

# EDB Math Modelling Workshop (Part II)

Presenter: LIU Liu, The Chinese University of Hong Kong

May 16, 2026

- 1 Introduction: ODE Systems and Modeling
- 2 Linear Systems: Solutions and Structure
  - Basic knowledge
  - Theoretical application of ODE system
- 3 Beyond Linear Systems: General ODE Systems
- 4 Classic Modeling Examples
  - SIR
  - Lotka-Volterra (Predator-Prey) Model
  - Theory and Parameter Choice
- 5 Numerical Methods for ODE Systems
- 6 Summary and Transition to Practice
- 7 Problems
- 8 Solutions

# What is a System of ODEs?

- Mathematical description of interacting quantities:

$$\begin{cases} \frac{dx_1}{dt} = f_1(t, x_1, x_2, \dots, x_n) \\ \frac{dx_2}{dt} = f_2(t, x_1, x_2, \dots, x_n) \\ \vdots \\ \frac{dx_n}{dt} = f_n(t, x_1, x_2, \dots, x_n) \end{cases}$$

## Compact Notation

$$\frac{d\mathbf{x}}{dt} = A(t)\mathbf{x} + \mathbf{f}(t)$$

where

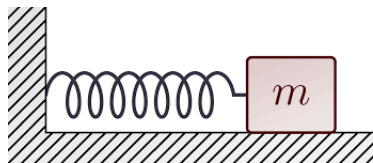
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad A(t) = \begin{bmatrix} a_{11}(t) & \cdots & a_{1n}(t) \\ \vdots & \ddots & \vdots \\ a_{n1}(t) & \cdots & a_{nn}(t) \end{bmatrix}, \quad \mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_n(t) \end{bmatrix}$$

## From Reality to Equations:

### Case Study: Spring-Mass-Damper System

#### Physical System

- Mass  $m$  attached to spring ( $k$ ) and damper ( $c$ )
- Displacement  $x(t)$  from equilibrium



# Spring-Mass-Damper Model Equations

## Modeling Steps

- 1 Newton's 2nd Law:

$$m\ddot{x} = -c\dot{x} - kx$$

- 2 Introduce state variables:  $x_1 = x$ ,  $x_2 = \dot{x}$

- 3 Convert to system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 \end{cases}$$

## Matrix Form

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

# Fundamental Classifications of ODE Systems

## System Types

- **Linear/Nonlinear:**

$$\frac{dx}{dt} = Ax \quad \text{vs} \quad \frac{dx}{dt} = \mathbf{f}(x)$$

- **Autonomous/Non-autonomous:**

$$\mathbf{f}(x) \quad \text{vs} \quad \mathbf{f}(t, x)$$

- **Constant/Variable coefficients**

## Reality

Most real-world systems are:

- Nonlinear
- High-dimensional
- Require numerical solutions

## Biology/Ecology

- Predator-Prey Dynamics:

$$\begin{cases} \frac{dR}{dt} = \alpha R - \beta RF \\ \frac{dF}{dt} = \delta RF - \gamma F \end{cases}$$

## Epidemiology

- SIR Model:

$$\begin{cases} \frac{dS}{dt} = -\beta SI \\ \frac{dI}{dt} = \beta SI - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases}$$

## Applicability

The framework of ODE systems applies broadly:

- Engineering systems
- Economic models
- Climate and environmental models
- Neural and biological systems

# From Scalar ODE to Systems

- Consider a first-order scalar ODE:

$$x'(t) = ax(t)$$

- Try a solution of the form:  $x(t) = e^{\lambda t}u$

$$\Rightarrow x'(t) = \lambda e^{\lambda t}u = ae^{\lambda t}u \Rightarrow \lambda = a$$

- Now consider a system:

$$\mathbf{x}'(t) = A\mathbf{x}(t)$$

- Can we try a similar form? Yes!

$$\mathbf{x}(t) = \mathbf{u}e^{\lambda t} \Rightarrow A\mathbf{u} = \lambda\mathbf{u}$$

- Leads to the eigenvalue problem for matrix  $A$

# Details of the Derivation

From Assumption to Eigenvalue Equation

## Step 1: Assume Exponential Solution

$$\mathbf{x}(t) = e^{\lambda t} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = e^{\lambda t} \mathbf{u}$$

## Step 2: Differentiate Solution

$$\frac{d\mathbf{x}}{dt} = \lambda e^{\lambda t} \mathbf{u}$$

## Step 3: Substitute into System

$$\lambda e^{\lambda t} \mathbf{u} = A(e^{\lambda t} \mathbf{u})$$

## Step 4: Simplify

$$\lambda \mathbf{u} = A\mathbf{u} \Rightarrow (A - \lambda I)\mathbf{u} = \mathbf{0}$$

# Critical Equation

## Characteristic Equation

Non-trivial solutions require:

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

## 2x2 Case Demonstration

For matrix  $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ :

$$\lambda^2 - (a + d)\lambda + (ad - bc) = 0$$

Solutions:

$$\lambda = \frac{(a + d) \pm \sqrt{(a - d)^2 + 4bc}}{2}$$

- Roots determine solution behavior
- Complex/Repeated roots require special handling

## Example(as exercise)

The initial condition would be given later.

**Given System:**

$$\begin{cases} \frac{dx_1}{dt} = 2x_1 - 2x_2 \\ \frac{dx_2}{dt} = 2x_1 - 3x_2 \end{cases}$$

**Matrix Representation:**

$$A = \begin{bmatrix} 2 & -2 \\ 2 & -3 \end{bmatrix}$$

**Characteristic Equation:**

$$\det \begin{bmatrix} 2 - \lambda & -2 \\ 2 & -3 - \lambda \end{bmatrix} = \lambda^2 + \lambda - 2 = 0$$

# Eigenvalue Problem

## Roots Calculation:

$$\lambda^2 + \lambda - 2 = (\lambda - 1)(\lambda + 2) = 0$$

$$\lambda_1 = 1, \quad \lambda_2 = -2$$

## Eigenvector Determination

For  $\lambda_1 = 1$ :

$$\begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \mathbf{0}$$

Solution:  $u_1 = 2u_2$

$$\mathbf{u}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

For  $\lambda_2 = -2$ :

$$\begin{bmatrix} 4 & -2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \mathbf{0}$$

Solution:  $v_1 = \frac{1}{2}v_2$

$$\mathbf{u}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

# Superposition Principle

## Theorem (Superposition Principle)

If  $\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_k(t)$  are solutions to the homogeneous linear system

$$\mathbf{x}' = A\mathbf{x},$$

then any linear combination

$$\mathbf{x}(t) = c_1\mathbf{x}_1(t) + c_2\mathbf{x}_2(t) + \dots + c_k\mathbf{x}_k(t)$$

is also a solution, where  $c_1, \dots, c_k \in \mathbb{R}$ .

## Remark

This principle allows us to build the general solution by combining basic solutions.

- Helps us construct solutions from eigen-solutions.

# Eigenvalue Method for Linear Systems

## Theorem (General Solution of a Linear System)

Suppose  $A \in \mathbb{R}^{n \times n}$  has  $n$  linearly independent eigenvectors  $\mathbf{v}_1, \dots, \mathbf{v}_n$  with corresponding eigenvalues  $\lambda_1, \dots, \lambda_n$ . Then the general solution of the system

$$\mathbf{x}' = A\mathbf{x}$$

is given by

$$\mathbf{x}(t) = c_1\mathbf{v}_1e^{\lambda_1 t} + c_2\mathbf{v}_2e^{\lambda_2 t} + \dots + c_n\mathbf{v}_ne^{\lambda_n t},$$

where  $c_1, \dots, c_n \in \mathbb{R}$  are constants determined by initial conditions.

## Remark

Each term  $\mathbf{v}_i e^{\lambda_i t}$  represents a fundamental mode of the system. The sign and nature (real/complex) of  $\lambda_i$  determines the behavior:

- $\lambda_i > 0$ : exponential growth
- $\lambda_i < 0$ : exponential decay
- $\lambda_i \in \mathbb{C}$ : oscillation (spiral behavior)

## Theorem (General Solution Formula)

$$\mathbf{x}(t) = c_1 e^{\lambda_1 t} \mathbf{u}_1 + c_2 e^{\lambda_2 t} \mathbf{u}_2$$

**Applied to Our Case:**

$$\mathbf{x}(t) = c_1 e^t \begin{bmatrix} 2 \\ 1 \end{bmatrix} + c_2 e^{-2t} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

## Component Form

$$\begin{cases} x_1(t) = 2c_1 e^t + c_2 e^{-2t} \\ x_2(t) = c_1 e^t + 2c_2 e^{-2t} \end{cases}$$

# Determining Constants with the Initial Condition

**Given Initial Condition:**

$$\mathbf{x}(0) = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$$

**System of Equations:**

$$\begin{cases} 2c_1 + c_2 = -1 \\ c_1 + 2c_2 = 4 \end{cases}$$

**Matrix Form:**

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$$

**Solution:**

$$c_1 = -2, \quad c_2 = 3$$

## Complete Particular Solution

$$\mathbf{x}(t) = -2e^t \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 3e^{-2t} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

**Component Form:**

$$\begin{cases} x_1(t) = -4e^t + 3e^{-2t} \\ x_2(t) = -2e^t + 6e^{-2t} \end{cases}$$

- Demonstrates superposition principle
- Shows an exponential dominant evolution
- Illustrates time system behavior

# Vector Fields: Geometry of the System

- A system  $\mathbf{x}' = A\mathbf{x}$  defines a **vector field**:

$$\mathbf{F}(\mathbf{x}) = A\mathbf{x}$$

- At each point  $\mathbf{x} \in \mathbb{R}^2$ , assign a vector  $\mathbf{x}'$
- This field describes the **instantaneous direction** of motion

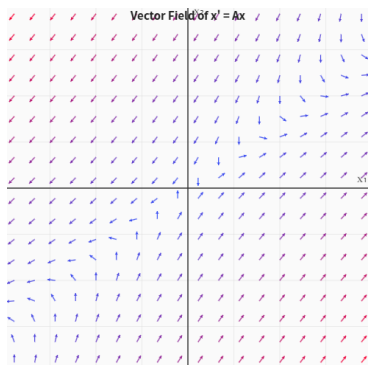


Figure: Visualizing it helps understand solution dynamics

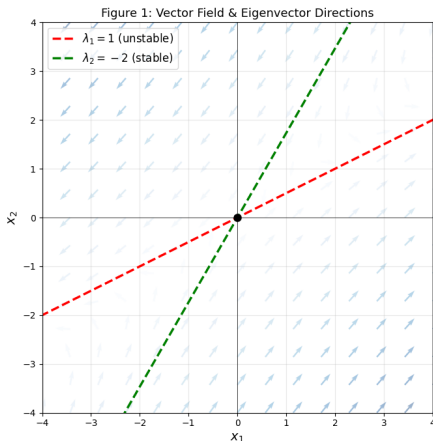
# Phase Portrait (plain)

## Key Features:

- Eigenvectors define solution directions
- $\lambda > 0$ : Exponential growth along  $\mathbf{v}_1$
- $\lambda < 0$ : Exponential decay along  $\mathbf{v}_2$

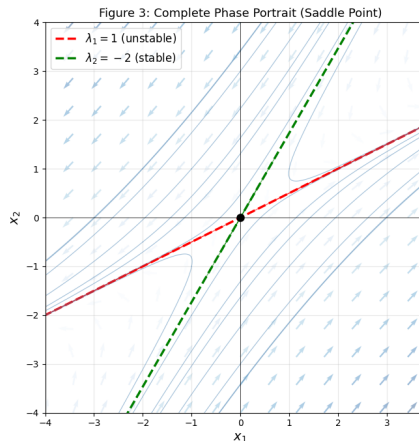
## Classification:

- Saddle point: Eigenvalues with opposite signs
- Node: Real eigenvalues with same sign



# Phase Portraits and Dynamics

- A **phase portrait** shows the trajectories of solutions in the phase plane
- **Eigenvectors** define invariant lines (solutions stay on them)
- If  $\mathbf{x}(t) = \mathbf{v}e^{\lambda t}$ :
  - $\lambda > 0$ : solution diverges (unstable)
  - $\lambda < 0$ : solution decays (stable)
- Classification of equilibrium at origin:
  - **Stable node**: two negative real eigenvalues
  - **Unstable node**: two positive real eigenvalues
  - **Saddle point**: eigenvalues with opposite signs
  - **Spiral / Center**: complex eigenvalues



- Phase portraits show:
  - Straight-line or spiral solutions
  - Direction aligned with eigenvectors
  - Growth or decay rates driven by eigenvalues
- This geometric behavior matches the algebraic solution:

$$\mathbf{x}(t) = c_1 \mathbf{v}_1 e^{\lambda_1 t} + c_2 \mathbf{v}_2 e^{\lambda_2 t} + \dots$$

- **The vector field encodes the entire solution structure!**

# Original Higher-Order ODE

Consider the third-order ODE:

$$y''' - 2y'' + y' - 3y = \sin(t)$$

with initial conditions:

$$y(0) = 1, \quad y'(0) = 0, \quad y''(0) = 2$$

Our goal is to convert this into a system of first-order ODEs.

# Define Substitutions

Let us define:

$$y_1 = y$$

$$y_2 = y'$$

$$y_3 = y''$$

Then their derivatives are:

$$y_1' = y_2$$

$$y_2' = y_3$$

$$y_3' = \sin(t) + 2y_3 - y_2 + 3y_1$$

So we now have a system of first-order ODEs.

# First-Order System with Initial Conditions

The first-order system is:

$$\frac{d}{dt} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \sin(t) + 2y_3 - y_2 + 3y_1 \end{bmatrix}$$

With initial conditions:

$$y_1(0) = 1, \quad y_2(0) = 0, \quad y_3(0) = 2$$

or  $\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$ , pls solve this ODE system with colab.

# Model Examples

- SIR Infectious Disease Model
- Lotka-Volterra (Predator-Prey) Model

# SIR Model: Introduction

- A compartmental model in epidemiology
- Population divided into three groups:
  - **S**usceptible: individuals who can contract the disease
  - **I**nfectious: individuals who have the disease and can transmit it
  - **R**ecovered: individuals who have recovered or died, and are no longer infectious
- Total population size:  $N = S(t) + I(t) + R(t)$
- **Key parameters:**
  - $\beta$ : transmission rate — the average number of contacts per person per time that are sufficient for transmission;
  - $\gamma$ : recovery rate — the rate at which infectious individuals recover and move into the recovered population.

# SIR Model: Structure

- The SIR model describes how a contagious disease spreads through a population over time.
- It is governed by a system of ordinary differential equations (ODEs),
- **The dynamics of the model are given by:**

$$\frac{dS}{dt} = -\beta \frac{SI}{N} \quad (\text{susceptible individuals become infected})$$

$$\frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I \quad (\text{infected individuals recover or continue spreading})$$

$$\frac{dR}{dt} = \gamma I \quad (\text{infected individuals recover and gain immunity})$$

where  $N = S + I + R$  is the total population.

# SIR Model: Assumptions

- Closed population (no births, deaths, or migration)
- Homogeneous mixing (each individual equally likely to meet any other)
- Constant transmission and recovery rates
- Instantaneous transition between compartments (no intermediate state for individuals)

- **Basic Reproduction Number:**

$$R_0 = \frac{\beta}{\gamma}$$

- $R_0 > 1$ : outbreak can spread in the population
- $R_0 < 1$ : infection will die out
- Helps in understanding:
  - Epidemic threshold
  - Peak infection time and size
  - Impact of interventions like vaccination

# SIR Model: Applications

- Applied to modeling outbreaks such as SARS, COVID-19, etc.
- Example: Hong Kong and Shanghai during an epidemic
- Model calibration using real-world data
- Comparison of intervention strategies between regions

# Lotka-Volterra Model: Introduction

- A classic model in mathematical biology for predator-prey interactions.
- Describes the dynamics of biological systems in which two species interact:
  - **Prey** population (e.g., rabbits)
  - **Predator** population (e.g., foxes)
- The model assumes:
  - Prey population grows exponentially in absence of predators.
  - Predators rely entirely on prey for food.
  - Interaction between species affects population sizes.

# Lotka-Volterra Model: Structure

- The model is defined by a system of two ordinary differential equations:

$$\frac{dx}{dt} = \alpha x - \beta xy \quad (\text{prey population})$$

$$\frac{dy}{dt} = \delta xy - \gamma y \quad (\text{predator population})$$

- Where:
  - $x(t)$ : number of prey at time  $t$ , and if  $y(t) \equiv 0$ , then  $x(t) = e^{\alpha t}$
  - $y(t)$ : number of predators at time  $t$ , and if  $x(t) \equiv 0$ , then  $y$  would be decreasing to a semi-positive constant, which is 0
  - $\alpha$ : natural growth rate of prey
  - $\beta$ : predation rate coefficient
  - $\gamma$ : natural death rate of predators
  - $\delta$ : rate at which consumed prey is converted into predators

# Lotka-Volterra Model: Assumptions

- The environment does not change in ways that affect the interaction.
- Predators only survive by consuming prey.
- Prey have unlimited food supply (no intra-species competition).
- No immigration or emigration.
- Interaction between species is proportional to the product of their populations.

# Lotka-Volterra Model: Interpretation

- The model produces oscillatory dynamics:
  - Prey population rises  $\rightarrow$  predator population increases.
  - More predators  $\rightarrow$  prey population declines.
  - Less prey  $\rightarrow$  predator population declines.
  - Fewer predators  $\rightarrow$  prey can recover.
- These cycles continue indefinitely under ideal conditions.
- Real ecosystems have additional complexities (e.g., carrying capacities, multiple species).

# Lotka-Volterra Model: Applications

- Used to understand population dynamics in ecology.
- Applied to:
  - Predator-prey systems
  - Host-parasite interactions
  - Competition between species
- Provides insight into:
  - Sustainability of ecosystems
  - Effects of hunting, conservation, and environmental changes

# Modeling: Role of Theory

- Mathematical models are abstractions of real systems guided by biological, physical, or social theory.
- Theory informs:
  - Which variables to include (e.g., compartments, species)
  - The form of interactions (e.g., linear vs. nonlinear terms)
  - The structure of equations (e.g., ODEs, difference equations)
- Example: SIR model based on assumptions about disease transmission and recovery.

# Choosing Parameters: Theoretical and Empirical Sources

- Parameters can come from:
  - **Theory:** e.g., recovery time  $\Rightarrow \gamma = \frac{1}{T}$
  - **Empirical data:** e.g., case counts, population surveys
  - **Experiments:** e.g., lab measurements of growth or infection rates
- Role of units and dimensional analysis in parameter consistency
- Sensitivity: small changes in parameters can lead to large changes in outcomes

# Parameter Estimation: Example from SIR

- Suppose we observe an outbreak and gather time-series data on infected individuals.
- Use the reported average infectious period  $T$  to estimate  $\gamma = 1/T$
- Estimate  $\beta$  by fitting the model to the data (e.g., least squares or MLE)
- **Example:**

$$\gamma = \frac{1}{7} \approx 0.143 \quad (7\text{-day infectious period})$$

$$R_0 = 2.5 \Rightarrow \beta = R_0 \cdot \gamma = 2.5 \cdot 0.143 \approx 0.357$$

# Impact of Assumptions and Simplifications

- Every model involves simplifying assumptions.
- Unrealistic assumptions can lead to misleading predictions.
- Examples:
  - Homogeneous mixing in SIR model may not hold in real populations.
  - Constant rates may ignore seasonality or behavior change.
- Sensitivity analysis and model validation are critical.

## Next part: Numerical solutions

- Most nonlinear systems cannot be solved analytically, and when analysis is too difficult, we turn to numerical methods
- Idea: approximate solution step-by-step
- Widely used in engineering, physics, biology, finance, etc.

# Why Numerical Solutions?

- Numerical methods approximate solutions given:
  - A differential equation:  $\dot{x} = f(x, t)$
  - An initial condition:  $x(t_0) = x_0$
- Main goals:
  - Accuracy
  - Stability
  - Efficiency
- Will also discuss practical implementation in colab with Python
- Basic methods: Euler, Runge-Kutta

# Euler's Method

- Simplest numerical method
- Given  $\dot{x} = f(x, t)$  and  $x(t_0) = x_0$
- Step forward using:

$$x_{n+1} = x_n + hf(x_n, t_n)$$

- $h$  is the time step (step size)
- Pros: easy to implement
- Cons: low accuracy, unstable for **stiff systems**.

# Backward Euler: Method and Stability

- Implicit method:

$$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1})$$

- Taylor expansion: global error  $\mathcal{O}(h)$
- Stability analysis (test eq.  $\dot{x} = \lambda x$ ):

$$x_{n+1} = x_n + h\lambda x_{n+1} \Rightarrow x_{n+1} = \frac{x_n}{1 - h\lambda}$$

- So  $|x_{n+1}/x_n| = \left| \frac{1}{1-h\lambda} \right|$
- If  $\text{Re}(\lambda) < 0$ , then method is **A-stable**

# Runge-Kutta Methods (RK4)

- More accurate than Euler
- Classical 4th-order Runge-Kutta (RK4):

$$k_1 = f(x_n, t_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}k_1, t_n + \frac{h}{2}\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}k_2, t_n + \frac{h}{2}\right)$$

$$k_4 = f(x_n + hk_3, t_n + h)$$

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

- Widely used due to balance of accuracy and efficiency

# Step Size, Error, and Stability

- Step size  $h$  affects:
  - Accuracy: smaller  $h \Rightarrow$  more accurate
  - Computation time: smaller  $h \Rightarrow$  more steps
- Error types:
  - Local truncation error (per step)
  - Global error (cumulative)
- Stability:
  - Some methods unstable for large  $h$
  - Especially important for stiff equations

# Forward Euler: Error and Stability

- Consider IVP:  $\dot{x} = f(x, t)$ , with  $x(t_0) = x_0$
- Taylor expansion about  $t_n$ :

$$x(t_{n+1}) = x(t_n) + hf(x(t_n), t_n) + \frac{h^2}{2}x''(\xi), \quad \xi \in [t_n, t_{n+1}]$$

- Forward Euler uses:

$$x_{n+1} = x_n + hf(x_n, t_n)$$

- So local truncation error (LTE):

$$\text{LTE} = x(t_{n+1}) - x_{n+1} = \frac{h^2}{2}x''(\xi) \Rightarrow \mathcal{O}(h^2)$$

- Global error accumulates:  $\mathcal{O}(h)$
- Stability analysis (test eq:  $\dot{x} = \lambda x$ ):

$$x_{n+1} = (1 + h\lambda)x_n$$

- Method is stable if  $|1 + h\lambda| < 1$
- For stiff problems (large negative  $\lambda$ ), this requires very small  $h$

- **Local truncation error (LTE):**  $\mathcal{O}(h^2)$
- **Global error:**  $\mathcal{O}(h)$
- **Stability:**
  - Consider test equation:  $\dot{x} = \lambda x$
  - Forward Euler:  $x_{n+1} = x_n + h\lambda x_n = (1 + h\lambda)x_n$
  - Stable if  $|1 + h\lambda| < 1$
  - For stiff problems (large negative  $\lambda$ ), requires very small  $h$
- **Conclusion:** Not suitable for stiff systems

# RK4: Error and Stability

- RK4 uses intermediate slopes  $k_1, k_2, k_3, k_4$
- Taylor expansion of exact solution gives:

$$x(t_{n+1}) = x(t_n) + hf + \frac{h^2}{2} f'f + \frac{h^3}{6} f''f^2 + \dots$$

- RK4 matches up to 4th-order terms  $\Rightarrow$  LTE:  $\mathcal{O}(h^5)$
- Global error:  $\mathcal{O}(h^4)$
- RK4 has a larger stability region than Forward Euler

# Backward Euler Method

- **Implicit method**, good for stiff systems
- Given:  $\dot{x} = f(x, t)$ , then

$$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1})$$

- Requires solving nonlinear equation at each step (e.g., via Newton's method)
- **Stability:**
  - For  $\dot{x} = \lambda x$ :  $x_{n+1} = \frac{x_n}{1-h\lambda}$
  - Unconditionally stable if  $\text{Re}(\lambda) < 0$
- **Conclusion:** Preferred for stiff systems despite higher computational cost

# Backward Euler: Solving the System Practically

If  $\mathbf{f}$  is not complex, there are other ways to solve the implicit scheme, e.g. with inverse matrices

- For system:  $\dot{\mathbf{x}} = f(\mathbf{x}, t)$ ,  $\mathbf{x} \in \mathbb{R}^n$
- Backward Euler update:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + hf(\mathbf{x}_{n+1}, t_{n+1})$$

- This is an implicit equation in  $\mathbf{x}_{n+1}$
- To solve:
  - Apply Newton's method: iterate

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[ I - hJ_f(\mathbf{x}^{(k)}) \right]^{-1} \left[ \mathbf{x}^{(k)} - \mathbf{x}_n - hf(\mathbf{x}^{(k)}, t_{n+1}) \right]$$

- Each iteration solves a linear system:

$$\left[ I - hJ_f \right] \delta \mathbf{x} = - \left( \mathbf{x}^{(k)} - \mathbf{x}_n - hf(\mathbf{x}^{(k)}, t_{n+1}) \right)$$

- This is a nonlinear system  $\Rightarrow$  use iterative solvers at each time step

# Example: Nonlinear 2D System

- Consider:

$$\begin{cases} \dot{x} = x(1 - y) \\ \dot{y} = y(x - 1) \end{cases}$$

- Forward Euler:

$$\begin{cases} x_{n+1} = x_n + h x_n(1 - y_n) \\ y_{n+1} = y_n + h y_n(x_n - 1) \end{cases}$$

- Runge-Kutta 4 (RK4):

$$\begin{aligned} k_1^x &= x_n(1 - y_n), & k_1^y &= y_n(x_n - 1) \\ k_2^x &= (x_n + \frac{h}{2}k_1^x)(1 - (y_n + \frac{h}{2}k_1^y)), & k_2^y &= (y_n + \frac{h}{2}k_1^y)((x_n + \frac{h}{2}k_1^x) - 1) \\ k_3^x &= (x_n + \frac{h}{2}k_2^x)(1 - (y_n + \frac{h}{2}k_2^y)), & k_3^y &= (y_n + \frac{h}{2}k_2^y)((x_n + \frac{h}{2}k_2^x) - 1) \\ k_4^x &= (x_n + h k_3^x)(1 - (y_n + h k_3^y)), & k_4^y &= (y_n + h k_3^y)((x_n + h k_3^x) - 1) \end{aligned}$$

$$\begin{cases} x_{n+1} = x_n + \frac{h}{6}(k_1^x + 2k_2^x + 2k_3^x + k_4^x) \\ y_{n+1} = y_n + \frac{h}{6}(k_1^y + 2k_2^y + 2k_3^y + k_4^y) \end{cases}$$

# Summary and Next Steps

- Key theoretical concepts in ODE systems are complete.
- Now, we move on to applying the theory through practical computation.
- Next, we have prepared a session to practice everything we've learned using Colab and Python. You can refer back to these slides later as a guide when coding.

**Will be covered in Colab notebook**

# Problem 1 — Qualitative Phase Portraits & Linearization

For each system below, complete parts (a)–(d):

- (a) Find all equilibrium points.
- (b) Write down the nullclines ( $x' = 0$  and  $y' = 0$ ). Use them to sketch a typical phase portrait.
- (c) Compute the Jacobian matrix  $J(x, y)$ . For each equilibrium point, compute the eigenvalues.
- (d) Classify each equilibrium point and state its stability.

**System A:**

$$\begin{cases} x' = x(1 - x) - xy, \\ y' = -y + xy. \end{cases}$$

**System B:**

$$\begin{cases} x' = x(1 - y), \\ y' = y(2 - x). \end{cases}$$

## Problem 2 — Euler's Method for a System

Consider the linear system

$$\begin{cases} x' = -x + y, \\ y' = -2x - y, \end{cases} \quad x(0) = 1, y(0) = 0.$$

With  $\Delta t = 0.05$  and the explicit Euler method:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + 0.05 \begin{pmatrix} -x_n + y_n \\ -2x_n - y_n \end{pmatrix}$$

approximate the solution on  $t = 0, 0.05, \dots, 0.50$ .

### Tasks:

- Present a table of  $(t_n, x_n, y_n)$  for  $n = 0, 1, \dots, 10$ .
- Plot the discrete trajectory in the  $xy$ -plane (connect the points and indicate the direction of increasing  $t$ ).
- Describe whether the trajectory moves inward, outward, or circles around the origin. Justify by discussing  $\sqrt{x_n^2 + y_n^2}$ .

# Solutions

# System A — (a) Equilibria

Rewrite:

$$x' = x(1 - x - y), \quad y' = y(x - 1).$$

Solve  $x' = 0$  and  $y' = 0$ :

$$x' = 0 \iff x = 0 \text{ or } 1 - x - y = 0,$$

$$y' = 0 \iff y = 0 \text{ or } x = 1.$$

Equilibrium points:

$$(0, 0), \quad (1, 0).$$

# System A — (b) Nullclines and Phase Directions

Nullclines:

$$x\text{-nullclines: } x = 0, \quad y = 1 - x$$

$$y\text{-nullclines: } y = 0, \quad x = 1$$

Signs in the first quadrant:

$$x' = x(1 - x - y) \Rightarrow \begin{cases} x' > 0 & \text{below } y = 1 - x, \\ x' < 0 & \text{above } y = 1 - x, \end{cases}$$

$$y' = y(x - 1) \Rightarrow \begin{cases} y' > 0 & x > 1, \\ y' < 0 & x < 1. \end{cases}$$

## System A — (c) Jacobian and Eigenvalues

$$J(x, y) = \begin{pmatrix} 1 - 2x - y & -x \\ y & x - 1 \end{pmatrix}$$

At  $(0, 0)$ :

$$J(0, 0) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \lambda_1 = 1, \quad \lambda_2 = -1.$$

At  $(1, 0)$ :

$$J(1, 0) = \begin{pmatrix} -1 & -1 \\ 0 & 0 \end{pmatrix}, \quad \lambda_1 = -1, \quad \lambda_2 = 0.$$

# System A — (d) Classification and Stability

- $(0, 0)$ : **Saddle** (eigenvalues  $+1, -1$ ); **unstable**.
- $(1, 0)$ : **Non-hyperbolic** (one eigenvalue  $0$ ); linearization **inconclusive**.

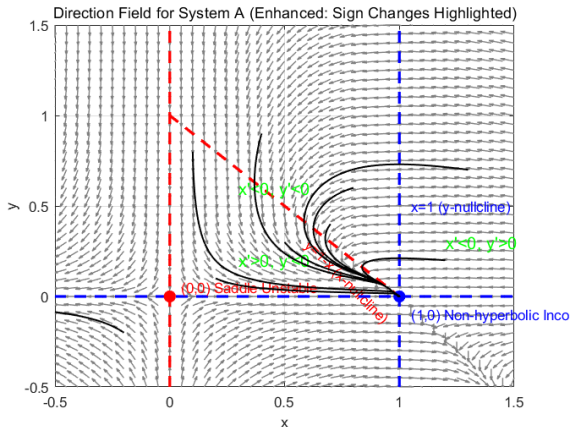


Figure: Phase portrait of System A

## System B — (a) Equilibria

Solve  $x' = 0$  and  $y' = 0$ :

$$x' = 0 \iff x = 0 \text{ or } y = 1,$$

$$y' = 0 \iff y = 0 \text{ or } x = 2.$$

Equilibrium points:

$$(0, 0), \quad (2, 1).$$

## System B — (b) Nullclines and Phase Directions

Nullclines:

$$x\text{-nullclines: } x = 0, \quad y = 1$$

$$y\text{-nullclines: } y = 0, \quad x = 2$$

Signs:

$$x' = x(1 - y) \Rightarrow \begin{cases} x' > 0 & y < 1, \\ x' < 0 & y > 1, \end{cases}$$

$$y' = y(2 - x) \Rightarrow \begin{cases} y' > 0 & x < 2, \\ y' < 0 & x > 2. \end{cases}$$

In  $0 < x < 2$ ,  $0 < y < 1$ : flow is up-and-right.

## System B — (c) Jacobian and Eigenvalues

$$J(x, y) = \begin{pmatrix} 1 - y & -x \\ -y & 2 - x \end{pmatrix}$$

At (0, 0):

$$J(0, 0) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad \lambda_1 = 1, \quad \lambda_2 = 2.$$

At (2, 1):

$$J(2, 1) = \begin{pmatrix} 0 & -2 \\ -1 & 0 \end{pmatrix}, \quad \lambda^2 - 2 = 0, \quad \lambda_{1,2} = \pm\sqrt{2}.$$

# System B — (d) Classification and Stability

- $(0, 0)$ : **Source** (both eigenvalues  $> 0$ ); **unstable**.
- $(2, 1)$ : **Saddle** (eigenvalues  $\pm\sqrt{2}$ ); **unstable**.

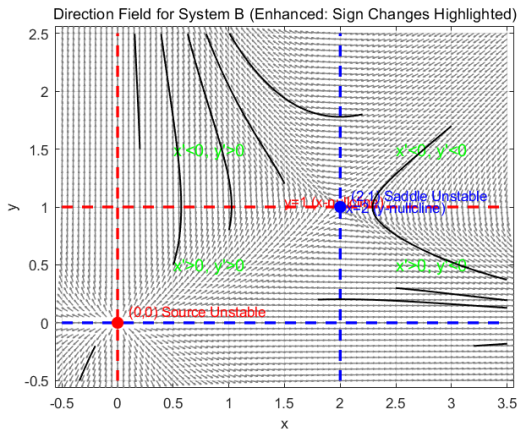


Figure: Phase portrait of System B

## Problem 2 — Solution Setup

Euler update with  $\Delta t = 0.05$ :

$$x_{n+1} = 0.95 x_n + 0.05 y_n,$$

$$y_{n+1} = -0.10 x_n + 0.95 y_n.$$

## Problem 2 — (a) Table of Values

$n$	$t_n$	$x_n$	$y_n$
0	0.00	1.000000	0.000000
1	0.05	0.950000	-0.100000
2	0.10	0.897500	-0.190000
3	0.15	0.843125	-0.270250
4	0.20	0.787456	-0.341050
5	0.25	0.731031	-0.402743

## Problem 2 — (a) Table of Values (cont.)

$n$	$t_n$	$x_n$	$y_n$
6	0.30	0.674342	-0.455709
7	0.35	0.617840	-0.500358
8	0.40	0.561930	-0.537124
9	0.45	0.506977	-0.566461
10	0.50	0.453305	-0.588835

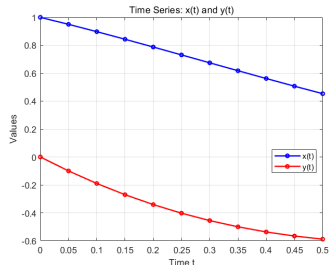


Figure: Time series

## Problem 2 — (b) Discrete Trajectory

Plot  $(x_n, y_n)$  in the  $xy$ -plane, connected in order of increasing  $n$ .  
Direction: from  $(1, 0)$  curving into the fourth quadrant.

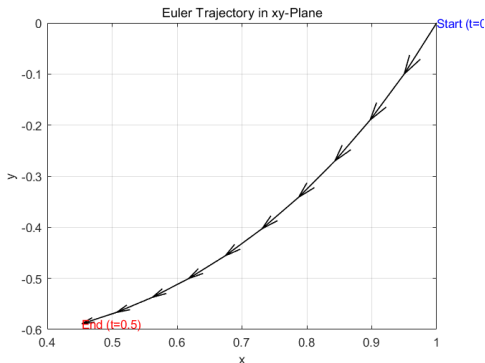


Figure: Euler trajectory in  $xy$ -plane

## Problem 2 — (c) Qualitative Behavior

The trajectory **spirals inward** toward the origin:

- Both  $|x_n|$  and  $|y_n|$  stay bounded.
- $\sqrt{x_n^2 + y_n^2}$  decreases over these steps.
- The path rotates into the fourth quadrant while decaying.

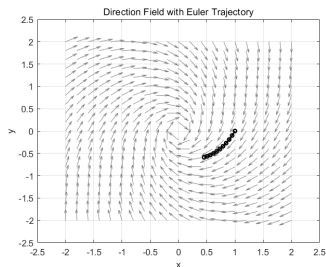


Figure: Direction field (optional)