CDT
Autonomous and Intelligent Machines & Systems

Introduction to Modern Control
MT 2014

Alessandro Abate
Lecture 2

- First-order ordinary differential equations (ODE)
- Solution of a linear ODE
- Hints to nonlinear case and linearization procedure
- From continuous to discrete time: simulation of an ODE
- From state-space models to frequency domain: Transfer functions
Linear Ordinary Differential Equations (Linear ODE)

- Recall dynamical model for spring-damper system:

\[ \ddot{q}(t) = \frac{1}{m} (-c(\dot{q}(t)) - kq(t) + u(t)) \]

- We’ve seen it can be formulated as:

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{1}{m} (-c(x_2(t)) - kx_1(t) + u(t))
\end{align*}
\]

- Introduce linear approximation of nonlinear term (e.g., arctan):

\[ c(x_2(t)) \approx cx_2(t) \]

- Consider output equation: \[ y(t) = x_1(t) \]
• Consider state vector \[
\begin{bmatrix}
  x_1(t) \\
  x_2(t)
\end{bmatrix}
\]

• Obtain state-space representation as linear ODE:

\[
\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t)
\]

\[
y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}
\]

• if \( u(t) \equiv 0, \forall t \geq 0 \), ODE is said to be autonomous

• model is time-invariant: the behavior does not change under time-shifts (we thus usually pick initial time to be equal to 0)
General State-Space Model for Linear ODE

- Linear ODE in standard state-space form:

\[
\frac{d}{dt} x(t) = Ax(t) + Bu(t), \quad x(0) = x_0
\]

\[
y(t) = Cx(t) + Du(t)
\]

- \( x(t) \in \mathbb{R}^n \), state variable

- \( u(t) \in \mathbb{R}^m \), input variable

- \( y(t) \in \mathbb{R}^l \), output variable

- Dimensions of \( x, u, y \rightarrow \) dimensions \( (A, B, C, D) \)
General Model for Linear ODE: comments

- We shall often consider SISO systems \((m = l = 1)\)
- Each integrator output is state variable
- One initial condition needed for each state variable
- `ss` structures “Control System Toolbox” of MATLAB
- Use of Simulink to represent these models
Block Diagram for Single Model (linear ODE)

\[
\frac{d}{dt} x(t) = Ax(t) + Bu(t), \quad x(0) = x_0
\]
\[
y(t) = Cx(t) + Du(t)
\]
Block Diagrams for Models Interconnection

• in series

\[
\begin{align*}
\dot{x}_1 &= f(x_1, u_1) \\
y_1 &= h(x_1, u_1) \\
\dot{x}_2 &= f(x_2, u_2) \\
y_2 &= h(x_2, u_2)
\end{align*}
\]

• in parallel

\[
\begin{align*}
\dot{x}_1 &= f(x_1, u) \\
y_1 &= h(x_1, u) \\
\dot{x}_2 &= f(x_2, u) \\
y_2 &= h(x_2, u) + u
\end{align*}
\]

• And of course in feedback (as discussed in previous lecture)
From high-order ODE to system of first order ODE

• Leitmotif of previous examples: exploit theory of first-order ODE

• Consider n-th order system:

\[
\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \ldots + a_n y = u
\]

• Introduce variables: \( x_1 = \frac{d^{n-1} y}{dt^{n-1}}, x_2 = \frac{d^{n-2} y}{dt^{n-2}}, \ldots, x_n = y \)

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_{n-1} \\
  x_n
\end{bmatrix}
= \begin{bmatrix}
  -a_1 x_1 - a_2 x_2 - \ldots - a_n x_n \\
  \vdots \\
  x_1 \\
  \vdots \\
  x_{n-2} \\
  x_{n-1}
\end{bmatrix}
+ \begin{bmatrix}
  u \\
  0 \\
  \vdots \\
  0 \\
  0
\end{bmatrix}
\]

• n-dim system of first-order ODE – non-uniqueness of construction
More on Linearization: From Nonlinear to Linear ODE

- Recall model for Inverted Pendulum (Balance System, Segway)
More on Linearization: From Nonlinear to Linear ODE

- Overall state $q = (p, \theta)$, input $u = F$, output $y = (p, \theta)$

- Can write dynamics as:

\[
M(q)\ddot{q} + K(q, \dot{q}) = Bu
\]

\[
\begin{bmatrix}
(M + m) & -ml \cos \theta \\
-ml \cos \theta & ml^2
\end{bmatrix}
\begin{bmatrix}
\ddot{p} \\
\ddot{\theta}
\end{bmatrix}
+ \begin{bmatrix}
ml \sin \theta \dot{\theta}^2 \\
-mgl \sin \theta
\end{bmatrix}
= \begin{bmatrix}
F \\
0
\end{bmatrix}
\]

- It is often of interest to work around (equilibrium) point

\[
\theta \approx 0 \Rightarrow \sin \theta \approx \theta, \cos \theta \approx 1, \dot{\theta}^2 = o(\dot{\theta})
\]
• This approximation yields

\[
\begin{bmatrix}
(M + m) & -ml \\
-ml & ml^2
\end{bmatrix}
\begin{bmatrix}
\ddot{p} \\
\ddot{\theta}
\end{bmatrix} +
\begin{bmatrix}
0 \\
-mgl\theta
\end{bmatrix} =
\begin{bmatrix}
F \\
0
\end{bmatrix}
\]

• Express as first order ODE (considering \([q, \dot{q}]^T = [p, \theta, \dot{p}, \dot{\theta}]^T\)):

\[
\frac{d}{dt}
\begin{bmatrix}
p \\
\theta \\
\dot{p} \\
\dot{\theta}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & mg/M & 0 & 0 \\
0 & (M + m)g/ML & 0 & 0
\end{bmatrix}
\begin{bmatrix}
p \\
\theta \\
\dot{p} \\
\dot{\theta}
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
1/M \\
1/ML
\end{bmatrix} u
\]

\[
y =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix} x
\]

• (A more formal introduction to linearization later)
Solution to Autonomous ODE

- Let us focus on the “dynamical” (autonomous) part of the ODE:

\[
\frac{d}{dt} x(t) = Ax(t), \quad x(0) = x_0
\]

- has solution

\[
x(t) = e^{At} x_0, \quad (1)
\]

- where

\[
e^{At} := I + At + A^2 \frac{t^2}{2!} + A^3 \frac{t^3}{3!} + \ldots
\]
• Matrix exponential:

• Use of expm function in MATLAB

• Proof of (1): check if putative solution satisfies ODE, and i.c.:

\[
\frac{d}{dt} x(t) = \frac{d}{dt} \left( e^{At} x_0 \right) = A(\ldots), \quad x(0) = x_0 \quad \square
\]

• Assume \( A \in \mathbb{R}^{n,n} \) is diagonal, \( A = \begin{pmatrix} \lambda_1 & 0 & 0 & \ldots & 0 \\ 0 & \lambda_2 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \lambda_n \end{pmatrix} \)
• Then

\[ \forall i = 1, \ldots, n, x_i(t) = e^{\lambda_i t} x_{0,i} \Rightarrow x(t) = \begin{pmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & & \vdots \\ \cdots & & \ddots & 0 \\ 0 & \cdots & 0 & e^{\lambda_n t} \end{pmatrix} x_0 \]

• What if A is not diagonal?

• State coordinate change via similarity transformation

• \( z = T x \) (\( T \) invertible \( n \times n \)) \( \Rightarrow \dot{z} = T A T^{-1} z \)

• “Ideal situation:” \( T A T^{-1} \) is diagonal

• Procedure: study \( z \)-model, to derive properties for \( x \)-model

• Condition: \( A \) diagonalizable – in MATLAB, command \( \text{eig} \)
– Sufficient condition: $A$ has distinct eigenvalues

- What if $A$ is not diagonalizable? (e.g.: $A$ has eigenvalues with multiplicity)

- For any $A$, existence of a *Jordan form* – in MATLAB, command `jordan`
Transformation of state equations
(state coordinate change)

• \( z = Tx \) (\( T \) invertible \( n \times n \)) similarity transformation

• State space representation

\[
\dot{x} = Ax + Bu, \quad y = Cx + Du
\]

becomes

\[
\dot{z} = TAT^{-1}z + TBu, \quad y = CT^{-1}z + Du
\]

• (we will see shortly) two systems related by a similarity transformation have same I/O response

\[
CT^{-1}e^{TAT^{-1}t}TBu(t) = Ce^{At}Bu(t)
\]
Eigenvalues and Dynamical Modes

- Importance of the eigenvalues of state matrix $A$: they are the solution of the characteristic polynomial:

$$\det (\lambda I - A) = 0$$

- in MATLAB `poly(A)`
Solution to ODE with Input

- The controlled ODE

\[
\frac{dx(t)}{dt} = Ax(t) + Bu(t), \quad x(0) = x_0
\]

- (assuming \( A \) is invertible) has solution

\[
x(t) = e^{At} \left[ x_0 + \int_0^t e^{-A\lambda} Bu(\lambda) d\lambda \right] \\
= e^{At} x_0 + \int_0^t e^{A(t-\lambda)} Bu(\lambda) d\lambda
\]

convolution of \( u(t) \) and \( e^{At}B \)
• Sum of response to initial condition and to input signal: **superposition** principle, which comes from model linearity

• Additionally, the output part:

\[ y(t) = Cx(t) + Du(t) \]

• has solution

\[ y(t) = Ce^{At}x_0 + \int_0^t Ce^{A(t-\lambda)}Bu(\lambda)d\lambda + Du(t) \]

• Extension to non-linear, time-varying case: use of **state-transition matrix** \( \Phi(t) \) (not in this course)
Impulse Response

- Assume for simplicity $x_0 = 0$. Impulse on (1-d) input $u(t) = \delta(t)$:

$$y(t) = C \int_0^t e^{A(t-\lambda)} B \delta(\lambda) d\lambda + D \delta(t)$$

$$= Ce^{At} B + D \delta(t)$$

- Identification of system dynamics from data (recall end of Lec. 1):

- Actually, LTI systems are completely characterized by impulse response
Step Response - Transience and Steady State

- Step input (Heaviside function):

\[ u(t) = \begin{cases} 
0, & t = 0 \\
1, & t > 0 
\end{cases} \]

- Assuming again \( x_0 = 0 \), we obtain

\[ y(t) = C \int_0^t e^{A(t-\lambda)} B d\lambda + D = CA^{-1}e^{At}B - CA^{-1}B + D, \quad t > 0 \]

Presence of *transient part*, and *steady-state term*
• Define controller “specifications” based on profile of step response

• Focus on “dominant mode” in model dynamics

![Graph showing input/output response with key performance properties highlighted: rise time $T_r$, overshoot $M_p$, settling time $T_s$, steady-state value $y_{ss}$, and time [sec] axis from 0 to 30.](image)
A Note on Non-Linear ODE

- Features:
  - non-linear dynamics and observations
  - non-autonomous
  - time-varying

\[
\frac{dx(t)}{dt} = f(t, x(t), u(t)), \quad x(t_0) = x_0
\]

\[
y(t) = g(t, x(t), u(t))
\]

- existence of “non-standard” cases (non-existence or non-uniqueness of solution)

- will resort to local *linearization*
Integration of ODE: Discrete-Time Systems

• Consider a general ODE of the form

\[ \dot{x}(t) = f(x(t)), \quad x(0) = x_0 \]

• Introduce integration time-step \( h, 0 < h < \infty \)

• Approximate time-derivative \( \dot{x}(t) \) at time \( t \geq 0 \) as

\[ \dot{x}(t) \approx \frac{x(t + h) - x(t)}{(t + h) - t} \]

• This yields

\[ x(t + h) \approx x(t) + hf(x(t)) \]
• The above is a forward-Euler first-order integration scheme
  Obtain ordinary difference equation describing discrete-time system

• WARNING: Property disruption through sampling
  Can employ more complex schemes (higher-order, variable step-size)

Integration of ODE in MATLAB performed with functions ode23, ode45

Transfer Functions and Frequency Domain

- Recall the block-diagram representation of dynamical models. *Transfer Functions* compactly represent the linear relationship between inputs and outputs of a system. As discussed, they can be easily constructed from measured data.

- Start with a state-space representation $\dot{x} = Ax + Bu, \ y = Cx + Du$

- Consider inputs of the form $u(t) = e^{st}, s = \sigma + i\omega \Rightarrow u(t) = e^{\sigma t}(\cos \omega t + i\sin \omega t)$

- Solution is $x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Be^{s\tau}d\tau = e^{At}x_0 + e^{At} \int_0^t e^{(sI-A)\tau} B d\tau$

- If $s \neq \lambda(A) \Rightarrow x(t) = e^{At}x_0 + e^{At}(sI - A)^{-1} (e^{(sI-A)t} - I) B$
• Thus, \( y(t) = Ce^{At} \left( x_0 - (sI - A)^{-1}B \right) + \left( C(sI - A)^{-1}B + D \right) e^{st} \)

• Consider steady-state term: let us call

\[
G_{yu}(s) = C(sI - A)^{-1}B + D
\]

the *Transfer Function* of the system:

\[
y(t) = G_{yu}(s)u(t), \quad u(t) = e^{st}
\]

• Notice that if \( s = \lambda_i(A) \), \( G_{yu} \) presents a singularity
Transfer Functions, example

- Consider canonical 2nd order system: \( \ddot{q} + 2\zeta\omega_o \dot{q} + \omega_o^2 q = k\omega_o^2 u, \ y = q \)

- Can be re-written as:

\[
\frac{dx}{dt} = \left[ \begin{array}{cc} 0 & \omega_o \\ -\omega_o & -2\zeta\omega_o \end{array} \right] x + \left[ \begin{array}{c} 0 \\ k\omega_o \end{array} \right] u, \quad y = [1 \ 0]x
\]

- Assume \( \zeta, \omega_o > 0 \rightarrow \) model is stable

- Steady-state response to \( u(t) = e^{st} \) is \( C(sI - A)^{-1}B \):

\[
[1 \ 0] \left( sI - \left[ \begin{array}{cc} 0 & \omega_o \\ -\omega_o & -2\zeta\omega_o \end{array} \right] \right)^{-1} \left[ \begin{array}{c} 0 \\ k\omega_o \end{array} \right] = \frac{k\omega_o^2}{s^2 + 2\zeta\omega_os + \omega_o^2}
\]
Transfer Functions, alternative derivation

- Consider integrable function \( f(t), f : \mathbb{R}^+ \rightarrow \mathbb{R} \) and s.t. \( f(t) < e^{s_0 t} \)

- Laplace transform \( \mathcal{L} \) yields an \( F = \mathcal{L}f : \mathbb{C} \rightarrow \mathbb{C} \), defined by

\[
F(s) = \int_0^\infty e^{-s\tau} f(\tau) d\tau, \quad \Re(s) > s_0
\]

- Consider the system \( \dot{x} = Ax + Bu, \quad y = Cx + Du \)

- Take Laplace transforms (with zero initial conditions):

\[
sX(s) = AX(s) + BU(s), \quad Y(s) = CX(s) + DU(s)
\]

- Substitute for \( X(s) \) to obtain:

\[
Y(s) = (C(sI - A)^{-1}B + D) U(s) = G_{yu}(s)U(s)
\]