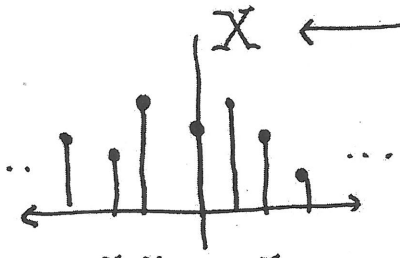
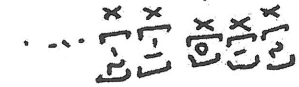
Signals

In 6.01, a signal is a sequence of discrete values, indexed by time.



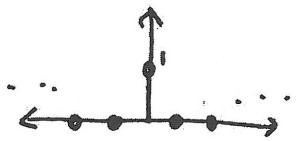
← signals are labeled by uppercase letters



← and values of a signal at a specific point are labeled by lowercase letters with an index.

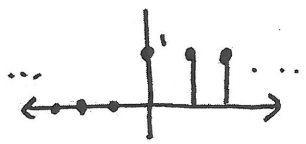
Unit sample:

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



Unit step:

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



Systems

Systems transform signals, changing one signal into another.

There are many ways to describe systems:

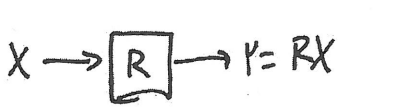
- Block diagrams
- Difference equations
- Operator equations
- System functionals
- State machines

System building blocks:

Gain

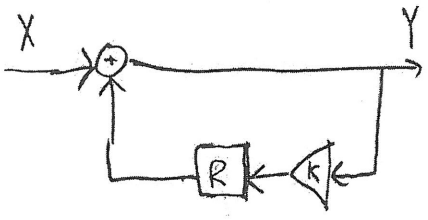
$y[n] = k \cdot x[n]$

Delay:



$y[n] = x[n-1]$

Block diagrams:



- Shows "flow" of system components.
- Makes it easy to label parts of the system.

Difference equation:

$$y[n] = x[n] + Ky[n-1]$$

- Gives a "formula" for computing a discrete value from other discrete values

Operator equation:

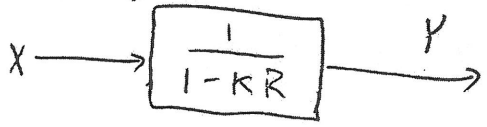
$$Y = X + KRY$$

- Manipulates entire signals, rather than single values, as in a difference equation
- Can use normal algebra

System Functionals:

$$\frac{Y}{X} = \frac{1}{1-KR} = H$$

- Solve operator equations for $\frac{Y}{X}$
- often labeled "H"
- Makes it easy to find poles
- Lets systems be "blackboxed"



State machines:

- State machines formalize the concept of system memory, or <u>state</u>.
- System behavior depends on
  - Current input
  - Current state
- From this, the system generates a new state, and some output.
- In our state machine class, there are two methods that define a system:

  getStartState(self) - returns an initial state.

  getNextValues(self, state, input) -

  - Based on current input and state (input arguments) return output and new st

The SM class has other functions that do not need to be over-ridden:

- start(self) - initializes a state machine
- step(self, inp) - applies one input to the state machine
- transduce(self, inps) - applies a sequence of inputs to the state machine.

How can we implement our system as a state machine?

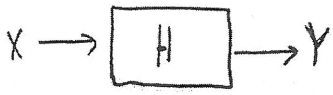
- We need to decide on a state representation.
  - From difference equation, all we need is
    - x[n] - current input
    - y[n-1] - last output
  - So, state is just y[n-1]

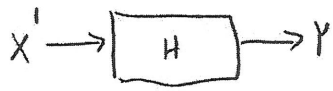
```
class ExampleSystem(SM):
    def getStartState(self):
        return 0                    #start from rest
    def getNextValues(self, state, input):
        output = input + 2*state
        newState = output
        return (output, newState)
```

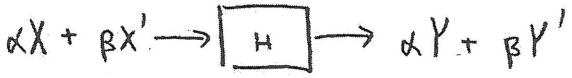
State machines can represent any LTI system. State machines can also represent other processes as well, such as the ribosome SM or the Farmer-Cabbage-Wolf SM you explored in homeworks.

LTI

Linear:

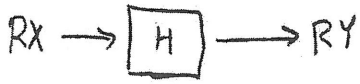






Time invariant:

- Delay the input, delay the output



Response of a system

- The output of a system when the input is a unit sample signal is known as the response.
- What is the response to our system?

$$y[0] = x[0] + Ky[-1] = 1 + K \cdot 0 = 1$$
$$y[1] = x[1] + Ky[0] = 0 + K = K$$
$$y[2] = x[2] + Ky[1] = 0 + K \cdot K = K^2$$

$$y[n] = \quad \ldots \quad = K^n$$

- K is what is known as a "pole."
- The response of a system with a single pole is proportional to that pole raised to the nth power, as shown in the example.

Finding poles

- In the system functional, replace R with $1/z$.
- Find the roots of the denominator (values of z where the system functional $\longrightarrow \pm\infty$)

From the example system:

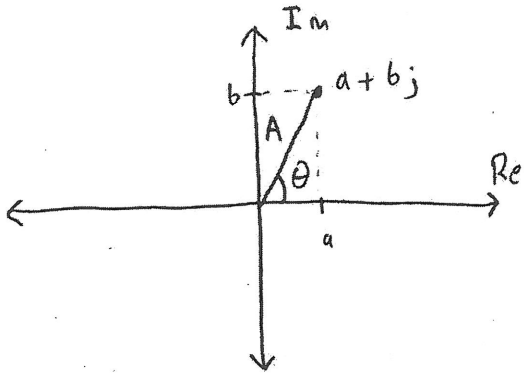
$$H = \frac{1}{1-kR} \longrightarrow \frac{1}{1-\frac{1}{z}k} = \frac{z}{z-k} \longrightarrow \text{pole is at } z = k$$

- If the system is more complicated, the denominator can have multiple roots,
  - We say the system has multiple poles.
  - In general, these can be complex.

- A system with multiple poles can always be shown to be the sum of multiple single pole systems via partial fraction decomposition.
  - See book, section 5.5.2

Complex numbers

- Complex numbers are typically represented by a real part and an imaginary part: $a + bj$
- We can plot these in 2D:

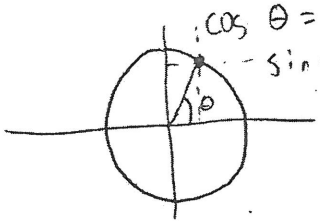


- This suggests another way we might represent these values, as a magnitude and an angle.

- The magnitude is just distance: $A = \sqrt{a^2 + b^2}$
- The angle can be found by the arctangent: $\arctan\left(\frac{b}{a}\right)$
  - In python, we used $\text{atan2}(b, a)$ in order to handle quadrants correctly.

Polar form:

- Remember the unit circle:



$\cos\theta$ = x position = real part

$\sin\theta$ = y position = imag part

- $a + bj = A(\cos\theta + j\sin\theta)$
- Euler's formula: $\cos\theta + j\sin\theta = e^{j\theta} \longrightarrow a + bj = Ae^{j\theta}$

What happens when we raise complex numbers to powers?

$(a + bj)^n$ = ugly

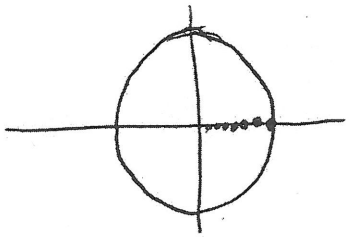
$(Ae^{j\theta})^n = A^n e^{j(\theta n)}$

- θn: Angle increases linearly
- A^n: Amplitude increases/decreases geometrically

Some examples:

$0.9 + 0j \longrightarrow A = 0.9,\ \theta = 0$

$0.9^0 = 1$
$0.9^1 = 0.9$
$0.9^2 = 0.81$
$0.9^3 = 0.72$

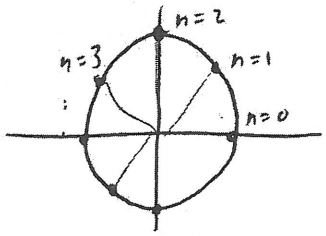


$\frac{\sqrt{2}}{2}(1 + 1j) \longrightarrow A = 1,\ \theta = \pi/4$

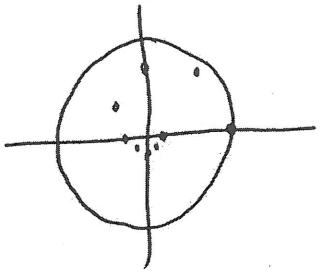
$\left(e^{j\pi/4}\right)^0 = e^{j0} = 1$

$\left(e^{j\pi/4}\right)^1 = e^{j\pi/4} = \frac{\sqrt{2}}{2}(1 + 1j)$

$\left(e^{j\pi/4}\right)^2 = e^{j\pi/2} = j$



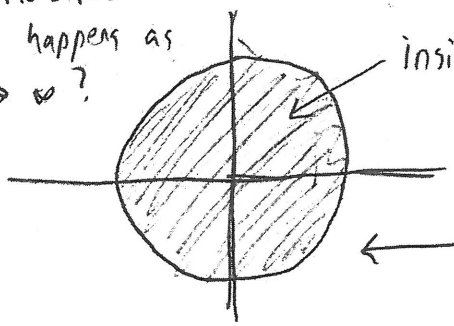
What would happen with $0.9e^{j\pi/4}$?



- Magnitude decreases as angle increases
- Spiral

CONVERGENCE

- What happens as $n \longrightarrow \infty$?



inside the unit circle ($A < 1$), it "converges" (approaches 0)

← outside the unit circle ($A > 1$), it "diverges" (goes towards $\pm\infty$)

System responses (again):

- If there are multiple poles, which one matters most?
- The one that overpowers the rest
  - This will be the pole with the largest magnitude
  - Called "dominant pole"

- What are the types of system responses?

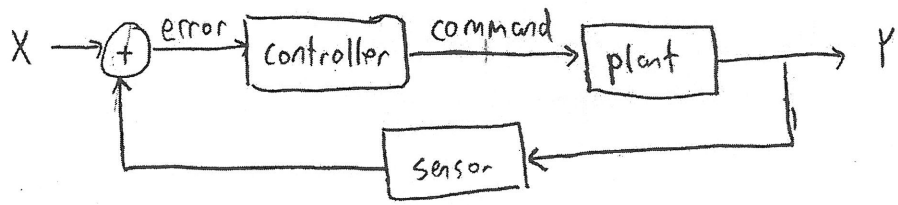
Dominant pole	Response
Real, positive	monotonic
Real, negative	alternating
Complex	oscillating

Dominant pole magnitude	Response
< 1	Convergent
$= 1$	Neither
> 1	Divergent

- For oscillating responses, we're often interested in the period of oscillation
  - $\text{Period} = \frac{2\pi}{\theta} = \frac{\text{total angle}}{\text{angle per timestep}}$

- If the poles are complex, they always come in conjugate pairs
- This is why the system output is real

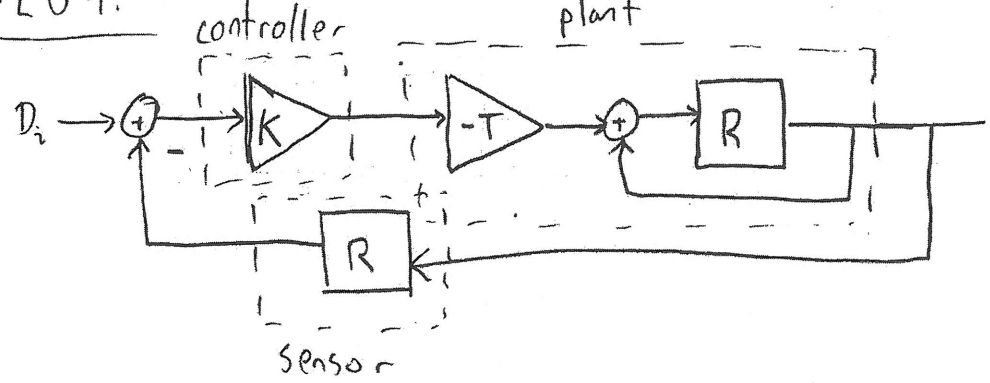
Control systems



controller - generates a command signal from an error signal. This is where you, the control designer, have flexibility.
plant - the physics of the system being controlled
sensor - the subsystem that detects the real world

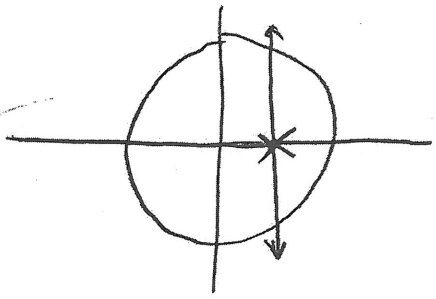
Reviewing the design lab systems:

DL04:



system functional: $\frac{-TRK}{1-R-TR^2K}$

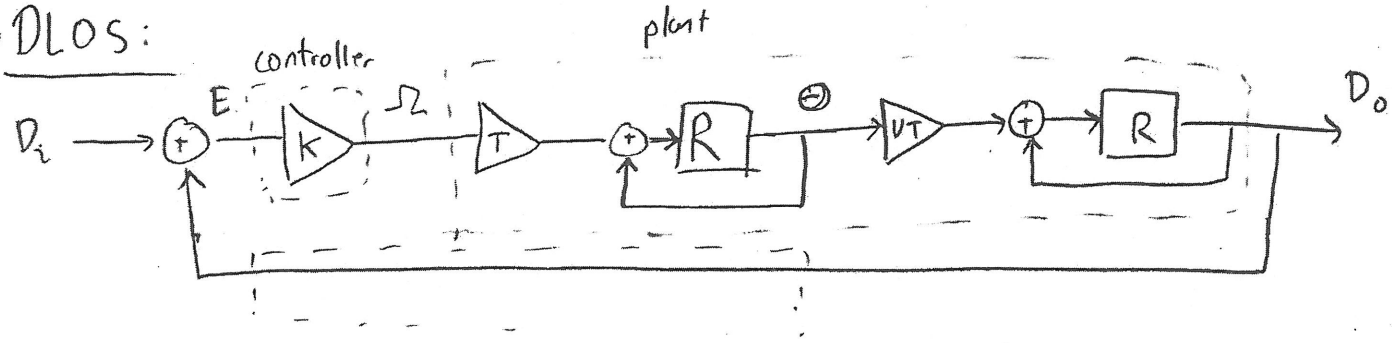
Root locus:



poles: $1-R-TR^2K \rightarrow z^2-z-TK$

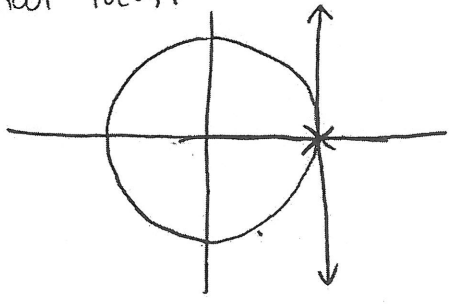
$z = \frac{1}{2} \pm \frac{\sqrt{1-4TK}}{2}$

DL05:



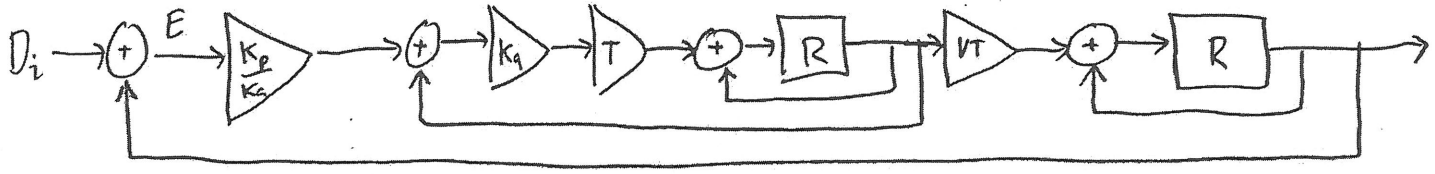
system functional: $\frac{T^2VKR^2}{(1+T^2VK)R^2-2R+1}$

Root locus:



poles: $\frac{2}{2} \pm \frac{\sqrt{4-4(1+T^2VK)}}{2} = 1 \pm jT\sqrt{VK}$

DL06:



- This system uses two nested loops
- This makes it a system with three poles