Today: # State Space: Matrix Equations

Lecture 9
ECE 147B
Digital Control

Reading: FPW 6.1
or: FV 7.6, 7.7

Midterm ~~Was Feb. 12 (Tues.)~~
Now Feb. 14 (Thurs.)

Topics:
- Controls overview (high-level viewpoint)
- **State Space Equations**
- Numerical approx's: NOW VIA MATRIX EQNS!!
- ZOH effect: NOW VIA MATRIX EQNS!!
- MATLAB commands (in the land of state space...)
- Review for Midterm (to be continued...)
- Midterm evaluations

Next Class: Midterm Review and Midterm Course Evaluations.
- Review all material to date, especially class handouts

Controls Overview: Why "State Space"?

Controls techniques can be grouped, broadly-speaking, into one of two camps:

1. Transform Methods (i.e., using Laplace, Z, and/or Fourier transforms)
2. State Space Methods

Here, at mid-term in 147B, we cross over from 1 to introduce 2.

But first, an overview in: **Approaches to Control** (...state space vs transforms...)

"Classical Control"
(aka **Transform Methods**)

Transforms allow one to understand and manipulate dynamics "intuitively".

e.g., Root locus, Bode, Nyquist, "Direct Design" in z domain, etc.

Can do calculations BY HAND.

But normally used for single-input single-output (SISO), rather than MIMO.

"Modern Control"
(aka **State Space Methods**)

Matrix equations allow for multi-input multi-output (MIMO) control.

e.g., Pole placement, linear quadratic regulator (LQR) control, etc.

COMPUTERS make this a practical approach, as system complexity goes up.

(But not as "intuitive"...)

Recall our motivation for learning digital control...

- We want to use computers to control cool stuff in the real world... MIMO control needed.

robot arm

dish washers & cloth washers

stand-alone CNC machine millers

4-legged

6-legged

& more legged robots

automated guided vehicles

unmanned aerial vehicles

Humanoid robots & 2-legged robots

wheeled robots

caterpillar-tracked robots

- ...so, where did Modern Control all begin???

State Space Equations

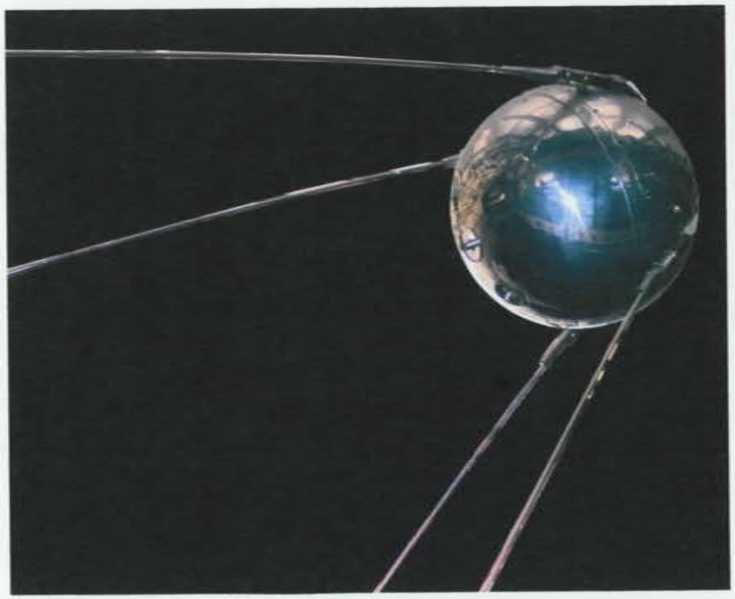
1957: The space race is off! Dawning of the age of "Modern Control" - aka "State Space Methods".

1957 Computers: IBM switches from vacuum tubes to transistors.

1957 Space Sci.: Sputnik orbits earth!

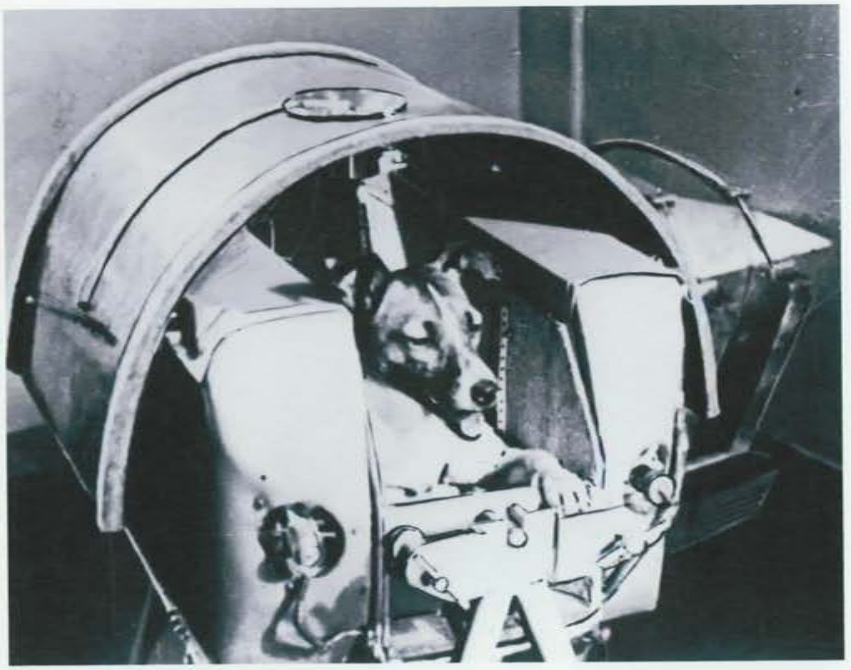
1957 Rockets: R7 launch vehicle (for Sputnik). Gateway to ICBMs... "Rocket science"

1957 Defense: ARPA (now "DARPA") founded.



Sputnik 1
[replica, Air and Space Museum]

8K71PS
Sputnik (PS) launcher
1957



Sputnik 2 [with stray-dog "hero" Laika]

Matrix form Equations of Motion

(Differential Eqn)

CT:

This part sets poles (Stability)

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

(nxn) (mxn)

n states, nth order.

matrix equations for multi-input, multi-output systems ("MIMO")

DT:

(Difference Eqns)

$$x(k+1) = A_d x(k) + B_d u(k)$$
$$y(k) = C_d x(k) + D_d u(k)$$

1. "State space" eqns (above) describe LINEAR, time-invariant (LTI) dynamics.
   const. coeff. differential (or difference) eqn.
2. An n^{th}-order system has n STATES (in vector X).
3. There will be n "state equations" to represent n^{th}-order dynamics.
4. The number of outputs, m, in output vector Y may be less than, equal to, or greater than the number of states, n, in X.

Matrix form Equations of Motion

CT: $\dot{x} = Ax + Bu$, $y = Cx + Du$

x: vector w/ n states
u: vector w/ k inputs
y: vector w/ m outputs

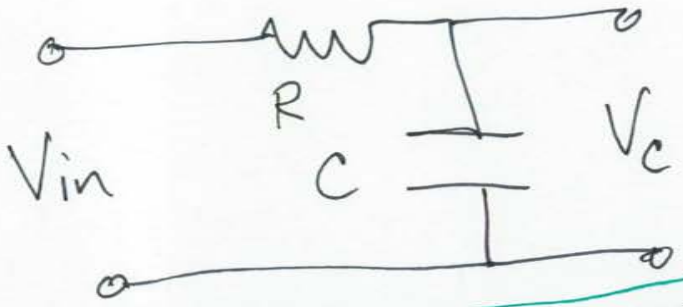
Toy example 1: (1st-order system)

$$\frac{\text{output}}{\text{input}} = \frac{V_c(s)}{V_{in}(s)} = \frac{1}{RCs+1}$$

1 pole: $s = \frac{-1}{RC}$

∴ 1st-order

∴ n = 1 states needed.



Output: $y = V_c$

Input: $u = V_{in}$

States? $x = V_c$ (we could have used "V_R"...)

$V_{in} = (RCs + 1) V_c$

$u = RC\dot{x} + x$ → s.s. form

$\dot{x} = (\ ?\ )x + (\ ?\ )u$

Output: $y = V_c = x$

∴ $y = (1)x$

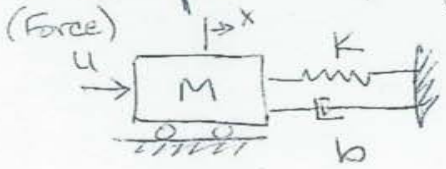
∴ $C = 1, D = 0$

$\dot{x} = \left(\frac{-1}{RC}\right)x + \left(\frac{1}{RC}\right)u$ ∴ $A = \frac{-1}{RC}$, $B = \frac{1}{RC}$

Matrix form Equations of Motion (cont...)

Toy example 2: (2nd-order system)

CT Example: spring-mass-damper (2nd-order) system



$$m\ddot{x} + b\dot{x} + kx = u$$

(equation(s) of motion...)

Let $X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$, $Y = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{C} X + \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{D} u$

$\dot{X} = AX + Bu$

Eq.1 → $\dot{x}$

Eq.2 → $\ddot{x}$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}}_{A} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1/m \end{bmatrix}}_{B} u$$

from the EOM...

Eq2 $$\ddot{x} = \frac{-k}{m}x - \frac{b}{m}\dot{x} + \frac{1}{m}u$$

Eq1 $\dot{x} = \dot{x}$ (?) Just says:

$\frac{d}{dt}(x_1) = x_2$, where $x_1 = x$, $x_2 = \dot{x}$

Note on the 1st-order RC circuit...

Say $y = \begin{bmatrix} V_C \\ V_R \end{bmatrix}$ (2 outputs of interest)

We want

$y = Cx + Du$

$$\begin{bmatrix} V_C \\ V_R \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} V_C + \begin{bmatrix} 0 \\ 1 \end{bmatrix} V_{in}$$

$V_R = u - V_C$

$= u - x$

$$y = \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{C} x + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{D} u$$

State Space: Introduction

Often, one can use MULTIPLE definitions of the "states", and of course it doesn't really matter what order you list them in "X"... Just be CONSISTENT!

e.g., same "toy example 2": (2nd-order system). Let's swap x1 and x2 within vector X:

Same as last page, EXCEPT, $X_1 \leftrightarrow X_2$ ↑ now swapped.

vs Alternate form: $X = \begin{bmatrix} \dot{x} \\ x \end{bmatrix}$, $Y = \begin{bmatrix} x \\ x \end{bmatrix}$

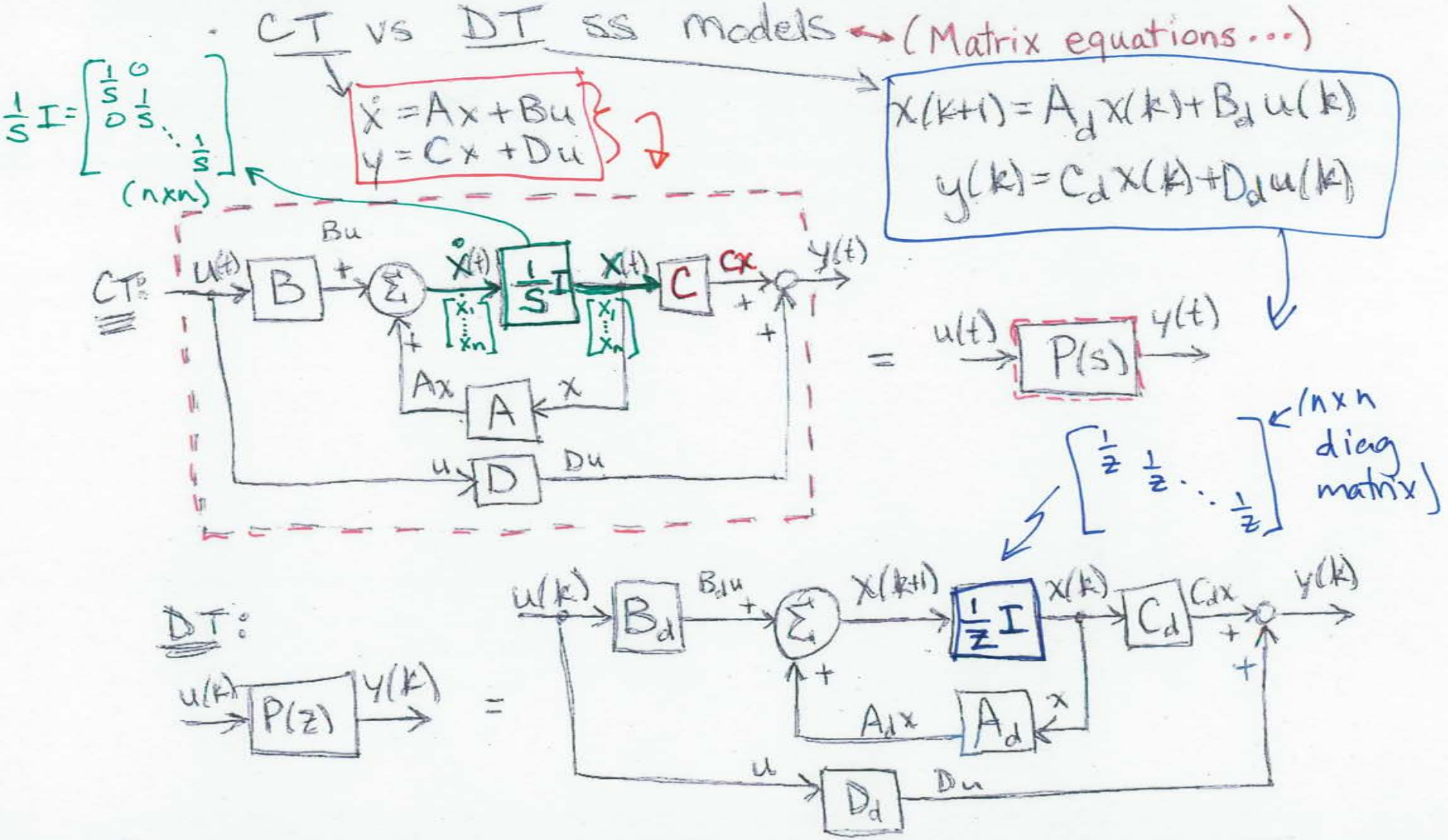
$$Y = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{C\ (m\times n)} X + \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{D\ (m\times k)} u$$

$$\dot{X} = \begin{bmatrix} \ddot{x} \\ \dot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} -b/m & -k/m \\ 1 & 0 \end{bmatrix}}_{A\ (n\times n)} \begin{bmatrix} \dot{x} \\ x \end{bmatrix} + \underbrace{\begin{bmatrix} 1/m \\ 0 \end{bmatrix}}_{B\ (n\times k)} u$$

, k inputs in u.

*Matlab comment:
tf2ss may yield STRANGE definitions for the "states" in X!

State Space: CT vs DT Block Diagram Form



(Recall, for SISO, e.g. $\frac{Y(s)}{X(s)} = \frac{ds+e}{as^2+bs+c}$: STABILITY? from POLES! characteristic eqn.

State Space: CT vs DT Block Diagram Form

STABILITY:

CT (continuous time)

- Eigenvalues of $A_{(n\times n)}$ all in LHP.

  (note, A is SQUARE, as req'd when taking eig.vals)

- i.e., roots of char. eqn all have neg. real part.

- i.e., $\det[sI - A] = 0$

  ↑(eigenvalues)

DT (discrete time)

- Eigenvalues of $A_{d\ (n\times n)}$ all in unit circle

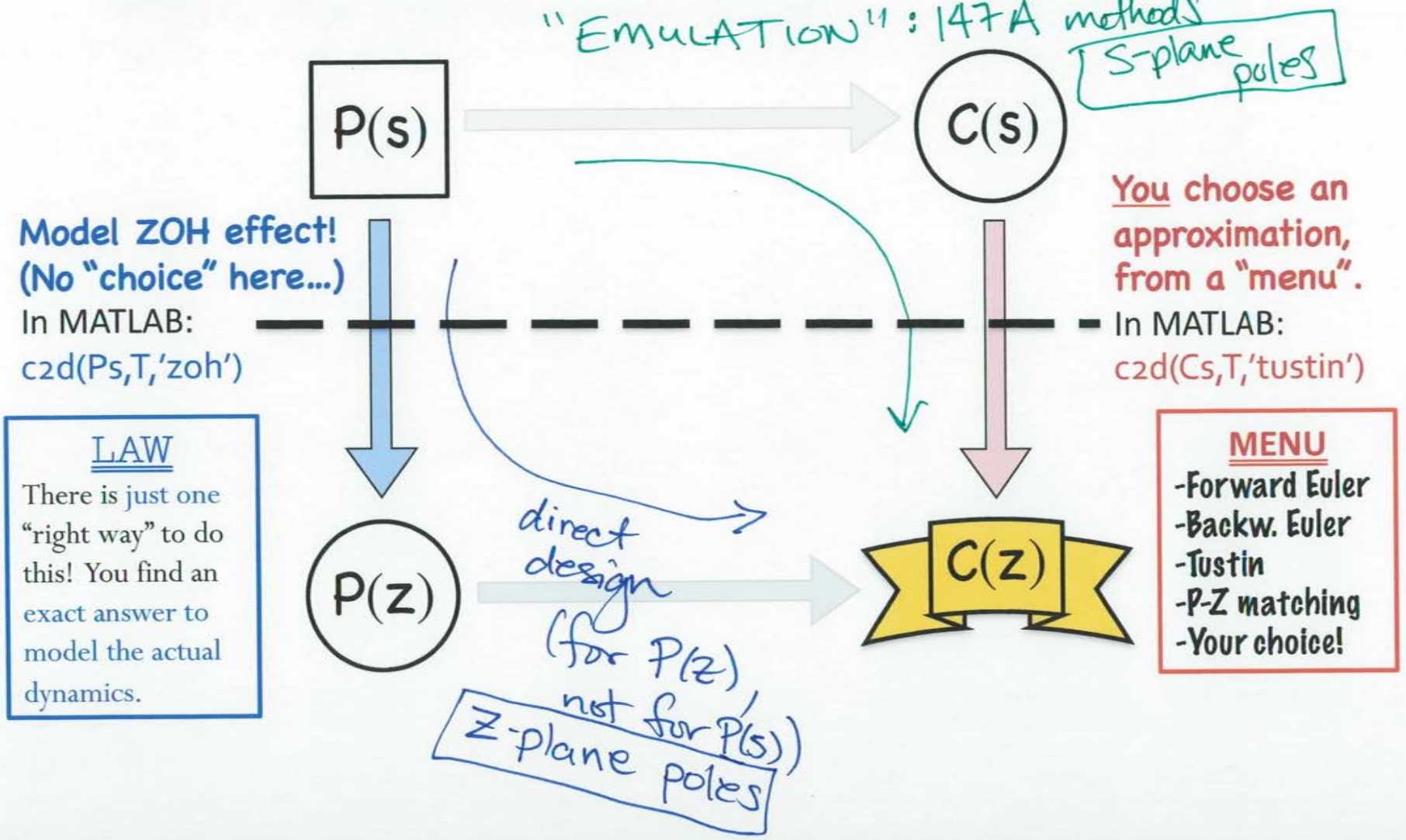
  (for stability)

- $\det[zI - A_d] = 0$

  ↑(eigenvalues)

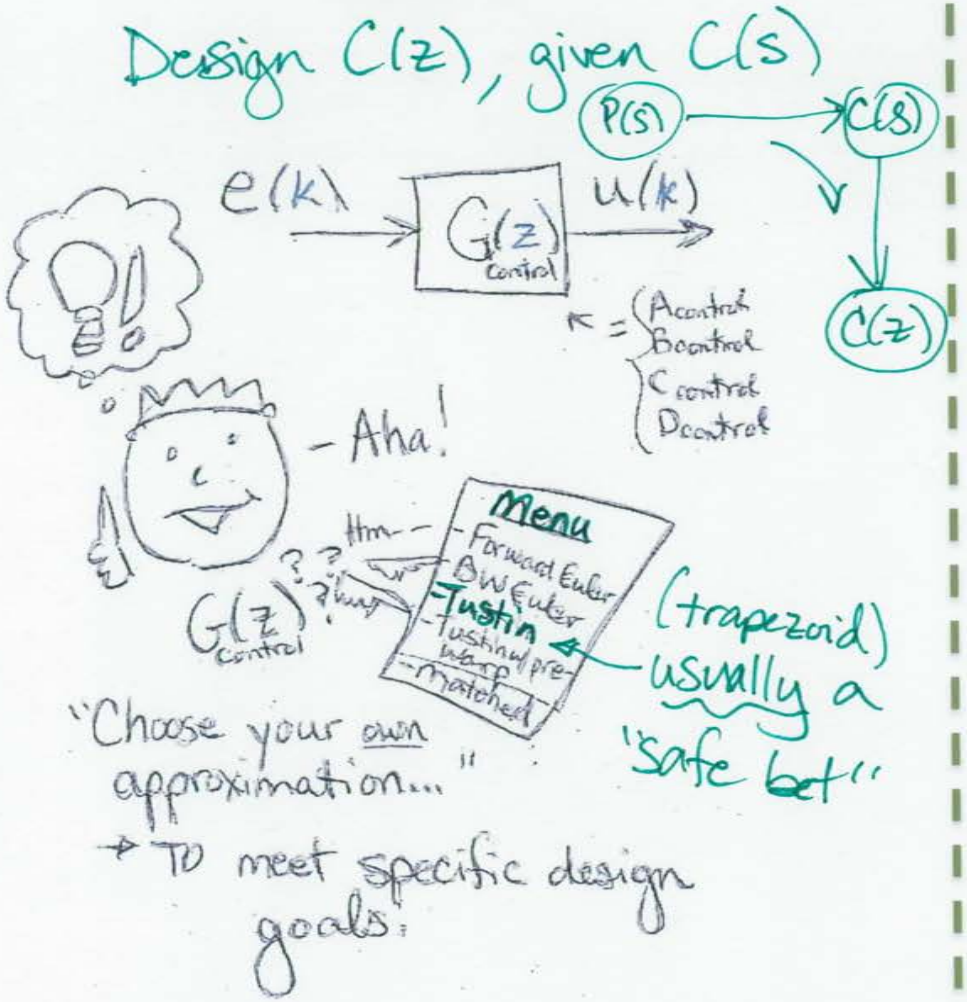
Recall we use Z xforms in 2 ways...

- There are two DIFFERENT ways we will represent dynamics via DT SS models:

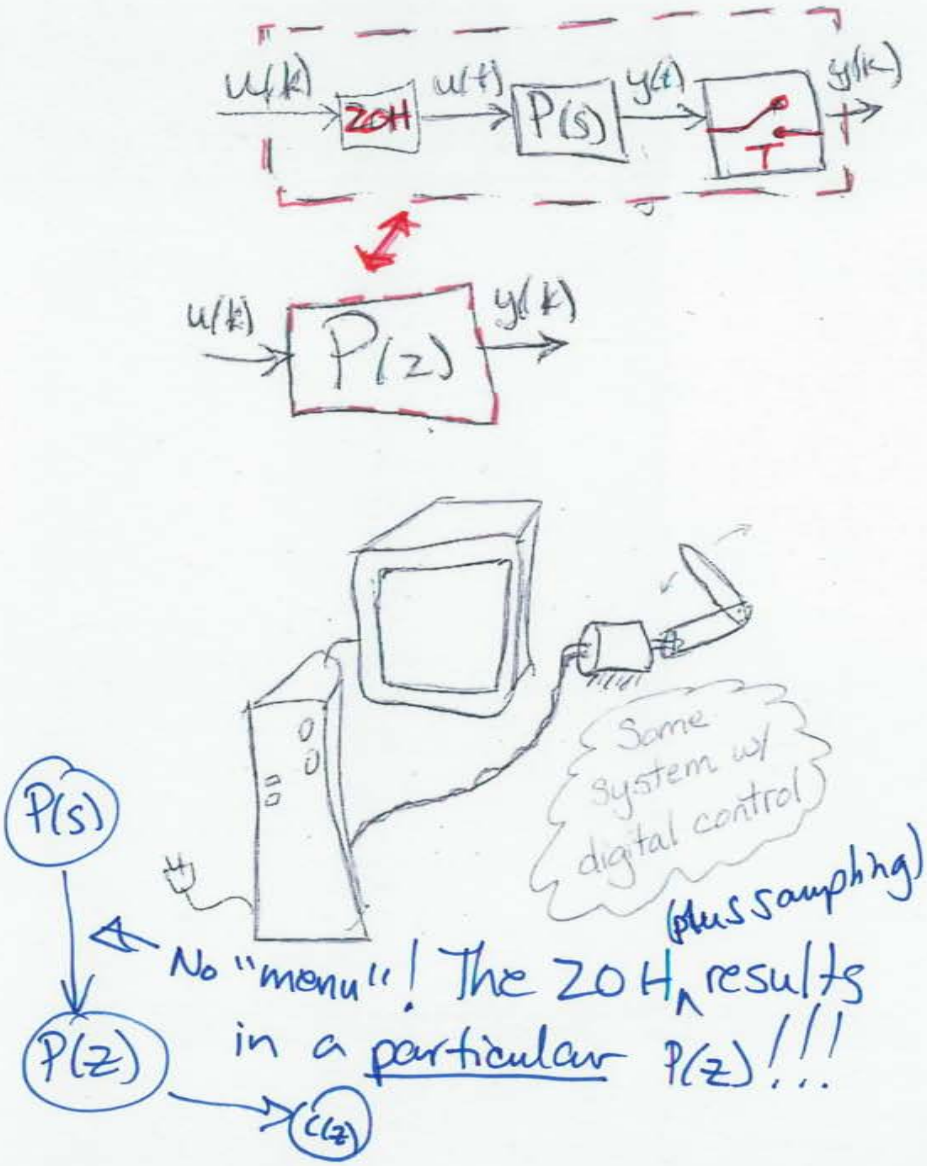


SS Representation: Again, we have 2 cases!!!

Case 1:



Case 2:



Case 1: Numeric Integration

A) Forward Euler [by far the easiest approximation, when done by hand!]

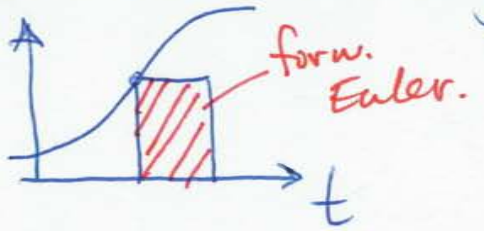
$\dot{X} = AX + Bu \rightarrow \boxed{sX = AX + Bu}$ (Laplace)

(also has the biggest problems w/ stability---)

"s" represents $\frac{d}{dt}$; $\frac{1}{s} \leftrightarrow \int \cdot dt$

→ How can we represent this numerically?

$$\boxed{s \leftrightarrow \frac{z-1}{T}}$$



$sX = Ax + Bu$

$(z-1)X = T(Ax + Bu)$

$X_{k+1} - X_k = TAX_k + BTu_k$ ← Recall, we want:

$\boxed{X_{k+1} = A_d X_k + B_d u_k}$

$X_{k+1} = (I + AT)X_k + (TB)u_k$

$\therefore \boxed{A_d = I + AT}$

$\boxed{B_d = BT}$

CT: $y = CX + Du$

DT? what is y now?

$y_k = CX_k + Du_k$ $\therefore$ $\boxed{C_d = C}$ $\boxed{D_d = D}$

Note:

$C = C_d$ & $D_d = D$

b/c same states are used (x) for CT & DT description of dynamics

Case 1: Numeric Integration (cont...)

B) Backward Euler: A little trickier to implement for state space...

① $s \leftrightarrow \frac{z-1}{zT}$ ← similar to F. Euler, except this z...

$sX = AX + Bu$

$(z-1)X = Tz(AX + Bu)$

$(I - AT)\underbrace{X_{k+1}}_{zX} = X_k + BTu_{k+1}$ ⟷ $X_{k+1} = A_d X_k + B_d u_k$

Wrong Form! We cannot have "u_{k+1}" in our matrix eqns...

The Solution is a change of variables:

$X_k = W_{k+1}$

$(I - AT)W_{k+2} = W_{k+1} + BTu_{k+1}$

$W_{k+1} = \underbrace{(I-AT)^{-1}}_{A_d} W_k + \underbrace{(I-AT)^{-1}BT}_{B_d} u_k$

② Now,

$y_k = Cx_k + Du_k$

(where $x_k = w_{k+1}$)

$y_k = C\left[(I-AT)^{-1}w_k + (I-AT)^{-1}BTu_k\right] + Du_k$

Regrouping: Cd, Dd

$y_k = C(I-AT)^{-1}w_k + \left((I-AT)^{-1}BT + D\right)u_k$

Recall, we want

$A_d = (I-AT)^{-1}$

$B_d = (I-AT)^{-1}BT$

C_d, D_d ← see above!

Oops! Out of room for y_{k+1} above... see above...