ECE 5551 Autumn 2020, OSU

Course Notes

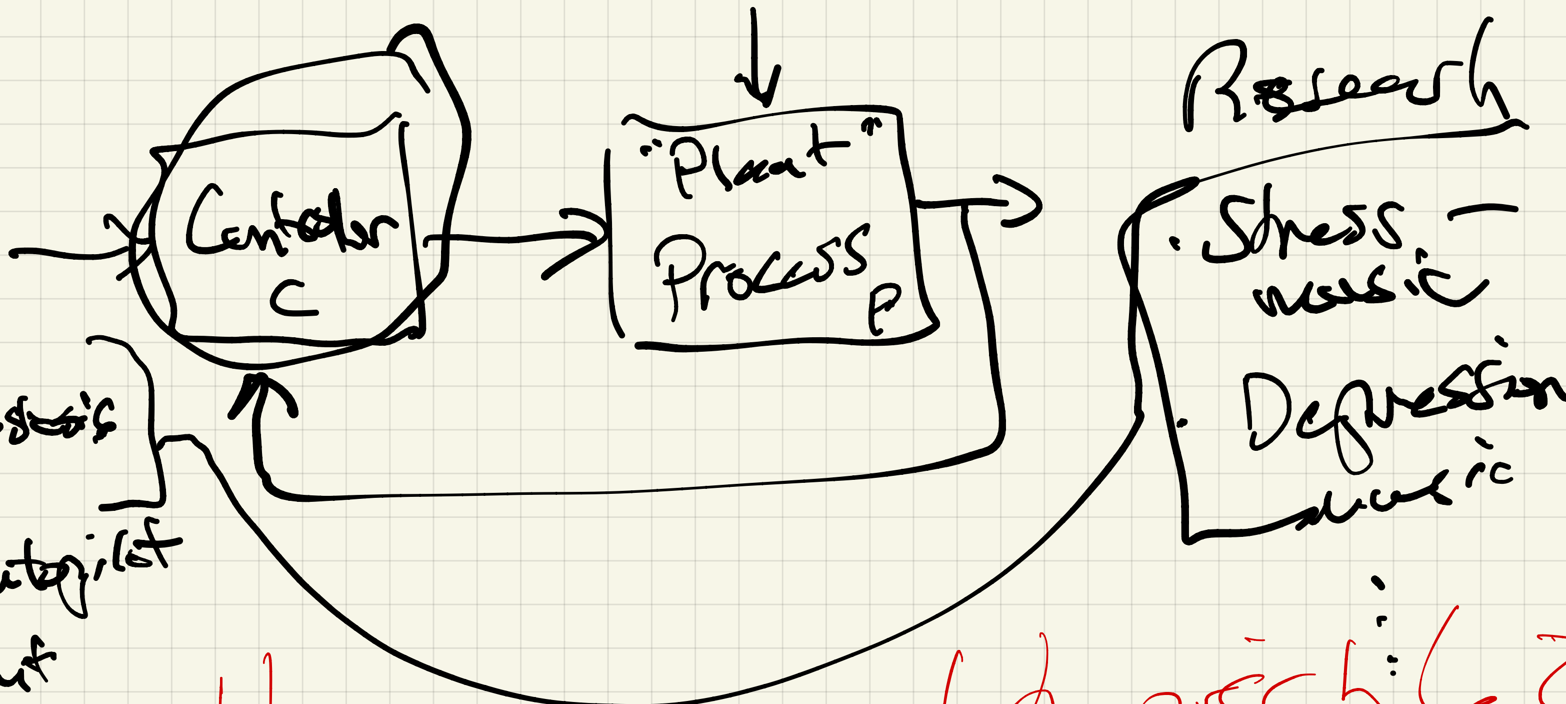
Prof Kevin M Passino

Part 1

Introduction:

3551 - CT - "classical control"
*5551 - DT - "modern control"

- Auto
- A/C
- Cassette
- homeostasis
- Aircraft autopilot
- Traffic light
- Escalator



How is this generality possible?

Generality

- Mathematics – common language
  ↓
- principles
  - Stability
  - Controllability, observability
  - Tracking, disturbance rejection
  - → Robustness (out of scope)

* LTI, DT (easy connect to app., coding)
Matlab .m

Roadmap for Modeling:

Reality
↓
Science (physics, chemistry, biology, ..., social sciences)
↓
Model

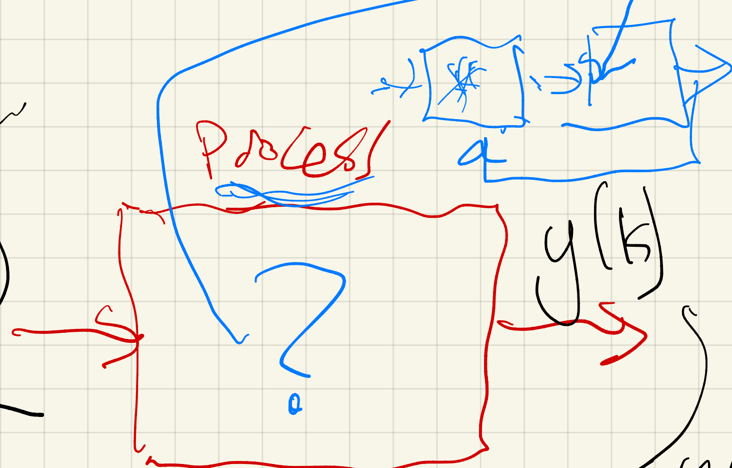
Reality
↓
Experiments
↓
Data
↓
System ID (Machine learning)
↓
Model

$K = 0, 1, 2, \ldots$

$u(k)$

$u(kT)$

Sampled, T



$y(k)$

$y(kT)$

Generating a state space model

Example: From physics, Newton's Second Law

$p = 2$

$q = 3$

$P = 3$

$$F = ma$$

- $F \in \mathbb{R}^p$ (force) (; vector)] vary with t
- $a \in \mathbb{R}^p$ (accel.)
- m, scalar (mass)

Let $v \in \mathbb{R}^p$ be velocity; let $\bar{x} \in \mathbb{R}^p$ be position

$a = \dot{v}$, $\dot{\bar{x}} = v$

$u = F$

SV

$$\boxed{\begin{aligned} \dot{\bar{x}} &= v \\ \dot{v} &= \frac{1}{m} u \end{aligned}}$$

$\rightarrow$ "state" $\quad x = \begin{bmatrix} \bar{x} \\ v \end{bmatrix}$ $2p \times 1$

$$\underset{2p\times1}{\begin{bmatrix}\dot{X}\\ \dot{V}\end{bmatrix}} = \underset{2p\times 2p}{\begin{bmatrix}0_{p\times p} & I_{p\times p}\\ 0_{p\times p} & 0_{p\times p}\end{bmatrix}}\underset{2p\times1}{\begin{bmatrix}X_{p\times1}\\ V_{p\times1}\end{bmatrix}} + \underset{2p\times1}{\begin{bmatrix}0_{p\times q}\\ \frac{1}{m} I_{p\times q}\end{bmatrix}} \, u_{p\times1}$$

$= p\times1$

$p = 2$

$I_{p\times p} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

$p\times p$

$p\times q$

$q =$

Let output $y = v \in \mathbb{R}^p$

$\downarrow$

$$y = \begin{bmatrix}0 & I_{p\times p}\end{bmatrix}\begin{bmatrix}X\\ V\end{bmatrix}$$

$2p\times p$

General

Let $u \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^l$, $w \in \mathbb{R}^m$

"State space model"

$$\boxed{\begin{aligned} \underset{n\times1}{\dot{x}} &= \underset{n\times n}{F}\,\underset{n\times1}{x} + \underset{n\times m}{G}\,\underset{m\times1}{u} + \underset{n\times m}{G_1}\,\underset{m\times1}{w}\\ \underset{l\times1}{y} &= \underset{l\times n}{H}\,\underset{n\times1}{x} + \underset{l\times m}{J}\,\underset{m\times1}{u}\end{aligned}}$$

Initial cond $x(0)$

$\Rightarrow q = p$

Dimensions?

Process

$$u \longrightarrow \boxed{F, G, G_1, H, J \;\; (x)} \longrightarrow y, \qquad w \downarrow$$

Return to $F = ma$ example (discrete)

$$x = \begin{bmatrix} \bar{x} \\ v \end{bmatrix} \in \mathbb{R}^{2p}$$

$$u \in \mathbb{R}^{p}$$

$$y \quad p \times 1$$

$$F = \underbrace{\begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}}_{2p \times 2p} \qquad 0 : p \times p$$

$$G = \begin{bmatrix} 0 \\ \frac{1}{m} I \end{bmatrix}, \quad G_1 = 0$$

$$H = \begin{bmatrix} 0 & I \end{bmatrix}$$

$$J = 0$$

* possible to load the info. into F, G, G_1, H, J

(*) is not a unique representation

Let $\overset{n\times 1}{\xi} = \overset{n\times n}{T}\,\overset{n\times 1}{x}$, T nonsingular

state of "the transformed system"

$$\dot{\xi} = T\dot{x}$$
$$= T(Fx + Gu + G_1 w)$$
$$= TFx + TGu + TG_1 w$$

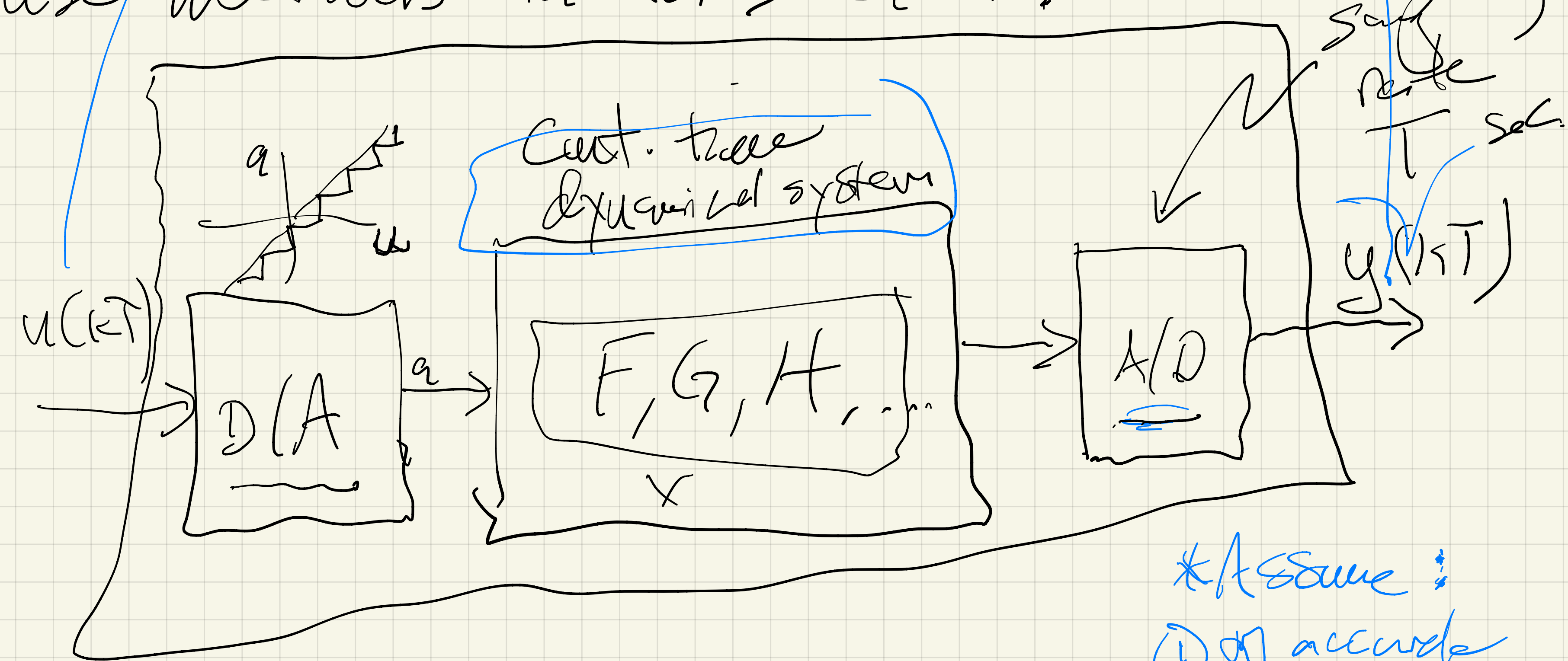
$$\begin{cases} \dot{\xi} = TFT^{-1}\xi + TGu + TG_1 w \\ y = HT^{-1}\xi + Ju \end{cases}$$

∞ number of realizations (depend on T)

$$u \to \boxed{\xi} \to y$$

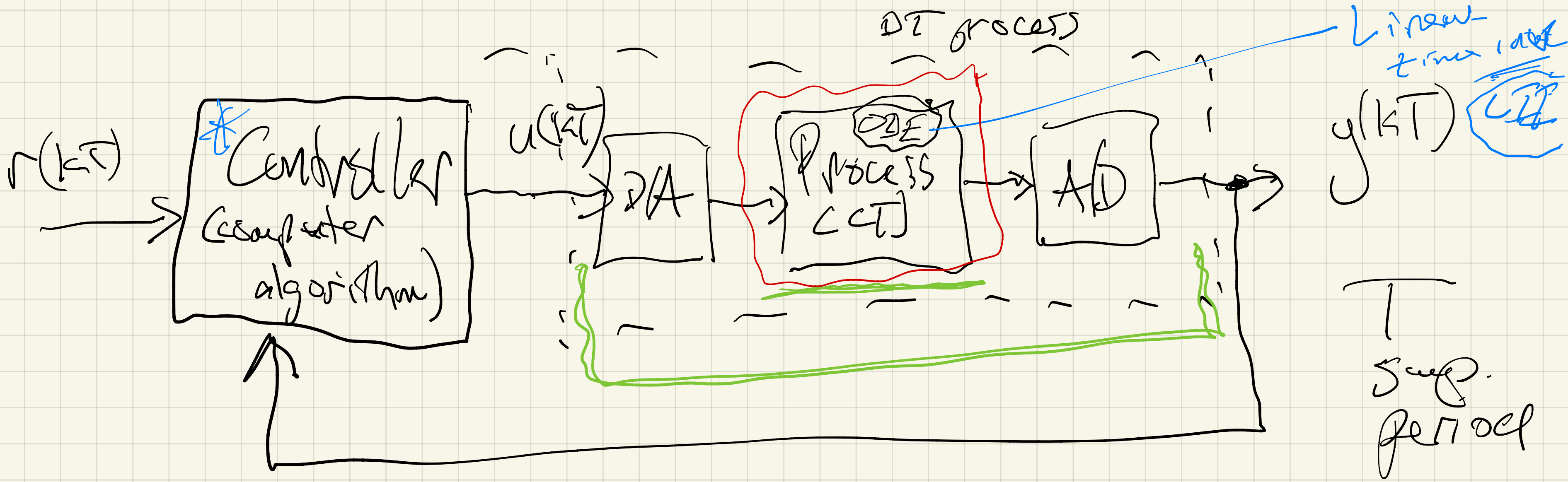
I/O behavior is same

We need a DT SV representation to use methods in this class.



*Assume:
① accurate A/D in level
② We will pick T

* Convenience, we say "u(k)" and "y(k)"



Analysis approaches: z-transform, SS approach
+ Matlab / Simulink for simulation

"homogeneous equation"

$$\dot{x}_h(t) = F x_h(t)$$

$$x_h(t_0) = x_0 \quad , \quad t_0 = \text{initial time}$$

Assume solution is

$$\overset{n\times 1}{x_n(t)} = A_0 + A_1(t-t_0) + A_2(t-t_0)^2 + \dots$$

Let $t=t_0$, $x_n(t_0) = \boxed{A_0 = x_0}$

Differentiate (w.r.t. $t-t_0$)

$$\dot{x}_n(t) = \overset{n\times 1}{A_1} + 2A_2(t-t_0) + \dots = Fx_n$$

and substitute into $\dot{x}_n = Fx_n$

at $t=t_0$, $\boxed{A_1 = Fx_n(0) = Fx_0}$

Repeat this process: (take the second deriv., third....)

$$x_n(t) = \left[I + F(t-t_0) + \frac{F^2(t-t_0)^2}{2} + \frac{F^3(t-t_0)^3}{6} + \dots\right]x_0$$

series - "matrix exponential"

$$x_h(t) = e^{\overset{n\times n}{F}\overset{1\times 1}{(t-t_0)}} x(t_0)$$

$$= \sum_{k=0}^{\infty} F^k \frac{(t-t_0)^k}{k!}$$

Side note:

$$\underbrace{I}_{n\times n} = \underbrace{e^{-\overset{n\times n}{F}\underbrace{(t_1-t_0)}_{1\times 1}}}_{n\times n} \underbrace{e^{\overset{n\times n}{F}(t_1-t_0)}}_{n\times n}$$

Consider $n=1$ case

Particular solution:

$$\dot{x} = Fx + Gu$$

(figure w)

Guess the solution...

$$\overset{n\times 1}{x_p(t)} = \underbrace{\underbrace{e^{\overset{n\times n}{F(t-t_0)}}}_{n\times n} \overset{n\times 1}{v(t)}}_{n\times 1}$$

Substitute into ODE to see if it works

$$\dot{x} = Fx + Gu$$

$$\dot{x}_p = Fx_p + Gu$$

$x(t)\,y(t)$

$$\frac{d}{dt}\left(e^{F(t-t_0)}v(t)\right) = Fe^{F(t-t_0)}v(t) + Gu$$

need product rule

$$Fe^{F(t-t_0)}v(t) + \boxed{e^{F(t-t_0)}\dot{v}(t)} = Fe^{F(t-t_0)}v(t) + Gu$$

Mult. through by $e^{-F(t-t_0)}$

$$\Rightarrow \boxed{\dot{v}(t) = e^{-F(t-t_0)}Gu}$$

ODE

For $u(t) = 0$, $t < t_0$,

$$v(t) = \int_{t_0}^{t} \dot{v}\, d\tau = \int_{t_0}^{t} e^{-F(\tau - t_0)} G u(\tau)\, d\tau$$

"Fundamental thm of calculus"

So,

$$x_p(t) = e^{F(t-t_0)} v(t) = e^{F(t-t_0)} \int_{t_0}^{t} e^{-F(\tau - t_0)} G u(\tau)\, d\tau$$

$$= \int_{t_0}^{t} e^{F(t-t_0)} e^{-F(\tau - t_0)} G u(\tau)\, d\tau$$

$$\boxed{x_p(t) = \int_{t_0}^{t} e^{F(t-\tau)} G u(\tau)\, d\tau}$$

"convolution integral"

Total solution, $x(t)$

LTI

$$x(t) = x_h(t) + x_p(t)$$

(due to superposition)

so CT

$$\underset{n\times 1}{x(t)} = \boxed{\underbrace{e^{F(t-t_0)}}_{n\times n}\ \underset{n\times 1}{x(t_0)}} + \int_{t_0}^{t} e^{F(t-\tau)} G u(\tau)\, d\tau$$

→ solution to the ODE

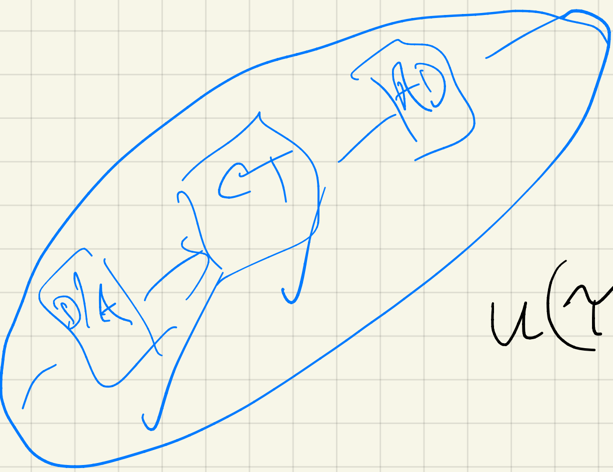
$$\dot{x} = Fx + Gu$$
$$y = Hx + Du$$

Zero-order-hold (ZOH): (sample and hold)

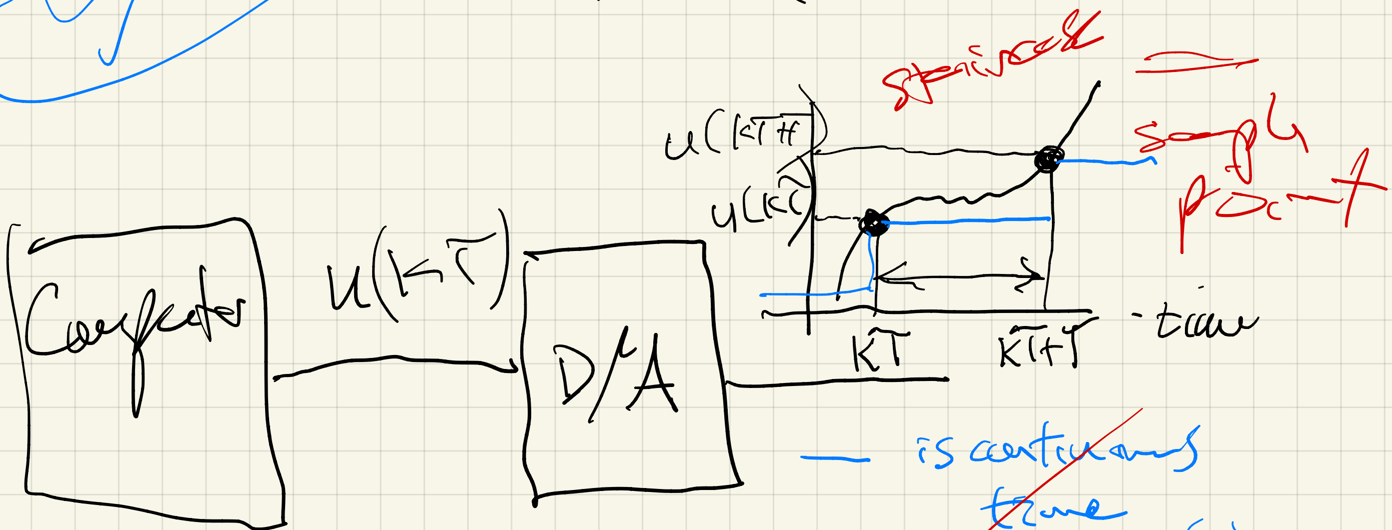
- Defines $u(t)$, $t \geq 0$, in terms of $u(kT)$,

T = sampling period, > 0

$k = 0, 1, 2, \ldots$



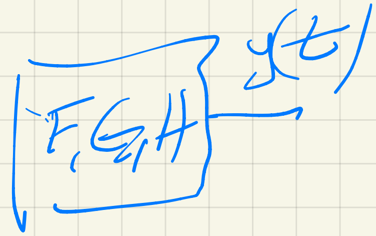
$$u(\tau) = u(kT), \quad kT \leq \tau < kT + T$$

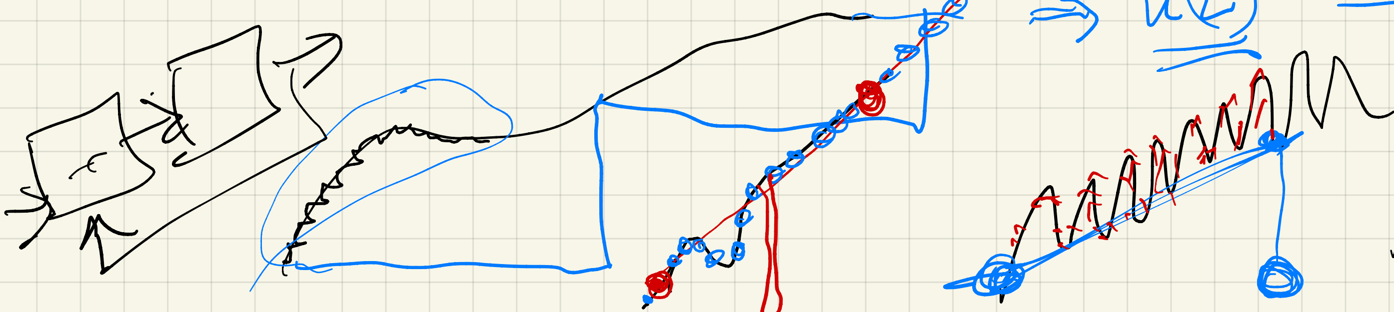


next sample point

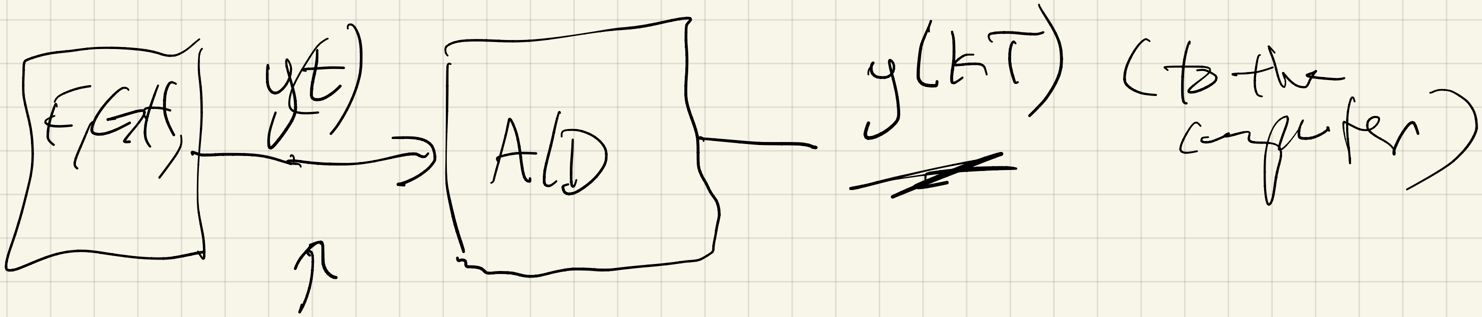
— is continuous time

$\Rightarrow u(t) \rightarrow$ F,G,H $\rightarrow y(t)$





Nyquist samp. Thm.



- Have $y(t)$, $t \geq 0$
- Take samples at times $0, T, \ldots, kT, kT+T, \ldots$

  (clocks on D/A, A/D, computer assumed to be the same)
- $y(kT) = H\,x(kT) + J\,u(kT)$

Completing the derivation...

- Change of variables, $\eta = kT + T - \tau$

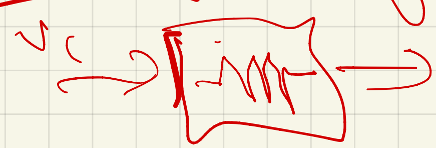
$$x(kT+T) = e^{FT}x(kT) + \left(\int_0^T e^{F\eta}\,d\eta\right) G\,u(kT)$$

Define

$$\Phi = e^{FT}$$

$$\Gamma = \int_0^T e^{F\eta} \, d\eta \; G$$

Circuit



$V_2 = \alpha V_1$

α const

So, drop "T" from kT and kT+T

DT SS

in unit disk

$$\underset{n\times 1}{x(k+1)} = \underset{n\times n}{\Phi} \, \underset{n\times 1}{x(k)} + \underset{n\times m}{\Gamma} \, \underset{m\times 1}{u(k)} + \Gamma_1 w(k)$$

$$\underset{p\times 1}{y(k)} = \underset{p\times n}{H} \, \underset{n\times 1}{x(k)} \quad (+ J u)$$

$x(0)$ = initial condition

$x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$

$y(k) \in \mathbb{R}^p$

$e^{i\Phi}$

e^{FT}

LTI?

Amazing: At sampling point same

Example: Point-mass dynamics (e.g. a ground robot)

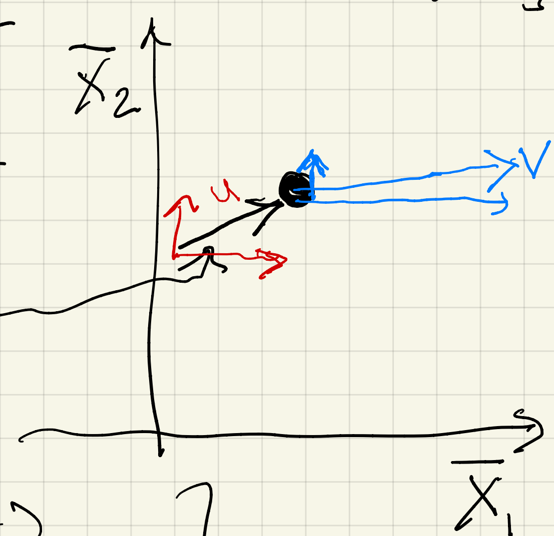
$F = ma \rightarrow$

$\bar{x}$ position $\in \mathbb{R}^2$ $(p = 2)$ $= \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}$

v velocity $\in \mathbb{R}^2$

u force $\in \mathbb{R}^2$

m mass



We showed

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & I_{2\times2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} I_{2\times2} \end{bmatrix} u$$

0*ones(2,2)

eye(2,2)

$$y = \begin{bmatrix} 0 & I_{2\times2} \end{bmatrix} \begin{bmatrix} \bar{x} \\ v \end{bmatrix}$$

(C shows what can be sensed)

modeling sensing velocity only

Let $u = \frac{V}{2}$ $(p = 2)$

$$\dot{x} = \begin{bmatrix} \dot{\bar{X}} \\ \dot{V} \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \bar{X} \\ V \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{X} \\ V \end{bmatrix}$$

Convert to DT

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$

$$y(k) = \underline{H} x(k)$$

no conversion necessary

$$\Phi = e^{FT}$$

(inf. series)

$$\Gamma = \int_0^T e^{F\eta} d\eta \, G$$

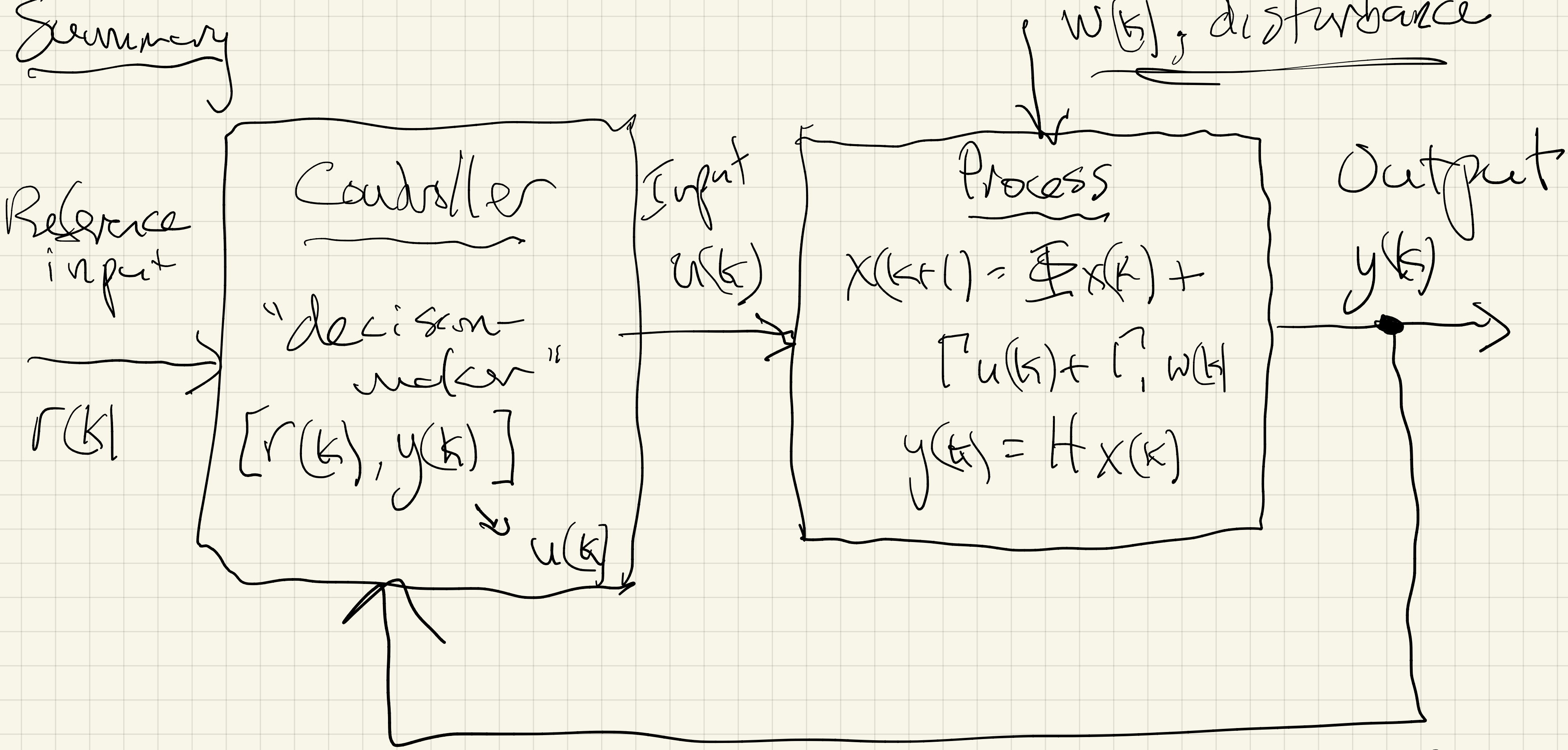
($\int$ of an inf. series)

<u>Matlab:</u> - Choose $T = 0.1$

- See ConvertCTtoDTss.m

(Cat Carmen)

Summary

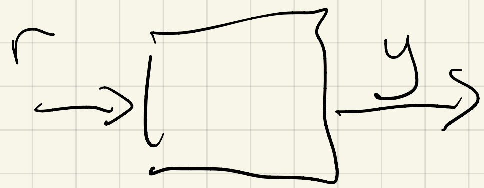


Goal: Good "closed-loop" performance. ⇒ Given the process model, design the controller to achieve the ε.

Closed-loop specifications (performance)

- "Rise time" — how fast $y(k)$ reacts to a step input on $r(k)$
  t_r

(others, "settling time", % overshoot)



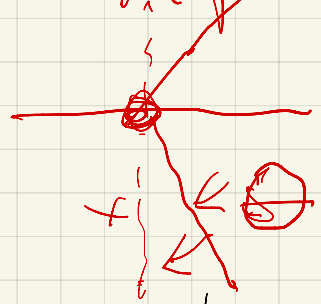
when does $y(k)$ get to within 90% of $r(k)$

- "Stability" — don't want big oscillations or excessive values of variables

- "Disturbance rejection" — keep $y(k)$ close to $r(k)$ no matter what $w(k)$ is

Stability

inverted pendulum



$$\boxed{\dot{x} = Fx + \underbrace{G_2 u + G_1 w}_{=0}}$$

consider "internal stability"

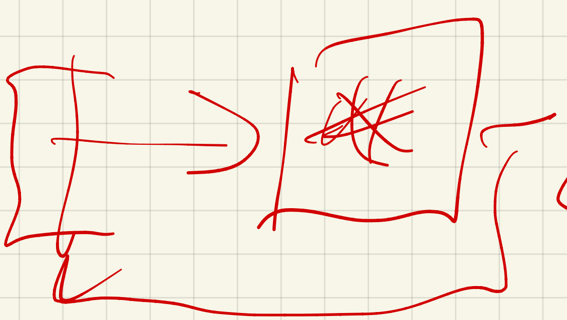
IC: x_0 $\quad (x(0) = x_0)$

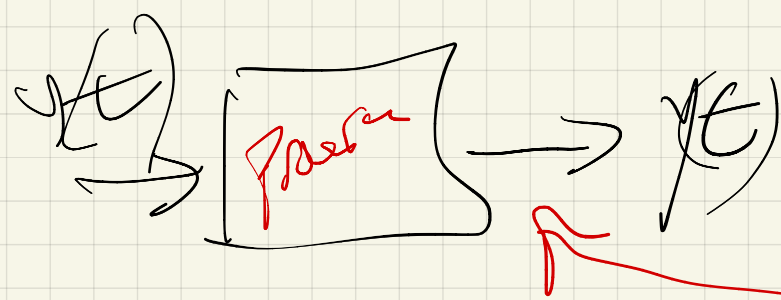
Equilibrium: $\dot{x} = 0, \; t \geq 0$

$\rightarrow$ Find x^* s.t. $\underset{n\times 1}{0} = \underset{n\times n}{F}\,\underset{n\times 1}{x^*}, \quad t \geq 0$

One solution: $x^* = 0$

x^* is asymptotically stable if





Normally for I/O stability focus on 3130

① $\forall \varepsilon > 0, \ \exists \delta > 0$ st. if

$$|x_0 - x^*| < \delta$$

Then $|x(t) - x^*| < \varepsilon, \ t \geq 0$

$\forall$ = "for all"

$\exists$ = "There exists"

$z \in \mathbb{R}^n$

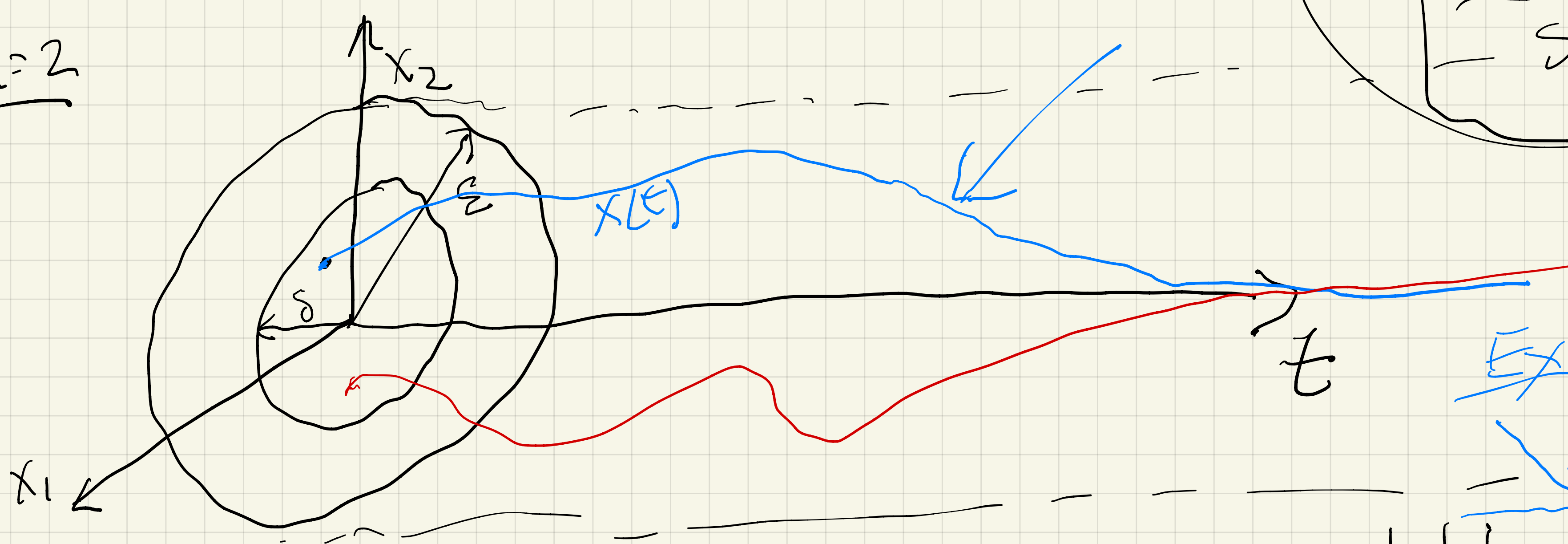
$|z| = \sqrt{z^T z}$

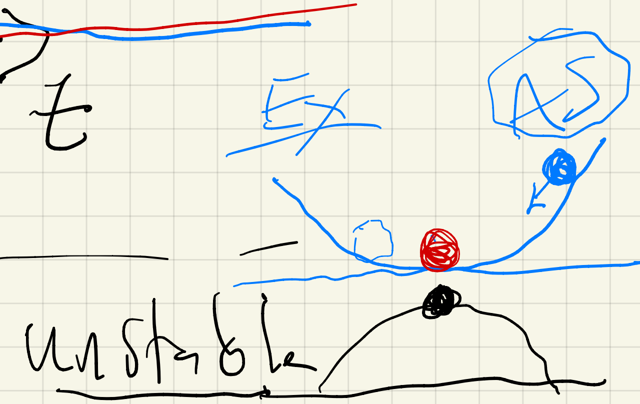
"distance"

and ② $\lim_{t \to \infty} x(t) = x^* = 0 \quad \forall x_0$

For linear or nonlinear systems

$n = 2$



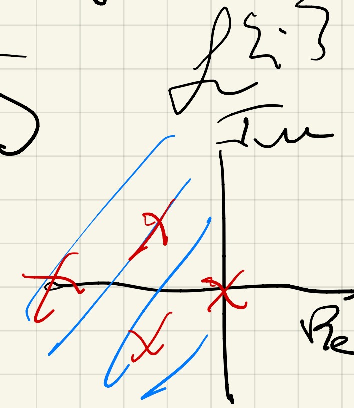


Linear case....

Computationally:

① If eigs. of F are in the strict LHP $\Rightarrow x^*$ is AS

(eig(F) ...)



CT

challenge:

Given $\dot{x} = Fx$

(or $x(k+1) = \Phi x(k)$)

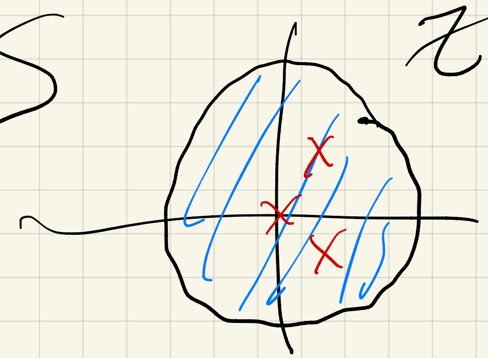
is this system A. Stable ?

② If eigs. of Φ are in the unit disk $\Rightarrow x^*$ is AS

(eig Φ)

strictly

DT



Where are we at?

Physics

$\downarrow$ modeling

Linear

CT $\begin{bmatrix} \dot{x} = Fx + G u + G_1 w \\ y = Hx \end{bmatrix}$ ODE

$\downarrow$ CT $\rightarrow$ DT

DT $\begin{bmatrix} x(k+1) = \Phi x(k) + \Gamma u + \Gamma_1 w \\ y = Hx \end{bmatrix}$

Today

Physics

$\downarrow$

(Nonlinear model)

$\downarrow$

Linear model (CCT $\to$ DT)

To start

- General case
- Example

Nonlinear systems:

Examples:

- From first day of class, all those systems were nonlinear
- Robot, ground ($p=2$): Model came from $F = ma$

Linear all by itself

* Also, the outputs are limited in size

$$F = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \Rightarrow \boxed{\text{Robot}} \rightarrow \left.\begin{matrix} x_1 \\ x_2 \end{matrix}\right] \text{position} \quad \left.\begin{matrix} v_1 \\ v_2 \end{matrix}\right] \text{velocity}$$

* input f_1 and f_2 are limited in magnitude $\Rightarrow$ Nonlinear

Given n first-order ODEs:

state equations
$$\begin{cases} \dot{x}_1 = f_1(\underbrace{x_1, \ldots x_n}_{\text{"state"}}, \underbrace{u_1, \ldots, u_m}_{\text{inputs}}, \underbrace{t}_{\text{time}}) \\ \vdots \\ \dot{x}_n = f_n(x_1, \ldots, x_n, u_1, \ldots, u_m, t) \end{cases}$$

output equations
$$\begin{cases} y_1 = h_1(\quad \cdot \quad) \\ \vdots \\ y_p = h_p(\quad \cdot \quad) \end{cases}$$

or

$$\begin{cases} \underset{n\times 1}{\dot{x}} = \underset{n\times 1}{f}(\underset{n\times 1}{x}, \underset{m\times 1}{u}), \quad \underset{n\times 1}{x(t_0)} = \underset{n\times 1}{x_0} \\ \underset{p\times 1}{y} = \underset{p\times 1}{h}(\underset{n\times 1}{x}, \underset{m\times 1}{u}) \end{cases}$$

- Let $\overset{n\times 1}{x_0}$ and $\overset{m\times 1}{u_0}$ be "reference values"

- Assume $x(t)$ is close to x_0 and
  $u(t)$ is close to u_0, $t \geq 0$
  ("small signal assumption")

- Assume x_0 is an "equilibrium point" for a fixed value of u_0

so $f(x_0, u_0) = 0 \quad (= \dot{x})$

If x and u are close to x_0 and u_0, write

$$\underset{n\times 1}{x} = x_0 + \boxed{\delta x}$$

$$\underset{m\times 1}{u} = u_0 + \boxed{\delta u}$$

often called a "perturbation" off x_0, u_0

Substitute into ☆

$$\frac{d}{dt}(x_0 + \delta x) = f(x_0 + \delta x,\ u_0 + \delta u)$$

$$\dot{\delta x} = f(x_0, u_0) + f_x(x_0,u_0)\,\delta x + f_u(x_0,u_0)\,\delta u + \cancel{\text{HOT}}$$

Use Taylor ser. expansion

where $\left(\underline{\underline{X}}^{n\times 1}\right)$

$$\underset{n\times n}{f_x} = \begin{bmatrix} \overset{1\times 1}{\dfrac{\partial f_1}{\partial X_{n\times 1}}} \\ \dfrac{\partial f_2}{\partial X} \\ \vdots \\ \dfrac{\partial f_n}{\partial X} \end{bmatrix} \begin{matrix} \text{"gradient" } 1\times n \\ 1\times n \\ \\ 1\times n \end{matrix}$$

$i = 1, \dots, n$

$$\underset{1\times n}{\frac{\partial f_i}{\partial X}} = \left[\underset{1\times 1}{\underline{\underline{\frac{\partial f_i}{\partial X_1}}}}, \dots, \underline{\underline{\frac{\partial f_i}{\partial X_n}}} \right]$$

$$\underset{n\times m}{f_u} = \begin{bmatrix} \dfrac{\partial f_1}{\partial u} \\ \vdots \\ \dfrac{\partial f_n}{\partial u} \end{bmatrix},$$

$$\underset{1\times m}{\frac{\partial f_i}{\partial u}} = \left[\frac{\partial f_i}{\partial u_1}, \dots, \frac{\partial f_i}{\partial u_m} \right]$$

Evaluate at x_0, u_0

$$\dot{\delta x} \approx f_x(x_0, u_0)\, \delta x + f_u(x_0, u_0)\, \delta u$$

$$\delta y = h_x\, \delta x + h_u\, \delta u$$

Simplify notation $\rightarrow$ drop the δ and define matrices

$$F = f_x(x_0, u_0), \qquad G = f_u(x_0, u_0)$$

$$H = h_x(x_0, u_0), \qquad J = h_u(x_0, u_0)$$

$$\Rightarrow \boxed{\begin{aligned} \dot{x} &= Fx + Gu \\ y &= Hx + Ju \end{aligned}}$$

Careful! x is actually δx!

This represents dynamics of δx

Taylor series exp. above was valid for small δx, δu $\Rightarrow$ if trajectories δx, δu stay small the linear approximation is valid (otherwise, not!)

Above is CT, can convert to DT

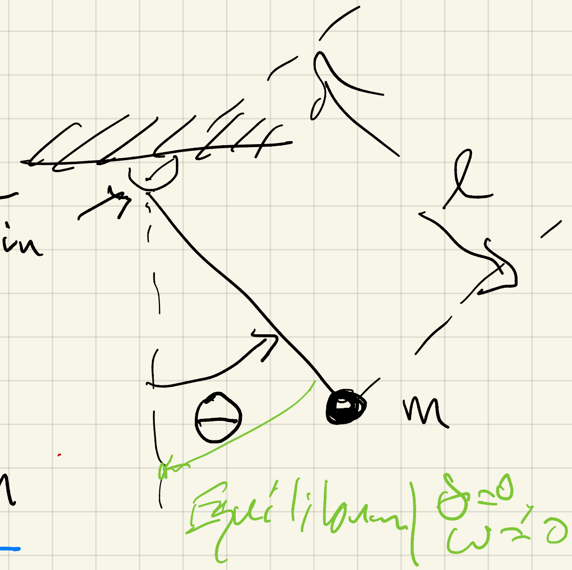
$$\begin{cases} x(k+1) = \Phi x(k) + \Gamma u(k) + \Gamma_1 w(k) \\ y = H\, x(k) \end{cases}$$

can do stability analysis here easily

Example: Pendulum

— = linear

— = nonlinear



ODE

$$\dot{\theta} = \underline{\omega}$$

$$\dot{\omega} = \underline{-b\omega} - \underline{\Omega^2 \sin\theta} + \underline{\frac{1}{m\ell^2} T_{in}}$$

For small θ, $\sin\theta \approx \theta$

$b =$ viscous friction

$$\Omega = \sqrt{g/\ell}$$ oscillation frequency for small initial conditions

"state vector"

$$\overset{2\times 1}{x} = \begin{bmatrix} \theta \\ \omega \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} \omega \\ -b\omega - \Omega^2 \sin\theta + \frac{1}{m\ell^2} u \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

$$y = \underbrace{\theta}_{h}$$

ODE solver

Runge Kutta ← Matlab

Could use Simulink

Equilibrium, reference values:

- Find one, in general there are many

- $\dot{x} = 0 = \begin{bmatrix} \omega \\ -b\omega - \Omega^2 \sin\theta + \frac{1}{ml^2} u_0 \end{bmatrix}$

$\Rightarrow \omega = 0$

Choose $u_0 = 0$

$\Rightarrow \theta = 0$

$x_0 = \begin{bmatrix} \omega_0 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, equilibrium point

$$f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ -\Omega^2 \cos\theta & -b \end{bmatrix} \Bigg|_{(x_0, u_0) = (0,0)} = \begin{bmatrix} 0 & 1 \\ -\Omega^2 & -b \end{bmatrix}$$

$$f_u = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}$$

" From only. small neighborhood of x_0, u_0 "

$$h_x = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$h_u = \begin{bmatrix} \frac{\partial h}{\partial u} \end{bmatrix} = 0$$

Skip the "δ":

matlab, linear solvers

Valid only for when δx, δu are "small"

$$\delta \dot{x} = \begin{bmatrix} 0 & 1 \\ -\Omega^2 & -b \end{bmatrix} \delta x + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \delta u$$

$$\delta y = \begin{bmatrix} 1 & 0 \end{bmatrix} \delta x$$

Linear system

Now, we learn

Nonlinear model

linearization

Linear model

$$\dot{x} = Fx + G_2 u + G_1 w$$
$$y = Hx$$

$$\Downarrow$$

Use this in class

$$\begin{cases} x(k+1) = \Phi x(k) + \Gamma u(k) + \Gamma_1 w(k) \\ y(k) = H x(k) \end{cases}$$

Next, "system ID"

System identification (lots of tools in Matlab)

- Aim at getting a model, linear DT

Physics approach to getting a model

→ physics can help set model struct

SysID help set parameter values

ML

Physical reality

Experiments

(I/O) Data

Least squares (other optim)

Linear model

Auto-regressive moving average (ARMA) model

"autoregressive"

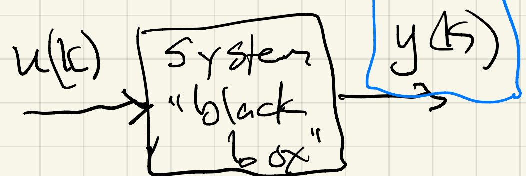
$n > 0$, $k \ge 0$

$$y(k) + a_1 y(k-1) + a_2 y(k-2) + \cdots + a_n y(k-n)$$

$$= b_1 u(k-1) + \cdots + b_n u(k-n)$$

"moving average"

$k \to kT$



$k \ge 0$

$$\frac{1}{n}\sum_{i=1}^{n} u(k-i)$$

- This is a linear DT model
- Assumes $T > 0$.

$a_1, \dots, a_n$, $b_1, \dots, b_n$ — are not known

(sometimes this "n" is replaced with $m \le n$)

Note: In practice, n and m are not known

$\Rightarrow$ if pick $b_i = \frac{1}{n}$, $i = 1, \dots, n$

$\therefore$ it's MA

Create a DT SS model

- Choose the state

$$x(k) = [\, y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-n) \,]^T$$

e.g. $n = 3$

$$\Phi = \begin{bmatrix} -a_1 & -a_2 & -a_3 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

check this

HW find it

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$

$$H = [\, -a_1 \; -a_2 \; -a_3 \; b_1 \; b_2 \; b_3 \,]$$

If know $a_1, a_2, a_3, b_1, b_2, b_3 \Rightarrow$ know Φ, Γ, H

For the $n = 3$ case,

$u(k) \rightarrow$

$$\boxed{H(z) = \frac{b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3}}$$

$\rightarrow y(k)$

$\Rightarrow$ 2 zeros

3 poles

6 parameters to describe the model

system ID

$\downarrow$ a_i, b_i

Experiment:

$$u(k) \longrightarrow \boxed{\begin{array}{c} ? \\ \text{(assume linear system)} \end{array}} \longrightarrow y(k) \qquad k \geq 0$$

Measure set of inputs & outputs

$$\{ -y(0), -y(1), \ldots, -y(N), u(0), \ldots, u(N) \}$$

Example Experimental construction of the Bode plot

plant $G(s)$

System ID

choose $u(t) = A\sin(\omega t)$ for a series of ω, say ω_i

$\Rightarrow y(t) = A' \sin(\omega t + \phi)$,

Bode plot — $\begin{cases} 20 \log M \text{ dB} \\ \phi \to \text{deg.} \end{cases}$

$$M(\omega) = \left| \frac{A'}{A} \right|$$

Objective:

"parameter vector"

$$\theta = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^{2n}$$

Find these, using $u(k)$, $y(k)$

- Unknown
- Want it to "best fit" the data
- Assumes linearity, and that we know n

Let

$$y(k) + a_1 y(k-1) + \dots + a_n y(k-n)$$
$$- b_1 u(k-1) - b_2 u(k-2) - \dots - b_n u(k-n)$$
$$= e(k; \theta)$$

- "error" in the linear model, given n, to fit the data
- errors are for a specific k, for one parameter vector θ

Error $e(k;\theta)$ depends on past values of y, u

$\Rightarrow$ The first e we know is $\boxed{e(n, \theta)}$ $k=n$

The "state" of the ARMA model (up to a minus sign)

is

"regression vector"

$$\phi(k) = [-y(k-1) \ \cdots \ -y(k-n) \ u(k-1) + \cdots + u(k-n)]^T$$

Consider $k = n, n+1, \ldots, N$

$$\underset{1\times 1}{y(n)} = \underbrace{\phi^T(n)}_{1\times 2n} \underset{2n\times 1}{\theta} + \underset{1\times 1}{e(n;\theta)}$$

$$y(n+1) = \phi^T(n+1)\theta + e(n+1;\theta)$$

$$\vdots$$

$$y(N) = \phi^T(N)\theta + e(N;\theta)$$

$(N-n+1) \times 1$

$$Y(N) = [\overset{1\times 1}{y(n)}, \ldots, \overset{1\times 1}{y(N)}]^T$$

number of output data points we have

$(N-n+1) \times 2n$

$$\Phi(N) = \underbrace{[\overset{2n\times 1}{\phi(n)} \quad \overset{2n\times 1}{\phi(n+1)} \quad \cdots \quad \overset{2n\times 1}{\phi(N)}]}_{\#\text{ columns } = N-n+1}{}^{T}$$

$(N-n+1) \times 1$

$$\varepsilon(N;\theta) = [\overset{1\times 1}{e(n;\theta)} \qquad \overset{1\times 1}{e(N;\theta)}]^T$$

Gives

$$\underbrace{Y = \Phi\,\theta}_{\text{linear model}} + \underbrace{\boxed{\varepsilon(N;\theta)}}_{\text{error due to use of a linear model with assumed order } n}$$

Do not know this

Do not know it

Least squares approach to find Θ

- choose a "cost function" ("bigger is bad")

$$J(\theta) = \underline{\varepsilon^T(N;\theta)}\, \varepsilon(N;\theta)$$

$$= [\quad]\begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$\boxed{J(\theta) = \sum_{k=n}^{N} \underbrace{e^2(k;\theta)}_{1\times 1}}$$

"sum of the squares"

* [We want $J(\theta)$ to be as small as possible.]

other approaches: $\sum_{k=n}^{N} |e(k,\theta)|$

Assume θ^0 is the best parameter vector estimate

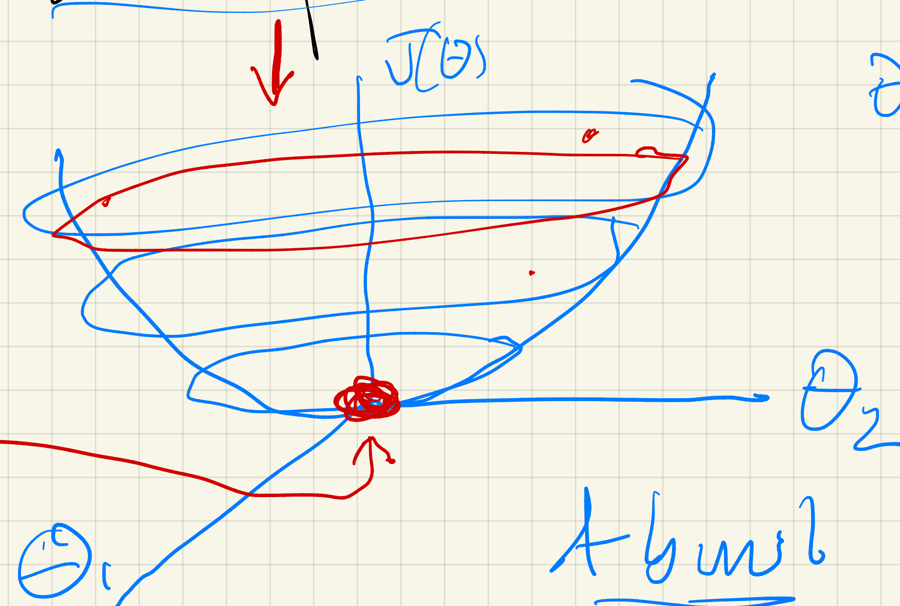
Find $\hat{\theta}_{LS}$, the (best) least squares estimate of θ^0 s.t.

$$J(\hat{\theta}_{LS}) \leq J(\theta), \quad \forall \theta$$

$\underbrace{\overset{1\times 1}{J(\theta)}}$ is quadratic in $2n$ parameters in θ

Consider $n=1$:

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$



✳ A "globally optimal solution"

A bowl

A necessary condition on $\hat{\theta}_{LS}$ is

$$\left.\frac{\partial J}{\partial \theta}\right|_{\theta = \hat{\theta}_{LS}} = 0$$

(slope of the bottom of the bowl = 0)

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}$$

$$\boxed{J(\theta) = \underbrace{\varepsilon^T \varepsilon}}$$

("errors, sum of squares of error)

$$\begin{bmatrix} \varepsilon_1 & \varepsilon_2 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} = \varepsilon_1^2 + \varepsilon_2^2$$

$$= (Y - \Phi\theta)^T (Y - \Phi\theta)$$

$$= Y^T Y - \underbrace{\theta^T \Phi^T Y} - Y^T \Phi \theta + \theta^T \Phi^T \Phi \theta$$

Note

Given $a^T Q a$, $\frac{\partial a^T Q a}{\partial a} = 2a^T Q$

For home: Show it for the 2-dimensional case

$a \in \mathbb{R}^2$, $Q \in \mathbb{R}^{2\times 2}$

$\overset{1\times 1}{J(\theta)}$

$$\frac{\partial J(\theta)}{\partial \theta} = \underbrace{\frac{\partial}{\partial \theta}\left(Y^T Y\right)}_{=0}$$

$$-\frac{\partial}{\partial \theta}\left(\theta^T \overset{1\times 2n}{\Phi^T} \overset{2n\times 1}{Y}\right) \Rightarrow -Y^T \Phi$$

This is a transpose of

$\frac{d}{dx}(ax)$, a const $= a$

$$-\frac{\partial}{\partial \theta}\left(Y^T \Phi \theta\right) \Rightarrow -Y^T \Phi$$

$$+\frac{\partial}{\partial \theta}\left(\theta^T \Phi^T \Phi \theta\right) \rightarrow 2\theta^T \Phi^T \Phi$$

So,

$$\overbrace{\frac{\partial J(\theta)}{\partial \theta}}^{1\times 2n} = -2\,\overbrace{Y^T \Phi}^{1\times 2n} + 2\,\overbrace{\theta^T}^{1\times 2n}\overbrace{\Phi^T \Phi}^{2n\times 2n}$$

Let this $= 0 \triangleq \frac{\partial J}{\partial \theta}$

∴ this gives $\theta = \hat{\theta}_{LS}$ $\underline{2n\times 1}$

Take transpose of both sides ... reconsider the dimensions

$$\Rightarrow \boxed{\underbrace{\Phi^T \Phi}_{\text{known}} \underbrace{\hat{\theta}_{LS}}_{\text{unknown}} = \underbrace{\Phi^T Y}_{\text{known}}}$$

Solve for $\hat{\theta}_{LS}$?

A set of $2n$ equations in $2n$ unknowns

Depends...

- How was n selected?
- What is the sequence of $u(k)$, $k \geq 0$ that was used?

- Need "persistent excitation" (PE) a "sufficiently rich" signal
- If $u(k)$ is PE, then

$$\Phi^T \Phi \text{ is } \underline{\underline{\text{invertible}}}$$

(oops, $\Phi^T \Phi$ ~~may~~ $\underline{\underline{\text{must}}}$ be invertible !!!)

- $\underline{\underline{\text{In practice}}}$: (i) $u(k)$ be white noise (excite the whole freq. spectrum)

② Sum of sinusoids cover the freq. range of interest

Assume u is PE so there is one and only one θ that makes $J(\theta)$ a minimum...

$$\boxed{\hat{\theta}_{LS} = \underbrace{(\Phi^T \Phi)^{-1} \Phi^T Y}_{\text{known}}}$$

Computational issues - sometimes there is a problem with computing $(\Phi^T \Phi)^{-1}$

① Matlab: Use $\text{inv}(\Phi'\Phi)$?

Often, no.

$$\hat{\theta}_{LS} = \text{inv}(\Phi'\Phi) * \Phi' * Y$$

* Could just do this ; check the result

② Matlab:

$$\theta_{LS} = (\Phi' * \Phi) \backslash \Phi' * Y$$

= "left division"

③ Singular value decomposition (svdtester.m in Canvas)

Summary: Models

"physics approach"

physical reality

science (physics)

Nonlinear model

$\dot{x} = f(x,u)$

$y = h(x,u)$

Linear model

$\dot{x} = Fx + Gu + G_1 w$

$y = Hx + Ju$

$\rightarrow G(s)$

"linearization"

DT

$$x(k+1) = \Phi x(k) + \Gamma u(k) + \Gamma_1 w(k)$$
$$y(k) = Hx(k) + Ju(k)$$

"system identification approach"

Experiments

$\rightarrow$ input-output data

BLS

$G(z)$

ARMA

Control System Design Using (DT) State Space Models

Process ("plant"): $T > 0$

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = H x(k)$$

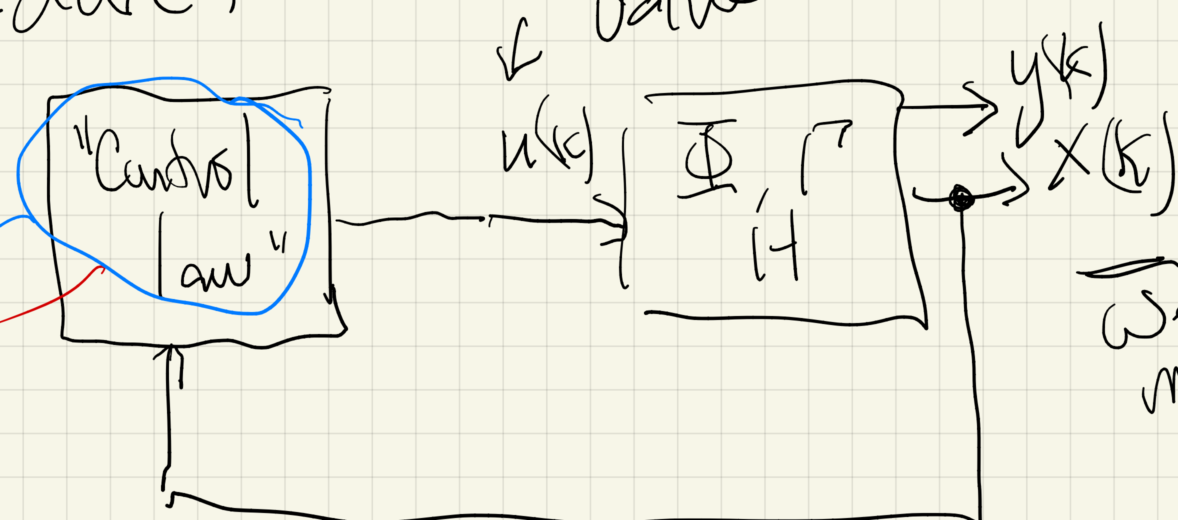
$w = 0$

$\Gamma_1 = 0$

$J = 0$

General procedure:

1



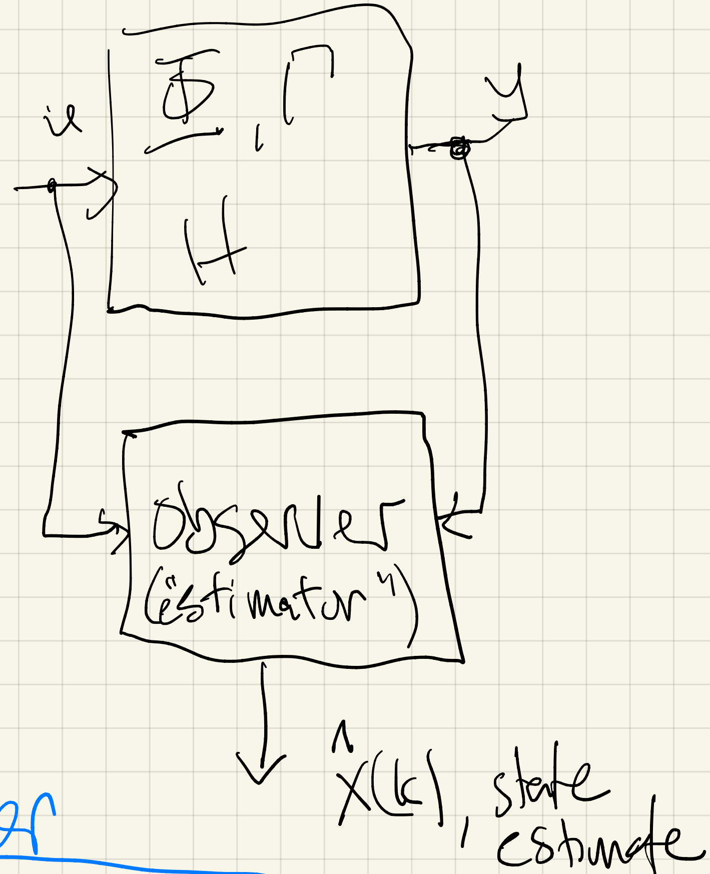
We can measure these

Synthesize this

⇒ implement in code

Start: model
Goto meth

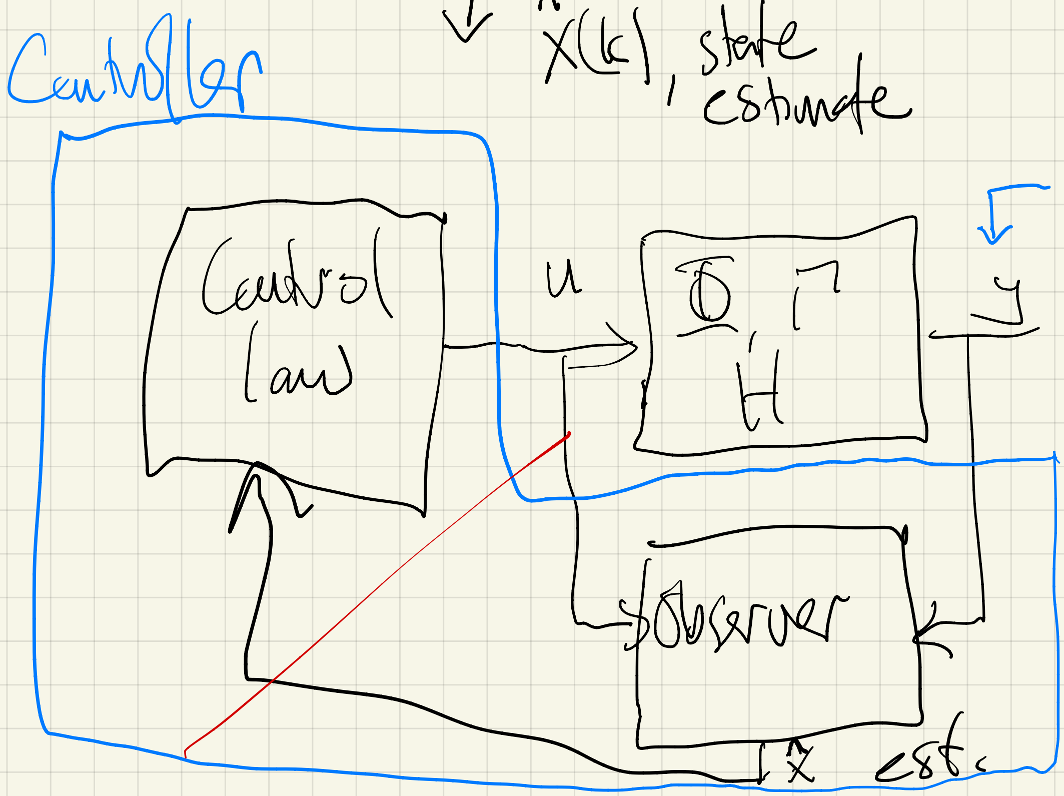
2



"Separation principle"

– we can separate control law design from observer design – then combine at the end.

3



Does not assume x is measurable, just $y = Hx$

Control law: Choice – common

$$\underset{m\times 1}{u} = -\underset{m\times n}{K}\,\underset{n\times 1}{x} = -\underbrace{\overset{n}{\overbrace{\begin{bmatrix} K_1 & K_2 & \cdots & K_n \end{bmatrix}}}}_{m\times n}\underset{n\times 1}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}$$

$K_1 = m\times 1$

To start, let $m=1$ (that is, one input)

$$\Rightarrow \boxed{u = -K_1 x_1 - K_2 x_2 - \cdots - K_n x_n}$$

states assumed to be meas.

(linear combination of "state variables")

gains to be chosen

"control design"

* Does not allow for the reference input $r(t)$ ("goal")

~ assume $r(t) = 0$

Substituting

$$x(k+1) = \Phi x(k) + \Gamma(-Kx(k))$$

$$x(k+1) = (\Phi - \Gamma K)\underline{x(k)}$$

Its z-transform

$$(zI - \Phi + \Gamma K)X(z) = 0$$

∴ characteristic eq.

$$\boxed{\beta_c(z) =} \det|zI - \Phi + \Gamma K| = 0$$

solutions give the eigenvalues of $\Phi - \Gamma K$

Matlab: $\mathrm{eig}(\underset{n\times n}{\Phi} - \underline{\Gamma K})$

Pole placement

$m = 1$,

- n gains in K
- n eigenvalues for ⊛

Pick desired closed-loop root locations (eigenvalue locations in ⊛)

Your pick!

roots: z_i : $\beta_1, \beta_2, \ldots, \beta_n$

or $\alpha_c(z) = (z-\beta_1)(z-\beta_2)\cdots(z-\beta_n) = 0$

Find K to make $\beta_c(z) = \alpha_c(z)$ ✗

In practice:

1. Equate coefficients in ✗ and solve for K_i values $i = 1, 2, \dots, n$

2. Go to Matlab → use "place()"

Example ($n=3$):

$$\Phi = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$H = [b_1 \;\; b_2 \;\; b_3]$$

Characteristic polynomial:

$$a(z) = z^3 - a_1 z^2 - a_2 z - a_3 \Bigg]$$ Gives the poles of the plant

$$\underline{\underline{\Phi - \Gamma K}} = \begin{bmatrix} -a_1 - K_1 & -a_2 - K_2 & -a_3 - K_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\beta_c(z) = z^3 + \underline{(a_1 + K_1)}\, z^2 + (a_2 + K_2)\, z + (a_3 + K_3) = 0 \quad ①$$

Choose the desired closed-loop root locations

$$z^3 + \underline{\alpha_1}\, z^2 + \alpha_2 z + \alpha_3 = 0 \quad ②$$

Equate terms of ① and ② ; solve for K_i

$a_1 + K_1 = \alpha_1$ or $\boxed{K_1 = \alpha_1 - a_1}$

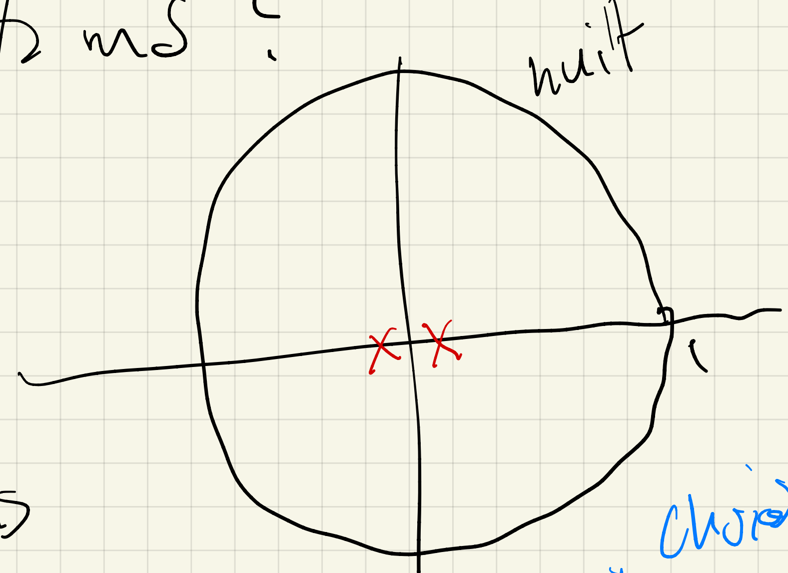
$\boxed{K_2 = \alpha_2 - a_2}$ $\boxed{K_3 = \alpha_3 - a_3}$

This gives the control law $u = -Kx$

$$K = [K_1 \ K_2 \ K_3]$$

* This is a very general procedure, one that works for $n \geq 1$

* Consider, you choose closed loop root locations!



Limitations due to physics

This flexibility is good but it demands careful choice of locations

choice

* But you can make unrealistic

* Often, the problem is how to pick locations that will give both good CL behavior and realistic behavior

Example: Suppose all roots of the open-loop CE

$$\det|zI - \Phi| = 0$$

are distinct $\Rightarrow$ can write

$$X(k+1) = \underbrace{\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}}_{n\times n} \underset{n\times 1}{X(k)} + \overset{\Gamma}{\begin{bmatrix} \Gamma_1 \\ \vdots \\ \Gamma_n \end{bmatrix}} \underset{1\times 1}{u(k)}$$

"diagonalization"

* No element of Γ_i, $i = 1, 2, \ldots, n$, can be zero

If $\Gamma_j = 0$ (some j), then $u(k)\Gamma_j = 0$ $\Rightarrow$ the control cannot affect $x_j(k)$

(can lose control of the system)

Raises the issue of "controllability"

can pick inputs to steer the state

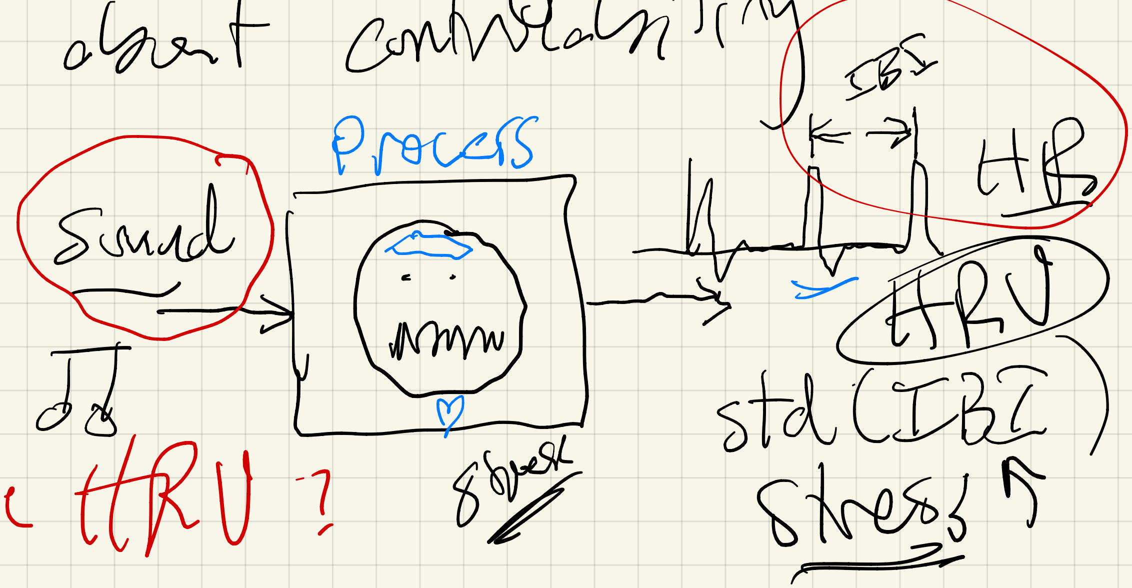
We must have a process that is controllable to do pole placement

1 Matlab:

- ctrb to test controllability
  → yes/no answer
- place() – it gives either a warning or error saying the system is not controllable

② Practice:

- Contructing the model of the process — test for controllability — may have to physically redesign so you controllability
- Often, via process knowledge, can "take a guess" about controllability
- Example 1:



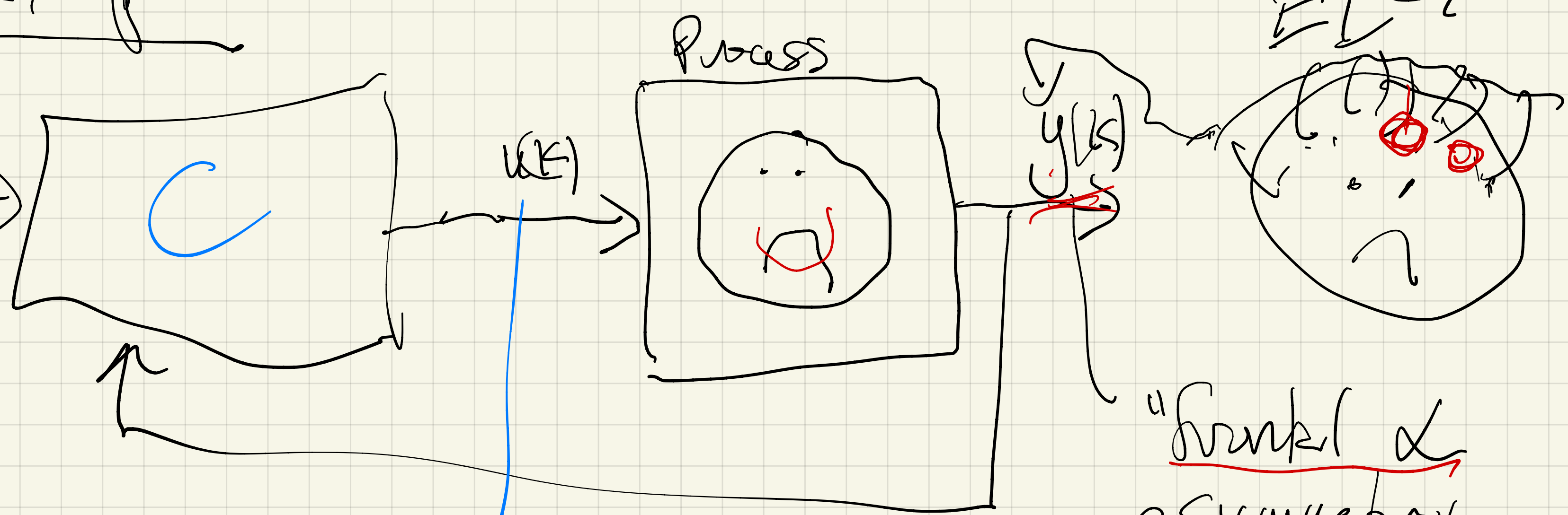
~~Stress control~~

↳ Controllability!

* Can sound modu HRV?

• Example 2:

Depression alleviation



"frontal α asymmetry prop. to level of depression

Want to reduce y(k)

* ① music/sound
② light

Controllability for

* Need controllability to be able to do pole placement

Ackerman's Formula:

Can show that

$$K = \underbrace{[0 \cdots\cdots 0 \; 1]}_{1\times n} \underbrace{[\Gamma \;\; \Phi\Gamma \;\; \Phi^2\Gamma \cdots \Phi^{n-1}\Gamma]^{-1}}_{} \alpha_c(\Phi)$$

C, controllability matrix

(matlab can use C directly via ctrb(·))

Substitute Φ for z in $\alpha_c(z)$

$$\alpha_c(\Phi) = \Phi^n + \alpha_1 \Phi^{n-1} + \alpha_2 \Phi^{n-2} + \ldots + \alpha_n I$$

α_i's are coeff of the desired CLCE

$$\alpha_c(z) = \det|zI - \Phi + \Gamma K| = z^n + \alpha_1 z^{n-1} + \dots + \alpha_n$$

Notes

① If z is replaced by s above, Φ with F, Γ with G, get CT pole placement

② Using Matlab

- For $n \geq 10$ Ackermann's formula breaks down comp. (problem often in $[\cdot]^{-1}$)
- A numerically stable approach in Matlab is "place(.)"

Estimator Design (also, "observer design")

Why?

- Control law $u = -Kx$, assumes "full state feedback" $x \in \mathbb{R}^n$

  Many (or most) systems — not possible to measure x
  (or at least it is expensive)

- Can we estimate the state x so it can be used in the control law?

- "Estimator" provides
  - Estimate of states not sensed
  - Provide smoothed estimates of x even ones that are sensed.

Two types of estimates of $x(k)$

- "current" estimate, $\hat{x}(k)$, based on measurements $y(k)$ up to and including the k^{th} instant
- "predictor" estimate, $\bar{x}(k)$, is based on measurements up to $y(k-1)$

Prediction estimators:

- Construct a model of process dynamics and use it to predict

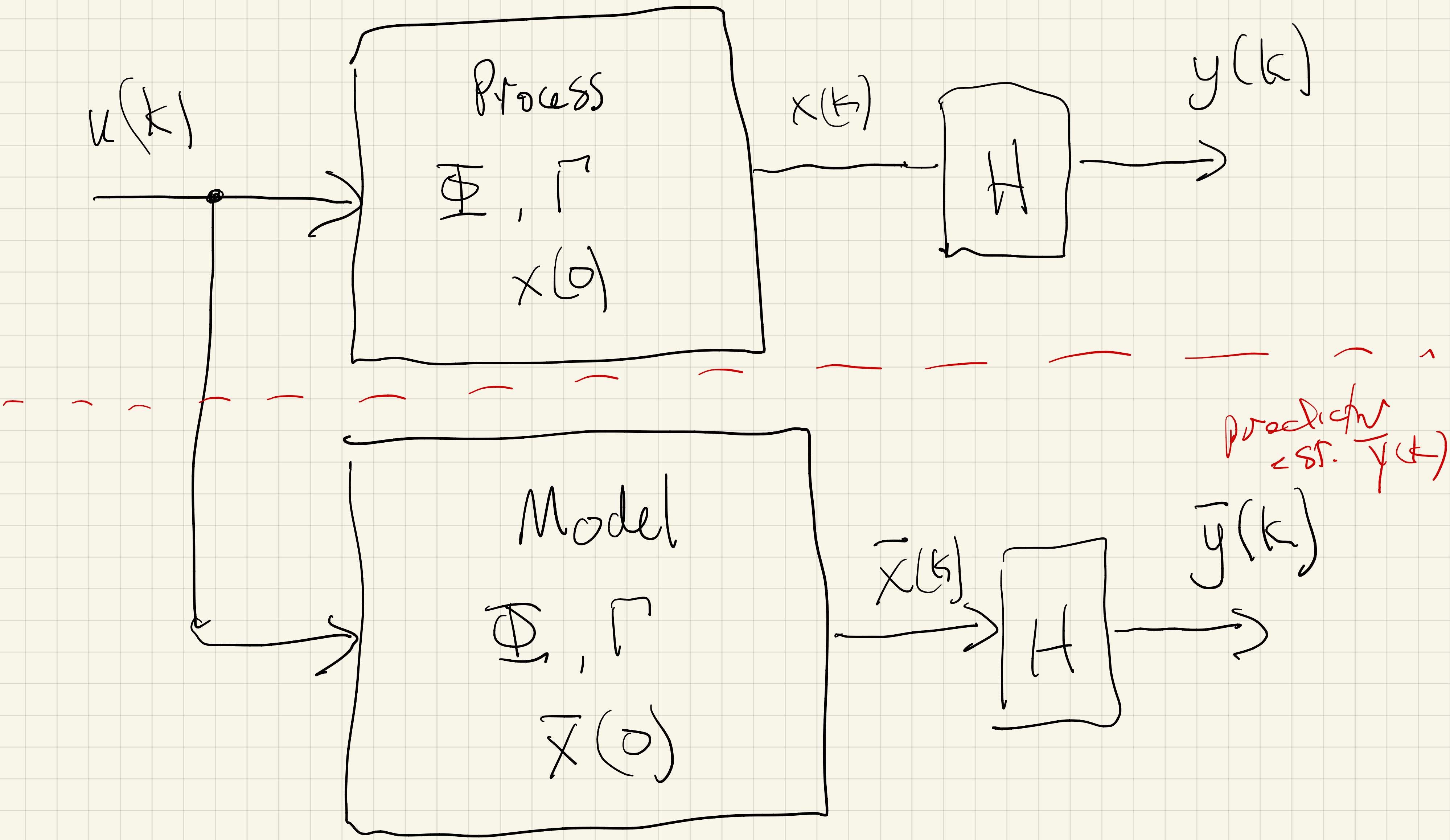
Φ, Γ

$$\underset{n\times 1}{\bar{x}(k+1)} = \underset{n\times n}{\Phi}\,\underset{n\times 1}{\bar{x}(k)} + \underset{n\times 1}{\Gamma}\,\underset{1\times 1}{u(k)} \quad \leftarrow \text{assumed}$$

known

But $\rightarrow$ would need to know $x(0)$ and then let $\bar{x}(0) = x(0)$

"Open-loop" estimator



Let $\tilde{x} = \bar{x} - x$, at all $k \geq 0$

"error of the prediction estimate"

$$\tilde{x}(k+1) = \left(\Phi \bar{x}(k) + \boxed{\Gamma u(k)}\right) - \left(\Phi x(k) + \boxed{\Gamma u(k)}\right)$$

$$\boxed{\tilde{x}(k+1) = \Phi \tilde{x}(k)}$$

Dynamics - governed by an "uncompensated process"

Assume $\bar{x}(0) \neq x(0)$

$\Rightarrow \tilde{x}(0) \neq 0$

What happens?

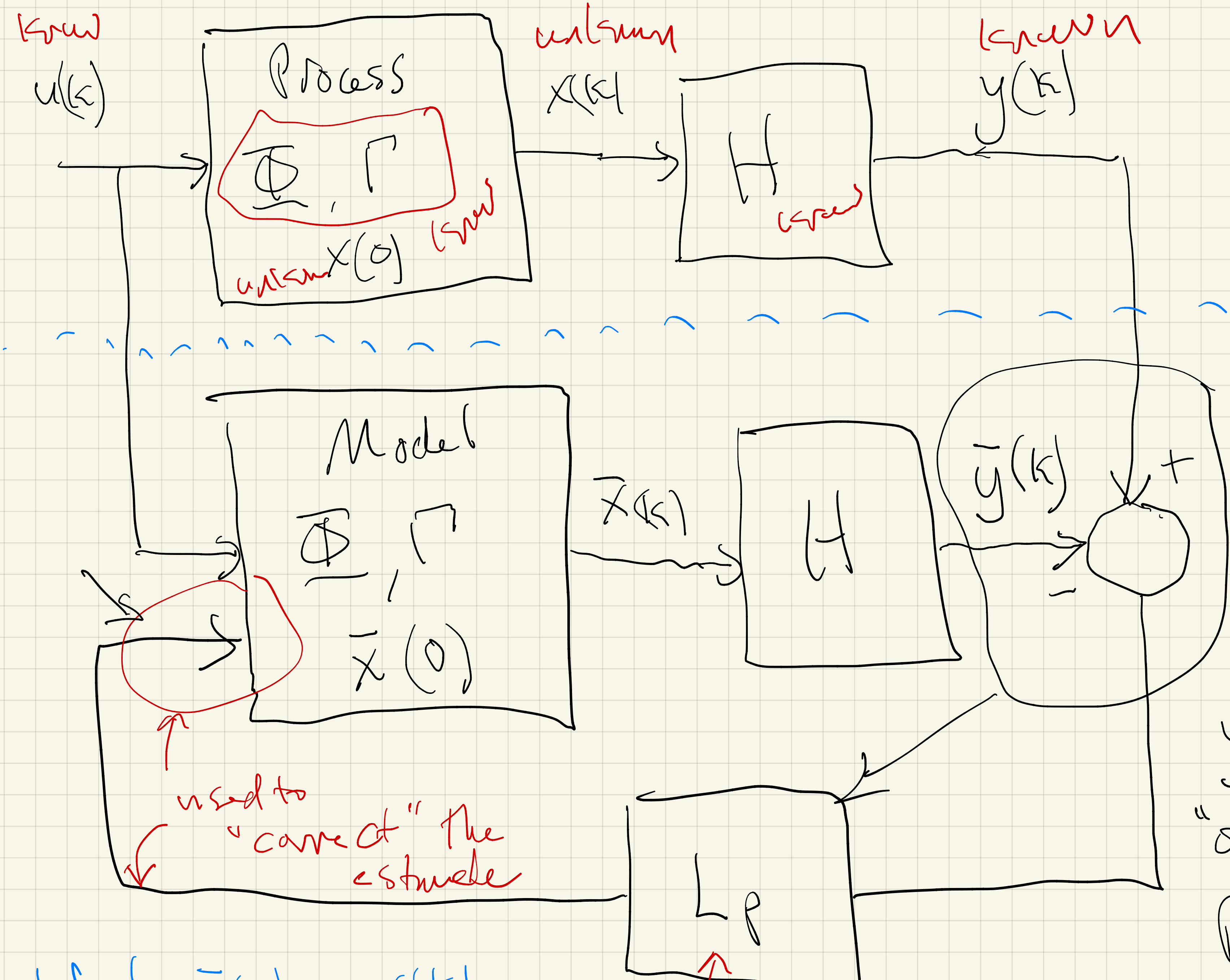
- Process is unstable $\Rightarrow$ $\tilde{x}(k)$ will grow larger than $\tilde{x}(0)$
- Process is AS (eig(Φ) in unit disk)

  $x(k)$ and $\bar{x}(k)$ both $\rightarrow 0$ as $k \rightarrow \infty$ (not really because $\bar{x}$ is a good estimate)

  - keep mind, we do want

    $\tilde{x} = \bar{x} - x \rightarrow 0$, $k \rightarrow \infty$

    as $\bar{x} \rightarrow x$ (i.e. good estimate)



"prediction estimator"
(known, what we are designing)

$y(k) - \bar{y}(k)$
"output prediction error"
(for $\bar{x}$ and x)

Want $\bar{x}(k) \to x(k)$
"good estimate"

This is feedback control

$$\underset{n\times 1}{\bar{x}(k+1)} = \underset{n\times n}{\Phi}\,\underset{n\times 1}{\bar{x}(k)} + \underset{n\times 1}{\Gamma}\,\underset{1\times 1}{u(k)} + \underset{n\times 1}{\boxed{L_p}}\Big(\underbrace{\underset{1\times 1}{y(k)} - \underset{1\times n}{H}\,\underset{n\times 1}{\bar{x}(k)}}_{1\times 1}\Big)$$

$$\frac{m=1}{p=1}$$

One variable here that is not known at this point in the development

$\bar{y}$

"Corrector"

- Good job estimating means

$\bar{y} \approx y \Rightarrow$ last term ≈ 0

$\Rightarrow$ not a correction

- Bad job estimating means

$\bar{y} \neq y \Rightarrow$ last term is big $\Rightarrow$ ~~correct~~ fix it!

Estimation error: $\tilde{x} = \bar{x} - x, \quad k \geq 0$

$$\tilde{x}(k+1) = \left(\Phi \bar{x} + \cancel{\Gamma u} + L_p\left(y - H\bar{x}\right)\right) - \left(\Phi x + \cancel{\Gamma u}\right)$$

$y = Hx$

$L_p(y - H\bar{x}) = -L_p H \tilde{x}$

$$= \Phi \tilde{x} - L_p H \tilde{x}$$

$$\boxed{\tilde{x}(k+1) = (\Phi - L_p H)\,\tilde{x}(k)}$$

Notice similarity to "control law" equations

If AS, i.e., all eigenvalues,

$$\mathrm{eig}(\overbrace{\Phi - L_p H}) \text{ in strict LHP}$$

Then

$$\lim_{k\to\infty} \tilde{x}(k) = 0 \qquad \text{for any } \tilde{x}(0)$$

$$\Rightarrow \bar{x}(k) \to x(k), \quad k \to \infty \quad \text{for any } \bar{x}(0)$$

(it does not have to be that $\bar{x}(0) = x(0)$)

The estimator works!

- The estimator can "see inside" the dynamical system

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = H x(k)$$

- it can asymptotically estimate (perfectly) the variable $x(k)$

$t \to \infty$

We will design for speed of the estimator

using

- $u(k)$
- $y(k)$
- Φ, Γ, H

$k \geq 0$

- Can result in savings - may be able to use fewer sensors

How do we find the estimator gain L_p?

- Same approach as for the control law $u = -Kx$

① Specify desired estimator root locations in the z-plane – desired estimator CE

$$\beta_e(z) = (z - \beta_1) \cdots (z - \beta_n) = 0$$

- β_i are real or complex
- quantify how fast the estimator should converge

② Form the estimator-error CE

$$\gamma_e(z) = \det|zI - \Phi + L_p H| = 0$$

③ Let

$$\beta_e(z) = \gamma_e(z)$$

$\rightarrow$ equate coefficients ; solve for the n values in k_p

<u>Next time:</u>

- Observability
- Ackermann's formula
- Matlab for finding k_p (place(·))

Observability:

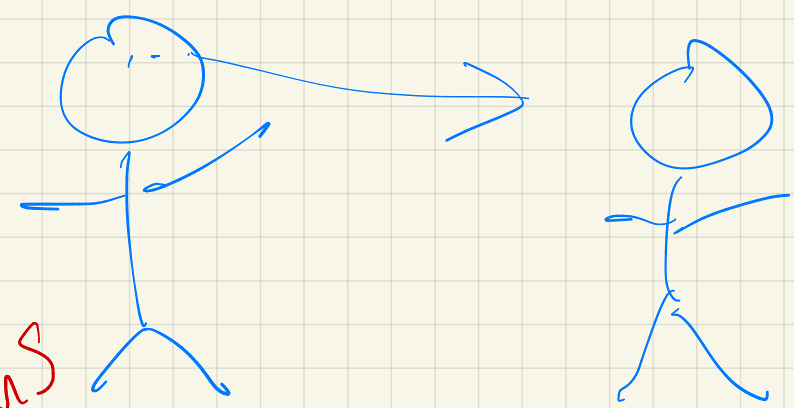
- Given desired estimator roots, is L_p unique? Exist?
- If y is a scalar, and the system is "observable", yes

"unobservable features"

outward → smile

inside → sad

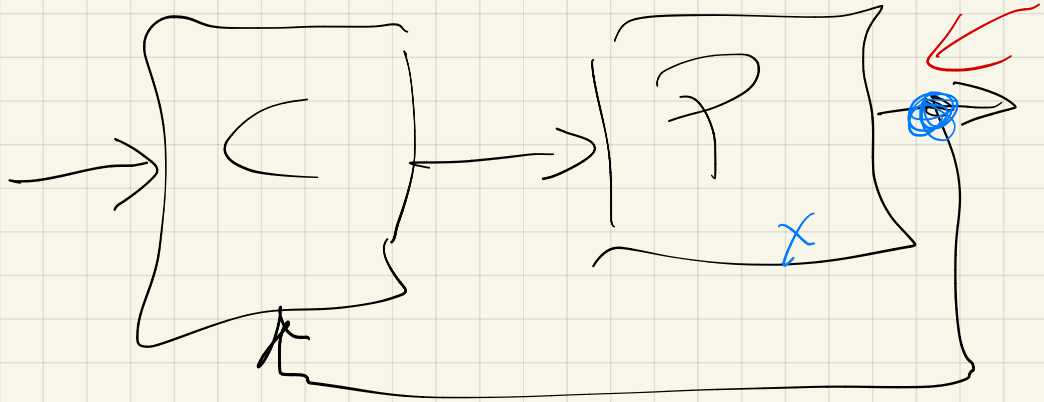
Unknowns



"observable features"

what we can see

Know



What can we know about the process

Estimator can provide

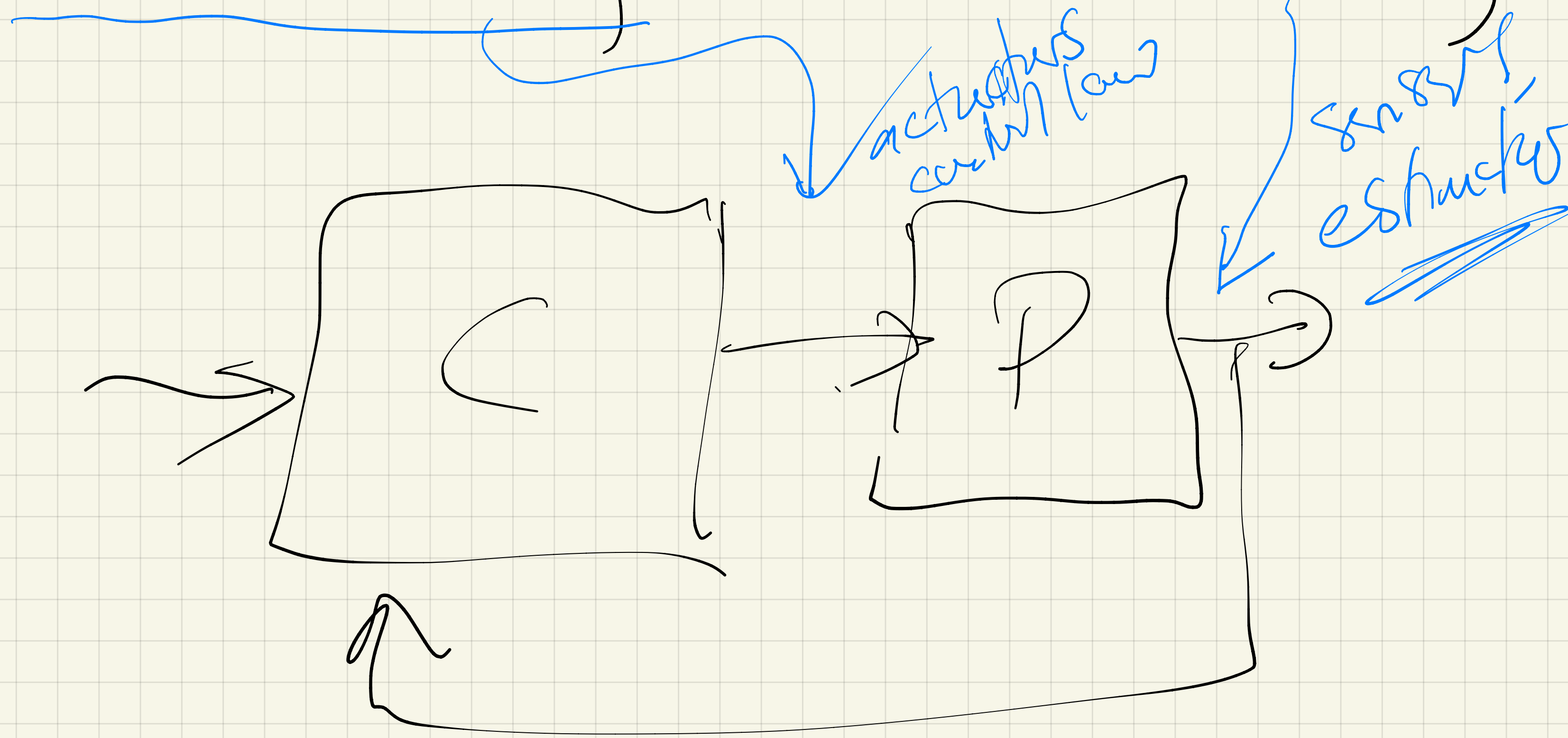
$\hat{x}_i$

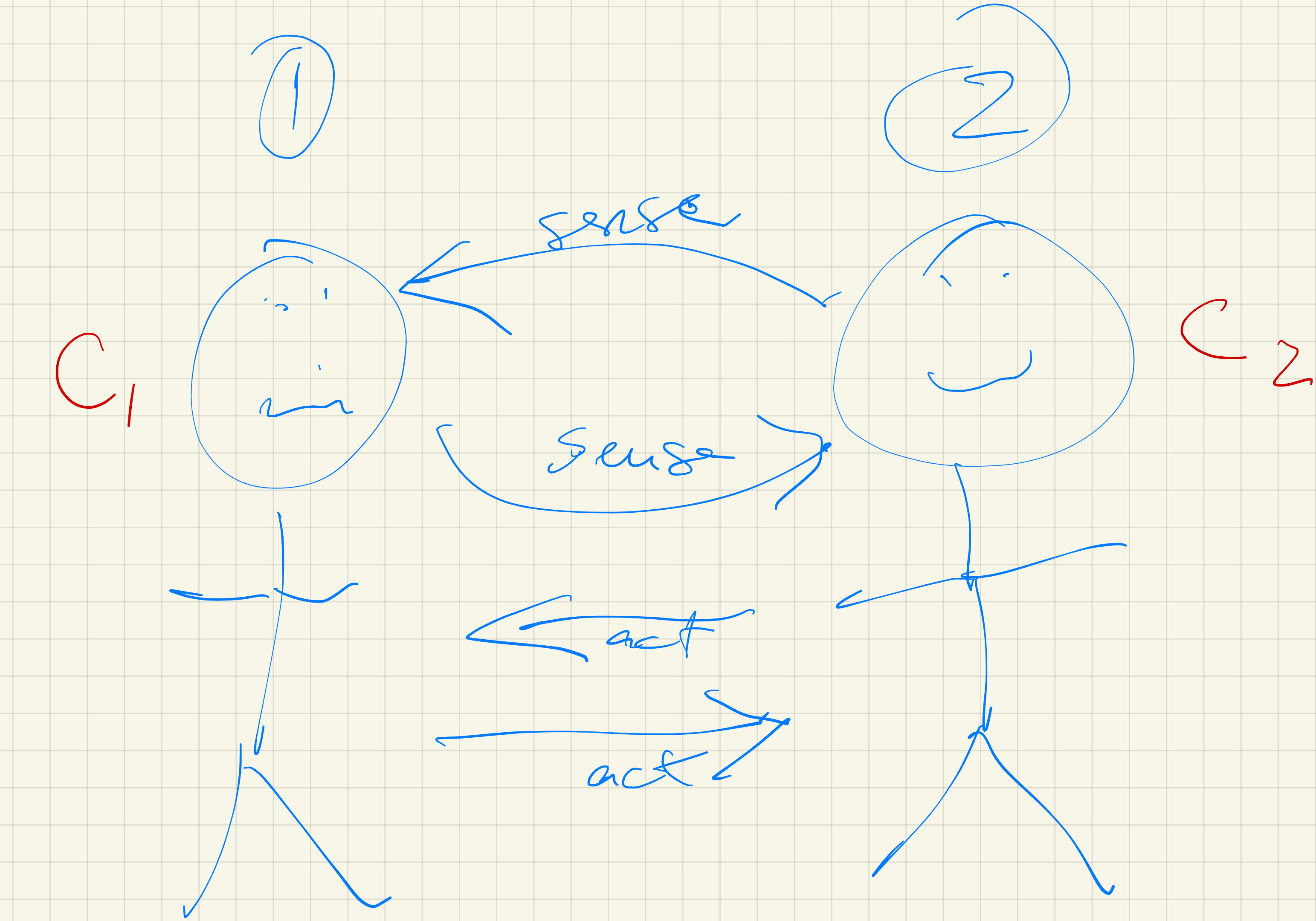
Cases when sensors give observability

other cases cannot observe some x_i

Note: Symmetry:

Controllability   Observability

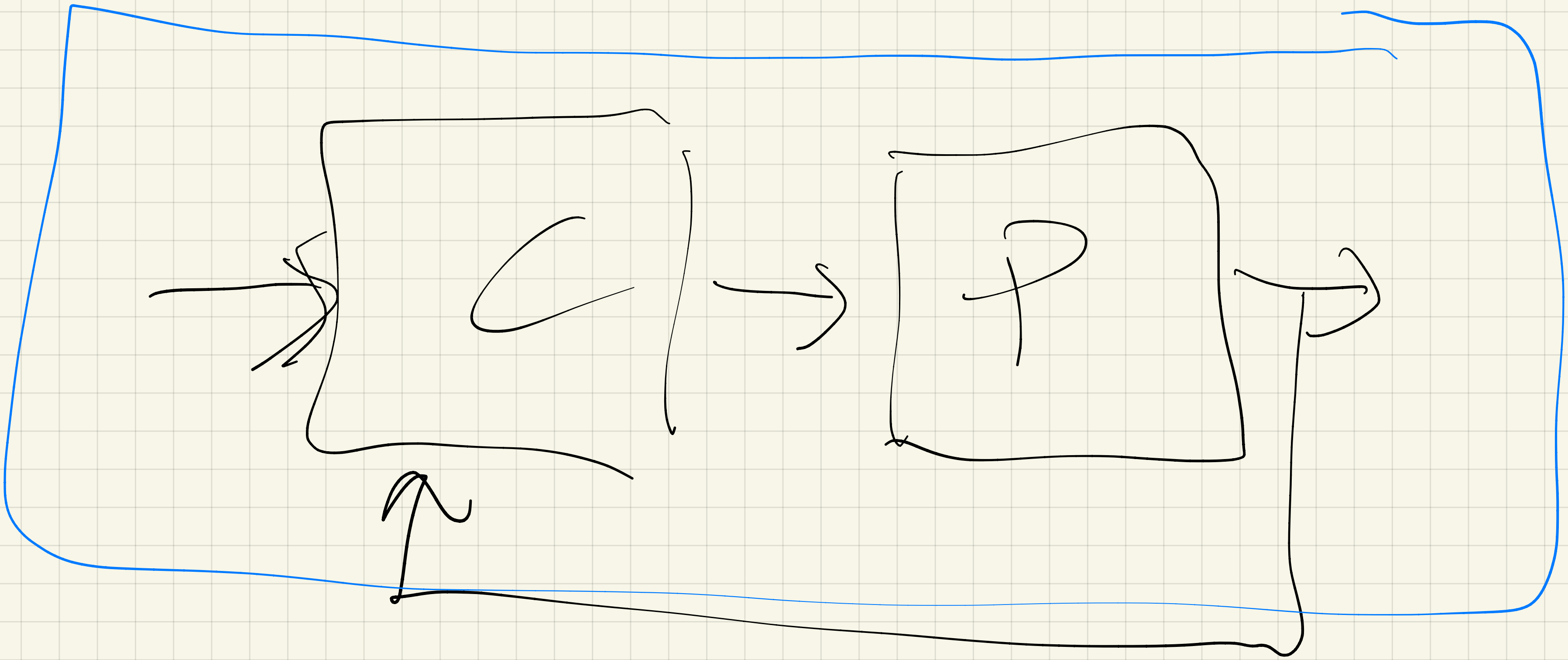




Dialog

① may want to "control" ② (eg. convince)

② " " " " ① (" ")



- Need P to be controllable.
- Need P to be observable

If have both $\Rightarrow$ Can design C to get
- stability
- performance

Test for observability

$$\text{rank}\left(\begin{bmatrix} H \\ H\Phi \\ H\Phi^2 \\ \vdots \\ H\Phi^{n-1} \end{bmatrix}\right) = n$$

$1\times n$, $n\times n$, $1\times n$, $n-1$, $n \times n$

rank = Matlab command

It may be "close to" unobservable

$\Rightarrow$ Matlab can have comp. prob.

If so, $\Rightarrow$ system

$$\left\{\begin{aligned} x(k+1) &= \Phi x + \Gamma u \\ y &= Hx \end{aligned}\right.$$

is **observable**

Warning

1. suspect (*) is observable
2. Not sure if (*) is observable

Ackerman's Formula (for estimators)

$$L_p = \alpha_e(\underline{\Phi}) \begin{bmatrix} H \\ H\underline{\Phi}^2 \\ \vdots \\ H\underline{\Phi}^{n-1} \end{bmatrix}^{\boxed{-1}} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$n \times n$

reason we need $\circledast$ to be observable

$$\alpha_e(\underline{\Phi}) = \underline{\Phi}^n + \alpha_1 \underline{\Phi}^{n-1} + \alpha_2 \underline{\Phi}^{n-2} + \cdots + \alpha_n \underline{I}$$

Where $\alpha_e(z)$ is above

$\ast$ Note: for $n \geq 10$ (or so) Matlab can fail

BUT you can use "place(·)"

Given $\Phi - L_p H$, take transpose

$$\Phi^T - H^T L_p^T$$

same form as $\Phi - \Gamma K$ (know how to use "place()" to compute K, given Γ, Φ, for this)

$\Rightarrow$ in "place(·)" (give it desired roots) substitute

$\Phi_{n \times n} \longrightarrow \Phi^T_{n \times n}$

$\Gamma_{n \times 1} \longrightarrow H^T_{n \times 1}$

$K_{1 \times n} \longrightarrow L_p^T{}_{1 \times n}$

Here, "place(·)" gives output L_p^T

output of arguments in place(·)

for control law $u = -Kx$

Current estimators: (normally used rather than pred. est.)

- Predictor estimator

$\bar{x}(k)$ uses measurements up to $y(k-1)$

$\Rightarrow$ if we used $u = -K\bar{x}$ we would not be using the current information

Let's fix that ...

- Current estimate $\hat{x}(k)$ based on the latest information (i.e, $y(k)$)

- Choose

$$\hat{x}(k) = \underbrace{\bar{x}(k)}_{\text{predicted value of } x(k)} + \underbrace{L_c\big(\overset{k^{th}\text{ data}}{y(k)} - \underbrace{\overbrace{H\bar{x}(k)}^{\text{predicted output}}}_{\text{error in the prediction of } y(k)}\big)}_{\text{"corrector term"}}$$

Where

$$\boxed{\bar{x}(k) = \Phi\hat{x}(k-1) + \Gamma u(k-1)} \quad (\forall k \geq 0)$$

To compare pred/curr. estimators,

$$\bar{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k)$$

$$\bar{x}(k+1) = \Phi(\bar{x}(k) + L_c(y(k) - H\bar{x}(k))) + \Gamma u(k)$$

$$\bar{x}(k+1) = \underbrace{\Phi\bar{x}(k) + \Gamma u(k)}_{\text{pred. based on model}} + \underbrace{\Phi L_c(y(k) - H\bar{x}(k))}_{\text{correction based on a measurement}}$$

"predictor/corrector" approach

Also, $\boxed{\tilde{x} = \bar{x} - x}$, $k \geq 0$

$$\tilde{x}(k+1) = \bar{x}(k+1) - x(k+1)$$

$$\tilde{x}(k+1) = \left(\Phi \bar{x}(k) + \cancel{\Gamma u(k)} + \Phi L_c\left(y(k) - H\bar{x}(k)\right)\right)$$

$$= H x(k)$$

$$= -H\tilde{x}(k)$$

$$- \left(\Phi x(k) + \cancel{\Gamma u(k)}\right)$$

$$\Phi \tilde{x}(k) - \Phi L_c H \tilde{x}(k)$$

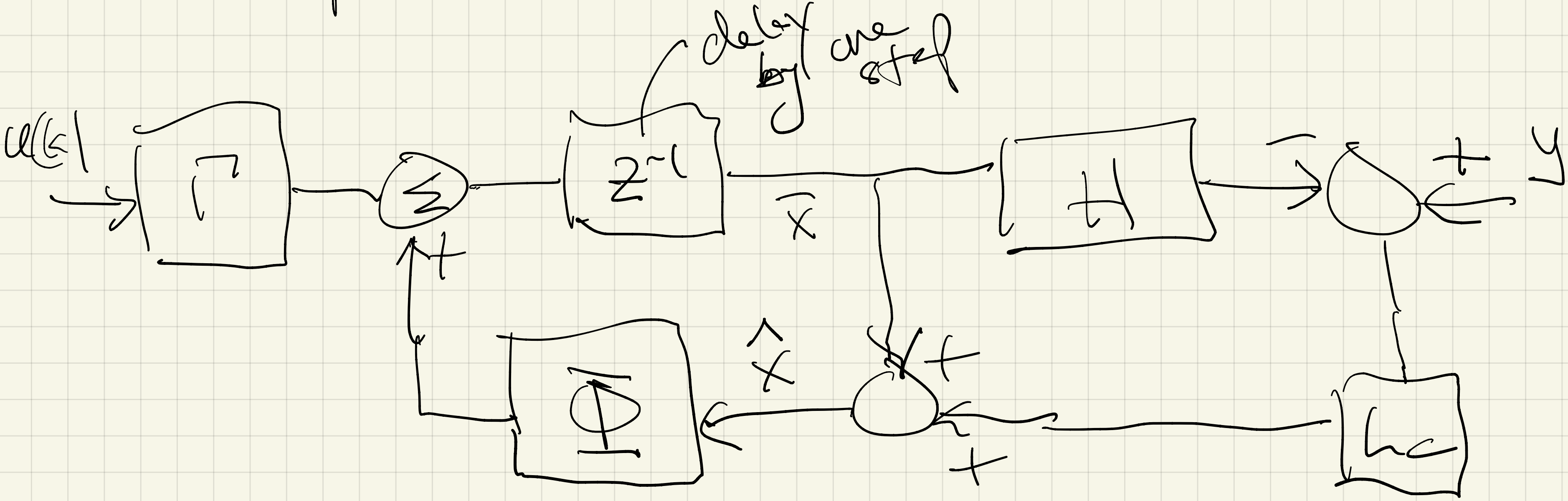
$$\Rightarrow \tilde{x}(k+1) = (\Phi - \underbrace{\Phi L_c}_{L_p} H)\tilde{x}(k)$$

Let $\boxed{L_p = \Phi L_c}$:

① ★ is the same as the prediction in the prediction estimator

② ★★ The estimator error equations is the same as in the predictor est. case

Relationships between variables:



Suppose $\tilde{x} = \hat{x} - x$, $k \geq 0$ (earlier we had $\tilde{x} = \bar{x} - x$)

Next, show that

$$\tilde{x}(k+1) = (\Phi - L_c H \Phi)\,\tilde{x}(k)$$

$$\hat{x}(k) = \left(\bar{x}(k) + L_c\left(y(k) - H\bar{x}(k)\right)\right)$$

$$\bar{x}(k) = \Phi\hat{x}(k-1) + \Gamma u(k-1)$$

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L_c\left(H x(k+1) - H\bar{x}(k+1)\right)$$

$$- \left(\Phi x(k) + \Gamma u(k)\right)$$

$$\tilde{x}(k+1) = \Phi\tilde{x}(k) + L_c H\left(\underbrace{x(k+1)}_{\Phi x(k) + \Gamma u(k)} - \underbrace{\bar{x}(k+1)}_{\Phi\hat{x}(k) + \Gamma u(k)}\right)$$

$$\Phi\left(x(k) - \hat{x}(k)\right)$$

$$- \Phi\tilde{x}(k)$$

$$\Rightarrow \boxed{\tilde{x}(k+1) = \left(\Phi - L_c H\Phi\right)\tilde{x}(k)}$$

Want $\tilde{x}(k) \to 0,\ k \to \infty$

From $(\star\star)$ and $(\star\star\star)$

$$\mathrm{eig}(\Phi - \Phi L_c H) = \mathrm{eig}(\Phi - L_c H \Phi)$$

Since dynamics of different outputs of the same system

$\Rightarrow$ Can use either for finding L_c

Note: $(\star\star\star)$ is similar to $\tilde{x}(k+1) = (\Phi - L_p H)\tilde{x}(k)$ (from pred. estimator)

but $H\Phi$ appears rather than H

Ackerman's formula:

$$L_c = \alpha_e(\Phi) \begin{bmatrix} H\Phi \\ H\Phi^2 \\ \vdots \\ H\Phi^n \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\alpha_e(\Phi) = \Phi^n + \alpha_1 \Phi^{n-1} + \cdots + \alpha_n I$$

$$\alpha_e = z^n + \alpha_1 z + \cdots + \alpha_n \Bigg)$$ Desired root locations

Can compute L_c this way

take transpose of $\Phi - L_c H \Phi$ to get

$\Phi^T - \Phi^T H^T L_c^T$

Same form as $\Phi - \Gamma K$

$K = \mathrm{place}(\Phi, \Gamma, P)$

Substitute Φ^T for Φ

$\Phi^T H^T$ for Γ

$L_c^T = \mathrm{place}(\Phi^T, \Phi^T H^T, P)$

Then computation via $\mathrm{place}(\cdot)$ gives L_c^T instead of K

To get L_c take the transpose of the result from Matlab

Reduced order estimators

- Above, estimating the <u>entire</u> state vector x

given $\overset{1\times 1}{y} = \overset{1\times n}{H}\,\overset{n\times 1}{x}$

known

want to estimate x

$\Rightarrow$ Not possible to invert H to get x

$\boxed{\text{— there we use } y,\ u,\ \Phi,\ \Gamma,\ H \longrightarrow \text{est of } x}$

Why estimate state elements that are measured directly?

widow

$$X_{n\times 1} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{n-1} \\ X_n \end{bmatrix}$$

Why estimate these when we know them?

often, in applications we can measure some of the state components e.g. measure all these

Earlier we aimed at producing $\bar{X}_{n\times 1}$ or $\hat{X}_{n\times 1}$ as estimates of X

* But if sensors are noisy (e.g. on X_2) then $\bar{X}$ or $\hat{X}$ can aid smoothing

Partition the state vector

$$\underset{n\times 1}{x} = \begin{bmatrix} x_a \\ x_b \end{bmatrix}$$

sets their order

new ordering

directly measure ~~This~~

portion we will estimate

Process model

$$\begin{bmatrix} x_a(k+1) \\ x_b(k+1) \end{bmatrix} = \underset{n\times n}{\begin{bmatrix} \underline{\Phi}_{aa} & \underline{\Phi}_{ab} \\ \underline{\Phi}_{ba} & \underline{\Phi}_{bb} \end{bmatrix}} \underset{n\times 1}{\begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix}} + \underset{n\times 1}{\begin{bmatrix} \Gamma_a \\ \Gamma_b \end{bmatrix}} \underset{1\times 1}{u(k)}$$

comes from Φ, Γ due to reorder

$$\underset{1\times 1}{\underline{y}(k)} = \underset{1\times n}{\begin{bmatrix} I & \underline{0} \end{bmatrix}} \underset{n\times 1}{\begin{bmatrix} \underline{x_a(k)} \\ x_b(k) \end{bmatrix}}$$

closer to represent what we are studying

$$x_b(k+1) = \Phi_{bb}\, x_b(k) + \underbrace{\Phi_{ba}\, x_a(k) + \Gamma_b\, u(k)}$$

known, view it as the "input"

$$\underbrace{x_a(k+1) - \Phi_{aa}\, x_a(k) - \Gamma_a\, u(k)} = \Phi_{ab}\, x_b(k)$$

known   unknown

known, view as a "measurement"

To get an estimator, make substitutions

$$x \leftarrow x_b$$

$$\Gamma u \leftarrow \boxed{\Phi_{ba}\, x_a(k) + \Gamma_b\, u(k)}$$

$$y(k) \leftarrow \boxed{x_a(k+1) - \Phi_{aa}\, x_a(k) - \Gamma_a\, u(k)}$$

$$H \leftarrow \Phi_{ab}$$

Reduced-Order Est.

$n_b \times n_y \qquad n_y \times 1$

$$\hat{x}_b(k+1) = \bar{\Phi}_{by}\,\hat{x}_y(k) + \underbrace{\bar{\Phi}_{ba}\,x_a(k) + \bar{\Gamma}_b\,u(k)}_{\text{predictor}}$$

$n_b \times 1$

$$+ \boxed{L_r}\left(\underbrace{x_a(k+1) - \bar{\Phi}_{aa}\,x_a(k) - \bar{\Gamma}_a\,u(k) - \bar{\Phi}_{ay}\,\hat{x}_b(k)}_{\text{"corrector" "( )" prediction error}}\right)$$

$n_y \le n$

$\Rightarrow$ dimension of this estimator is lower than our original estimator.

$\Rightarrow$ computationally it is simpler.

Value? Maybe....

Cause fun algebra

* However, using it as a filter is quite useful - often

$$\tilde{x}_b = x_b - \hat{x}_b, \quad k \geq 0$$

Skip algebra...

$$\tilde{x}_b(k+1) = \underline{\left(\Phi_{bb} - L_r \Phi_{ab}\right)} \tilde{x}(k)$$

$\Rightarrow$ Pick L_r as before...

- Pick roots of

$$\det\left| zI - \Phi_{bb} + L_r \Phi_{ab} \right| =$$

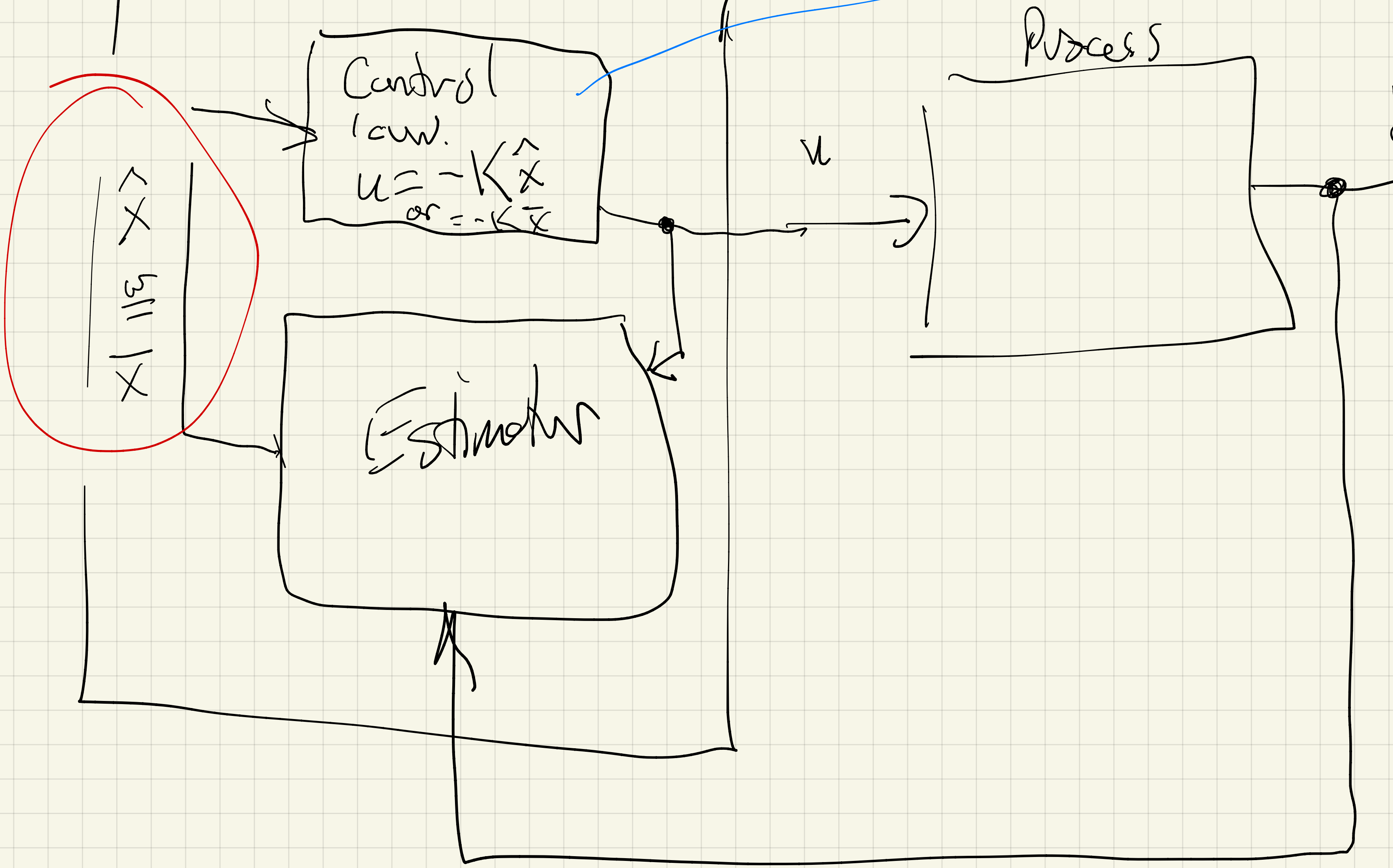
to be in desired locations $\alpha_e(z)$

- Use Ackerman's formula $\underline{\underline{\text{or}}}$ "place()" in Matlab.

Next, design a "Regulator"

based on $u = -Kx$

$\underline{\underline{FSF}}$



* Above: Replace x by $\hat{x}$ or $\bar{x}$ depending on what estimator we use