Chapter 3

Numerical Solution of Differential Equations

Euler, Heun, and Runge–Kutta Methods

3.1 Introduction

Differential equations describe a wide range of phenomena in science and engineering, such as motion, heat transfer, and population dynamics. However, analytical solutions are not always attainable, which motivates the development of numerical methods. We consider the initial value problem (IVP)

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{3.1}$$

where $f(t, y)$ is known, and we wish to compute $y(t)$ for $t > t_0$. Let h be the step size, and define discrete time points:

$$t_n = t_0 + nh, \quad n = 0, 1, 2, \ldots$$

We seek numerical approximations $y_n \approx y(t_n)$.

3.2 Euler Method

3.2.1 Mathematical Derivation

From the Taylor expansion of $y(t)$ around t_n:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi_n)$$

Neglecting higher-order terms and using $y'(t_n) = f(t_n, y_n)$, we obtain:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{3.2}$$

This is the **explicit Euler method**.

3.2.2 Error Analysis

The local truncation error (LTE) is:

$$\tau_{n+1} = \frac{h^2}{2}y''(\xi_n) = O(h^2)$$

Since the global error accumulates over $O(1/h)$ steps, the global error is $O(h)$. Hence, Euler's method is a **first-order method**.

3.2.3 Stability

For the linear test equation $y' = \lambda y$:

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

The numerical solution is stable when $|1+h\lambda| \leq 1$. Thus, Euler's method is conditionally stable, and small step sizes are required when $\lambda < 0$.

3.2.4 Geometrical Interpretation

Euler's method approximates the true solution by moving along the tangent line at each step. The slope $f(t_n, y_n)$ gives the direction of motion, and the method advances by a distance h along this tangent.

3.2.5 Example 1

$$\frac{dy}{dt} = y - t^2 + 1, \quad y(0) = 0.5, \quad h = 0.2$$

The exact solution is $y(t) = (t + 1)^2 - 0.5e^t$. A few computed steps yield:

n	t_n	y_n (Euler)	Exact $y(t_n)$
0	0.0	0.500	0.500
1	0.2	0.800	0.8293
2	0.4	1.152	1.2141

Table 3.1: Euler method results for Example 1.

3.2.6 Example 2

$$\frac{dy}{dt} = -2y + e^{-t}, \quad y(0) = 1$$

Using $h = 0.1$, numerical results converge slowly to the decaying exponential behavior of the exact solution $y(t) = e^{-2t} + \frac{1}{3}(e^{-t} - e^{-2t})$.

3.3 Heun's Method (Improved Euler)

3.3.1 Derivation

Starting from the integral form:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt$$

Approximating the integral by the trapezoidal rule gives:

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right]$$

Since y_{n+1} is unknown, a predictor–corrector form is used:

$$y^*_{n+1} = y_n + hf(t_n, y_n) \quad \text{(Predictor)}$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\right] \quad \text{(Corrector)}$$

3.3.2 Order of Accuracy

Expanding by Taylor series and comparing with the true solution gives:

$$\text{Local error } = O(h^3), \quad \text{Global error } = O(h^2)$$

Thus, Heun's method is a **second-order accurate** method.

3.3.3 Stability Analysis

For $y' = \lambda y$, the amplification factor becomes:

$$R(z) = 1 + z + \frac{z^2}{2}, \quad z = h\lambda$$

This provides a larger stability region than Euler's method.

3.3.4 Interpretation

Heun's method improves on Euler's by averaging the slopes at the start and end of each interval, producing a more accurate trajectory.

3.3.5 Example 1

$$\frac{dy}{dt} = y - t^2 + 1, \quad y(0) = 0.5, \quad h = 0.2$$

n	t_n	y_n (Heun)	Exact $y(t_n)$
0	0.0	0.500	0.500
1	0.2	0.818	0.8293
2	0.4	1.163	1.2141

Table 3.2: Heun's method results for Example 1.

3.3.6 Example 2

$$\frac{dy}{dt} = -2y + e^{-t}, \quad y(0) = 1$$

Heun's method rapidly converges with $h = 0.1$, producing much smaller global error than Euler's method.

3.4 Runge–Kutta Method (Fourth Order)

3.4.1 Derivation

Runge–Kutta methods evaluate multiple slopes within each step. The classical 4th-order Runge–Kutta (RK4) method computes:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \end{aligned}$$

The update formula is:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.3}$$

3.4.2 Error and Accuracy

Expanding $y(t_{n+1})$ and matching terms shows:

$$\text{Local error } = O(h^5), \quad \text{Global error } = O(h^4)$$

Thus RK4 is a **fourth-order method** and provides high accuracy even with large step sizes.

3.4.3 Stability

For $y' = \lambda y$:

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$$

This polynomial approximates e^z, giving RK4 a large stability region covering most of the left half-plane.

3.4.4 Interpretation

The slopes k_1, k_2, k_3, k_4 correspond respectively to the beginning, two midpoints, and the end of the interval. Their weighted combination provides a fourth-order accurate approximation of the integral of $f(t, y)$.

3.4.5 Example 1

$$\frac{dy}{dt} = y - t^2 + 1, \quad y(0) = 0.5, \quad h = 0.2$$

RK4 yields extremely accurate results compared to the exact solution.

3.4.6 Example 2

$$\frac{dy}{dt} = -2y + e^{-t}, \quad y(0) = 1$$

After a few steps, the RK4 solution matches the exact function within numerical precision.

3.5 Comparison of Methods

Method	Local Error	Global Error	Order of Accuracy
Euler	$O(h^2)$	$O(h)$	1
Heun (Improved Euler)	$O(h^3)$	$O(h^2)$	2
Runge–Kutta (4th)	$O(h^5)$	$O(h^4)$	4

Table 3.3: Comparison of error orders for Euler, Heun, and Runge–Kutta methods.

3.6 Conclusion

In this chapter, we presented three fundamental approaches for solving first-order ODEs numerically:

- The **Euler method**, simple but only first-order accurate.
- The **Heun method**, a second-order improvement using slope averaging.
- The **Runge–Kutta (4th order)** method, providing excellent accuracy and stability.

The analysis shows that higher-order methods yield significantly smaller global errors for the same step size. Among these, the RK4 method remains the most commonly used in practice due to its balance of efficiency and precision.

3.7 Python Implementation (Loop-Based)

The Python code below is provided for illustrative purposes only.

```
""" Dr. Samir Kenouche - 26/10/2025 """

import numpy as np
import matplotlib.pyplot as plt
import random

# Define the differential equation
def f(t, y):
    return y - t**2 + 1

# Exact analytical solution
def exact_solution(t):
    return (t + 1)**2 - 0.5 * np.exp(t)

# Step size and range
t0, y0 = 0, 0.5
h = 0.2
t_end = 2
```

```
N = int((t_end - t0) / h)

# --- Euler Method ---
def euler(f, t0, y0, h, N):
    t = np.zeros(N+1)
    y = np.zeros(N+1)
    t[0], y[0] = t0, y0
    for n in range(N):
        y[n+1] = y[n] + h * f(t[n], y[n])
        t[n+1] = t[n] + h
    return t, y
# --- Heun Method ---
def heun(f, t0, y0, h, N):
    t = np.zeros(N+1)
    y = np.zeros(N+1)
    t[0], y[0] = t0, y0
    for n in range(N):
        y_pred = y[n] + h * f(t[n], y[n])
        y[n+1] = y[n] + (h/2) * (f(t[n], y[n]) + f(t[n+1], y_pred))
        t[n+1] = t[n] + h
    return t, y

# --- Runge_Kutta 4th Order Method ---
def rk4(f, t0, y0, h, N):
    t = np.zeros(N+1)
    y = np.zeros(N+1)
    t[0], y[0] = t0, y0
    for n in range(N):
        k1 = f(t[n], y[n])
        k2 = f(t[n] + h/2, y[n] + h*k1/2)
        k3 = f(t[n] + h/2, y[n] + h*k2/2)
        k4 = f(t[n] + h, y[n] + h*k3)
        y[n+1] = y[n] + (h/6)*(k1 + 2*k2 + 2*k3 + k4)
        t[n+1] = t[n] + h
    return t, y

""" Compute solutions
t_euler, y_euler = euler(f, t0, y0, h, N)
t_heun, y_heun = heun(f, t0, y0, h, N)
t_rk4, y_rk4 = rk4(f, t0, y0, h, N)

# Exact solution (dense grid for smooth curve)
t_exact = np.linspace(t0, t_end, 200)
y_exact = exact_solution(t_exact) """

""" Exact solution (dense grid for smooth curve)
t_exact = np.linspace(t0, t_end, 200)
y_exact = exact_solution(t_exact) """
```

3.8 Additional Numerical Examples

3.8.1 Additional Numerical Examples (Euler Method)

Example E-1: Stiff Test Problem Consider the initial value problem:

$$\frac{dy}{dt} = -5y, \qquad y(0) = 1.$$

The analytical solution is:

$$y(t) = e^{-5t}.$$

Using the Euler method with step size $h = 0.2$:

$$y_{n+1} = y_n + h(-5y_n) = y_n(1 - 5h).$$

We have $y_{n+1} = y_n(1 - 1) = 0$. Thus:

$$y_1 = 0, \quad y_2 = 0, \quad \ldots$$

while the exact values are:

$$y(0.2) = e^{-1} = 0.367879, \quad y(0.4) = e^{-2} = 0.135335.$$

Comment: Euler's method fails completely for this stiff problem, since $1 + h\lambda = 0$ causes numerical instability.

Example E-2: Linear Nonhomogeneous Equation Consider:

$$\frac{dy}{dt} = t + y, \qquad y(0) = 0.$$

Exact solution:

$$y(t) = -t - 1 + e^t.$$

Using $h = 0.1$:

t_n	y_n	$f(t_n, y_n)$	$y_{n+1} = y_n + hf(t_n, y_n)$
0.0	0.0000	0.0000	0.0000
0.1	0.0000	0.1000	0.0100
0.2	0.0100	0.2100	0.0310

Exact values:

$$y(0.1) = 0.00517, \quad y(0.2) = 0.02140.$$

Comment: The method is stable but exhibits noticeable $O(h)$ global error.

3.8.2 Additional Numerical Examples (Heun Method)

Example H-1: Logistic Equation Consider the nonlinear IVP:

$$\frac{dy}{dt} = y(1-y), \qquad y(0) = 0.1.$$

Exact solution:

$$y(t) = \frac{1}{1+9e^{-t}}.$$

Using $h = 0.25$, Heun's predictor–corrector form:

$$y^*_{n+1} = y_n + hf(t_n, y_n),$$
$$y_{n+1} = y_n + \frac{h}{2}\big(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\big).$$

t_n	y_n	$f(t_n, y_n)$	y^*_{n+1}	y_{n+1}
0.0	0.1000	0.0900	0.1225	0.1247

Exact value:

$$y(0.25) = 0.1248.$$

Comment: Heun's method provides an error below 2×10^{-4}, a substantial improvement over Euler.

Example H-2: Linear Benchmark Problem Consider:

$$\frac{dy}{dt} = y - t^2 + 1, \qquad y(0) = 0.5.$$

Exact solution:

$$y(t) = (t+1)^2 - 0.5e^t.$$

Using $h = 0.2$, Heun's method significantly improves accuracy over Euler, exhibiting global error $O(h^2)$.

Additional Numerical Examples (Runge–Kutta 4th Order Method)

Example R-1: Linear Nonhomogeneous Equation Consider:

$$\frac{dy}{dt} = \sin t - y, \qquad y(0) = 0.$$

Exact solution:

$$y(t) = \frac{1}{2}\big(\sin t - \cos t + e^{-t}\big).$$

Using $h = 0.1$:

$$k_1 = \sin(0) - 0 = 0,$$
$$k_2 = \sin(0.05) - 0 = 0.04998,$$
$$k_3 = \sin(0.05) - 0.00250 = 0.04748,$$
$$k_4 = \sin(0.1) - 0.00475 = 0.09509,$$
$$y_1 = y_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.0048334.$$

Exact $y(0.1) = 0.0048333$.

Comment: RK4 achieves an error on the order of 10^{-8} in a single step, confirming its fourth-order accuracy.

Example R-2: Simple Integrable Function

$$\frac{dy}{dt} = \cos t, \qquad y(0) = 0.$$

Exact solution: $y(t) = \sin t$.

With $h = 0.25$:

$$\begin{aligned} k_1 &= \cos 0 = 1.0000, \\ k_2 &= \cos(0.125) = 0.9922, \\ k_3 &= \cos(0.125) = 0.9922, \\ k_4 &= \cos(0.25) = 0.9689, \\ y_1 &= \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.247404. \end{aligned}$$

Exact $y(0.25) = \sin(0.25) = 0.247404$.

Comment: RK4 yields an error less than 10^{-7}, even for moderate step size.

Chapter A

Taylor-Series Derivation of Numerical Methods

A.1 Taylor Expansion of the True Solution

For a smooth function $y(t)$, expanding around t_n gives:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2!}y''(t_n) + \frac{h^3}{3!}y^{(3)}(t_n) + \cdots \tag{A.1}$$

Using $y' = f(t, y)$, higher derivatives can be expressed as:

$$\begin{aligned} y'' &= f_t + f_y f, \\ y^{(3)} &= f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_y(f_t + f_y f), \quad \text{etc.} \end{aligned}$$

These expansions are the foundation for analyzing truncation errors and constructing higher-order schemes.

A.2 Euler Method via Taylor Truncation

If we keep only the first derivative term:

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

The neglected terms start with $\frac{h^2}{2}y''(t_n)$, giving local truncation error $O(h^2)$.

A.3 Heun Method from Taylor Consistency

The true Taylor expansion to second order is:

$$y(t_{n+1}) = y_n + hf_n + \frac{h^2}{2}(f_t + f_y f_n) + O(h^3)$$

The Heun formula is:

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f^*_{n+1})$$

where $f^*_{n+1} = f(t_n + h, y_n + hf_n)$. Expanding f^*_{n+1} in Taylor series and comparing coefficients shows that all terms up to $O(h^2)$ match the exact Taylor series, confirming second-order accuracy.

A.4 Runge–Kutta (4th Order) Matching Conditions

For RK4, we require that its expansion reproduces the true Taylor series of $y(t_{n+1})$ up to $O(h^4)$. Expanding each k_i in powers of h and matching terms with the Taylor expansion yields the coefficients:

$$b_1 = \frac{1}{6}, \quad b_2 = \frac{1}{3}, \quad b_3 = \frac{1}{3}, \quad b_4 = \frac{1}{6}$$

and

$$a_2 = a_3 = \frac{1}{2}, \quad a_4 = 1$$

This ensures fourth-order consistency.

A.5 Summary of Order Conditions

For a general Runge–Kutta method:

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

with

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

the order conditions are derived by matching Taylor terms:

$$\sum b_i = 1, \quad \sum b_i c_i = \frac{1}{2}, \quad \sum b_i c_i^2 = \frac{1}{3}, \quad \sum b_i c_i^3 = \frac{1}{4}, \text{ etc.}$$

The classical RK4 coefficients satisfy these up to fourth order.

A.6 Conclusion of Appendix

Taylor series analysis provides the theoretical foundation for designing and verifying the accuracy of numerical methods for differential equations. It reveals how truncating at different orders directly determines the accuracy of the resulting method.