

UNIT 5

Solution of Ordinary Differential Equations

(7 Hours) — Numerical Methods (CAC252)

5.1 Introduction to Differential Equations

A **differential equation** relates an unknown function to its derivatives. An **ordinary differential equation (ODE)** involves derivatives with respect to a single independent variable. Its **order** is the highest derivative that appears. Most physical laws (motion, circuits, growth and decay) are ODEs, and most of them have no closed-form solution, so numerical methods are needed.

5.1.1 Initial Value Problem (IVP)

A first-order **initial value problem** is

$$dy/dx = f(x, y), y(x_0) = y_0$$

If f and $\partial f/\partial y$ are continuous near (x_0, y_0) , a unique solution exists (Picard–Lindelöf theorem). Numerical methods do not give a formula for $y(x)$; they give approximate values $y_1, y_2, \dots$ at the grid points $x_n = x_0 + n \cdot h$, where h is the **step size**.

Running example. Throughout this unit we use $y' = x + y$, $y(0) = 1$, whose exact solution is $y = 2e^x - x - 1$, so that every method can be checked against the truth (e.g. $y(0.1) = 1.110342$, $y(0.5) = 1.797443$).

Type of method	Uses	Examples
Series / analytic-style	Derivatives of y at x_0	Taylor series, Picard's iteration
Single-step (one-step)	Only the previous point (x_n, y_n)	Euler, Heun, Runge–Kutta
Multi-step	Several previous points	Adams–Bashforth, Milne (beyond this unit)

5.2 Taylor Series Method

Expand $y(x)$ about x_0 : $y(x) = y_0 + (x - x_0)y_0' + [(x - x_0)^2/2!]y_0'' + [(x - x_0)^3/3!]y_0''' + \dots$. The derivatives are obtained by repeatedly differentiating $y' = f(x, y)$ (using the chain rule) and substituting x_0, y_0 . Truncating after the term of order m gives a method whose truncation error is $O(h^{m+1})$.

Example 5.1

Find $y(0.1)$ and $y(0.2)$ from $y' = x + y$, $y(0) = 1$ by the Taylor series method (terms up to y'''').

Solution

$$y' = x + y \Rightarrow y'(0) = 1. \quad y'' = 1 + y' \Rightarrow y''(0) = 2. \quad y''' = y'' \Rightarrow 2. \quad y'''' = y''' \Rightarrow 2.$$

$$y(x) = 1 + x + 2x^2/2! + 2x^3/3! + 2x^4/4! = 1 + x + x^2 + x^3/3 + x^4/12$$

$$y(0.1) = 1.110342 \text{ (exact 1.110342)}$$

$$y(0.2) = 1.242800 \text{ (exact 1.242806)}$$

Note: The Taylor method is very accurate near x_0 but needs analytic derivatives of f , which can be laborious. For that reason Runge–Kutta methods (which need only values of f) are preferred in practice.

5.3 Picard's Method of Successive Approximation

Integrating $y' = f(x, y)$ from x_0 to x gives the integral equation $y(x) = y_0 + \int_{x_0}^x f(t, y(t)) dt$ (from x_0 to x). Picard's method iterates

$$y^{(0)}(x) = y_0, \quad y^{(n+1)}(x) = y_0 + \int_{x_0}^x f(t, y^{(n)}(t)) dt \text{ (limits } x_0 \text{ to } x)$$

Under the Lipschitz condition the sequence converges to the true solution. Each step is an analytic integration, so the method is practical only when the integrals are easy.

Example 5.2

Use Picard's method to find $y^{(1)}, y^{(2)}, y^{(3)}$ for $y' = x + y$, $y(0) = 1$, and evaluate at $x = 0.1$ and 0.2 .

Solution

$$y^{(0)} = 1$$

$$y^{(1)} = 1 + \int_0^x (t + 1) dt = 1 + x + x^2/2$$

$$y^{(2)} = 1 + \int_0^x (t + 1 + t + t^2/2) dt = 1 + x + x^2 + x^3/6$$

$$y^{(3)} = 1 + \int_0^x (t + 1 + t + t^2 + t^3/6) dt = 1 + x + x^2 + x^3/3 + x^4/24$$

x	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	Exact
0.1	1.105000	1.110167	1.110337	1.110342
0.2	1.220000	1.241333	1.242733	1.242806

Each iteration adds one more correct Taylor term: $y^{(3)}$ agrees with the exact series $1 + x + x^2 + x^3/3 + x^4/12 + \dots$ up to the x^3 term.

5.4 Euler's Method and Its Accuracy

Euler's method follows the tangent line for one step of length h :

$$y_{n+1} = y_n + h \cdot f(x_n, y_n), \quad x_{n+1} = x_n + h$$

It is the first two terms of the Taylor series, so the **local truncation error** (error in one step) is $O(h^2)$, namely $(h^2/2) y''(\xi)$. Errors accumulate over about $(b - a)/h$ steps, giving a **global error of $O(h)$** : halving h roughly halves the error. Euler's method is simple but needs very small steps; it is a first-order method.

Example 5.3

Solve $y' = x + y$, $y(0) = 1$ by Euler's method with $h = 0.1$ up to $x = 0.5$.

$$y_{n+1} = y_n + 0.1(x_n + y_n): y_1 = 1 + 0.1(0 + 1) = 1.1, y_2 = 1.1 + 0.1(0.1 + 1.1) = 1.22, \dots$$

n	x_n	Euler y_n	Exact $y(x_n)$	Error
0	0.0	1.00000	1.00000	0.00000
1	0.1	1.10000	1.11034	0.01034
2	0.2	1.22000	1.24281	0.02281
3	0.3	1.36200	1.39972	0.03772
4	0.4	1.52820	1.58365	0.05545
5	0.5	1.72102	1.79744	0.07642

Accuracy check at $x = 0.5$: error = 0.07642 for $h = 0.1$, 0.03965 for $h = 0.05$, 0.02021 for $h = 0.025$ — each halving of h about halves the error, confirming first-order accuracy.

5.5 Heun's Method (Improved / Modified Euler)

Euler uses the slope at the *start* of the interval. Heun's method averages the slope at the start and the slope at the end (estimated by an Euler predictor), which is the trapezoidal rule applied to the ODE:

Predictor: $y_{n+1}^* = y_n + h f(x_n, y_n)$

Corrector: $y_{n+1} = y_n + (h/2) [f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)]$

The local error is $O(h^3)$ and the global error $O(h^2)$ — a second-order method needing two evaluations of f per step. The corrector may be repeated (iterated) to improve the estimate further.

Example 5.4

Solve $y' = x + y$, $y(0) = 1$ by Heun's method with $h = 0.1$ up to $x = 0.5$.

n	x_n	y_n	$k_1 = f(x_n, y_n)$	y_{n+1}^*	$k_2 = f(x_{n+1}, y_{n+1}^*)$	y_{n+1}
0	0.0	1.000000	1.00000	1.10000	1.20000	1.110000
1	0.1	1.110000	1.21000	1.23100	1.43100	1.242050
2	0.2	1.242050	1.44205	1.38626	1.68626	1.398465
3	0.3	1.398465	1.69847	1.56831	1.96831	1.581804
4	0.4	1.581804	1.98180	1.77998	2.27998	1.794894

At $x = 0.5$ Heun gives 1.794894, the exact value is 1.797443, so the error is 0.002549 (Euler with the same h : 0.07642).

5.6 Runge–Kutta Methods

Runge–Kutta (RK) methods obtain the accuracy of a Taylor series of order m **without** differentiating f , by evaluating f at several carefully chosen points inside each step.

5.6.1 Second-order RK (midpoint form)

$k_1 = h f(x_n, y_n)$, $k_2 = h f(x_n + h/2, y_n + k_1/2)$, $y_{n+1} = y_n + k_2$. (Heun's method is another second-order RK method.)

5.6.2 Fourth-order RK (classical RK4)

$k_1 = h f(x_n, y_n)$

$k_2 = h f(x_n + h/2, y_n + k_1/2)$

$k_3 = h f(x_n + h/2, y_n + k_2/2)$

$k_4 = h f(x_n + h, y_n + k_3)$

$$y_{n+1} = y_n + (1/6)(k_1 + 2k_2 + 2k_3 + k_4)$$

The local error is $O(h^5)$ and the global error $O(h^4)$. It needs four function evaluations per step. The method is self-starting and easy to program, and is the workhorse of numerical ODE solving.

Example 5.5

Apply RK4 to $y' = x + y$, $y(0) = 1$ with $h = 0.1$. Show the first step and tabulate up to $x = 0.5$.

First step ($x_0 = 0$, $y_0 = 1$)

$$k_1 = 0.1(0 + 1) = 0.10000$$

$$k_2 = 0.1(0.05 + 1 + 0.05000) = 0.110000$$

$$k_3 = 0.1(0.05 + 1 + 0.055000) = 0.1105000$$

$$k_4 = 0.1(0.1 + 1 + 0.1105000) = 0.12105000$$

$$y_1 = 1 + (1/6)(0.10000 + 2(0.110000) + 2(0.1105000) + 0.12105000) = 1.1103417 \text{ (exact } 1.1103418)$$

n	x_n	k_1	k_2	k_3	k_4	y_{n+1}	Exact
0	0.1	0.10000	0.11000	0.11050	0.12105	1.1103417	1.1103418
1	0.2	0.12103	0.13209	0.13264	0.14430	1.2428051	1.2428055
2	0.3	0.14428	0.15649	0.15711	0.16999	1.3997170	1.3997176
3	0.4	0.16997	0.18347	0.18415	0.19839	1.5836485	1.5836494
4	0.5	0.19836	0.21328	0.21403	0.22977	1.7974413	1.7974425

At $x = 0.5$, RK4 gives 1.7974413 with an error of only 1.3×10^{-6} , versus 0.0764 for Euler at the same step size.

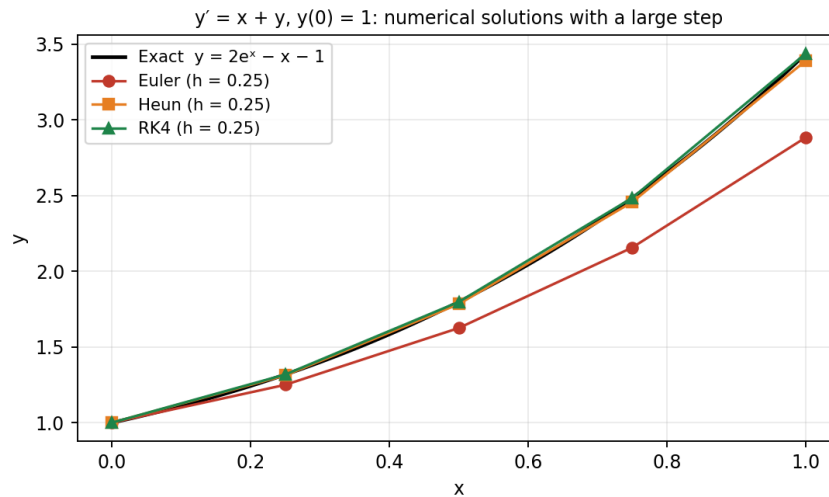


Figure 5.1 — With a coarse step $h = 0.25$ Euler drifts below the true curve while Heun and RK4 stay close.

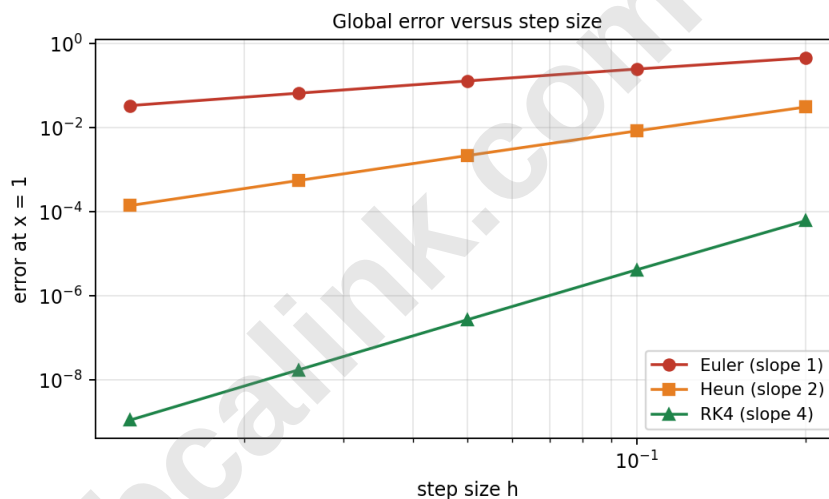


Figure 5.2 — Error at $x = 1$ versus step size: the slopes on the log-log plot are the orders 1, 2, 4.

Method	f-evaluations / step	Local error	Global error
Euler	1	$O(h^2)$	$O(h)$
Heun / RK2	2	$O(h^3)$	$O(h^2)$
RK4	4	$O(h^5)$	$O(h^4)$

5.7 Solution of Higher-Order Equations

Any higher-order ODE can be converted into a **system of first-order equations**, and then any of the above methods (applied to a vector) can be used. For $y'' = f(x, y, y')$ with $y(x_0) = y_0$ and $y'(x_0) = y_0'$, put $u = y$ and $v = y'$:

$$\mathbf{u}' = \mathbf{v}, \mathbf{v}' = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{v}), \mathbf{u}(\mathbf{x}_0) = \mathbf{y}_0, \mathbf{v}(\mathbf{x}_0) = \mathbf{y}_0'$$

For an equation of order m the same substitution gives m first-order equations. RK4 for the system uses vector k -values:

$k_1 = h F(x_n, w_n)$, $k_2 = h F(x_n + h/2, w_n + k_1/2)$, $k_3 = h F(x_n + h/2, w_n + k_2/2)$, $k_4 = h F(x_n + h, w_n + k_3)$, $w_{n+1} = w_n + (k_1 + 2k_2 + 2k_3 + k_4)/6$, where $w = (u, v)^T$.

Example 5.6

Solve $y'' + y = 0$, $y(0) = 0$, $y'(0) = 1$ by RK4 with $h = 0.1$ (exact solution $y = \sin x$, $y' = \cos x$).
Set $u = y$, $v = y'$: $u' = v$, $v' = -u$, $(u_0, v_0) = (0, 1)$.

First step

$$k_1 = 0.1 \cdot (v, -u) = 0.1 \cdot (1, 0) = (0.10000, 0)$$

$$k_2 = 0.1 \cdot (v + k_1/2, -(u + k_1/2)) = 0.1 \cdot (1 + 0, -(0 + 0.05000)) = (0.10000, -0.00500)$$

$$k_3 = 0.1 \cdot (1 - 0.00250, -(0 + 0.05000)) = (0.09750, -0.00500)$$

$$k_4 = 0.1 \cdot (1 - 0.00500, -(0 + 0.09750)) = (0.09500, -0.00975)$$

$$(u_1, v_1) = (0, 1) + (1/6)(k_1 + 2k_2 + 2k_3 + k_4) = (0.0998333, 0.9950042) \text{ [exact: } \sin 0.1 = 0.0998334, \cos 0.1 = 0.9950042]$$

x	$u = y$ (RK4)	$\sin x$	$v = y'$ (RK4)	$\cos x$
0.0	0.0000000	0.0000000	1.0000000	1.0000000
0.1	0.0998333	0.0998334	0.9950042	0.9950042
0.2	0.1986692	0.1986693	0.9800666	0.9800666
0.3	0.2955200	0.2955202	0.9553365	0.9553365
0.4	0.3894180	0.3894183	0.9210611	0.9210610
0.5	0.4794252	0.4794255	0.8775827	0.8775826

5.8 Boundary Value Problems

In an IVP all conditions are given at one point. In a **boundary value problem (BVP)** conditions are given at two points. The standard second-order two-point BVP is

$$y'' = f(x, y, y'), y(a) = \alpha, y(b) = \beta$$

Examples: the deflection of a beam, steady-state heat conduction. BVPs may have no solution, one, or many, and cannot be marched forward from a single point. Two families of numerical methods exist: **shooting methods** (this section) and finite-difference methods (which replace derivatives by difference quotients and solve a linear system; see Unit 6 for the same idea).

5.8.1 The Shooting Method

The idea: **convert the BVP into an IVP** by guessing the missing initial slope $s = y'(a)$, “shoot” with an IVP solver from $x = a$ to $x = b$, and adjust s until the far end hits the target $y(b) = \beta$. Define the **miss distance** $\varphi(s) = y(b; s) - \beta$; we need a root of $\varphi(s) = 0$.

Algorithm

- Guess two slopes s_0 and s_1 . Solve the IVP $y'' = f(x, y, y')$, $y(a) = \alpha$, $y'(a) = s$ for each (e.g. with RK4 on the equivalent system) and obtain $y(b; s_0)$, $y(b; s_1)$.
- Compute the miss distances $\varphi(s_0) = y(b; s_0) - \beta$ and $\varphi(s_1) = y(b; s_1) - \beta$.
- Update the slope with the secant formula: $s_{k+1} = s_k - \varphi(s_k) \cdot (s_k - s_{k-1}) / [\varphi(s_k) - \varphi(s_{k-1})]$.
- Solve the IVP with s_{k+1} . If $|\varphi(s_{k+1})| < \epsilon$, stop; otherwise repeat step 3. The last IVP solution is the BVP solution.

Linear BVP. If the ODE is linear, $\varphi(s)$ is a linear function of s , so the secant formula gives the exact slope after **just two trial shots**. Equivalently, $y = y_1(x) + [(\beta - y_1(b)) / (y_2(b) - y_1(b))] \cdot [y_2(x) - y_1(x)]$.

Note: For a nonlinear BVP, $\varphi(s)$ is nonlinear and several secant (or Newton) iterations are needed. A poor guess can make the IVP blow up, which is a known weakness of shooting; multiple shooting or finite differences are used for such problems.

Example 5.7

Solve the BVP $y'' = y$, $y(0) = 0$, $y(1) = 1$ by the shooting method (RK4, $h = 0.1$). Exact solution: $y = \sinh x / \sinh 1$.

Solution

Write $u = y$, $v = y'$: $u' = v$, $v' = u$, $u(0) = 0$, $v(0) = s$.

Trial 1: $s_0 = 0.5 \Rightarrow y(1; s_0) = 0.58760$, $\varphi(s_0) = -0.41240$

Trial 2: $s_1 = 1.5 \Rightarrow y(1; s_1) = 1.76280$, $\varphi(s_1) = 0.76280$

Secant update: $s = 0.5 - (-0.41240)(1.5 - 0.5) / (0.76280 - (-0.41240)) = 0.85092$

Exact slope $y'(0) = 1/\sinh 1 = 0.85092$, so one secant step already gives the correct slope (the problem is linear).

x	Shooting $y(x)$	Exact $\sinh x / \sinh 1$	Error
0.0	0.000000	0.000000	0
0.2	0.171320	0.171320	3.3×10^{-8}
0.4	0.349517	0.349517	6.1×10^{-8}

x	Shooting y(x)	Exact $\sinh x / \sinh 1$	Error
0.6	0.541740	0.541740	7.4×10^{-8}
0.8	0.755706	0.755705	5.8×10^{-8}
1.0	1.000000	1.000000	1.0×10^{-12}

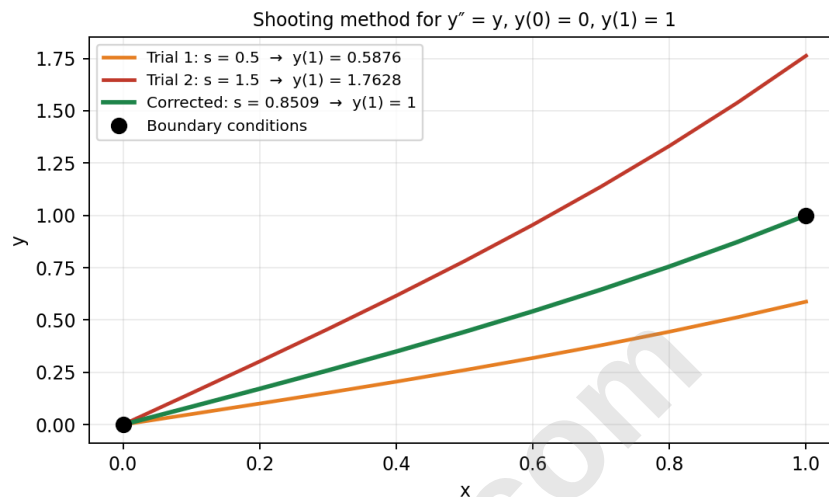


Figure 5.3 — Two trial trajectories overshoot / undershoot the target $y(1) = 1$; the corrected slope hits it.

Key Takeaways — Unit 5

- ✓ IVP: $y' = f(x, y)$, $y(x_0) = y_0$; numerical methods produce y_n at $x_n = x_0 + nh$.
- ✓ Taylor series uses derivatives of y at x_0 ; Picard iterates $y^{(n+1)} = y_0 + \int f(t, y^{(n)}) dt$.
- ✓ Euler: $y_{n+1} = y_n + h f(x_n, y_n)$; local error $O(h^2)$, global error $O(h)$.
- ✓ Heun: predictor-corrector average of two slopes; global error $O(h^2)$.
- ✓ RK4: $y_{n+1} = y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6$; global error $O(h^4)$; no derivatives of f are needed.
- ✓ Higher-order ODE $\Rightarrow$ system of first-order equations ($u = y$, $v = y'$, ...).
- ✓ Shooting method: guess $y'(a)$, solve the IVP, correct the slope with the secant formula until $y(b) = \beta$; linear BVPs need two shots.

— End of Unit 5 —