Lecture 2, Sep 13, 2023

System Modeling (Continuous and Discrete Time)

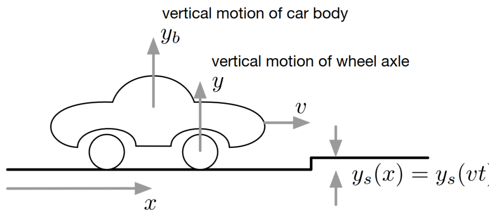

Figure 1: Example 1: Modeling the wheel motion of a car.

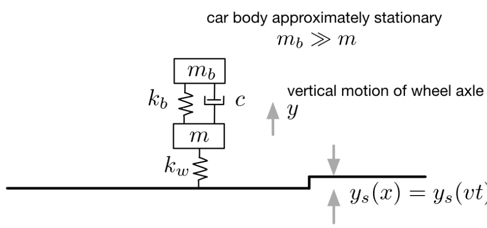

Figure 2: Modeling the example as a simple mechanical system.

- Example 1: modeling the wheel motion of a car when it encounters a bump in the road
  - Assumptions: constant velocity, 2D system model, variable step height $y_s(x)$
  - Begin with a simple mechanical system:
    * k_w is a spring representing the wheel; m is the mass of the wheel axle
    * The spring-damper system k_b and c model the shock absorber in the car
    * In addition, we assume $m_b \gg m$, so that the car itself is approximately stationary and only the wheel axle moves; we also assume the suspension is 1D and that the car is at rest in the vertical direction before we hit the bump
  - Now we can use Newton's second law to form a mathematical model:
    * $m\ddot{y} = \sum f = -c\dot{y} - k_b y - k_w(y - y_s)$
    * $m\ddot{y} + c\dot{y} + (k_b + k_w)y = m\ddot{y} + c\dot{y} + ky = k_w y(s) = u(t)$
      - This is a second order, linear, nonhomogeneous, time-invariant system
      - The initial conditions are $y(0) = \dot{y}(0) = 0$ and we wish to find $y, \dot{y}, \ddot{y}$ for $t \geq 0$
    * Now we need to represent it in a standard form
      - $\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{c}{m} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} u$
      - This is now in standard form: $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u, \boldsymbol{x}(0) = \begin{bmatrix} y(0) \\ \dot{y}(0 \end{bmatrix}$
      - This form corresponds to the following simulation block diagram:

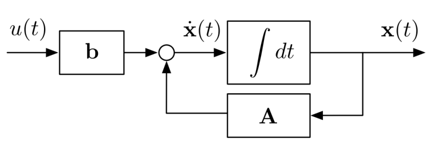

Figure 3: Simulation block diagram of a standard linear system.

- Example 2: modelling how a drone reacts to given motor inputs
  - Assumptions: horizontal motion is stabilized

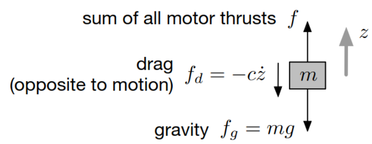

Figure 4: Simple mechanical system for Example 2.

- Applying Newton's laws:
  * $m\ddot{z} = -mg - c\dot{z} + f$
  * We again have a second order, linear, nonhomogeneous, time-invariant system
- In standard form: $\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\dfrac{c}{m} \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} u$
- But how do we actually perform the simulation?

Summary

In general for any linear system we have: a set of inputs $\boldsymbol{u}(t)$ (which we partially control); a set of outputs $\boldsymbol{y}(t)$ which we can measure; and states $\boldsymbol{x}(t)$ that are internal to the system which we cannot directly manipulate or measure.
To model a linear system in continuous time:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \boldsymbol{x}(0) = \boldsymbol{x}_0$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ are *state matrices*.
To model a nonlinear system in continuous time:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \boldsymbol{x}(0) = \boldsymbol{x}_0$$
$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

- If we have n states, m inputs and p outputs, then $\boldsymbol{A}$ is $n \times n$, $\boldsymbol{B}$ is $n \times m$, $\boldsymbol{C}$ is $p \times n$ and $\boldsymbol{D}$ is $p \times m$
- In practice, we often need to represent things in discrete time, since the computers running simulations are discrete
- To represent things in discrete time, we replace continuous signals with a sequence of regular samples at $t_k = kh$
  - $t_k = kh$ is the *sampling time*
  - $f_s = \dfrac{1}{h}$ is the *sampling frequency*
- So how do we convert our continuous model to a discrete one?
  - Recall that the solution for $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}$ is $\boldsymbol{x} = e^{\boldsymbol{A}t}\boldsymbol{x}_0$
  - Over a short time interval $t_k \leq t \leq t_{k+1}$ the solution evolves as:
    * $\boldsymbol{x}(t) = e^{\boldsymbol{A}(t-t_k)}\boldsymbol{x}(t_k) + \int_{t_k}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\,\mathrm{d}\tau$
      $= e^{\boldsymbol{A}(t-t_k)}\boldsymbol{x}(t_k) + \int_{t_k}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\,\mathrm{d}\tau\boldsymbol{u}(t_k)$
      $= \boldsymbol{\Phi}(t, t_k)\boldsymbol{x}(t_k) + \Gamma(t, t_k)\boldsymbol{u}(t_k)$
      - Note that we have assumed $\boldsymbol{u}(\tau) = \boldsymbol{u}(t_k)$ (i.e. $\boldsymbol{u}$ stays constant over the timestep), which is referred to as a *zero-order hold*
    * $\boldsymbol{x}(t_{k+1}) = \boldsymbol{\Phi}(t_{k+1}, t_k)\boldsymbol{x}(t_k) + \Gamma(\boldsymbol{t}_{k+1}, t_k)\boldsymbol{u}(t_k) = \boldsymbol{A}_d\boldsymbol{x}(t_k) + \boldsymbol{B}_d\boldsymbol{u}(t_k)$
      - We have discretized the system
- We now have difference equations for the system (h is the sampling period):
  - $\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k$

- $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k$
- $\boldsymbol{A}_d = \boldsymbol{\Phi}(t_{k+1}, t_k) = e^{\boldsymbol{A}h}$
- $\boldsymbol{B}_d = \Gamma(t_{k+1}, t_k) = \int_0^h e^{\boldsymbol{A}\tau'}\,\mathrm{d}\tau' \boldsymbol{B}$

- To solve for $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$:
  - Note that $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(t) = \boldsymbol{\Phi}(t)\boldsymbol{A}$ and $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Gamma}(t) = \boldsymbol{\Phi}(t)\boldsymbol{B}$
  - Using this we have: $\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{\Phi}(t) & \boldsymbol{\Gamma}(t) \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}(t) & \boldsymbol{\Gamma}(t) \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$
    - Now we can use another matrix exponential to solve this
  - $\begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_d \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \exp\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} h\right)$

Note

The matrix exponential can be calculated with the Matlab function `expm()` or scipy function `scipy.linalg.expm()`. We can also manually do a series expansion of the matrix exponential function. Alternatively, `c2d()` in Matlab or `control.matlab.c2d()` can be used to do the same conversion (which computes the matrix exponentials internally).

- We can now recursively apply the difference equations to propagate the state:
  - $\boldsymbol{x}_1 = \boldsymbol{A}_d\boldsymbol{x}_0 + \boldsymbol{B}_d\boldsymbol{u}_0$
  - $\boldsymbol{x}_2 = \boldsymbol{A}_d(\boldsymbol{A}_d\boldsymbol{x}_0 + \boldsymbol{B}_d\boldsymbol{u}_0) + \boldsymbol{B}_d\boldsymbol{u}_1$
  - $\boldsymbol{x}_3 = \boldsymbol{A}_d(\boldsymbol{A}_d^2\boldsymbol{x}_0 + \boldsymbol{A}_d\boldsymbol{B}_d\boldsymbol{u}_0 + \boldsymbol{B}_d\boldsymbol{u}_1) + \boldsymbol{B}_d\boldsymbol{u}_2$ and so on
  - We can stack all these together: $\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = \boldsymbol{F}\begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{bmatrix} + \boldsymbol{F}_0\boldsymbol{x}_0$

Definition

A system is *reachable* or *controllable* if and only if

$$\operatorname{rank}\begin{bmatrix} \boldsymbol{A}_d^{N-1}\boldsymbol{B}_d & \boldsymbol{A}_d^{N-2}\boldsymbol{B}_d & \cdots & \boldsymbol{B}_d \end{bmatrix} = N$$

where N is the dimension of the state $\boldsymbol{x}_k$. Physically this means that there is a given sequence of control inputs to reach any state.

Definition

A system is *observable* if and only if

$$\operatorname{rank}\begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A}_d \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}_d^{N-1} \end{bmatrix} = N$$

Physically this means that given some sequence of outputs, we can derive the system state.

- Example: consider the system: $\dot{x}(t) = u(t), y(t) = x(t), x(0) = x_0$; find the difference equations assuming a zero-order hold at the input and sampled output, with sampling period h
  - Note that we have $\boldsymbol{A} = 0$ and $\boldsymbol{B} = 1$
  - For $t_k \leq t \leq t_{k+1}$:

* $x(t) = x(t_k) + \int_{t_k}^{t} u(\tau)\,\mathrm{d}\tau$
  $= x(t_k) + \int_{t_k}^{t} 1\,\mathrm{d}\tau u(t_k)$
  $= x(t_k) + (t - t_k)u(t_k)$
* Applying this at $t = t_{k+1} = t_k + h$ we get $x_{k+1} = x_k + hu_k$

- Example: $\ddot{x} = u$
  - State: $\boldsymbol{z} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$
  - Continuous time: $\dot{\boldsymbol{z}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}u$
  - $\begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_d \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \exp\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} h\right) = \exp\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} h\right) = \exp(\boldsymbol{E}h)$
  - Notice that $\boldsymbol{E}^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $\boldsymbol{E}^3 = \boldsymbol{0}$ so $\boldsymbol{E}$ is nilpotent
  - Using the series expansion: $\exp(\boldsymbol{E}h) = \boldsymbol{I} + \boldsymbol{E}h + \frac{1}{2}\boldsymbol{E}^2h^2 = \begin{bmatrix} 1 & h & \frac{1}{2}h^2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}$
  - Therefore $\boldsymbol{A}_d = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \boldsymbol{B}_d = \begin{bmatrix} \frac{1}{2}h^2 \\ h \end{bmatrix}$
  - $\boldsymbol{z}_{k+1} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \boldsymbol{z}_k + \begin{bmatrix} \frac{1}{2}h^2 \\ h \end{bmatrix} u_k$
    * Notice that if we substitute the definition of $\boldsymbol{z}$, we get the simple kinematic equations $x_{k+1} = x_k + hv_k + \frac{1}{2}h^2u_k, v_{k+1} = v_k + hu_k$
    * Assuming a zero-order hold, we can see that this is exact

Summary

Given some linear continuous system given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \boldsymbol{x}(0) = \boldsymbol{x}_0$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$

we can discretize it to find discrete state matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ so that the system can be equivalently modelled discretely by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k$$

which is an exact model with the assumption of a zero-order hold on the input. The matrices $\boldsymbol{A}_d, \boldsymbol{B}_d$ can be found by the matrix exponential

$$\begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_d \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \exp\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} h\right)$$

where h is the size of each discrete time step.