Computational Applied Mathematics

Bachelor's Degree in Aeronautical and Space Sciences

Emanuel A. R. Camacho

emanuel.camacho@iseclisboa.pt
earc@earc96.com

Instituto Superior de Educação e Ciências (ISEC Lisboa)

Bibliography

Evaluation

Computational Applied Mathematics (100% [20/20])

Frequencies (70% [14/20]) + Project (30% [6/20])

- Frequency 1 (35% [7/20]) (15/05/2025)
- Frequency 2 (35% [7/20]) (05/06/2025)
- Report (30% [6/20] (13/06/2025)

or

Exam (100% [20/20])

- Exam (100% [20/20])

There is no minimum score for any component of the evaluation. [10/20] is required to pass.

Table of Contents

Outline

Table of Contents

Taylor's Theorem for $f(x)$

If the function f possesses continuous derivatives of orders $0, 1, 2, \ldots, (n+1)$ in a closed interval $I = [a, b]$, then for any c and x in I,

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1}, \tag{1}$$

where the error term E_{n+1} can be given in the form

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}. \tag{2}$$

Here ξ is a point that lies between c and x and depends on both.

Numerical Differentiation

First Derivative Approximation

$$f'(x) = \frac{1}{h}\left[f(x_{i+1}) - f(x_i)\right] - \frac{h}{2}f''(\xi) \tag{3}$$

$$f'(x) = \frac{1}{h}\left[f(x_i) - f(x_{i-1})\right] - \frac{h}{2}f''(\xi) \tag{4}$$

$$f'(x) = \frac{1}{2h}\left[f(x_{i+1}) - f(x_{i-1})\right] - \frac{h^2}{6}f'''(\xi) \tag{5}$$

$$f'(x) = \frac{1}{2h}\left[f(x_{i-2}) - 4f(x_{i-1}) + 3f(x_i)\right] - \frac{h^2}{3}f'''(\xi) \tag{6}$$

$$f'(x) = \frac{1}{2h}\left[-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})\right] - \frac{h^2}{3}f'''(\xi) \tag{7}$$

Numerical Differentiation

Second Derivative Approximation

$$f'(x) = \frac{1}{h^2}\left[f(x_{i-1}) - 2f(x_i) - f(x_{i+1})\right] - \frac{h^2}{12}f^{(4)}(\xi) \quad (8)$$

$$f'(x) = \frac{1}{h^2}\left[f(x_{i-2}) - 2f(x_{i-1}) - f(x_i)\right] - hf'''(\xi) \quad (9)$$

$$f'(x) = \frac{1}{h^2}\left[f(x_i) - 2f(x_{i+1}) - f(x_{i+2})\right] - hf'''(\xi) \quad (10)$$

Table of Contents

The Bolzano Theorem

If f is a continuous function on the closed interval $[a, b]$ and $f(a) \cdot f(b) < 0$, then:

$$\exists c \in [a, b] : f(c) = 0. \tag{11}$$

The Bolzano Theorem

If f is a continuous function on the closed interval $[a, b]$ and $f(a) \cdot f(b) < 0$, then:

$$\exists c \in [a, b] : f(c) = 0. \tag{11}$$

The Rolle Theorem

If f is a continuous function on the closed interval $[a, b]$, differentiable in $]a, b[$ and $f(a) = f(b)$, then:

$$\exists d \in]a, b[: f'(d) = 0. \tag{12}$$

Bisection Method

1. At each step in this algorithm, we have an interval $[a, b]$ and the values $u = f(a)$ and $v = f(b)$. The numbers u and v satisfy $uv < 0$.
2. Next, we construct the midpoint of the interval, $c = \frac{1}{2}(a+b)$, and compute $w = f(c)$.
3. Compute wu and if:
   - $wu < 0$, we store the value of c in b and w in v.
   - $wu > 0$, we store the value of c in a and w in u.
4. This step can now be repeated until the interval is satisfactorily small, say

$$|b - a| \leq \varepsilon \tag{13}$$

Bisection Method Theorem

If the bisection algorithm is applied to a continuous function f on an interval $[a, b]$, where $f(a)f(b) < 0$, then, after n steps, an approximate root will have been computed with error at most $(b-a)/2^{n+1}$.

If an error tolerance has been prescribed in advance, it is possible to determine the number of steps required by solving the following inequality for n:

$$\frac{b-a}{2^{n+1}} < \varepsilon \tag{14}$$

$$n > \frac{\log(b-a) - \log(2\varepsilon)}{\log 2} \tag{15}$$

False-Position (*Regula Falsi*) Method

Rather than selecting the midpoint of each interval, as observed in the bisection method, this method uses the point where the secant lines intersect the x-axis.

1. At the k^{th} step, it computes

$$c_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)} \tag{16}$$

2. Compute $f(a_k)f(c_k)$ and if
   - $f(a_k)f(c_k) > 0$, set $a_{k+1} = c_k$ and $b_{k+1} = b_k$
   - $f(a_k)f(c_k) < 0$, set $a_{k+1} = a_k$ and $b_{k+1} = c_k$.
3. The process is repeated until the root is approximated sufficiently well.

Newton Method (or Newton-Raphson Iteration)

Suppose again that x_0 is an initial approximation to a root of f. We ask: What correction h should be added to x_0 to obtain the root precisely? Obviously, we want

$$f(x_0 + h) = f(x_1) = 0 \tag{17}$$

$$f(x_1) = f(x_0) + (x_1 - x_0)f'(x_0) + \ldots = 0 \tag{18}$$

$$f(x_0) + hf'(x_0) + \ldots = 0 \tag{19}$$

Ignoring all but the first two terms in the series

$$f(x_0) + hf'(x_0) = 0 \Rightarrow h = -\frac{f(x_0)}{f'(x_0)} \tag{20}$$

Newton Method (or Newton-Raphson Iteration)

$$h = -\frac{f(x_0)}{f'(x_0)} \tag{21}$$

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{22}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{23}$$

Secant Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{24}$$

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{25}$$

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{26}$$

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{27}$$

Müller's Method

Fixed-Point Method?

Golden-Section Search

Quadratic Interpolation Method

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + \frac{1}{2} f''(x^*)(x - x^*)^2 + \cdots \tag{28}$$

$$f(x) = f(x^*) + \frac{1}{2} f''(x^*)(x - x^*)^2 + \cdots \tag{29}$$

$$x_{k+1} = \frac{f(x_{k-2})(x_{k-1}^2 - x_k^2) + f(x_{k-1})(x_k^2 - x_{k-2}^2) + f(x_k)(x_{k-2}^2 - x_{k-1}^2)}{2f(x_{k-2})(x_{k-1} - x_k) + 2f(x_{k-1})(x_k - x_{k-2}) + 2f(x_k)(x_{k-2} - x_{k-1})} \tag{30}$$

Newton's Method

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \tag{31}$$

Table of Contents

Gauss Elimination

Gauss Elimination

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}$$

Gauss Elimination

$$2 \leq i \leq n$$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - \left(\dfrac{a_{i1}}{a_{11}}\right) a_{1j} & (1 \leq j \leq n) \\ b_i \leftarrow b_i - \left(\dfrac{a_{i1}}{a_{11}}\right) b_1 \end{cases}$$

Gauss Elimination

$$3 \leq i \leq n$$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - \left(\dfrac{a_{i2}}{a_{22}}\right) a_{2j} & (2 \leq j \leq n) \\ b_i \leftarrow b_i - \left(\dfrac{a_{i2}}{a_{22}}\right) b_2 \end{cases}$$

Gauss Elimination

$$k+1 \leq i \leq n$$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - \left(\dfrac{a_{ik}}{a_{kk}}\right) a_{kj} & (k \leq j \leq n) \\ b_i \leftarrow b_i - \left(\dfrac{a_{ik}}{a_{kk}}\right) b_k & \end{cases}$$

Gauss Elimination

The back substitution starts by solving the nth equation for x_n:

$$x_n = \frac{b_n}{a_{nn}}$$

We continue working upward, recovering each x_i by the formula

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right) \qquad (i = n-1,\ n-2,\ \ldots,\ 1)$$

Gauss-Jordan Elimination

Matrix Inverse?

LU Decomposition

Cholesky Decomposition

Vector Norms

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

$$\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

Matrix Norms

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| \qquad \|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2} \qquad \|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|$$

Jacobi Method

$$x_i^{(k)} = \left[-\sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{a_{ij}}{a_{ii}} \right) x_j^{(k-1)} + \left(\frac{b_i}{a_{ii}} \right) \right] \qquad (1 \leq i \leq n) \tag{32}$$

Gauss-Seidel Method

$$x_i^{(k)} = \left[-\sum_{\substack{j=1 \\ j<i}}^{n} \left(\frac{a_{ij}}{a_{ii}} \right) x_j^{(k)} - \sum_{\substack{j=1 \\ j>i}}^{n} \left(\frac{a_{ij}}{a_{ii}} \right) x_j^{(k-1)} + \left(\frac{b_i}{a_{ii}} \right) \right] \tag{33}$$

$$x_i^{(k)} = \omega \left\{ \left[-\sum_{\substack{j=1 \\ j<i}}^{n} \left(\frac{a_{ij}}{a_{ii}} \right) x_j^{(k)} - \sum_{\substack{j=1 \\ j>i}}^{n} \left(\frac{a_{ij}}{a_{ii}} \right) x_j^{(k-1)} + \left(\frac{b_i}{a_{ii}} \right) \right] \right\} + (1-\omega) x_i^{(k-1)} \tag{34}$$

Newton's Method

$$\begin{cases} f_1(x_1, x_2, \ldots, x_N) = 0 \\ f_2(x_1, x_2, \ldots, x_N) = 0 \\ \quad\vdots \\ f_N(x_1, x_2, \ldots, x_N) = 0 \end{cases} \tag{35}$$

Using vector notation, we can write this system in a more elegant form:

$$\mathbf{F}(\mathbf{X}) = \mathbf{0}. \tag{36}$$

The extension of Newton's method for nonlinear systems is

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \left[\mathbf{F}'\left(\mathbf{X}^{(k)}\right)\right]^{-1} \mathbf{F}\left(\mathbf{X}^{(k)}\right), \tag{37}$$

where $\mathbf{F}'\left(\mathbf{X}^{(k)}\right)$ is the **Jacobian matrix**, which will be defined presently.

Newton's Method

$$\mathbf{F}'(\mathbf{X}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Table of Contents

Linear Interpolation

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

$$p(x) = \left(\frac{x - x_1}{x_0 - x_1}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) y_1 \quad (38)$$

$$= y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0) \quad (39)$$

Linear Interpolation

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

$$p(x) = \left(\frac{x - x_1}{x_0 - x_1}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) y_1 \tag{38}$$

$$= y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0) \tag{39}$$

Lagrange Polynomial

$$p_n(x) = \sum_{i=0}^{n} \ell_i(x) f(x_i) \tag{40}$$

where

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \qquad (0 \leq i \leq n) \tag{41}$$

$$\ell_i(x) = \left(\frac{x - x_0}{x_i - x_0} \right) \left(\frac{x - x_1}{x_i - x_1} \right) \cdots \left(\frac{x - x_{i-1}}{x_i - x_{i-1}} \right) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}} \right) \cdots \left(\frac{x - x_n}{x_i - x_n} \right) \tag{42}$$

Newton Polynomial

$$p_n(x) = \sum_{i=0}^{n} a_i \pi_i(x) \tag{43}$$

where

$$\pi_i(x) = \prod_{j=0}^{i-1} (x - x_j) \tag{44}$$

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}) \tag{45}$$

$$a_k = f\left[x_0, x_1, \dots, x_n\right] \tag{46}$$

where $f\left[x_0, x_1, \dots, x_n\right]$ is called the divided difference of order k.

$$a_0 = f[x_0] = f(x_0) \tag{47}$$

$$a_1 = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{48}$$

$$a_2 = f[x_0, x_1, x_2] = \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} \tag{49}$$

$$f\left[x_0, x_1, \ldots, x_n\right] = \sum_{j=0}^{n} \frac{f(x_j)}{\displaystyle\prod_{\substack{k=0 \\ k\neq j}}^{n} (x_j - x_k)} \tag{50}$$

Natural Cubic Spline

$$S(x) = \begin{cases} S_0(x) & (t_0 \leq x \leq t_1) \\ S_1(x) & (t_1 \leq x \leq t_2) \\ \vdots & \vdots \\ S_{n-1}(x) & (t_{n-1} \leq x \leq t_n) \end{cases} \tag{51}$$

$$S(t_i) = y_i \qquad (0 \leq i \leq n) \tag{52}$$

$$\lim_{x \to t_i^-} S^{(k)}(t_i) = \lim_{x \to t_i^+} S^{(k)}(t_i) \qquad (k = 0, 1, 2) \tag{53}$$

$$S''(t_0) = S''(t_n) = 0 \tag{54}$$

Natural Cubic Spline

$$\begin{bmatrix} 1 & 0 & & & & \\ h_0 & u_1 & h_1 & & & \\ & h_1 & u_2 & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & u_{n-1} & h_{n-1} \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ 0 \end{bmatrix} \tag{55}$$

$$h_i = t_{i+1} - t_i \tag{56}$$

$$u_i = 2(h_{i-1} + h_i) \tag{57}$$

$$v_i = 6(b_i - b_{i-1}) \tag{58}$$

$$b_i = \frac{1}{h_i}(y_{i+1} - y_i) \tag{59}$$

Natural Cubic Spline

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \\ + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6} z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6} z_i\right)(t_{i+1} - x) \tag{60}$$

$$h_i = t_{i+1} - t_i \tag{61}$$

Method of Least Squares

$$f(x) \approx p(x) \tag{62}$$

$$p(x) = \sum_{i=0}^{n} a_i \varphi_i(x) \tag{63}$$

where

$$\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\} \tag{64}$$

is a set of basis functions.

Method of Least Squares

$$\begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_m) & \varphi_1(x_m) & \cdots & \varphi_n(x_m) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

$$Ax = b \tag{65}$$

$$A^+ = (A^T A)^{-1} A^T \tag{66}$$

$$a = A^+ b \tag{67}$$

Table of Contents

$$\int_a^b f(x)\ \mathrm{d}x \tag{68}$$

Here, $a = x_0$, $b = x_n$, $h = (b-a)/n$, $x_i = x_0 + ih$, for $i = 0, 1, \ldots, n$, where $h = (b-a)/n$, $f_i = f(x_i)$, and $a = x_0 < \xi < x_n = b$ in the error terms.

Trapezoid Rule:

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{1}{2}h[f_0 + f_1] - \frac{1}{12}h^3 f''(\xi) \tag{69}$$

Simpson's 1/3 Rule:

$$\int_{x_0}^{x_2} f(x)\,dx = \frac{1}{3}h[f_0 + 4f_1 + f_2] - \frac{1}{90}h^5 f^{(4)}(\xi) \tag{70}$$

Simpson's 3/8 Rule:

$$\int_{x_0}^{x_3} f(x)\,dx = \frac{3}{8}h[f_0 + 3f_1 + 3f_2 + f_3] - \frac{3}{80}h^5 f^{(4)}(\xi) \tag{71}$$

Boole's Rule:

$$\int_{x_0}^{x_4} f(x)\,dx = \frac{2}{45}h[7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4] - \frac{8}{945}h^7 f^{(6)}(\xi) \tag{72}$$

Six-Point Newton-Cotes Closed Rule:

$$\int_{x_0}^{x_5} f(x)\,dx = \frac{5}{288}h\,[19f_0 + 75f_1 + 50f_2 + 50f_3 + 75f_4 + 19f_5] - \frac{275}{12096}h^7 f^{(6)}(\xi) \tag{73}$$

$$\int_a^b f(x)\ \mathrm{d}x \tag{74}$$

Here, $a = x_0$, $b = x_n$, $h = (b-a)/n$, $x_i = x_0 + ih$, for $i = 0, 1, \ldots, n$, where $h = (b-a)/n$, $f_i = f(x_i)$, and $a = x_0 < \xi < x_n = b$ in the error terms.

Midpoint Rule:

$$\int_{x_0}^{x_2} f(x)\,dx = 2hf_1 + \frac{1}{24}h^3 f''(\xi) \tag{75}$$

Two-Point Newton-Cotes Open Rule:

$$\int_{x_0}^{x_3} f(x)\,dx = \frac{3}{2}h[f_1 + f_2] + \frac{1}{4}h^3 f''(\xi) \tag{76}$$

Three-Point Newton-Cotes Open Rule:

$$\int_{x_0}^{x_4} f(x)\,dx = \frac{4}{3}h[2f_1 - f_2 + 2f_3] + \frac{28}{90}h^5 f^{(4)}(\xi) \tag{77}$$

Four-Point Newton-Cotes Open Rule:

$$\int_{x_0}^{x_5} f(x)\,dx = \frac{5}{24}h[11f_1 + f_2 + f_3 + 11f_4] + \frac{95}{144}h^5 f^{(4)}(\xi) \tag{78}$$

Five-Point Newton-Cotes Open Rule:

$$\int_{x_0}^{x_6} f(x)\,dx = \frac{6}{20}h[11f_1 - 14f_2 + 26f_3 - 14f_4 + 11f_5] - \frac{41}{140}h^7 f^{(6)}(\xi) \tag{79}$$

$$\int_a^b f(x)\ \mathrm{d}x \approx \frac{h}{2}\left[f(a) + f(b)\right] \tag{80}$$

$$\int_a^b f(x)\ \mathrm{d}x \approx \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right] \tag{81}$$

$$\int_a^b f(x)\ \mathrm{d}x \approx \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \tag{82}$$

$$\int_a^b f(x)\ \mathrm{d}x \approx \frac{h}{3}\left[f(a) + f(b)\right] + \frac{4h}{3}\sum_{i=1}^{n/2} f\left[a + (2i-1)h\right] + \frac{2h}{3}\sum_{i=1}^{(n-2)/2} f(a+2ih) \tag{83}$$

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(a) + 2\sum_{k=1}^{m-1} f(x_k) + f(b)\right] - \frac{(b-a)h^2}{12} f^{(2)}(\xi) \tag{84}$$

For even number of subintervals m

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(a) + 4\sum_{k=1,3,5,\ldots}^{m-1} f(x_k) + 2\sum_{k=2,4,6,\ldots}^{m-2} f(x_k) + f(b)\right] - \frac{(b-a)h^4}{180} f^{(4)}(\xi) \tag{85}$$

Table of Contents

Initial-Value Problem

In this chapter, we concentrate on one type of differential equation and one type of auxiliary condition: the initial-value problem for a first-order differential equation. The standard form that has been adopted is

$$\begin{cases} x' = f(t, x) \\ x(a) \text{ is given} \end{cases} \tag{86}$$

It is understood that x is a function of t, so the differential equation is written in more detail looks like:

$$\frac{dx(t)}{dt} = f(t, x(t)) \tag{87}$$

Taylor Series Methods

Its principle is to represent the solution of a differential equation locally by a few terms of its Taylor series.

$$x(t+h) = x(t) + h\,x'(t) + \\ + \frac{1}{2!}h^2\,x''(t) + \frac{1}{3!}h^3\,x'''(t) + \frac{1}{4!}h^4\,x^{(4)}(t) + \cdots + \frac{1}{m!}h^m\,x^{(m)}(t) + \cdots \quad (88)$$

For numerical purposes, the Taylor series truncated after $m+1$ terms enables us to compute $x(t+h)$ rather accurately if h is small and if $x(t)$, $x'(t)$, $x''(t), \ldots, x^{(m)}(t)$ are known. When only terms through $h^m x^{(m)}(t)/m!$ are included in the Taylor series, the method that results is called the **Taylor series method of order** m.

Euler Method

The Taylor series method of order 1 is known as **Euler's method**. To find approximate values of the solutions to the initial-value problem

$$\begin{cases} x' = f(t, x(t)) \\ x(a) = x_a \end{cases} \tag{89}$$

over the interval $[a, b]$, the first two terms in the Taylor series (5) are used:

$$x(t+h) \approx x(t) + hx'(t) \tag{90}$$

Hence, the formula

$$x(t+h) = x(t) + hf\big(t, x(t)\big) \tag{91}$$

can be used to step from $t = a$ to $t = b$ with n steps of size $h = (b-a)/n$.

Runge-Kutta Methods

The methods named after Carl Runge and Wilhelm Kutta are designed to imitate the Taylor series method without requiring analytic differentiation of the original differential equation.
The resulting **second-order Runge-Kutta method** is

$$x(t+h) = x(t) + \frac{1}{2}(K_1 + K_2) \tag{92}$$

where

$$\begin{cases} K_1 = hf(t, x) \\ K_2 = hf\left(t+h,\ x+K_1\right) \end{cases} \tag{93}$$

Runge-Kutta Methods

The classical **fourth-order Runge-Kutta method** uses the following formulas:

$$x(t+h) = x(t) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{94}$$

where

$$\begin{cases} K_1 = hf(t, x) \\ K_2 = hf\left(t + \frac{1}{2}h,\ x + \frac{1}{2}K_1\right) \\ K_3 = hf\left(t + \frac{1}{2}h,\ x + \frac{1}{2}K_2\right) \\ K_4 = hf\left(t + h,\ x + K_3\right) \end{cases} \tag{95}$$

The Adams-Bashforth-Moulton methods are a family of predictor-corrector numerical techniques used for solving ordinary differential equations (ODEs). The Adams-Bashforth method provides an initial guess for the new value. The Adams-Moulton method then corrects this guess to improve accuracy.

Adams-Bashforth-Moulton Methods

- Second-order multistep method

$$\tilde{x}(t+h) = x(t) + \frac{h}{2}\Big(3f(t, x(t)) - f(t-h, x(t-h))\Big) \tag{96}$$

$$x(t+h) = x(t) + \frac{h}{2}\Big(f(t+h, \tilde{x}(t+h)) + f(t, x(t))\Big) \tag{97}$$

Adams-Bashforth-Moulton Methods

- Third-order multistep method

$$\tilde{x}(t+h) = x(t) + \frac{h}{12}\Big(23f(t,x(t)) - 16f(t-h,x(t-2h)) + 5f(t-2h,x(t-2h))\Big) \quad (98)$$

$$x(t+h) = x(t) + \frac{h}{12}\Big(5f(t+h,\tilde{x}(t+h)) + 8f(t,x(t)) - f(t-h,x(t-h))\Big) \quad (99)$$

Adams-Bashforth-Moulton Methods

- Fourth-order multistep method

$$\begin{aligned}\tilde{x}(t+h) = x(t) + \frac{h}{24}\Big(&55f(t,x(t)) - 59f(t-h,x(t-h)) \\ &+ 37f(t-2h,x(t-2h)) - 9f(t-3h,x(t-3h))\Big) \quad (100)\end{aligned}$$

$$\begin{aligned}x(t+h) = x(t) + \frac{h}{24}\Big(&9f(t+h,\tilde{x}(t+h)) + 19f(t,x(t)) \\ &- 5f(t-h,x(t-h)) + f(t-2h,x(t-2h))\Big) \quad (101)\end{aligned}$$

$$\begin{cases} x_1' = f_1(t, x_1, x_2, \dots, x_n) \\ x_2' = f_2(t, x_1, x_2, \dots, x_n) \\ \vdots \\ x_n' = f_n(t, x_1, x_2, \dots, x_n) \\ x_1(a) = s_1,\ x_2(a) = s_2,\ \dots,\ x_n(a) = s_n \qquad \text{all given} \end{cases} \tag{102}$$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{X}' = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \tag{103}$$

m-order Taylor Series method

$$\mathbf{X}(t+h) = \mathbf{X}(t) + h\mathbf{X}(t)' + \frac{h^2}{2}\mathbf{X}(t)'' + \cdots + \frac{h^m}{m!}\mathbf{X}(t)^{(m)} \tag{104}$$

4^{th}-order Runge-Kutta method

$$\mathbf{X}(t+h) = \mathbf{X}(t) + \frac{h}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4) \tag{105}$$

$$\begin{cases} \mathbf{K}_1 = \mathbf{F}(t, x) \\ \mathbf{K}_2 = \mathbf{F}(t + 1/2h, \mathbf{X} + 1/2h\mathbf{K}_1) \\ \mathbf{K}_3 = \mathbf{F}(t + 1/2h, \mathbf{X} + 1/2h\mathbf{K}_2) \\ \mathbf{K}_4 = \mathbf{F}(t + h, \mathbf{X} + h\mathbf{K}_3) \end{cases} \tag{106}$$

4^{th}-order Adams-Bashforth-Moulton method

- Adams-Bashforth method (predictor)

$$\begin{aligned}\tilde{\mathbf{X}}(t+h) = \mathbf{X}(t) + \frac{h}{24}\Big(&55\mathbf{F}(t,\mathbf{X}(t)) - 59\mathbf{F}(t-h,\mathbf{X}(t-h)) \\ &+ 37\mathbf{F}(t-2h,\mathbf{X}(t-2h)) - 9\mathbf{F}(t-3h,\mathbf{X}(t-3h))\Big) \qquad (107)\end{aligned}$$

- Adams-Moulton method (corrector)

$$\begin{aligned}\mathbf{X}(t+h) = \mathbf{X}(t) + \frac{h}{24}\Big(&9\mathbf{F}(t+h,\tilde{\mathbf{X}}(t+h)) + 19\mathbf{F}(t,\mathbf{X}(t)) \\ &- 5\mathbf{F}(t-h,\mathbf{X}(t-h)) + \mathbf{F}(t-2h,\mathbf{X}(t-2h))\Big) \qquad (108)\end{aligned}$$

Computational Applied Mathematics

Bachelor's Degree in Aeronautical and Space Sciences

Emanuel A. R. Camacho

emanuel.camacho@iseclisboa.pt
earc@earc96.com

Instituto Superior de Educação e Ciências (ISEC Lisboa)