Week #11: Ordinary Differential Equations

Numerical Methods: Mathematical Foundations

June 22, 2025

Overview

Ordinary differential equations (ODEs) model dynamic systems across virtually every field of science and engineering. This week we develop the mathematical theory underlying numerical methods for solving initial value problems, exploring both single-step and multistep methods while analyzing their stability, accuracy, and convergence properties.

The fundamental challenge in numerical ODE solving is to approximate continuous evolution with discrete time steps while maintaining essential properties of the solution such as stability and conservation laws. We examine how different discretization strategies balance accuracy and computational efficiency, and investigate the mathematical principles that determine method performance.

Our analysis emphasizes the mathematical foundations of ODE solver design, the crucial role of stability analysis in method selection, and the error analysis that guides adaptive step size strategies and method implementation.

---

1 Initial Value Problems

1.1 Mathematical Framework

Definition. Initial Value Problem

An initial value problem (IVP) for a first-order ODE is:

$$y'(t) = f(t, y(t)), \quad t \in [t_0, T] \tag{1}$$

$$y(t_0) = y_0 \tag{2}$$

where $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ and $y_0 \in \mathbb{R}^d$.

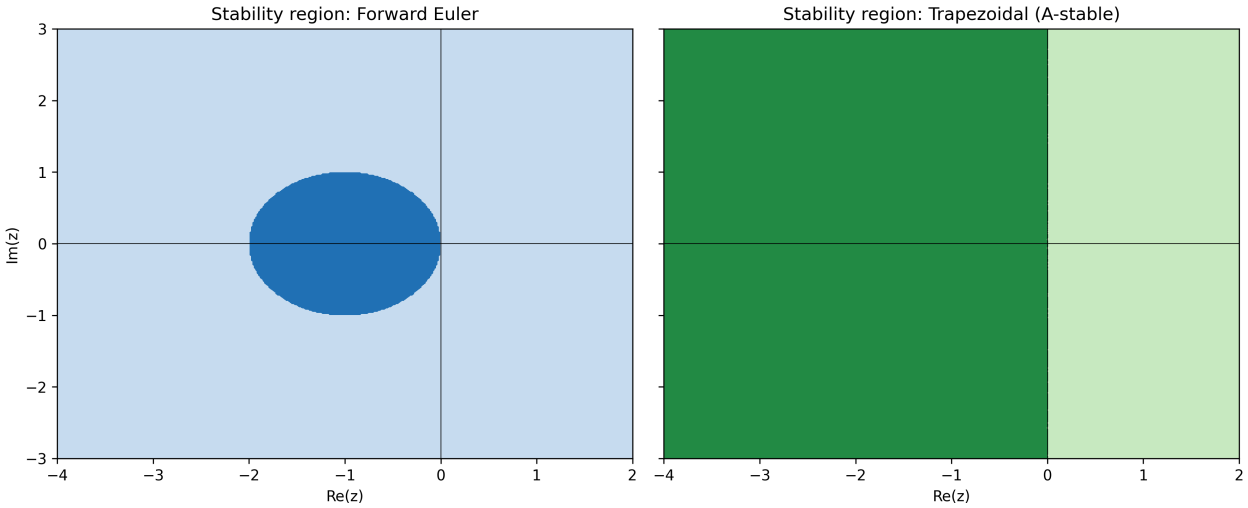

Figure 1: Absolute stability regions for Forward Euler and Trapezoidal (implicit midpoint) methods in the complex $z = h\lambda$ plane.

Theorem 1.1 (Existence and Uniqueness - Picard-Lindelöf)**.** *If f and $\partial f/\partial y$ are continuous in a neighborhood of (t_0, y_0), then the IVP has a unique solution in some interval $[t_0, t_0 + h]$.*

1.2 Well-Posedness and Conditioning

Definition. Lipschitz Condition

Function $f(t, y)$ satisfies a Lipschitz condition in y if there exists $L > 0$ such that:

$$\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\|$$

for all t in the domain and y_1, y_2 in the region of interest.

Theorem 1.2 (Sensitivity to Initial Conditions)**.** *If $y(t)$ and $z(t)$ are solutions to the IVP with initial conditions y_0 and z_0 respectively, then:*

$$\|y(t) - z(t)\| \leq e^{L(t-t_0)}\|y_0 - z_0\|$$

where L is the Lipschitz constant.

2 Euler Methods

2.1 Forward Euler Method

Definition. Forward Euler Method

The simplest numerical method for ODEs:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{3}$$
$$t_{n+1} = t_n + h \tag{4}$$

where h is the step size and $y_n \approx y(t_n)$.

Theorem 2.1 (Forward Euler Local Truncation Error)**.** *If $y \in C^2[t_n, t_{n+1}]$, then the local truncation error is:*

$$\tau_{n+1} = y(t_{n+1}) - y(t_n) - hf(t_n, y(t_n)) = \frac{h^2}{2} y''(\xi_n)$$

for some $\xi_n \in (t_n, t_{n+1})$.

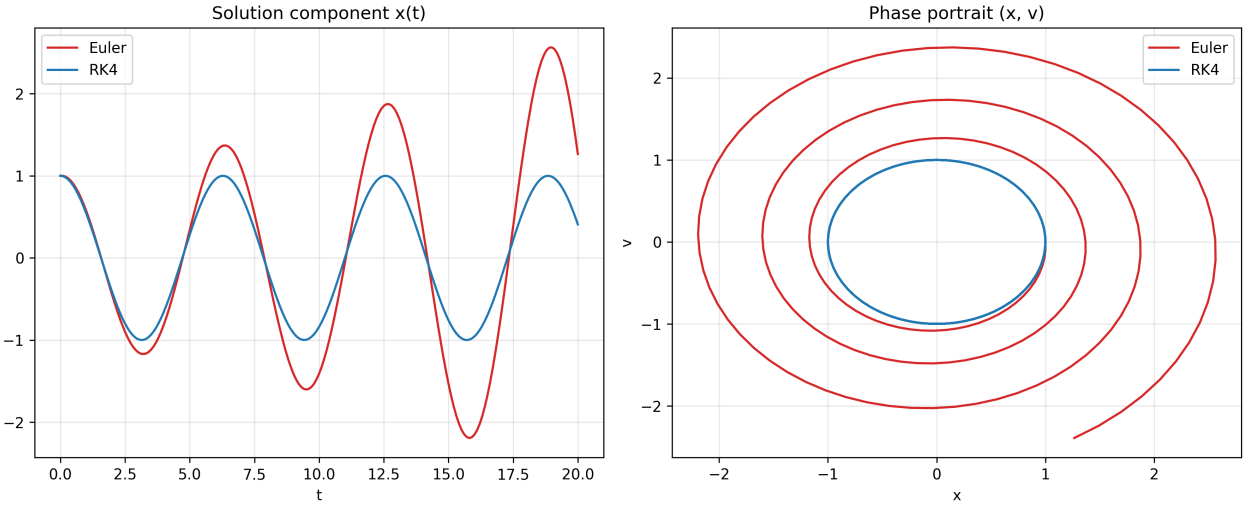

Figure 2: Comparison between Forward Euler and RK4 on a harmonic oscillator. Left: solution component $x(t)$. Right: phase portrait (x, v).

2.2 Backward Euler Method

Definition. Backward Euler Method

An implicit method:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

This requires solving a (generally nonlinear) equation at each step.

2.3 Improved Euler Method

Definition. Improved Euler (Heun's) Method

A predictor-corrector method:

$$\tilde{y}_{n+1} = y_n + hf(t_n, y_n) \quad \text{(predictor)} \tag{5}$$

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})] \quad \text{(corrector)} \tag{6}$$

3 Runge-Kutta Methods

Runge-Kutta methods achieve higher accuracy by evaluating the right-hand side at multiple points within each step.

3.1 Second-Order Runge-Kutta

Definition. RK2 Method

The general second-order Runge-Kutta method:

$$k_1 = f(t_n, y_n) \tag{7}$$
$$k_2 = f(t_n + \alpha h, y_n + \beta h k_1) \tag{8}$$
$$y_{n+1} = y_n + h(a_1 k_1 + a_2 k_2) \tag{9}$$

where parameters satisfy $a_1 + a_2 = 1$, $a_2\alpha = 1/2$, $a_2\beta = 1/2$.

3.2 Classical Fourth-Order Runge-Kutta

Definition. RK4 Method

The most popular Runge-Kutta method:

$$k_1 = f(t_n, y_n) \tag{10}$$
$$k_2 = f(t_n + h/2, y_n + hk_1/2) \tag{11}$$
$$k_3 = f(t_n + h/2, y_n + hk_2/2) \tag{12}$$
$$k_4 = f(t_n + h, y_n + hk_3) \tag{13}$$
$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{14}$$

Theorem 3.1 (RK4 Local Truncation Error)**.** *The classical RK4 method has local truncation error $O(h^5)$, making it a fourth-order method.*

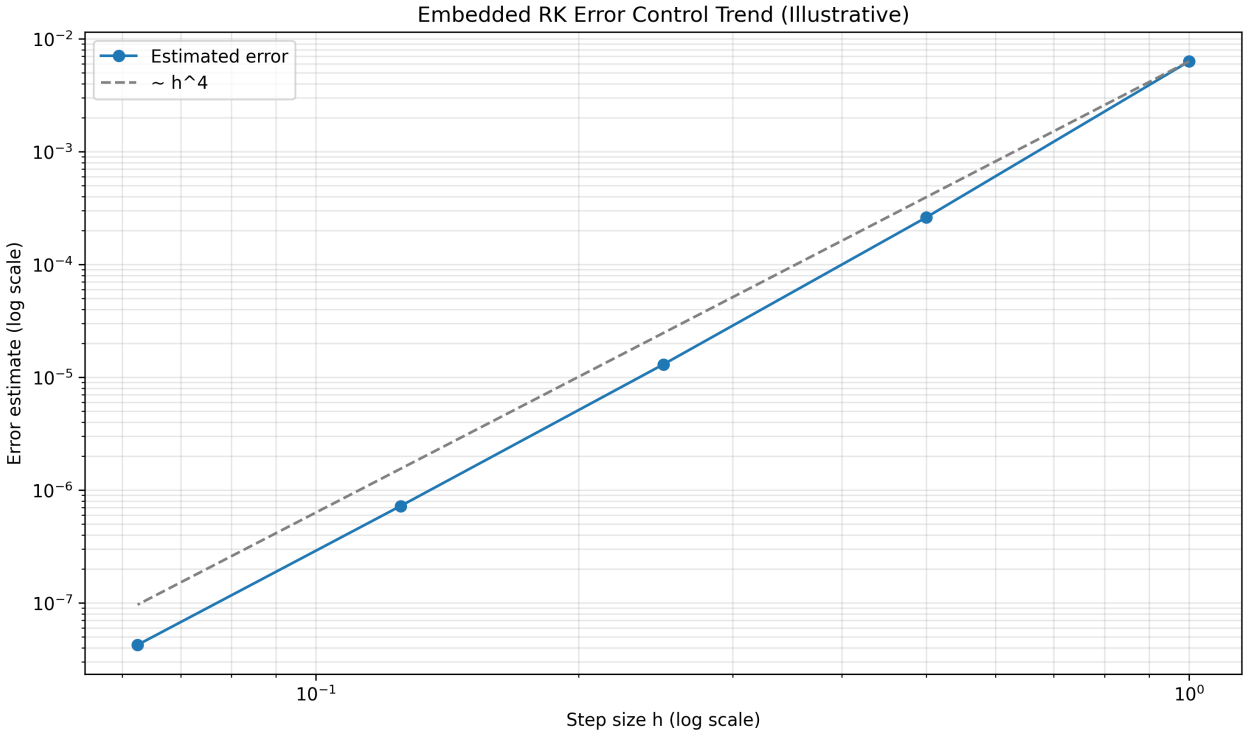

Figure 3: Embedded Runge–Kutta illustrative error trend showing $\sim h^4$ behavior for a fourth-order method.

3.3 Butcher Tableaux

Definition. **Butcher Tableau**

A Runge-Kutta method can be represented by its Butcher tableau:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

where $c_i = \sum_{j=1}^{s} a_{ij}$, and the method is:

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} k_j\right) \tag{15}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \tag{16}$$

4 Multistep Methods

Multistep methods use information from several previous points to achieve higher accuracy.

4.1 Adams Methods

Definition. **Adams-Bashforth Methods**

Explicit multistep methods based on polynomial extrapolation:

$$y_{n+1} = y_n + h\sum_{j=0}^{k-1} \beta_j f(t_{n-j}, y_{n-j})$$

Definition. **Adams-Moulton Methods**

Implicit multistep methods:

$$y_{n+1} = y_n + h\sum_{j=0}^{k-1} \beta_j^* f(t_{n+1-j}, y_{n+1-j})$$

4.2 Backward Differentiation Formulas

Definition. **BDF Methods**

Implicit methods particularly suited for stiff problems:

$$\sum_{j=0}^{k} \alpha_j y_{n+1-j} = h f(t_{n+1}, y_{n+1})$$

The coefficients α_j are chosen so that the method has maximum order of accuracy.

5 Stability Analysis

Stability is crucial for reliable numerical integration, especially for stiff problems.

5.1 Linear Stability Analysis

Definition. Test Equation

The Dahlquist test equation:

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

where $\lambda \in \mathbb{C}$ with $\text{Re}(\lambda) \leq 0$ for stability.

Definition. Stability Function

For a numerical method applied to the test equation, the stability function $R(z)$ is defined by:

$$y_{n+1} = R(h\lambda) y_n$$

where $z = h\lambda$.

5.2 Stability Regions

Definition. Stability Region

The stability region S is the set of complex numbers z for which $|R(z)| \leq 1$.

Theorem 5.1 (A-Stability)**.** *A method is A-stable if its stability region contains the entire left half-plane:* $S \supseteq \{z \in \mathbb{C} : Re(z) \leq 0\}$.

Theorem 5.2 (Dahlquist Barriers)**.**

- *No explicit multistep method can be A-stable*
- *The maximum order of an A-stable multistep method is 2*
- *A-stable multistep methods of order 2 are unique (trapezoidal rule)*

5.3 Stiff Differential Equations

Definition. Stiffness

A system is stiff if it contains solution components with vastly different time scales, requiring very small step sizes for explicit methods to maintain stability.

Intuition: Why Stiffness Matters

Stiff problems have fast-decaying transient components that don't affect the long-term solution but severely restrict step sizes for explicit methods. Implicit methods can take much larger steps by being unconditionally stable.

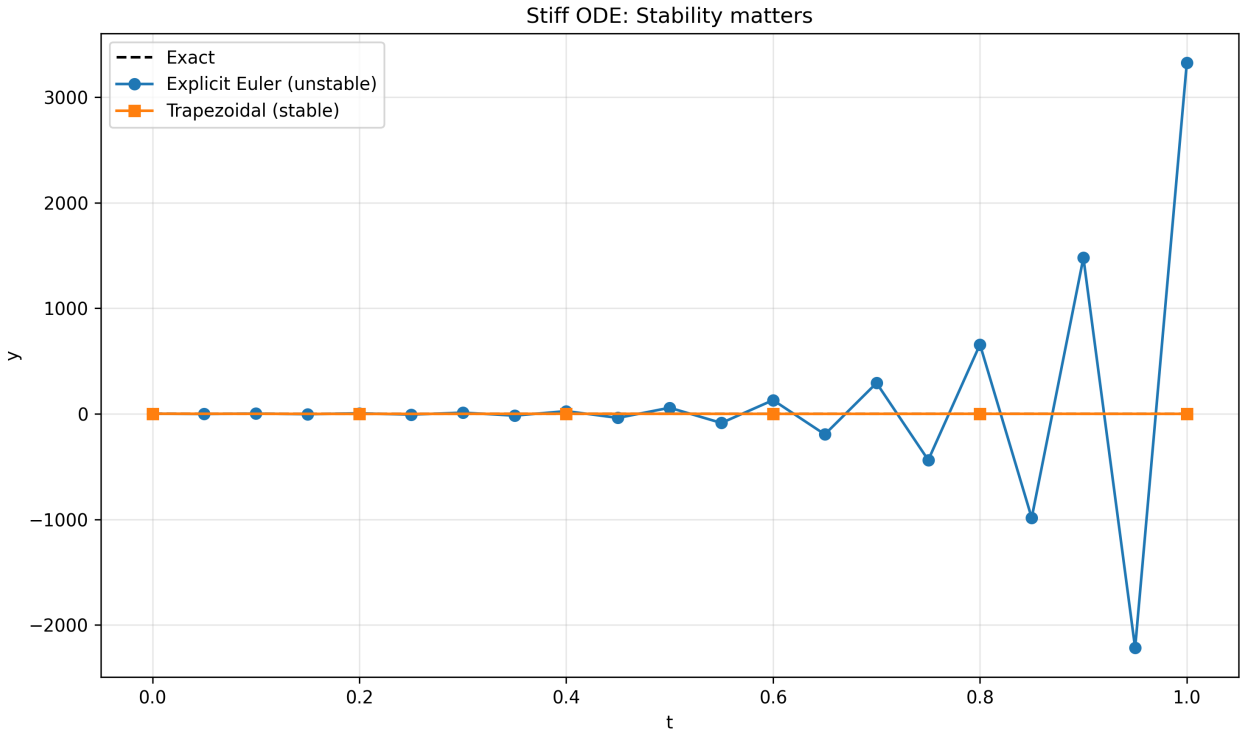

Figure 4: Stiff ODE example: explicit Euler becomes unstable for large $h\lambda$, while Trapezoidal remains stable.

6 Error Analysis and Convergence

6.1 Global Error Analysis

Theorem 6.1 (Global Error for One-Step Methods)**.** *For a consistent one-step method with local truncation error $O(h^{p+1})$ applied to a Lipschitz continuous problem:*

$$\|y(t_n) - y_n\| \leq \frac{Ch^p}{L}(e^{L(t_n - t_0)} - 1)$$

where C depends on bounds on the solution derivatives.

6.2 Convergence of Multistep Methods

Theorem 6.2 (Dahlquist Equivalence Theorem)**.** *For a multistep method, the following are equivalent:*

1. *The method is convergent*
2. *The method is consistent and zero-stable*

Definition. Zero-Stability

A multistep method is zero-stable if the roots of its characteristic polynomial $\rho(r) = \sum_{j=0}^{k} \alpha_j r^j$ satisfy $|r_i| \leq 1$, with simple roots on the unit circle.

7 Adaptive Methods

Adaptive methods automatically adjust step size to maintain accuracy while minimizing computational cost.

7.1 Error Estimation

Definition. Embedded Runge-Kutta Methods

Use two RK methods of different orders with the same function evaluations:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \quad \text{(order } p\text{)} \tag{17}$$

$$\hat{y}_{n+1} = y_n + h\sum_{i=1}^{s} \hat{b}_i k_i \quad \text{(order } p+1\text{)} \tag{18}$$

The difference $\hat{y}_{n+1} - y_{n+1}$ estimates the local error.

7.2 Step Size Control

Definition. PI Controller for Step Size

A proportional-integral controller for step size adjustment:

$$h_{n+1} = h_n \left(\frac{\text{TOL}}{\text{ERR}_n}\right)^{k_P} \left(\frac{\text{ERR}_{n-1}}{\text{ERR}_n}\right)^{k_I}$$

where TOL is the desired tolerance and ERR_n is the estimated error.

8 Systems of ODEs and Higher-Order Equations

8.1 Systems of First-Order ODEs

Definition. System of ODEs

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)) \tag{19}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0 \tag{20}$$

where $\mathbf{y}, \mathbf{f} \in \mathbb{R}^d$.

All methods extend naturally to systems by treating $\mathbf{y}$ as a vector.

8.2 Higher-Order ODEs

Definition. Reduction to First-Order System

The n-th order ODE $y^{(n)} = f(t, y, y', \ldots, y^{(n-1)})$ can be written as:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}' = \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ f(t, y_1, y_2, \ldots, y_n) \end{pmatrix}$$

9 Boundary Value Problems

9.1 Shooting Method

Definition. Shooting Method

Convert the BVP to an IVP by guessing initial conditions and adjusting until boundary conditions are satisfied.

9.2 Finite Difference Methods

Definition. Finite Difference for BVPs

Discretize the domain and replace derivatives with finite differences, leading to a system of algebraic equations.

Practice Problems

1. Derive the stability function for the forward Euler method and determine its stability region.
2. Prove that the trapezoidal rule (implicit midpoint method) is A-stable.
3. Implement the classical RK4 method and verify its fourth-order convergence rate.
4. Analyze the stability properties of the Adams-Bashforth methods of orders 1-4.
5. Design an adaptive RK method using embedded formulas and test it on both smooth and stiff problems.