Introduction to Feedback Control

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

Lecture 15

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

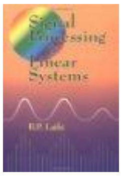
May 2, 2017

http://elec3004.com

Lecture Schedule:

Week	Date	Lecture Title
1	28-Feb	Introduction
	2-Mar	Systems Overview
2	7-Mar	Systems as Maps & Signals as Vectors
	9-Mar	Systems: Linear Differential Systems
3	14-Mar	Sampling Theory & Data Acquisition
	16-Mar	Aliasing & Antialiasing
4	21-Mar	Discrete Time Analysis & Z-Transform
	23-Mar	Second Order LTID (& Convolution Review)
5	28-Mar	Frequency Response
	30-Mar	Filter Analysis
6	4-Apr	Digital Filters (IIR) & Filter Analysis
	6-Apr	Digital Filter (FIR)
7	11-Apr	Digital Windows
	13-Apr	FFT
	18-Apr	Holiday
	20-Apr	
	25-Apr	
8	27-Apr	Active Filters & Estimation
9	2-May	Introduction to Feedback Control
	4-May	Servoregulation/PID
10	9-May	Introduction to (Digital) Control
	11-May	Digitial Control
11	16-May	Digital Control Design
	18-May	Stability
12	23-May	Digital Control Systems: Shaping the Dynamic Response
	25-May	Applications in Industry
13	30-May	System Identification & Information Theory
	1-Jun	Summary and Course Review

Follow Along Reading:



B. P. Lathi
Signal processing and linear systems
1998
TK5102.9.L38 1998

G. Franklin,
J. Powell,
M. Workman
Digital Control of Dynamic Systems
1990

TJ216.F72 1990
[Available as UQ Ebook]

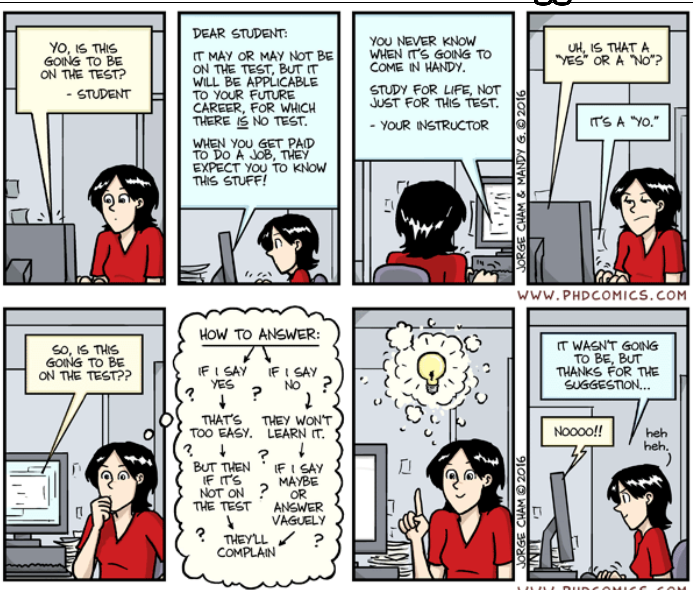
Today

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

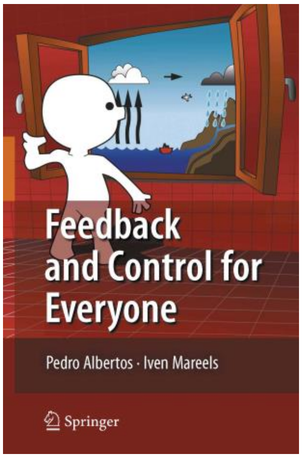
- FPW
  - Chapter 3: Sampled-Data Systems
  - Chapter 2: Linear, Discrete, Dynamic-Systems Analysis

Next Time

Announcements: "Thanks for the Suggestion" ☺



Announcements: Controls Resources



- Free e-Book [via UQ]:
  https://library.uq.edu.au/record=b2508512~S7

Lab 4 – LeviLab II:
[AKA “Revenge of the Tuning!”]

- *“not a long time ago in a lab down, down the way”*

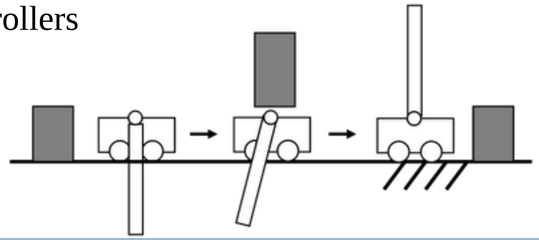
Introducing Feedback

Control

Once upon a time…

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

- Humans developed sophisticated tools for designing reliable analog controllers



Control

Once upon a time…

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

➔Idea: Digital computers in real-time control
  - Transform approach (classical control)
    - Root-locus methods (pretty much the same as METR 3200)
    - Bode's frequency response methods (these change compared to METR 3200)
  - State-space approach (modern control)

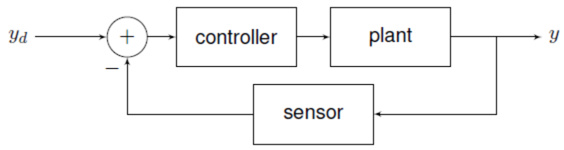
→ Model Making: Control of frequency response as well as Least Squares Parameter Estimation

Many advantages

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

Feedback Control

(Simple) control systems have three parts:

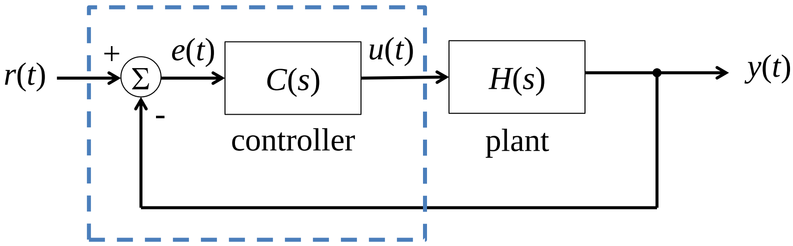


- The plant is the system to be controlled (e.g. the robot).
- The sensor measures the output of the plant.
- The controller sends an input command to the plant based on the difference from the actual output and the desired output.

Digital Control

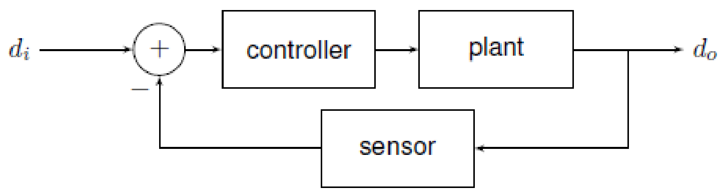
Archetypical control system

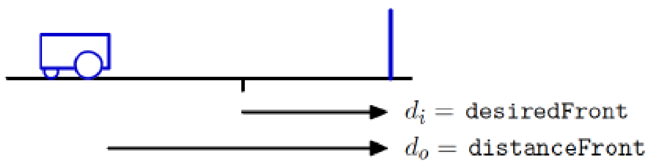
- Consider a continuous control system:



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

Simple Controller Goes Digital



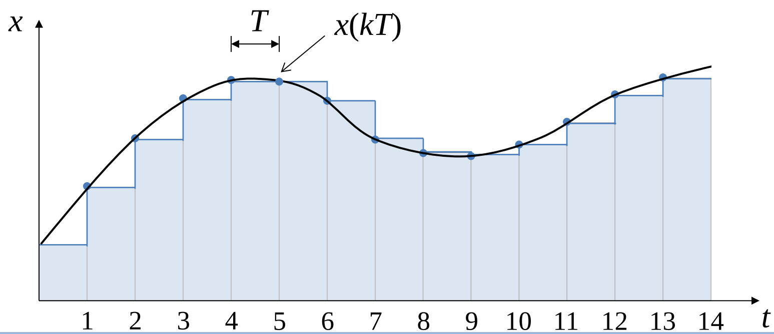


$$\begin{aligned}\text{plant:} \quad & y[n] = y[n-1] - Tu[n-1] \\ \text{sensor:} \quad & y[n] = u[n-1] \\ \text{controller:} \quad & y[n] = Ku[n]\end{aligned}$$

Complex system behaviors, depending on K

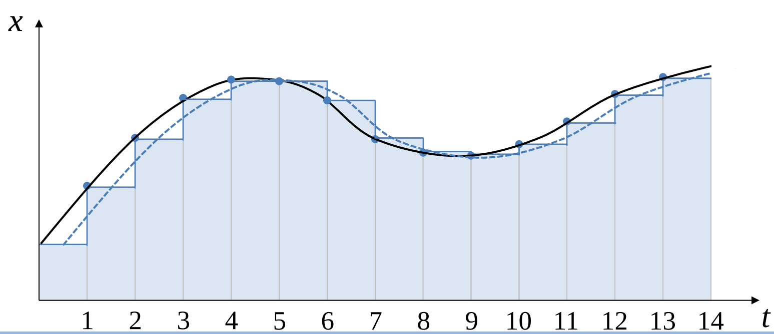
Return to the discrete domain

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



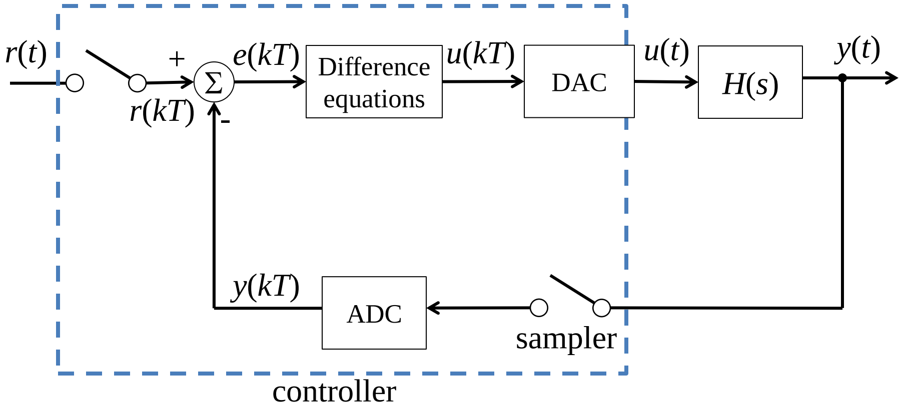
Zero Order Hold

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



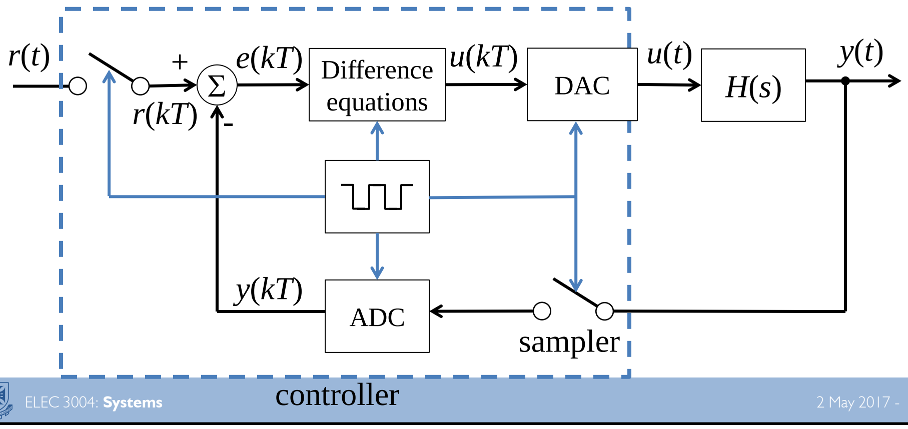
Digitisation

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

➔ Discrete-time transfer function

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

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

yields

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

solve for $X(z)$ to get

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

(note extra z in first term!)

hence

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

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

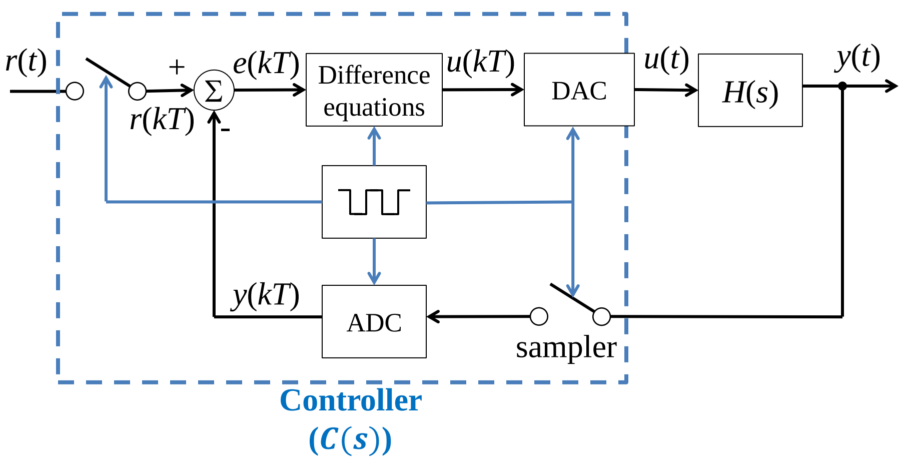
note power series expansion of resolvent:

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

Source: Boyd, Lecture Notes for EE263, 13-39

Modelling

(Digital) **Feedback Control**



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

C(s): PID = Control for the PID-dly minded

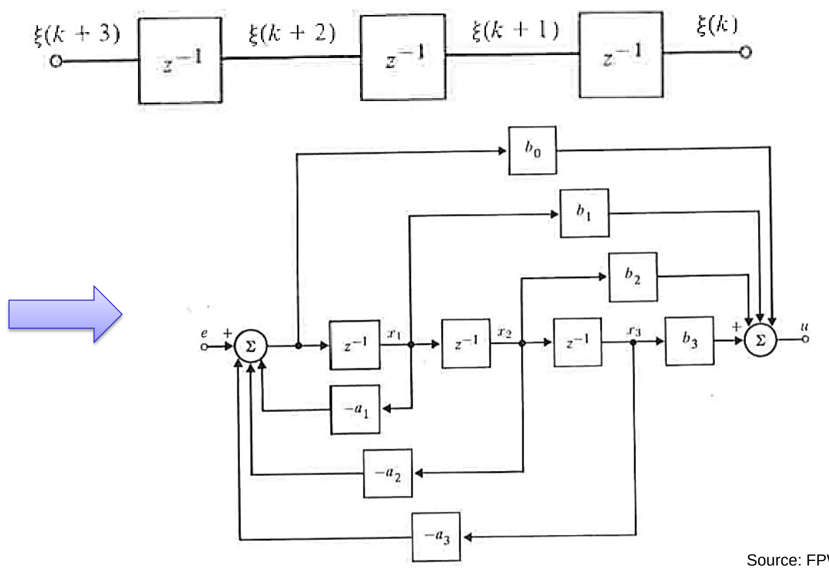
- Proportional-Integral-Derivative control is the control engineer's hammer*
  - For P,PI,PD, etc. just remove one or more terms

$$C(s) = k\left(1 + \frac{1}{\tau i s} + \tau d s\right)$$

Proportional
Integral
Derivative

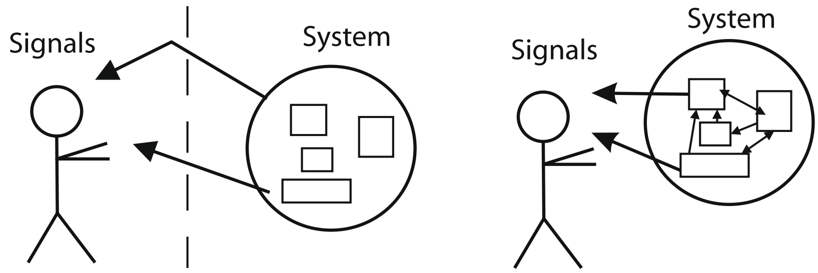
*Everything is a nail. That's why it's called "Bang-Bang" Control ☺

Feedback Control: Tuning Nightmare!



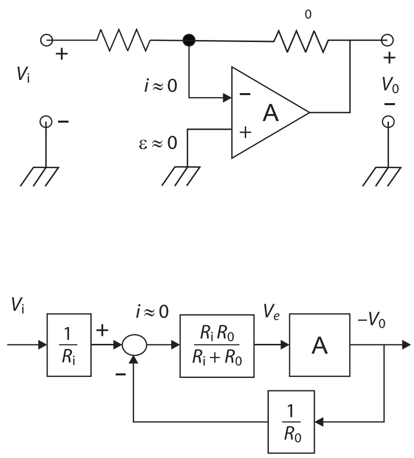
Source: FPW, Fig. 2.8

Signals and Systems: Modelling Tools!



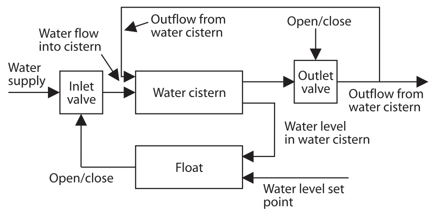
- Signals: Often came from a system
- Now we want feedback tools so as to understand the structure of the systems and how they interact so as to get desired signals

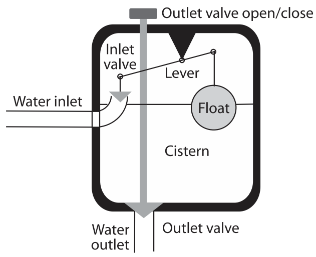
Feedback Control + Models: Everywhere

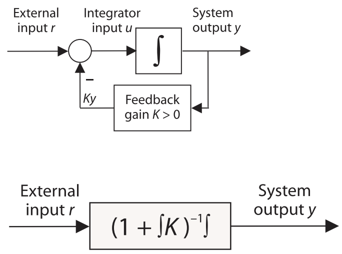


- $\frac{V_0}{V_1} = -\frac{R_0}{R_1}\frac{1}{1+\frac{1}{A}\left(1+\frac{R_0}{R_1}\right)}$

Feedback Control + Models: Everywhere





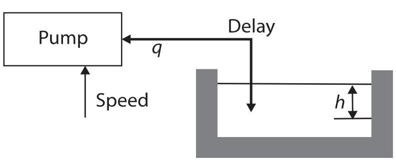


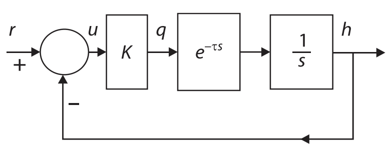
- Integrator:

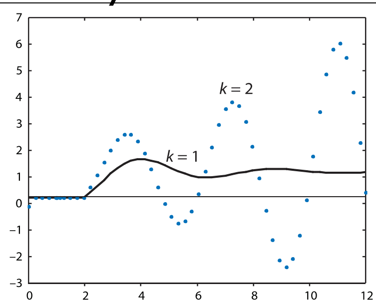
$$y(t) = y_0 + \int_0^t u(\tau)d\tau$$

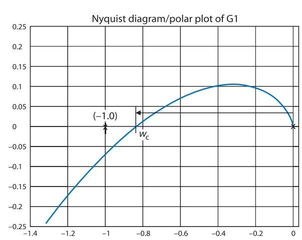
$$y(t) = y_0 + \frac{A\sin(\omega t)}{\omega}$$

Feedback Control + Models: Everywhere



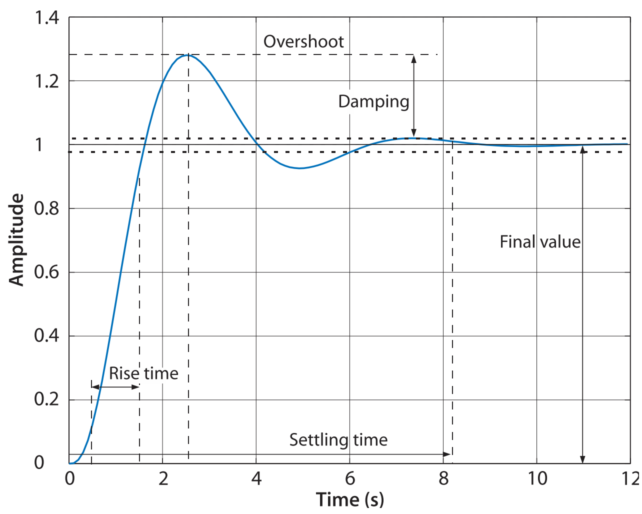




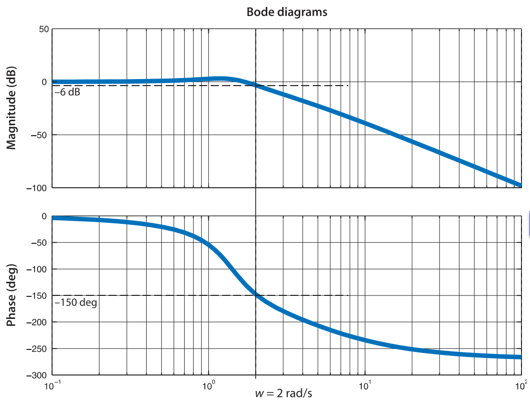


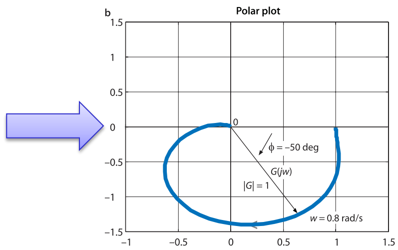
Feedback as a Filter

Time Response



Frequency Domain Analysis



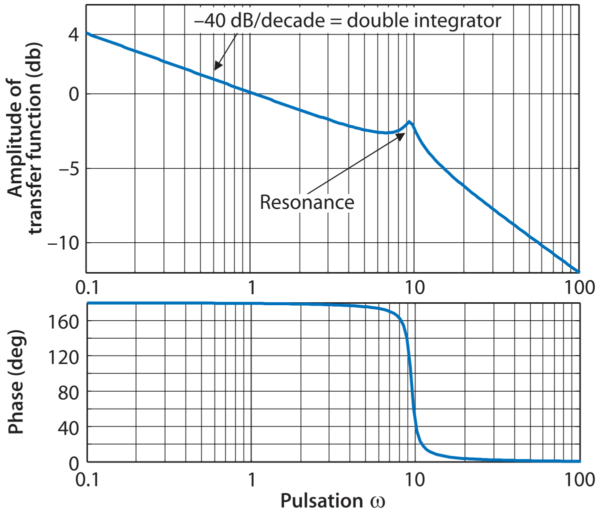


- Bode
(Magnitude + Phase Plots)

- Nyquist Plot
(Polar)

In This Way Feedback May Be Seen as a Filter

- Ex: Lightly Damped Robot Arm



How to Design?
Back to Analog !

Two cases for control design

The system…

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

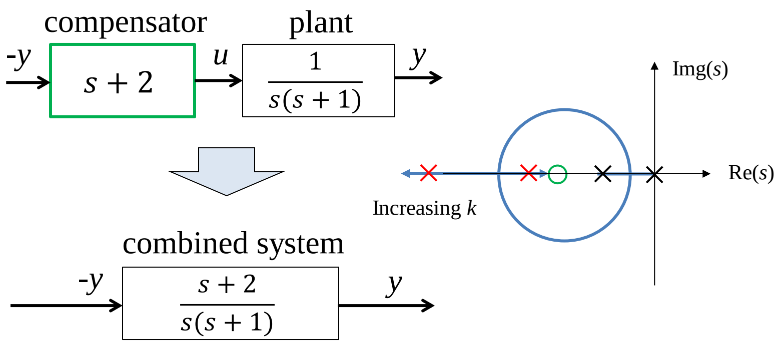
("Performance")

- Attempts to spontaneously disassemble itself

("Stability")

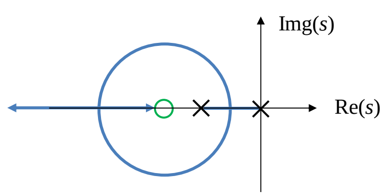
Dynamic compensation

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



But what dynamics to add?

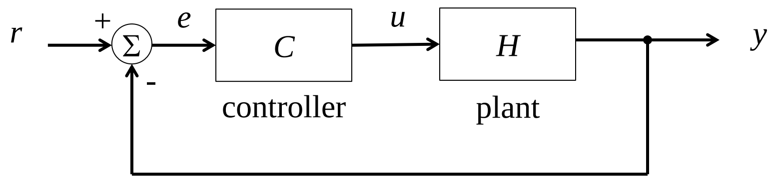
- Recognise the following:
  - A root locus starts at poles, terminates at zeros
  - “Holes eat poles”
  - Closely matched pole and zero dynamics cancel
  - The locus is on the real axis to the left of an odd number of poles (treat zeros as ‘negative’ poles)



The Root Locus (Quickly)

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

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



The Root Locus (Quickly)

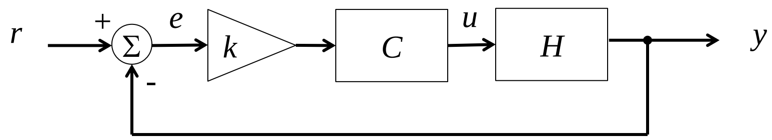
- We often care about the effect of increasing gain of a control compensator design:

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

Multiplying by denominator:

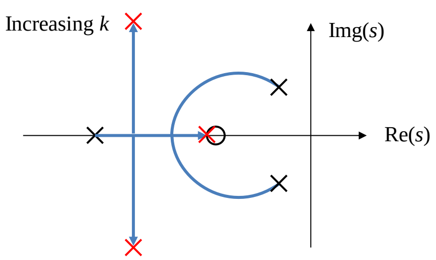
$$\frac{y}{r} = \frac{kC_n H_n}{C_d H_d + kCnHn}$$

characteristic polynomial



The Root Locus (Quickly)

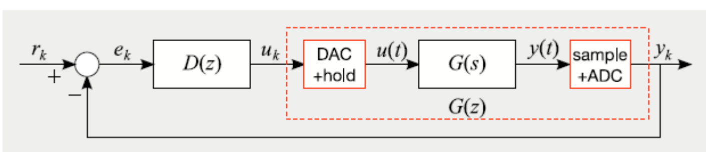
- Pole positions change with increasing gain
  - The trajectory of poles on the pole-zero plot with changing k is called the “root locus”
  - This is sometimes quite complex



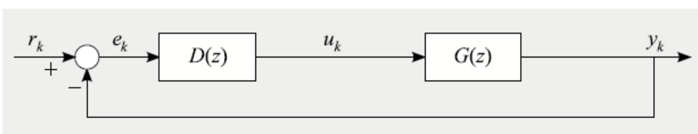
(In practice you’d plot these with computers)

Designing in the Purely Discrete…

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



by considering the purely discrete time system:



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 $\rightarrow$ stability?

$\rightarrow$ performance?

Now in discrete

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

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

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

But, where do we get the control design parameters from?
The s-domain?

Sampling a continuous-time system

suppose $\dot{x} = Ax$

sample x at times $t_1 \le t_2 \le \cdots$: 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)

Source: Boyd, Lecture Notes for EE263, 10-22

Piecewise constant system

consider *time-varying* LDS $\dot{x} = A(t)x$, with

$$A(t) = \begin{cases} A_0 & 0 \le t < t_1 \\ A_1 & t_1 \le t < t_2 \\ \vdots \end{cases}$$

where $0 < t_1 < t_2 < \cdots$ (sometimes called jump linear system)

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

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

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

ith 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 *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 *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)

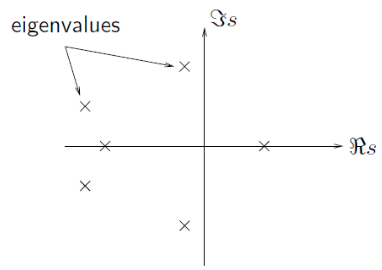
eigenvalues determine (possible) qualitative behavior of x:

- 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 *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 *x(t)* [5]

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

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

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{r} p_{ij}(t)e^{\lambda_j t}$$

where $p_{ij}(t)$ is a polynomial of degree $< n_j$ (that depends linearly on $x(0)$)

Source: Boyd, Lecture Notes for EE263, 10-28

Emulation vs Discrete Design

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

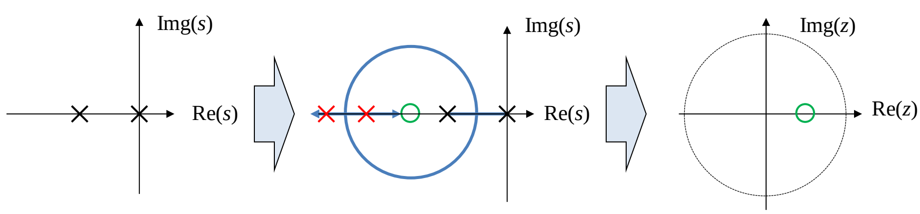
$$\text{eg. } s^2 + 2s + 1 = (s+1)^2$$
$$z^2 + 2z + 1 = (z+1)^2$$

Root loci behave the same on both planes!

- Therefore, we have two choices:
  - Design in the s-domain and digitise (emulation)
  - Design only in the z-domain (discrete design)

Emulation design process

1. Derive the dynamic system model ODE
2. Convert it to a continuous transfer function
3. Design a continuous controller
4. Convert the controller to the z-domain
5. Implement difference equations in software



Emulation design process

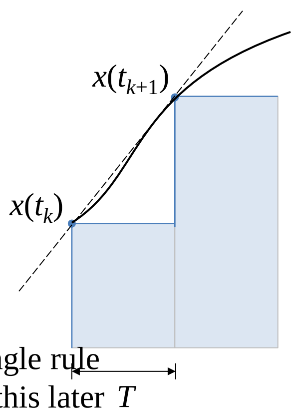
- Handy rules of thumb:
  - Use a sampling period of 20 to 30 times faster than the closed-loop system bandwidth
  - Remember that the sampling ZOH induces an effective $T/2$ delay
  - There are several approximation techniques:
    - Euler's method
    - Tustin's method
    - Matched pole-zero
    - Modified matched pole-zero

Euler's method*

- Dynamic systems can be approximated† 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

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

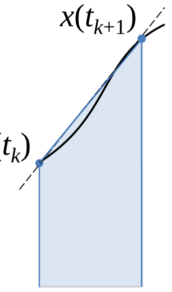
- Tustin uses a trapezoidal integration approximation (compare Euler's rectangles)
- Integral between two samples treated as a straight line:

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

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 $(z+1)$ until they are of equal order*

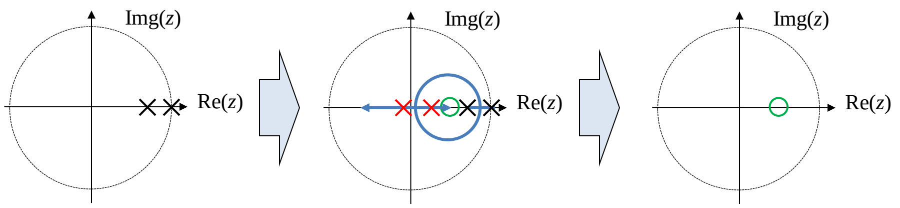
* This introduces an averaging effect like Tustin's method

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

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

Lead/Lag

Some standard approaches

- Control engineers have developed time-tested strategies for building compensators
- Three classical control structures:
  - Lead
  - Lag
  - Proportional-Integral-Derivative (PID)
    (and its variations: P, I, PI, PD)

How do they work?

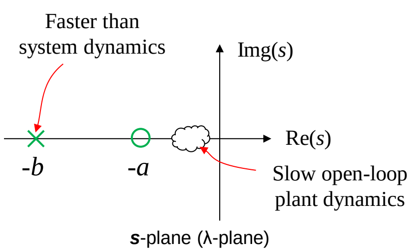
Lead/lag compensation

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

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

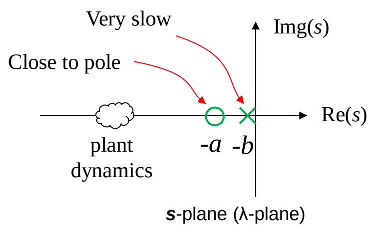
Note:
Lead-lag compensators come from the days when control engineers cared about constructing controllers from networks of op amps using frequency-phase methods. These days pretty much everybody uses PID, but you should at least know what the heck they are in case someone asks.

Lead compensation: $a < b$



- Acts to decrease rise-time and overshoot
  - Zero draws poles to the left; adds phase-lead
  - Pole decreases noise
- Set a near desired 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 – the Good Stuff

- Proportional-Integral-Derivative control is the control engineer's hammer*
  - For P,PI,PD, etc. just remove one or more terms

$$C(s) = k\left(1 + \frac{1}{\tau i s} + \tau d s\right)$$

Proportional
Integral
Derivative

*Everything is a nail. That's why it's called "Bang-Bang" Control ☺

PID – the Good Stuff

- PID control performance is driven by three parameters:
  - k: system gain
  - 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: P Control only

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

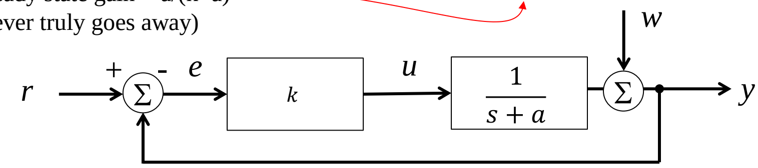
$$y = k\frac{1}{s+a}(r-y)+w$$

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

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

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

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



Now with added integral action

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

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

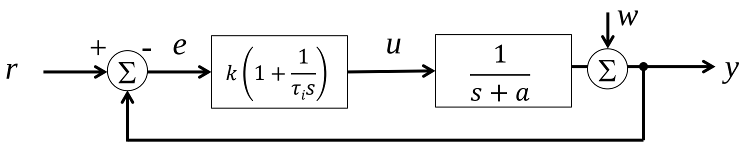
Same dynamics

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

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

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

Must go to zero
for constant w!



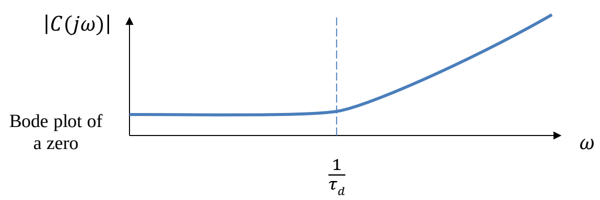
Derivative

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

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

Derivative

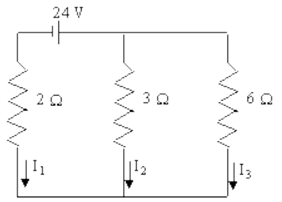
- It is easy to see that PD control simply adds a zero at $s = -\frac{1}{\tau_d}$ with expected results
  - Decreases dynamic order of the system by 1
  - Absorbs a pole as $k \to \infty$
- Not all roses, though: derivative operators are sensitive to high-frequency noise



PID

- Collectively, PID provides two zeros plus a pole at the origin
  - Zeros provide phase lead
  - Pole provides steady-state tracking
  - Easy to implement in microprocessors
- Many tools exist for optimally tuning PID
  - Zeigler-Nichols
  - Cohen-Coon
  - Automatic software processes

Break!: Fun Application: Linear Algebra & KVL!



We can write this as:

$$\begin{pmatrix} 1 & 1 & 1 \\ -2 & 3 & 0 \\ 0 & -3 & 6 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 24 \\ 0 \end{pmatrix}$$

So we have:

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -2 & 3 & 0 \\ 0 & -3 & 6 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 24 \\ 0 \end{pmatrix}$$

Using a computer algebra system to perform the inverse and multiply by the constant matrix, we get:

$$I_1 = -6 \text{ A}$$

$$I_2 = 4 \text{ A}$$

$$I_3 = 2 \text{ A}$$

We observe that I_1 is negative, as expected from the circuit diagram.

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

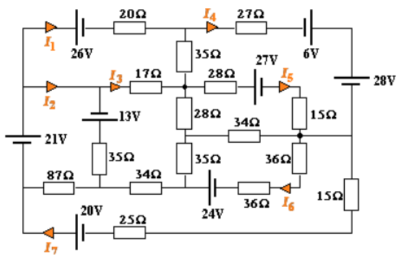
Break!: Fun Application: Linear Algebra & KCL!

We solve this using a computer as follows. We just write the coefficient matrix on the left, find the inverse (raise the matrix to the power -1) and multiply the result by the constant matrix.

You can use Matlab, Mathcad or similar math software to do this. Wolfram|Alpha is a free alternative.

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

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



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

Next Time…

- **Digital Feedback Control**

- Review:
  - Chapter 2 of FPW

- More Pondering??