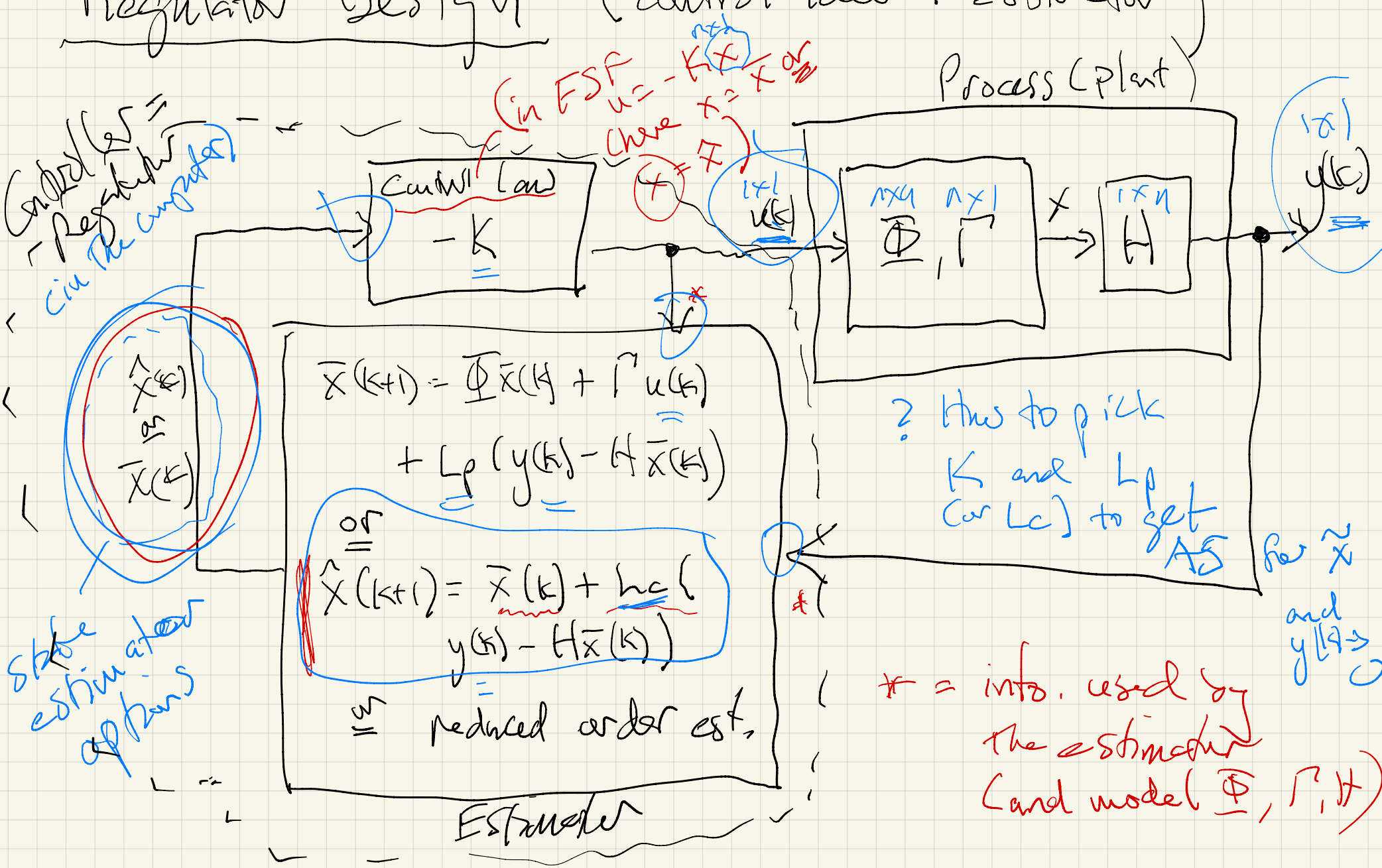
ECE 5551 Passino Part 2

Regulator Design (control law + estimator)



Separation Principle:

- Prediction estimator case (same results below hold for current estimator or ROE)

$u = -K\tilde{x}$

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

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

From earlier,

$$\begin{bmatrix} \tilde{x}(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} \Phi - L_p C & 0 \\ -\Gamma K & \Phi - \Gamma K \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ x(k) \end{bmatrix}$$

Characteristic Eq.

$$\det \begin{vmatrix} zI - \Phi + LpH & 0 \\ \Gamma K & zI - \Phi + \Gamma K \end{vmatrix} = 0$$

$$= \left| zI - \Phi + LpH \right| \left| zI - \Phi + \Gamma K \right| =$$

est. ctrl

$$\alpha_e(z)\,\alpha_c(z) = 0$$

The same ones from earlier

Gives "separation"

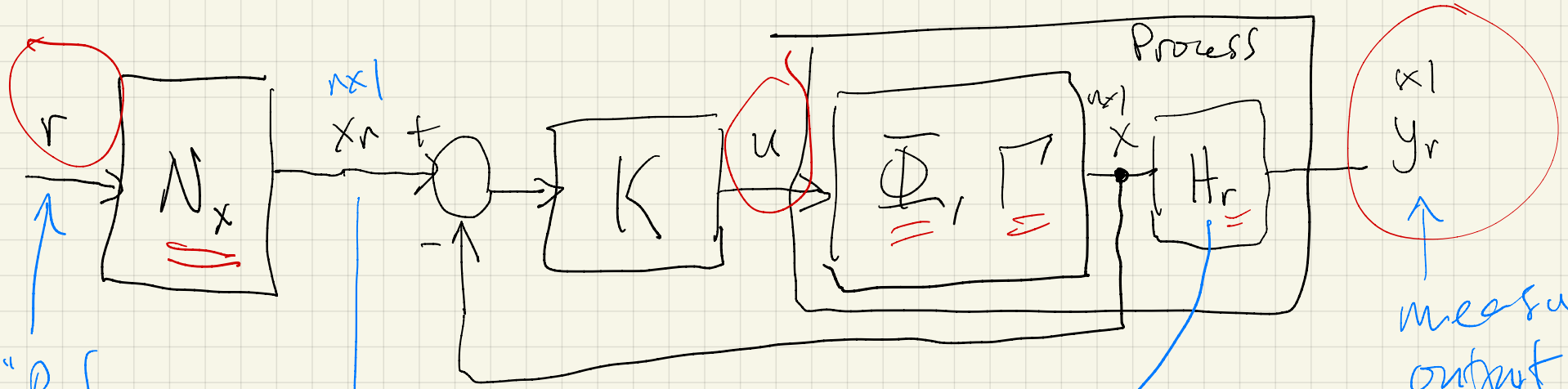
$\Rightarrow$ can independently design the estimator and control laws

- Specifying:
- Design strategy:
  - Control law design (to meet desired root locations for CLCP)
  - Estimator design (to ")

In practice: Usually choose desired estimator roots to be faster than those of the CLCP for the control law.

Adding The Reference Input

Full-state feedback case ($u = -Kx$)



"Reference input" (desired value of y_r)

Desired value of the state x

Chosen to select the output (y_r) that you want to control

measured output (y_r can be different from $y = Hx$

Goal: Find N_x so y_r is at a desired value

Suppose: $u \in \mathbb{R}^m$ (m×1), $y_r \in \mathbb{R}^m$ (m×1)

Have

$$u = -K(x - x_r)$$

Want: N_x such that $N_x r = x_r$ ← an equilibrium for this chosen r

Example: In the earlier $F = ma$ example

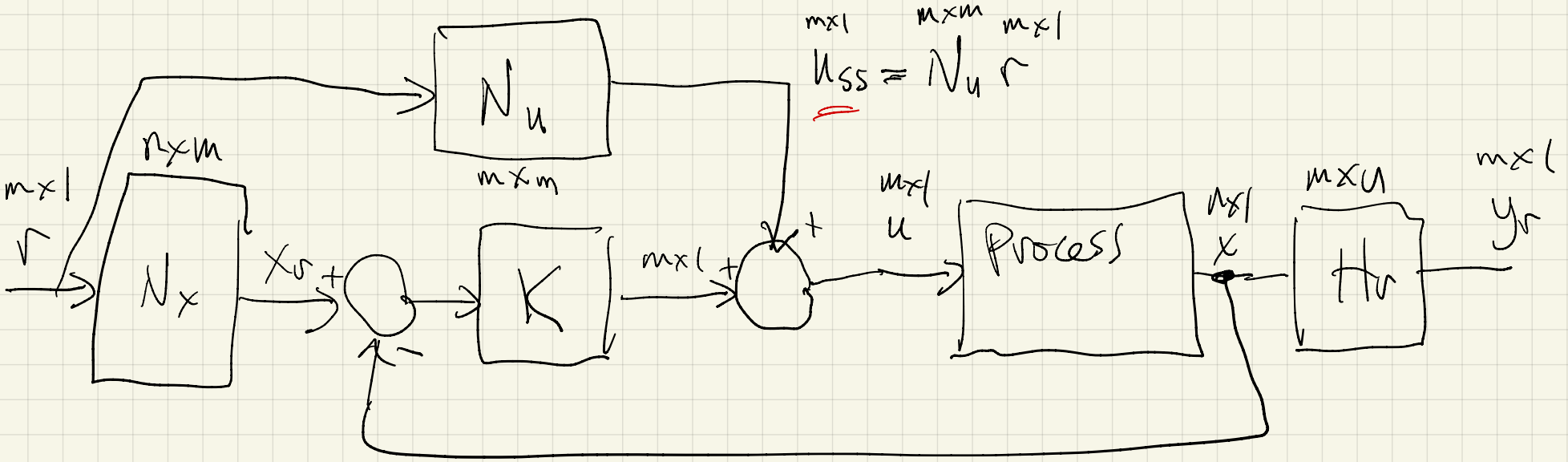
$$x = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

and

$$\underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{N_r} r = x_r = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$$

$r = 10$

* But whether this is achieved, depends on the system dynamics



steady state requirements

$$N_x r = x_r = x_{ss}$$

$$H_r x_{ss} = y_r = r$$

$$H_r N_x r = r \Rightarrow H_r N_x = I$$

(★★)

If system is at steady state

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

$$x_{ss} = \Phi x_{ss} + \Gamma u_{ss}$$

or

$$(\Phi - I) x_{ss} + \Gamma u_{ss} = 0$$

$$(\Phi - I) N_x r + \Gamma N_u r = 0$$

$$(\Phi - I) N_x + \Gamma N_u = 0 \quad (*)$$

Collecting (*) and (**),

$$\begin{bmatrix} \Phi - I \\ H \end{bmatrix} \begin{bmatrix} \Gamma \\ 0 \end{bmatrix} \begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

(Gives gains we need to help define the controller)

$$\underset{n\times m}{\begin{bmatrix} N_x \\ N_u \end{bmatrix}} = \underset{n\times n}{\begin{bmatrix} \Phi - I \\ H \end{bmatrix}} \underset{n\times m}{\begin{bmatrix} \Gamma \\ 0 \end{bmatrix}}^{-1} \underset{n\times m}{\begin{bmatrix} 0 \\ I \end{bmatrix}}$$

If the inverse exists

$\Leftrightarrow$

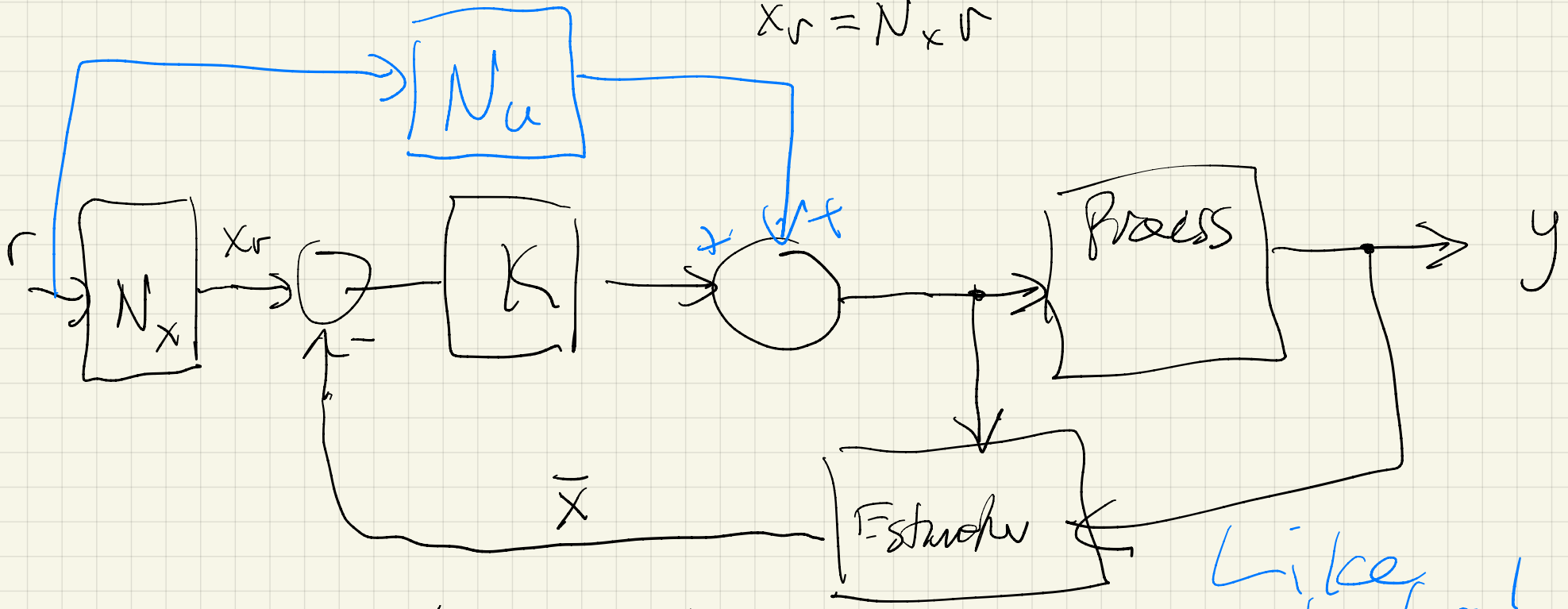
(if it doesn't exist, then it can't track the desired ref. input)

Reference input when the estimator is used.

Choose $u = -K(\bar{x} - x_r) + N_u r$

$=$

$x_r = N_x r$



Like before!

* With the current estimator

$$u(t) = -K(\hat{x}(t) - x_r) + N_u r$$

* Can do similar for the reduced-order estimator

Side note: For simulating the regulator

With this, can do performance analysis of reg.

Prediction estimator: $(n, m=1)$

$1\times 1 \quad 1\times n \quad n\times 1$

$$u = -Kx$$

$2n\times 1$

$$\begin{bmatrix} x(k+1) \\ n\times 1 \\ \bar{x}(k+1) \\ n\times 1 \end{bmatrix} = \begin{bmatrix} \Phi & -\Gamma K \\ n\times n & n\times 1 \; 1\times n \\ & n\times n \\ L_p H & \Phi - \Gamma K - L_p H \\ n\times 1 \; 1\times n & n\times 1 \; 1\times n \\ n\times n & n\times n \; n\times n \; n\times n \end{bmatrix} \begin{bmatrix} x(k) \\ n\times 1 \\ \bar{x}(k) \\ n\times 1 \end{bmatrix}$$

"State" of the regulator, $k+1$

Φ_{rp}, state transition matrix for the regulator (Cn)

State IC

$$\begin{bmatrix} x(0) \\ \bar{x}(0) \end{bmatrix}$$

Current estimator

$u = -Kx$

$$\begin{bmatrix} x(k+1) \\ \hat{x}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi - \Gamma K & \\ LcH\Phi & \Phi - \Gamma K - LcH\Phi \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix}$$

$2n \times 1$ $\qquad$ $2n \times 1$

Φ_{sc}

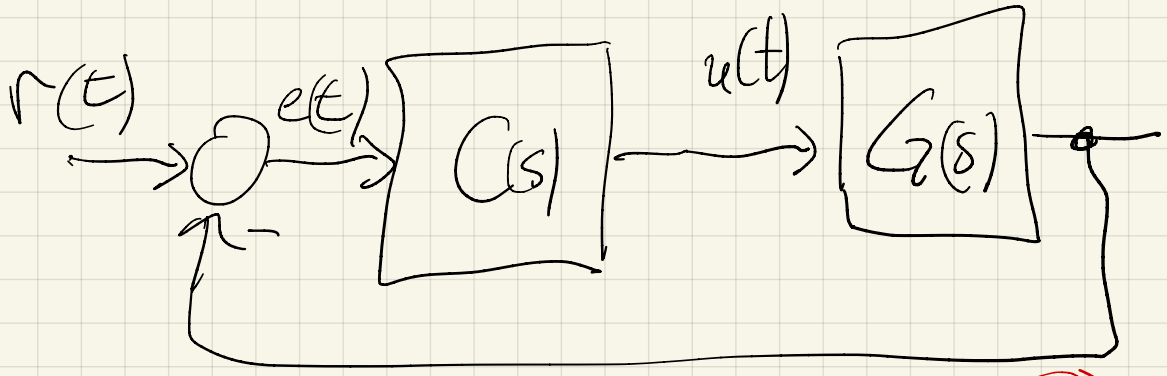
What you may have done — separate

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

$x(:, k+1)$

Integral Control

Why? For CT systems, consider



scalar SISO

* $G(s) = \frac{1}{s+1}$, $C(s) = \underbrace{K}_{1\times 1}$ $K \geq 0$

⇒ Can show $\lim_{t\to\infty} \underbrace{\left(r(t) - y(t)\right)}_{e(t)} \neq 0$ (actually, a constant)

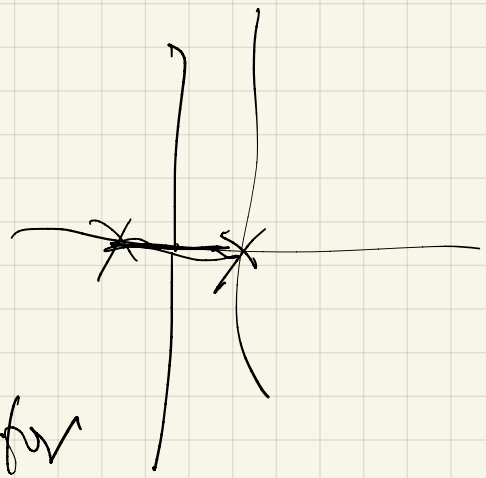
tracking error

- Use FVT
- System type

We can say that the FCS fails!

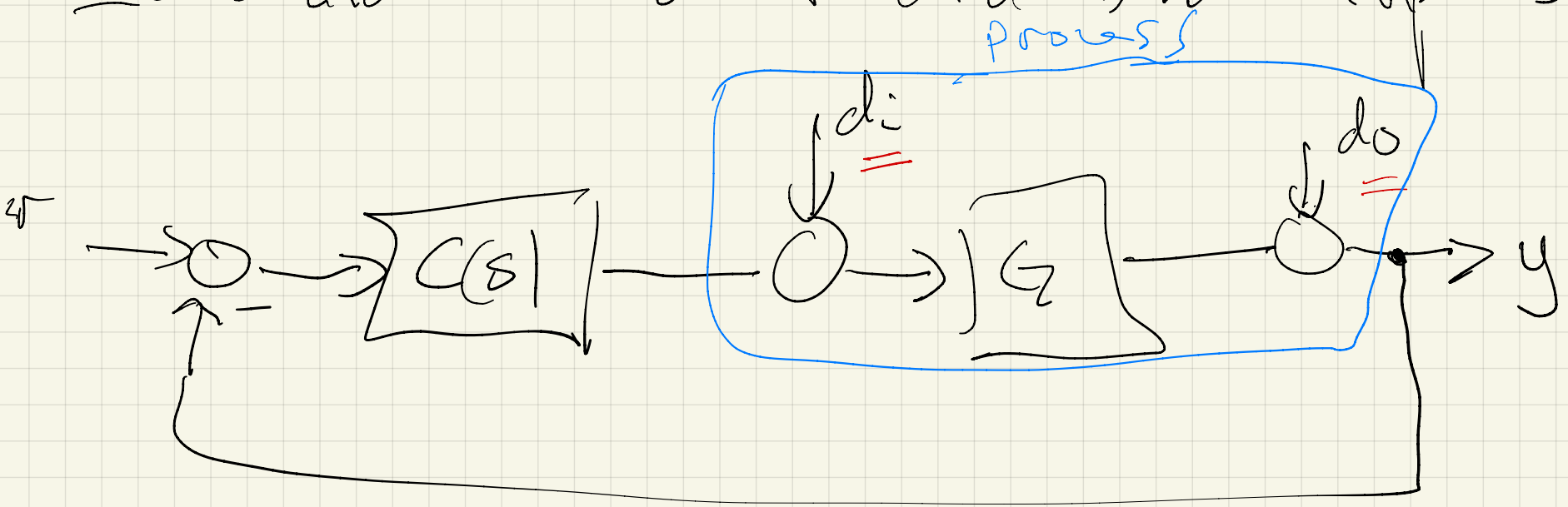
But, if you choose $C(s) = \frac{K}{s}$ ("integral control" - "integral action")

"Tracking objective" $\lim_{t \to \infty} (r(t) - y(t)) = 0$



This is why you "put an integrator in the loop"

Idea also holds for disturbance inputs



With step changes on either d_i or d_o will get perfect tracking also!

(Related to PI control - "proportional integral control"

- PID - prop. int. derivative control -)

- like adding I.S

Two approaches to adding integral action

(1) Integral control by state augmentation

- Given a model of the process
  - add an integrator to the output of the process
- Design for this augmented model to get gains for the process and integrator
- Translate non-process part into a controller implementation

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

$y(k) = H x(k)$

- Augment state with $x_I(k)$ where

For time index $k \geq 0$

$x_I(k+1) = x_I(k) + y(k) = x_I(k) + H x(k)$

DT

known - part of the controller

measurable

derivative ⇒ it is a sum

(output - integral)

- Augmented process model

$$\begin{bmatrix} x_I(k+1) \\ x(k+1) \end{bmatrix}_{(n+1)\times 1} = \underbrace{\begin{bmatrix} 1 & H \\ 0 & \Phi \end{bmatrix}}_{*} \begin{bmatrix} x_I(k) \\ x(k) \end{bmatrix}_{\text{State } (n+1)\times 1} + \underbrace{\begin{bmatrix} 0 \\ \Gamma \end{bmatrix}}_{*} u(k)$$

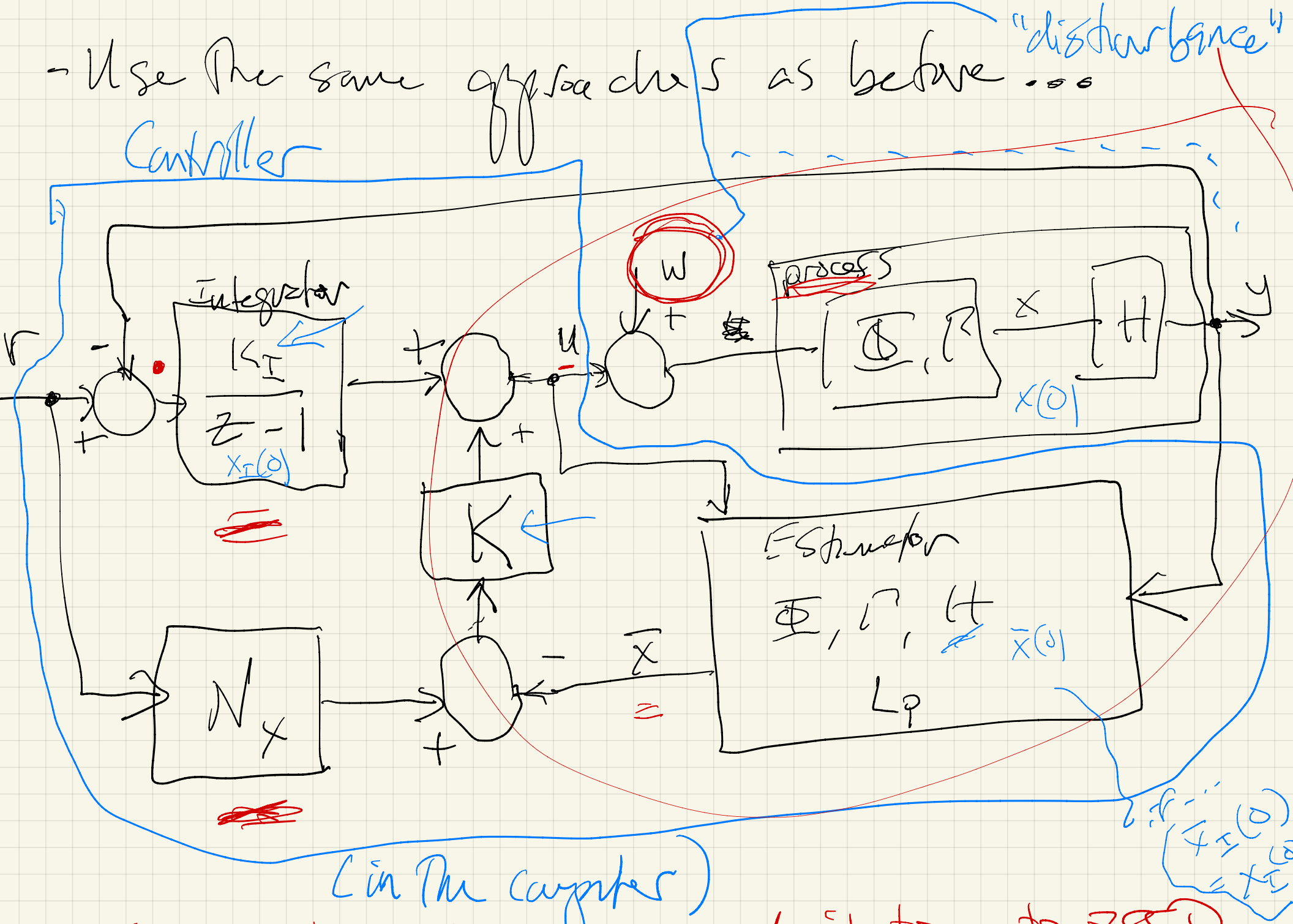
(dimensions: 1×1, $1\times n$, $n\times 1$, $n\times n$)

- Control Law

$$u(k) = -\underbrace{\begin{bmatrix} K_I & K \end{bmatrix}}_{1\times(n+1)} \begin{bmatrix} x_I(k) \\ x(k) \end{bmatrix}$$

Note: A FSF problem with $n+1$ states $\Rightarrow$ AF or "place" to get

- Use the same approaches as before ...



(in the computer)

- $e(k) = r(k) - y(k)$ → want it to go to zero

- Control law based on the augmented model (with the integrator (above)

- Estimator based on the unaugmented model (we know x_I so there is no need to estimate it)

Note: If you want another integrator, repeat this process

(recall we sometimes add multiple integrators to get a higher system type)

(2) Bias estimation approach

- Try to estimate the disturbance and cancel it

- Need model of the disturbance signal $w(k)$

To explain, easiest to do so in CT

Consider some model options

options $w(t)$, $t \geq 0$ — "input" disturbance (unknown)

① Constant: $\dot{w} = 0$

$\Rightarrow w(t) = w(0) = \text{const}$

$\forall t \geq 0$

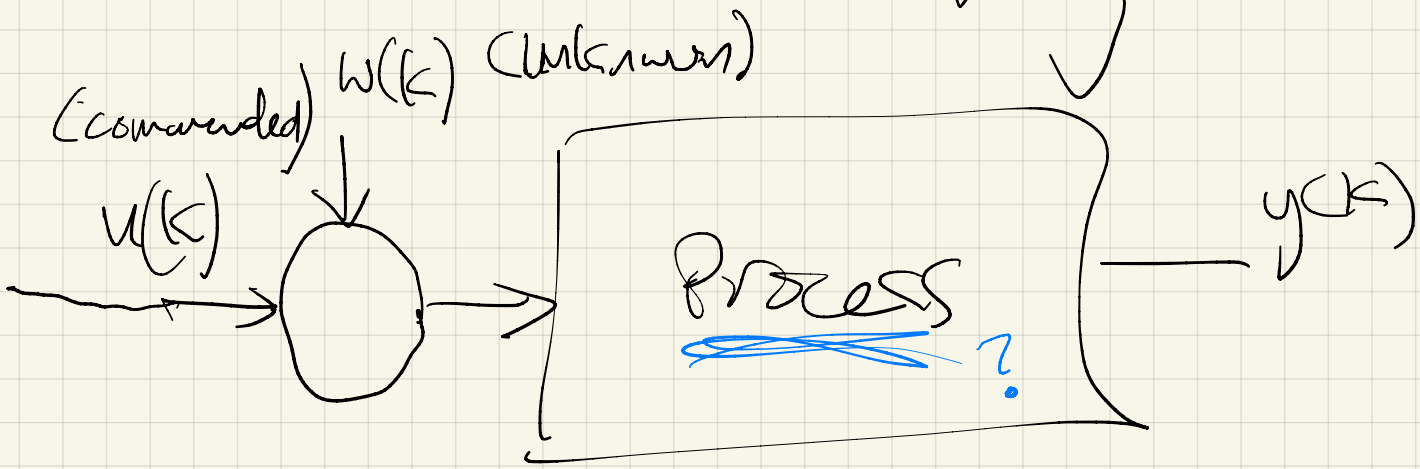
(if assume $w(0)$ is unknown $\Rightarrow w(t)$, $t \geq 0$ is unknown

② Oscillating: $\ddot{w}(t) = -\omega_0^2 w(t)$ (ω_0^2: constant)

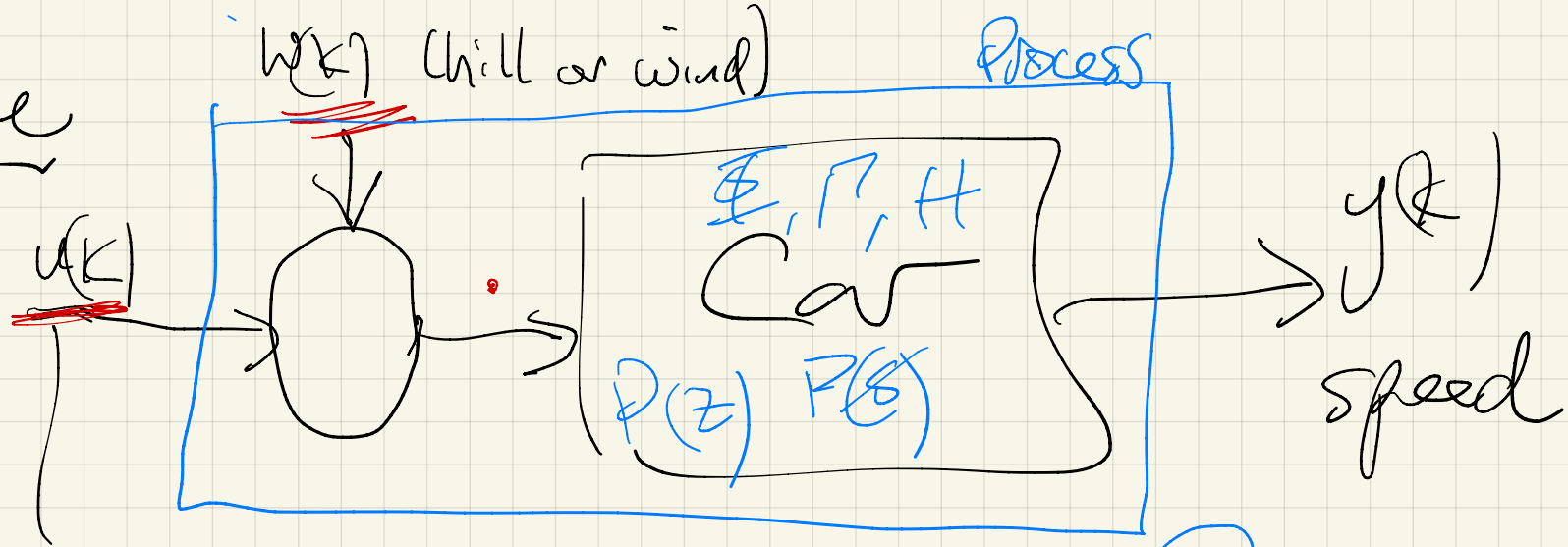
(poles on the $j\omega$ axis)

③ Other linear system models (e.g. a decaying oscillation)

Where is the disturbance entering?



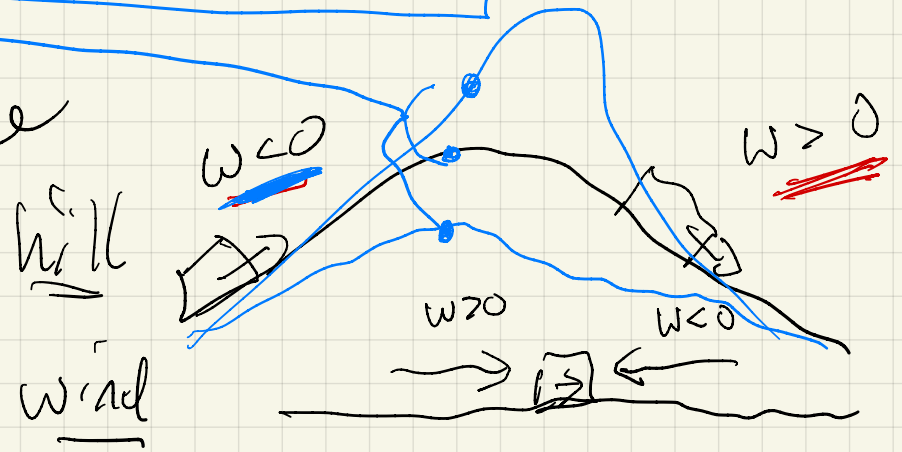
Example



Can we use one of the w(k) dynamics cases earlier

Commanded Throttle level

We do not know which will we encounter

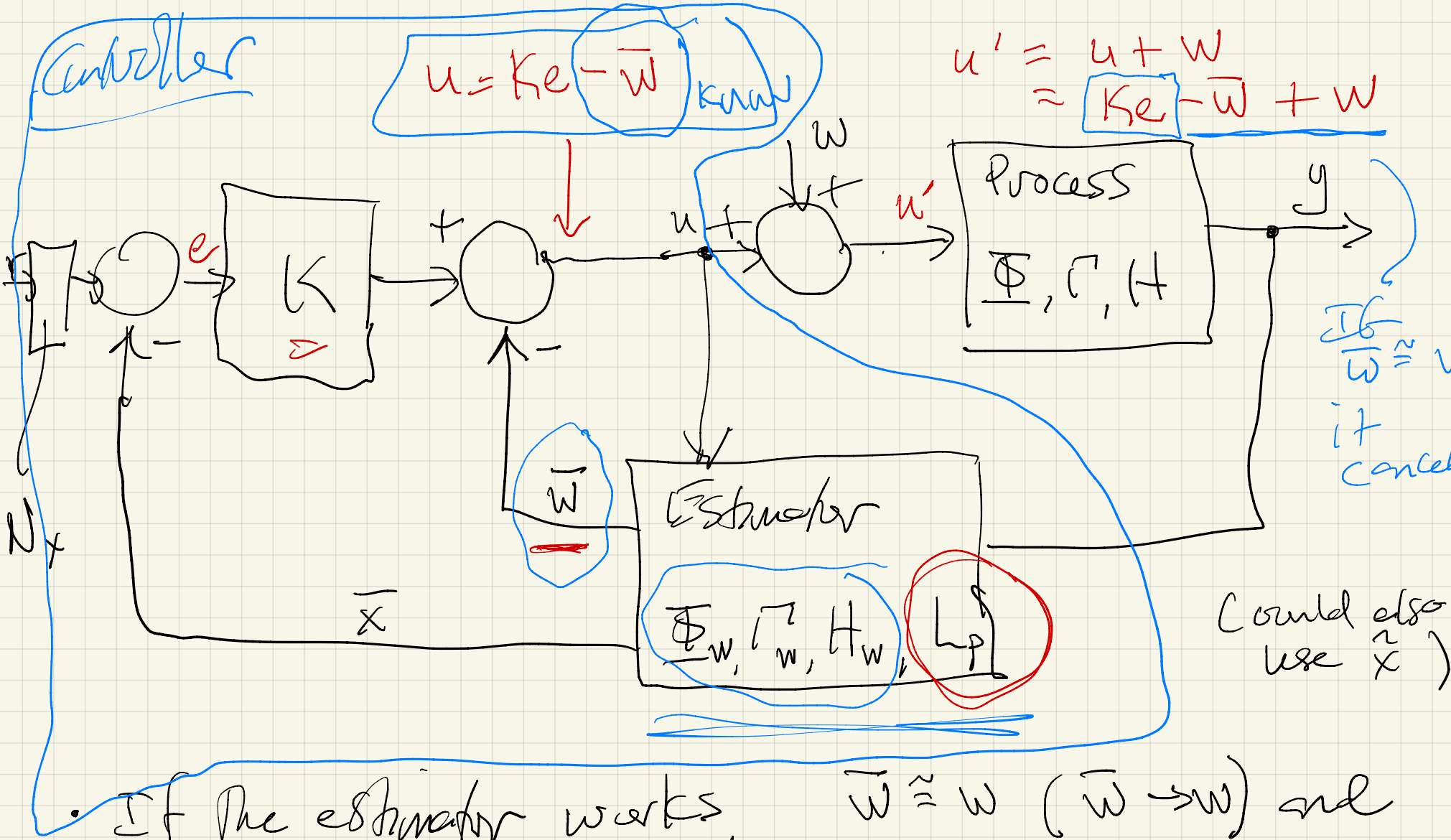


Estimation...

- Try to estimate the disturbance and then cancel it

- Need a model of the disturbance signal $w(k)$, and start in CT

- Recall options: white noise, OEC, LSM



- If the estimator works, $\bar{w} \approx w$ ($\bar{w} \to w$) and get a cancellation of w, the disturbance

- Start with example of constant w (any constant)

$$\begin{bmatrix} \dot{x} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} F & G \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} + \begin{bmatrix} G \\ 0 \end{bmatrix} u$$

- Sample, ZOH

$$\begin{bmatrix} x(k+1) \\ w(k+1) \end{bmatrix} = \Phi_w \begin{bmatrix} x(k) \\ w(k) \end{bmatrix} + \Gamma_w u$$

$$y = H x(k) \quad , \quad H_w = [H \; 0]$$

(can't measure w)

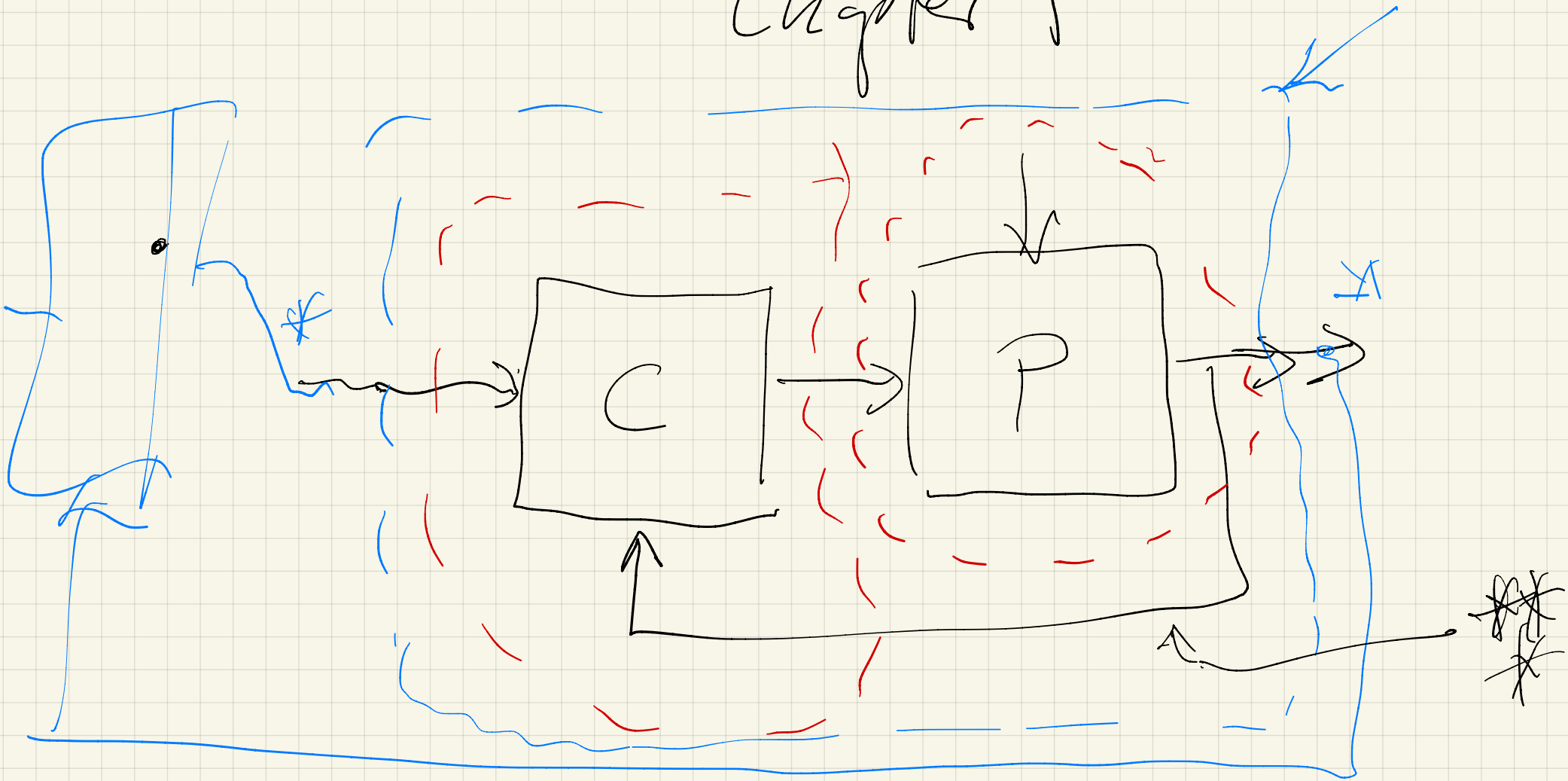
1. add disturbance
2. Model it
3. From $\bar{\Phi}_w, \Gamma_w$ Hw
4. Construct estimator.
5. Use $\bar{w}$ to modify the control input to cancel w

(bad)

* IF model for $w(t)$ is good, this method can be very effect (in)

Philosophy of Control

- Reference: K. Passino, "Biomimicry ..." at my website, Chapter 1



Modeling: (Do I even have a model?)

- Never, ever, a perfect model

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

$y(k) = H x(k)$

Is it accurate enough to design an effective feedback controller

Many implications from this!!!

- Can simulate very complex models and do comp. analysis

- Only models in certain forms can be used in mathematical analysis due to tractability problems
  - Can work with:
    - Linear models
    - Some nonlinear models (eg. $\dot{x} = f(x) + g(x)u$, $y = h(x)$)
  - Generally, difficult/impossible to work with nonlinear $\dot{x} = F(x, u)$, $y = H(x, u)$

Controller synthesis:

- Goals — "closed-loop specifications"
  (sometimes goals can be softened
  or they may be made
  more challenging)
- Systematic synthesis procedures
  generally depend on the model
  being in a certain form
  - CT
  - ED
  - ⋮

model

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

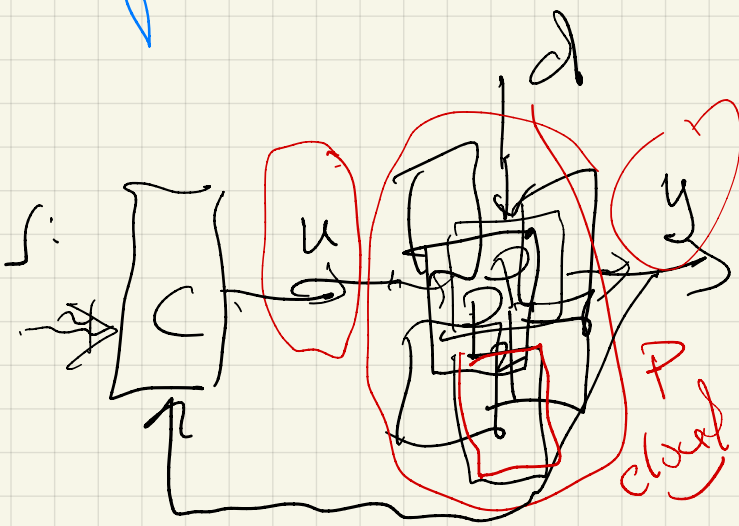
- Robustness ~~is almost~~ always a goal (eg. that the ECS will perform well in spite of uncertainty — like disturbances, inaccuracies, noise)

eg. hills in the auto cruise control problem

in model of the process

sensor noise

How to think about this:



- Theory for this is called "Robust Control Theory"

Introduction to robust control (: sensitivity analysis)

I recommend book:

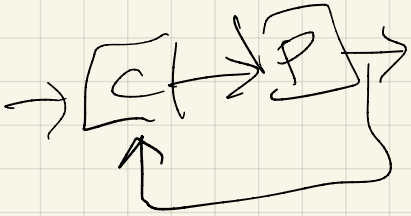
K.J Astrom, R. Murray at Caltech

- free .pdf at

- you already have the pre-req. to read this book

- Chapters on RL ; SRAS.

Control system verification:



1. Mathematics for analysis
2. Computational analysis

— both of these in class

3. Implementation studies (try C on many P)

When do you need each of these?

- 1 and 2 are not done if you do not have a model
- 1, 2, 3 are used in safety-critical appli

Multivariable and Optimal Control:

(1) Consider $m = p = 1$ (one input, one output) case

But all that held for the MIMO

(2) Choosing desired root locations (up till now, needed for estimators and regulators) can be difficult.

* We are going to address these issues...

- Here, will need design iterations on a "cost function" J (a measure of how well a system is behaving)

★ "Optimality" is only measured via J (not necessarily a measure of what you think of as "best")

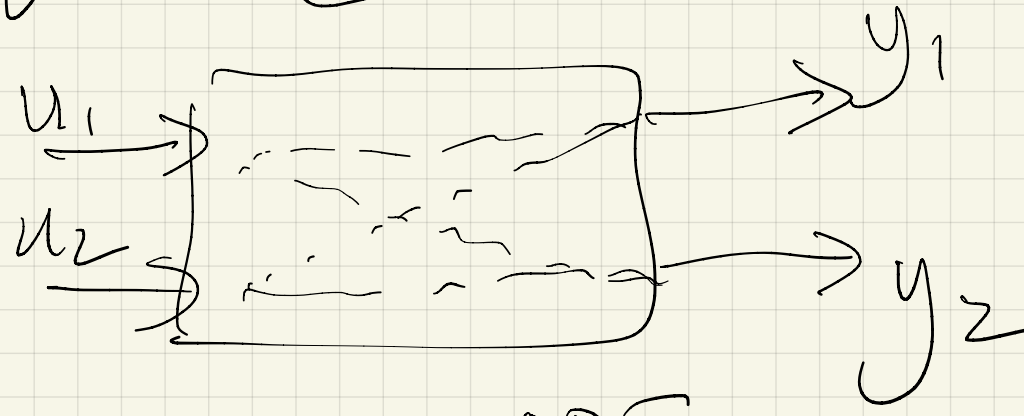
Note

(1) Parameters of J will allow for some degree of compromise between control effort and speed of response

+ will guarantee AS

[2] + great Matlab tools

Decoupling — Try to simplify. The difficulty for the MIMO case is coupling. Eg., u_1 and u_2 both affect x_1 and x_2 AND then y_1 and y_2



..... = ODEs

Time-varying optimal control:

- Control law

$$u = \underbrace{-K}_{m \times n} x \qquad (m \times 1 = m \times n \; n \times 1)$$

We want to design this!

- Process

$$x(k+1) = \underbrace{\Phi}_{n \times n} x(k) + \underbrace{\Gamma}_{n \times m} u(k) \qquad (n \times 1 = n \times n \; n \times 1 + n \times m \; m \times 1)$$

- Want to find K so that the cost function

$$J = \frac{1}{2} \sum_{k=0}^{N} \left[x^T(k) Q_1 x(k) + u^T(k) Q_2 u(k) \right]$$

(J: 1×1; $x^T(k)$: $1 \times n$, Q_1: $n \times n$, $x(k)$: $n \times 1$; $u^T(k)$: $1 \times m$, Q_2: $m \times m$, $u(k)$: $m \times 1$; $x^T(k) Q_1 x(k)$: 1×1; $u^T(k) Q_2 u(k)$: 1×1)

state weighting form

input weighting form

is minimized

(state small, input small)

$Q \geq 0$

$n=1 \Rightarrow u Q_2 u = Q_2 u^2, \; Q_2 > 0$

$n=1 \Rightarrow x Q x = Q x^2, \; Q \geq 0$

* Want to find a relationship between $u(k)$ and $x(k)$ (i.e. $u = -Kx$) so that when $u(k)$ is put in the process we get the smallest J

Note:

① $Q_1 = Q_1^T$, $Q_2 = Q_2^T$ (symmetric)

"weighting matrices" chosen by the designer (you!)

② often choose diagonal

* $Q_1 = \begin{bmatrix} q_1^1 & & 0 \\ & \ddots & \\ 0 & & q_n^1 \end{bmatrix}$, $Q_2 = \begin{bmatrix} q_1^2 & & 0 \\ & \ddots & \\ 0 & & q_m^2 \end{bmatrix}$

$n \times n$ $\quad$ $n \times m$

Then, pick

$$q_i^1 \geq 0, \quad i = 1, \ldots, n$$

$$q_i^2 > 0, \quad i = 1, \ldots, m$$

* Choose these values to set the relative importance of states and controls

* Usually, must iterate to find good q_i^1, q_i^2

* Initial choices for Q_1 and Q_2?

Wild guess $Q_1 = I_{n \times n}$, $Q_2 = I_{m \times m}$

Rewrite

$\bar{J}(x,u)$

Minimize

$$\bar{J} = \frac{1}{2}\sum_{k=0}^{N}\left[x^T(k)Q_1x(k) + u^T(k)Q_2u(k)\right]$$

subject to the constraint

$$-x(k+1) + \underline{\underline{\Phi}}\,x(k) + \Gamma u(k) = 0,$$

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

- A "constrained minimization problem"
  → can be solved with a Lagrange multiplier approach

- Let "Lagrange multiplier", $\underline{n \times 1}$; $\lambda(k+1)$, $k \geq 0$

- Define

Quadratic optimization problem ($J = x^2 + u^2 + \lambda x + \lambda u \cdots$)

$$J' = \sum_{k=0}^{N} \left[\frac{1}{2} x(k+1) Q_1 x(k) + \frac{1}{2} u^T(k) Q_2 u(k) \right.$$

$$\left. + \lambda^T(k+1) \underbrace{\left(-x(k+1) + \Phi x(k) + \Gamma u(k) \right)}_{\text{constraint}} \right]$$

$J'(x, u, \lambda)$

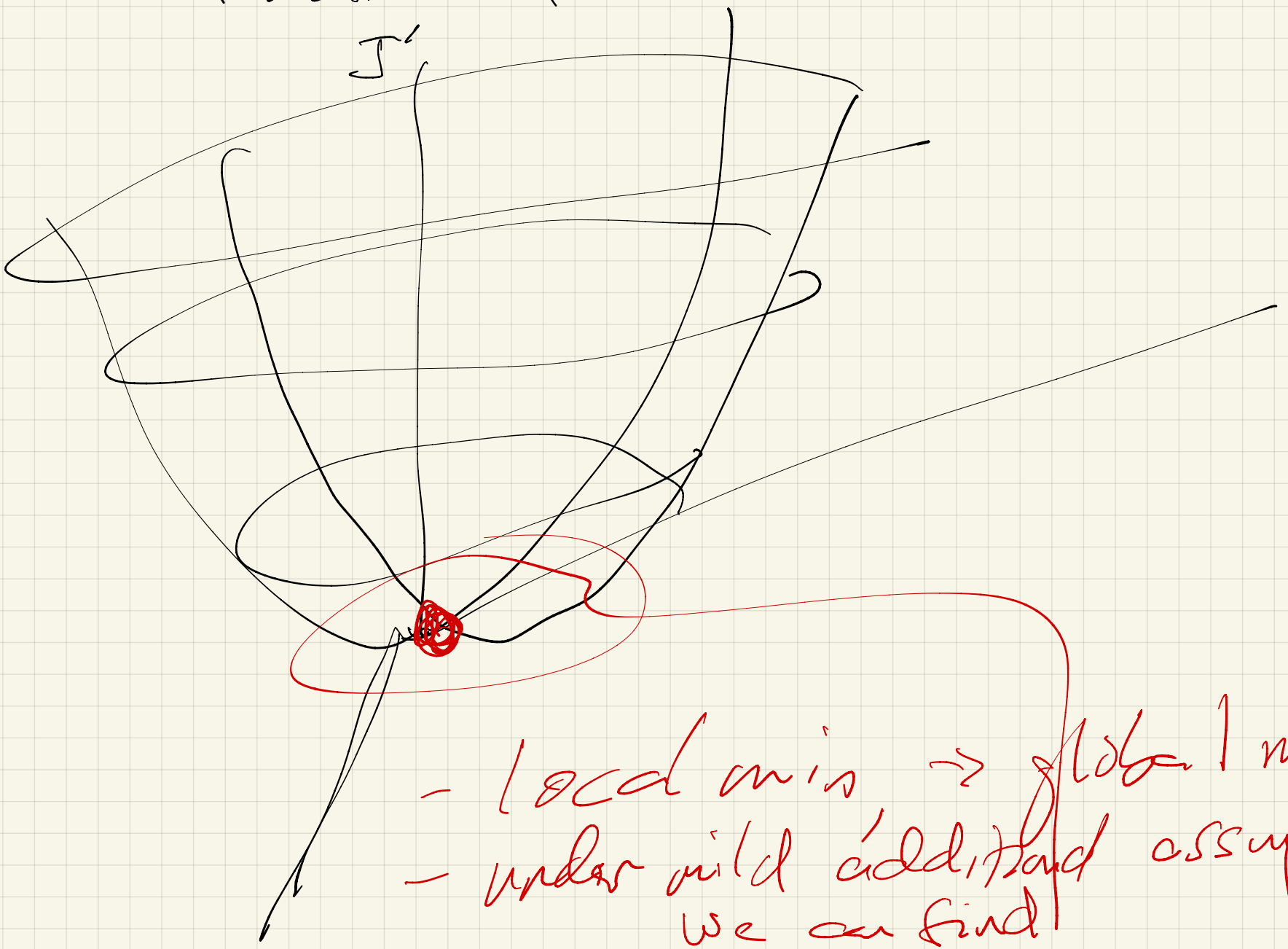
an unconstrained problem, but new function

$$\min J', \text{ w.r.t } x, u, \lambda$$

$n \times 1$, $m \times 1$, $n \times 1$

Note: If the constraint is met, then $= 0$ and $J = J'$

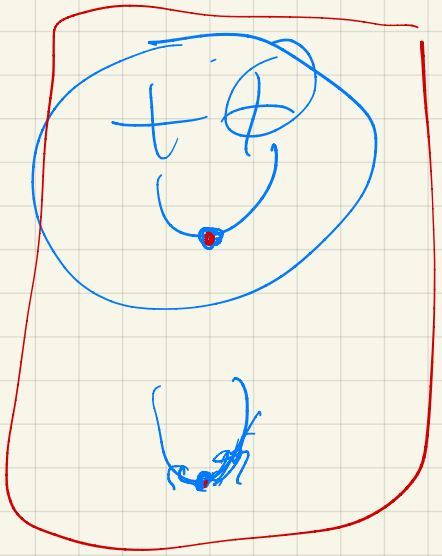
J' is a quadratic problem is unconstrained

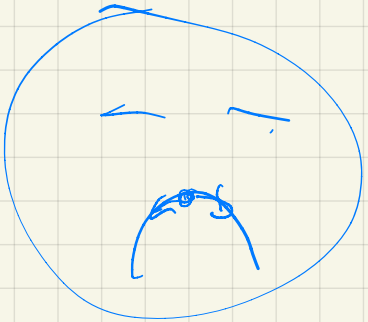


- local min → global min
- under mild additional assumption we can find

At the bottom of the bowl

$J'(x(k), u(k), \lambda(k+1))$





$$\frac{\partial J'}{\partial u(k)} = u^T Q_2 + \lambda^T(k+1)\Gamma = 0 \quad \text{"control equations"}$$

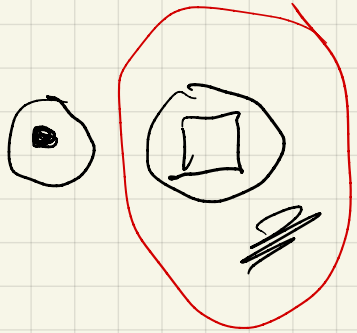
$$\frac{\partial J'}{\partial \lambda(k+1)} = -x(k+1) + \Phi x(k) + \Gamma u(k) = 0 \quad \text{"state equation"}$$

$$\frac{\partial J'}{\partial x(k)} = x^T(k) Q_1 - \lambda^T(k) + \lambda^T(k+1)\Phi = 0$$

"adjoint eq."

- Adjoint, take the transpose, get

$$\lambda(k) = \Phi^T \lambda(k+1) + Q_1 x(k)$$



and know

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

- From control equations, and above,

$$Q_2^T u(k) + \Gamma^T \lambda(k+1) = 0 \quad (= Q_2^T)$$

$$u(k) = -Q_2^{-1} \Gamma^T \lambda(k+1)$$

if Q_2^{-1} exists due to symmetry and $q_2^i > 0$ $\forall i = 1, \ldots, m$

- From the state equations, and above,

$$\lambda(k+1) = \Phi^{-T}\lambda(k) - \Phi^{-T} Q_1 x(k)$$

All → set of coupled linear difference equations

- Defines optimal solution of $x(k), u(k), \lambda(k)$ given initial conditions
- Initial conditions on $x(0)$ must be known
- But $\lambda(0)$ is not normally known so usually try to use $\lambda(N)$

How? We had

$$J = \frac{1}{2}\sum_{k=0}^{N}\left(x^T(k) Q_1 x(k) + u^T(k) Q_2 u(k) \right)$$

$\to 0$

- $u(k)$ affects $x(k+1)$ (one step ahead)

  via the process model

  (but it does not affect $x(k)$)

- So, if x arrives at $x(N)$, meaning $u(N)$ can do nothing about it $\Rightarrow$ best to pick $u(N) = 0$ to zero out

  (otherwise this would add to J)

- From the control equations, at $k = N$

$$u^T(N) Q_2 + \lambda^T(N+1) \Gamma = 0$$

$u^T(N) Q_2 = 0$

suggests $p = 2k$ → $\lambda^T(N+1) = 0$

- From the adjoint equations

$$x^T(k-1) Q_1 - \lambda^T(k) + \lambda^T(k+1) \Phi = 0$$

Let $k = N$

$$\lambda(N) = Q_1 x(N)$$

- Know $x(0)$ and $u(k) \Rightarrow$ know $x(N)$

Let's take a math break... Where are we headed?

Min

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[x^T(k)\,Q_1\,x(k) + u^T(k)\,Q_2\,u(k)\right]$$

Defines Best

to get K, in

$$u = -Kx \qquad (m\times 1 = m\times n \;\; n\times 1)$$

control law

Note:

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

- No pole placement "place()" via
- But you do have to pick Q_1, Q_2

$\Rightarrow$ Get an "optimal controller" in the sense that K is the one that minimizes J

Q_1 PSD (positive semi-definite)

Q_2 PD (positive definite)

Given Q_1, Q_2 methods

- Assume

$$\lambda(k) = S(k)\, x(k)$$

- Need to solve a two-point boundary problem (0, N)

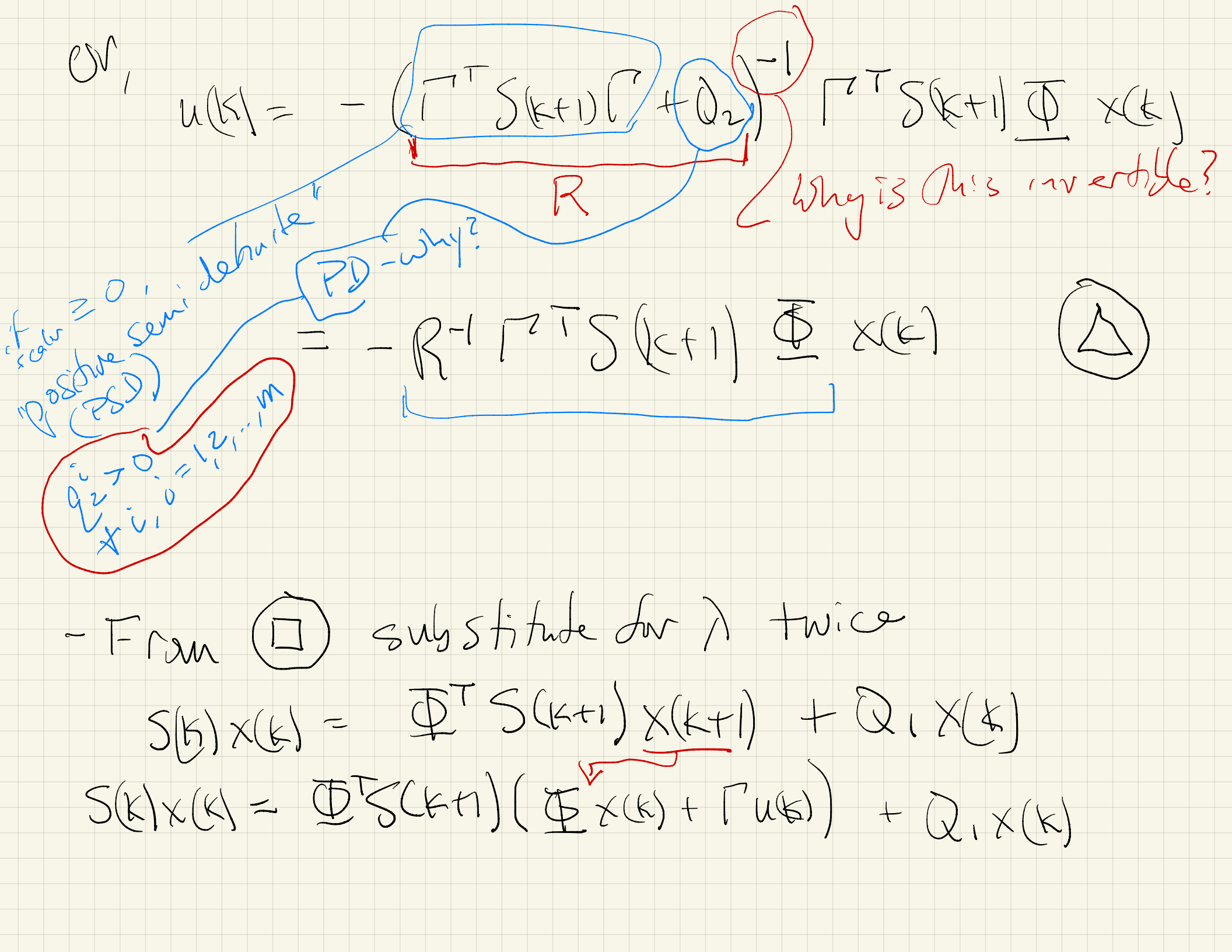
- Seek to transform the problem to a single-point boundary

- Control equations (with a transpose)

$$Q_2 u(k) = -\Gamma^T \overbrace{S(k+1)\, x(k+1)}^{\lambda(k+1)}$$

$$= -\Gamma^T S(k+1)\left(\Phi\, x(k) + \Gamma\, u(k)\right)$$

or, $\left(\Gamma^T S(k+1)\,\Gamma + Q_2\right) u(k) = -\Gamma^T S(k+1)\,\Phi\, x(k)$


or,
$u(k) = -\left(\Gamma^T S(k+1)\Gamma + Q_2\right)^{-1} \Gamma^T S(k+1)\Phi\, x(k)$
R
Why is this invertible?
PD - why?
$Q_2 \geq 0$, positive semi definite (PSD)
$Q_2 > 0,\ i = 1, 2, \ldots, m$
$= -R^{-1}\Gamma^T S(k+1)\Phi\, x(k)$
- From □ substitute for λ twice
$S(k)x(k) = \Phi^T S(k+1)x(k+1) + Q_1 x(k)$
$S(k)x(k) = \Phi^T S(k+1)\left(\Phi x(k) + \Gamma u(k)\right) + Q_1 x(k)$

Using (Δ),

$$S(k)\,x(k) = \Phi^T S(k+1)\left[\Phi x(k) - \Gamma R^{-1}\Gamma^T S(k+1)\,\Phi\, x(k)\right] + Q_1 x(k)$$

(the bracketed $u(k)$ from (Δ))

$$\left[S(k) - \Phi^T S(k+1)\,\Phi + \Phi^T S(k+1)\,\Gamma R^{-1}\Gamma^T S(k+1)\,\Phi - Q_1\right] x(k) = 0$$

This must hold for all $x(k) \in \mathbb{R}^n$

$$\Rightarrow [\,\cdot\,] = 0 \Rightarrow$$

$$S(k) = \Phi^T\left[S(k+1) - S(k+1)\,\Gamma R^{-1}\Gamma^T S(k+1)\right]\Phi + Q_1 \qquad (\Delta\Delta)$$

A "backward difference eq." ⇒ it goes from $k+1$ to k

Backward in time!

- Or, sometimes written as

$$S(k) = \underline{\Phi}^T M(k+1)\,\underline{\Phi} + Q_1$$

$$M(k+1) = S(k+1) - S(k+1)\Gamma\left[Q_2 + \Gamma^T S(k+1)\Gamma\right]^{-1}\Gamma^T S(k+1)$$

(oo) (ΔΔ) is called "discrete Riccati equation"

- Can be difficult to solve as it is nonlinear due to $S(k+1)\Gamma R^{-1}\Gamma^T S(k+1)$

Introduces squared terms

- often $m < n$ so R is often not of high dimension
  $\Rightarrow R^{-1}$ is easy to calculate

To solve, need $S(k+1)$, $k+1 = N \Rightarrow k = N-1$

so we need $S(N)$

Above $\lambda(N) = Q_1 \times(N) = S(N) \times(N)$

$\Rightarrow S(N) = Q_1$

Solution of ⊙⊙ proceeds backwards...

Backwards in time ↓

Time Step

$k+1 = N$

$(k = N-1)$

⋮

Variables

$S(N) = Q_1$ → $S(N-1) = S(k)$

$M(N)$

$S(N-1)$ gives $M(N-1)$

$S(N-2)$

from ΔD

How do we get the control $u(k)$,

$$u = -\underbrace{K(k)}_{m \times n} \underbrace{x(k)}_{n \times 1}$$

$m \times 1$

$K(k)$: 1×1 scalar index

Use (A)

$$K(k) = \left[Q_2 + \Gamma^T S(k+1)\Gamma\right]^{-1} \Gamma^T S(k+1)\Phi$$

$=$ L: An optimal time-varying feedback gain

Must compute backwards in time

Notes

(1) Can compute $K(k)$ $\Rightarrow$ for a finite-time problem, can store gains and start at $k=0$

(2) No knowledge of $x(0)$ was needed for the above calculations
$\Rightarrow$ this will work independent of the initial conditions

Concern:

If $T = 0.001$ (realistic) $\Rightarrow$ run the control system for 24 hrs

150 stored for $K(k)$ for $k = N$

$\rightarrow \frac{150}{0.001}(60)(60)(24)$ $\rightarrow$ big nr. for $n=150$, $m=10$

Linear Quadratic Regulator (LQR)

- let $N \to \infty$ ("infinite time problem")
  - In the steady state

$$S(k) = S(k+1) \Rightarrow$$

call both of them S_∞

- Riccatti equation

$$S_\infty = \Phi^T \left[S_\infty - S_\infty \Gamma \left(R^{-1} \Gamma^T S_\infty \right) \right] \Phi + Q_1$$

"Algebraic Riccatti equation"

(Matlab can solve this for S_∞)

- Could have had in J,

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left(x^T Q_1 x + u^T Q_2 u + 2x^T Q_3 u\right)$$

$x^T Q_3 u$: $1\times n$, $n\times m$, $m\times 1$

"penalize cross-terms"

Could go back and rederive everything to get the same basic results...

- Very good computational approaches for finding the min J to provide K by computing S_∞

- Matlab:

"dlqr" discrete LQR

[K, S, E] = dlqr(Φ, Γ, Q1, Q2, Q3)

also lqr for CT

dlqr[K] =

mxn

nxn nxm nxn mxm nxm

Ricatti eq. solution

E = closed-loop eigenvalues, E = eig(Φ - ΓK)

u = -Kx

mx1 mxn nx1

If all q_i and q_i are same => 2 param.

n+m terms total

Design parameters Q1 and Q2 often diagonal

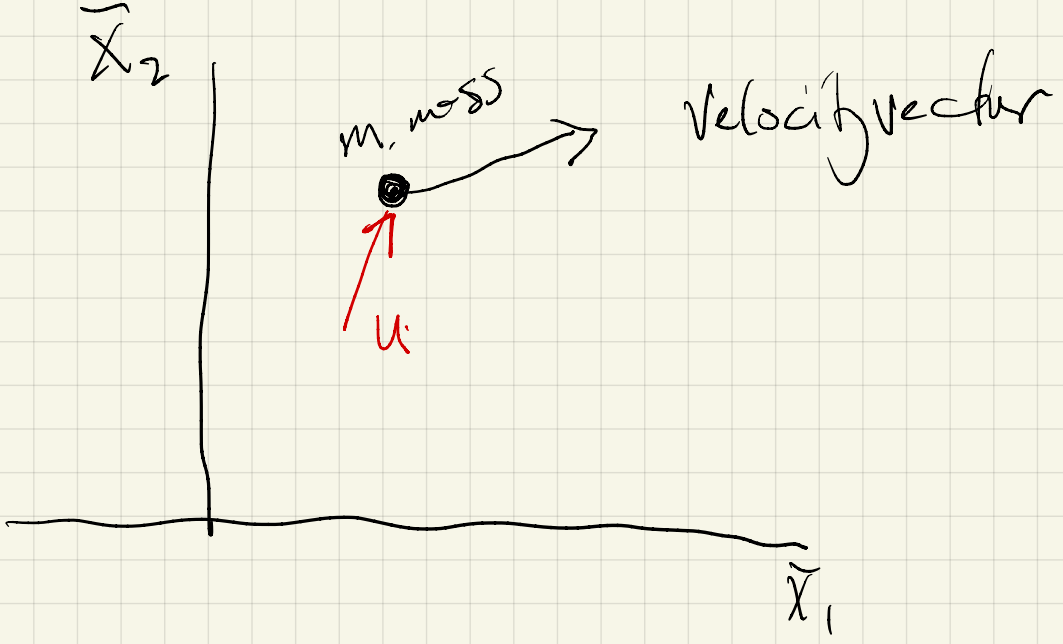
Robot example:

$\bar{x}$ position $\in \mathbb{R}^2$

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

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

m mass



$$\underset{4\times 1}{\begin{bmatrix}\dot{\bar{x}}\\ \dot{v}\end{bmatrix}} = \underbrace{\begin{bmatrix} 0_{2\times 2} & I_{2\times 2}\\ 0_{2\times 2} & 0_{2\times 2}\end{bmatrix}}_{4\times 4}\underbrace{\begin{bmatrix}\bar{x}\\ v\end{bmatrix}}_{4\times 1} + \underbrace{\begin{bmatrix} 0_{2\times 2}\\ \frac{1}{m} I_{2\times 2}\end{bmatrix}}_{4\times 2}\underbrace{u}_{2\times 1}$$

$$\underset{2\times 1}{y} = \underbrace{\begin{bmatrix} 0_{2\times 2} & I_{2\times 2}\end{bmatrix}}_{2\times 4}\underbrace{\begin{bmatrix}\bar{x}\\ v\end{bmatrix}}_{4\times 1}$$

Assumes velocity is measured (not position)

(or we could use FST)

$$\underbrace{\begin{bmatrix} I_{2\times 2} & 0\\ 0 & I_{2\times 2}\end{bmatrix}}_{4\times 4}\begin{bmatrix}\bar{x}\\ v\end{bmatrix} = \underset{4\times 1}{y}$$

can measure $\bar{x}$ and v

$\begin{bmatrix}\bar{x}\\ v\end{bmatrix}_{4\times 1}$ ($\bar{x}$: 2×1, v: 2×1)

Consider LQRobust.m in Matlab

Review:

(1) Modeling ← Physics / Data | → DT Linear SS

Based on pole-placement:

(2) Control law design ($u = -Kx$) FSF

(3) Estimator design $\hat{x}$ (pred./curr.)

Combine (2) + (3) → Regulator

(4)

(5) Optimal control (LQR), FSF

(6) Optimal estimation (also, comb. (5) + (6))

Now — *

Optimal estimation

- Dual of optimal control / LQR, but now for an estimator
- Approach: RLS (parameter estimation problem)

→ have a "state estimation problem"

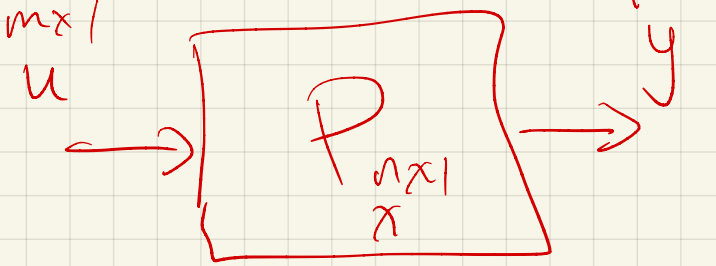
(also, KF / RLS is a special case of Gauss-Newton methods)

The Kalman Filter

- Process:

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

$$\underset{p\times 1}{y(k)} = \underset{p\times n}{H}\,\underset{n\times 1}{x(k)} + \underset{p\times 1}{v(k)} \quad (2)$$



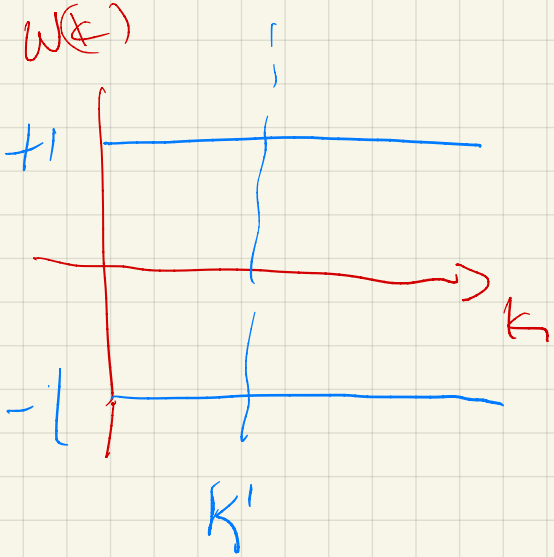
MIMO

(1) $w(k)$ is called "process noise"

(2) $v(k)$ is called "measurement noise"

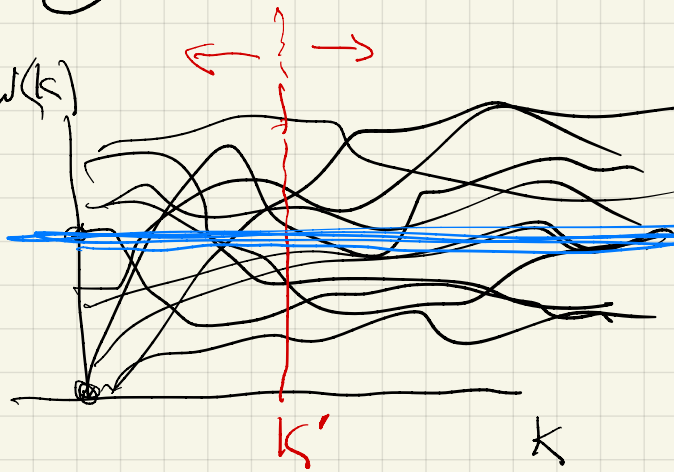
- Due to (1) & (2) this process model is more general than earlier.

① Noise processes are assumed to be zero mean for each $k \geq 0$



$$E[w(k)] = 0 \quad (m \times 1)$$

$$E[v(k)] = 0 \quad (p \times 1)$$



$\Rightarrow E[w(k')] = 0 \quad \forall k'$

② Also, $w(k)$ and $v(k)$ have no time correlation (are "white") so

$$E[w(i)w^T(j)] = 0 \quad (m \times m) \quad ; \quad E[v(i)v^T(j)] = 0 \quad (p \times p) \quad \forall i \neq j$$

$(m \times 1)(1 \times m) \rightarrow m \times m$

$(p \times 1)(1 \times p) \rightarrow p \times p$

(2) Also, covariances ("noise levels") are $\forall k$, (since zero mean),

$$E\left[\underset{m\times 1}{w(k)}\ \underset{1\times m}{w^T(k)}\right] = \underset{m\times m}{R_w}$$

$$E\left[\underset{p\times 1}{v(k)}\ \underset{1\times p}{v^T(k)}\right] = \underset{p\times p}{R_v}$$

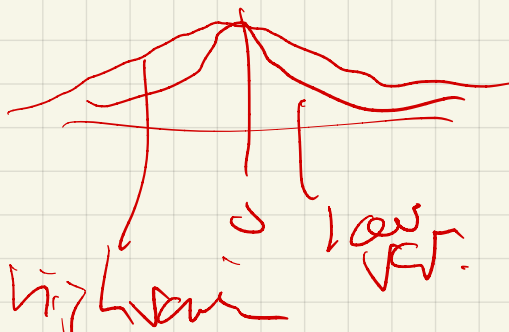
Need to know values of R_w, R_v here

$$w(k) = \begin{bmatrix} w_1(k) \\ \vdots \\ w_m(k) \end{bmatrix}_{m\times 1}$$

⇒ First diagonal element, $E[w_1^2(k)]$ - variance

To compute ... e.g.

$E[w_i(k)\, w_j(k)]$ is element of R_w (its i-j^{th} component)



Goal:

- Perform state estimation with $w(k)$, $v(k)$ present

Approach

- Develop a stochastic estimation approach for "parameters", then apply it to the above state estimation problem

RLS:

- Showed

$$\hat{\theta}_{WLS} = (\Phi^T W \Phi)^{-1} \Phi^T W Y \quad (*)$$

(W diagonal, elements > 0)

called "batch" Least Sq.

Had used this for modeling (model development (DT ARMA)

- Now, data arrives sequentially
  - want to estimate immediately
  - Cannot just expand the amount of memory used to store $\underline{\Phi}$ and Y as $K \to \infty$
- Collected N data points, Collect one more

Effect on $\underline{\Phi}^T W \underline{\Phi}$ and $\underline{\Phi}^T W Y$?

$\Phi^T W \Phi$

⇒ weights emphasize most recent data The most

$$W = \begin{bmatrix} w(n) & & & \\ & w(n+1) & & \\ & & \ddots & \\ & & & w(N) \end{bmatrix}$$

Below,

$$\begin{bmatrix} w(n) \\ < \\ w(n+1) \\ < \\ \cdots \\ < w(N) \end{bmatrix}$$

→ Assume $w(k) = a\,\gamma^{N-k}$ (a, γ scalars, $a>0$, $\gamma>0$)

① if $a = \gamma = 1$, $w(k) = 1 \ \forall k$ ⇒ ordinary (unweighted LS)

② if $a = 1-\gamma$, $\gamma < 1$, $w(k) = (1-\gamma)\gamma^{N-k}$

✓ $n \uparrow \Rightarrow w(n) \uparrow$

$k = n \Rightarrow w(n) = (1-\gamma)\gamma^{N-n}$

$k = n+1 \Rightarrow w(n+1) = (1-\gamma)\gamma^{N-n-1}$

$N-n > N-n-1$

We had

$$Y(N) = [y(n), \ldots, y(N)]^T$$

$$\underset{2n\times 1}{\Phi}(N) = [\phi(n), \phi(n+1), \ldots, \phi(N)]^T$$

$$\varepsilon(N;\theta) = [e(n), \ldots, e(N)]^T$$

$$\underset{2n\times 1}{\theta} = [a_1, \ldots, a_n, b_1, \ldots, b_n]^T$$

$$\phi(k) = [-y(k-1) \quad -y(k-2) \quad -y(k-n)$$

$$u(k-1) \ldots, u(k-n)]^T$$

$$\underline{\Phi}^T(N+1) = \begin{bmatrix} \phi(1) & \cdots & \phi(N) & \phi(N+1) \end{bmatrix}$$

$$\underline{\Phi}^T(N+1)\, W\, \underline{\Phi}(N+1) =$$

$$\sum_{k=1}^{N+1} \phi(k)\, w(k)\, \phi^T(k) = \sum_{k=1}^{N+1} \phi(k)\, a\, \gamma^{N+1-k}\, \phi(k)$$

Split N and $N+1$ terms

$$\underline{\Phi}^T(N+1)\, W\, \underline{\Phi}(N+1) = \left(\sum_{k=1}^{N} \phi(k)\, a\, \gamma\, \gamma^{N-k}\, \phi^T(k) \right) +$$

$$\left(\phi(N+1)\, a\, \phi^T(N+1) \right)$$

$N+1-k$

for $k = N+1 \Rightarrow a\gamma^0 = a$

Define

$$P(N+1) = \left[\underbrace{\Phi^T(N+1)\, W(N+1)\, \Phi(N+1)}_{2n \times 2n} \right]^{-1}$$

so

$$P(N+1) = \left[\gamma P^{-1}(N) + \phi(N+1)\, Q\, \phi^T(N+1) \right]^{-1}$$

$P^{-1}(N)$ ← N

$N+1$

Need to take the inverse of the sum of two matrices

Matrix inversion lemma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

let

$$A = \gamma P^{-1}(N)$$

$$B = \phi(N+1) \triangleq \phi$$

$$C = \omega(N+1) \triangleq q$$

$$D = \phi^T(N+1) = \phi^T$$

so

$$\odot\ P(N+1) = \frac{1}{\gamma}P(N) - \frac{1}{\gamma}P(N)\phi\left(\frac{1}{q} + \phi^T\frac{P(N)}{\gamma}\phi\right)^{-1}\phi^T P(N)\frac{1}{\gamma}$$