Cyber-Physical Systems:

Basics of Control for Computer Scientists

March 26, 2019

Lecture @ University of Trieste

Jyo Deshmukh

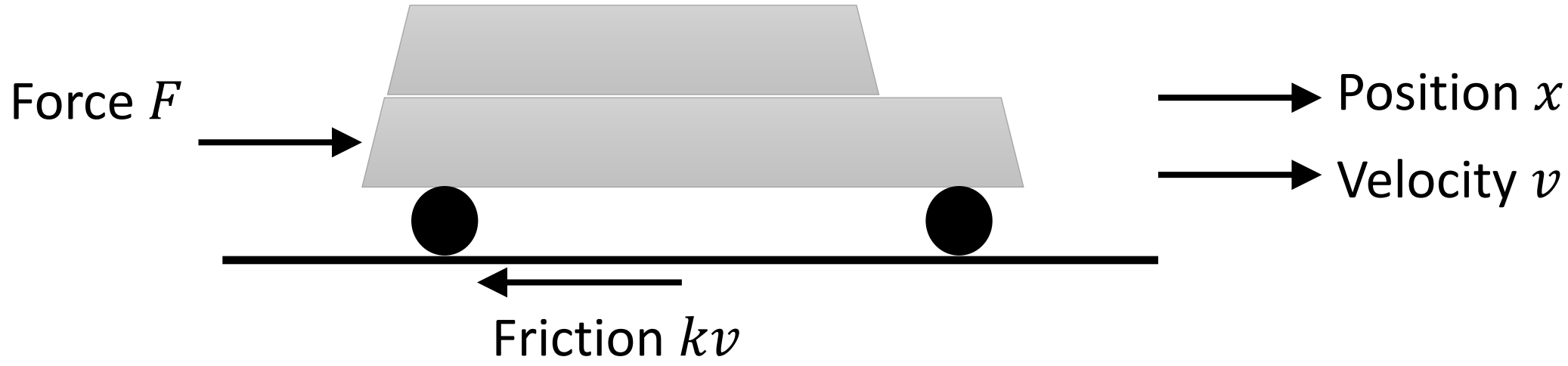
USC Viterbi
School of Engineering
Department of Computer Science

UNIVERSITY OF TRIESTE

Layout

- Introduction to basics of linear control (without any Laplace transforms)
- Introduction to nonlinear control
- Introduction to observer design

Model of a simple car



Newton's law of motion: $F - kv = m\frac{d^2x}{dt^2}; v = \frac{dx}{dt}$

Linear Systems

- Equation of simple car dynamics can be written compactly as:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -k/m \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [F]$$

- Letting $A = \begin{bmatrix} 0 & 1 \\ 0 & -k/m \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, we can re-write above equation in the form:

- $\dot{\mathbf{x}} = A\mathbf{x} + \mathrm{B}\mathbf{u}$, where $\mathbf{x} = [x \quad v]$, and $\mathbf{u} = [F]$

What does a control algorithm do?

- Given $\dot{\mathbf{x}} = A\mathbf{x} + \mathrm{B}\mathbf{u}$
  - Looks at the state $\mathbf{x}$
  - Decides what $\mathbf{u}$ should be fed to the "plant"

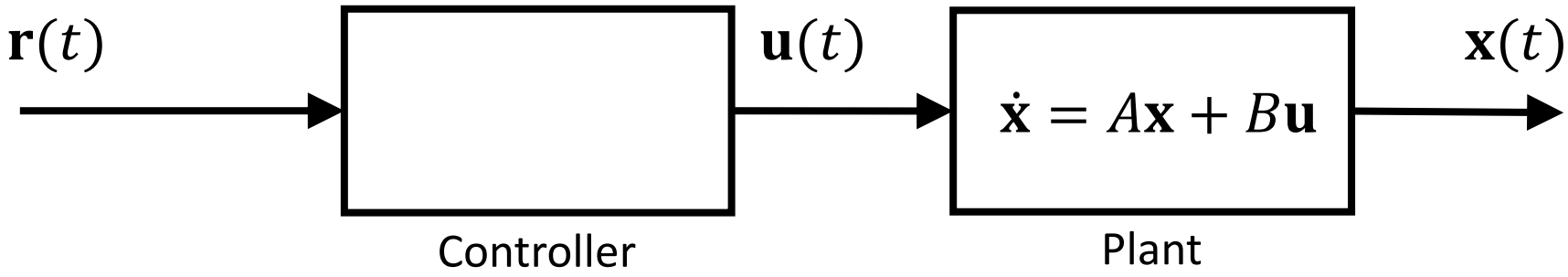
- Control objectives:
  - Reject disturbances (if there is some perturbation in state, making it get back to initial state)
  - Follow reference trajectories (if we want the system to have a certain $\mathbf{x_{ref}}$)
  - Make system follow some other "desired behavior"

Linear vs. Nonlinear Control

- Linear control
  - Very well-understood and mature area
  - More than 100 years of research
  - Assumes linear plant models

- Nonlinear control
  - Research area with fewer “solved problems”
  - Allows more general plant models
  - Harder, with more math!

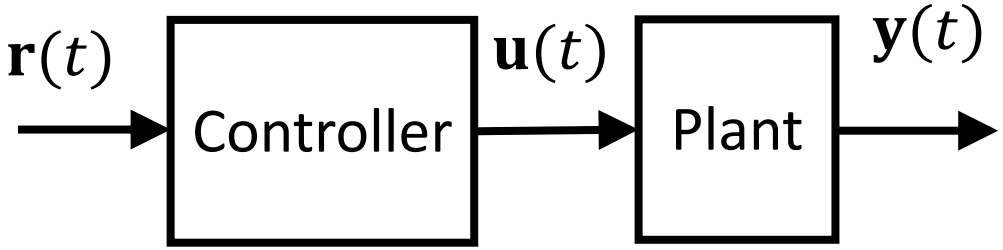
Reference Tracking



- Given a reference trajectory $\mathbf{r}(t)$, design $\mathbf{u}(t)$ such that $\mathbf{x}(t)$ closely follows $\mathbf{r}(t)$
- Applications: Making a thermostat reach a set-point, Path following robots, etc.

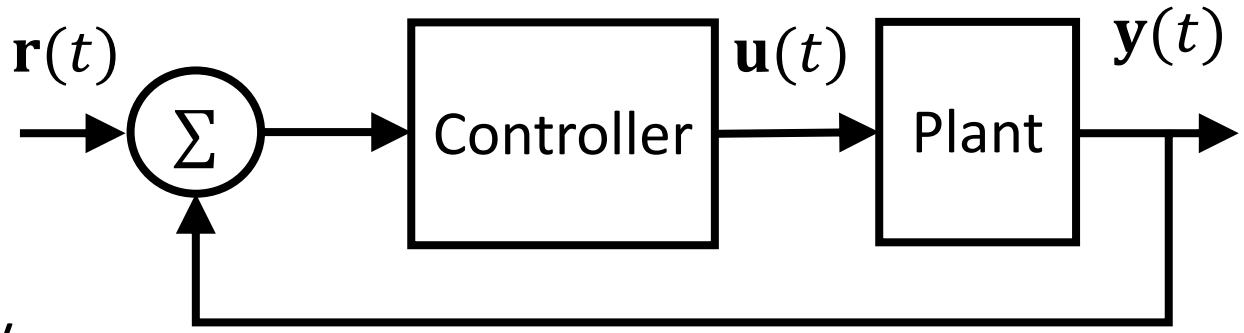
Open-loop control

- Open-loop or feed-forward control
  - Control action generally does not depend on plant output
  - Quite common in many CPS applications!
- Pros: Cheaper, may require fewer sensors
- Cons: Requires lots of tuning before acceptable quality Is achieved; may not be robust to disturbances

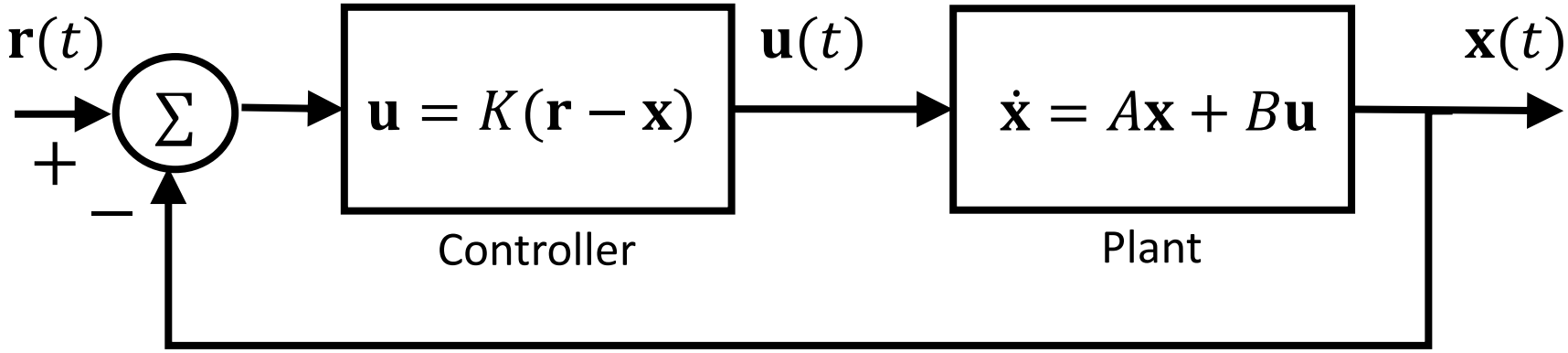


Closed-loop or Feedback Control

- Controller adjusts controllable inputs in response to observed outputs
  - Can respond better to variations in disturbances
  - Performance depends on how well outputs can be sensed, and how quickly controller can track changes in output
- Many different flavors of feedback control!

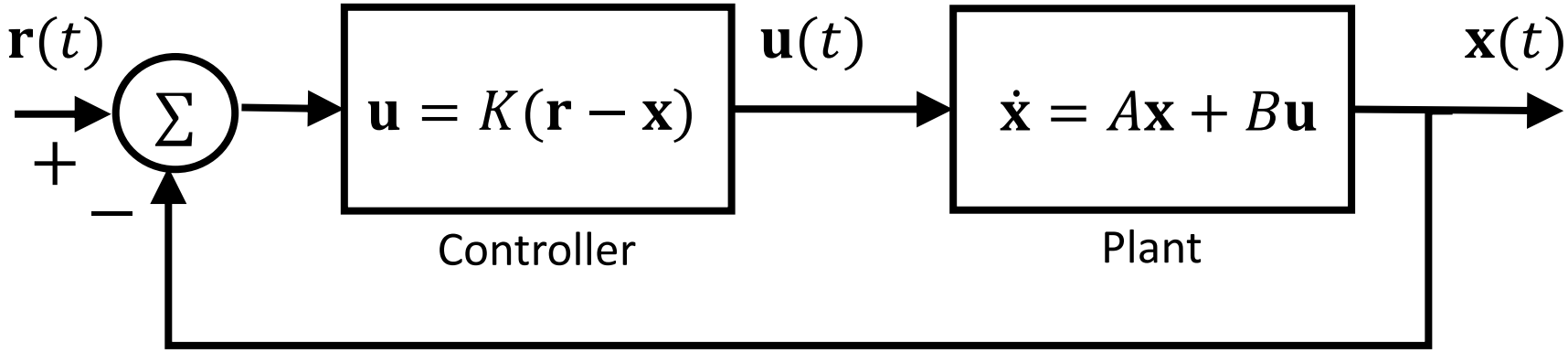


Simple Linear Feedback Control: Reference Tracking



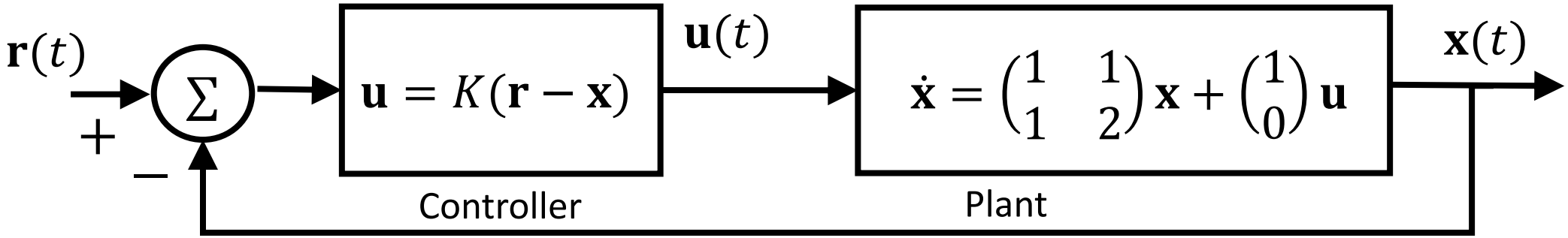
- Closed-loop dynamics: $\dot{\mathbf{x}} = A\mathbf{x} + BK(\mathbf{r} - \mathbf{x}) = (A - BK)\mathbf{x} + BK\mathbf{r}$
- Pick K such that closed-loop system has desirable behavior
- Main idea: Suppose, the system $\dot{\mathbf{x}} = (A - BK)\mathbf{x}$ is stable, then as time $t \to \infty$, $\mathbf{x}$ approaches some constant multiple of $\mathbf{r}$

Linear Feedback Control: Pole placement



- To make closed-loop system stable, pick K such that eigenvalues of $(A - BK)$ have negative real-parts
- Controller designed this way also called ***pole placement*** controller

Designing a pole placement controller



- Note eigs$(A) = 0.382, 2.618 \Rightarrow$ unstable plant!
- Let $K = (k_1 \quad k_2)$. Then, $A - BK = \begin{pmatrix} 1-k_1 & 1-k_2 \\ 1 & 2 \end{pmatrix}$
- eigs$(A - BK)$ satisfy equation $\lambda^2 + (k_1 - 3)\lambda + (1 - 2k_1 + k_2) = 0$
  - Suppose we want eigenvalues at $-5, -6$, then equation would be: $\lambda^2 + 11\lambda + 30 = 0$
  - Comparing two equations, $(k_1 - 3) = 11$, and $(1 - 2k_1 + k_2) = 30$
  - This gives $k_1 = 14, k_2 = 57$. Thus controller with $K = (14 \quad 57)$ stabilizes the plant!

Linear Quadratic Regulator

- Pole placement involves heuristics (we arbitrarily decided where to put the eigenvalues)
- Principled approach is to put the poles such that the closed-loop system optimizes the cost function:

$$J_{LQR} = \int_0^\infty [\mathbf{x}(t)^T Q\mathbf{x}(t) + \mathbf{u}(t)^T R\mathbf{u}(t)]\, dt$$

- $\mathbf{x}(t)^T Q\mathbf{x}(t)$ is called state cost, $\mathbf{u}(t)^T R\mathbf{u}(t)$ is called control cost
- Given a feedback law: $\mathbf{u}(t) = -K_{\text{lqr}}\mathbf{x}(t)$, K_{lqr} can be found precisely
- In Matlab, there is a simple one-line function `lqr` to do this!

Linear Control System Basics

- Closed-loop control system has the following form:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$

- $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$:describes state evolution
- $\mathbf{y} = C\mathbf{x} + D\mathbf{u}$: describes how states are observed, D usually set to $\mathbf{0}$

$\mathbf{x}$: State [Internal to the process being controlled]
$\mathbf{u}$: Control Input [actuator command]
$\mathbf{y}$: Output [what sensor reads]

Controllability

- Can we always choose eigenvalues to find a stabilizing controller? NO!
- For $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, what if A is unstable, and B is $[0 \dots 0]^T$?
  - No controller can ever stabilize the system
- How do we determine for a given A, B whether there is a controller?
- Controllability:
  - Can we find the condition on the system design that ensures that we can always move the system to whichever state/output we want?
  - Important question that affects which actuators we pick for the system

Checking Controllability

- Find *controllability matrix* R
- $R = (B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B)$ [n is the state-dimension]
- System is controllable if R has full row rank. (i.e. rows are linearly independent)
- Example:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Checking Controllability

- $A = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

- $R = \begin{bmatrix} 1 & 1 & 1 & -1 & 7 & 5 \\ 0 & 1 & 2 & 3 & 4 & 1 \\ 1 & 0 & 4 & 2 & 15 & 8 \end{bmatrix}$

  B AB A^2B

- rank(R) = 3 (i.e. full rank)
- So system is controllable: uses 2 actuators (u_1, u_2)!

Checking Controllability

- $A = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

Tip: Given matrices A, B use Matlab command $\mathrm{R} = \mathrm{ctrb}(\mathrm{A}, \mathrm{B})$ to find controllability Gramian.

Tip: Use $rank(R)$ to find rank of R

- $R = \begin{bmatrix} 1 & 0 & -1 & 0 & 3 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 4 & 0 \end{bmatrix}$

$B \qquad AB \qquad A^2B$

- rank(R) = 3 (i.e. full rank)
- So system is controllable: but uses only 1 actuator (u_1)!

Observability

- Very rarely are all system states $\mathbf{x}$ visible to the external world
  - E.g. model may have internal physical states such as temperature, pressure, object velocity: that may not be measurable by an external observer
  - Only things made available by a sensor are visible to the real world
- Observability:
  - Can we reconstruct an arbitrary internal state of the system if we have only the system outputs available?
  - Important question that affects which sensors we pick for the system

Checking Observability

- Find *observability matrix* W
- $W = (C \quad CA \quad CA^2 \quad \dots \quad CA^{n-1})^T$ [n is the state-dimension]
- System is observable if W has full row rank. (i.e. rows are linearly independent)
- Example:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}; \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Checking Observability

- $A = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix}$ $C = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$
- $W = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 3 & 2 & 3 \\ 1 & 1 & 2 \\ 4 & 5 & 15 \\ 3 & 3 & 8 \end{bmatrix}$, $rank(R) = 3$

- Matrix W is full rank
- $\Rightarrow$ Pair (A, C) is observable
- Assuming sensors measure x_1, x_2, x_3 independently, we need three sensors
- Assuming we have one sensor that measures $x_2 + x_3$, another measures $x_1 + x_2$, we use two sensors

Checking Observability

- $A = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 1 & 3 \end{bmatrix}$ $C = [1\ 1\ 1]$
- $W = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 7 & 7 & 19 \end{bmatrix}$, $rank(R) = 2$

- What if we used only one sensor that measures sum of all states?
- I.e. $y = x_1 + x_2 + x_3$?
- Observability matrix is not full rank! Cannot reconstruct some state using only one sensor!
- Tip: use matlab command $\mathrm{obsv(A, C)}$ to find W

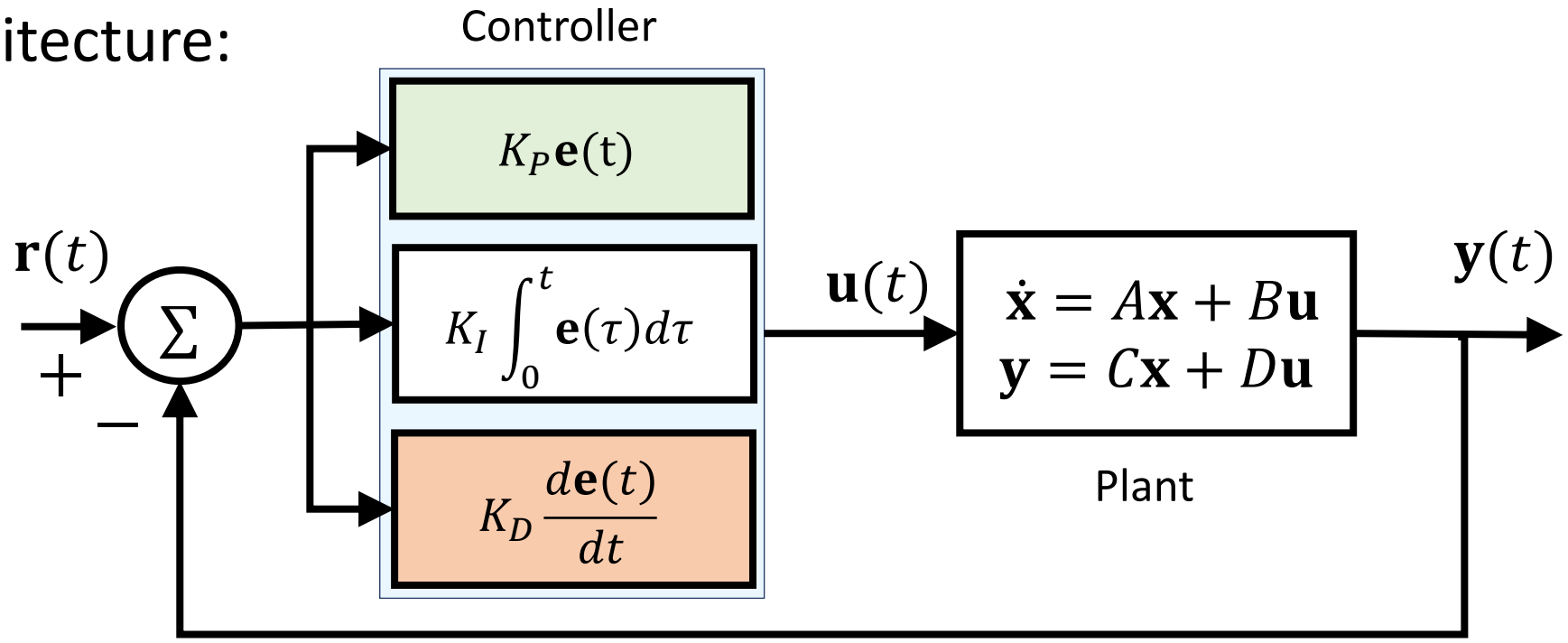
How do we reconstruct internal state?

- For linear systems (with no noise), this is done with the use of state estimators or observers
- For linear systems with noisy measurements and possible “process noise” in the system itself : we use Kalman filter

- The most popular control method in the world started without any concerns of controllability, observability etc.
- Purpose: Tracking a given reference signal

PID controllers

- While previous controllers used systematic use of linear systems theory, PID controllers are the most widely-used and most prevalent in practice (> 90%)
- Main architecture:



P-only controller

- Compute error signal $\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{y}(t)$
- Proportional term $K_p\mathbf{e}(t)$:
  - K_p proportional gain;
  - Feedback correction proportional to error
- Cons:
  - If K_p is small, error can be large! [undercompensation]
  - If K_p is large,
    - system may oscillate (i.e. unstable) [overcompensation]
    - may not converge to set-point fast enough
  - P-controller always has steady state error or offset error

PD-controller

- Compute error signal $\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{y}(t)$
- Derivative term $K_d \dot{\mathbf{e}}(t)$:
  - K_d derivative gain;
  - Feedback proportional to how fast the error is increasing/decreasing
- Purpose:
  - "Predictive" term, can reduce overshoot: if error is decreasing slowly, feedback is slower
  - Can improve tolerance to disturbances
- Disadvantages:
  - Still cannot eliminate steady-state error
  - High frequency disturbances can get amplified

PI/PID controller

- Integral term: $K_I \int_0^t \mathbf{e}(\tau) d\tau$
  - K_I integral gain;
  - Feedback action proportional to cumulative error over time
  - If a small error persists, it will add up over time and push the system towards eliminating this error): eliminates offset/steady-state error
- Disadvantages:
  - Integral action by itself can increase instability
    - (adding a "D" term can help)
  - Integrator term can accumulate error and suggest corrections that are not feasible for the actuators (integrator windup)
    - Real systems "saturate" the integrator beyond a certain value

PID controller in practice

- Many heuristics to *tune* PID controllers, i.e., find values of K_P, K_I, K_D
- Several *recipes* to tune, usually rely on designer expertise
- E.g. *Ziegler-Nichols* method: increase K_P till system starts oscillating with period T (say till $K_P = K^*$), then set $K_P = 0.6K^*, K_I = \frac{1.2K^*}{T}, K_D = \frac{3}{40}K^*T$
- Matlab/Simulink has PID controller blocks + PID auto-tuning capabilities
- Work well with linear systems or for small perturbations,
- For non-linear systems use "gain-scheduling"
  - (i.e. using different K_P, K_I, K_D gains in different operating regimes)

Measuring control performance

- Typical to excite closed-loop system with a “step input”
  - I.e. sudden change in reference set-point
- “Step” input of (say) 2.5 at time 0
- Step Response in blue
  - Peak/Overshoot (corr. undershoot)
  - Settling Time/Settling Region
  - Rise Time
  - Peak Time
  - Steady State Error

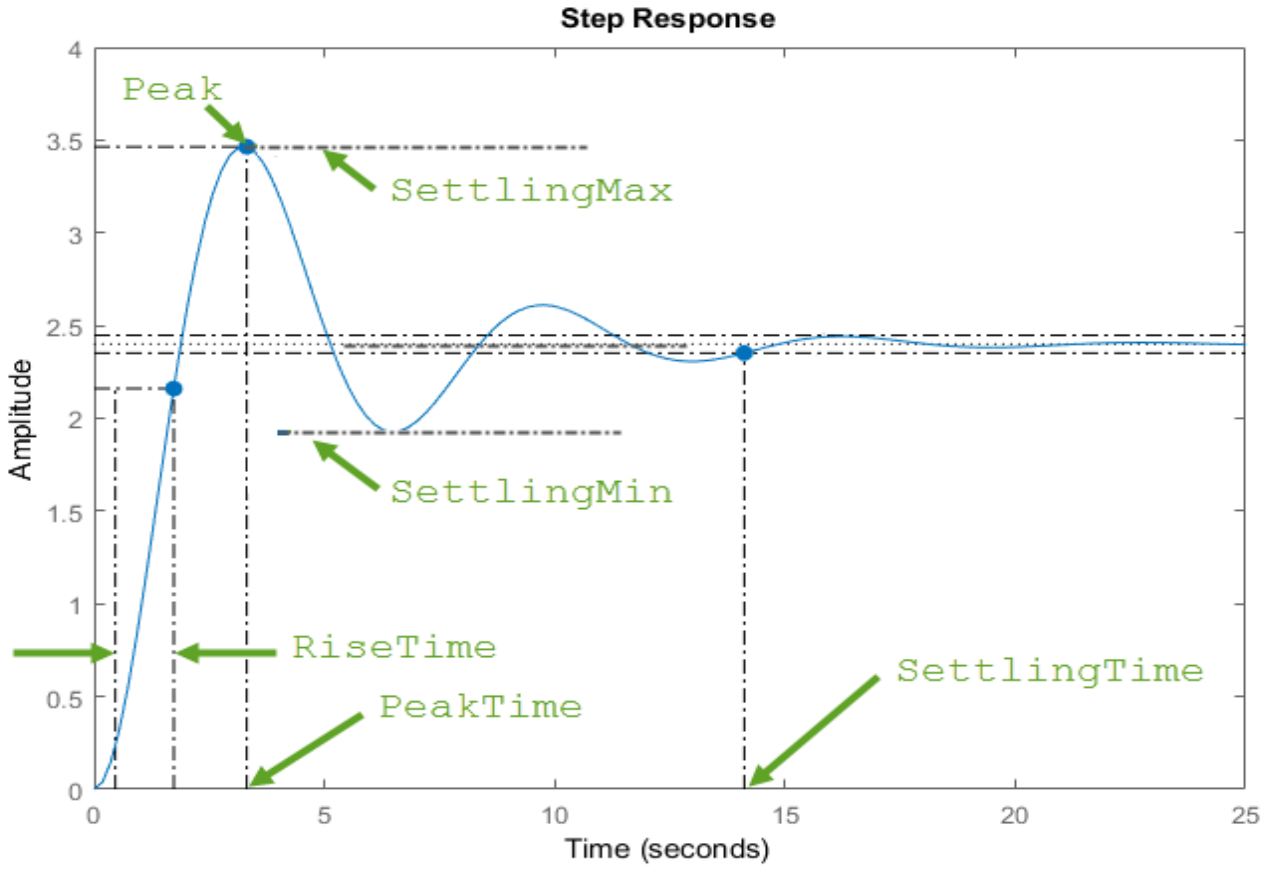


Image © from Mathworks

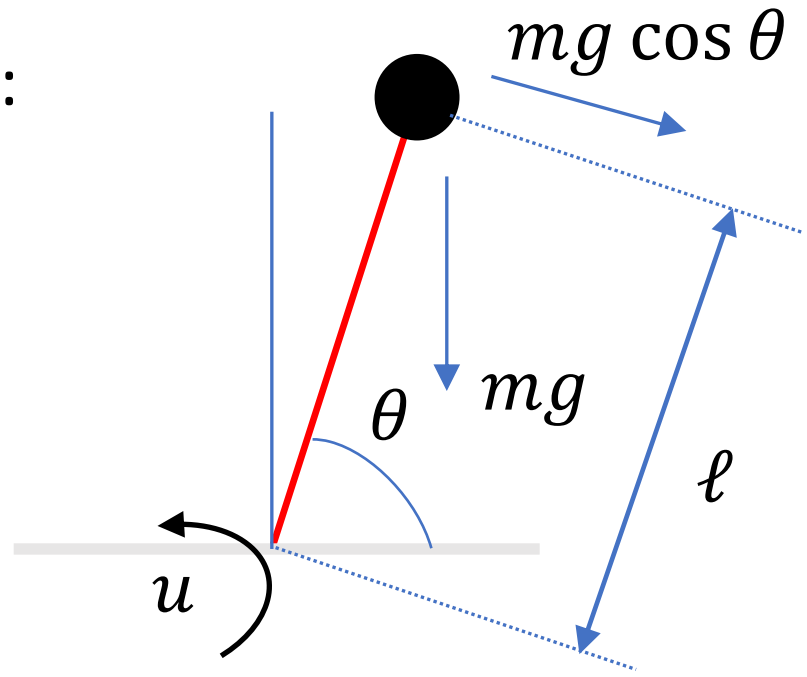
Nonlinear Control

Feedback Linearization

- Main idea: Try to choose control such the nonlinear system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ becomes linear
- Equations of motion for inverted pendulum:

$$m\ell^2\ddot{\theta} + d\dot{\theta} + m\ell g\cos\theta = u$$

- Control Input: Torque u
- Rewriting, with $x_1 = \theta, x_2 = \dot{\theta}$:
- $\dot{x_1} = x_2$
- $\dot{x_2} = \left(-\frac{d}{m\ell^2}x_2 - \frac{g}{\ell}\cos x_1\right) + \left(\frac{1}{ml^2}u\right)$



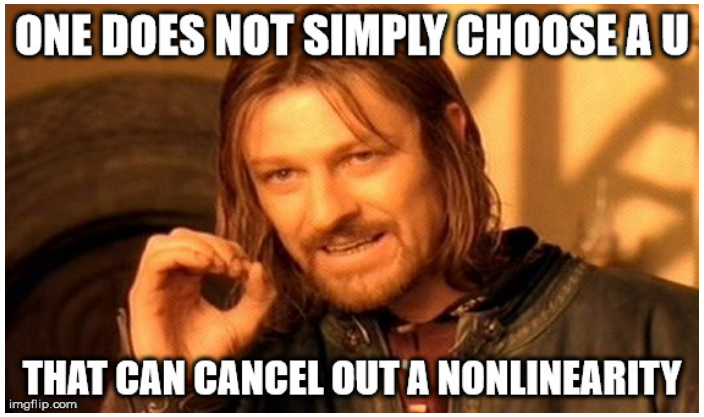
Feedback linearization continued

- To make our life easier, let $ml^2 = d = \frac{1}{b}$, and let $\ell = g$, then we get:
  - $\dot{x_1} = x_2$
  - $\dot{x_2} = (-x_2 \ - \cos x_1) + bu$
- Let's define a new control input v such that, $u = \frac{1}{b}(v + x_2 + \cos x_1)$
- Voila!
  - $\dot{x_1} = x_2$
  - $\dot{x_2} = v$
- This is a linear system, with $A = [0\ 1; 0\ 0], \mathrm{B} = [0; 1]$ which we can stabilize by finding K such that $(A - BK)$ has eigenvalues with negative real parts.

Input Transformation

- This operation is called input transformation, which leads to exact cancellation of a nonlinearity, giving rise to a linear equation
- Also known as exact feedback linearization or dynamic inversion
- Note that this is NOT the same as computing the Jacobian of the nonlinear system and trying to stabilize the resulting linear system at the origin (this would make the system stable only locally)
- We are using feedback ***to linearize*** the system
- Unfortunately, we cannot always do this

State Transformation

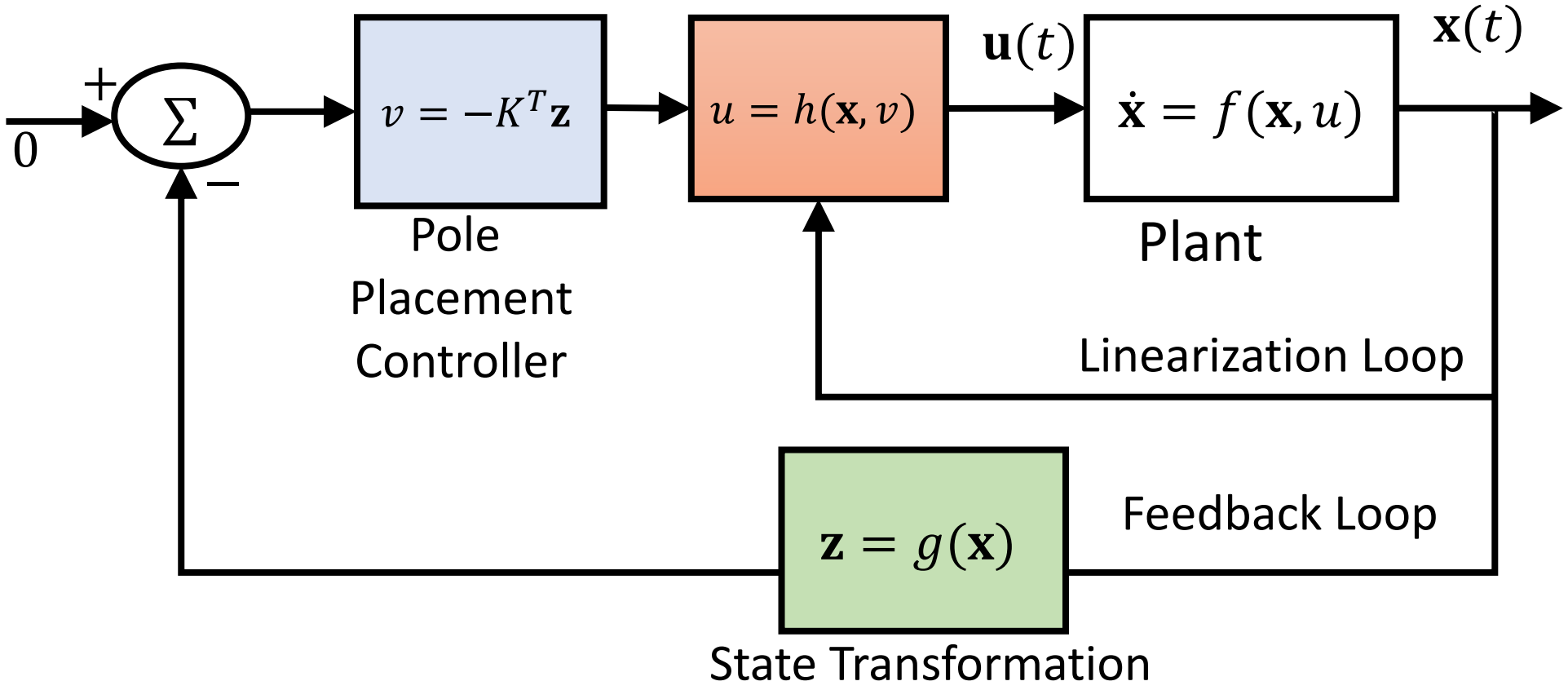


- Consider system:
  - $\dot{x_1} = a \ \sin x_2$
  - $\dot{x_2} = -x_1^2 + u$
- How do we cancel out $\sin x_2$?
- We can first change variables by a nonlinear transformation:
  - $z_1 = x_1, z_2 = a \sin x_2$
- Now, $\dot{z_1} = z_2$, and
  - $\dot{z_2} = \dot{x_2}\ a \cos x_2 = a(-x_1^2 + u) \cos x_2 = a(-z_1^2 + u) \cos \sin^{-1} \frac{z_2}{a}$

State transformation continued

- Equations rewritten:
  - $\dot{z_1} = z_2$
  - $\dot{z_2} = a(-z_1^2 + u)\cos\sin^{-1}\frac{z_2}{a}$
- Now we can pick $u = z_1^2 + \frac{1}{a\cos\sin^{-1}\frac{z_2}{a}}v$
- Rewriting in terms of x's:
  - $u = x_1^2 + \frac{1}{a\cos x_2}v$
- This gives us a linear system $\dot{z_1} = z_2; \dot{z_2} = v$, which we can again stabilize using linear system methods

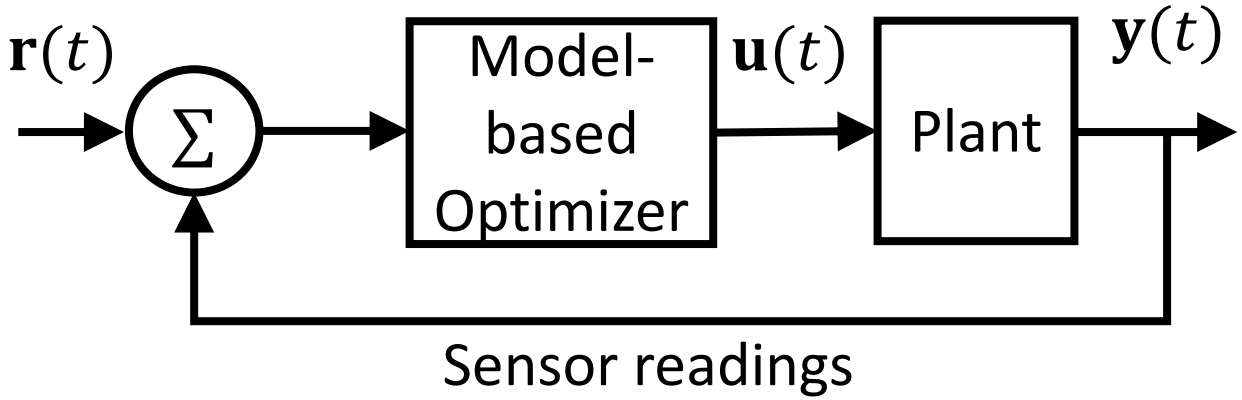
Form of the controller: two "loops"



Input Transformation

Model Predictive Control

- Main idea: Use a dynamical model of the plant (inside the controller) to predict the plant's future evolution, and optimize the control signal over possible futures



Receding Horizon Philosophy

- Create difference equation: $\mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k]); \mathbf{y}[k] = g(\mathbf{x}[k])$
- At time t, solve an optimal control problem over next N steps:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{k=0}^{N-1} \|\mathbf{y}[t+k] - r[t]\|^2 + \rho\|\mathbf{u}[\mathrm{t}+\mathrm{k}]\|^2$$

$$s.t.\ \mathbf{x}[t+k+1] = f(\mathbf{x}[t+k], \mathbf{u}[t+k])$$
$$\mathbf{y}[t+k] = \mathbf{g}(\mathbf{x}[t+k])$$

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}, \mathbf{y}_{\min} \leq \mathbf{y} \leq y_{\max}$$

- Only apply optimal control input value $\mathbf{u}^*$ at time t
- At time $t+1$: get new measurements, repeat optimization

Observer design

What is state estimation and why is it needed?

- Given a “black box” component, we can try to use a linear or nonlinear system to model it (maybe based on physics, or data-driven)
- Model may posit that the plant has n internal states, but we typically have access only to the outputs of the model (whatever we can measure using a sensor)
- May need internal states to implement controller: how do we estimate them?
- State estimation: Problem of determining internal states of the plant

Deterministic vs. Noisy case

- Typically sensor measurements are noisy (manufacturing imperfections, environment uncertainty, errors introduced in signal processing, etc.)
- In the absence of noise, the model is deterministic: for the same input you always get the same output
  - Can use a simpler form of state estimator called an observer (e.g. a Luenberger observer)
- In the presence of noise, we use a state estimator, such as a Kalman Filter
- Kalman Filter is one of the most fundamental algorithm that you will see in autonomous systems, robotics, computer graphics, …

Random variables and statistics refresher

- For random variable w, $\mathbb{E}[w]$: expected value of w, also known as mean
- Suppose $\mathbb{E}[x] = \mu$: then $\mathrm{var}(\mathrm{w})$: variance of w, is $\mathbb{E}[(w-\mu)^2]$
- For random variables x and y, $\mathrm{cov}(x, y)$: covariance of x and y
  - $\mathrm{cov}(x, y) = \mathbb{E}[(x - \mathbb{E}(x)(y - \mathbb{E}(y)]$
- For random ***vector*** $\mathbf{x}$, $\mathbb{E}[\mathbf{x}]$is a vector
- For random vectors, $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$, cross-covariance matrix is $m \times n$ matrix: $\mathrm{cov}(\mathbf{x}, \mathbf{y}) = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{y} - \mathbb{E}[\mathbf{y}])^{\mathrm{T}}]$
- $w \sim N(\mu, \sigma^2)$: w is a normally distributed variable with mean μ and variance σ

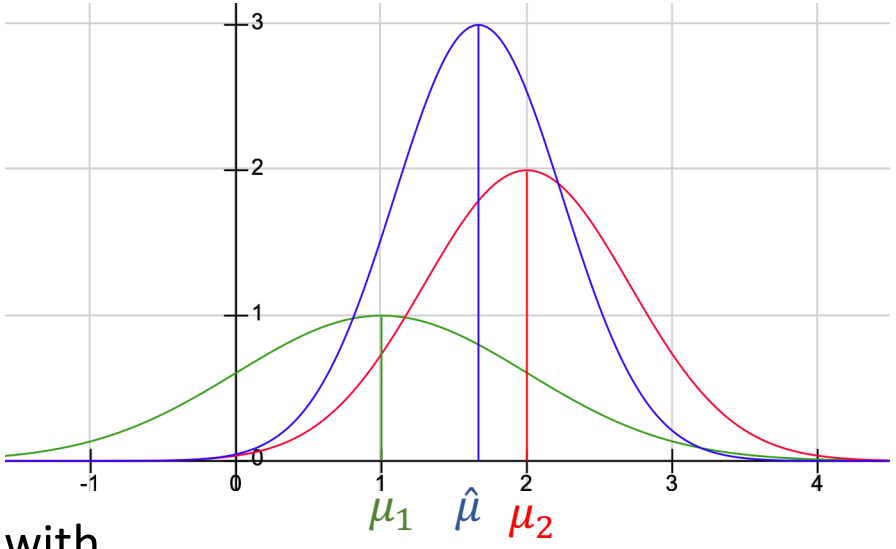
Data fusion example

- Using radar and a camera to estimate the distance to the lead car:
  - Measurement is never free of noise
  - Actual distance: x
  - Measurement with radar: $z_1 = x + v_1$ ($v_1 \sim N(\mu_1, \sigma_1^2)$ is radar noise)
  - With camera: $z_2 = x + v_2$ ($v_2 \sim N(\mu_2, \sigma_2^2)$ is camera noise)
  - How do you combine the two estimates?
- Use a weighted average of the two estimates, prioritize more likely measurement
  - $\hat{x} = \frac{(z_1/\sigma_1^2) + (z_2/\sigma_2^2)}{(1/\sigma_1^2)+(1/\sigma_2^2)} = kz_1 + (1-k)z_2$, where $k = \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}$
  - $\hat{\mu} = \hat{x}, \hat{\sigma}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$
- Observe: uncertainty reduced, and mean is closer to measurement with lower uncertainty

$\mu_1 = 1, \sigma_1^2 = 1$

$\mu_2 = 2, \sigma_2^2 = 0.5$

$\hat{\mu} = 1.67, \sigma_2^2 = 0.33$



Multi-variate sensor fusion

- Instead of estimating one quantity, we want to estimate n quantities, then:
- Actual value is some vector $\mathbf{x}$
- Measurement noise for i^{th} sensor is $v_i \sim N(\boldsymbol{\mu}_i, \Sigma_i)$, where $\boldsymbol{\mu}_i$ is the mean vector, and Σ_i is the covariance matrix
- $\Lambda = \Sigma^{-1}$ is the information matrix
- For the two-sensor case:
  - $\hat{\mathbf{x}} = (\Lambda_1 + \Lambda_2)^{-1}(\Lambda_1 \mathbf{z}_1 + \Lambda_2 \mathbf{z}_2)$, and $\hat{\Sigma} = (\Lambda_1 + \Lambda_2)^{-1}$

Motion makes things interesting

- What if we have one sensor and making repeated measurements of a moving object?
- Measurement differences are not all because of sensor noise, some of it is because of object motion
- Kalman filter is a tool that can include a motion model (or in general a dynamical model) to account for changes in internal state of the system
- Combines idea of ***prediction*** using the system dynamics with ***correction*** using weighted average (Bayesian inference)

Stochastic Difference Equation Models

- We assume that the plant (whose state we are trying to estimate) is a stochastic discrete dynamical process with the following dynamics:

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k \text{ (Process Model)}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \text{ (Measurement Model)}$$

$\mathbf{x}_k, \mathbf{x}_{k-1}$	State at time k,$k-1$
$\mathbf{u}_k$	Input at time k
$\mathbf{w}_k$	Random vector representing noise in the plant, $\mathbf{w} \sim N(\mathbf{0}, Q_k)$
$\mathbf{v}_k$	Random vector representing sensor noise, $\mathbf{v} \sim N(\mathbf{0}, R_k)$
$\mathbf{z}_k$	Output at time k

n	Number of states
m	Number of inputs
p	Number of outputs
A	$n \times n$ matrix
B	$n \times m$ matrix
H	$p \times n$ matrix

Step I: Prediction

- We assume an estimate of $\mathbf{x}$ at time $k-1$, fusing information obtained by measurements till time $k-1$: this is denoted $\hat{\mathbf{x}}_{k-1|k-1}$
- We also assume that the error between the estimate $\hat{\mathbf{x}}_{k-1|k-1}$ and the actual $\mathbf{x}_{k-1}$ has 0 mean, and covariance $P_{k-1|k-1}$
- Now, we use these values and the state dynamics to predict the value of $\mathbf{x}_k$
- Because we are still using measurements only up to time $k-1$, we can denote this predicted value as $\hat{\mathbf{x}}_{k|k-1}$, and compute it as follows:

$$\hat{\mathbf{x}}_{k|k-1} \coloneqq A\hat{\mathbf{x}}_{k-1|k-1} + B\mathbf{u}_k$$

Step I: Prediction continued

- We also need to update the predicted covariance between the predicted value and the "actual" value of $\mathbf{x}_k$

$$\begin{aligned} P_{k|k-1} &= \mathrm{cov}\left(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}\right) = \mathrm{cov}\left(A\mathbf{x}_{k-1} + w_k - A\hat{\mathbf{x}}_{k-1|k-1}\right) \\ &= A\mathrm{cov}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k|k-1}\right)A^T + cov(w_k) \\ &= AP_{k-1|k-1}A^T + Q_k \end{aligned}$$

- Thus, the *a priori* estimated state and error covariance are:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &:= A\hat{\mathbf{x}}_{k-1|k-1} + B\mathbf{u}_k \\ P_{k|k-1} &:= AP_{k-1|k-1}A^T + Q_k \end{aligned}$$

Step II: Correction

- This is where we basically do data fusion between new measurement and old prediction to obtain new estimate
- Note that data fusion is not straightforward like before because we don't really observe/measure $\mathbf{x}_k$ directly, but we get measurement $\mathbf{z}_k$, for an observable output!
- Idea remains similar: Do a weighted average of the prediction $\hat{\mathbf{x}}_{k|k-1}$ and new information
- We integrate new information by using the difference between the predicted output and the observation

Step II: Correction continued

- Predicted output: $H_k\hat{\mathbf{x}}_{k|k-1}$, and error in predicted output = $\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k|k-1}$
- We denote this expression as the *innovation* $\mathbf{y}_k := \mathbf{z}_k - H_k\hat{\mathbf{x}}_{k|k-1}$
- Covariance of innovation (S_k) can be shown to be $R_k + H_k P_{k|k-1} H_k^T$
- Then to do data fusion, we use a matrix known as (optimal) Kalman gain K_k

$$\hat{\boldsymbol{x}}_{k|k} := \hat{\boldsymbol{x}}_{k|k-1} + K_k y_k$$

- Where, K_k is given by $P_{k|k-1} H_k^T S_k^{-1}$
- Finally, the updated error covariance estimate is given by:

$$P_{k|k} := P_{k|k-1}(I - K_k H_k)$$

Correction step summary

Innovation	$\mathbf{y}_k := \mathbf{z}_k - H_k \hat{\mathbf{x}}_{k\|k-1}$
Innovation Covariance	$S_k := R_k + H_k P_{k\|k-1} H_k^T$
Optimal Kalman Gain	$K_k := P_{k\|k-1} H_k^T S_k^{-1}$
A posteriori state estimate	$\hat{\boldsymbol{x}}_{k\|k} := \hat{\boldsymbol{x}}_{k\|k-1} + K_k \mathbf{y}_k$
A posteriori error covariance estimate	$P_{k\|k} := P_{k\|k-1}(I - K_k H_k)$

What are all these equations? How to make sense?

- Let's take a simple one-dimensional example, with perfect observability (i.e. $z = x$). So, at each step, x_k is the measurement.
- Then, Kalman filter prediction equations become:
  - $\hat{x}_{k|k-1} \coloneqq a\hat{x}_{k-1|k-1} + bu$; $\sigma^2_{k|k-1} \coloneqq \underbrace{a^2\sigma^2_{k-1|k-1}}_{\text{prior uncertainty in estimate}} + \underbrace{\sigma^2_q}_{\text{uncertainty in process}}$
- Also, the correction equations become:
  - Innovation: $y_k \coloneqq x_k - \hat{x}_{k|k-1}$, Innovation variance = $\sigma_r^2 + \sigma^2_{k|k-1}$
  - Optimal gain: $k = 1/(\sigma_r^2 + \sigma^2_{k|k-1})$,
  - Updated state estimate: $\hat{x}_{k|k} \coloneqq \hat{x}_{k|k-1} + k(x_k - \hat{x}_{k|k-1})$
  - I.e. updated state estimate: $\hat{x}_{k|k} \coloneqq (1-k)\ \hat{x}_{k|k-1} + kx_k$ (Weighted average!)

Extended Kalman Filter

- We skipped derivations of equations of the Kalman filter, but a fundamental property assumed is that the process model and measurement model are both linear.
- Under linear models and Gaussian process/measurement noise, a Kalman filter is an *optimal* state estimator (minimizes mean square error between estimate and actual state)
- In an EKF, state transitions and observations need not be linear functions of the state, but can be any differentiable functions
- I.e., the process and measurement models are as follows:

$$\mathbf{x}_k = f(x_{k-1}, u_k) + w_k$$
$$z_k = h(x_k) + v_k$$

EKF updates

- Functions f and h can be used directly to compute state-prediction, and predicted measurement, but cannot be directly used to update covariances
- So, we instead use the Jacobian of the dynamics at the predicted state
- This linearizes the non-linear dynamics around the current estimate
- Prediction updates:

$$\hat{\mathbf{x}}_{k|k-1} := f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$$
$$P_{k|k-1} := F_k P_{k-1|k-1} F_k^T + Q_k$$

$$F_k := \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}=\mathbf{u}_k}$$

EKF updates continued

- Correction updates

$$H_k := \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}$$

Innovation	$\mathbf{y}_k := \mathbf{z}_k - h(\hat{\mathbf{x}}_{k\|k-1})$
Innovation Covariance	$S_k := R_k + H_k P_{k\|k-1} H_k^T$
Near-Optimal Kalman Gain	$K_k := P_{k\|k-1} H_k^T S_k^{-1}$
A posteriori state estimate	$\hat{\boldsymbol{x}}_{k\|k} := \hat{\boldsymbol{x}}_{k\|k-1} + K_k \mathbf{y}_k$
A posteriori error covariance estimate	$P_{k\|k} := P_{k\|k-1}(I - K_k H_k)$

Summary

- Linear Control (State-Space View)
  - Reference Tracking
  - Pole Placement controller
  - LQR
  - PID
- Nonlinear Control
  - Feedback linearization
  - Model-Predictive Control
- Kalman filters: observers for dynamical systems