6.01 Midterm 1 Review
LTI

→ Goal: - framework for analyzing behavior
- model physical systems (e.g. robot, bunnies)

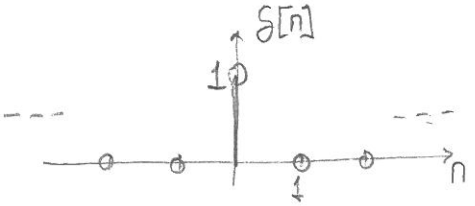
→ Signal: - Function of time
- In 6.01 we just consider DT signals, $x[n]$ defined for $n \in \{\ldots, -1, 0, 1, \ldots\}$
- Notation: - Signals represented by upper case, e.g. X
  - Samples represented by lowercase, e.g. $x[n]$

$x[n]$ = the value of signal X at time n

CT 18.03

- Useful Signals:
  - Unit Sample Signal (Δ)

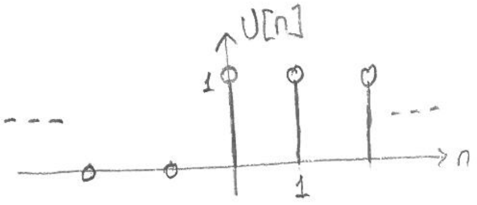
$$\delta[n] = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise} \end{cases}$$



→ Any signal can be written as a sum of scaled and delayed versions of Δ.

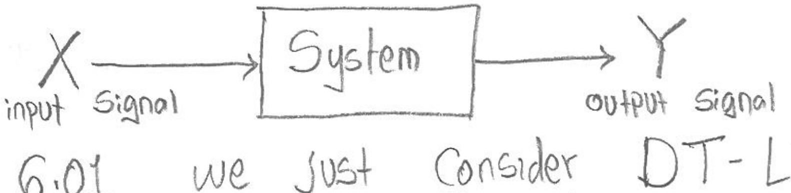
- Unit Step Signal (U)

$$u[n] = \begin{cases} 1, & \text{if } n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

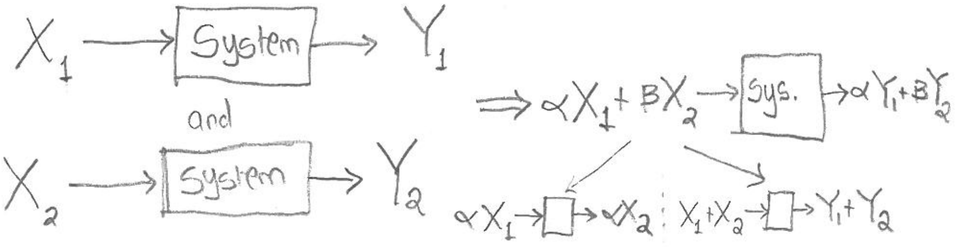


$$u[n] = \sum_{i=0}^{n} \delta[n-i]$$

→ System: - Transforms Signals



- In 6.01 we just consider DT-LTI systems.
  → LTI = Linear Time-Invariant
  → Linear:



→ Time-Invariant:

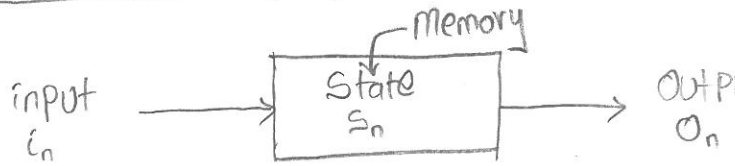
$$X \rightarrow \boxed{\text{System}} \rightarrow Y \Rightarrow R^n X \rightarrow \boxed{\text{System}} \rightarrow R^n Y$$

where R is the right-shift operator.

- We have many different representations for systems:
  1. State Machines
  2. Block Diagrams
  3. Difference Equations
  4. Operator Equations
  5. System Functionals
  6. Poles

Moving between repr's:
→ Labeling individual signals usually helps.

1. State Machines: Computational framework



→ gets input i_n
→ generates output o_n
→ moves to new state s_{n+1}

→ Good for: Seeing exact output of a system to a particular input.

→ 6.01 infrastructure for Statemachines

start()
step(inp)
transduce (inps)

→ change internal state of SM
→ already defined for you.

★ startState / getStartState()
★ getNextValues(state, inp)
    ↳ (nextState, out)

→ do NOT change internal state
→ should be redefined for each new type of SM.

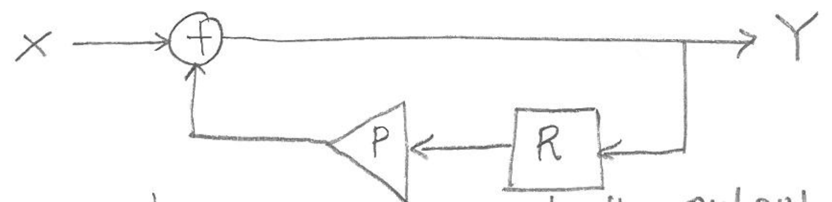
→ Example:
```
Class MysterySM(SM):
    def __init__(self, p):
        self.p = p
        self.startState = 0
    def getNextValues (self, state, inp):
        out = self.p * state + inp
        return (out, out)
```

→ MysterySM(0.5).transduce([1,0,0,0]) → [1.0, 0.5, 0.25, 0.125]
→ MysterySM(2).transduce([1,0,0,0]) → [1, 2, 4, 8]

(problem 1)

② Block Diagrams: Visualize Signal flow Paths

→ Example:



→ Easy to see how each sample of the output (Y) is computed.

③ Difference Equations: let us compute output to specific inputs

→ Example: $y[n] = x[n] + p\,y[n-1]$

→ Unit Sample Response:

n	-1	0	1	2	3	4	
$\delta[n]$	0	1	0	0	0	0	---
$h[n]$	0	1	p	p^2	p^3	p^4	

rest

$$\rightarrow h[n] = \begin{cases} 0, & n < 0 \\ p^n, & n \geq 0 \end{cases}$$

→ Another description of LTI using difference equations:

$$y[n] = c_0\,y[n-1] + c_1\,y[n-2] + \_\_\_ + c_{K-1}\,y[n-K] + \\ d_0\,x[n] + d_1\,x[n-1] + \_\_\_ + d_j\,x[n-j]$$

where $c_0, ---, c_{K-1}, d_0, ---, d_j$ are fixed constants.

④ Operator Equations: use the right Shift $\boxed{\text{operator}}$ R

→ Very easy to manipulate using polynomial algebra

→ Polynomial algebra "works" because we're dealing with LTI systems (you're not responsible for knowing the details of why polynomial algebra works).

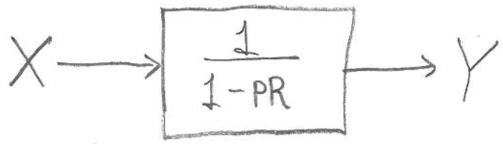
→ Example: $Y = X + PRY$

→ Description of LTI Systems:

$$Y = C_0RY + C_1R^2Y + \_\_\_ + C_K R^{K-1} Y + \\ d_0X + d_1RX + \_\_\_ + d_j R^j X.$$

⑤ <u>System Functionals</u>: ratio of output signal to input signal in R.

→ <u>Example</u>: $\frac{Y}{X} = \frac{1}{1-PR}$

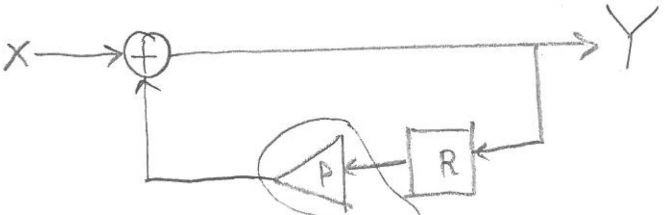


→ Very nice method of abstraction (problems 2, 3)

⑥ <u>Poles</u>: to understand <u>long-term</u> behavior of System

→ The first fire representations specify the exact system, but poles do not! That is, two different systems may have the same set of poles.

→ Basis for Poles:



→ Unit Sample response: $h[n] = p^n u[n]$

Pole: P

→ <u>Idea</u>: Any LTI system can be described as a sum of small components like the small system above.

⇒ The unit sample response to an arbitrary system can be written as:

$$h[n] \sim \sum_{i=1}^{K} c_i p_i^n$$

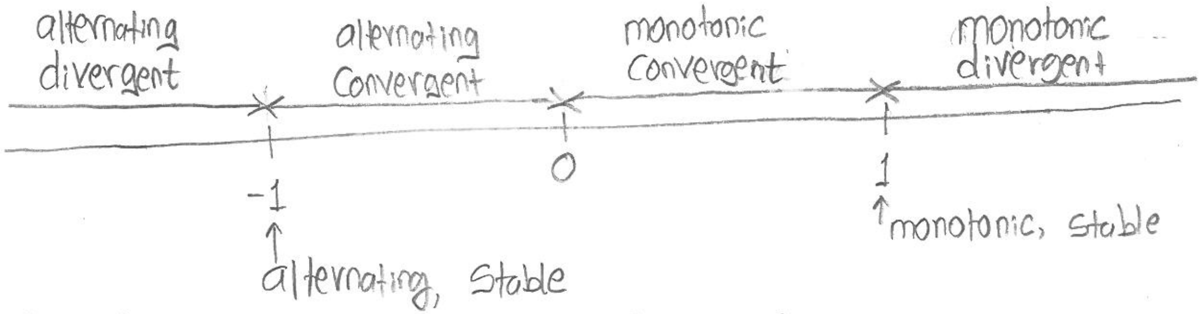
We call $P_1, P_2, \dots, P_K$ the poles of the system.

Don't worry too much about these details

$C_1, C_2, \dots, C_K$ will all be <u>Polynomials</u> in n. If $P_1, P_2, \dots, P_K$ are <u>distinct</u>, $C_1, \dots, C_K$ will be constants. The key is that the response is dominated by the <u>exponential</u> P_i^n terms.

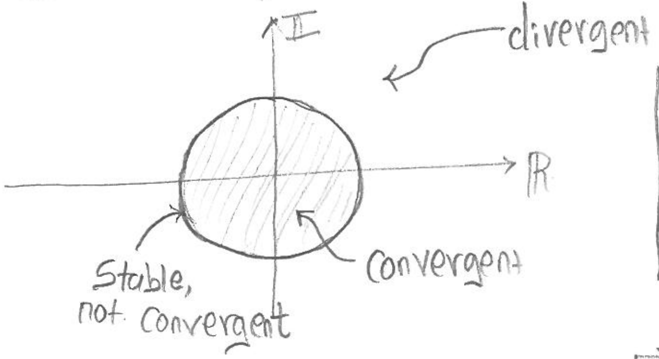
→ Often we look at the dominant pole(s), the pole(s) of largest magnitude, to Understand the long-term behavior of Systems.

→ Real dominant pole:



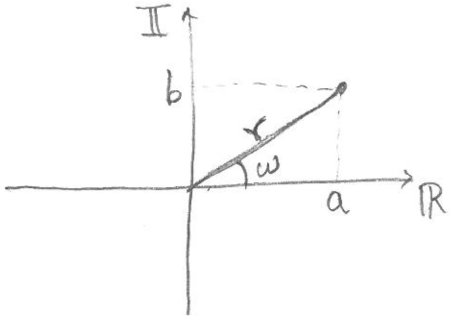
→ Complex poles come in Conjugate pairs ← system response is real.

→ Complex dominant pole:



→ $\omega \neq 0 \Rightarrow$ oscillation
→ Period of oscillation $\frac{2\pi}{\omega}$

→ Complex numbers: two representations → rectangular → polar



→ Rectangular: $a + bj$
→ Polar: $re^{j\omega}$
→ Relationships: $r = \sqrt{a^2+b^2}$, $\omega = \tan^{-1}\left(\frac{b}{a}\right)$ | $a = r\cos(\omega)$, $b = r\sin(\omega)$

→ Computing poles from System Functional.

→ Trick: $R \to \frac{1}{z}$ (more on this in 6.003 ☺)

→ Example: $H(R) = \frac{R}{1 - \frac{3}{2}R + R^2} \longrightarrow H(z) = \frac{\frac{1}{z}}{1 - \frac{3}{2}\frac{1}{z} + \frac{1}{z^2}}$

$$= \frac{z}{z^2 - \frac{3}{2}z + 1}$$

→ Poles = roots of denominator of H(z)

$$z^2 - \frac{3}{2}z + 1 = (z-1)(z-\tfrac{1}{2}) \rightarrow \boxed{1, \tfrac{1}{2}}$$

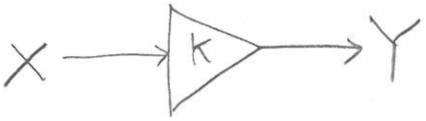
(mention practice problems
in past exams)

- Primitives and Combinations for LTI Systems:

- Primitives: 1) Gain: $y[n] = K \cdot x[n]$

$$Y = KX$$

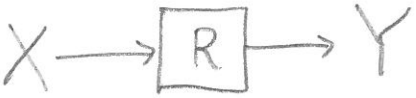
$$\frac{Y}{X} = K$$



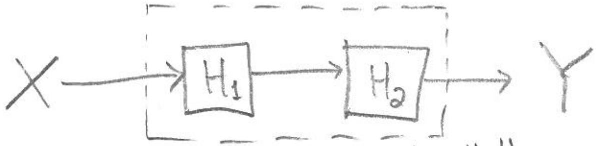
2) Delay: $y[n] = x[n-1]$

$$Y = RX$$

$$\frac{Y}{X} = R$$

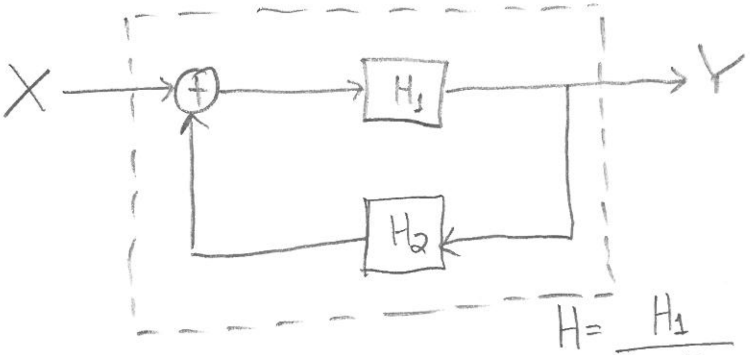


- Combinations 1) Cascade:



$H = H_1 H_2$

2) Feedback:

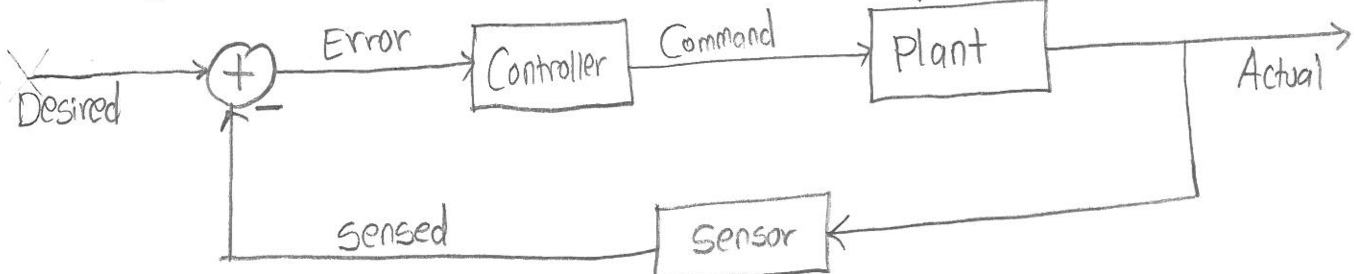


$$H = \frac{H_1}{1 - H_1 H_2}$$

(problem 4)

- Feedback Systems

6.01 T-shirts ☺



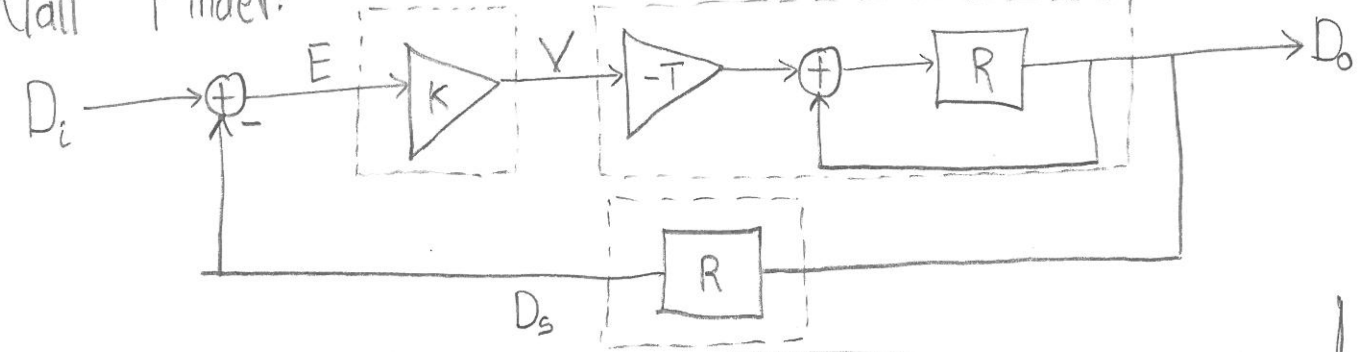
→ Controller: Commands to physical system

→ Plant: model of physical system

→ Sensor: model of the information we have about the attribute we are trying to control.

- Feedback Systems from Design Labs 4,5,6

- Wall Finder:

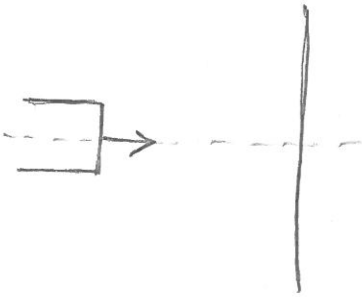


$V = KE$

$D_o = RD_o - TRV$

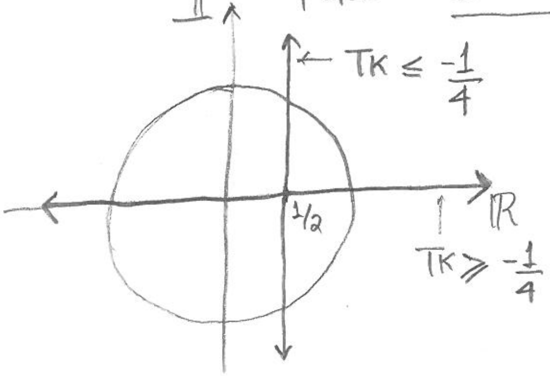
$D_s = RD_o$

$$\boxed{\frac{D_o}{D_i} = \frac{-TKR}{1-R-TKR^2}}$$

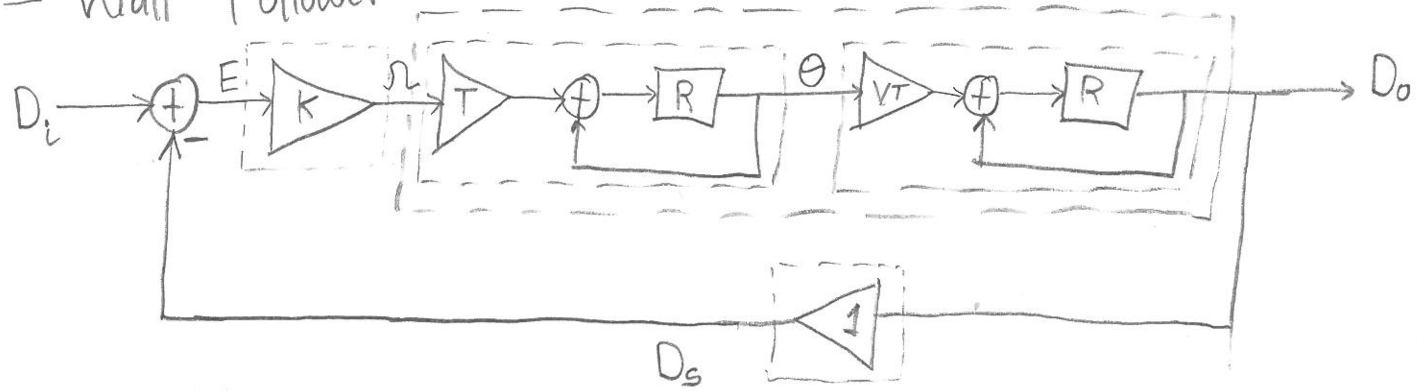


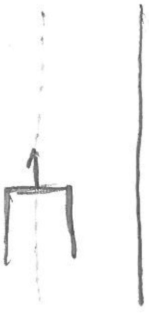
Poles: $\dfrac{1 \pm \sqrt{1+4TK}}{2}$

Root Locus:



- Wall Follower





$\dfrac{\Omega}{E} = K$

$\dfrac{\Theta}{\Omega} = \dfrac{TR}{1-R}$

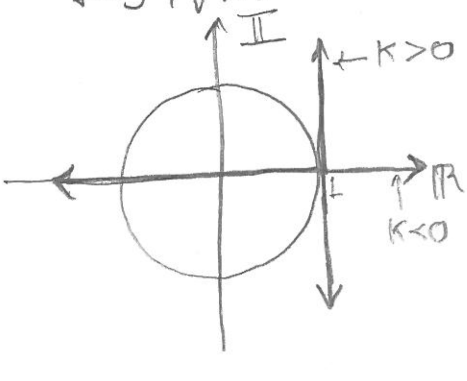
$\dfrac{D_o}{\Theta} = \dfrac{VTR}{1-R}$

$\dfrac{D_s}{D_o} = 1$

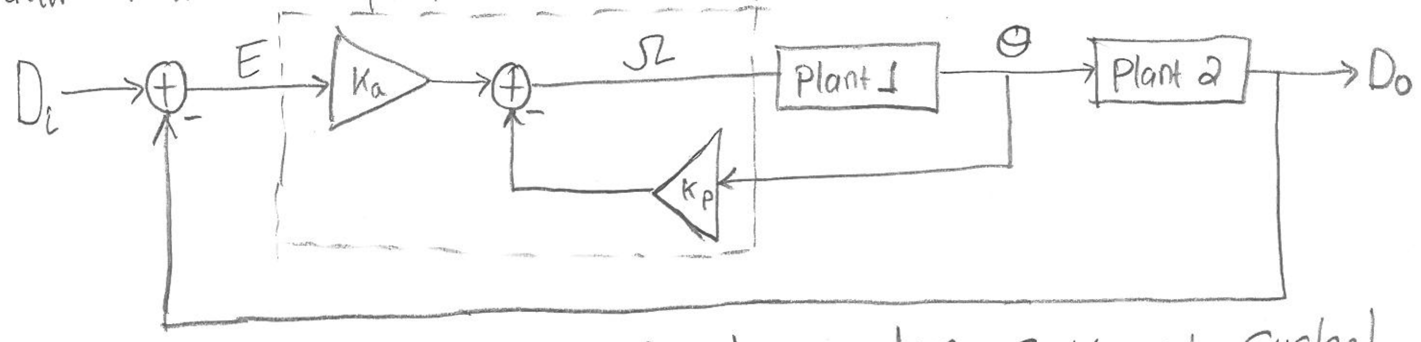
$\dfrac{D_o}{\Omega} = \dfrac{VT^2R^2}{(1-R)^2}$ $\dfrac{D_o}{E} = \dfrac{KVT^2R^2}{(1-R)^2}$

$$\boxed{\frac{D_o}{D_i} = \frac{KVT^2R^2}{1-2R+(1+KVT^2)R^2}}$$

Poles: $1 \pm jT\sqrt{KV}$



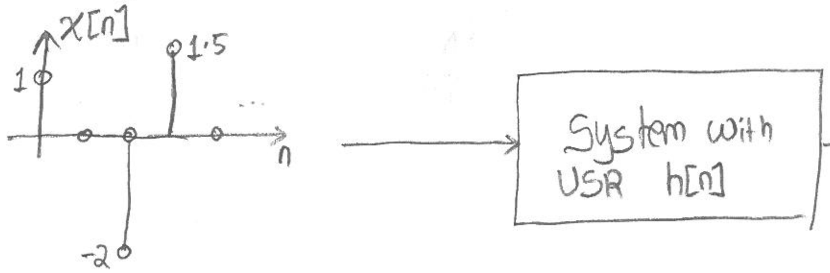
— Wall Follower: proportional + angle



→ Can Choose K_a and K_p to produce Convergent System!

Superposition

→ If we Know the Unit Sample response $h[n]$, we can Compute the response to any Signal.



$y[n] = h[n] - 2h[n-2] + 1.5\ h[n-3]$

$x[n] = \delta[n] - 2\delta[n-2] + 1.5\ \delta[n-3]$