# ME547: Linear Systems
## State-Space Dynamic System Models

Xu Chen

University of Washington

# Why state space?

- static/memoryless system: *present* output depends only on its present input: $y(k) = f(u(k))$
- dynamic system: *present* output depends on past and its present input,
  - e.g., $y(k) = f(u(k), u(k-1), \ldots, u(k-n), \ldots)$
  - described by differential or difference equations, or have time delays
- how much information from the past is needed?

# The concept of states of a dynamic system

- the *state* $x(t)$ is the information you need at time $t$ that together with future values of the input, will let you compute future values of the output $y$
- loosely speaking:
  - ▶ the "aggregated effect of past inputs"
  - ▶ the necessary "memory" that the dynamic system keeps at each time instance

# Example



position: $y(t)$

$u = F$
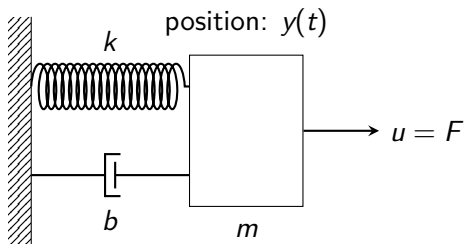
- to predict the future motion, we need to know
  - *current* position and velocity
  - *future* force
- $\Rightarrow$ states: position and velocity

# The order of a dynamic system



position: $y(t)$

$k$

$u = F$

$b$

$m$
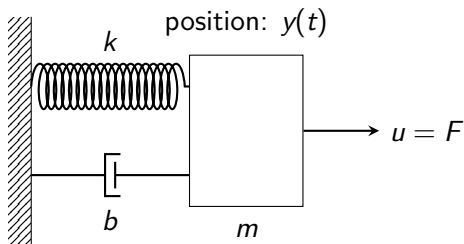
- the number, $n$ of state variables that is *necessary and sufficient* to uniquely describe the system
- for a given dynamic system,
  - ▸ the choice of state variables is *not unique*
  - ▸ however, its order $n$ is fixed
  - ▸ i.e. you need not more than $n$ but not less than $n$ state variables

# States of a discrete-time system
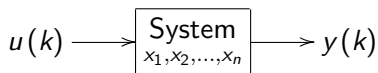
consider a discrete-time dynamic system:

$$u(k) \longrightarrow \boxed{\begin{array}{c} \text{System} \\ x_1, x_2, \dots, x_n \end{array}} \longrightarrow y(k)$$
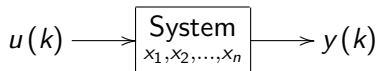
- the state at any instance $k_o$ is the minimum set of variables,

$$x_1(k_o), x_2(k_o), \cdots, x_n(k_o)$$

  that fully describe the system and its response for $k \geq k_o$ to any given set of inputs
- loosely speaking, $x_1(k_o), x_2(k_o), \cdots, x_n(k_o)$ defines the system's memory

# Discrete-time state-space description

$$u(k) \longrightarrow \boxed{\begin{array}{c} \text{System} \\ x_1, x_2, \ldots, x_n \end{array}} \longrightarrow y(k)$$

general case

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

- $u(k)$: input; $y(k)$: output
- $x(k)$: state
- $x(k+1) = f(\cdot)$: state Eq
- $y(k) = h(\cdot)$: output Eq

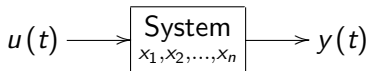linear time-invariant (LTI) case

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

- $\Sigma(A, B, C, D)$ denotes a state-space realization
- also written as $\Sigma = \left[ \begin{array}{c|c} A & B \\ \hline C & D \end{array} \right]$

# Continuous-time state-space description

$$u(t) \longrightarrow \boxed{\begin{array}{c} \text{System} \\ x_1, x_2, \ldots, x_n \end{array}} \longrightarrow y(t)$$
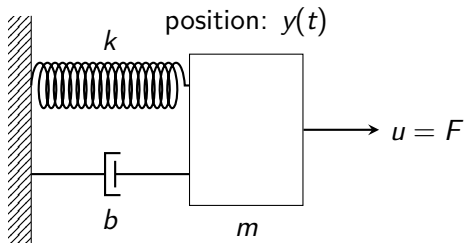
general case

$$\frac{dx(t)}{dt} = f(x(t), u(t), t)$$
$$y(t) = h(x(t), u(t), t)$$

LTI case

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$$
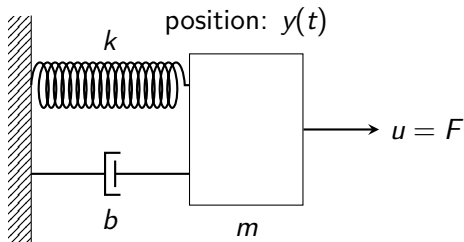$$y(t) = Cx(t) + Du(t)$$

# Example: mass-spring-damper



$$x(t) = \begin{bmatrix} \overbrace{y(t)}^{\text{mass position}} \\ \underbrace{v(t)}_{\text{mass velocity}} \end{bmatrix} \in \mathbb{R}^2$$

# Example: mass-spring-damper



$$\frac{d}{dt} \underbrace{\begin{bmatrix} y(t) \\ v(t) \end{bmatrix}}_{x(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} y(t) \\ v(t) \end{bmatrix}}_{x(t)} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{B} u(t)$$

$$y(t) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} y(t) \\ v(t) \end{bmatrix}}_{x(t)}$$

# Coding a continuous-time state-space system in MATLAB

```
A = [0,1;-3,-2];
B = [0;1];
C = [2,1];
D = 0;
sys_ss = ss(A,B,C,D)

[yout, T] = step(sys_ss);
figure, plot(T, yout)
```

# Coding a continuous-time state-space system in Python

```python
import control as co
import matplotlib.pyplot as plt
import numpy as np
A = np.array([[0,1],[-3,-2]])
B = np.array([[0],[1]])
C = np.array([2,1])
D = np.array([0])

sys_ss = co.ss(A,B,C,D)
print(sys_ss)

T,yout = co.step_response(sys_ss)

plt.figure(1,figsize = (6,4))
plt.plot(T,yout)
plt.grid(True)
plt.ylabel("y")
plt.xlabel("Time (sec)")
plt.show()
```