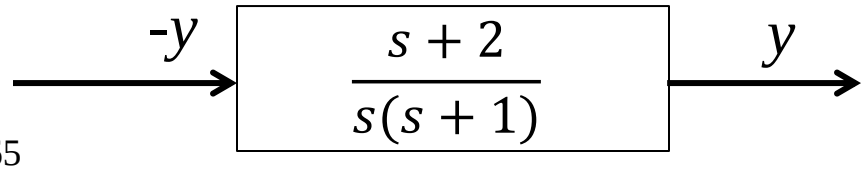
- We can do more than just apply gain!
  - We can add dynamics into the controller that alter the open-loop response







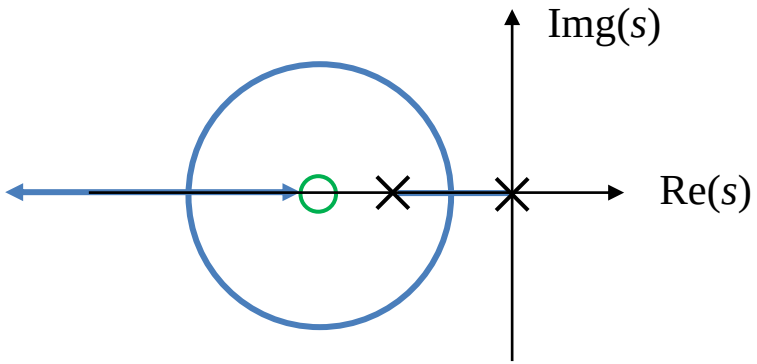
combined system



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)



Common Control Structures

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

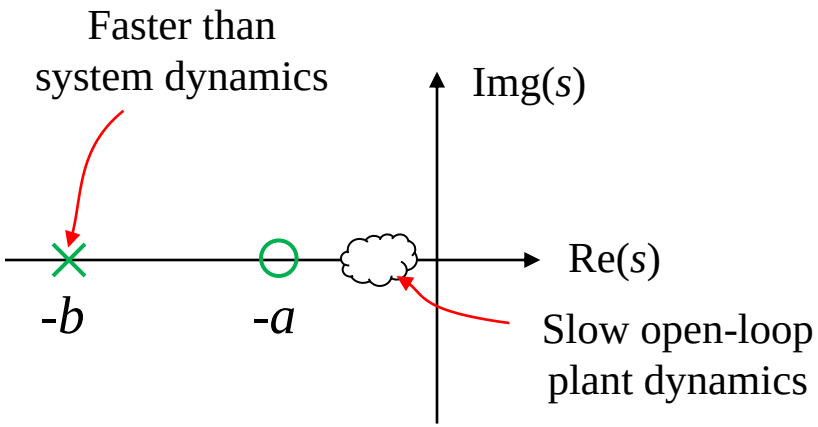
- Serve different purposes, but have a similar dynamic structure:

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

Note:

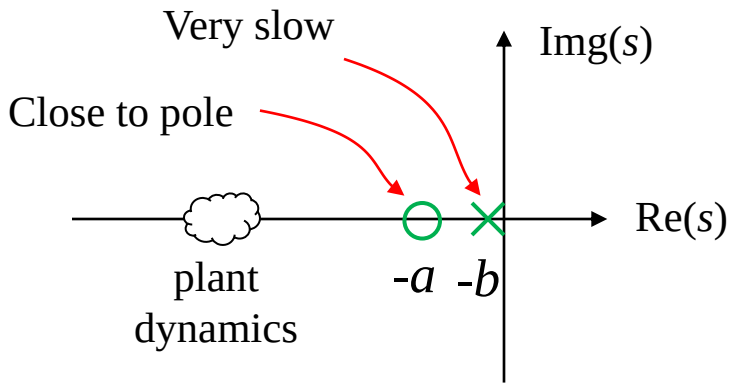
Lead-lag compensators come from the days when control engineers cared about constructing controllers from networks of op amps using frequency-phase methods. These days pretty much everybody uses PID, but you should at least know what the heck they are in case someone asks.

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*

PID

- 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

PID

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

You're already familiar with the effect of gain.

What about the other two?

Integral

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

Let's try it!

Integral

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

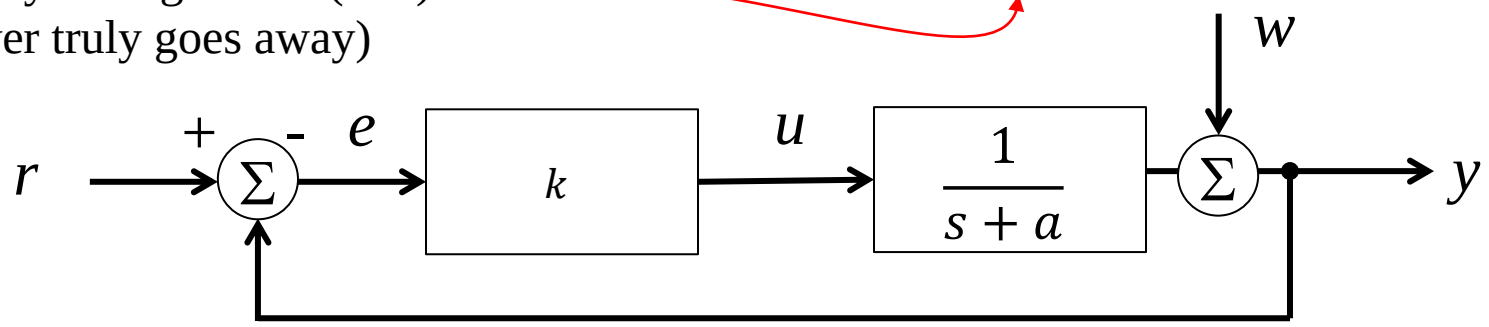
$$y = k\frac{1}{s+a}(r-y) + w$$

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

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

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

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



Now with added integral action

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

Same dynamics

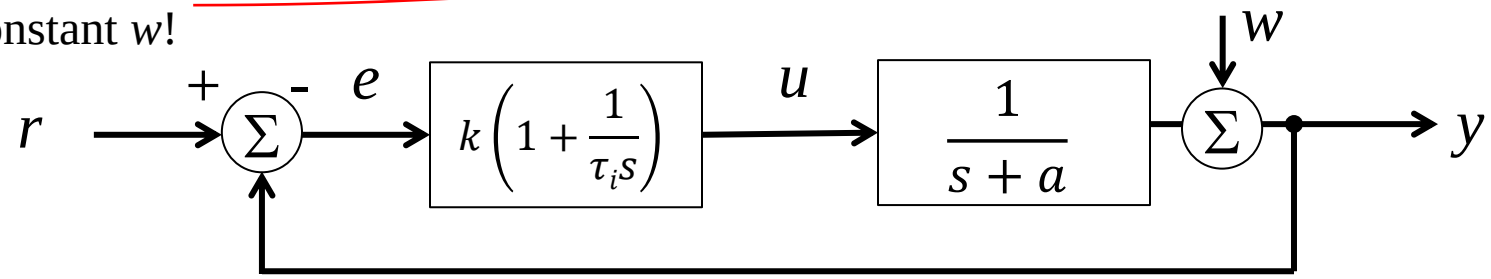
$$y = k\frac{s+\tau_i^{-1}}{s}\frac{1}{s+a}(r-y) + w$$

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

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

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

Must go to zero for constant *w*!



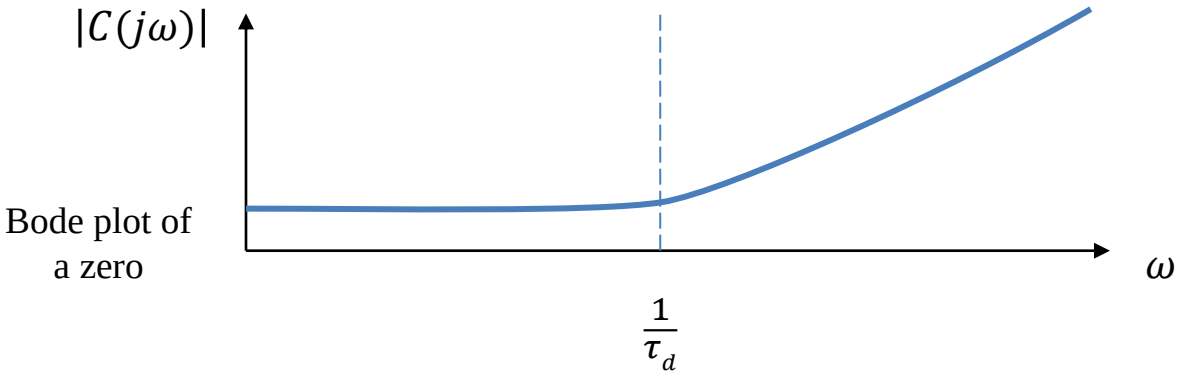
Derivative

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

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

Derivative

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



PID

- Collectively, PID provides two zeros plus a pole at z = 1 (discrete) or s = 0 (continuous)
  - 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

Be alert

- If gains and time-constants are chosen poorly, all of these compensators can induce oscillation or instability.

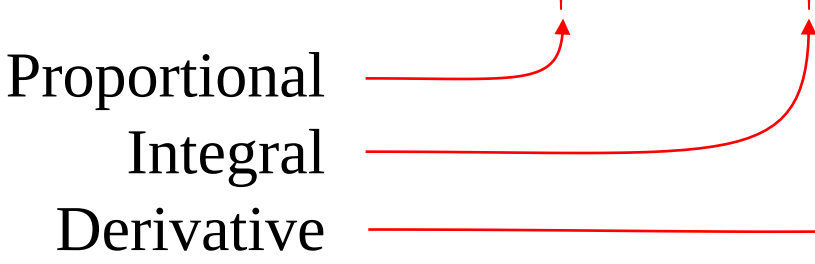
- However, when used properly, PID can stabilise even very complex unstable systems

Now in discrete

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

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

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



Control Design Process

Emulation vs Discrete Design

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

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

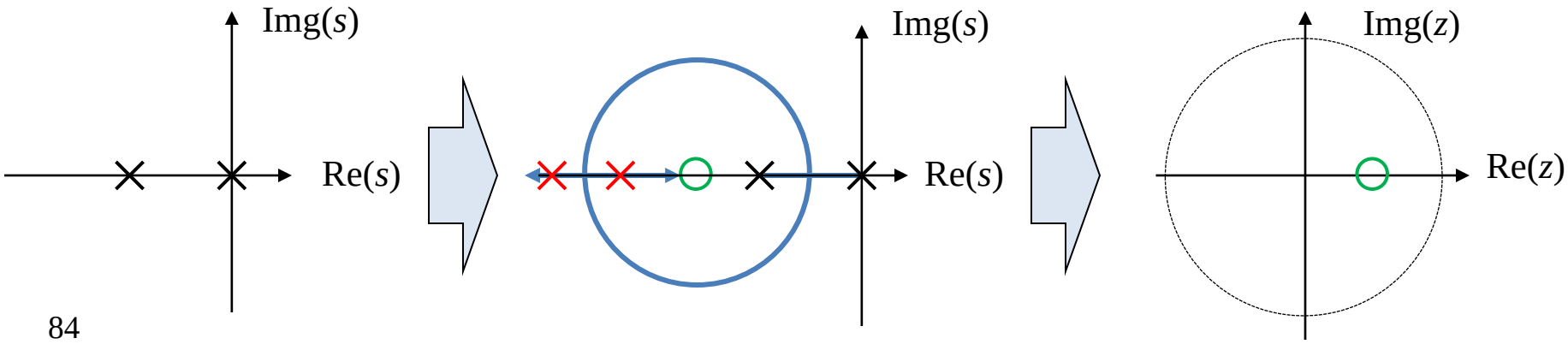
$$z^2 + 2z + 1 = (z+1)^2$$

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

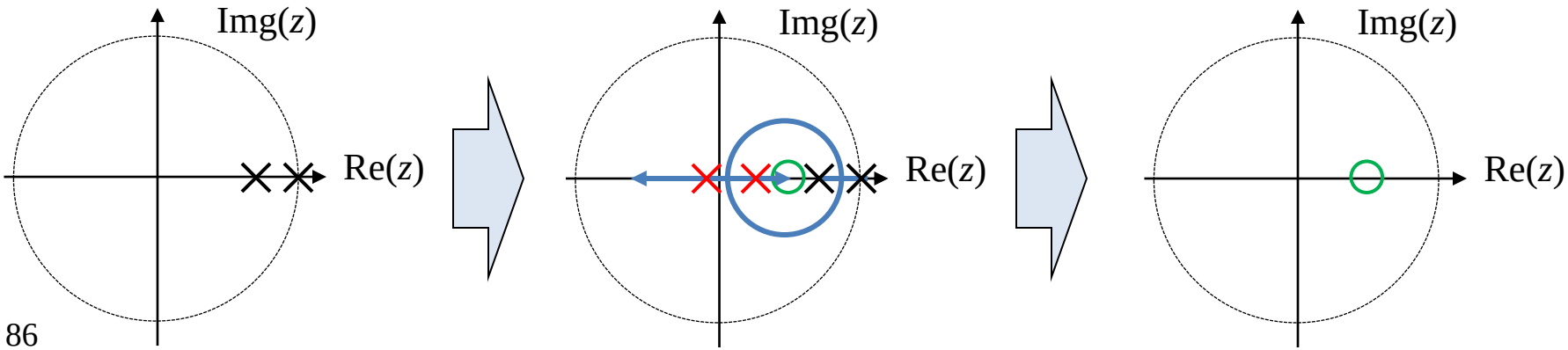


Emulation design process

- 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

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. Pub



Discrete design process

- Handy rules of thumb:
  - Sample rates can be as low as twice the system bandwidth (but 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

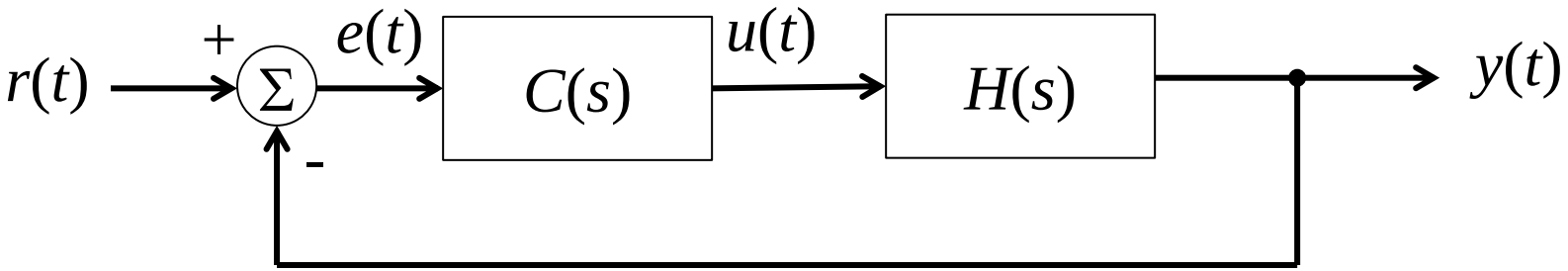
Practical Digital Control

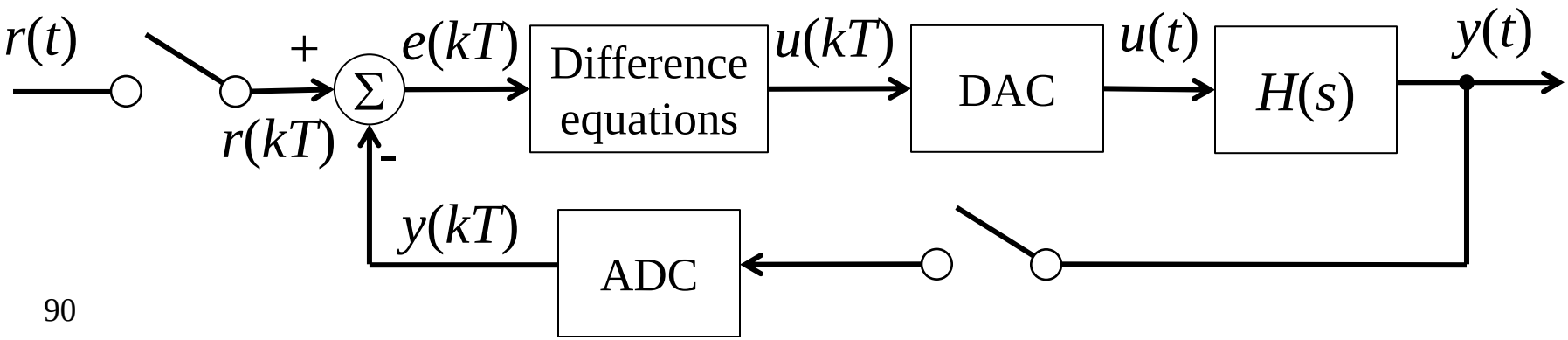
Practical design lolwut?

- Make a controller that works…
  - To specification
  - Reliably
  - Affordably

The digital approximation

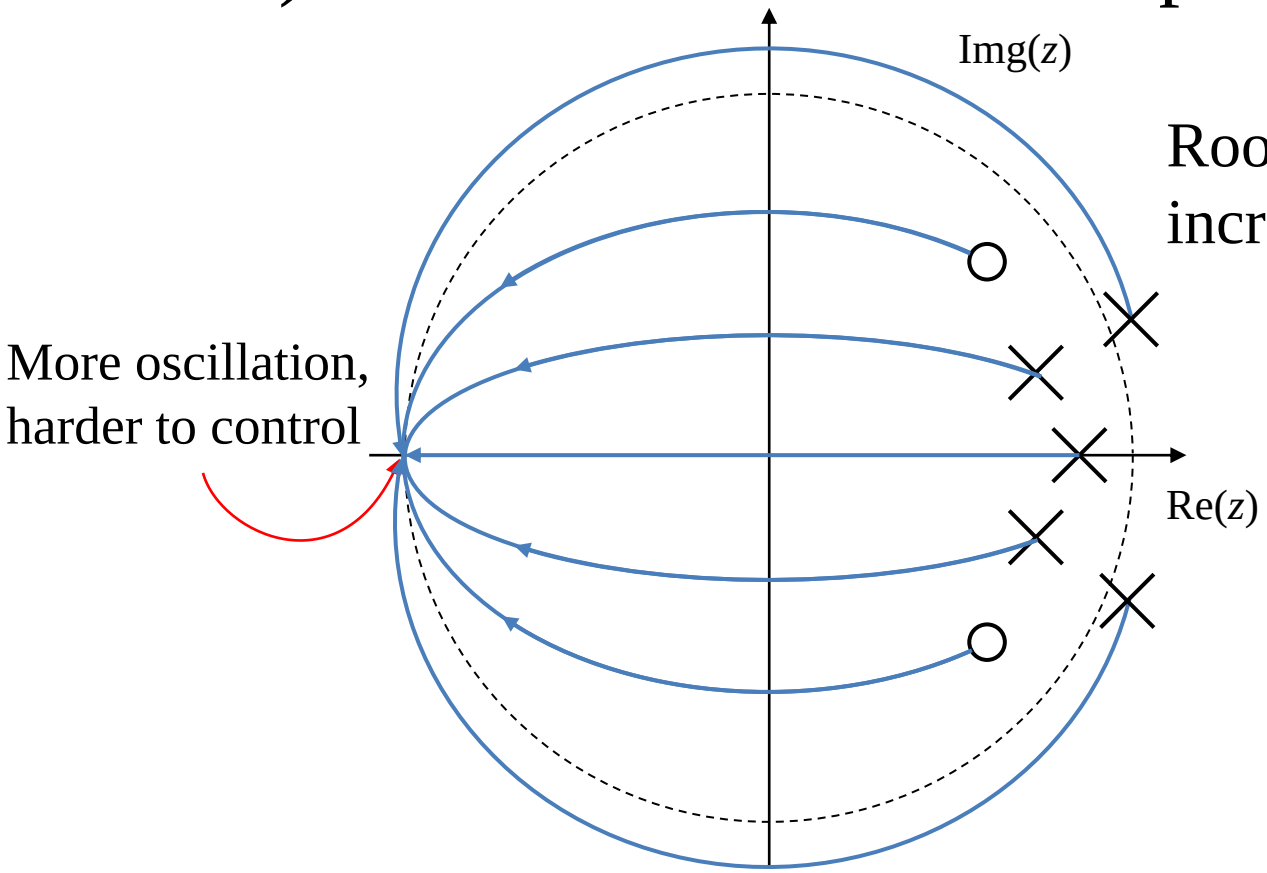
- Recall that digital controllers synthesize the response of a continuous system





Effect of changing *T*

- As *T* is increased, the location of poles (and zeros!) tracks isoclines of the z-plane



How slow can you go?

- Answer: it depends!
  - Simply increasing *T* cannot (on its own) destabilise a stable system
  - In practice, increasing *T* makes meeting arbitrary performance goals more difficult
  - A lower bound on *T* is twice the required input tracking bandwidth of the closed-loop system

How closely does your output need to match the continuous equivalent?

How slow can you go?

- Generally speaking, the faster you go, the better the digital approximation…
  - Closer mapping to the s-plane
  - Closer differentiator implementation
  - But fast hardware is more expensive – slower sample time gives more processing time between successive outputs

Also, why does small T put all the poles near the origin?

Causality

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!)

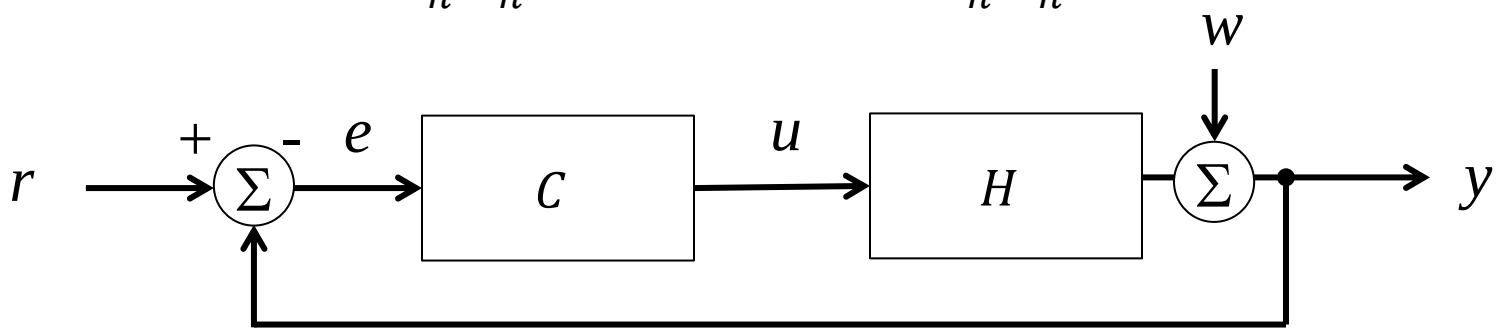
- Real systems always run with a delay:

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

Disturbance rejection

- Controllers must also reject disturbances introduced into the system
- Recall:

$$y = \frac{kC_nH_n}{kC_nH_n + CdHd} r + \frac{C_dH_d}{kC_nH_n} w$$



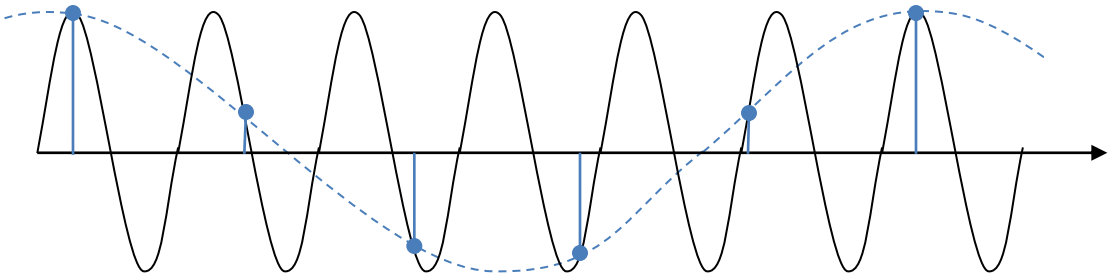
w is attenuated by high-frequency roll-off of $(CH)^{-1}$

Note on ‘improper’ controllers

- Systems with numerators of higher order than the denominator are ‘improper’
  - They are inherently non-causal
  - Eg. derivative control
- Problem for controllers: PID is improper
  - Fasts poles added to the controller can balance dynamic order, but can effect response/stability (and still assumes instant computation)
  - Better to use a velocity measurement if possible

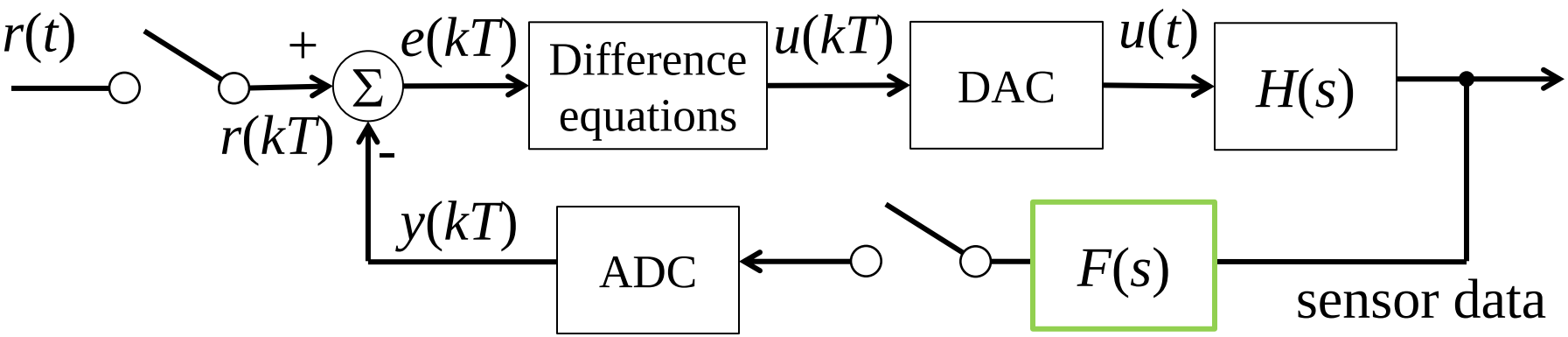
Noise and filtering

- What if you don't have a velocity sensor? What if you *have* to use a dirty derivative?
- HF noise is amplified by digital zeros
  - Ditto continuous controllers, but analog parts have non-idealities that cause roll-off
- However, sub-sampled noise can be aliased into the dynamic range of the system



Noise and filtering

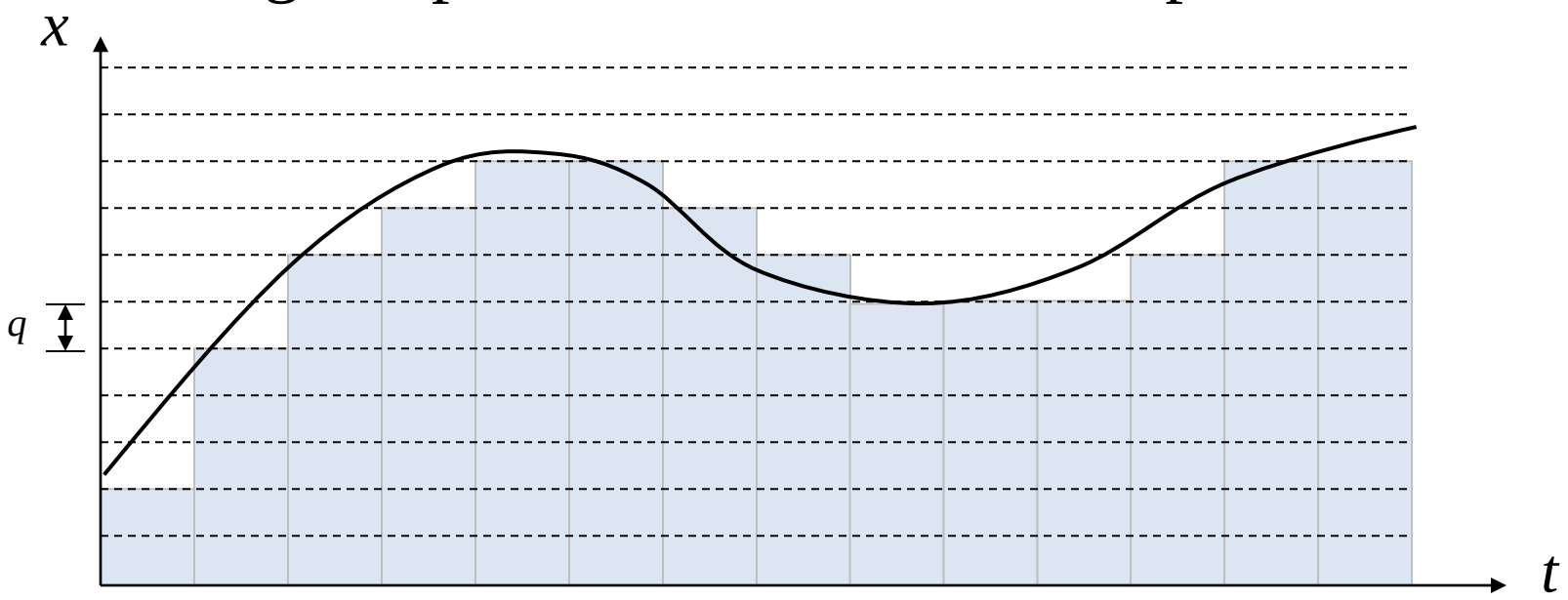
- Use an analog-prefilter:



- Configure the roll-off of $F(s)$ to be slightly below the sampling Nyquist frequency
  - Be careful – too much lower risks effecting the closed-loop system dynamics

Quantisation

- ADC measurement of voltages to digital bytes/words introduces quantisation
  - Higher resolution improves the approximation
  - Large steps can lead to non-linear phenomena



Quantisation

- Fortunately, it's easy to get good resolution:
  - 8-bit resolution: 0.4% accuracy
  - 12-bit resolution: 0.025% accuracy
  - 16-bit resolution: 1/1000th of a per cent
- Output quantisation can be smooth with an analog filter stage
  - Remember that filters induce delay and must be included with the plant for control design!

Numerical non-linearity

- Beware of non-linear effects in fixed-point arithmetic in calculations.
  - Integer division remainders are discarded
  - Words have a fixed maximum range
- Trade-off between precision and range:

```
u = x*(y/z); //higher dynamic range, but less accurate
due to rounding

u = (x*y)/z; //more accurate for small x and y, but
lower dynamic range from saturating (x*y) < word_size
```

Numerical non-linearity

- Code rounding imprecision can be treated like disturbance noise
  - Linear systems theory shows that noise is rejected by closed-loop control
- Dynamic range saturation produces a strong non-linearity and should be avoided!
  - Check the maximum possible value of variables
  - Test sanity bounds on variables just-in-case

Approximation is the enemy

- The limiting factor in discrete control is how closely the 'ideal' continuous response can be synthesised
- For best performance:
  - T^{-1} above 20x the required system bandwidth
  - Use high-order discretisation approximations or go direct to the z-transform
  - Reduce computational delay (approximate a quasi-causal system)

But approximation is your friend

- With good control design, extremely cheap and robust compensators can be built
- Often times control problems will be a consequence of improper filtering or insufficiently accurate approximation

Golden rule: Understand your system

</assessable>

Nothing beyond this point is on the exam.

Do not pay attention.

Do not attempt to learn.

Introduction to State-Space

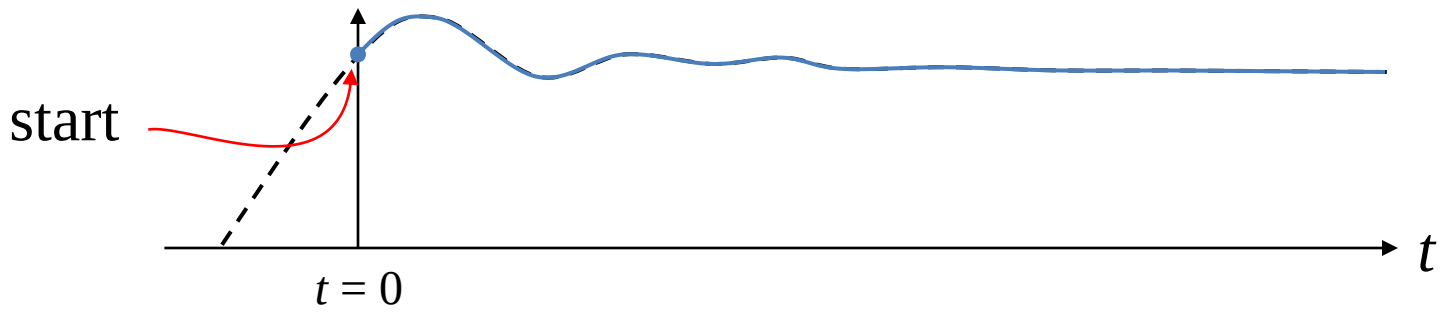
State-space lolwut?

- A ‘clean’ way of representing systems
- Easy implementation in matrix algebra
- Simplifies understanding Multi-Input-Multi-Output (MIMO) systems

Affairs of state

- Introductory brain-teaser:
  - If you have a dynamic system model with history (ie. integration) how do you represent the instantaneous state of the plant?

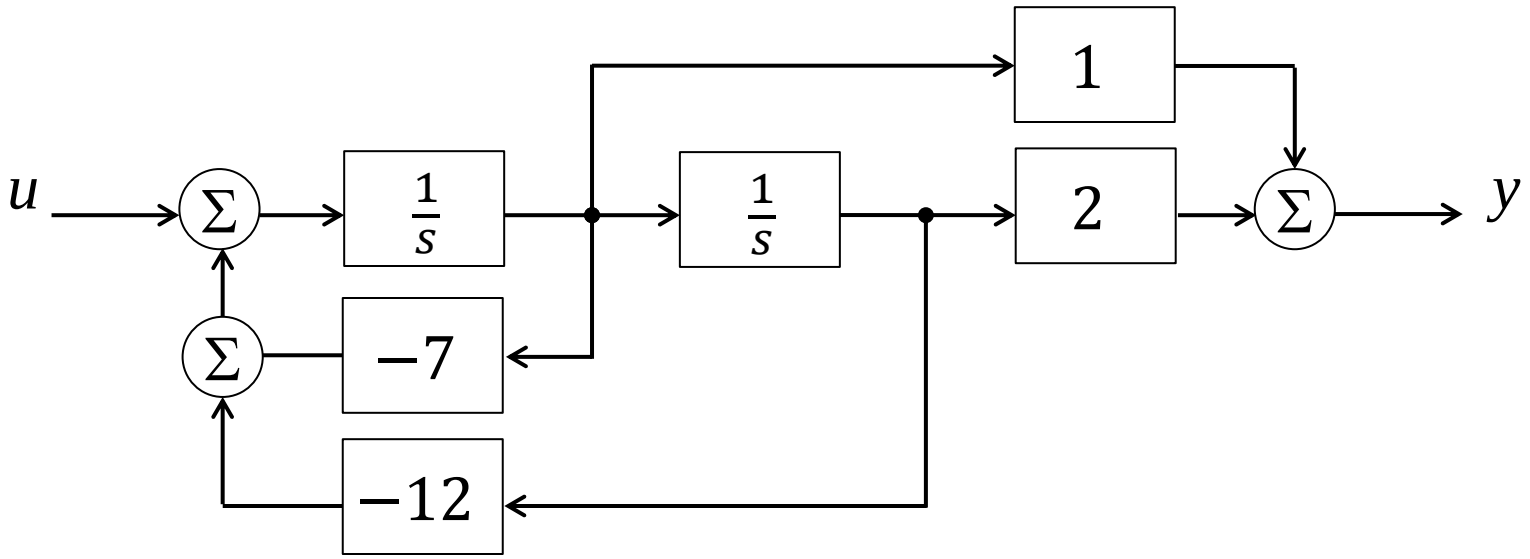
Eg. how would you setup a simulation of a step response, mid-step?



Introduction to state-space

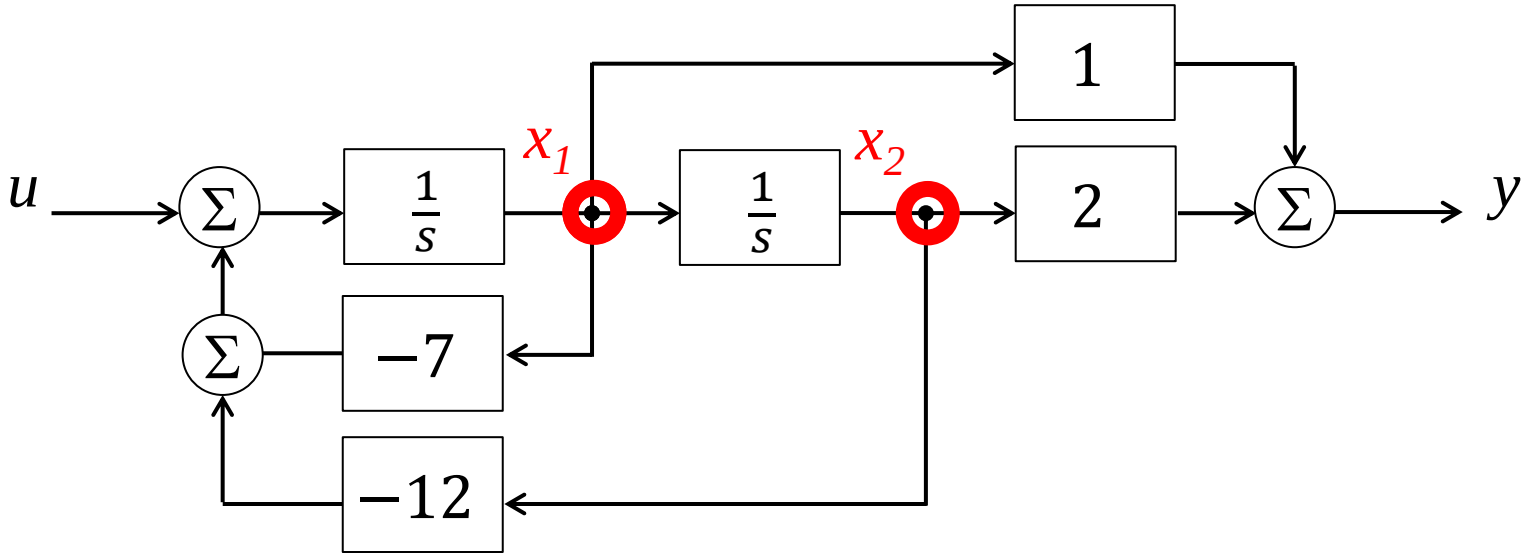
- Linear systems can be written as networks of simple dynamic elements:

$$H = \frac{s+2}{s^2+7s+12} = \frac{2}{s+4} + \frac{-1}{s+3}$$



Introduction to state-space

- We can identify the nodes in the system
  - These nodes contain the integrated time-history values of the system response
  - We call them “states”



Linear system equations

- We can represent the dynamic relationship between the states with a linear system:

$$\dot{x_1} = -7x_1 - 12x_2 + u$$
$$\dot{x_2} = x_1 + 0x_2 + 0u$$
$$y = x_1 + 2x_2 + 0u$$

State-space representation

- We can write linear systems in matrix form:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -7 & 12 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$\boldsymbol{y} = [1 \quad 2]\boldsymbol{x} + 0u$$

Or, more generally:

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}u$$
$$y = \mathbf{C}\boldsymbol{x} + Du$$

"State-space equations"

State-space representation

- State-space matrices are not necessarily a unique representation of a system
  - There are two common forms
- Control canonical form
  - Each node – each entry in $\boldsymbol{x}$ – represents a state of the system (each order of s maps to a state)
- Modal form
  - Diagonals of the state matrix **A** are the poles ("modes") of the transfer function

Control canonical form

- CCF matrix representations have the following structure:

$$\begin{bmatrix} -a_1 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} & -a_n \\ 1 & 0 & & 0 & 0 & 0 \\ 0 & 1 & & & & \\ \vdots & & \ddots & & & \vdots \\ & & & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

Pretty diagonal!

State variable transformation

- Important note!
  - The states of a control canonical form system are not the same as the modal states
  - They represent the same dynamics, and give the same output, but the vector values are different!
- However we can convert between them:
  - Consider state representations, $\boldsymbol{x}$ and $\boldsymbol{q}$ where

$$\boldsymbol{x} = \mathbf{T}\boldsymbol{q}$$

T is a “transformation matrix”

State variable transformation

- Two homologous representations:

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}u \qquad y = \mathbf{C}\boldsymbol{x} + Du$$

and

$$\dot{\boldsymbol{q}} = \mathbf{F}\boldsymbol{q} + \mathbf{G}u \qquad y = \mathbf{H}\boldsymbol{q} + Ju$$

We can write:

$$\dot{\boldsymbol{x}} = \mathbf{T}\dot{\boldsymbol{q}} = \mathbf{A}\mathbf{T}\boldsymbol{z} + \mathbf{B}u$$

$$\dot{\boldsymbol{q}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\boldsymbol{z} + \mathbf{T}^{-1}\mathbf{B}u$$

Therefore, $\mathbf{F} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ and $\mathbf{G} = \mathbf{T}^{-1}\mathbf{B}$

Similarly, $\mathbf{C} = \mathbf{H}\mathbf{T}$ and $D = J$

Controllability matrix

- To convert an arbitrary state representation in **F**, **G**, **H** and *J* to control canonical form **A**, **B**, **C** and *D*, the "controllability matrix"

$$\mathcal{C} = [\mathbf{G} \quad \mathbf{FG} \quad \mathbf{F}^2\mathbf{G} \quad \cdots \quad \mathbf{F}^{n-1}\mathbf{G}]$$

must be nonsingular.

>deep think<

Why is it called the "controllability" matrix?

Controllability matrix

- If you can write it in CCF, then the system equations must be linearly independent.
- Transformation by any nonsingular matrix preserves the controllability of the system.
- Thus, a nonsingular controllability matrix means $\boldsymbol{x}$ can be driven to any value.

Kind of awesome

- The controllability of a system depends on the particular set of states you chose
- You can't tell just from a transfer function whether all the states of $\boldsymbol{x}$ are controllable
- The poles of the system are the Eigenvalues of $\mathbf{F}$, (p_i).

State evolution

- Consider the system matrix relation:

$$\dot{\boldsymbol{x}} = \mathbf{F}\boldsymbol{x} + \mathbf{G}u$$
$$y = \mathbf{H}\boldsymbol{x} + Ju$$

The time solution of this system is:

$$\boldsymbol{x}(t) = e^{\mathbf{F}(t-t_0)}\,\boldsymbol{x}(t_0) + \int_{t_0}^{t} = e^{\mathbf{F}(t-\tau)}\,\mathbf{G}u(\tau)d\tau$$

If you didn't know, the matrix exponential is:

$$e^{\mathbf{K}t} = \mathbf{I} + \mathbf{K}t + \frac{1}{2!}\mathbf{K}^2t^2 + \frac{1}{3!}\mathbf{K}^3t^3 + \cdots$$

Stability

- We can solve for the natural response to initial conditions $\boldsymbol{x_0}$:

$$\boldsymbol{x}(t) = e^{p_i t}\boldsymbol{x}_0$$

$$\therefore \dot{\boldsymbol{x}}(t) = p_i e^{p_i t}\boldsymbol{x}_0 = \mathbf{F}e^{p_i t}\boldsymbol{x}_0$$

Clearly, a system will be stable provided

$$\text{eig}(\mathbf{F}) < 0$$

Characteristic polynomial

- From this, we can see $\mathbf{F}\boldsymbol{x}_0 = p_i\boldsymbol{x}_0$

$$\text{or, } (p_i\mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$$

$$\text{which is true only when } \det(p_i\mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$$

Aka. the characteristic equation!

- We can reconstruct the CP in s by writing:

$$\det(s\mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$$

Great, so how about control?

- Given $\dot{\boldsymbol{x}} = \mathbf{F}\boldsymbol{x} + \mathbf{G}u$, if we know $\mathbf{F}$ and $\mathbf{G}$, we can design a controller $u = -\mathbf{K}\boldsymbol{x}$ such that

$$\text{eig}(\mathbf{F} - \mathbf{GK}) < 0$$

- In fact, if we have full measurement and control of the states of $\boldsymbol{x}$, we can position the poles of the system in arbitrary locations!

Of course, that never happens in reality.

Example: PID control

- Consider a system parameterised by three states: x_1, x_2, x_3 where $x_2 = \dot{x}_1$ and $x_3 = \dot{x}_2$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \boldsymbol{x} - \mathbf{K}u$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} + 0u$$

x_2 is the output state of the system; x_1 is the value of the integral; x_3 is the velocity.

- We can choose **K** to move the eigenvalues of the system as desired:

$$\det\begin{bmatrix} 1-K_1 & & \\ & 1-K_2 & \\ & & -2-K_3 \end{bmatrix} = \mathbf{0}$$

All of these eigenvalues must be positive.

It's straightforward to see how adding derivative gain K_3 can stabilise the system.

Discrete State-Space

Discretisation FTW!

- We can use the time-domain representation to produce difference equations!

$$\boldsymbol{x}(kT+T) = e^{\mathbf{F}T}\,\boldsymbol{x}(kT) + \int_{kT}^{kT+T} e^{\mathbf{F}(kT+T-\tau)}\mathbf{G}u(\tau)d\tau$$

Notice $\boldsymbol{u}(\tau)$ is not based on a discrete ZOH input, but rather an integrated time-series.

We can structure this by using the form:

$$u(\tau) = u(kT), \qquad kT \le \tau \le kT+T$$

Discretisation FTW!

- Put this in the form of a new variable:

$$\eta = kT + T - \tau$$

Then:

$$\boldsymbol{x}(kT+T) = e^{\boldsymbol{F}T}\boldsymbol{x}(kT) + \left(\int_{kT}^{kT+T} e^{\boldsymbol{F}\eta}\,d\eta\right)\boldsymbol{G}u(kT)$$

Let's rename $\boldsymbol{\Phi} = e^{\boldsymbol{F}T}$ and $\boldsymbol{\Gamma} = \left(\int_{kT}^{kT+T} e^{\boldsymbol{F}\eta}\,d\eta\right)\boldsymbol{G}$

Discrete state matrices

So,

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Gamma}u(k)$$
$$y(k) = \mathbf{H}\boldsymbol{x}(k) + \mathbf{J}\boldsymbol{u}(k)$$

Again, $\boldsymbol{x}(k+1)$ is shorthand for $\boldsymbol{x}(kT+T)$

Note that we can also write $\boldsymbol{\Phi}$ as:

$$\boldsymbol{\Phi} = \mathbf{I} + \mathbf{F}T\boldsymbol{\Psi}$$

where

$$\boldsymbol{\Psi} = \mathbf{I} + \frac{\mathbf{F}T}{2!} + \frac{\mathbf{F}^2T^2}{3!} + \cdots$$

Simplifying calculation

- We can also use $\boldsymbol{\Psi}$ to calculate $\boldsymbol{\Gamma}$
  - Note that:

$$\Gamma = \sum_{k=0}^{\infty} \frac{\mathbf{F}^k T^k}{(k+1)!} T\mathbf{G}$$
$$= \boldsymbol{\Psi} T \mathbf{G}$$

$\boldsymbol{\Psi}$ itself can be evaluated with the series:

$$\boldsymbol{\Psi} \cong \mathbf{I} + \frac{\mathbf{F}T}{2}\left\{\mathbf{I} + \frac{\mathbf{F}T}{3}\left[\mathbf{I} + \cdots \frac{\mathbf{F}T}{n-1}\left(\mathbf{I} + \frac{\mathbf{F}T}{n}\right)\right]\right\}$$

State-space z-transform

We can apply the z-transform to our system:

$$(z\mathbf{I} - \mathbf{\Phi})\boldsymbol{X}(z) = \mathbf{\Gamma}U(k)$$

$$Y(z) = \mathbf{H}\boldsymbol{X}(z)$$

which yields the transfer function:

$$\frac{Y(z)}{\boldsymbol{X}(z)} = G(z) = \mathbf{H}(z\mathbf{I} - \mathbf{\Phi})^{-\mathbf{1}}\mathbf{\Gamma}$$

State-space control design

- Design for discrete state-space systems is just like the continuous case.
  - Apply linear state-variable feedback:

$$u = -\mathbf{K}\boldsymbol{x}$$

such that $\det(z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K}) = \alpha_c(z)$

where $\alpha_c(z)$ is the desired control characteristic equation

This requires the system controllability matrix

$\mathcal{C} = [\mathbf{\Gamma} \quad \mathbf{\Phi}\mathbf{\Gamma} \quad \mathbf{\Phi}^2\mathbf{\Gamma} \quad \cdots \quad \mathbf{\Phi}^{n-1}\mathbf{\Gamma}]$ to be full-rank.