Advanced Control Systems

Lecture 2: Trajectory, Motion and Equilibrium

Aidan O. T. Hogg

EECS, Queen Mary University of London

a.hogg@qmul.ac.uk

Spring 2025

Introduction to Linear Systems

Standard Equations:

$$\sigma x = Ax + Bu$$
$$y = Cx + Du$$

Components:

- x: *State* (internal variable of the system)
- u: *Input* (external action applied to the system)
- y: *Output* (measurable response of the system)

Continuous vs. Discrete-Time Systems

Continuous-Time Systems:

- $\sigma = \dot{x}$ (time derivative of x)

Discrete-Time Systems:

- $\sigma = x^+$ (state evolves as: $x[k+1] = Ax[k] + Bu[k]$)

Internal Properties of Linear Systems

In this module, we will focus on the following properties:

- Internal: Characteristics of matrix A
- Input-to-State: Dependent on (A, B)
- State-to-Output: Governed by (A, C)

In this lecture, we will focus on the **internal properties** of the system

State Trajectories: Continuous-Time Systems

System Behavior:

- Initial condition: $x(0) = x_0$
- Input: $u(t)$ for $t \geq 0$

State Evolution:

$$\dot{x}(0) = Ax_0 + Bu(0)$$

- Velocity at $t = 0$ represented as a vector in state space

Trajectories in State Space

Continuous-Time Systems:

- Smooth curve in state space

Discrete-Time Systems:

- Sequence of states: $x(0), x(1), x(2), \ldots$

Trajectories

Trajectory:

$$\text{Trajectory} = \{x(t) : t \geq 0\}$$

- A set of points in state space
- Evolution depends on initial state x_0 and input $u(t)$

Motions

Motion:

The pair $(t, x(t))$. i.e. the state and the time it is visited

For example

- *Trajectory: Train's path from London to Manchester*
- *Motion: Train's path with arrival and departure times*

Discrete vs. Continuous Systems

Continuous Systems:

- Physical systems (e.g., mechanical systems, celestial orbits)

Discrete Systems:

- Digital systems or sampled physical systems
- Example: Optimization algorithms or clocked circuits

Special Trajectories and Motions

- Certain trajectories and motions reveal intrinsic properties of linear systems
- In linear systems, these special cases generalize to other trajectories
- Understanding these trajectories offers deeper insights into system behaviour

Equilibrium of Linear Systems

Linear systems share properties across all trajectories

Equilibrium: A specific trajectory where the system state remains constant

- Originates from analytical mechanics, where equilibrium refers to a state of no motion

Definition of Equilibrium

Given:

- Initial condition: $x(0) = x_0$
- Constant input: $u(t) = u_0$

Definition: An equilibrium is a point x_0 such that:

$$x(t) = x_0 \quad \forall\, t \geq 0$$

Graphical Representation:

- Continuous-time: Straight line parallel to the t-axis in (x_1, x_2, t) space
- Discrete-time: Discrete points along the same straight line

Characterizing Equilibrium

Continuous-time system:

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

Steps:

1. If $x(t)$ is constant, $\dot{x}(t) = 0$
2. Substituting: $0 = Ax_0 + Bu_0$
3. Solve: $-Ax_0 = Bu_0$

Solution:

- If A is invertible: $x_0 = -A^{-1}Bu_0$.
- If $\det(A) = 0$:
  - No solution, or
  - Infinitely many solutions

Discrete-Time Systems

Discrete-time system:

$$x[k+1] = Ax[k] + Bu[k]$$

Equilibrium: x_0 satisfies:

$$x_0 = Ax_0 + Bu_0$$

Rearranging:

$$(I - A)x_0 = Bu_0$$

Solution:

- If $\det(I - A) \neq 0$: $x_0 = (I - A)^{-1}Bu_0$
- If $\det(I - A) = 0$:
  - No equilibrium, or
  - Infinitely many equilibria

Continuous-Time vs Discrete-Time Systems

Differences:

- Continuous: Check $\det(A)$ and solve $Ax_0 + Bu_0 = 0$
- Discrete: Check $\det(I - A)$ and solve $(I - A)x_0 = Bu_0$

Therefore

- *Eigenvalues at* 0 *(continuous) or* 1 *(discrete) are critical*
- *These eigenvalues affect steady-state performance and control design*

Example: Continuous-Time System

System:

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = -x_2 + u$$

Step 1: Formulate Matrices

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Step 2: Check Determinant

$$\det(A) = 0 \quad \Rightarrow \quad A \text{ is not invertible}$$

Example: Solution

Equilibrium Conditions:

$$0 = x_2, \quad 0 = -x_2 + u$$

Case 1: $u = 0$

- $x_2 = 0$, x_1 can take any value
- **Equilibrium Points:** $\{(x_1, 0) \mid x_1 \in \mathbb{R}\}$

Case 2: $u \neq 0$

- No equilibrium exists

Some observations

- Equilibrium analysis involves solving linear equations
- Eigenvalues at 0 (continuous) or 1 (discrete) are crucial for control design
- Unlike linear systems, non-linear systems may exhibit multiple distinct equilibria

Computation of Trajectories for Linear Systems

How do we compute trajectories for linear systems with time-varying inputs?

Given the state equation:

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

where $x(0)$ in the initial state and $u(t)$ in the input for all $t \geq 0$

We want to find $x(t)$ for $t \geq 0$

In other words, we want to find all future values of the state

Case 1: $A = 0$

State equation:

$$\dot{x} = Bu$$

Solution:

$$x(t) = x(0) + \int_0^t Bu(\tau)\, d\tau$$

- Contribution from initial state: $x(0)$
- Contribution from input: Integral term

Case 2: $A = \alpha$ (scalar)

State equation:

$$\dot{x} = \alpha x$$

Solution:

$$\ln \frac{x(t)}{x(0)} = \alpha t$$
$$x(t) = x(0)e^{\alpha t}$$

- Contribution from initial state: $x(0)$
- Contribution from the internal properties: $\mathrm{A} = \alpha$

General Case: Matrix A

State equation:

$$\dot{x} = Ax$$

Define a matrix exponential:

$$e^{At} = I + At + \frac{A^2 t^2}{2!} + \frac{A^3 t^3}{3!} + \ldots$$

- This definition is a tool to solve the general case for matrix A

Properties of Matrix Exponential

Property 1: Commutativity with A

$$e^{At}A = Ae^{At}$$

Property 2: Time derivative

$$\frac{d}{dt}e^{At} = Ae^{At}$$

- Analogous to scalar exponential properties

Many more properties shown in the module notes

Solving the Differential Equation

State equation:

$$\dot{x} = Ax + Bu, \quad \text{where } x(0) = x_0$$

Define:

$$z(t) = e^{-At}x(t)$$

Therefore:

$$\begin{aligned} \dot{z} &= e^{-At}Bu, \\ z(t) &= x(0) + \int_0^t e^{-A\tau}Bu(\tau)\,d\tau \end{aligned}$$

Solution:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)\,d\tau$$

This expression is known as the **Lagrange formula**

Free and Forced Response of the State of the System

The solution to the state-space equation can be expressed as:

Free response of the state of the system:

$$x_{\text{free}}(t) = e^{At}x(0)$$

Depends only on the initial state $x(0)$ and represents the natural dynamics of the system when $u(t) = 0$

Forced response of the state of the system::

$$x_{\text{forced}}(t) = \int_0^t e^{A(t-\tau)}Bu(\tau)\,d\tau$$

Driven by the input $u(t)$ and capturing the external forcing effect

Free and Forced Response of the Output

The output $y(t)$ is given by $y(t) = Cx(t) + Du(t)$

Substituting for $x(t)$:

$$y(t) = Ce^{At}x(0) + C\int_0^t e^{A(t-\tau)}Bu(\tau)\,d\tau + Du(t)$$

Free response of the output:

$$y_{\text{free}}(t) = Ce^{At}x(0)$$

Forced response of the output:

$$y_{\text{forced}}(t) = C\int_0^t e^{A(t-\tau)}Bu(\tau)\,d\tau + Du(t)$$

Due to linearity, the response leverages the principle of superposition

Transfer Function and Forced Response

Assuming $x_0 = 0$, the system has only the forced response:

$$y(t) = \int_0^t Ce^{A(t-\tau)}Bu(\tau)\,d\tau + Du(t)$$

Taking the Laplace transform:

$$Y(s) = \underbrace{\left[C(sI - A)^{-1}B + D\right]}_{G(s)} U(s)$$

- The transfer function $G(s)$ describes the input-output relationship under the assumption $x_0 = 0$
- For non-zero initial states, the transfer function cannot capture the complete system behaviour

Computing the Matrix Exponential

Trajectory computation requires calculating the matrix exponential e^{At}

Example 1: $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

- Compute $A^2 = 0$, so the series terminates:

$$e^{At} = I + At = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

Example 2: $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

- Recognize the cyclic pattern: $A^2 = -I$, $A^3 = -A$, $A^4 = I$
- Result:

$$e^{At} = \begin{bmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{bmatrix}$$

Python Example: Plotting Forced Response

```
import control as ct
import numpy as np
from matplotlib import pyplot as plt

A = [[-1, -2], [3, -4]]
B = [[5], [7]]
C = [[6, 8]]
D = [[9]]

t = np.linspace(0.0, 1.5, 100)
sys = ct.ss(A, B, C, D)
T, y = ct.forced_response(sys, T=t, X0=[1, 0])

plt.plot(T, y, 'r')
plt.show()
```