Fall 2021 ME424 Modern Control and Estimation

Lecture Note 6
Control Design and Testing in Drake with Python

Prof. Wei Zhang
Department of Mechanical and Energy Engineering
SUSTech Institute of Robotics
Southern University of Science and Technology

zhangw3@sustech.edu.cn
https://www.wzhanglab.site/

Outline

- **Short introduction to Drake**
- Example 1: Observer and Controller Design
- Example 2: Cart-Pole Balancing
- From regulation to tracking control
- Example 3: DC Motor speed tracking control

- **What is Simulation?**
  - Real-world physics are often described by functions, ODE or PDE

static systems

dynamic

  - All simulators essentially solve the ODEs and/or PDEs corresponding to a physical process of interest

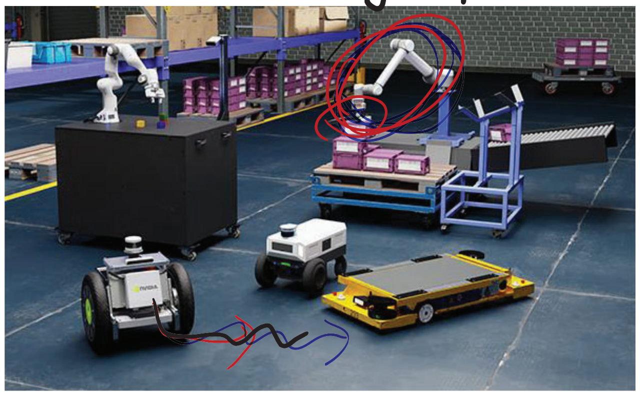
- **Three pillars of a simulator:**

urdf
xml

functions
static
- sensor
- environment
- kinematic $y = FK(\theta)$

1. Constructing the differential equations/models

- Dynamic model : ODE / PDE
- static model : IMU sensor $y_{meas} = \begin{bmatrix} a_x \\ a_y \\ a_z \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = f(x) + v$



2. Solving differential equations

- solver

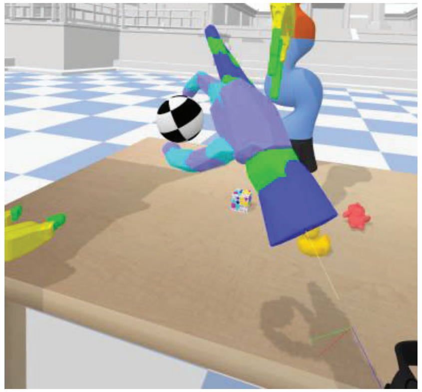
physics engine $\begin{cases} \text{accurate} \\ \text{fast} \end{cases}$

3. Visualization of the simulation results

▪ Popular simulators in robotics

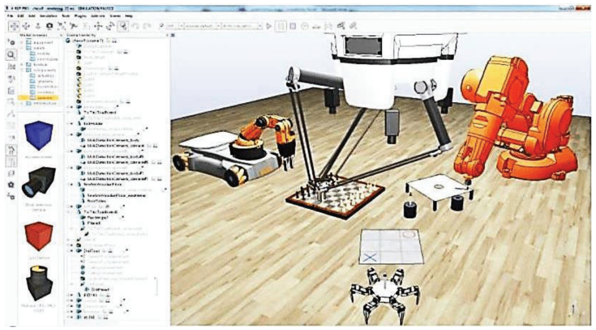
Mujoco (Roboti LLC)

deepmind openAI

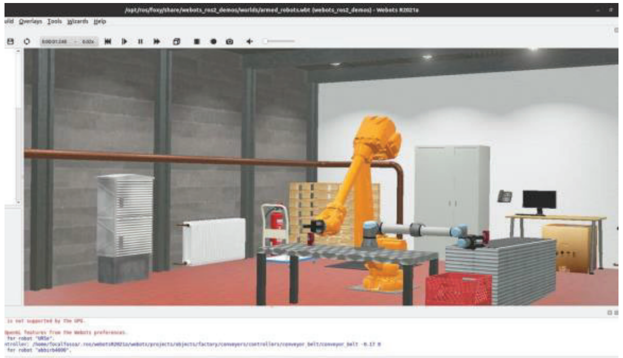


PyBullet (open source)

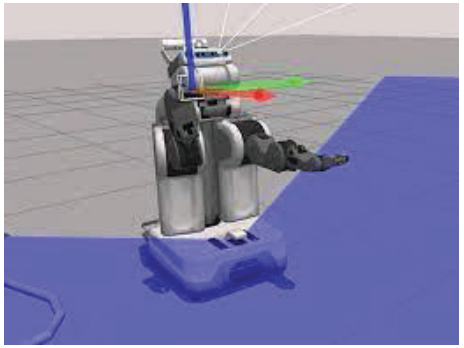
ISAAC (NVIDIA)



V-REP (CoppeliaSim)



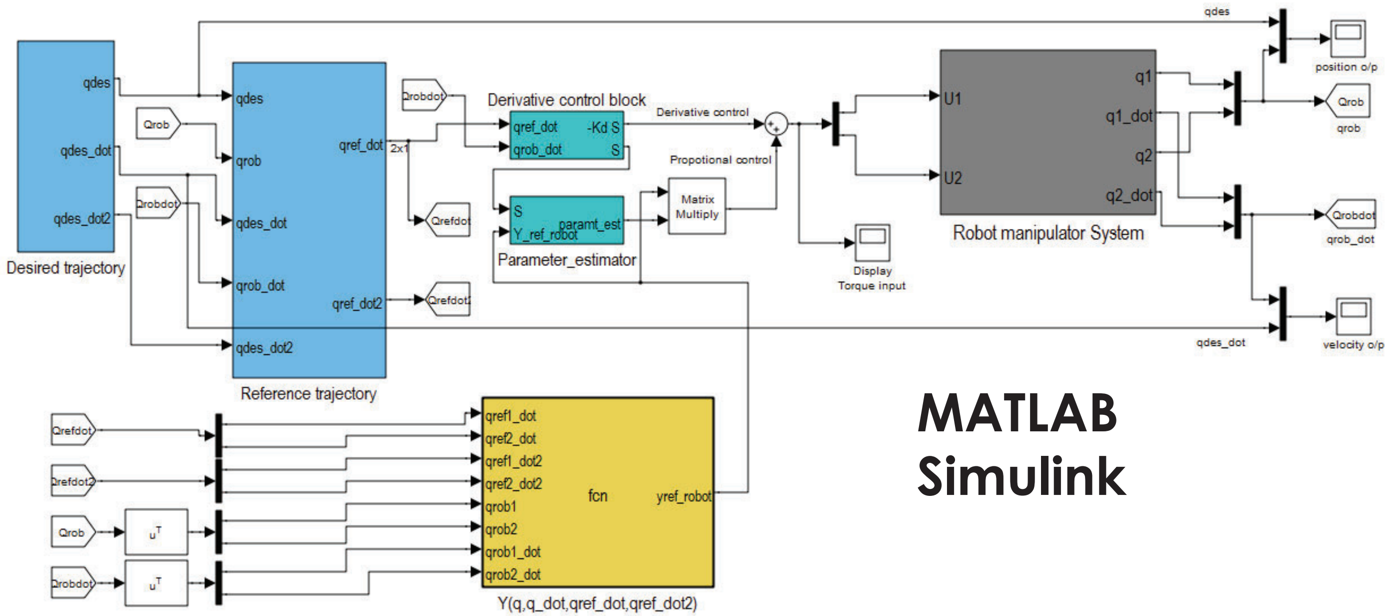
Webot (Cyberbotics)



Gazebo

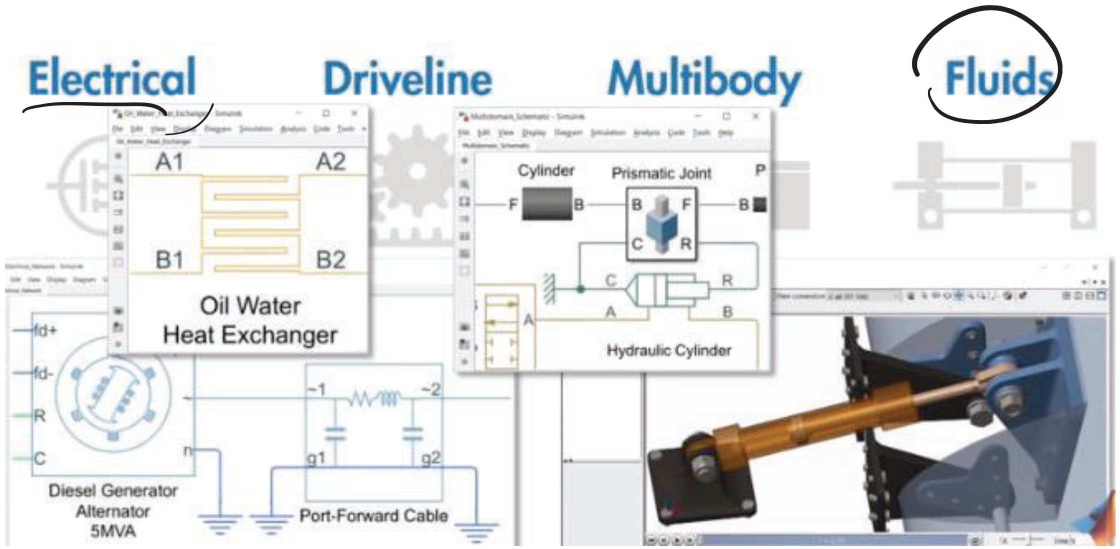
bullet

▪ Popular simulators for control systems



MATLAB Simulink

MATLAB Simscape



▪ **Drake:** Model-Based Design and Verification for Robotics

DRAKE

A PLANNING, CONTROL, AND ANALYSIS TOOLBOX FOR NONLINEAR DYNAMICAL SYSTEMS

- Happy marriage between MATLAB Simulink with and robotic simulators

Drake adopts block diagram concept

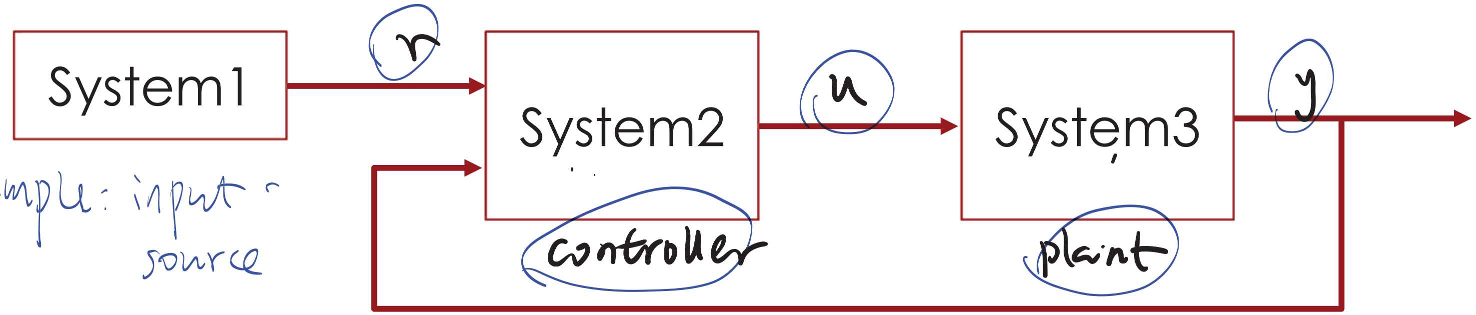
- Great support for dynamic system modeling, optimization, robotic kinematics and dynamics

- Very accurate simulation (contact handling)

- Visualization is not great but good enough

- Open-source and support Python

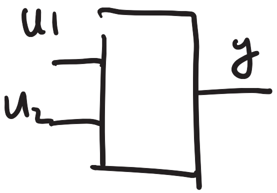
Drake: Block Diagram Overview



Example: input source

e.g. { step, sin

- How to define a system block (static or dynamic)?



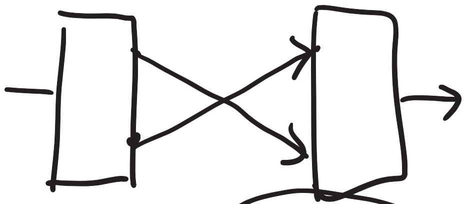
static: what info needed?

- # of input ports
- # of output ports
- relation : $y = f(u_1, u_2)$

Dynamic

- ✓
- ✓
- continuous time: $\dot{x} = f(x, u)$ ← vector field
- Discrete time: $x_{k+1} = f(x_k, u_k)$ ← update equation

- How to connect blocks to make the overall diagram?



- which output port connects to which input port

- How to simulate? - we need to know
  - Block diagram (systems ⊕ connection)
  - parameters, initial states — context

Drake: How to Define a Static System?

```
class StaticSysExample(LeafSystem):
    def __init__(self, myParameter):
        LeafSystem.__init__(self)
        self.DeclareVectorInputPort("u1", BasicVector(num_input1))
        self.DeclareVectorInputport("u2", BasicVector(num_input2))
        self.DeclareVectorOutputPort("y", BasicVector(num_output), self.CalcOutputY)

    def CalcOutputY(self, context, output):
        u1 = self.get_input_port(0).Eval(context)
        u2 = self.get_input_port(1).Eval(context)
        y = "your output function"
        output.SetFromVector(y)
```

BasicVector:

- a = BasicVector(3)   # 3-d vector
- a.SetFromVector([arrray])   # from numpy array to Basic vector
- a.CopyToVector()   # from BasicVector to numpy array

▪ Drake: How to Define a Continuous-Time Dynamic System?

$$\begin{cases} \dot{x} = f(x,u) \\ y = h(x,u) \end{cases}$$

```python
class CTSysExample(LeafSystem):
    def __init__(self, myParameter):
        LeafSystem.__init__(self)
        self.DeclareContinuousState(num_state)
        self.DeclareVectorInputPort("u", BasicVector(num_input))
        self.DeclareVectorOutputPort("y", BasicVector(num_output), self.CalcOutputY)

    def DoCalcTimeDerivatives(self, context, derivatives):
        x = context.get_continuous_state_vector().CopyToVector()
        u = self.get_input_port(0).Eval(context)
        xdot = "your vector field"
        derivatives.get_mutable_vector().SetFromVector(xdot)

    def CalcOutputY(self, context, output):
        x = context.get_continuous_state_vector().CopyToVector()
        y = "your output function"
        output.SetFromVector(y)
```

4

$\dot{x}$

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

$Ax+Bu$, $\sin(x[0]) + \cos(u[0])$

$y = h(x,u)$

$h(x,u)$

Drake: How to Define a Discrete-Time Dynamic System?

$x_{k+1} = f(x_k, u_k)$

$y_k = h(x_k, u_k)$

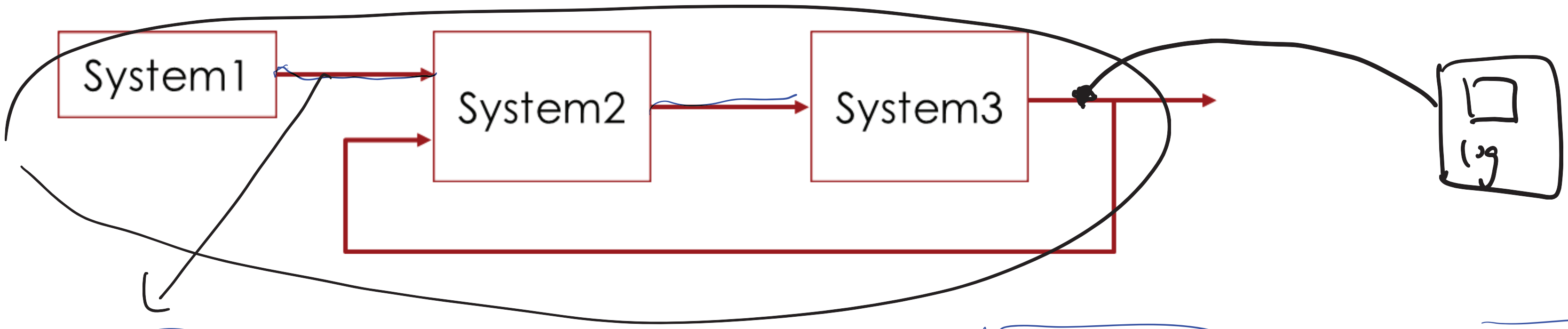
```
class DTLinearSys(LeafSystem):
    def __init__(self):
        LeafSystem.__init__(self)
        self.DeclareDiscreteState(num_state)
        self.DeclareVectorInputPort("u", BasicVector(num_input))
        self.DeclareVectorOutputPort("y", BasicVector(num_output), self.CalcOutputY)
        self.DeclarePeriodicDiscreteUpdate(0.5)  # dt
```

Sampling time

```
    def DoCalcDiscreteVariableUpdates(self, context, events, discrete_state):
        x = context.get_discrete_state_vector().CopyToVector()
        u = self.get_input_port(0).Eval(context)
        xnext = 0.8*x + np.sin(u)
        discrete_state.get_mutable_vector().SetFromVector(xnext)

    def CalcOutputY(self, context, output):
        x = context.get_discrete_state_vector().CopyToVector()
        u = self.get_input_port(0).Eval(context)
        y = x + 2*u
        output.SetFromVector(y)
```

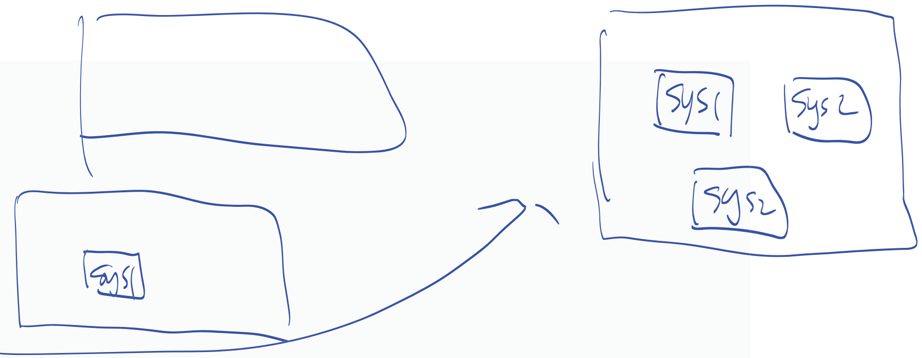
▪ Drake: Block Diagram Construction



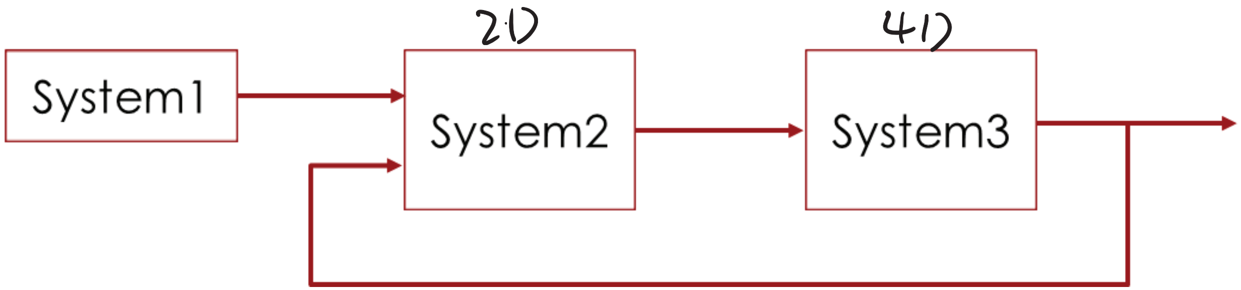

```
builder = DiagramBuilder()
Sys1 = builder.AddSystem(Sys1)
Sys2 = builder.AddSystem(Sys2)
Sys3 = builder.AddSystem(Sys3)

builder.Connect(Sys1.get_output_port(0), Sys2.get_input_port(0))
builder.Connect(Sys2.get_output_port(0), Sys3.get_input_port(0))
builder.Connect(Sys3.get_output_port(0), Sys2.get_input_port(1))

logger_output = LogOutput(Sys3.get_output_port(0), builder)
diagram = builder.Build()
```



Drake: Simulate a Block Diagram




```
simulator = Simulator(diagram)
simulator.set_target_realtime_rate(1)
context = simulator.get_mutable_context()
context.SetContinuousState(CT_x0)
context.SetDiscreteState(DT_x0)
simulator.AdvanceTo(sim_time)
```

initial state

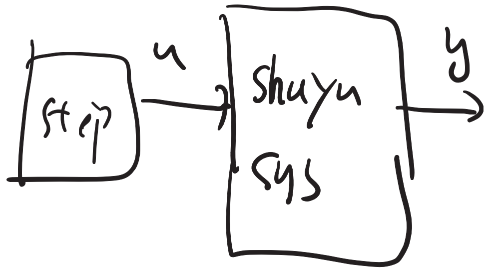
Regarding **context** class:

- get_continuous_state_vector()
- get_discrete_state_vector()
- get_mutable_continuous_state_vector()
- get_mutable_discrete_state_vector()
- SetContinuousState(..)
- SetDiscreteState(..)

▪ Drake: Simple Simulation Examples

- DT system:

$$\begin{cases} x_{k+1} = 0.2 x_k + 5 u_k \\ y_k = \begin{bmatrix} \sin x_k \\ u_k \end{bmatrix} \end{cases}$$



Outline

- Short introduction to Drake
- **Example 1: Observer and Controller Design**
- Example 2: Cart-Pole Balancing
- From regulation to tracking control
- Example 3: DC Motor speed tracking control

Example 1: design output feedback controller for plant

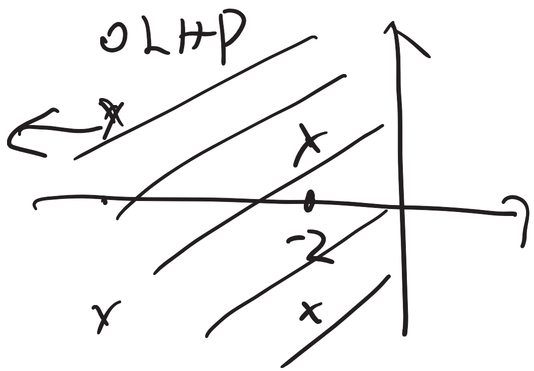
$$A_c = \begin{bmatrix} 33 & -60 \\ 20 & -33.2 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad C_c = [2 \quad 1]$$

- **Step 1:** Discretization: e.g. with sampling time T = 0.01

$$A_d = (I + A_c T) = \begin{bmatrix} 1.33 & -0.6 \\ 0.2 & 0.668 \end{bmatrix}, \qquad B_d = B_c T = \begin{bmatrix} 0.01 \\ 0.01 \end{bmatrix}, \qquad C_d = C_c = [2 \quad 1]$$

- **Step 2:** Select desired closed-loop eigs (suppose we want the continuous time poles: $s_{1,2}$ ={-2+j, -2-j}

$z_{1,2} = e^{s_{1,2} \cdot T}$



- **Step 3:** Design feedback gain K

place

- **Step 4:** Observer eigs: suppose we want: observer_$s_{1,2}$ ={-8+j, -8+j}

Find L such that $\mathrm{eig}(A - LC) = \mathrm{eig}_{desired} = e^{obs\_eig \cdot T}$

$L = (\mathrm{place}(A^T, C^T))^T$

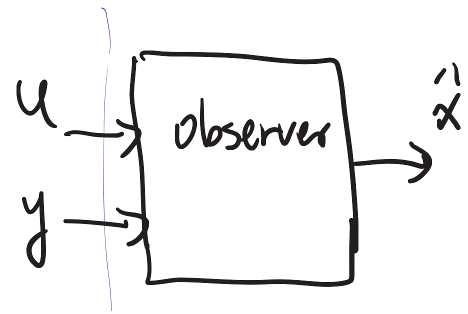
$A \in \mathbb{R}^{n\times n}$

$B \in \mathbb{R}^{n\times m}$, $y \in \mathbb{R}^{p\times n}$

- **Step 5:** Observer gain L

Observer dynamical system:

$$\hat{x}_{k+1} = A_d\hat{x}_k + B_d u_k + L(y_k - C_d\hat{x}_k) = (A_d - LC_d)\hat{x}_k + [B_d\ L]\begin{bmatrix}u_k\\ y_k\end{bmatrix}$$

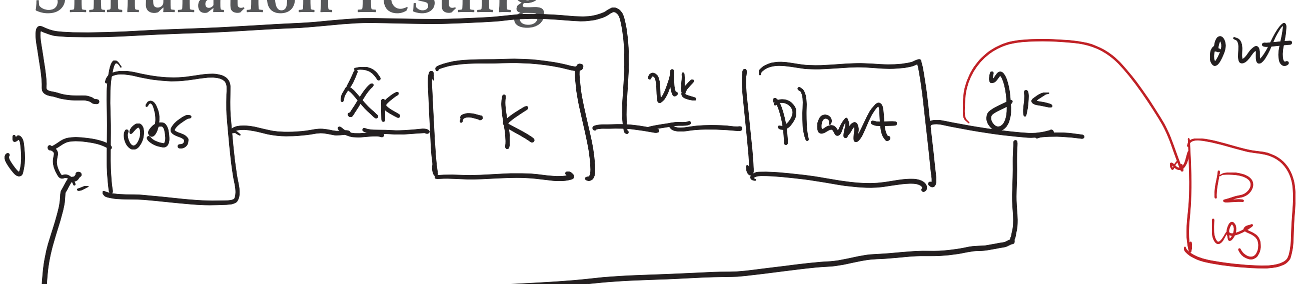


this is a linear system with $\begin{bmatrix}u_k\\ y_k\end{bmatrix} \in \mathbb{R}^{m+p}$ as input

and $\hat{x}_k \in \mathbb{R}^n$

output $\hat{x}_k \in \mathbb{R}^n$

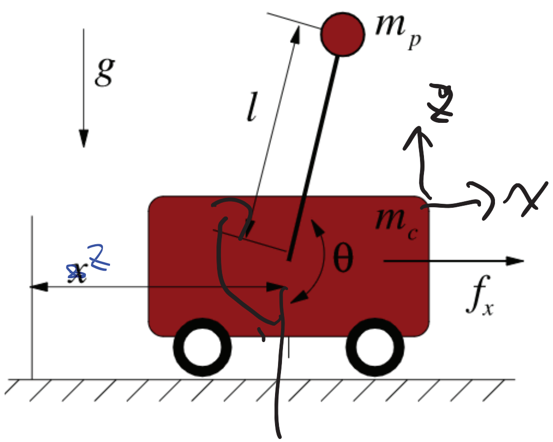
- **Simulation Testing**



Outline

- Short introduction to Drake
- Example 1: Observer and Controller Design
- **Example 2: Cart-Pole Balancing**
- From regulation to tracking control
- Example 3: DC Motor speed tracking control

Cart-Pole Model



$$\ddot{x} \overset{\ddot z}{=} \frac{1}{m_c + m_p \sin^2\theta}\left[f_x + m_p \sin\theta(l\dot{\theta}^2 + g\cos\theta)\right] \quad \cdots ①$$

$$\ddot{\theta} = \frac{1}{l(m_c + m_p \sin^2\theta)}\left[-f_x \cos\theta - m_p l\dot{\theta}^2 \cos\theta\sin\theta - (m_c + m_p)g\sin\theta\right] \quad \cdots ②$$

$$x = \begin{bmatrix} z \\ \theta \\ \dot z \\ \dot\theta \end{bmatrix} \Rightarrow \dot x = \begin{bmatrix} \dot z \\ \dot\theta \\ ①(z,\theta,\dot z,\dot\theta) \\ ②(z,\theta,\dot z,\dot\theta) \end{bmatrix} \leftarrow f(x) \neq Ax+Bu$$

$u \leftarrow f_z$

$f(\hat x, \hat u) = 0$, linearize around $\hat x = \begin{bmatrix} 0 \\ \pi \\ 0 \\ 0 \end{bmatrix}$, $\hat u = 0$

$$\dot x = f(x,u) \approx \underbrace{f(\hat x, \hat u)}_{0} + \underbrace{\left(\frac{\partial f}{\partial x}\Big|_{\hat x, \hat u}\right)}_{\hat A} \cdot (x - \hat x) + \underbrace{\left(\left(\frac{\partial f}{\partial u}\right)\Big|_{\hat x, \hat u}\right)}_{\hat B} \cdot (u - \hat u) + \text{H.O.T.}$$

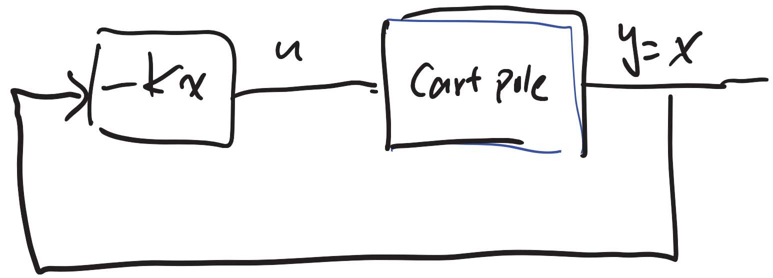
$\dot x = \frac{d}{dt}(x - \hat x) \rightarrow \Delta x$

$= \dot x - 0$

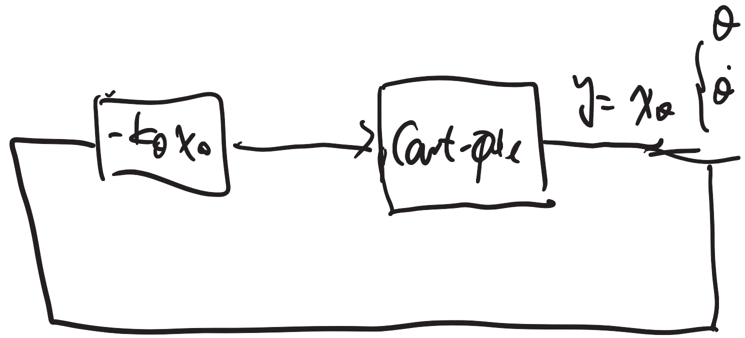
$$\hat A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \\ & & \ddots \end{bmatrix}$$

$$\Delta\dot x = \hat A \,\Delta x + \hat B \,\Delta u$$

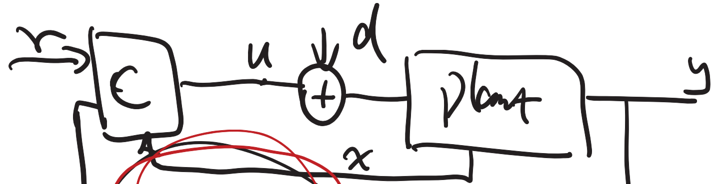
Outline



- Short introduction to Drake
- Example 1: Observer and Controller Design
- Example 2: Cart-Pole Balancing



- **From regulation to tracking control**
- Example 3: DC Motor speed tracking control



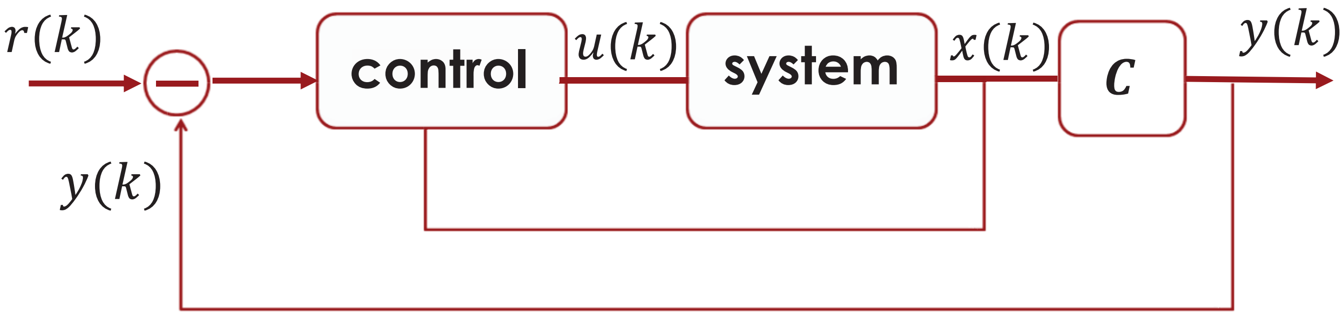
- **Robust tracking problem:**

$$x(k+1) = Ax(k) + Bu(k) + B_d d(k)$$
$$y(k) = Cx(k)$$

  - $d(k)$: disturbance entering the systems

- **Goal:** design u to make output $y(k)$ track a reference $r(k)$

- To simplify discussion, we assume:
  - $r(k)$ and $y(k)$: scalar
  - $u(k)$: full state feedback (add observer if full state is not available)



- **Illustrating example**: consider a simple scalar system

$$x(k+1) = a\,x(k) + u(k), \qquad y(k) = x(k)$$

Suppose tracking reference: $r(k) = r \neq 0$

- Linear feedback doesn't work: $u(k) = -Kx(k) \Rightarrow \quad x(k+1) = \underbrace{(a-K)x(k)}$ $\cancel{(a-K)x(k)}$

If: $|a-K|>1$, then $y(k)=x(k) \uparrow \infty$

$|a-K|<1$, $y(k)=x(k) \downarrow 0$

$|a-K|=1$, $y(k)=\pm x(0)$

- Can we compute the correct input? E.g. assume $|a|<1$, then $x(k) = a^k x_0 + \sum_{j=0}^{k-1} a^{k-j-1} u(j)$

Note: $\sum_{j=0}^{k-1} a^{k-1-j} = \underline{1 + a + a^2 + \cdots + a^{k-1}} = \frac{1-a^k}{1-a} \xrightarrow[|a|<1]{k\to\infty} \frac{1}{1-a}$

$\Rightarrow |a|<1 \quad x(k) = \underbrace{a^k x(0)}_{\substack{k\to\infty \\ \downarrow \\ 0}} + \underbrace{\sum_j a^{k-j-1} u(j)}_{\substack{k\to\infty \\ \frac{1}{1-a}\hat{u}}}$ (assume $u(j) \equiv \hat{u}$)

we want to have $\underline{\frac{1}{1-a}\hat{u} = r} \Rightarrow \underline{\hat{u} = r(1-a)}$

- Challenges for more general tracking problems:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + B_d d(k) \\ y(k) = Cx(k) \end{cases}$$

- $x(k) \in R^n$ can be multi-dimensional ($a \rightarrow A$), so can input $u(k)$

- System may be unstable

- Uncertainties:
  - reference $r(k)$ may not be know a priori
  - we have nontrivial **unknown** disturbance $d(k)$

- How to improve transient performance (track promptly)

→ scalar: sum of all past tracking error

- Introduce an ``integral state'': $z(k+1) = z(k) + \underbrace{r(k) - y(k)}_{\text{tracking error } (e(k))}$

* Fact: if $z(k) \to z^*$ (constant) $\Rightarrow$ $\Rightarrow (r(k) - y(k)) \to 0$

- Extended state space: $\tilde{x}(k) = \begin{bmatrix} x(k) \\ z(k) \end{bmatrix}$ $\begin{matrix} \to \in \mathbb{R}^n \\ \to \mathbb{R} \end{matrix} \Big\} \in \mathbb{R}^{n+1}$

- Feedback control: $\underbrace{u(k)}_{\in \mathbb{R}^m} = \underbrace{[K_x \quad K_z]}_{\in \mathbb{R}^{m \times (n+1)}} \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} = K_x x(k) + K_z z(k)$

- CL dynamics: $\boxed{\tilde{x}(k+1) = \underbrace{\begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix}}_{\tilde{A}} \tilde{x}(k) + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{B}} u(k) + \underbrace{\begin{bmatrix} B_d d(k) \\ r(k) \end{bmatrix}}_{B(k)}}$

$\Downarrow$

$$\tilde{x}(k+1) = \begin{bmatrix} x(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} A x(k) + B u(k) + B_d \cdot d(k) \\ z(k) + r(k) - C x(k) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} + \cdots$$

- Under mild conditions: $(\tilde{A}, \tilde{B})$ is controllable so we can design $\tilde{K}$ such that $\tilde{A} + \tilde{B}\tilde{K}$ has desired eigenvalues

- Closed-loop dynamics: $\tilde{x}(k+1) = (\tilde{A} + \tilde{B}\tilde{K})\tilde{x}(k) + \underbrace{\begin{bmatrix} B_d d(k) \\ r(k) \end{bmatrix}}_{\beta(k)}$

$$\Rightarrow \tilde{x}(k) = \tilde{A}_{CL}^{k} \cdot \tilde{x}(0) + \sum_{j=0}^{k-1} \tilde{A}_{CL}^{k-1-j} \beta(j)$$

- For constant or slowly changing $d(k) \approx d, r(k) \approx r$,
  - the extended state $\tilde{x}(k)$ converges to a finite value.
  - Thus, $z(k+1) = z(k) \Rightarrow$ error becomes zero

– suppose $\tilde{K}$ is chosen such that $\tilde{A}_{CL} = (\tilde{A} + \tilde{B}\tilde{K})$ is stable

then ① $\tilde{A}_{CL}^{k} \to 0$

② $$\sum_{j=0}^{k-1} \tilde{A}_{CL}^{k-1-j} \beta(j) \overset{\beta(j) \approx \bar{\beta}}{=\!=} I \cdot \bar{\beta} + \tilde{A}_{CL} \cdot \bar{\beta} + \cdots + \tilde{A}_{CL}^{k-1} \bar{\beta}$$

$$= (I + \tilde{A}_{CL} + \cdots + \tilde{A}_{CL}^{k-1}) \cdot \bar{\beta} \xrightarrow{k \to \infty} \underbrace{C}_{\text{some constant}} \cdot \bar{\beta}$$

Proof and Discussions

where: $c = I + \tilde{A}_{CL} + \hat{A}_{CL}^2 + \cdots \quad \cdots \quad (i)$

$$\tilde{A}_{CL} \cdot c = \tilde{A}_{CL} + \hat{A}_{CL}^2 + \cdots \qquad (ii)$$

(i) − (ii) $(I - \tilde{A}_{CL}) \cdot c = I$

$$c = (I - \tilde{A}_{CL})^{-1}$$

---

so: $\tilde{x}(k) = \begin{bmatrix} x(k) \\ \underset{\sim}{z(k)} \end{bmatrix} \rightarrow c \cdot \bar{\beta}$

$z(k) \rightarrow \overset{\frown}{\text{some}}$ constant $\Rightarrow e(k) \rightarrow 0$

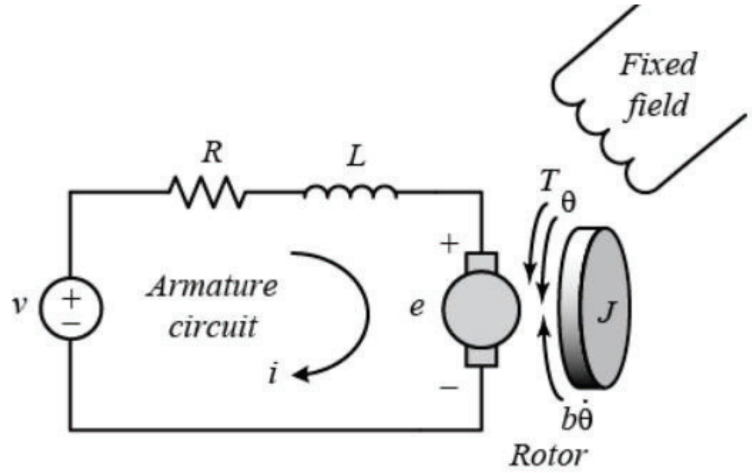
Outline

- Short introduction to Drake
- Example 1: Observer and Controller Design
- Example 2: Cart-Pole Balancing
- From regulation to tracking control
- **Example 3: DC Motor speed tracking control**

▪ DC Motor Speed Tracking Control Example

$$\frac{d}{dt}\begin{bmatrix}\dot\theta\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot\theta\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot\theta\\ i\end{bmatrix}$$



– Motor tracking Example.

$$H(z) = \frac{0.002\,z + 0.00164}{z^2 - 1.52\,z + 0.552}$$

$$u \to \boxed{\text{DC Motor}} \to y$$

⇒ Find state-space model:

$$\frac{Y(z)}{U(z)} = H(z) = \frac{\overbrace{(0.002\,z^{-1} + 0.00164\,z^{-2})}^{N(z)}\,p(z)}{\underbrace{(1 - 1.52\,z^{-1} + 0.552\,z^{-2})}_{D(z)}\,p(z)}$$

$$\Rightarrow \underbrace{Y(z) = N(z)\,p(z)}_{z^{-1}}$$

time domain: $\Downarrow$

$$y(k) = 0.002\,p(k-1) + 0.00164\,p(k-2)$$

- Summary

$\Rightarrow U(z) = D(z)\,p(z) \Longleftrightarrow$ time domain $\quad u(k) = p(k) - 1.52\,p(k-1) + 0.552\,p(k-2)$

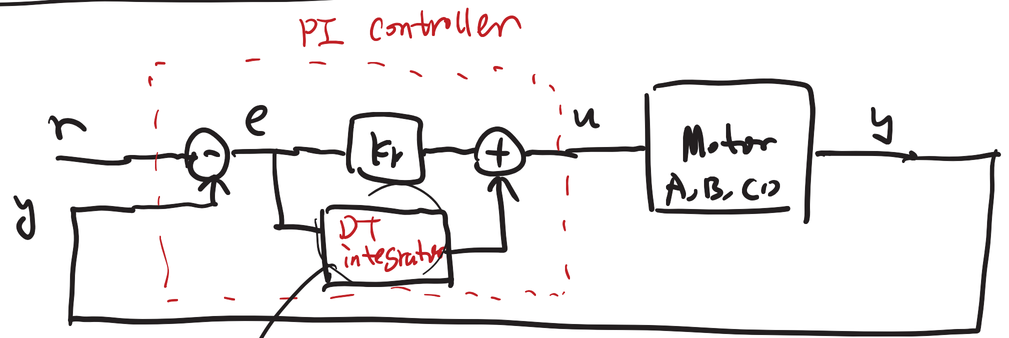
$p(k) = -0.552\,p(k-2) + 1.52\,p(k-1) + u(k)$

$\Rightarrow$ state: let $x_1(k) = p(k-2)$

$x_2(k) = p(k-1)$

$$x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -0.552 & 1.52 \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 0.0164 & 0.002 \end{bmatrix}}_{C} x(k)$$



✓ Test in Drake

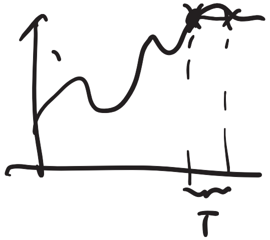
$$\frac{K_I \cdot 0.05}{1 - z^{-1}} \qquad \frac{K_I \cdot T}{1 - z^{-1}}$$

sampling time

both correct

$$\frac{K_I \cdot T}{z - 1}$$

Define: $x(k) = \underbrace{\int_0^{k\cdot T} e(t)\,dt}$



$$\Rightarrow x(k+1) = x(k) + \underbrace{\int_{k\cdot T}^{(k+1)T} e(t)\,dt}$$

$$= x(k) + e(k\cdot T)\cdot T$$

$$\Rightarrow \boxed{\begin{aligned} x(k+1) &= x(k) + T\cdot e(k) \\ y(k) &= k_I \cdot x(k) \end{aligned}} \Leftrightarrow \left\{\begin{aligned} A &= 1 \\ B &= T \\ C &= k_I \\ D &= 0 \end{aligned}\right\} \Rightarrow$$

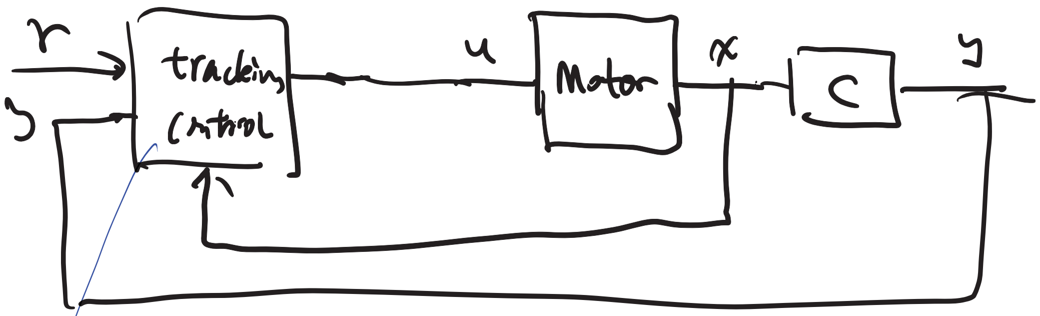
$$H(z) = C(zI - A)^{-1}B + D$$

$$= k_I \frac{1}{z-1} T$$

$$= \frac{k_I \cdot T}{z-1}$$

PI Block:

$$\left\{\begin{aligned} x(k+1) &= x(k) + T\cdot(r(k) - y(k)) \\ \underline{y}(k) &= k_I x(k) + k_p\cdot \underline{(r(k) - y(k))} \end{aligned}\right.$$



$$u = k_x \cdot x + k_I \cdot z$$

$$\tilde{K} = \begin{bmatrix} k_x & k_I \end{bmatrix}$$

$$\tilde{A} = \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

$$z(k+1) = z(k) + (r - y)$$