Digital Design Practice

or

Why doesn't my compensator work?

Paul Pounds

21 May 2012

University of Queensland

Previously on Digital Controls…

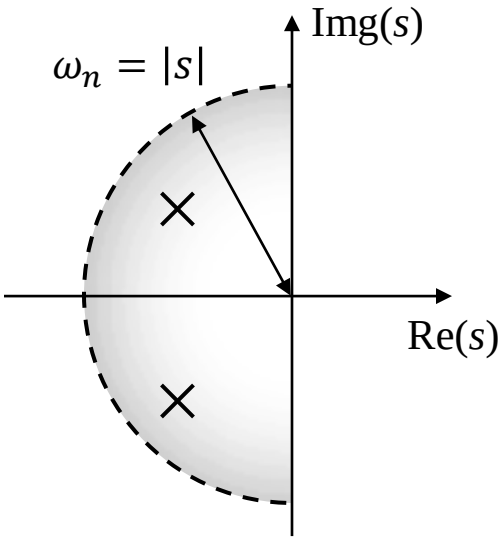
- We reviewed the most common control designs: lead, lag and PID
- We saw the two broad approaches of designing digital compensators: emulation and direct design
- We introduced three approximation techniques: Tustin's method, Matched Pole-Zero and Modified Matched Pole-Zero

Practical design

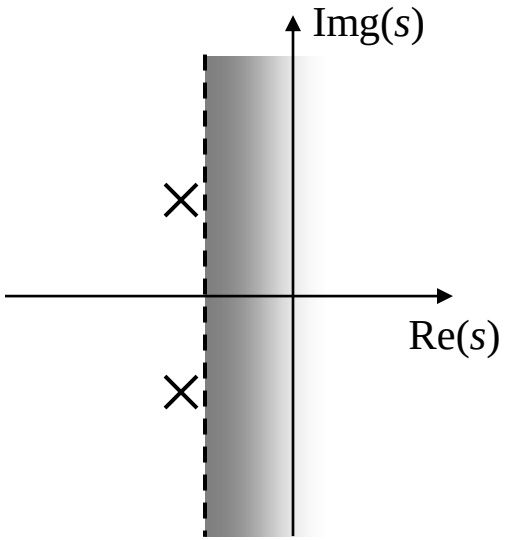
- What do I mean by ‘practical’?
  - Ie. works in practice, not just in Matlab
- Make a controller that works…
  - To specification
  - Reliably
  - Affordably

Specification bounds

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

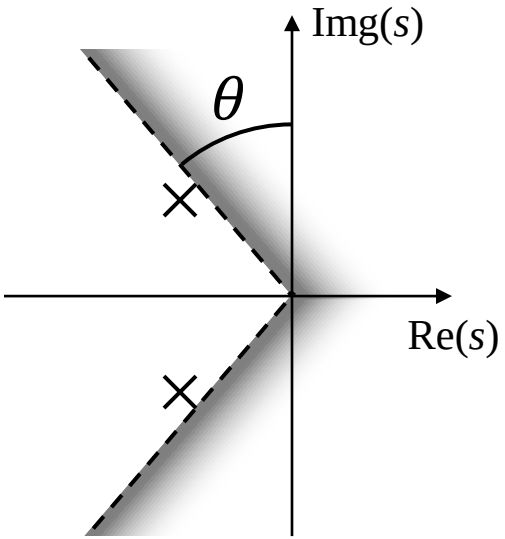


$$|s| = \frac{1.8}{t_r}$$



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

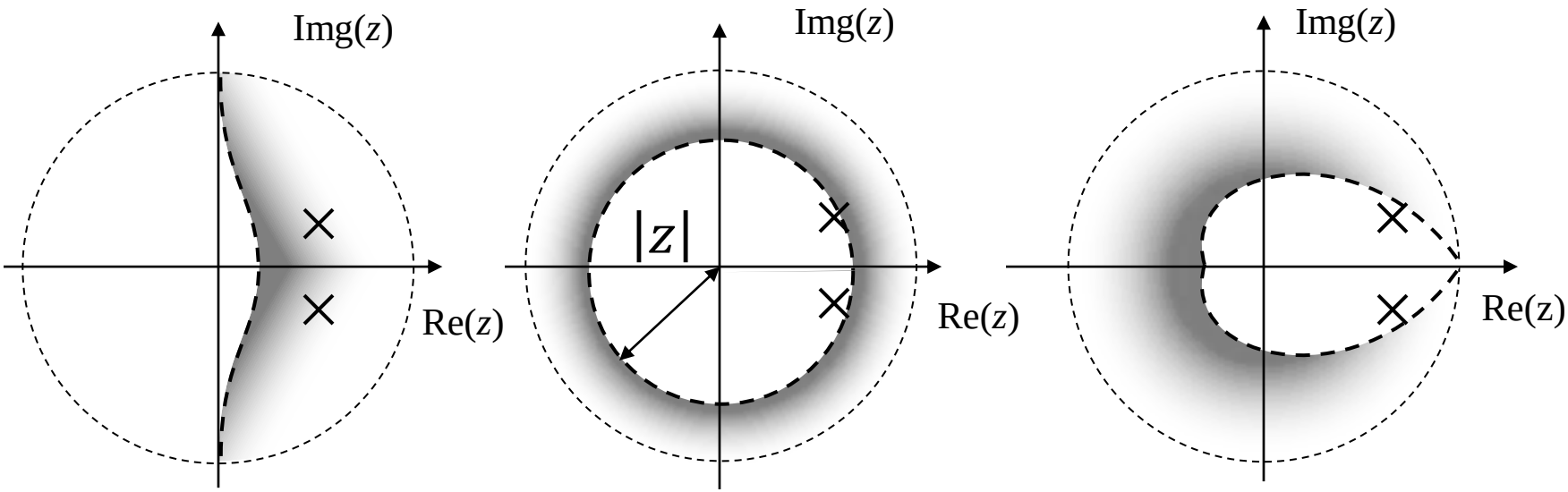
$s = \sigma$



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

Discrete bounds

- These map to the discrete domain:

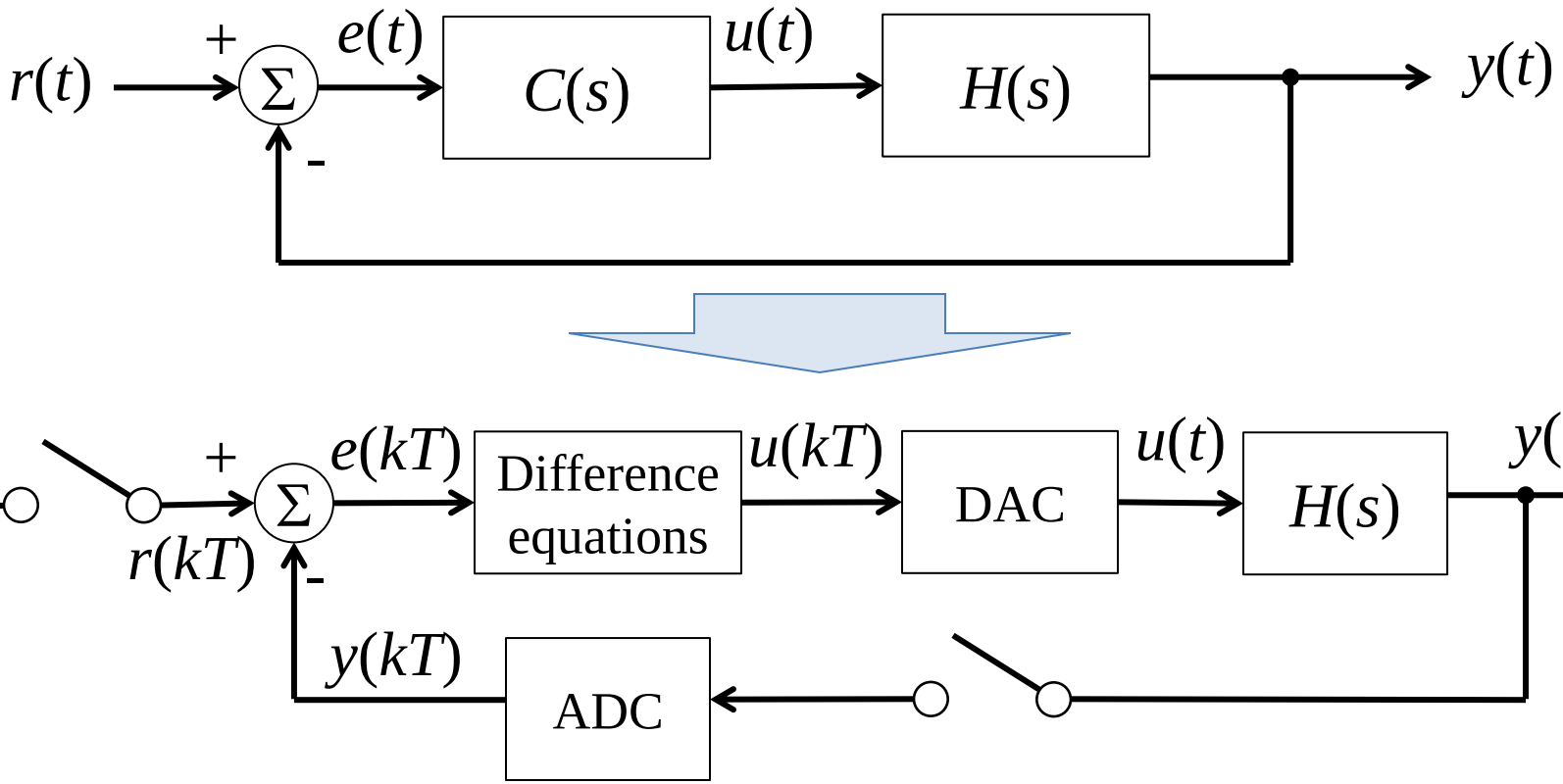


$$|z| = e^{-t_s T}$$

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

The digital approximation

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



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

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

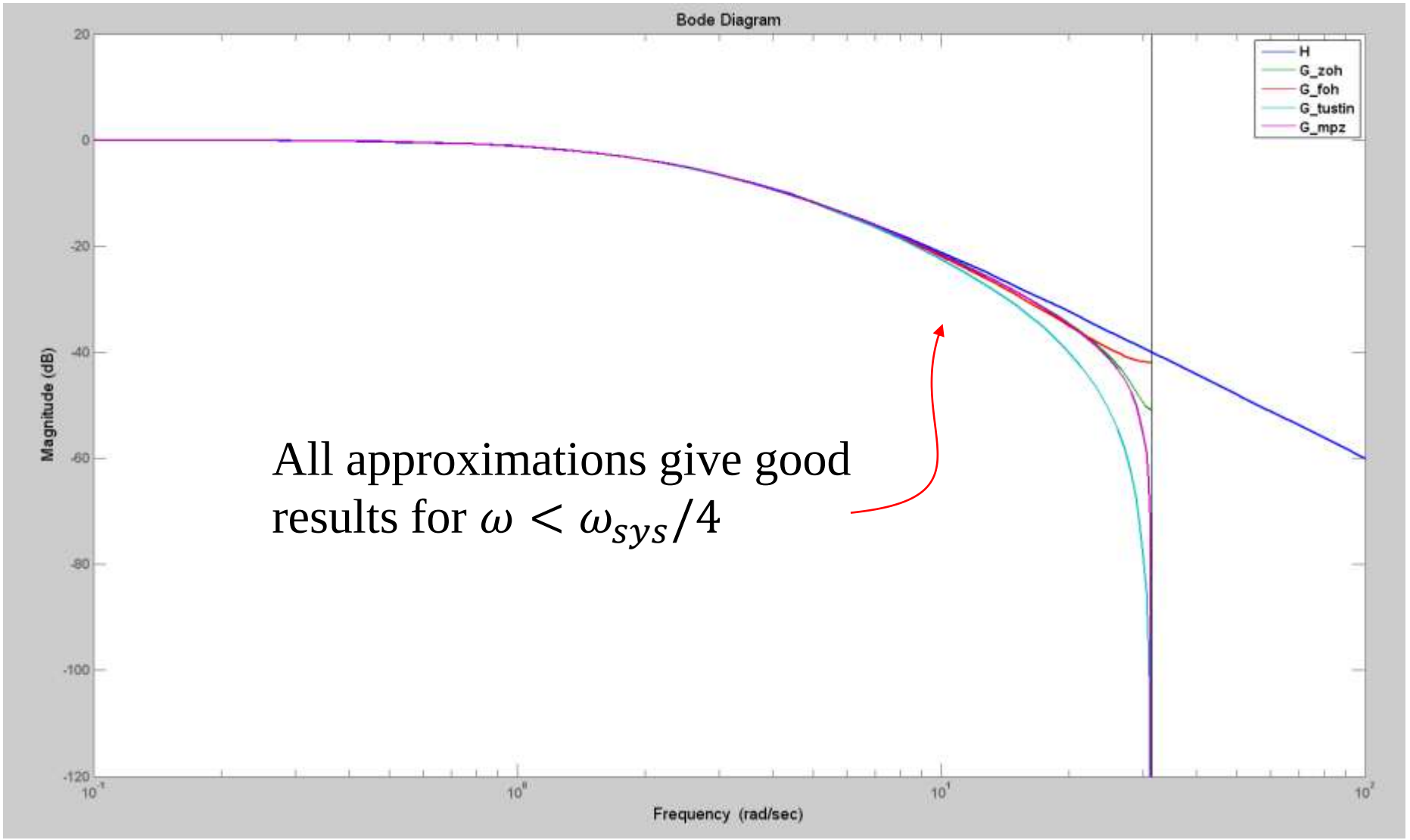
sampled at 10 Hz

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

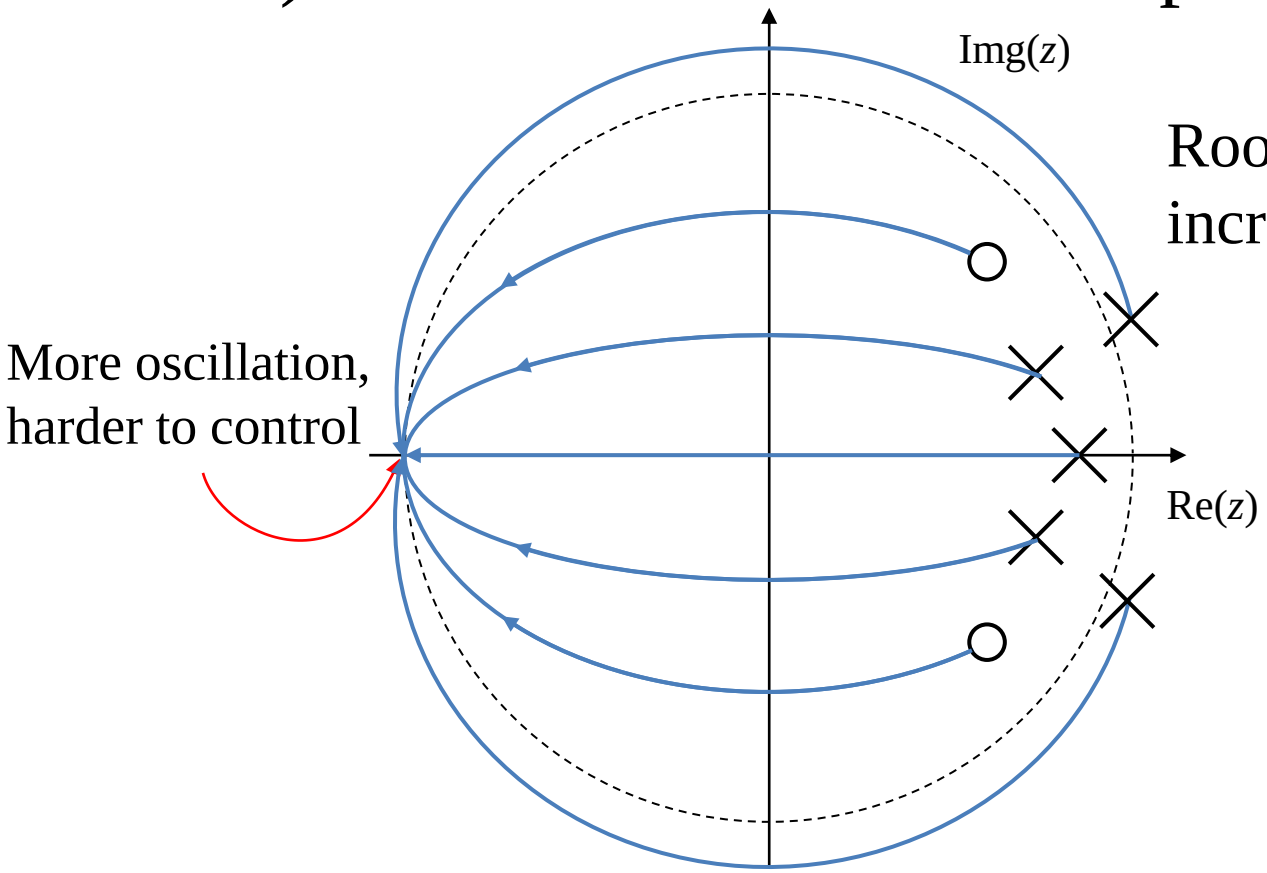
- Recall the intrinsic time-delay associated with ZOH? What about that?
- This is the discrete domain; we can model the delay exactly:

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

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

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

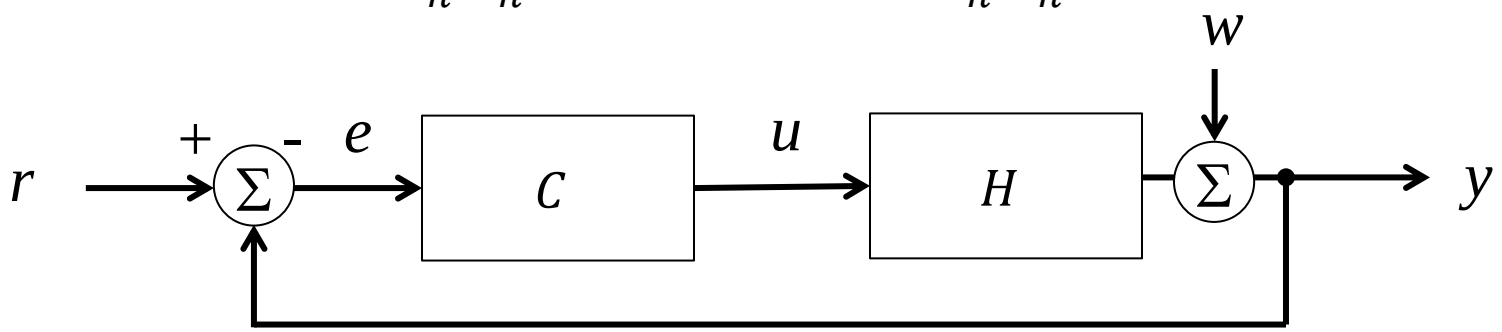
Also, why does small T put all the poles near the origin?

(hard to control for high-order systems)

Disturbance rejection

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

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



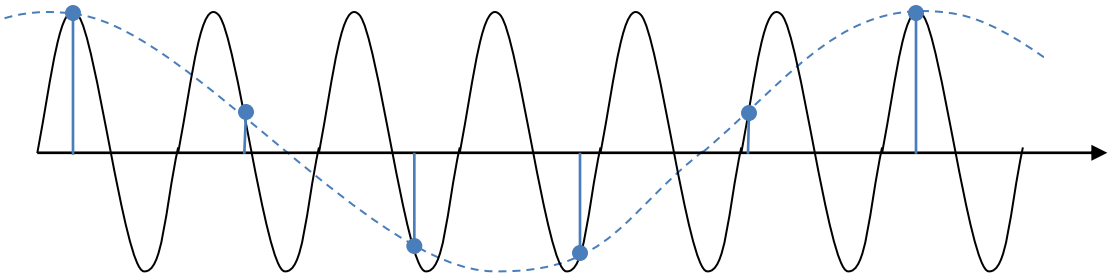
w is attenuated by high-frequency roll-off of $(CH)^{-1}$

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

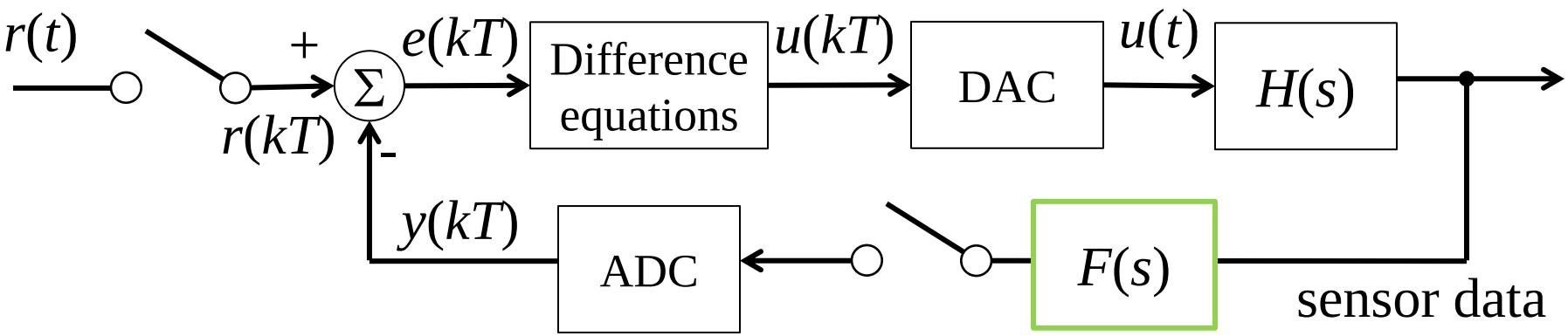
Noise and filtering

- 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

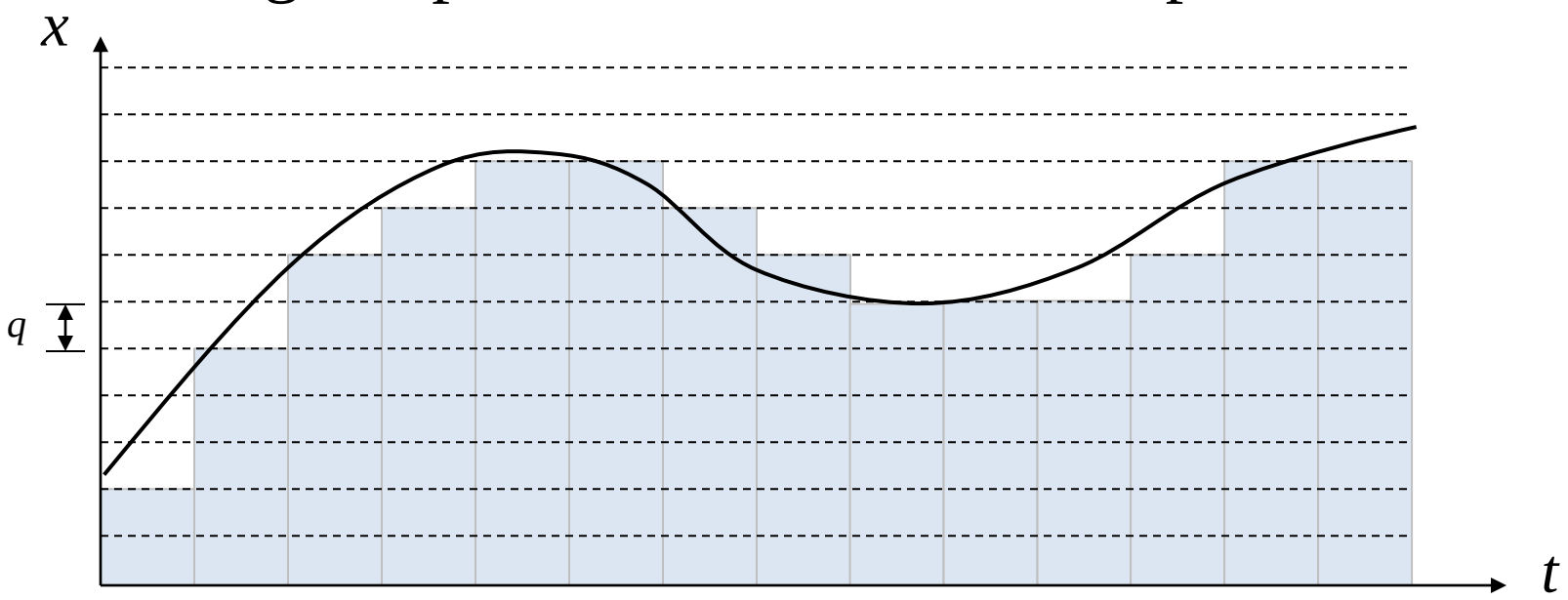
- Use an analog-prefilter:



- 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

Discrete state-space

And now for something completely different...

Recall state-space

- We can represent any strictly proper linear system as a 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:

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

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

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

This requires the system controllability matrix

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

Ask Surya about this ☺

Why?

- Why am I telling you this?
  - It's how many controllers are really designed
  - It's provides insight into the relationship between continuous and discrete systems
  - It's really cool!

But most importantly, it's needed for…

Tune-in next time for…

Estimation and Kalman Filtering

starring

Surya Singh!

Fun fact: Digital control systems **hate** all living things

Get the latest slides!

- These slides are available on the website
  - They are periodically updated as I find typos
  - They contain all you 'need' to know
- But control theory is awesome!*
  - A copy of Franklin, Powell and Emami-Naeini is guaranteed to help you understand control
  - Also check out "Modern Control Engineering" by Katsuhiko Ogata

*Level of awesome you experience may vary

</assessable>

Nothing beyond this point is on the exam.

Do not pay attention.

Do not attempt to learn.

Runga-Kutta just cus'

- Runga-Kutta methods are a family of numerical solution methods for ODEs
  - The fourth-order RK method is very common; called RK4, or often just "Runga-Kutta":

For $\dot{y} = f(y,t)$:

$$y(t+\Delta t) = y(t) + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$k_1 = \Delta t f(t, y(t))$$
$$k_2 = \Delta t f\left(t + \tfrac{1}{2}\Delta t, y(t) + \tfrac{1}{2}k_1\right)$$
$$k_3 = \Delta t f\left(t + \tfrac{1}{2}\Delta t, y(t) + \tfrac{1}{2}k_2\right)$$
$$k_4 = \Delta t f(t + \Delta t, y(t) + k_3)$$

Runga-Kutta just cus'

- The 4^{th} order RK is an exact solution for 4^{th} order mechanics problems like the elastica
  - There are n^{th} order RK solutions to suit
- RK is iterative; this is useful for non-linear dynamic system simulation, where the form of the solution is not known *a priori*
- Matlab has a host of tools for RK, as well as a variety of other ODE solvers

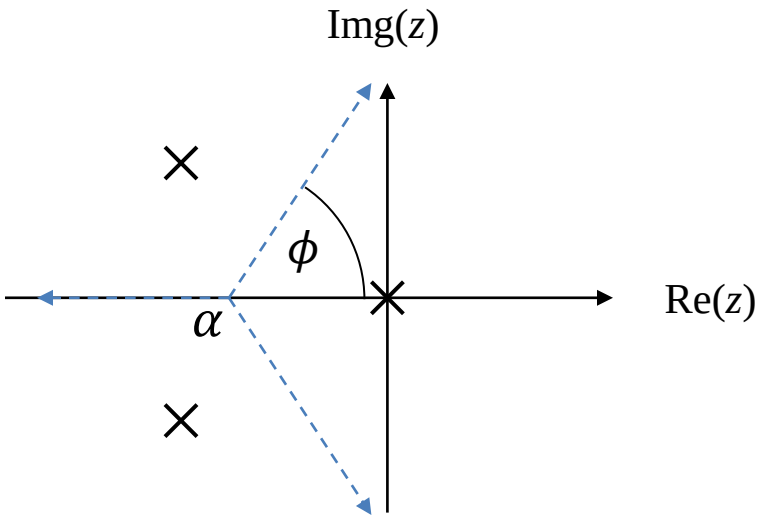
Careers are built on coming up with better ODE solvers and numerical methods

Quick ‘n dirty guide to root loci

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

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

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



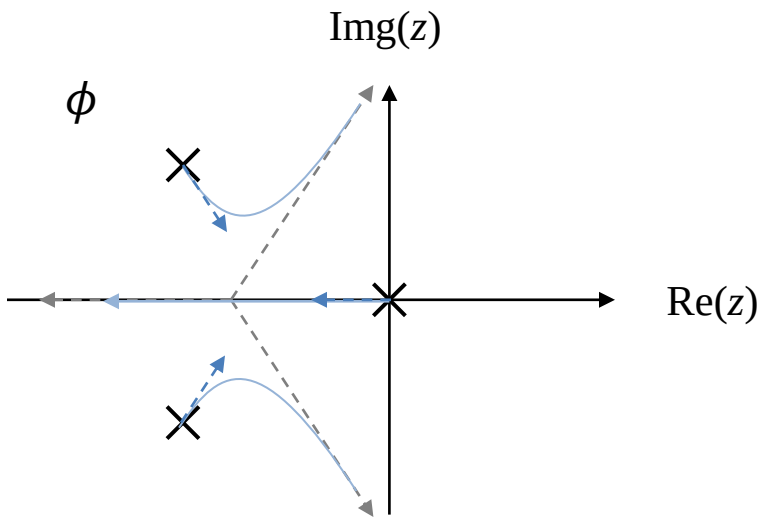
Quick ‘n dirty guide to root loci

- Pole departure/zero arrival angles given by:

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

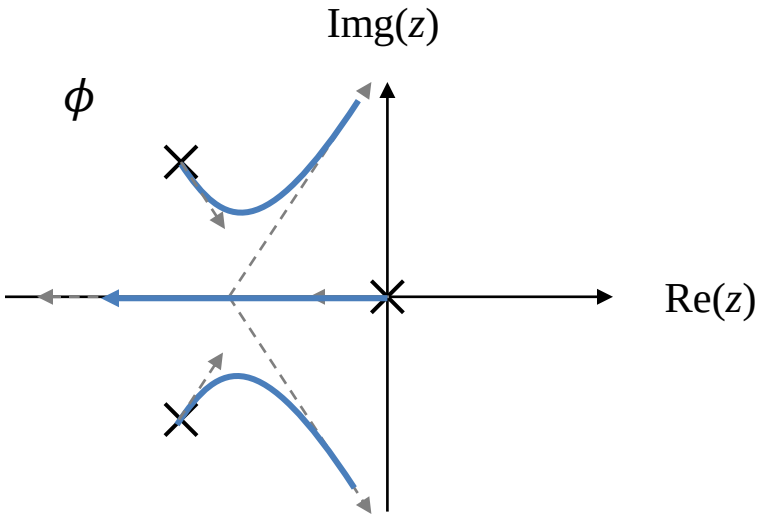
$$q\psi_{\text{arr}} = \sum\phi_i - \sum\psi_i + \pi + 2\pi l$$

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



Useful Matlab commands

- Construct symbolic elemental transforms

  ```
  s = tf('s'); z = tf('z',T)
  s = k/(s+a); z = k(z+a); etc
  ```

- Discretise a continuous plant

  ```
  c2d{H,T,'method'}
  ```

  where method = {'zoh','foh','tustin','matched'…}

- Write out poles or zeros of the system

  ```
  pole(H); zero(H)
  ```

- Plot poles/zeros on the s and z planes

  ```
  pzmap(H)
  ```

- Plot system bode:

  ```
  bode(H); bodemag(H)
  ```

- Built-in design tool:

  ```
  rltool(H); sisotool(H)
  ```