SDM366 Optimal Control and Estimation

Lecture Note 2
State Space Model and Stability

~

Prof. Wei Zhang
Southern University of Science and Technology

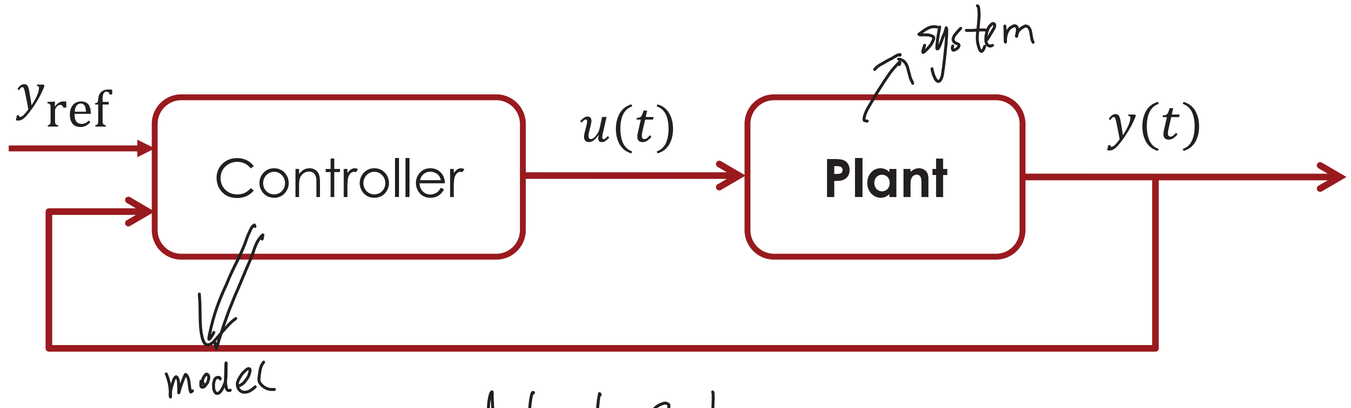
zhangw3@sustech.edu.cn
https://www.wzhanglab.site/

Outline

- **State space model: definition and examples**
- From continuous-time to discrete time model
- From nonlinear to linear model
- System solution and stability

State-space model based feedback control system:

- Goal: determine control input to achieve desired output



model ≠ system

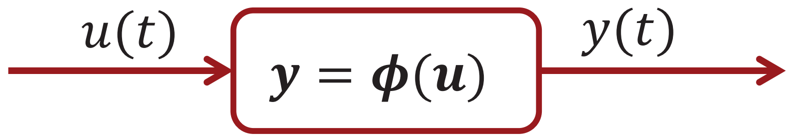
Mamba

- Controller design is based on plant model
  - Model is different from the actual plant
  - "all models are wrong, but some are useful"

- Modeling approach:
  - First principle ← physics
  - Data driven (System ID) ←

Static vs. Dynamic Systems

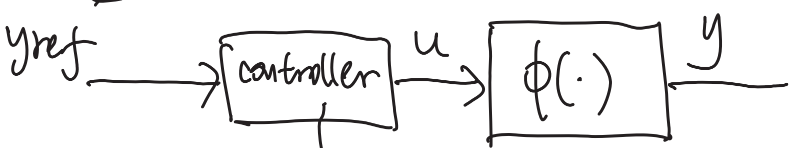
- Static system



$y = 2u$

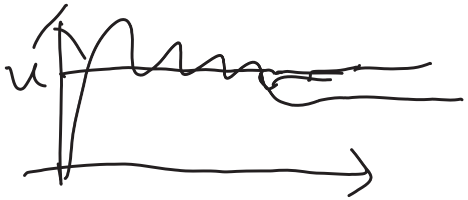
  - $u(t)$ completely and immediately determines $y(t)$
  - Desired output y_{ref} can be perfectly tracked (in absence of disturbance) by open-loop plant inversion

$y = y_{ref}$



$u = \phi^{-1}(y_{ref})$

$\ddot{y}+2y=-u$



- **Static vs. Dynamic Systems**
  - **Dynamic system:** differential or difference equation

$$u(t) \longrightarrow \boxed{\text{ODE}(y,u)} \longrightarrow y(t)$$

ordinary differential (difference) equation

- $u(t)$ does not fully determines $y(t)$
- At time t_0, the output $y(t_0)$ does not fully captures the system "behavior"

y = position $\Rightarrow m\ddot{y}=u$

$u \rightarrow \boxed{\text{cart}} \rightarrow y$

at time $t=t_0$, $y(t_0)=p$

we need $\left\{ \begin{matrix} y(t_0) \\ \dot{y}(t_0) \end{matrix} \right.$ + future u $\Rightarrow$ system behavior

- **"State": info needed for future evolution,** it separates future from past
- **State** $x(t_0)$ at time t_0 and **input** $\boldsymbol{u}(\boldsymbol{t})$ over time $[t_o, t_f]$, **completely determines** the system behaviors

$x \in \mathbb{R}^n$, $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

- **General continuous-time state space model**

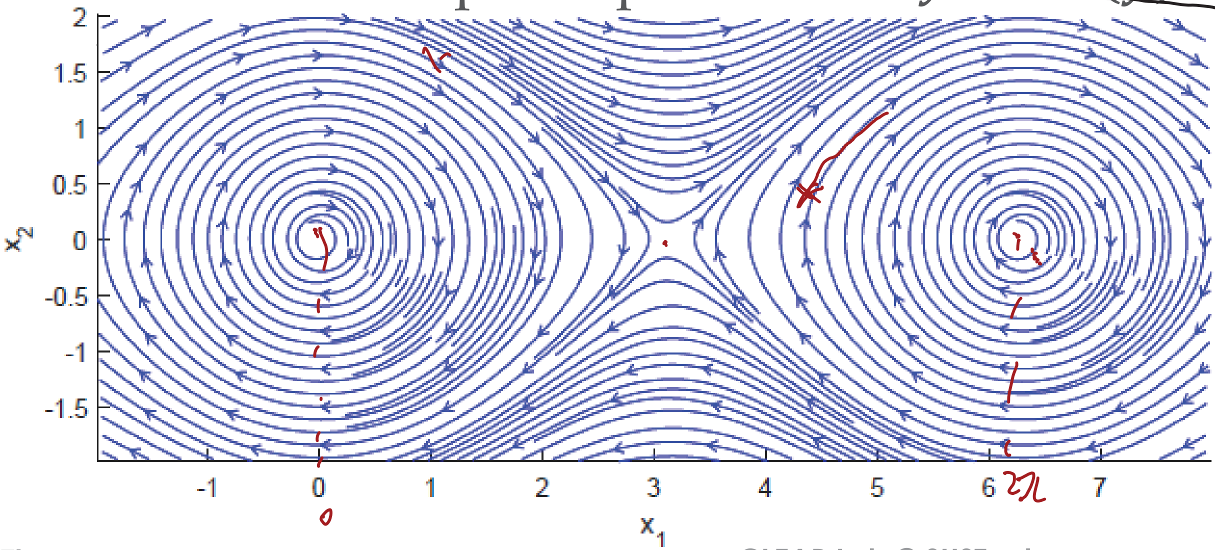
All "finite-dim" dynamic system can be written in this "state-space" form

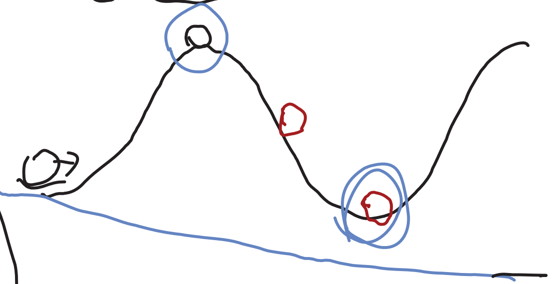
$$\boxed{\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}}$$

→ 1st order differential equ in $\mathbb{R}^n$

  - $x \in R^n$ state vector, $u \in R^m$ control input, $y \in R^p$ output,
  - $f: R^n \times R^m \to R^n$: called **vector field** specify velocity of state
  - $h: R^n \times R^m \to R^p$: output function
  - Called "autonomous" system if there is no control $f(x,u) = f(x)$
  - For autonomous sys, $\hat{x} \in \boldsymbol{R^n}$ is called **equilibrium** if $f(\hat{x}) = 0$

Vector field example of pendulum: $\ddot{y} + \sin(y) = 0$





$x_1 = y$

$x_2 = \dot{y}$

$$\frac{d}{dt}\underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} x_2 \\ -\sin(x_1) \end{bmatrix}}_{f(x)}$$

- **General discrete-time state space model**

$x_{k+1} = f(x_k) = \sin x_k$

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = h(x(k), u(k))$$

$x = \sin x$

  - $x \in R^n$ state vector, $u \in R^m$ control input, $y \in R^p$ output
  - $f: R^n \times R^m \to R^n$: state update equation
  - $h: R^n \times R^m \to R^p$: output function
  - Called autonomous system if there is no control $f(x,u) = f(x)$
  - For autonomous sys, $\hat{x} \in \boldsymbol{R^n}$ is called **equilibrium** if $\hat{x} = f(\hat{x})$

- Discrete-time system:
  - Some discrete-time system is obtained from continuous time model by sampling
  - Some systems naturally evolve in discrete time.

- **Linear Systems:** system is called linear if:

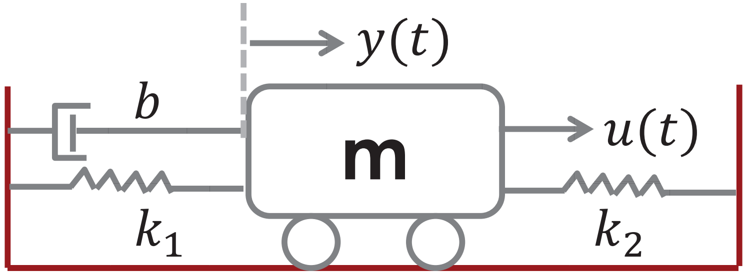
Continuous time
$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{Ax} + \boldsymbol{Bu},$$
$$\mathbf{y} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{Cx} + \boldsymbol{Du},$$

Discrete time
$$\boldsymbol{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{k}),\boldsymbol{u}(\boldsymbol{k})) = \boldsymbol{Ax}(\boldsymbol{k}) + \boldsymbol{Bu}(\boldsymbol{k}),$$
$$\mathbf{y}(\boldsymbol{k}) = \boldsymbol{h}(\boldsymbol{x}(\boldsymbol{k}),\boldsymbol{u}(\boldsymbol{k})) = \boldsymbol{Cx}(\boldsymbol{k}) + \boldsymbol{Du}(\boldsymbol{k}),$$

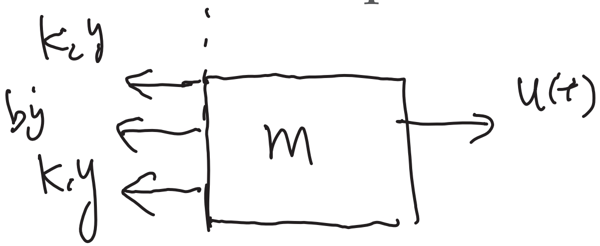
for some matrices A, B, C, D

- **State-space modeling:**
  - Find the functions $f(\cdot,\cdot), h(\cdot,\cdot)$
  - Or find A, B, C, D matrices if the system is linear

Example 1: Consider spring-damper cart system with zero initial conditions (initially at y = 0 and not moving). No friction



- Differential equation model



force analysis

$$\Rightarrow \quad m\ddot{y} = u(t) - k_1 y - k_2 y - b\dot{y}$$

$$= u(t) - (k_1+k_2)y - b\dot{y}$$

- State space model of Example 1 (infinitely many)

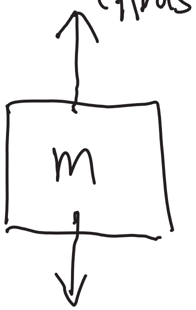
Let's define $x_1 = y$, $x_2 = \dot{y}$ $\Rightarrow$ $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$ $\Longleftrightarrow$ $\dot{x} = f(x, u)$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}(u - bx_2 - (k_1+k_2)x_1) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k_1+k_2}{m} & -\frac{b_2}{m} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{B} \underbrace{u}_{u}$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{0}_{D} \cdot u$$

- **Example 2**: soft landing of a lunar module, $u = \dfrac{dm}{dt}$

thrust $= k \cdot \dfrac{dm}{dt}$



mg

$$\begin{cases} m\ddot{y} = -mg + ku \\ \dot{m} = u \end{cases}$$

$\Rightarrow \quad x_1 = y, \quad x_2 = \dot{y}, \quad x_3 = m$

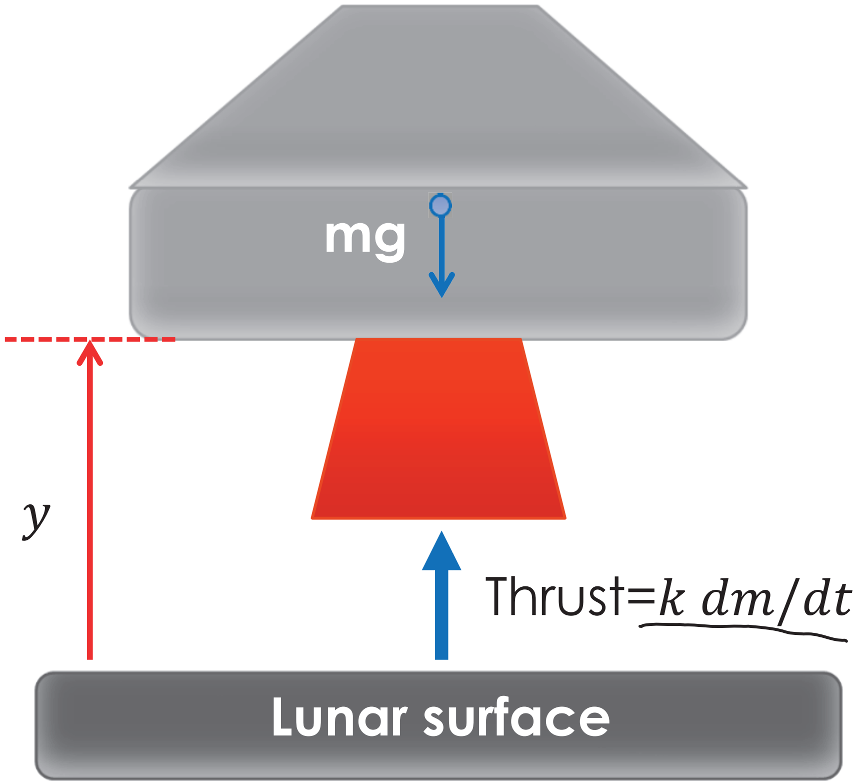
$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ -g + \dfrac{ku}{x_3} \\ u \end{bmatrix} = \underbrace{f(x,u)}$$

$$\Downarrow$$

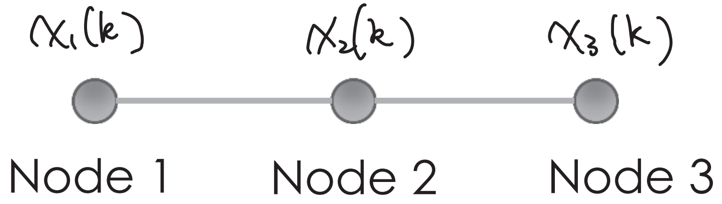
$$\begin{bmatrix} f_1(x_1, x_2, x_3, u) \\ f_2(\cdots\cdots) \\ f_3(\quad) \end{bmatrix}$$

$f_2 \to g + \dfrac{ku}{x_3}$

$\neq Ax + Bu$



- **Example 3**: Sensor Network
  - Each iteration, exchange measurements with neighbors
  - The updated value is the average of its own value with the neighbors



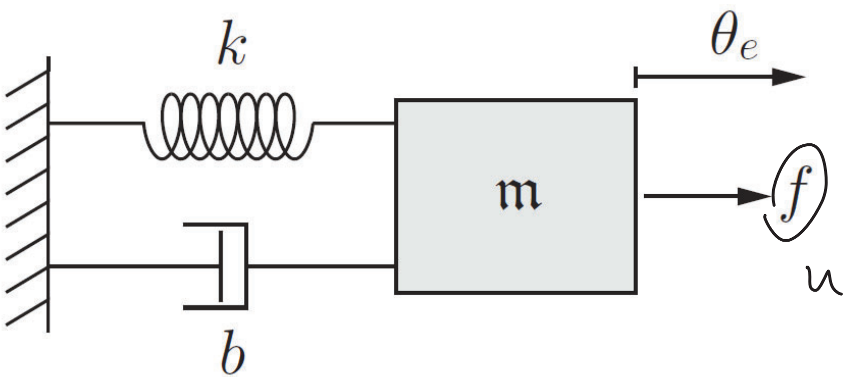
← value stored at node i at time k

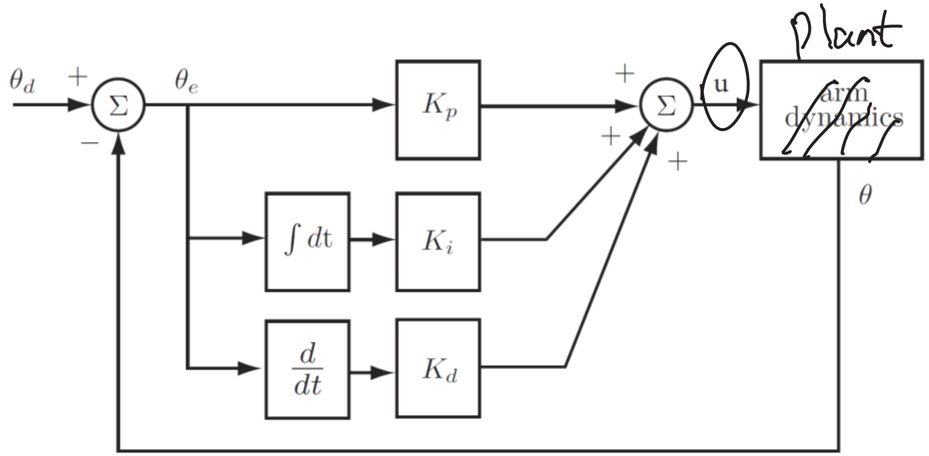
$$x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left(x_1(k) + x_2(k)\right) \\ \frac{1}{3}\left(x_1(k) + x_2(k) + x_3(k)\right) \\ \frac{1}{2}\left(x_2(k) + x_3(k)\right) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}}_{x(k)}$$

$$A \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

closed-loop

- **Example 4:** PID for spring-damper system





plant dynamics: $m\ddot{\theta}_e + b\dot{\theta}_e + k\theta_e = u$

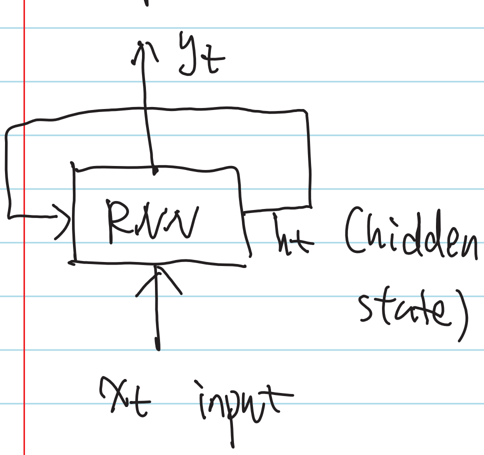
using PID: $u = k_p\theta_e + k_i\int \theta_e dt + k_d \cdot \dot{\theta}_e$

closed-loop dynamics: $m\ddot{\theta}_e + (b-k_d)\dot{\theta}_e + (k-k_p)\theta_e - k_i\int \theta_e dt = 0$

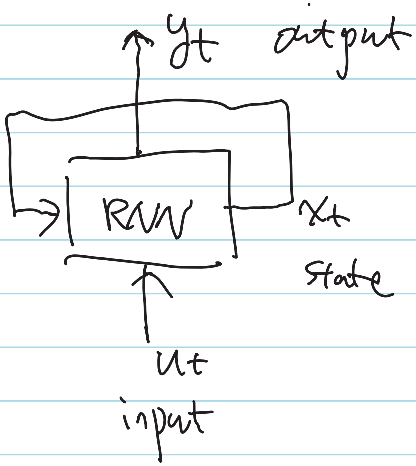
state variable: $x_1 = \int \theta_e dt$, $x_2 = \theta_e$, $x_3 = \dot{\theta}_e$

$$\Rightarrow \dot{x} = \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ \frac{k_i}{m}x_1 + \frac{k_p-k}{m}x_2 + \frac{k_d-b}{m}x_3\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ \frac{k_i}{m} & \frac{k_p-k}{m} & \frac{k_d-b}{m}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

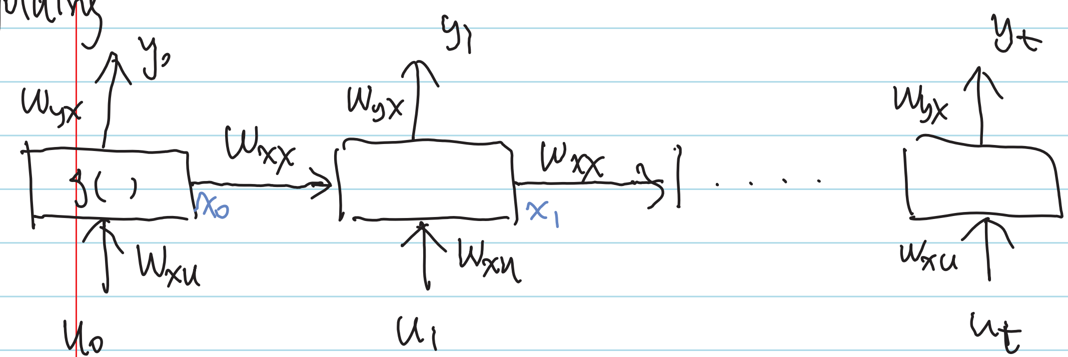
- Simple RNN : Recurrent Neural Network.



$\Longrightarrow$



unfolding

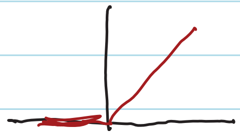


$$\Longrightarrow \quad x_{t+1} = \underbrace{g}_{\text{activation func}}\left(W_{xx}^T x_t + W_{xu}^T u_t \right) = f(x_t, u_t)$$

$$y_t = W_{yx} \cdot x_t$$

$= \begin{cases} \text{- ReLu :} \\ \text{- tanh :} \end{cases}$

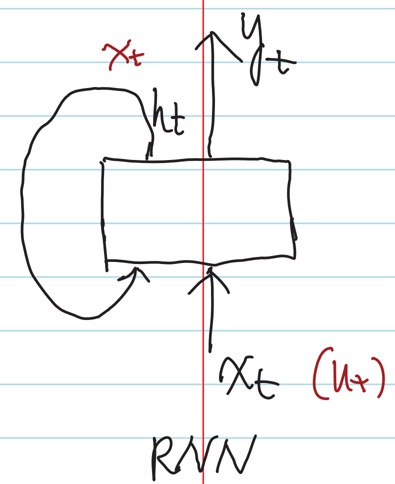
max(0, z)



※ RNN is a special case of state space model.

This lecture

- Pytorch implementation of a simple RNN ←
- "Simulation" of RNN
- state space model implementation of RNN in Python.



$u \longrightarrow \boxed{NN(w)} \longrightarrow y$ function mapping from $u \in \mathbb{R}^{\text{input\_dim}}$ to $y \in \mathbb{R}^{\text{output\_dim}}$

$u_1, \cdots, u_L \longrightarrow \boxed{RNN(w)} \longrightarrow y_1, \cdots, y_L$ maps from sequence of input vectors to . . . . . output . . .

$$\begin{bmatrix} u_1 \\ \vdots \\ u_L \end{bmatrix} \not\longrightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix}$$

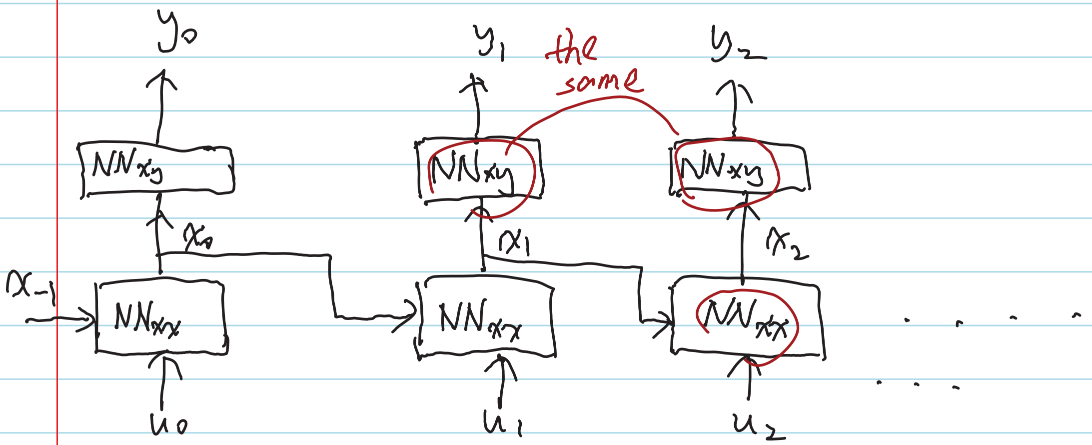
- Assumption: ① Causality: $y_k = g(u_1, \cdots, u_k)$

② Finite-dim representation: $\exists$ State $x_t \in \mathbb{R}^{n_x}$ such that $\{ y_t = g(x_t, u_t)$

$(x_{t+1} = f(x_t, u_t))$

Pytorch :

$$\begin{cases} x_t = g_{xx}(x_{t-1}, u_t) \overset{\text{1-layer case}}{=} \tanh(w_{xx}^T x_{t-1} + b_{xx} + w_{ux}^T u_t + b_{ux}) \\ y_t = g_{xy}(x_t, u_t) \overset{\text{Linear case}}{=} w_{xy}^T x_t + b_{xy} \end{cases}$$



- Specify : dim of u_t, x_t, y_t .

NN_{xx} , NN_{xy}

$K=0$

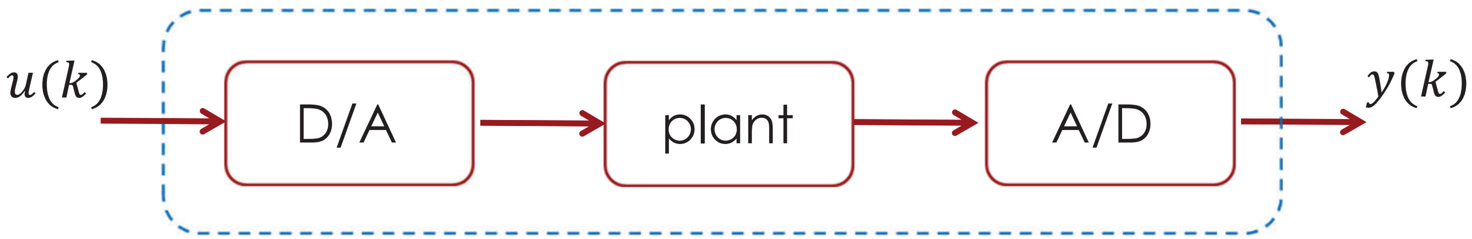
$x = [x_0 ; x_1 ; x_2 ; \dots]$

$u = [u_0 ; u_1 ; u_2 \dots$

Outline

- State space model: definition and examples
- **From continuous-time to discrete time model**
- From nonlinear to linear model
- System solution and stability

From continuous time to discrete time model



- Approximate differential equation with difference equation
  - Euler forward rule:

$\in \mathbb{R}^n$

From calculus: $$\dot{g}(t) = \lim_{\Delta t \to 0} \frac{g(t+\Delta t) - g(t)}{\Delta t}$$

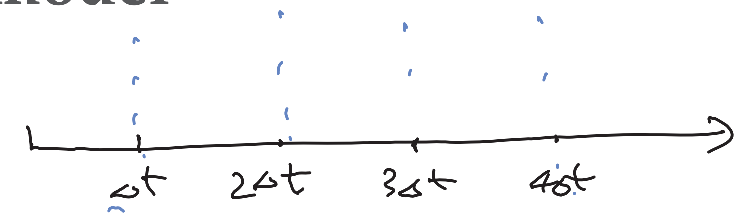
For small enough Δt: $$\dot{g}(t) \approx \frac{g(t+\Delta t) - g(t)}{\Delta t}$$

$$\Rightarrow \underline{g(t+\Delta t) = g(t) + \dot{g}(t) \cdot \Delta t}$$

From continuous-time to discrete-time model

- General nonlinear case:

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



Define: $x[k] \triangleq x(k\Delta t)$, $u[k] = u(k\Delta t)$, $y[k] = y(k\Delta t)$

$$x[k+1] = x((k+1)\Delta t) \approx x(k\Delta t) + \dot{x}(k\Delta t)\,\Delta t$$
$$= x[k] + f(x[k], u[k])\cdot\Delta t$$
$$\triangleq f_d(x[k], u[k])$$

$$\left.\begin{aligned} y[k] &= h(x[k], u[k]) \\ &= h_d(\ \cdot\ \cdot\ \cdot\ ) \end{aligned}\right\} \Rightarrow \begin{cases} x[k+1] = f_d(x[k], u[k]) \\ y[k] = h_d(x[k], u[k]) \end{cases}$$

discretization

From continuous-time to discrete-time model

- Linear case:

$$\dot{x} = A_c x + B_c u,$$
$$y = C_c x + D_c u,$$

$\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Delta t$

$x \in \mathbb{R}^n$ $\quad u \in \mathbb{R}^m$

using previous result:

$$x(k+1) = x(k) + \left(A_c x(k) + B_c u(k)\right) \cdot \Delta t$$

discrete time

$$= \underbrace{\left[\underbrace{I_{2\times 2} + A_c \cdot \Delta t}_{n\times n}\right]}_{A_d} \underbrace{x(k)}_{\mathbb{R}^{n\times 1}} + \underbrace{\underbrace{B_c \cdot \Delta t}_{n\times m}}_{B_d} \underbrace{u(k)}_{m\times 1}$$

$\Rightarrow$ DT sys with Δt sampling interval.

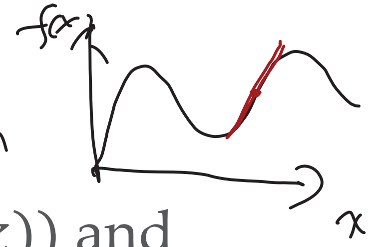
$$\begin{cases} x(k+1) = A_d x(k) + B_d u(k) \\ y(k) = C_d x(k) + D_d u(k) \end{cases}$$

$C_d = C_c, \quad D_d = D_c$

Outline

- State space model: definition and examples
- From continuous-time to discrete time model
- **From nonlinear to linear model**
- System solution and stability

From nonlinear to linear (in discrete time)



- Given model: $x(k+1) = f(x(k), u(k))$, $y(k) = h(x(k), u(k))$ and operating point: $(\hat{x}, \hat{u})$

  nonlinear

  nonlinear

- Goal: find a linearized model around $(\hat{x}, \hat{u})$

Define: $\Delta x = x - \hat{x}$, $\Delta u = u - \hat{u}$, $\Delta y = y - h(\hat{x}, \hat{u})$

Goal: $$\underbrace{\Delta x(k+1)}_{\mathbb{R}^n} \approx \hat{A}\, \underbrace{\Delta x(k)}_{\mathbb{R}^n} + \hat{B}\, \underbrace{\Delta u(k)}_{\mathbb{R}^m} + C$$

- Jacobian matrix of multivariable function $f: R^n \to R^m$

$f: \mathbb{R}^3 \to \mathbb{R}^2$

$$\begin{bmatrix} f_1(z_1, z_2, z_3) \\ f_2(z_1, z_2, z_3) \end{bmatrix} \in \mathbb{R}^2$$

$$\frac{\partial f}{\partial z} \triangleq \left[\frac{\partial f_i}{\partial z_j}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \frac{\partial f_1}{\partial z_3} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \frac{\partial f_2}{\partial z_3} \end{bmatrix}, \quad \boxed{df = \frac{\partial f}{\partial z} dz}$$

$i = 1, 2$

$j = 1, 2, 3$.

$$\begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots \end{bmatrix} \begin{bmatrix} dz_1 \\ dz_2 \\ dz_3 \end{bmatrix}$$

- Example of Jacobian matrix: $f(z) = \begin{bmatrix} 2z_1 + e^{z_2} \\ \log(z_3) + \frac{1}{z_2} \end{bmatrix}$, $\hat{z} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

$f: \mathbb{R}^3 \to \mathbb{R}^2$

$$\frac{\partial f}{\partial z}(z) = \begin{bmatrix} 2 & e^{z_2} & 0 \\ 0 & \frac{-1}{z_2^2} & \frac{1}{z_3} \end{bmatrix}\Bigg|_{\hat{z}=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}} = \begin{bmatrix} 2 & e^2 & 0 \\ 0 & \frac{-1}{4} & \frac{1}{3} \end{bmatrix}$$

- Taylor expansion of multivariate function

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

$f(\hat{z}) \in \mathbb{R}^m$, $f: \mathbb{R}^n \to \mathbb{R}^m$

  - General expression: $f(z) = \underbrace{f(\hat{z})}_{\in \mathbb{R}^m} + \underbrace{\left(\frac{\partial f}{\partial z}(z)\Big|_{z=\hat{z}}\right)}_{\mathbb{R}^{m\times n}} \underbrace{\Delta z}_{\in \mathbb{R}^n} + \text{H.O.T}$

$f: \mathbb{R}^n \to \mathbb{R}^m$

Given model: $\left(x(k+1) = f(x(k), u(k)),\ y(k) = h(x(k), u(k))\right)$

- Linearization around $\boxed{(\hat{x}, \hat{u})}$ using Taylor expansion:

$$f(z) = f(\hat{z}) + \frac{\partial f}{\partial z} \cdot \Delta z + \text{H.O.T.}$$

$$z = \begin{bmatrix} x \\ u \end{bmatrix}, \quad \hat{z} = \begin{bmatrix} \hat{x} \\ \hat{u} \end{bmatrix}$$

$$f(x,u) \approx f(\hat{x},\hat{u}) + \overbrace{\left(\frac{\partial f(x,u)}{\partial x}\bigg|_{x=\hat{x},u=\hat{u}}\right)}^{\hat{A}} \cdot \overbrace{(x-\hat{x})}^{\Delta x} + \overbrace{\left(\frac{\partial f(x,u)}{\partial u}\bigg|_{x=\hat{x},u=\hat{u}}\right)}^{\hat{B}} \cdot \overbrace{(u-\hat{u})}^{\Delta u}$$

$$= \hat{A} \cdot \Delta x + \hat{B} \cdot \Delta u + f(\hat{x},\hat{u})$$

Define: $\Delta x_k = x_k - \hat{x}$, $\quad \Delta u_k = u_k - \hat{u}$, $\quad \Delta y_k = y_k - h(\hat{x}, \hat{u})$

Goal: $\Delta x_{k+1} \approx \hat{A}\, \Delta x_k + \hat{B}\, \Delta u_k + C$

$\Delta x_{k+1} = f(x_k, u_k) - \hat{x}$

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

$$f(z) = f(x,u) = f(\hat{z}) + \frac{\partial f}{\partial z}\underbrace{(z - \hat{z})}$$

$$\frac{\partial f}{\partial z} = \left[\begin{array}{ccc|ccc} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \cdots & \frac{\partial f_1}{\partial z_i} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \frac{\partial f_2}{\partial z_1} & \cdots & & & \cdots & \end{array}\right]$$

$$\underbrace{\qquad\qquad}_{\frac{\partial f}{\partial x}} \quad \underbrace{\qquad\qquad}_{\frac{\partial f}{\partial u}}$$

$$= \begin{bmatrix} x - \hat{x} \\ u - \hat{u} \end{bmatrix}$$

$$= f(\hat{x}, \hat{z}) + \left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial u}\right]\begin{bmatrix} x - \hat{x} \\ u - \hat{u} \end{bmatrix} + \text{H.O.T.} = f(\hat{x}, \hat{u}) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial u}\Delta u$$

$+ \text{H.O.T.}$

Apply to state space model: $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ $f: \mathbb{R}^{n+m} \to \mathbb{R}^n$

$$h(x,u) \approx h(\hat{x},\hat{u}) + \underbrace{\left(\frac{\partial h(x,u)}{\partial x}\bigg|_{x=\hat{x},u=\hat{u}}\right)}_{\hat{C}} \cdot \underbrace{(x-\hat{x})}_{\Delta x} + \underbrace{\left(\frac{\partial h(x,u)}{\partial u}\bigg|_{x=\hat{x},u=\hat{u}}\right)}_{\hat{D}} \cdot \underbrace{(u-\hat{u})}_{\Delta u}$$

$$\Delta x_{k+1} \triangleq x_{k+1} - \hat{x} = f(\underbrace{x_k, u_k}) - \hat{x}$$

$$\approx f(\hat{x},\hat{u}) + \underbrace{\left[\frac{\partial f}{\partial x}\bigg|_{\hat{x},\hat{u}}\right]}_{A} \cdot \Delta x_k + \underbrace{\left[\frac{\partial f}{\partial u}\bigg|_{\hat{x},\hat{u}}\right]}_{B} \cdot \Delta u_k - \hat{x}$$

$$= A \cdot \Delta x_k + B \cdot \Delta u_k + f(\hat{x},\hat{u}) - \hat{x}$$

← ① known if $\hat{x}, \hat{u}$ are given

② is zero if $\hat{x}, \hat{u}$ is equilibrium

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

$$\boxed{\Delta y := y - h(\hat{x},\hat{u}) \approx \hat{C} \cdot \Delta x + \hat{D} \cdot \Delta u}$$

$$\Delta y_k = y_k - h(\hat{x},\hat{u}) = \underbrace{h(x_k,u_k)} - \underbrace{h(\hat{x},\hat{u})}$$

$$= \cancel{h(\hat{x},\hat{u})} + \underbrace{\left[\frac{\partial h}{\partial x}\right]}_{C} \cdot \Delta x_k + \underbrace{\left[\frac{\partial h}{\partial u}\right]}_{D} \Delta u_k - \cancel{h(\hat{x},\hat{u})}$$

$$= C \Delta x_k + D \Delta u_k$$

- **Example:**
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \overbrace{\begin{bmatrix} \sin(x_2(k)) + \cos(u_2(k)) \\ x_1(k)x_2(k) + u_1u_2(k) \end{bmatrix}}^{f(x_k, u_k)}$$

$$y(k) = \cos(x_2(k)) + 2x_1(k) \qquad \hat{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \hat{u} = \begin{bmatrix} 0 \\ \frac{\pi}{2} \end{bmatrix}$$

: 2-input, u_1, u_2

2-dim state

1-dim output

$$\hat{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}\Bigg|_{\hat{x},\hat{u}} = \begin{bmatrix} 0 & \cos(x_2(k)) \\ x_2(k) & x_1(k) \end{bmatrix}\Bigg|_{\hat{x},\hat{u}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\hat{B} = \frac{\partial f}{\partial u}\Bigg|_{\hat{x},\hat{u}} = \begin{bmatrix} 0 & -\sin(u_2(k)) \\ u_2(k) & u_1(k) \end{bmatrix}\Bigg|_{\hat{x},\hat{u}} = \begin{bmatrix} 0 & -1 \\ \frac{\pi}{2} & 0 \end{bmatrix}$$

$$\hat{C} = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix}\Bigg|_{\hat{x},\hat{u}} = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

$$\hat{D} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\Delta x_{k+1} = \hat{A}\Delta x_k + \hat{B}\Delta u_k$$

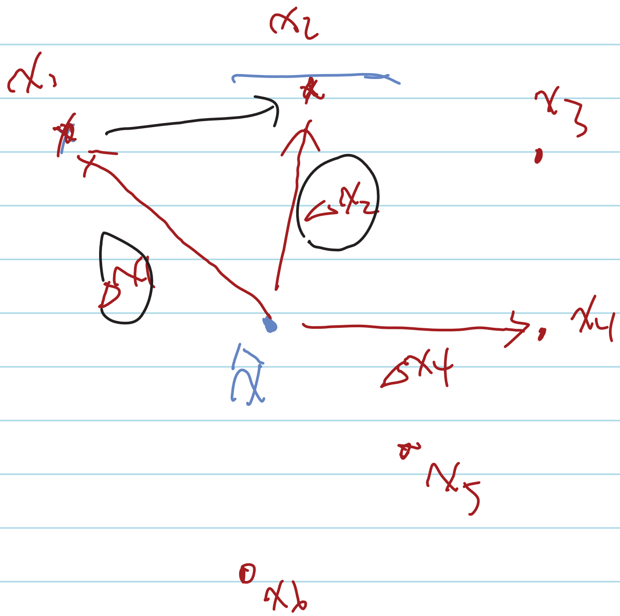
$$\Delta y_k = \hat{C}\Delta x_k + 0$$

$$x_{k+1} = f(x_k, u_k) \qquad f(x,u) \qquad f(q,p)$$

$$= 2x_k + 5u_k \qquad = 2x+5u$$

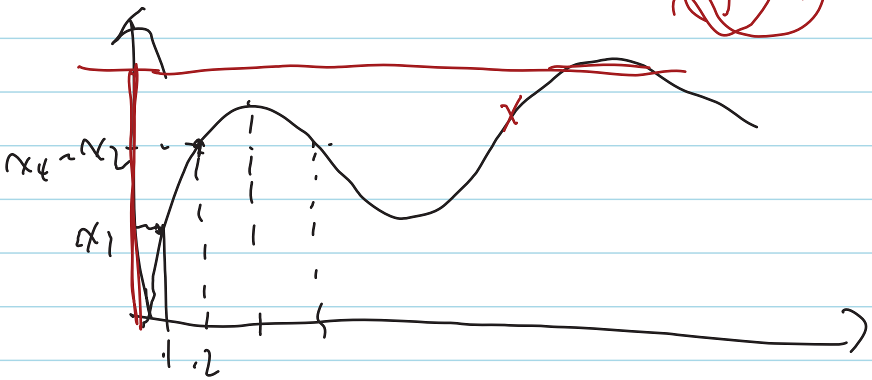
$$x_2 = f(x_1) \qquad \Delta x_k = x_k - \hat{x}$$

$$x_3 = f(x_2)$$

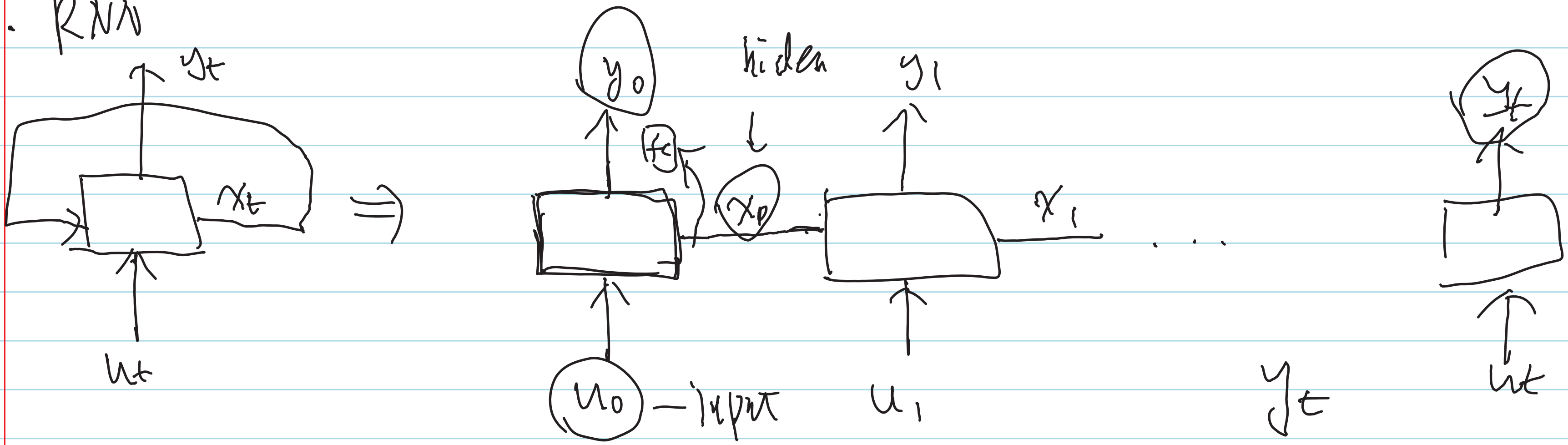


$$\Delta x_3 = \hat{A}\, \Delta x_2$$

$f(x)$ $\mathbb{R}^1$



· RNN



$$x_t = g(W_{xx}^T x_{t-1} + W_{ux} \cdot U_t)$$

$$y_t = \underbrace{W_{xy} \cdot x_t}$$

$x_0 \to x_1 \to x_2 \cdots$

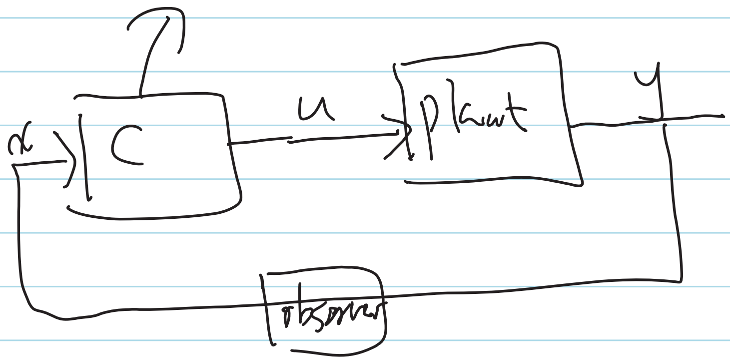
x_0

x_1

x_1

⋮

$u = f\mu(x)$



Outline

- State space model: definition and examples
- From continuous-time to discrete time model
- From nonlinear to linear model
- **System solution and stability**

- General linear state space model:

$$x(k+1) = A(k)x(k) + B(k)u(k),$$
$$y(k) = C(k)x(k) + D(k)u(k)$$

  - If system matrices $\big(A(k), B(k), C(k), D(k)\big)$ change over time k, then system is called **Linear Time Varying (LTV)** system

  - If system matrices are constant w.r.t. to time, then the system is called a **Linear Time Invariant (LTI)** System

$$x(k+1) = Ax(k) + Bu(k),$$
$$y(k) = Cx(k) + Du(k)$$

- Derivation of Solution to LTI state space system:

$$x(k+1) = Ax(k) + Bu(k),$$
$$y(k) = Cx(k) + Du(k)$$

- given initial state $x(0) = \hat{x}$, and control sequence $u(0), \ldots, u(k)$, $k \geq 0$, we have $\boldsymbol{x(k) = A^k \hat{x} + \sum_{j=0}^{k-1} A^{k-j-1} Bu(j)}$

- $x(1) = Ax(0) + Bu(0) = A\hat{x} + Bu(0)$

$x(2) = Ax(1) + Bu(1) = A^2\hat{x} + ABu(0) + Bu(1)$

$x(3) = Ax(2) + Bu(2) = A^3\hat{x} + A^2Bu(0) + ABu(1) + Bu(2)$

$\vdots$

For any k $\quad x(k) = A^k\hat{x} + A^{k-1}Bu(0) + A^{k-2}Bu(1) + \cdots + Bu(k-1)$

$= A^k\hat{x} + \sum_{j=0}^{k-1} A^{k-1-j}Bu(j)$

$y(k) = \boxed{CA^k\hat{x}} + \boxed{\sum_{j=0}^{k-1} CA^{k-1-j}Bu(j) + Du(k)}$

zero-input response

zero-state response

Given system (A, B, C, D)

- Stability
- ~~controllability~~
- ~~observability~~

- A large portion of control applications can be transformed into a ***regulation* problem**

  - Regulation problem: keep certain function of the state $x(k)$ or output $y(k)$ close to a known constant reference value under disturbances and model uncertainties

For example:

- Keep inverted pendulum at upright position ($\theta = 0$)
- Maintain a desired attitude of spacecraft or aircraft
- Air conditioner regulate temperate close to setpoint (e.g. 75F)
- Cruise control maintain a constant speed despite uncertain road conditions
- Converter maintains a desired voltage level for different loads

  - If reference $y_{\text{ref}}(t)$ is changing, this is no longer a regulation problem (becomes a **tracking problem**)

- stability: ① Bounded input Bounded output. ← external

② "convergence" ← internal

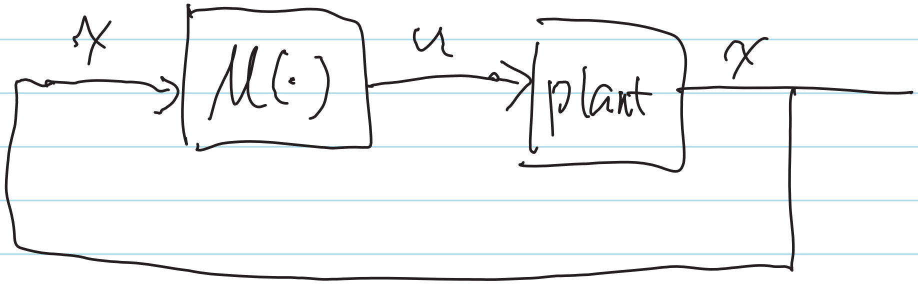
- stability ~~of~~ a system ⟹

- equilibrium: Autonomous system. $\dot{x} = f(x)$, $x_{k+1} = f(x_k)$

  $\hat{x}$ is equilibrium: $f(\hat{x}) = 0$, $f(\hat{x}) = \hat{x}$

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

↳ controller : $u(\cdot)$, $\Rightarrow$ closed-loop system



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

$$= f_{cL}(x)$$

$\hat{x}$ is equilibrium of f_{cL} if $f_{cL}(\hat{x}) = 0$

in discrete time: $f_{cL}(\hat{x}) = \hat{x}$

$u(x) \equiv 2$

$$\dot{x} = \underbrace{f(x, 2)} = f_{cL}(x)$$

- **Internal Stability** (with $u(k) \equiv 0$, i.e. concerned with zero-input state response)

$$x(k+1) = Ax(k) + Bu(k),$$
$$y(k) = Cx(k) + Du(k)$$

  - Asymptotic stable: $||x(k)|| \to 0$, as $k \to \infty$, for all initial state $\hat{x}$
  - Marginal stable: $||x(k)|| \leq M$, for all $k = 1,2, \ldots$

- Recall state space solution for linear systems:

$$\boldsymbol{x(k) = A^k \hat{x} + \sum_{j=0}^{k-1} A^{k-j-1} B u(j)}$$

- Therefore, or linear system, the key for stability analysis is to understand how A^k behave as $k \to \infty$

- **Case 1:** diagonal matrix: e.g. $A = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$

- **Case 2:** diagonalizable matrix, i.e. $\exists\ T$ such that $A = TDT^{-1}$

- **Case 3:** Unfortunately, ***not all*** square matrices are diagonalizable
  - e.g.: $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ is **not diagonalizable**

- **Theorem (Internal stability)**: LTI $(\boldsymbol{A}, \boldsymbol{B})$ is **asymptotically stable** if all eigs of A satisfies $|\boldsymbol{\lambda_i}| < \mathbf{1}$

- More discussions