10. Ordinary differential equations

- ordinary differential equations
- Euler method
- Heun and midpoint methods
- Runge-Kutta methods
- systems of ODEs
- boundary value problems

Ordinary differential equations

differential equations are composed of an unknown function and its derivatives

Example: recall our derivation of the ODE for a falling parachutist velocity $v(t)$:

$$\frac{dv}{dt} = g - \frac{c}{m}\, v$$

- v is the *dependent* variable; t is the *independent* variable
  - one independent variable $\Rightarrow$ *ordinary differential equation* (ODE)
  - two or more $\Rightarrow$ *partial differential equation* (PDE)
- it is a *rate equation* because it gives a rate of change as a function of variables
- order = highest derivative present; in our case it is first order

Higher order and reduction to first order

higher-order equations can be reduced to a *system* of first-order equations

Example: a damped mass-spring position $x(t)$ satisfies the second-order ODE:

$$m\,\frac{d^2x}{dt^2} + c\,\frac{dx}{dt} + k\,x = 0$$

- reduce to system of first order by defining

$$y = \frac{dx}{dt} \qquad \Rightarrow \qquad \frac{dy}{dt} = \frac{d^2x}{dt^2}$$

- substitute into back, gives

$$\frac{dx}{dt} = y, \qquad \frac{dy}{dt} = -\frac{c\,y + k\,x}{m}$$

  this is a first-order system equivalent to original system

- we focus on first-order ODEs and first-order systems

Noncomputer methods and analytical solutions

- without computers, ODEs are solved via analytical integration
- for our previous example, multiplying by dt and integrating

$$v = \int \left(g - \frac{c}{m}\, v \right) dt$$

this is an *indefinite* integral (limits unspecified)

- for $v(0) = 0$, the closed-form solution is

$$v(t) = \frac{g\, m}{c} \left(1 - e^{-(c/m)\, t} \right)$$

- many practical ODEs lack exact solutions ⇒ numerical methods are essential

Linearization and linear ODEs

a general linear nth-order ODE has the form

$$a_n(x)\, y^{(n)} + \cdots + a_1(x)\, y' + a_0(x)\, y = f(x)$$

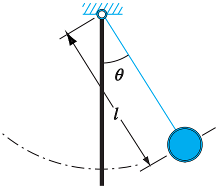
- linear means no products or nonlinear functions of y and its derivatives
- linear ODEs admit analytical solutions, whereas most nonlinear ODEs do not
- hence, *linearization* is a useful tactic

Example: nonlinear pendulum

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\, \sin\theta = 0$$

for small angles, $\sin\theta \approx \theta$,

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\, \theta = 0, \quad \text{which is linear}$$



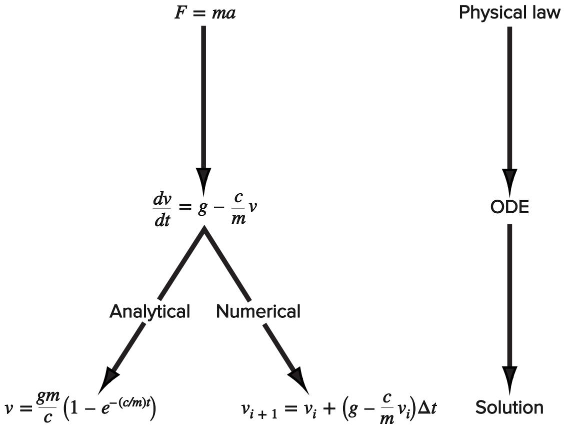
for large angles, linearization is invalid $\Longrightarrow$ use numerical methods

ODEs and engineering practice

Law	Mathematical Expression	Variables and Parameters
Newton's second law of motion	$\frac{dv}{dt} = \frac{F}{m}$	Velocity (v), force (F), and mass (m)
Fourier's heat law	$q = -k' \frac{dT}{dx}$	Heat flux (q), thermal conductivity (k') and temperature (T)
Fick's law of diffusion	$J = -D \frac{dc}{dx}$	Mass flux (J), diffusion coefficient (D), and concentration (c)
Faraday's law (voltage drop across an inductor)	$\Delta V_L = L \frac{di}{dt}$	Voltage drop (ΔV_L), inductance (L), and current (i)

- fundamental laws express how properties change over space and time
- these laws define *mechanisms of change*
- when combined with conservation (continuity) laws for energy, mass, or momentum, they lead to differential equations

Example



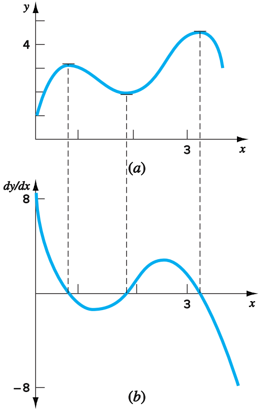
these models can be used for design (*e.g.*, determining parachute drag coefficient c to limit terminal velocity)

What is a solution of an ODE?

- a *solution* of an ODE is a specific function of the independent variable (and parameters) that satisfies the ODE for all points in its domain
- illustration starts from a given function (a fourth-order polynomial):

$$y = -0.5\,x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$



$y(x)$ gives the value of the function

dy/dx gives *slope* (rate of change) at each x

zeros of dy/dx occur where $y(x)$ is flat

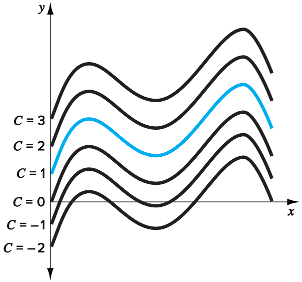
From the ODE back to the family of solutions

- given the ODE, retrieve a solution by integrating:

$$y = \int \big(-2x^3 + 12x^2 - 20x + 8.5 \big)\, dx$$

$$\Longrightarrow \quad y = -0.5\,x^4 + 4x^3 - 10x^2 + 8.5x + C$$

- note the *constant of integration* C
- differentiating then integrating *loses* the original additive constant ($+1$ became $+C$)
- conclusion: the ODE admits an *infinite family* of solutions parameterized by C



Selecting a unique solution with an initial condition

- to *pin down* C, specify an *auxiliary condition*
- for first-order ODEs, an *initial condition* suffices, *e.g.*,

$$\text{at} \quad x = 0 : \quad y = 1$$

- hence:

$$1 = -0.5(0)^4 + 4(0)^3 - 10(0)^2 + 8.5(0) + C \quad \Rightarrow \quad C = 1$$

$$\Rightarrow \quad y = -0.5\,x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

- this is the unique member of the family that satisfies both ODE and initial condition

ODEs and one-step methods

we consider ordinary differential equations of the form

$$\frac{dy}{dx} = f(x, y)$$

- *one-step methods:* advance the solution step by step using slope information

$$y_{i+1} = y_i + \phi\, h$$

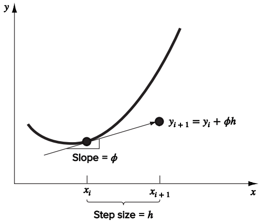
  - y_i: known approximate value at x_i
  - y_{i+1}: new value at $x_{i+1} = x_i + h$
  - h: step size
  - ϕ: estimate of the slope across the interval

- step-by-step application traces out the trajectory of the solution

Runge-Kutta methods

the simplest slope estimate: use the ODE directly at the beginning of the interval

$$\phi = f(x_i, y_i)$$



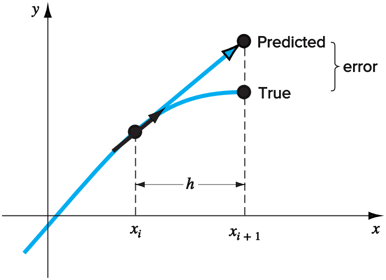
- this yields *Euler method*, the most basic one-step scheme
- limitation: slope may change significantly within the step $\longrightarrow$ reduced accuracy
- improved slope estimates $\longrightarrow$ more accurate methods
  - Heun method
  - midpoint method
  - classical Runge-Kutta (RK4)
- collectively called *Runge-Kutta methods*

Outline

- ordinary differential equations
- **Euler method**
- Heun and midpoint methods
- Runge-Kutta methods
- systems of ODEs
- boundary value problems

Euler method

$$y_{i+1} = y_i + f(x_i, y_i)\, h$$



- the first derivative gives a direct estimate of the slope at x_i: $\phi = f(x_i, y_i)$
- called **Euler method** (also called *Euler-Cauchy* or *point-slope method*)
- idea: use local slope at left endpoint (x_i, y_i) and extrapolate linearly over step h

Example

use Euler method to integrate

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$

from $x = 0$ to $x = 4$ with step size $h = 0.5$ and initial condition: $y(0) = y_0 = 1$

the exact solution is

$$y(x) = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

First step (to $x_1 = 0.5$)

$$y(0.5) \approx y_1 = y_0 + f(0,1)\,h = 1 + 8.5(0.5) = 5.25$$

true value at $x = 0.5$:

$$y(0.5) = -0.5(0.5)^4 + 4(0.5)^3 - 10(0.5)^2 + 8.5(0.5) + 1 = 3.21875$$

error:

$$E_t = 3.21875 - 5.25 = -2.03125, \qquad \varepsilon_t = \frac{-2.03125}{3.21875} \times 100\% \approx -63.1\%$$

Example

Second step (to $x_2 = 1.0$)

$$y(1.0) \approx y_2 = y_1 + f(0.5,\ y_1)\, h = 5.25 + (2.5)\,(0.5) = 5.875$$

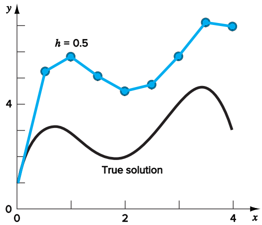
true value at $x = 1.0$:

$$y(1.0) = -0.5(1)^4 + 4(1)^3 - 10(1)^2 + 8.5(1) + 1 = 3.0$$

error:

$$E_t = 3.0 - 5.875 = -2.875$$

$$\varepsilon_t = \frac{-2.875}{3.0} \times 100\% \approx -95.8\%$$



- large errors arise because the true solution is highly curved
- Euler linear extrapolation over $h = 0.5$ is too crude
- accuracy improves with smaller h

Error sources in numerical ODEs

Two error classes

1. *truncation (discretization) error:*

   approximating continuous problem by discrete steps/formulas

2. *round-off error:*

   finite precision arithmetic limits significant digits carried by computer

Truncation error components

- *local truncation error (lte):* error made in a single step of the method
- *propagated truncation error:* accumulated error over many steps

Global truncation error = lte + propagated error

Interpretation of Euler method

$$y' = f(x, y), \qquad \text{with} \quad y = y(x)$$

- Taylor expansion of $y(x)$ at x_{i+1} about x_i is

$$y_{i+1} = y_i + y_i'\, h + \frac{y_i''}{2!} h^2 + \cdots + \frac{y_i^{(n)}}{n!} h^n + R_n$$

where $h = x_{i+1} - x_i$ and remainder $R_n = \frac{y^{(n+1)}(\xi)}{(n+1)!}\, h^{n+1}$ for some $\xi \in (x_i, x_{i+1})$

- using $y' = f(x, y)$, this becomes

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!} h^2 + \cdots + \frac{f^{(n-1)}(x_i, y_i)}{n!} h^n + O(h^{n+1})$$

where $f'(x, y) = \frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$

Euler update: keep only the first derivative term:

$$y_{i+1} = y_i + f(x_i, y_i)\, h$$

Local truncation error of Euler method

the **true** local truncation error (lte):

$$E_t = \frac{f'(x_i, y_i)}{2!}\, h^2 + \frac{f''(x_i, y_i)}{3!}\, h^3 + \cdots + O(h^{n+1})$$

for sufficiently small h, the leading term dominates, giving the **approximate** lte:

$$E_a \approx \frac{f'(x_i, y_i)}{2}\, h^2 = O(h^2)$$

- Euler method is *first-order accurate globally*
- lte $= O(h^2)$ per step, but global error typically $= O(h)$ over many steps
- round-off can dominate for very small h
- there is an optimal step size balancing truncation and round-off

Example

estimate the error of the first step of last example

determine the error due to each higher-order term of the Taylor expansion

- we have

$$E_t = \frac{f'(x_i, y_i)}{2!} h^2 + \frac{f''(x_i, y_i)}{3!} h^3 + \frac{f^{(3)}(x_i, y_i)}{4!} h^4$$

- for $x_0 = 0$, $h = 0.5$:

$$f'(x_i, y_i) = -6x^2 + 24x - 20 \quad E_{t,2} = \frac{-6(0)^2 + 24(0) - 20}{2}(0.5)^2 = -2.5$$

$$f''(x_i, y_i) = -12x + 24 \quad E_{t,3} = \frac{-12(0) + 24}{6}(0.5)^3 = 0.5$$

$$f^{(3)}(x_i, y_i) = -12 \quad E_{t,4} = \frac{-12}{24}(0.5)^4 = -0.03125$$

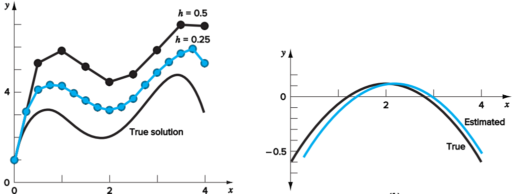
- total truncation error

$$E_t = -2.5 + 0.5 - 0.03125 = -2.03125$$

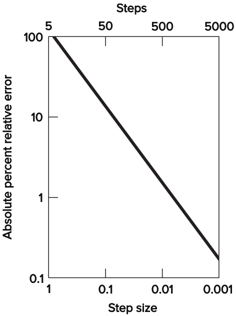
higher derivatives vanish since the ODE is a cubic polynomial

Example

reducing stepsize reduce error but increases computation



(a) (b)



Higher-order Taylor series methods

- using terms up to second order in Taylor expansion gives

$$y_{i+1} = y_i + f(x_i, y_i)\, h + \frac{f'(x_i, y_i)}{2}\, h^2$$

- local truncation error $E_a = \frac{f''(x_i, y_i)}{3!} h^3$
- if $f = f(x, y)$ the first derivative requires chain rule

$$f'(x_i, y_i) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\, \frac{dy}{dx}$$

- the second derivative becomes more complicated

$$f''(x_i, y_i) = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}\right) + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}\right)\frac{dy}{dx}$$

- higher derivatives increase in complexity rapidly
- therefore alternative one-step methods have been developed with accuracy comparable to high-order Taylor schemes

Outline

- ordinary differential equations
- Euler method
- **Heun and midpoint methods**
- Runge-Kutta methods
- systems of ODEs
- boundary value problems

Heun method

idea: average two slopes at beginning and end of interval,

$$y'_i = f(x_i, y_i), \quad y'_{i+1} = f(x_{i+1}, y^0_{i+1})$$

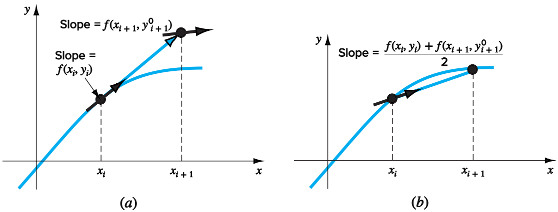
Heun method

1. predictor equation (Euler step):

$$y^0_{i+1} = y_i + f(x_i, y_i)h$$

2. corrector equation (average slope):

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2}h$$



Iterative corrector

corrector can be applied repeatedly:

$$y_{i+1}^{(j)} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(j-1)})}{2} h, \quad j = 1, 2, \ldots$$

- termination criterion:

$$|\varepsilon_a| = \left| \frac{y_{i+1}^{(j)} - y_{i+1}^{(j-1)}}{y_{i+1}^{(j)}} \right| 100\% < \varepsilon_s$$

- iteration does not always reduce error (large h may cause oscillations)

Example

consider the ODE

$$y' = 4e^{0.8x} - 0.5y, \quad y(0) = 2$$

with solution $y(x) = \frac{4}{1.3}(e^{0.8x} - e^{-0.5x}) + 2e^{-0.5x}$

use Heun method to solve the ODE for $x \in [0, 4]$, step size $h = 1$

Step 1

- initial slope: $y'_0 = f(0, 2) = 4e^0 - 0.5(2) = 3$
- predictor: $y_1^0 = 2 + 3(1) = 5$ (Euler method result)
- corrector slope:

$$y'_1 = f(1, 5) = 4e^{0.8(1)} - 0.5(5) = 6.402164$$

- corrected value:

$$y_1 = 2 + \frac{3 + 6.402164}{2}(1) = 6.701082$$

Example

- corrector iteration 2: estimate can be used to refine or correct the prediction of y_1

$$y_1 = 2 + \frac{3 + f(1, 6.701082)}{2}(1) = 6.275811$$

- corrector iteration 3:

$$y_1 = 2 + \frac{3 + f(1, 6.275811)}{2}(1) = 6.382129$$

- after 15 corrector iterations:

$$y_1 \approx 6.360865 \quad (\varepsilon_t = 2.68\%)$$

x	y_{true}	y_{heun} (1 iter)	$\lvert\varepsilon_t\rvert(\%)$	y_{heun} (15 iters)	$\lvert\varepsilon_t\rvert(\%)$
0	2.0000000	2.0000000	0.00	2.0000000	0.00
1	6.1946314	6.7010819	8.18	6.3608655	2.68
2	14.8439219	16.3197819	9.94	15.3022367	3.09
3	33.6771718	37.1992489	10.46	34.7432761	3.17
4	75.3389626	83.3377674	10.62	77.7350962	3.18

iterations converge, but not monotonically (errors may increase before stabilizing)

Special case: derivative depends only on x

for cases where the ODE depends only on x, $y' = f(x)$ (*e.g.*, polynomials)

- the predictor step is not required
- the corrector is applied only once for each iteration
- the method simplifies to:

$$y_{i+1} = y_i + \frac{f(x_i) + f(x_{i+1})}{2}\, h$$

- this is similar to trapezoidal rule
- Heun method has local and global errors of order $O(h^3)$ and $O(h^2)$

Midpoint (improved polygon) method

- predictor to the midpoint:

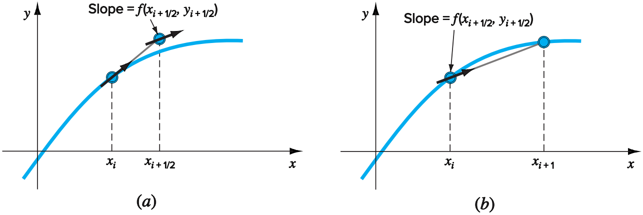
$$y_{i+\frac{1}{2}} = y_i + f(x_i, y_i)\,\frac{h}{2}$$

- slope at the midpoint ($x_{i+\frac{1}{2}} = x_i + \frac{h}{2}$):

$$y'_{i+\frac{1}{2}} = f(x_{i+\frac{1}{2}},\, y_{i+\frac{1}{2}})$$

- corrector (single evaluation; not iterated):

$$y_{i+1} = y_i + f(x_{i+\frac{1}{2}},\, y_{i+\frac{1}{2}})\,h$$



local and global errors of the midpoint method are $O(h^3)$ and $O(h^2)$

Outline

- ordinary differential equations
- Euler method
- Heun and midpoint methods
- **Runge-Kutta methods**
- systems of ODEs
- boundary value problems

Runge-Kutta methods

one-step update in generalized form :

$$y_{i+1} = y_i + \phi(x_i, y_i, h)\, h$$

- increment function ϕ as a weighted blend of stage slopes:

$$\phi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n$$

- stages (recurrence of function evaluations):

$$\begin{aligned}
k_1 &= f(x_i,\ y_i) \\
k_2 &= f(x_i + p_1 h, y_i + q_{11} k_1 h) \\
k_3 &= f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h) \\
&\ \vdots \\
k_n &= f(x_i + p_{n-1} h, y_i + \textstyle\sum_{j=1}^{n-1} q_{n-1,j} k_j h)
\end{aligned}$$

- $a_j, p_j, q_{r,s}$ are chosen so RK update matches Taylor expansion to desired order
- achieves higher accuracy without computing higher derivatives

Second-order Runge-Kutta (RK2)

RK2 update:

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)\, h, \qquad \begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \end{cases}$$

- a_1, a_2, p_1, q_{11} are evaluated by setting above equal to a 2nd-order Taylor series
- this leads to the constraints:

$$a_1 + a_2 = 1, \qquad a_2 p_1 = \frac{1}{2}, \qquad a_2 q_{11} = \frac{1}{2}$$

- one free parameter remains $\Longrightarrow$ a family of RK2 methods; choosing any a_2 gives

$$a_1 = 1 - a_2, \qquad p_1 = q_{11} = \frac{1}{2a_2}$$

- properties: exact for quadratics; local truncation error $O(h^3)$, global $O(h^2)$

RK2 family: special cases

Heun (trapezoidal) form: $a_2 = \frac{1}{2}$

$$a_1 = \tfrac{1}{2}, p_1 = q_{11} = 1 \quad \begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + h,\ y_i + hk_1) \\ y_{i+1} = y_i + \frac{h}{2}\,(k_1 + k_2) \end{cases}$$

Midpoint (improved polygon): $a_2 = 1$

$$a_1 = 0, p_1 = q_{11} = \tfrac{1}{2}, \quad \begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{h}{2},\ y_i + \frac{h}{2}k_1) \\ y_{i+1} = y_i + h\,k_2 \end{cases}$$

Ralston method: $a_2 = \frac{2}{3}$

$$a_1 = \tfrac{1}{3}, p_1 = q_{11} = \tfrac{3}{4}, \quad \begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h) \\ y_{i+1} = y_i + h(\frac{1}{3}k_1 + \frac{2}{3}k_2) \end{cases}$$

Example

use the midpoint and Ralston methods to numerically integrate

$$\frac{dy}{dx} = f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$$

from $x = 0$ to $x = 4$ with step size $h = 0.5$ and initial condition $y(0) = 1$

- since f depends only on x,

$$k_1 = f(0) = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

- *midpoint method:*

$$k_2 = f(0 + \tfrac{h}{2}) = f(0.25) = -2(0.25)^3 + 12(0.25)^2 - 20(0.25) + 8.5 = 4.21875$$

$$y(0.5) \approx y_1 = 1 + (0.5)\,k_2 = 1 + 0.5(4.21875) = 3.109375, \quad \varepsilon_t = 3.4\%$$

- *Ralston method:*

$$k_2 = f(0 + \tfrac{3}{4}h) = f(0.375) = 2.58203125$$

$$\phi = \tfrac{1}{3}k_1 + \tfrac{2}{3}k_2 = \tfrac{1}{3}(8.5) + \tfrac{2}{3}(2.58203125) = 4.5546875$$

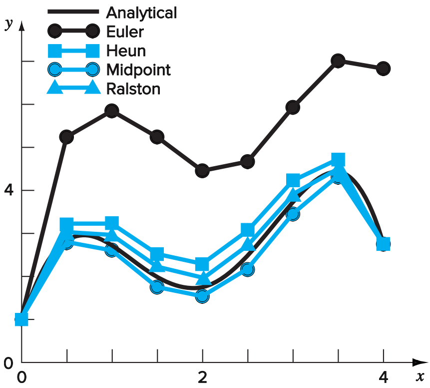
$$y(0.5) \approx y_1 = 1 + (0.5)\,\phi = 1 + 0.5(4.5546875) = 3.27734375, \quad \varepsilon_t = -1.82\%$$

Example

		Heun		Midpoint		Ralston	
x	y_{true}	y	$\lvert\varepsilon_t\rvert$ (%)	y	$\lvert\varepsilon_t\rvert$ (%)	y	$\lvert\varepsilon_t\rvert$ (%)
0.0	1.00000	1.00000	0.0	1.00000	0.0	1.00000	0.0
0.5	3.21875	3.43750	6.8	3.10938	3.4	3.27734	1.8
1.0	3.00000	3.37500	12.5	2.81250	6.3	3.10156	3.4
1.5	2.21875	2.68750	21.1	1.98438	10.6	2.34766	5.8
2.0	2.00000	2.50000	25.0	1.75000	12.5	2.14063	7.0
2.5	2.71875	3.18750	17.2	2.48438	8.6	2.85547	5.0
3.0	4.00000	4.37500	9.4	3.81250	4.7	4.11719	2.9
3.5	4.71875	4.93750	4.6	4.60938	2.3	4.80078	1.7
4.0	3.00000	3.00000	0.0	3.00000	0.0	3.03125	1.0

both midpoint and Ralston produce first-step errors far smaller than Euler method

Example



Third-order Runge-Kutta method (RK3)

- for $n = 3$, a derivation similar to the 2nd case results in 6 equations, 8 unknowns
- by specifying two unknowns a priori, one common version is obtained:

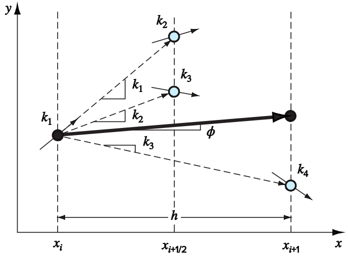
$$\begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h) \\ k_3 = f(x_i + h, y_i - k_1 h + 2k_2 h) \\ y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_2 + k_3)h \end{cases}$$

- reduces to Simpson 1/3 rule if f is a function of x only
- local truncation error: $O(h^4)$; global truncation error: $O(h^3)$
- exact for cubic differential equations and quartic solutions

Fourth-order Runge-Kutta method (RK4)

the *classical fourth-order RK method* is the most widely used form:

$$\begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h) \\ k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h) \\ k_4 = f(x_i + h, y_i + k_3 h) \\ y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \end{cases}$$



- multiple slope estimates (k_1, k_2, k_3, k_4) are computed
- the final slope is a weighted average:

$$\phi = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

- similar in spirit to Heun method but more accurate
- local and global errors of the midpoint method are $O(h^5)$ and $O(h^4)$

Example

use the classical RK4 method to integrate

$$f(x,y) = -2x^3 + 12x^2 - 20x + 8.5$$

with $h = 0.5$, $y(0) = 1$

compute the slopes:

$$\begin{aligned} k_1 &= f(0) = 8.5 \\ k_2 &= f(0.25) = 4.21875 \\ k_3 &= f(0.25) = 4.21875 \\ k_4 &= f(0.5) = 1.25 \end{aligned}$$

substitute into the RK4 formula:

$$y(0.5) \approx y_1 = 1 + \tfrac{1}{6}[8.5 + 2(4.21875) + 2(4.21875) + 1.25](0.5) = 3.21875$$

this matches the exact solution since it is quartic

Example

integrate

$$f(x, y) = 4e^{0.8x} - 0.5y$$

with $h = 0.5$, $y(0) = 2$ from $x = 0$ to $x = 0.5$

the slope at the beginning of the interval is computed as

$$k_1 = f(0, 2) = 4e^{0.8(0)} - 0.5(2) = 3$$

slopes at midpoints and end of interval,

$$y_0 + \tfrac{1}{2}k_1 h = 2 + 3(0.25) = 2.75, \qquad k_2 = f(0.25, 2.75) = 3.510611$$

$$y_0 + \tfrac{1}{2}k_2 h = 2 + 3.510611(0.25) = 2.877653, \qquad k_3 = f(0.25, 2.877653) = 3.446785$$

$$y_0 + k_3 h = 2 + 3.446785(0.5) = 3.723392, \qquad k_4 = f(0.5, 3.723392) = 4.105603$$

the average slope is then used to make the final prediction:

$$\phi = \tfrac{1}{6}\left[3 + 2(3.510611) + 2(3.446785) + 4.105603\right] = 3.503399$$

$$y(0.5) \approx y_1 = 2 + 3.503399(0.5) = 3.751699 \quad \text{(true solution } 3.751521\text{)}$$

Higher-order Runge-Kutta methods

for more accurate results, *Butcher 5th-order RK method* is recommended:

$$y_{i+1} = y_i + \frac{1}{90}\left(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6\right)h$$

with stage slopes

$$\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f(x_i + \tfrac{1}{4}h, y_i + \tfrac{1}{4}k_1 h) \\
k_3 &= f(x_i + \tfrac{1}{4}h, y_i + \tfrac{1}{8}k_1 h + \tfrac{1}{8}k_2 h) \\
k_4 &= f(x_i + \tfrac{1}{2}h, y_i - \tfrac{1}{2}k_2 h + k_3 h) \\
k_5 &= f(x_i + \tfrac{3}{4}h, y_i + \tfrac{3}{16}k_1 h + \tfrac{9}{16}k_4 h) \\
k_6 &= f(x_i + h, y_i - \tfrac{3}{7}k_1 h + \tfrac{2}{7}k_2 h + \tfrac{12}{7}k_3 h - \tfrac{12}{7}k_4 h + \tfrac{8}{7}k_5 h)
\end{aligned}$$

Example

use first- through fifth-order RK methods to solve

$$f(x, y) = 4e^{0.8x} - 0.5y, \qquad y(0) = 2,$$

from $x = 0$ to $x = 4$ with various step sizes

compare the accuracy of the result at $x = 4$ against the exact value $y(4) = 75.33896$

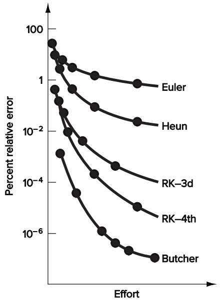
methods compared

- Euler (RK1)
- Heun noniterative (RK2)
- RK3
- classical RK4
- Butcher RK5

Example: computational effort metric

to compare fairly, measure effort by the number of function evaluations:

$$\text{effort} = n_f \, \frac{b-a}{h}$$



- n_f is the number of f evaluations per step
- for orders ≤ 4, n_f equals the order; Butcher RK5 uses six stages
- $(b-a)/h$ counts the RK steps to reach $x = b$
- since f evaluations dominate cost, the product above approximates runtime

Outline

- ordinary differential equations
- Euler method
- Heun and midpoint methods
- Runge-Kutta methods
- **systems of ODEs**
- boundary value problems

System of differential equations

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots, y_n)$$
$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \ldots, y_n)$$
$$\vdots$$
$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, \ldots, y_n)$$

- requires n initial conditions at the starting value of x
- applications in engineering and science problems

One-step methods

- all one-step methods (Euler, RK2, RK4, ...) extend directly to systems of ODEs
- slopes must be computed for each equation at every step

Example: Euler method for a system

solve the system

$$\frac{dy_1}{dx} = -0.5y_1, \qquad \frac{dy_2}{dx} = 4 - 0.3y_2 - 0.1y_1$$

with $y_1(0) = 4$, $y_2(0) = 6$, step size $h = 0.5$, to $x = 2$

we let $y_{j,i}$ be the iteration i for variable j

First step: implement Euler method for each variable

$$y_1(0.5) \approx y_{1,1} = 4 + [-0.5(4)]0.5 = 3$$
$$y_2(0.5) \approx y_{2,1} = 6 + [4 - 0.3(6) - 0.1(4)]0.5 = 6.9$$

Result: proceeding in a like manner for the remaining steps yields

x	y_1	y_2
0.0	4.000000	6.000000
0.5	3.000000	6.900000
1.0	2.250000	7.715000
1.5	1.687500	8.445250
2.0	1.265625	9.094087

Example: RK4 method for a system

apply RK4 to the same system from last example

Solution outline

- first, we must solve for all the slopes at the beginning of the interval:

$$k_{1,1} = f_1(0, 4, 6) = -0.5(4) = -2$$
$$k_{1,2} = f_2(0, 4, 6) = 4 - 0.3(6) - 0.1(4) = 1.8$$

  where $k_{r,j}$ is the rth value of k for the jth dependent variable

- next, we must calculate the first values of y_1 and y_2 at the midpoint:

$$y_{1,0} + \frac{k_{1,1}h}{2} = 4 + (-2)\frac{0.5}{2} = 3.5$$
$$y_{2,0} + \frac{k_{1,2}h}{2} = 6 + (1.8)\frac{0.5}{2} = 6.45$$

- which can be used to compute the first set of midpoint slopes,

$$k_{2,1} = f_1(0.25, 3.5, 6.45) = -1.75$$
$$k_{2,2} = f_2(0.25, 3.5, 6.45) = 1.715$$

Example: RK4 method for a system

- these are used to determine the second set of midpoint predictions,

$$y_{1,0} + \frac{k_{2,1}h}{2} = 4 + (-1.75)\frac{0.5}{2} = 3.5625$$
$$y_{2,0} + \frac{k_{2,2}h}{2} = 6 + (1.715)\frac{0.5}{2} = 6.42875$$

- which can be used to compute the second set of midpoint slopes,

$$k_{3,1} = f_1(0.25, 3.5625, 6.42875) = -1.78125$$
$$k_{3,2} = f_2(0.25, 3.5625, 6.42875) = 1.715125$$

- these are used to determine the predictions at the end of the interval,

$$y_{1,0} + k_{3,1}h = 4 + (-1.78125)(0.5) = 3.109375$$
$$y_{2,0} + k_{3,2}h = 6 + (1.715125)(0.5) = 6.857563$$

- which can be used to compute the endpoint slopes,

$$k_{4,1} = f_1(0.5, 3.109375, 6.857563) = -1.554688$$
$$k_{4,2} = f_2(0.5, 3.109375, 6.857563) = 1.631794$$

Example: RK4 results

- values of k can then be used to compute:

$$y_1(0.5) \approx y_{1,1} = 4 + \tfrac{1}{6}\Big[-2 + 2(-1.75 - 1.78125) - 1.554688\Big](0.5) = 3.115234$$
$$y_2(0.5) \approx y_{2,1} = 6 + \tfrac{1}{6}\Big[1.8 + 2(1.715 + 1.715125) + 1.631794\Big](0.5) = 6.857670$$

- proceeding in a like manner for the remaining steps yields

x	y_1	y_2
0.0	4.000000	6.000000
0.5	3.115234	6.857670
1.0	2.426171	7.632106
1.5	1.889523	8.326886
2.0	1.471577	8.946865

Outline

- ordinary differential equations
- Euler method
- Heun and midpoint methods
- Runge-Kutta methods
- systems of ODEs
- **boundary value problems**

Boundary-value problems

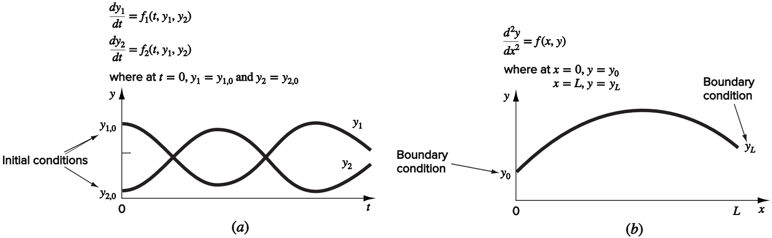
- an ODE solution introduces constants of integration
- *auxiliary conditions* are needed to evaluate them
- for an nth-order ODE, n conditions are required

Initial-value problem (IVP)

conditions specified at same value of independent variable (*e.g.*, $x = 0$ or $t = 0$)

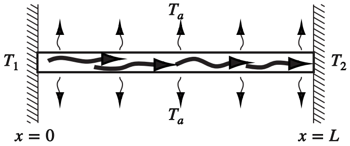
Boundary-value problem (BVP)

conditions specified at *different* points of independent variable



Example: heat balance in a rod

noninsulated uniform rod between two bodies with different constant temperatures



via conservation of heat, the rod temperature satisfies

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

- h' = heat transfer coefficient (m^{-2})
- T_a = ambient temperature ($^\circ$C)
- boundary conditions: $T(0) = T_1, \quad T(L) = T_2$
- for $L = 10$, $T_a = 20$, $T_1 = 40$, $T_2 = 200$, $h' = 0.01$, the solution is

$$T(x) = 73.4523e^{0.1x} - 53.4523e^{-0.1x} + 20$$

Example: shooting method

convert second-order ODE into two first-order ODEs:

$$\frac{dT}{dx} = z, \qquad \frac{dz}{dx} = h'(T - T_a)$$

- guess initial slope $z(0)$ and solve (RK4 with stepsize 2) $\rightarrow$ obtain $T(L)$
- compare with boundary condition $T(L) = 200$ and adjust $z(0)$ until correct

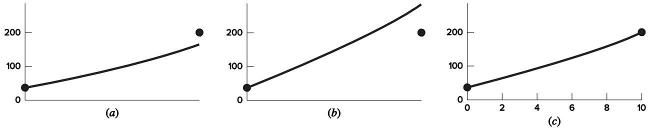
trial 1: $z(0) = 10 \Rightarrow T(10) = 168.3797$ (too low; left plot)

trial 2: $z(0) = 20 \Rightarrow T(10) = 285.8980$ (too high; middle plot)

because ODE is linear, $z(0)$ and $T(10)$ are linearly related; interpolate:

$$z(0) = 10 + \frac{20-10}{285.8980-168.3797}(200 - 168.3797) = 12.6907$$

using this initial value leads to correct solution shown on right



Nonlinear two-point problems

for nonlinear boundary-value problems:

- linear interpolation between two shots is not sufficient
- quadratic interpolation (3 shots) may improve estimate but rarely exact
- alternative: recast as a *roots problem*

Root formulation

- think of solution as function of z_0

$$T_{10} = f(z_0)$$

- goal: $T_{10} = 200$
- adjust z_0 such that

$$g(z_0) = f(z_0) - 200 = 0$$

thus, finding correct initial slope z_0 reduces to a root-finding problem

Example: nonlinear shooting method

$$\frac{d^2T}{dx^2} + h''(T_a - T)^4 = 0, \quad h'' = 5 \times 10^{-8},\ T_a = 20$$

boundary conditions: $T(0) = 40, T(10) = 200$

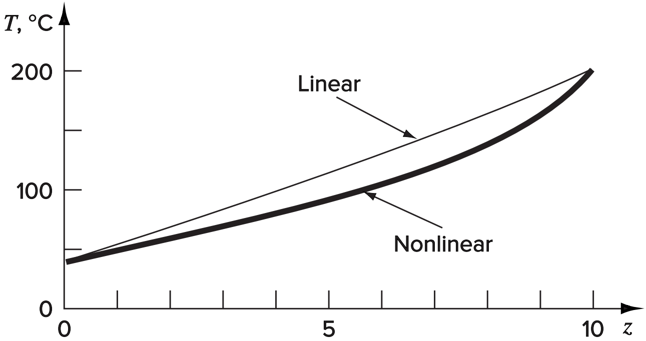
Transformation

$$\frac{dT}{dx} = z, \qquad \frac{dz}{dx} = h''(T - T_a)^4$$

Solution strategy

1. guess initial slope $z(0)$
2. solve system using RK4 with fixed step size
3. obtain $T(10) = f(z_0)$
4. adjust $z(0)$ (*e.g.*, solver, bisection) until $g(z_0) = f(z_0) - 200 = 0$

Remarks on nonlinear shooting



- requires iterative root-finding for initial slopes
- more complex for higher-order problems (need multiple guesses)
- nonlinearities (*e.g.*, radiation, $(T - T_a)^4$) increase curvature
- for large systems, finite-difference or finite-element approaches are more practical

Finite-difference methods

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

- replace derivatives with finite differences (discretize equation)
- for second derivative, the centered difference approximation is

$$\frac{d^2T}{dx^2} \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}$$

- substitute into differential equation:

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + h'(T_a - T_i) = 0$$

  or

$$-T_{i-1} + (2 + h'\Delta x^2)T_i - T_{i+1} = h'\Delta x^2 T_a$$

- applies at $n-1$ interior nodes
- first and last nodes, T_0 and T_n, are fixed by boundary conditions
- results in tridiagonal linear system, which can be solved efficiently

Example: finite-difference solution

solve last example for rod with

$$L = 10, \quad h' = 0.01, \quad T_a = 20, \quad T(0) = 40, \quad T(10) = 200$$

with 4 interior nodes ($\Delta x = 2$ m):

$$\begin{bmatrix} 2.04 & -1 & 0 & 0 \\ -1 & 2.04 & -1 & 0 \\ 0 & -1 & 2.04 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{bmatrix}$$

solution:

$$T = (65.9698, 93.7785, 124.5382, 159.4795)$$

Comparison of methods

x	true	shooting	finite-difference
0	40	40	40
2	65.9518	65.9520	65.9698
4	93.7478	93.7481	93.7785
6	124.5036	124.5039	124.5382
8	159.4534	159.4538	159.4795
10	200	200	200

- both numerical methods accurate
- smaller Δx improves results
- finite-difference preferred for extension to complex systems

Neumann (derivative) boundary condition

$$0 = \frac{d^2T}{dx^2} + h'(T_a - T)$$

- consider derivative boundary conditions:

$$\frac{dT}{dx}(0) = T'_a, \quad T(L) = T_b$$

  - one end has *derivative boundary condition*
  - other end has a *fixed boundary condition*

- we divide rod into a series of nodes and apply approximation to each interior node

$$-T_{i-1} + (2 + h'\Delta x^2)T_i - T_{i+1} = h'\Delta x^2 T_a$$

- because T_0 is not specified, the equation introduces an *imaginary node* T_{-1}

$$-T_{-1} + (2 + h'\Delta x^2)T_0 - T_1 = h'\Delta x^2 T_a$$

Finite-difference formulation with extra node

- we approximate the derivative at the left end ($x = 0$) using a centered difference:

$$\frac{dT}{dx}(0) = \frac{T_1 - T_{-1}}{2\Delta x}$$

- solving for T_{-1}:

$$T_{-1} = T_1 - 2\Delta x \cdot \frac{dT}{dx}(0)$$

- allows us to replace the imaginary node T_{-1}
- the derivative condition is embedded into the finite-difference scheme
- substituting the expression for T_{-1}:

$$(2 + h'\Delta x^2)T_0 - 2T_1 = h'\Delta x^2 T_a - 2\Delta x \cdot \frac{dT}{dx}(0)$$

Physical interpretation

insulated boundary,

$$\frac{dT}{dx} = 0$$

- from Fourier law: heat flux $q = -k\,\frac{dT}{dx}$
- insulation $\Rightarrow q = 0 \Rightarrow \frac{dT}{dx} = 0$
- equation simplifies because the derivative term vanishes

References and further readings

- S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers* (8th edition). McGraw Hill, 2021. (Ch.25)

- S. C. Chapra. *Applied Numerical Methods with MATLAB for Engineers and Scientists* (5th edition). McGraw Hill, 2023. (Ch.22)