MECH 6v29.002 – Model Predictive Control

Tuesday and Thursday 8:30 – 9:45am

L4 – Dynamic Systems

Outline

- State-space models
  - Nonlinear / Linear
  - Continuous / Discrete
  - Future State Prediction
- Discretization
- Equilibrium
- Stability
- Controllability

Nonlinear State Space Model

- The performance of MPC is highly dependent on the model
- When developing a model from first-principles (e.g. applying conservation of energy or mass, Kirchhoff's Laws, etc.) it is likely that your model will be nonlinear
- Most general state-space model

$$\dot{x} = \frac{dx}{dt} = f\left(x,u,t\right)$$

$$y = h\left(x,u,t\right)$$

$$x\left(t_0\right) = x_0$$

$$x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ y \in \mathbb{R}^p,\ t \in \mathbb{R},$$

- Wish to determine the solution of this differential equation for time greater than t_0 based on the initial condition x_0 and the input trajectory

Time-varying Linear Model

- While some MPC formulations utilize a nonlinear model, most rely on a linearized model
- Most general linear state-space model

Linear Time Varying (LTV) model

$$\dot{x} = A(t)x + B(t)u$$

$$y = C(t)x + \cancel{D(t)u}$$

$$x(0) = x_0$$

Often assume no feedthrough

Often assume starting at $t_0 = 0$

$A \in \mathbb{R}^{n \times n}$ - State transition matrix

$B \in \mathbb{R}^{n \times m}$ - Input matrix

$C \in \mathbb{R}^{p \times n}$ - Output matrix

$D \in \mathbb{R}^{p \times m}$ - Feedthrough matrix

Time-invariant Linear Model

- Most commonly used state-space model

Linear Time-Invariant (LTI) model

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$
$$x(0) = x_0$$

- Makes solution and analysis much easier
- Now we can solve for $x(t)$ as a function of x_0 and $u(t)$

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

$e^{At} \in \mathbb{R}^{n\times n}$ - matrix exponential

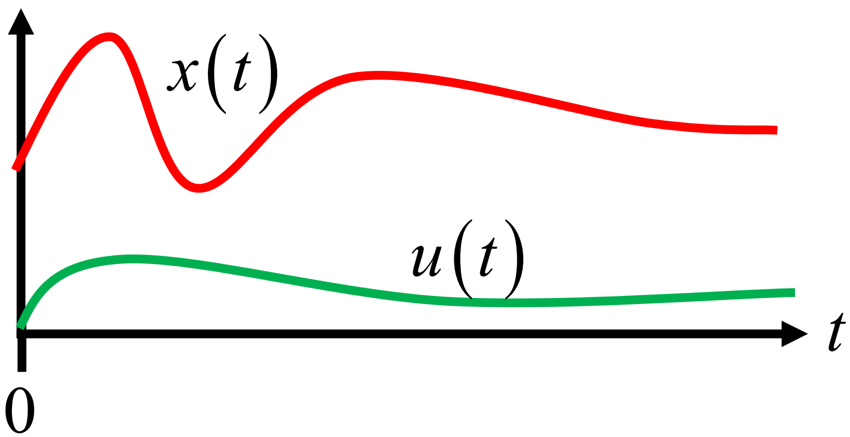
$\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$ - convolution integral

solution depends on entire $u(t)$, $t \geq 0$, and the effect of $u(t)$ is weighted by powers of the A matrix

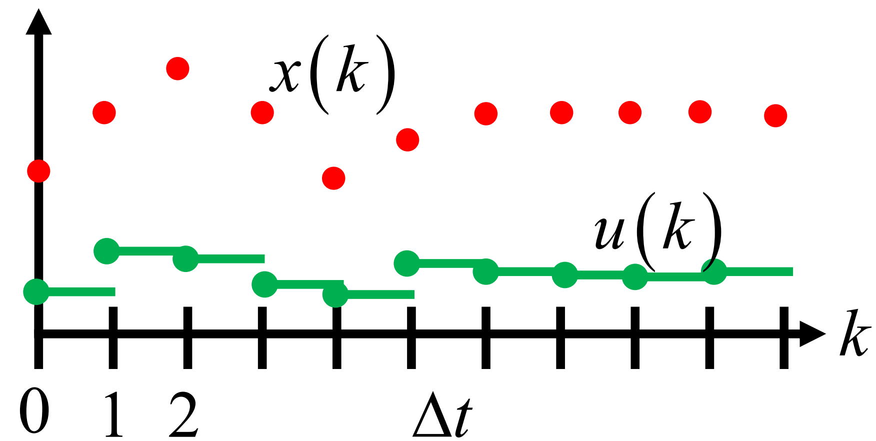
Discrete-time Models

- While some MPC formulations use a continuous-time state space model, most use a discrete-time model (almost exclusively used in practice)

Continuous-time



Discrete-time



$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$
$$x(0) = x_0$$

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$
$$x(0) = x_0$$

Not the same matrices!

$$t = k\Delta t$$

Discrete-time Models (cont.)

- Equivalent Notation:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$
$$x(0) = x_0$$

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$
$$x_0 \text{ given}$$

$$x^+ = Ax + Bu$$
$$y = Cx + Du$$
$$x_0 \text{ given}$$

<u>Continuous-time</u>

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

<u>Discrete-time</u>

$$x(k) = A^k x_0 + \sum_{j=0}^{k-1} A^{k-j-1}Bu(j)$$

Already easier to deal with!

Future State Prediction

- Discrete-time model: $x_{k+1} = Ax_k + Bu_k,\ x_0$ given
- Apply recursively:

$$x_1 = Ax_0 + Bu_0$$

$$\begin{aligned} x_2 &= Ax_1 + Bu_1 \\ &= A\left(Ax_0 + Bu_0\right) + Bu_1 \\ &= A^2x_0 + ABu_0 + Bu_1 \end{aligned}$$

$$\begin{aligned} x_3 &= Ax_2 + Bu_2 \\ &= A\left(A^2x_0 + ABu_0 + Bu_1\right) + Bu_2 \\ &= A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2 \end{aligned}$$

$$\Downarrow$$

$$x_N = A^N x_0 + A^{N-1}Bu_0 + A^{N-2}Bu_1 + \ldots + ABu_{N-2} + Bu_{N-1}$$

$$x_N = A^N x_0 + \sum_{j=0}^{N-1} A^{N-j-1}Bu_j$$

Future State Prediction (Vector Form)

- We can now predict all future states based on current state and future input trajectory

$$x_N = A^N x_0 + \sum_{j=0}^{N-1} A^{N-j-1} B u_j$$

- Assemble this in vector form

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^{n \cdot N} \qquad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^{m \cdot N}$$

- Derive: $X = P x_0 + HU$

Future State Prediction (Vector Form)

- Already saw that:

$$x_1 = Ax_0 + Bu_0$$
$$x_2 = A^2 x_0 + ABu_0 + Bu_1$$
$$x_3 = A^3 x_0 + A^2 Bu_0 + ABu_1 + Bu_2$$

- Want:

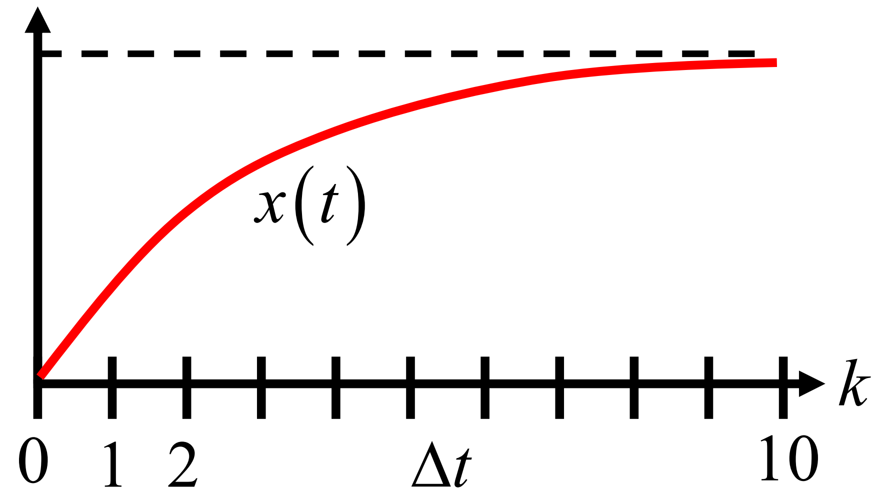
$$x_N = A^N x_0 + \sum_{j=0}^{N-1} A^{N-j-1} Bu_j$$

$$X = Px_0 + HU$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{P} x_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{H} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

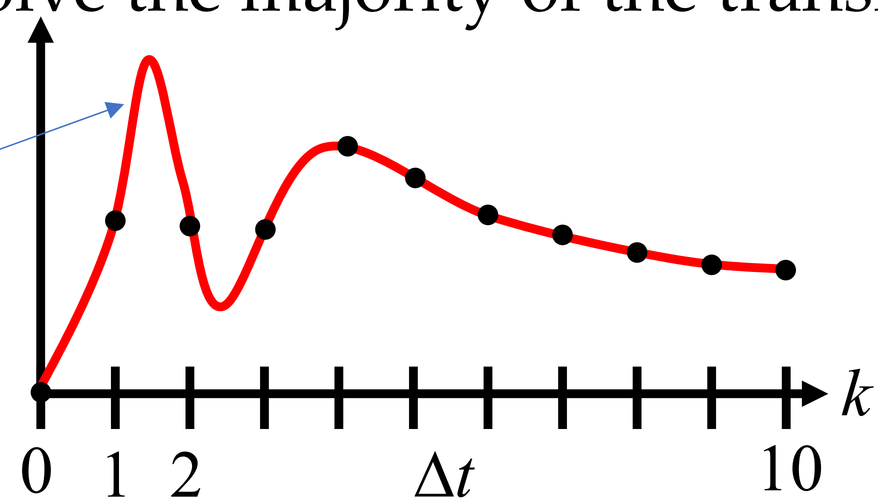
Discretization

- How to choose Δt?



- Typically choose ~10 samples within the transient response of the dynamic system
- Want to resolve the majority of the transients

Discrete model unaware of key transient



Discretization (cont.)

- How to discretize?
- Nonlinear models
  - Forward Euler approximation (most commonly used)

$$\dot{x} \approx \frac{x_{k+1} - x_k}{\Delta t}$$

$$\dot{x} = f\left(x, u, t\right) \quad \Rightarrow \quad x_{k+1} = x_k + \Delta t\, f\left(x_k, u_k, \Delta t k\right)$$

  - May be inaccurate (or unstable) depending on the nonlinearities and dynamics
  - More sophisticated methods available [1]

[1] https://en.wikipedia.org/wiki/List_of_Runge%E2%80%93Kutta_methods

Discretization (cont.)

- How to discretize?
- Linear models
  - Can use the same approximation

$$\dot{x} \approx \frac{x_{k+1} - x_k}{\Delta t}$$

$$\dot{x} = A_c x + B_c u \quad \Rightarrow \quad \begin{aligned} x_{k+1} &= (I + \Delta t\, A_c) x_k + (\Delta t\, B_c) u_k \\ &= A_d x_k + B_d u_k \end{aligned}$$

Discretization (cont.)

- How to discretize?
- Linear models
  - Or, you can get an exact model (no prediction error) assuming zero-order hold on inputs
    - Already saw the continuous-time solution

$$x(t) = e^{A_c t} x_0 + \int_0^t e^{A_c (t-\tau)} B_c u(\tau) d\tau$$

    - Let $t = \Delta t$

$$x(\Delta t) = e^{A_c \Delta t} x_0 + \int_0^{\Delta t} e^{A_c (\Delta t-\tau)} B_c u_0 d\tau$$

Constant over time interval

$$x(\Delta t) = e^{A_c \Delta t} x_0 + \int_0^{\Delta t} e^{A_c (\Delta t-\tau)} d\tau B_c u_0$$

Bring outside the integral

Discretization (cont.)

- How to discretize?
- Linear models
  - Or, you can get an exact model (no prediction error) assuming zero-order hold on inputs

$$x(\Delta t) = e^{A_c \Delta t} x_0 + \int_0^{\Delta t} e^{A_c(\Delta t - \tau)} d\tau B_c u_0$$

$$A_d = e^{A_c \Delta t} \qquad B_d = \int_0^{\Delta t} e^{A_c(\Delta t - \tau)} d\tau B_c$$

    - If A_c is nonsingular

$$\int_0^{\Delta t} e^{A_c(\Delta t - \tau)} d\tau = A_c^{-1}(A_d - I)$$

$$B_d = A_c^{-1}(A_d - I) B_c$$

  - Generalize to: $x_{k+1} = A_d x_k + B_d u_k$ c2d command in Matlab

Equilibrium

- In general, equilibrium corresponds to

$$\dot{x} = 0 \quad \Rightarrow \quad 0 = f(x,u,t)$$

- Linear system (continuous time)

$$\dot{x} = 0 = A_c x_{ss} + B_c u_{ss}$$

  - Can have multiple equilibrium depending on the value of the inputs (each unique input can drive the system to a different steady state)
  - Can solve for steady states if state matrix is invertible

$$x_{ss} = -A_c^{-1} B_c u_{ss}$$

- Linear system (discrete time)

$$x_{k+1} = x_{ss} = A_d x_{ss} + B_d u_{ss}$$

$$x_{ss} = (I - A_d)^{-1} B_d u_{ss}$$

Equilibrium (cont.)

- Without loss of generality (w.l.o.g), we will typically assume that the control objective is to drive the system to the origin
  - i.e. the only equilibrium we care about is

$$x_{ss} = 0,\ u_{ss} = 0$$

- What if this is not the case?

$$x_{ss} = \overline{x},\ u_{ss} = \overline{u}$$

- Then do a change of variables:

$$x = \overline{x} + \Delta x,\ u = \overline{u} + \Delta u$$

$$\Delta x = x - \overline{x},\ \Delta u = u - \overline{u} \qquad \Delta x(0) = x(0) - \overline{x}$$

- Continuous-time case (similar process for discrete-time)

Constant

$$\dot{x} = \frac{d\left(\overline{x} + \Delta x\right)}{dt} = \Delta \dot{x}$$

$$\dot{x} = A_c x + B_c u$$

$$\Delta \dot{x} = A_c\left(\overline{x} + \Delta x\right) + B_c\left(\overline{u} + \Delta u\right)$$

$$\Delta \dot{x} = A_c \Delta x + B_c \Delta u + \cancel{A\overline{x} + B\overline{u}}$$

$$\Delta \dot{x} = A_c \Delta x + B_c \Delta u$$

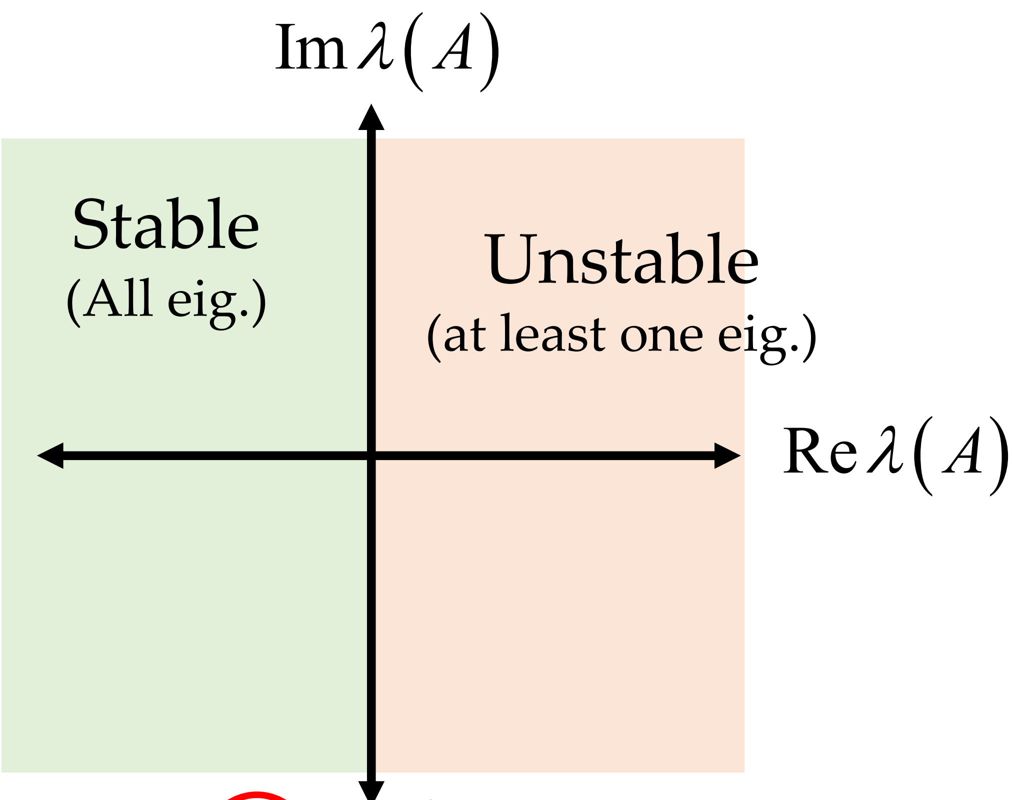
Equals zero by definition of steady state equilibrium

Stability

- Determined based on the eigenvalues of the state transition matrix

Continuous-time

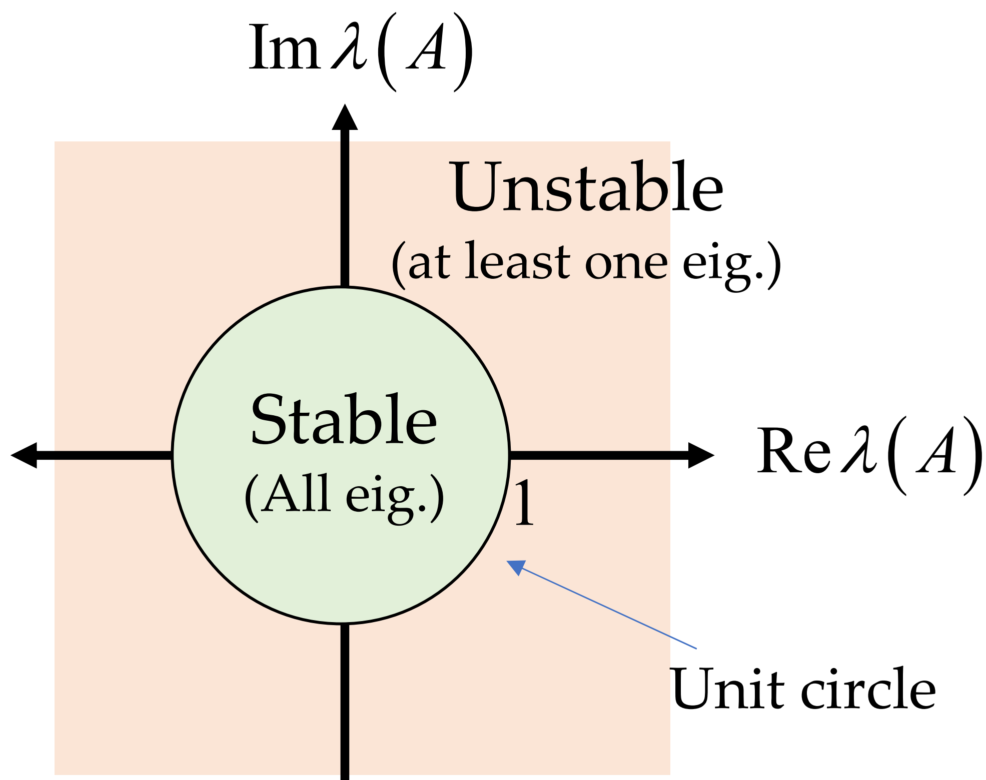
$$\dot{x} = Ax + Bu$$



$$x(t) = e^{A_c t} x_0 + \int_0^t e^{A_c(t-\tau)} B_c u(\tau) d\tau$$

Discrete-time

$$x_{k+1} = Ax_k + Bu_k$$



$$x_N = A^N x_0 + \sum_{j=0}^{N-1} A^{N-j-1} B u_j$$

Controllability

- Same condition for continuous- and discrete-time systems
- Compute controllability matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$$

- System is controllable if and only if

$$\text{rank}(\mathcal{C}) = n \qquad\qquad x \in \mathbb{R}^n$$

- Conceptual idea is clear from discrete-time model

$$x_N = A^N x_0 + A^{N-1} B u_0 + A^{N-2} B u_1 + \ldots + AB u_{N-2} + B u_{N-1}$$

- Let $N = n$, and hold input constant

$$x_n = A^n x_0 + \begin{bmatrix} A^{n-1}B & \cdots & AB & B \end{bmatrix} u_0$$

$$x_n = A^n x_0 + \mathcal{C} u_0$$

- Any state transition is possible if C is full rank