现代数学概论【科学计算】

Lecture 4 - Numerical Integration & Differentiation

Xian-Liang Hu

School of Mathematical Sciences, Zhejiang University, CHINA.

http://www.mathweb.zju.edu.cn:8080/xlhu/sc.html

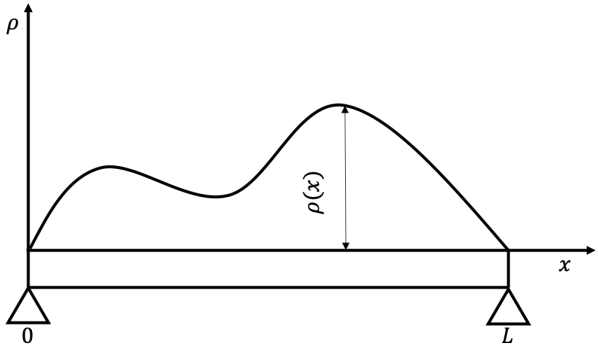
Main Contents of Lecture 4

10. Quadrature

11. ODE-based Initial Value Problem

12. ODE-based Boundary Value Problem

10. Quadrature



The Integration (Model)

Definition (Integration)

For a function $f:\mathbb{R} \to \mathbb{R}$ on $[a, b]$ with $a = x_1 < x_2 < \cdots < x_{n+1} = b$, let

$$I_n = \sum_{i=1}^{n} (x_{i+1} - x_i) f(\xi_i),$$

where $\xi_i \in [x_i, x_{i+1}]$ and define $h_n = \max_{i=1,\ldots,n-1} \{x_{i+1} - x_i\}$. As soon as

$$\lim_{h_n \to 0} I_n = I \neq \infty,$$

then we say the integration of f on $[a, b]$ exists, which is denoted by

$$\mathrm{I}(f) = \int_a^b f(x)dx.$$

The Integration (Model)

Definition (Integration)

For a function $f:\mathbb{R} \to \mathbb{R}$ on $[a, b]$ with $a = x_1 < x_2 < \cdots < x_{n+1} = b$, let

$$I_n = \sum_{i=1}^{n} (x_{i+1} - x_i) f(\xi_i),$$

where $\xi_i \in [x_i, x_{i+1}]$ and define $h_n = \max\limits_{i=1,\ldots,n-1} \{x_{i+1} - x_i\}$. As soon as

$$\lim_{h_n \to 0} I_n = I \neq \infty,$$

then we say the integration of f on $[a, b]$ exists, which is denoted by

$$\mathrm{I}(f) = \int_a^b f(x) dx.$$

Application of Integration (Model)

Integration is widely used in various fields, including:

1. Laplace and Fourier transform
2. finite element method and boundary element method for partial differential equation
3. integral equation, variational method
4. statistics problem in which many basic concepts, like probability distribution and expectation, are defined
5. system energy in classical and quantum physics
6. .....

Regularity of Integration (Model)

- Existence is that if the integration above is solvable, which is illustrated that the bounded and continuous integration is integrable.
- Uniqueness is that the existence is independent with the choice of ξ_i and mesh.
- Stability is one of the most important topics in integration, which can be represented the propagation of errors with the noise of data.

Stability Analysis - Noisy Information

1. Noise in integrand function:

$$|I(\hat{f}) - I(f)| \leq \int_a^b |\hat{f}(x) - f(x)|dx \leq (b-a)\|\hat{f} - f\|_\infty.$$

2. Noise in the domain/interval:

$$\left| \int_a^{\hat{b}} f(x)dx - \int_a^b f(x)dx \right| \leq \int_b^{\hat{b}} f(x)dx \leq (\hat{b} - b)\|f\|_\infty.$$

Stability Analysis - Noisy Information

1. Noise in integrand function:

$$|I(\hat{f}) - I(f)| \leq \int_a^b |\hat{f}(x) - f(x)|dx \leq (b-a)\|\hat{f} - f\|_\infty.$$

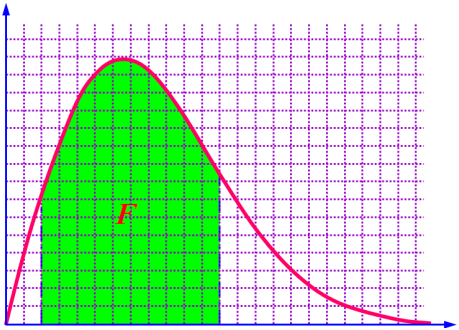
2. Noise in the domain/interval:

$$\left| \int_a^{\hat{b}} f(x)dx - \int_a^b f(x)dx \right| \leq \int_b^{\hat{b}} f(x)dx \leq (\hat{b} - b)\|f\|_\infty.$$

Numerical Integration - Quadrature Rule

1) To approximate $I(f) =$

$$\int_a^b f(x)\mathrm{d}x = \lim_{\lambda \to 0} \sum_{i=0}^{n} f(\xi_i) \triangle x_i$$



2) *Quadrature Rule*(approximation)

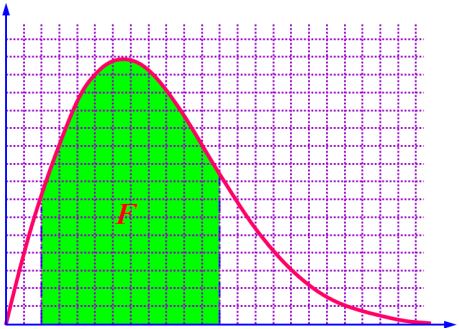
$$\mathcal{I}(f; a, b) = \int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{n} A_i f(x_i) := Q_n(f; a, b) \qquad (1)$$

where x_i is the *nodes*/*abscissas* and A_i is the *weights*/*coefficients* of Q_n.

Numerical Integration - Quadrature Rule

1) To approximate $I(f) =$

$$\int_a^b f(x)\mathrm{d}x = \lim_{\lambda \to 0} \sum_{i=0}^{n} f(\xi_i) \triangle x_i$$



2) *Quadrature Rule*(approximation)

$$\mathcal{I}(f; a, b) = \int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{n} A_i f(x_i) := Q_n(f; a, b) \qquad (1)$$

where x_i is the *nodes/abscissas* and A_i is the *weights/coefficients* of Q_n.

MATLAB: » help quad

```
Q = quad(FUN,A,B,TOL) uses an absolute error tolerance of TOL
instead of the default, which is 1.e-6.  Larger values of TOL
result in fewer function evaluations and faster computation,
but less accurate results.  The quad function in MATLAB 5.3 used
a less reliable algorithm and a default tolerance of 1.e-3.

Example:
   Q = quad(@myfun,0,2);
where the file myfun.m defines the function:
   %------------------%
   function y = myfun(x)
   y = 1./(x.^3-2*x-5);
   %------------------%

or, use a parameter for the constant:
   Q = quad(@(x)myfun2(x,5),0,2);
where the file myfun2.m defines the function:
   %----------------------%
   function y = myfun2(x,c)
   y = 1./(x.^3-2*x-c);
   %----------------------%
```

A Fundamental Approach

The integration of certain basis functions $\phi_i(x)$

$$\sum_{i=1}^{n} \omega_i \phi_j(x_i) = \int_a^b \phi_j(x) dx, \qquad j = 1, 2, \ldots, n,$$

leads to a linear system $A\omega = \mathrm{f}$, more precisely,

$$\begin{bmatrix} \phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_n) \\ \phi_2(x_1) & \phi_2(x_2) & \cdots & \phi_2(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_n(x_1) & \phi_n(x_2) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} \begin{bmatrix} \int_a^b \phi_1(x) dx \\ \int_a^b \phi_2(x) dx \\ \vdots \\ \int_a^b \phi_n(x) dx \end{bmatrix}$$

- A new quadrature rule is derived as soon as $\left\{\phi_i(x_j)\right\}_{i,j=1}^{n}$ is given.

A Fundamental Approach

The integration of certain basis functions $\phi_i(x)$

$$\sum_{i=1}^{n} \omega_i \phi_j(x_i) = \int_a^b \phi_j(x) dx, \qquad j = 1, 2, \ldots, n,$$

leads to a linear system $A\omega = \mathrm{f}$, more precisely,

$$\begin{bmatrix} \phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_n) \\ \phi_2(x_1) & \phi_2(x_2) & \cdots & \phi_2(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_n(x_1) & \phi_n(x_2) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} \begin{bmatrix} \int_a^b \phi_1(x) dx \\ \int_a^b \phi_2(x) dx \\ \vdots \\ \int_a^b \phi_n(x) dx \end{bmatrix}$$

- A new quadrature rule is derived as soon as $\left\{\phi_i(x_j)\right\}_{i,j=1}^{n}$ is given.

Example ($n = 3$)

Compute the three-nodes integral formula with

$$Q_3(f) = \omega_1 f(x_1) + \omega_2 f(x_2) + \omega_3 f(x_3).$$

- determin $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ as the quadrature nodes and the basic function is $\phi_j(x) = x^{j-1}, j = 1, 2, 3$.
- write out the linear system $A\omega = \mathrm{f}$ with

$$A = \begin{bmatrix} 1 & 1 & 1 \\ a & \frac{a+b}{2} & b \\ a^2 & \left(\frac{a+b}{2}\right)^2 & b^2 \end{bmatrix}, \qquad \mathrm{f} = \begin{bmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \end{bmatrix}$$

- Finally, obtain the solution(Quadrature weights)

$$\omega_1 = \frac{b-a}{6}, \omega_2 = \frac{2(b-a)}{3}, \omega_3 = \frac{b-a}{6}.$$

Example ($n = 3$)

Compute the three-nodes integral formula with

$$Q_3(f) = \omega_1 f(x_1) + \omega_2 f(x_2) + \omega_3 f(x_3).$$

- determin $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ as the quadrature nodes and the basic function is $\phi_j(x) = x^{j-1}, j = 1, 2, 3$.
- write out the linear system $A\omega = \mathrm{f}$ with

$$A = \begin{bmatrix} 1 & 1 & 1 \\ a & \frac{a+b}{2} & b \\ a^2 & \left(\frac{a+b}{2}\right)^2 & b^2 \end{bmatrix}, \qquad \mathrm{f} = \begin{bmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \end{bmatrix}$$

- Finally, obtain the solution(Quadrature weights)

$$\omega_1 = \frac{b-a}{6}, \omega_2 = \frac{2(b-a)}{3}, \omega_3 = \frac{b-a}{6}.$$

Example ($n = 3$)

Compute the three-nodes integral formula with

$$Q_3(f) = \omega_1 f(x_1) + \omega_2 f(x_2) + \omega_3 f(x_3).$$

- determin $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ as the quadrature nodes and the basic function is $\phi_j(x) = x^{j-1}, j = 1, 2, 3$.
- write out the linear system $A\omega = \mathrm{f}$ with

$$A = \begin{bmatrix} 1 & 1 & 1 \\ a & \frac{a+b}{2} & b \\ a^2 & \left(\frac{a+b}{2}\right)^2 & b^2 \end{bmatrix}, \qquad \mathrm{f} = \begin{bmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \end{bmatrix}$$

- Finally, obtain the solution(Quadrature weights)

$$\omega_1 = \frac{b-a}{6}, \omega_2 = \frac{2(b-a)}{3}, \omega_3 = \frac{b-a}{6}.$$

Example ($n = 3$)

Compute the three-nodes integral formula with

$$Q_3(f) = \omega_1 f(x_1) + \omega_2 f(x_2) + \omega_3 f(x_3).$$

- determin $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ as the quadrature nodes and the basic function is $\phi_j(x) = x^{j-1}, j = 1, 2, 3$.
- write out the linear system $A\omega = \mathrm{f}$ with

$$A = \begin{bmatrix} 1 & 1 & 1 \\ a & \frac{a+b}{2} & b \\ a^2 & \left(\frac{a+b}{2}\right)^2 & b^2 \end{bmatrix}, \qquad \mathrm{f} = \begin{bmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \end{bmatrix}$$

- Finally, obtain the solution(Quadrature weights)

$$\omega_1 = \frac{b-a}{6}, \omega_2 = \frac{2(b-a)}{3}, \omega_3 = \frac{b-a}{6}.$$

A Classical Approach: Newton-Cotes Quadrature

By using equidistributed nodes on $[a, b]$,say

$$x_i = a + i(b-a)/(n+1), i = 1, 2, \cdots, n.$$

and the integrand function f is approximation by the Lagrange interpolant.

In this sense, the *moment equations*

$$\sum_{i=1}^{n} \omega_i l_j(x_i) = \int_a^b l_j(x)dx, \qquad j = 1, 2, \ldots, n,$$

yields simple and constantly used quadrature rules, which are known as the *Newton-Cotes quadrature rules*.

A Classical Approach: Newton-Cotes Quadrature

By using equidistributed nodes on $[a, b]$,say

$$x_i = a + i(b-a)/(n+1), i = 1, 2, \cdots, n.$$

and the integrand function f is approximation by the Lagrange interpolant. In this sense, the *moment equations*

$$\sum_{i=1}^{n} \omega_i l_j(x_i) = \int_a^b l_j(x)dx, \qquad j = 1, 2, \ldots, n,$$

yields simple and constantly used quadrature rules, which are known as the *Newton-Cotes quadrature rules*.

Example (Trapzoid Rule)

Choosing two nodes $x = a$ and $x = b$, then the trapzoid rule can be written as

$$Q_n(f) = \frac{(b-a)}{2}\Big[f(a) + f(b)\Big].$$

- Discretization error:

$$E_2 = \left|\int_a^b f(x)dx - Q_n(f)\right| = \int_a^b \frac{(b-1)^2}{8} f''(\xi)dx = \left(\frac{b-a}{2}\right)^3 \|f''\|_\infty$$

- Compound trapezoidal formula(with $n+1$ nodes for the convenience):

$$T_n = \frac{h}{2}\sum_{i=0}^{n-1}\Big(f(x_i)+f(x_{i+1})\Big) = \frac{b-a}{n}\left[\frac{f(x_0)}{2} + f(x_1) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

with discretization error: $E_{rr}^T = -\frac{(b-a)\|f''\|_\infty}{12}h^2$

Example (Trapzoid Rule)

Choosing two nodes $x = a$ and $x = b$, then the trapzoid rule can be written as

$$Q_n(f) = \frac{(b-a)}{2}\Big[f(a) + f(b)\Big].$$

- Discretization error:

$$E_2 = \left|\int_a^b f(x)dx - Q_n(f)\right| = \int_a^b \frac{(b-1)^2}{8} f''(\xi)dx = \left(\frac{b-a}{2}\right)^3 \|f''\|_\infty$$

- Compound trapezoidal formula(with $n+1$ nodes for the convenience):

$$T_n = \frac{h}{2}\sum_{i=0}^{n-1}\Big(f(x_i)+f(x_{i+1})\Big) = \frac{b-a}{n}\left[\frac{f(x_0)}{2} + f(x_1) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

with discretization error: $E_{rr}^T = -\frac{(b-a)\|f''\|_\infty}{12}h^2$

Example (Trapzoid Rule)

Choosing two nodes $x = a$ and $x = b$, then the trapzoid rule can be written as

$$Q_n(f) = \frac{(b-a)}{2}\Big[f(a) + f(b)\Big].$$

- Discretization error:

$$E_2 = \left|\int_a^b f(x)dx - Q_n(f)\right| = \int_a^b \frac{(b-1)^2}{8} f''(\xi)dx = \left(\frac{b-a}{2}\right)^3 \|f''\|_\infty$$

- Compound trapezoidal formula(with $n+1$ nodes for the convenience):

$$T_n = \frac{h}{2}\sum_{i=0}^{n-1}\Big(f(x_i)+f(x_{i+1})\Big) = \frac{b-a}{n}\left[\frac{f(x_0)}{2} + f(x_1) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

with discretization error: $E_{rr}^T = -\frac{(b-a)\|f''\|_\infty}{12}h^2$

Example (Simpson Rule)

$$S(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

- Discretization error:

$$E_3 = \left|\int_a^b f(x)dx - S(f)\right| = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad a < \eta < b$$

- Compound Simpson rule(with $n+1$ knots):

$$S_n = \frac{h}{6}\sum_{i=0}^{n-1}\left[f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})\right].$$

with discretization error: $E_{rr}^S = -\frac{(b-a)\|f^{(4)}\|_\infty}{2880} h^4$.

Example (Simpson Rule)

$$S(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

- Discretization error:

$$E_3 = \left|\int_a^b f(x)dx - S(f)\right| = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad a < \eta < b$$

- Compound Simpson rule(with $n+1$ knots):

$$S_n = \frac{h}{6}\sum_{i=0}^{n-1}\left[f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})\right].$$

with discretization error: $E_{rr}^S = -\frac{(b-a)\|f^{(4)}\|_\infty}{2880} h^4$.

Example (Simpson Rule)

$$S(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

- Discretization error:

$$E_3 = \left|\int_a^b f(x)dx - S(f)\right| = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad a < \eta < b$$

- Compound Simpson rule(with $n+1$ knots):

$$S_n = \frac{h}{6}\sum_{i=0}^{n-1}\left[f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})\right].$$

with discretization error: $E_{rr}^S = -\frac{(b-a)\|f^{(4)}\|_\infty}{2880} h^4$.

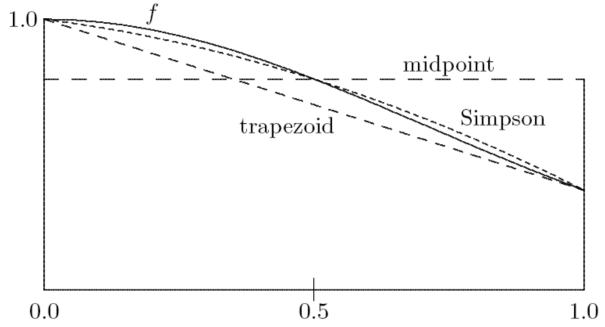
Using different formulas in Newton-Cotes to compute $I(f) = \int_0^1 e^{-x^2} dx$.

$$M(f) = (1-0)e^{-0.25} = 0.778801$$

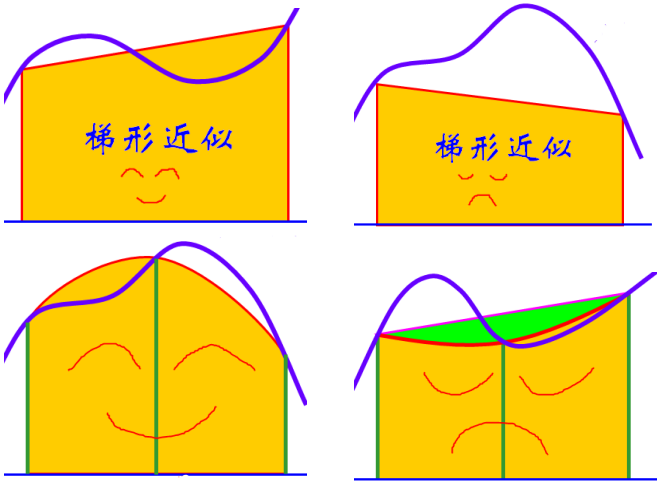
$$T(f) = \frac{1}{2}(e^0 + e^{-1}) = 0.683940$$

$$T(f) = \frac{1}{6}(e^0 + 4e^{-0.25} + e^{-1}) = 0.747180$$

The exact solution is 0.746824, we plot the interpolant as



Good or Bad?



Case N: Coefficients for Cotes

n	$C_i^{(n)}$
1	$\frac{1}{2}$, $\frac{1}{2}$
2	$\frac{1}{6}$, $\frac{4}{6}$, $\frac{1}{6}$
3	$\frac{1}{8}$, $\frac{3}{8}$, $\frac{3}{8}$, $\frac{1}{8}$
4	$\frac{7}{90}$, $\frac{16}{45}$, $\frac{2}{15}$, $\frac{16}{45}$, $\frac{7}{90}$
5	$\frac{19}{288}$, $\frac{25}{96}$, $\frac{25}{144}$, $\frac{25}{144}$, $\frac{25}{96}$, $\frac{19}{288}$
6	$\frac{41}{840}$, $\frac{9}{35}$, $\frac{9}{280}$, $\frac{34}{105}$, $\frac{9}{280}$, $\frac{9}{35}$, $\frac{41}{840}$
7	$\frac{1}{172\,80}$ {751, 357 7, 132 3, 298 9, 132 3, 357 7, 750}
8	$\frac{1}{283\,50}$ {989, 588 8, −928 104 96, −454 0, 104 96 −928, 588 8, 989}
⋮	⋮

Example (Construct two point formula $G_2(f)$ in the interval (-1,1))

$$I(f) = \int_{-1}^{1} f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) := G_2(f),$$

Assume it establishes for $f_j(x) = x^{j-1}, j = 1, 2, 3, 4$.

$$\begin{cases} \omega_1 + \omega_2 = \int_{-1}^{1} 1dx = 2, \\ \omega_1 x_1 + \omega_2 x_2 = \int_{-1}^{1} xdx = 0, \\ \omega_1 x_1^2 + \omega_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3}, \\ \omega_1 x_1^3 + \omega_2 x_2^3 = \int_{-1}^{1} x^3 dx = \frac{2}{3}, \end{cases}$$

The solution is

$$x_1 = -1/\sqrt{3}, x_2 = 1/\sqrt{3}, \omega_1 = 1, \omega_2 = 1,$$

which yeilds

$$G_2(f) = f(-1/\sqrt{3}) + f(1/\sqrt{3}).$$

Example (Construct two point formula $G_2(f)$ in the interval (-1,1))

$$I(f) = \int_{-1}^{1} f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) := G_2(f),$$

Assume it establishes for $f_j(x) = x^{j-1}, j = 1, 2, 3, 4.$

$$\begin{cases} \omega_1 + \omega_2 = \int_{-1}^{1} 1dx = 2, \\ \omega_1 x_1 + \omega_2 x_2 = \int_{-1}^{1} xdx = 0, \\ \omega_1 x_1^2 + \omega_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3}, \\ \omega_1 x_1^3 + \omega_2 x_2^3 = \int_{-1}^{1} x^3 dx = \frac{2}{3}, \end{cases}$$

The solution is

$$x_1 = -1/\sqrt{3}, x_2 = 1/\sqrt{3}, \omega_1 = 1, \omega_2 = 1,$$

which yeilds

$$G_2(f) = f(-1/\sqrt{3}) + f(1/\sqrt{3}).$$

Example (Construct two point formula $G_2(f)$ in the interval (-1,1))

$$I(f) = \int_{-1}^{1} f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) := G_2(f),$$

Assume it establishes for $f_j(x) = x^{j-1}, j = 1, 2, 3, 4$.

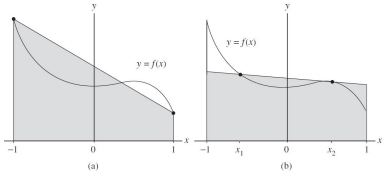
$$\begin{cases} \omega_1 + \omega_2 = \int_{-1}^{1} 1dx = 2, \\ \omega_1 x_1 + \omega_2 x_2 = \int_{-1}^{1} xdx = 0, \\ \omega_1 x_1^2 + \omega_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3}, \\ \omega_1 x_1^3 + \omega_2 x_2^3 = \int_{-1}^{1} x^3 dx = \frac{2}{3}, \end{cases}$$

The solution is

$$x_1 = -1/\sqrt{3}, x_2 = 1/\sqrt{3}, \omega_1 = 1, \omega_2 = 1,$$

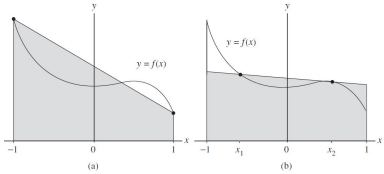
which yeilds

$$G_2(f) = f(-1/\sqrt{3}) + f(1/\sqrt{3}).$$



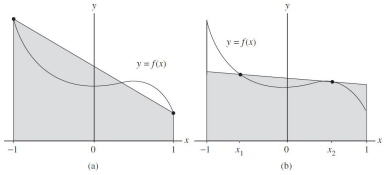
n	x_i	A_i	n	x_i	A_i
0	0	2		±0.6612093865	0.3607615370
1	±0.5773502692	1		±0.2386191861	0.4679139346
2	±0.7745966692	0.555555556	6	±0.9491079123	0.1294849662
	0	0.8888888889		±0.7415311856	0.2797053915
3	±0.8611363116	0.3478548451		±0.4058451514	0.3818300505
	±0.3399810436	0.6521451549		0	0.4179591837
4	±0.9061798459	0.2369268851	7	±0.9602898565	0.1012285363
	±0.5384693101	0.4786286705		±0.7966664774	0.2223810345
	0	0.5688888889		±0.5255354099	0.3137066459
5	±0.9324695142	0.1713244924		±0.1834346425	0.3626837834

- it is important to learn how to compute by the orthogonal polynomials such as Legendre polynomial.
- The Gauss rule points are given in the unit interval, affine transformation is necessary in application.
- The integration can be computed by Gauss-Kronrod recursive method.



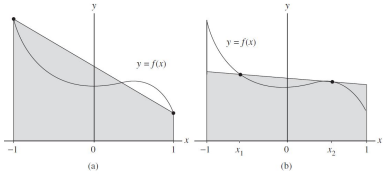
n	x_i	A_i	n	x_i	A_i
0	0	2		±0.6612093865	0.3607615370
1	±0.5773502692	1		±0.2386191861	0.4679139346
2	±0.7745966692	0.555555556	6	±0.9491079123	0.1294849662
	0	0.8888888889		±0.7415311856	0.2797053915
3	±0.8611363116	0.3478548451		±0.4058451514	0.3818300505
	±0.3399810436	0.6521451549		0	0.4179591837
4	±0.9061798459	0.2369268851	7	±0.9602898565	0.1012285363
	±0.5384693101	0.4786286705		±0.7966664774	0.2223810345
	0	0.5688888889		±0.5255354099	0.3137066459
5	±0.9324695142	0.1713244924		±0.1834346425	0.3626837834

- it is important to learn how to compute by the orthogonal polynomials such as Legendre polynomial.
- The Gauss rule points are given in the unit interval, affine transformation is necessary in application.
- The integration can be computed by Gauss-Kronrod recursive method.



n	x_i	A_i	n	x_i	A_i
0	0	2		±0.6612093865	0.3607615370
1	±0.5773502692	1		±0.2386191861	0.4679139346
2	±0.7745966692	0.555555556	6	±0.9491079123	0.1294849662
	0	0.8888888889		±0.7415311856	0.2797053915
3	±0.8611363116	0.3478548451		±0.4058451514	0.3818300505
	±0.3399810436	0.6521451549		0	0.4179591837
4	±0.9061798459	0.2369268851	7	±0.9602898565	0.1012285363
	±0.5384693101	0.4786286705		±0.7966664774	0.2223810345
	0	0.5688888889		±0.5255354099	0.3137066459
5	±0.9324695142	0.1713244924		±0.1834346425	0.3626837834

- it is important to learn how to compute by the orthogonal polynomials such as Legendre polynomial.
- The Gauss rule points are given in the unit interval, affine transformation is necessary in application.
- The integration can be computed by Gauss-Kronrod recursive method.



n	x_i	A_i	n	x_i	A_i
0	0	2		± 0.6612093865	0.3607615370
1	± 0.5773502692	1		± 0.2386191861	0.4679139346
2	± 0.7745966692	0.555555556	6	± 0.9491079123	0.1294849662
	0	0.8888888889		± 0.7415311856	0.2797053915
3	± 0.8611363116	0.3478548451		± 0.4058451514	0.3818300505
	± 0.3399810436	0.6521451549		0	0.4179591837
4	± 0.9061798459	0.2369268851	7	± 0.9602898565	0.1012285363
	± 0.5384693101	0.4786286705		± 0.7966664774	0.2223810345
	0	0.5688888889		± 0.5255354099	0.3137066459
5	± 0.9324695142	0.1713244924		± 0.1834346425	0.3626837834

- it is important to learn how to compute by the orthogonal polynomials such as Legendre polynomial.
- The Gauss rule points are given in the unit interval, affine transformation is necessary in application.
- The integration can be computed by Gauss-Kronrod recursive method.

Clenshaw-Curtis

- As with polynomial interpolation, use of Chebyshev points produces better results
- Improved accuracy results from good approximation properties of interpolation at Chebyshev points, such as:
  - Weights are always positive
  - approximate integral always converges to exact integral as $n \to \infty$
- Quadrature rules using Chebyshev points are known as Clenshaw-Curtis quadrature, which can be implemented very efficiently, such as utilizing FFT and recirsive calculation
- It is not optimal, pitful!

Clenshaw-Curtis

- As with polynomial interpolation, use of Chebyshev points produces better results
- Improved accuracy results from good approximation properties of interpolation at Chebyshev points, such as:
  - Weights are always positive
  - approximate integral always converges to exact integral as $n \to \infty$
- Quadrature rules using Chebyshev points are known as Clenshaw-Curtis quadrature, which can be implemented very efficiently, such as utilizing FFT and recirsive calculation
- It is not optimal, pitful!

Clenshaw-Curtis

- As with polynomial interpolation, use of Chebyshev points produces better results
- Improved accuracy results from good approximation properties of interpolation at Chebyshev points, such as:
  - Weights are always positive
  - approximate integral always converges to exact integral as $n \to \infty$
- Quadrature rules using Chebyshev points are known as Clenshaw-Curtis quadrature, which can be implemented very efficiently, such as utilizing FFT and recirsive calculation
- It is not optimal, pitful!

Clenshaw-Curtis

- As with polynomial interpolation, use of Chebyshev points produces better results
- Improved accuracy results from good approximation properties of interpolation at Chebyshev points, such as:
  - Weights are always positive
  - approximate integral always converges to exact integral as $n \to \infty$
- Quadrature rules using Chebyshev points are known as Clenshaw-Curtis quadrature, which can be implemented very efficiently, such as utilizing FFT and recirsive calculation
- It is not optimal, pitful!

Tricks for Accuracy Improvements

There are many methods to improve the accuracy of integration

1. Compound: be similar with the piecewise interpolation.
2. Adaptivity: be similar with recursion method
3. Richardson extrapolation: Romberg integration.

```
function t = Romberg(func, a, b, eps)
% function t = Romberg(func, a, b, eps)
% Test Case:
%       func = @(x) x.*(sqrt(1+x.*x));
%       Romberg(func,0,3,1e-4) % try 1e-6, 1e-8, 1e-10
s=1000; s0 = 0; k = 1; h = (b-a);
t(1,1) = 0.5*(b-a)*(func(a) + func(b));
while(abs(s - s0) > eps)
   k = k + 1; h = 0.5*h;   w = 0; % prepare current step
   for i = 1:(2^(k-1)-1)
       w = w + func(a+i*h);
   end
   t(k,1) = 0.5*h*(func(a) + 2*w + func(b)); % trapz(n)
   for l = 2:k  % Execute the Romberg's algorithm
       t(k,l) = (4^(l-1)*t(k,l-1) - t(k-1,l-1))/(4^(l-1)-1);
   end
   s = t(k,k);  s0 = t(k,k-1);  % results for calc error
end
```

Comments on a C-implementation

1. read the source code: main.c, integration.c, ~h, funciton.c, ~h
2. compile via makefile

```
xlhu@TabX:~/workspace/Computing-24/m$ make main
gcc     -O2 -Wall -fomit-frame-pointer -g  -c main.c
gcc     -O2 -Wall -fomit-frame-pointer -g  -c integration.c
gcc     -O2 -Wall -fomit-frame-pointer -g  -c function.c
gcc     -o main main.o integration.o function.o -lm
```

3. run demo and observe the output

```
xlhu@TabX:~/workspace/Computing-24/m$ ./main
Usage:   ./main epsilon
Example: ./main 1e-4
xlhu@TabX:~/workspace/Computing-24/m$ ./main 1e-4
result=3.1415632348954290, error=-2.9419e-05, cputime=0.0000, wtime=0.0000
37 function evaluations.
xlhu@TabX:~/workspace/Computing-24/m$ ./main 1e-6
result=3.1415923484323338, error=-3.0516e-07, cputime=0.0000, wtime=0.0000
333 function evaluations.
xlhu@TabX:~/workspace/Computing-24/m$ ./main 1e-8
result=3.1415926507985246, error=-2.7913e-09, cputime=0.0002, wtime=0.0002
2843 function evaluations.
xlhu@TabX:~/workspace/Computing-24/m$ ./main 1e-9
result=3.1415926532723408, error=-3.1745e-10, cputime=0.0000, wtime=0.0005
10557 function evaluations.
xlhu@TabX:~/workspace/Computing-24/m$ ./main 1e-12
result=3.1415926535894769, error=-3.1619e-13, cputime=0.0199, wtime=0.0199
336113 function evaluations.
```

Other Classical Problems/Models

- Tabular data quadrature. The solution is to interpolate first, and then integrate.
- Singular integration. The solution is to obtain the limitation or do some variable substitution.
- Double and Triple integration. The solution is to use tensor product or simplex.
- High dimensional integration. Monte Carlo method(usually ill-conditioned).

10. Quadrature

11. ODE-based Initial Value Problem

12. ODE-based Boundary Value Problem

Initial Value Problem

*n*th Ordinary Differential Equation

$$y^{(n)} = f(t, y, y', y'', \cdots, y^{(n-1)})$$

could be written into a first order system

$$\mathrm{y}' = \mathrm{f}(t, \mathrm{y}).$$

Equivalently,

$$\mathrm{y} := \begin{bmatrix} y_1'(t) \\ y_2'(t) \\ \cdots \\ y_n'(t) \end{bmatrix} = \begin{bmatrix} f_1(t, \mathrm{y}) \\ f_2(t, \mathrm{y}) \\ \cdots \\ f_n(t, \mathrm{y}) \end{bmatrix} := \mathrm{f}(t, \mathrm{y})$$

- Existence : $\mathrm{y}(t_0) = \mathrm{y}_0$

Initial Value Problem

nth Ordinary Differential Equation

$$y^{(n)} = f(t, y, y', y'', \cdots, y^{(n-1)})$$

could be written into a first order system

$$\mathrm{y}' = \mathrm{f}(t, \mathrm{y}).$$

Equivalently,

$$\mathrm{y} :'= \begin{bmatrix} y_1'(t) \\ y_2'(t) \\ \cdots \\ y_n'(t) \end{bmatrix} = \begin{bmatrix} f_1(t, \mathrm{y}) \\ f_2(t, \mathrm{y}) \\ \cdots \\ f_n(t, \mathrm{y}) \end{bmatrix} := \mathrm{f}(t, \mathrm{y})$$

- Existence : $\mathrm{y}(t_0) = \mathrm{y}_0$

例 9.2　常微分方程　最简单的常微分方程为 $y'=0$，即 $f(t,y)\equiv 0$，这似乎说明什么变化也没有. 这个常微分方程的解为 $y(t)=c, c\in\mathbb{R}^n$ 为任意常数. 解是一族解(由 c 确定). 由这个例子看出常微分方程的解是不惟一的.

另一个简单的常微分方程为 $y'=b$，其中 $b\in\mathbb{R}^n$ 为常量. 这个常微分方程的解为 $y(t)=bt+c, c\in\mathbb{R}^n$ 为任意常量. 图 9.1 画出了 $n=1, b=1/2$ 时，不同 c 值的解. ■

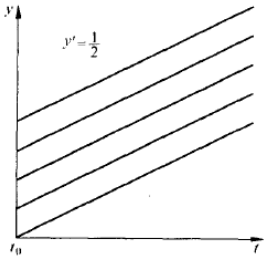


图 9.1　$y'=1/2$ 的部分解

实际中产生的往往是特殊的一阶显式常微分方程. 若 f 不直接与 t 有关，则常微分方程为自治的，并可以写成 $y'=f(y)$ 的形式. 通过引入自变量 $y_{n+1}(t)=t$ 可以将非自治的常微分方程组 $y'=f(t,y)$ 转化成自治形式

$$\begin{bmatrix} y' \\ y'_{n+1} \end{bmatrix}=\begin{bmatrix} f(y_{n+1},y) \\ 1 \end{bmatrix}.$$

如果 f 的形式为 $f(t,y)=A(t)y+b(t)$，其中 $A(t)$ 和 $b(t)$ 分别是关于 t 的矩阵和向量函数，则称常微分方程是线性的. 在线性常微分方程中，若矩阵 A 与 t 无关，则称常微分方程为常系数的，而且，若 $b(t)\equiv 0$，则称常微分方程为齐次的. 这样，一阶常系数齐次线性常微分方程的最简形式为 $y'=Ay$，其中 $A\in\mathbb{R}^{n\times n}$.

例 9.3　化学反应动力学　假设三种化学物质的浓度随时间变化的函数为 $y_1(t)$，$y_2(t)$，$y_3(t)$. 如果 $y_1 \to y_2$ 的反应速度与 y_1 成正比，$y_2 \to y_3$ 的反应速度与 y_2 成正比，则浓度由常系数线性齐次常微分方程

$$\boldsymbol{y}' = \begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} -k_1 y_1 \\ k_1 y_1 - k_2 y_2 \\ k_2 y_2 \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \boldsymbol{A}\boldsymbol{y}$$

决定，其中 k_1 和 k_2 为两个反应的速度常数. 如果某些物质具有补充项 $\boldsymbol{b}(t)$，则这个常微分方程就不是齐次的. ■

- In the literature, people referred to solve by integrating the ODE, its solution is given by the integral

$$y(t) = y_0 + \int_{t_0}^{t} f(s)ds \tag{2}$$

例 9.3　化学反应动力学　假设三种化学物质的浓度随时间变化的函数为 $y_1(t)$，$y_2(t)$，$y_3(t)$. 如果 $y_1 \to y_2$ 的反应速度与 y_1 成正比，$y_2 \to y_3$ 的反应速度与 y_2 成正比，则浓度由常系数线性齐次常微分方程

$$\boldsymbol{y}' = \begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} -k_1 y_1 \\ k_1 y_1 - k_2 y_2 \\ k_2 y_2 \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \boldsymbol{A}\boldsymbol{y}$$

决定，其中 k_1 和 k_2 为两个反应的速度常数. 如果某些物质具有补充项 $b(t)$，则这个常微分方程就不是齐次的. ■

- In the literature, people referred to solve by integrating the ODE, its solution is given by the integral

$$\mathrm{y}(t) = \mathrm{y}_0 + \int_{t_0}^{t} \mathrm{f}(s)ds \tag{2}$$

Example (Newton's Second Law of Motion)

- As well known as the formula $F = ma$, force equals mass times acceleration. It is straightforward that $a = s''(t)$, where t means the time and $s(t)$ is the displacement at time t.
- Considering the trajectory of falling object under the force of Earth's gravity, which means $F = -mg$.
- When the initial position $s(0)$ and initial velocity $v(0) = s'(0)$ are given, we have its solution

$$s(t) = -\frac{1}{2}gt^2 + s'(0)t + s(0).$$

Existence & Uniqueness

Theorem

Suppose that $\mathrm{f} : \mathrm{R}^{n+1} \to \mathrm{R}^n$ *is Lipschitz continuous on* D*, that is*

$$\|\mathrm{f}(t, \hat{\mathrm{y}}) - \mathrm{f}(t, \mathrm{y})\| \leq L\|\hat{\mathrm{y}} - \mathrm{y}\|,$$

For any given initial guess in D*, the solution to IVP is exist and unique!*

对初值问题来说，问题的数据为初值 y_0 及函数 f. 设 $\hat{y}(t)$ 为初值问题 $\hat{y}' = f(t, \hat{y})$ 的解，带有扰动的初始条件为 $\hat{y}(t_0) = \hat{y}_0$. 可以证明，对任意 $t \geqslant t_0$，有

$$\| \hat{y}(t) - y(t) \| \leqslant e^{L(t - t_0)} \| \hat{y}_0 - y_0 \|.$$

如果函数 f 也带有扰动，初值问题变为 $\hat{y}' = \hat{f}(t, \hat{y})$，则对任意 $t \geqslant t_0$，有

$$\| \hat{y}(t) - y(t) \| \leqslant e^{L(t - t_0)} \| \hat{y}_0 - y_0 \| + \frac{e^{L(t - t_0)} - 1}{L} \| \hat{f} - f \|,$$

其中 $\| \hat{f} - f \| = \max_{(t,y) \in D} \| \hat{f}(t, y) - f(t, y) \|$. 上述关于扰动的估计表明初值问题的惟一解是关于问题数据的连续函数，因而问题是适定的.

例 9.6　解的稳定性　对于例 9.2，当 $n=1$ 时，常微分方程为 $y'=b$，b 为给定常数，它的解是一族斜率为 b 的平行直线，如图 9.1 所示. 这个常微分方程的解是稳定的，但并不渐近稳定. 进一步，当 λ 为常数时，常微分方程

$$y'=\lambda y$$

的解为

$$y(t)=y_0 e^{\lambda t},$$

其中 $t_0=0$ 是初始时间，$y(0)=y_0$ 是初始值. 例 9.5 是这个常微分方程在 $\lambda=1$ 时的特殊情形，图 9.2 画出了它的部分解. 当 $\lambda>0$ 时，由于所有非零解都按指数递增，如图 9.2 所示，所以任何两个解之间都不相互收敛，因而每个解都是不稳定的. 另一方面，当 $\lambda<0$ 时，所有非零解都按指数下降，如图 9.3 所示，任何两个解都相互收敛. 因而在这种情形，每个解不仅稳定，而且渐近稳定. 如果 $\lambda=a+ib$ 是复数，则由 1.3.11 节知

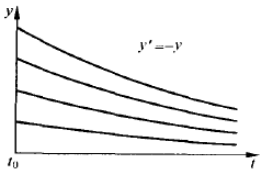


图 9.3　$y'=-y$ 的部分解

$$e^{\lambda t}=e^{(a+ib)t}=e^{at}e^{ibt}=e^{at}(\cos(bt)+i\sin(bt)),$$

这样，解是按指数增加还是减少将由 $\mathrm{Re}\lambda$ 的符号确定. 特别地，若 $\mathrm{Re}\lambda>0$，则解是不稳定的；若 $\mathrm{Re}\lambda<0$，则解是渐近稳定的；如果 $\mathrm{Re}\lambda=0$，则解是振荡的，但由于相互之间保持一定距离，所以虽然稳定，但不渐近稳定. ■

Approximate $f'(x)$ with Divided Difference

- Forward/Backward

$$f'(x) \approx D_h^+ := \frac{f(x+h) - f(x)}{h}$$

- Centeral Scheme:

$$f'(x) \approx D_{2h} := \frac{f(x+h) - f(x-h)}{2h}$$

- One sider(Higher order)

$$f'(x) \approx D_{2h}^+ := \frac{-3f(x) + 4f(x+h) - f(xi+2h)}{2h}$$

It is straightforward by Taylor's Theorem that the error being $o(h)$ and $o(h^2)$.

Approximation of $f''(x)$

$$f''(x) = (f(x+h) - 2f(x) + f(x-h))/h^2.$$

Define $D_h^2 f(x) = Af(x+h) + Bf(x) + Cf(x-h)$, then

$$\begin{cases} A + B + C = 0; \\ h(A - C) = 0; \\ \frac{h^2}{2}(A + C) = 1. \end{cases}$$

Discretization $-\frac{h^2}{12} f^{(4)}(x)$(prove?)

Approximation of $f''(x)$

$$f''(x) = (f(x+h) - 2f(x) + f(x-h))/h^2.$$

Define $D_h^2 f(x) = Af(x+h) + Bf(x) + Cf(x-h)$, then

$$\begin{cases} A + B + C = & 0; \\ h(A - C) = & 0; \\ \frac{h^2}{2}(A + C) = & 1. \end{cases}$$

Discretization $-\frac{h^2}{12} f^{(4)}(x)$(prove?)

Classical Numerical Methods for IVPs

1. Euler's Method
2. Taylor Expansion Method
3. Runge-Kutta Method
4. Linear Multi-step Method

Consider a one dimensional first order ODE system

$$\frac{dy}{dx} = f(x, y), x \in [a, b]$$

$$y(a) = y_0,$$

Fundamental idea:

From an initial value, integrate one step along the tangent direction:

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_{k-1}) = y(x_{k-1}) + h_k f(x_{k-1}, y).$$

The local/truncation error at step k:

$$R_k = \int_{x_{k-1}}^{x_k} f(s, y(s))ds - h_k f(x_{k-1}, y(x_{k-1})),$$

where $h_k := (x_k - x_{k-1})$ is the time steplength.

Consider a one dimensional first order ODE system

$$\frac{dy}{dx} = f(x, y), x \in [a, b]$$

$$y(a) = y_0,$$

Fundamental idea:

From an initial value, integrate one step along the tangent direction:

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_{k-1}) = y(x_{k-1}) + h_k f(x_{k-1}, y).$$

The local/truncation error at step k:

$$R_k = \int_{x_{k-1}}^{x_k} f(s, y(s))ds - h_k f(x_{k-1}, y(x_{k-1})),$$

where $h_k := (x_k - x_{k-1})$ is the time steplength.

Consider a one dimensional first order ODE system

$$\frac{dy}{dx} = f(x, y), x \in [a, b]$$

$$y(a) = y_0,$$

Fundamental idea:

From an initial value, integrate one step along the tangent direction:

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_{k-1}) = y(x_{k-1}) + h_k f(x_{k-1}, y).$$

The local/truncation error at step k:

$$R_k = \int_{x_{k-1}}^{x_k} f(s, y(s))ds - h_k f(x_{k-1}, y(x_{k-1})),$$

where $h_k := (x_k - x_{k-1})$ is the time steplength.

Explicit/Foward Euler's Method

Consider a one dimensional first order ODE system

$$\frac{dy}{dx} = f(x, y), x \in [a, b]$$

$$y(a) = y_0,$$

Fundamental idea:

From an initial value, integrate one step along the tangent direction:

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_{k-1}) = y(x_{k-1}) + h_k f(x_{k-1}, y).$$

The local/truncation error at step k:

$$R_k = \int_{x_{k-1}}^{x_k} f(s, y(s))ds - h_k f(x_{k-1}, y(x_{k-1})),$$

where $h_k := (x_k - x_{k-1})$ is the time steplength.

Example (Explicit/Foward Euler Method for IVPs)

Considering the following initial problem:

$$y' = y, y(0) = 1.$$

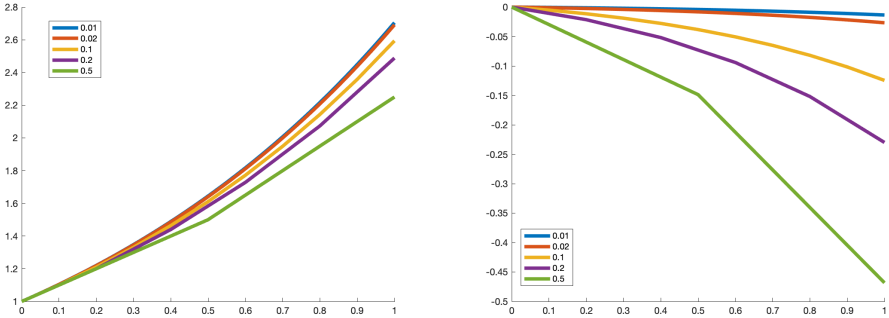


Figure: (Left)The Euler solution, (Right) Local/Truncation Errors

Global and Local Errors

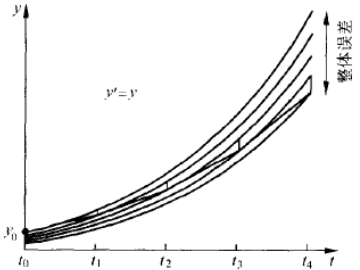


图 9.6 求解 $y'=y$ 的欧拉方法的局部误差和整体误差

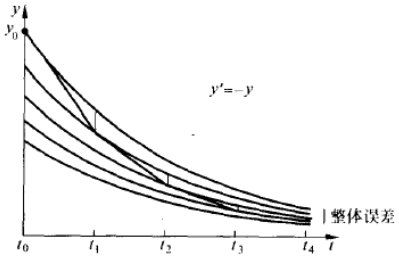


图 9.7 求解 $y'=-y$ 的欧拉方法的局部误差和整体误差

The global error is not simply the summation or integration of local error:

- IF divergence, sum of the local error $<$ total error
- IF convergence, total error $<$ sum of the local error

Global and Local Errors

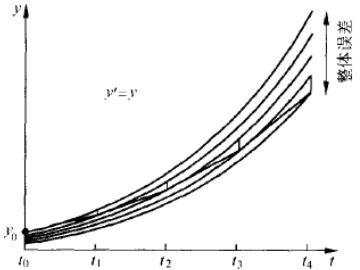


图 9.6 求解 $y'=y$ 的欧拉方法的局部误差和整体误差

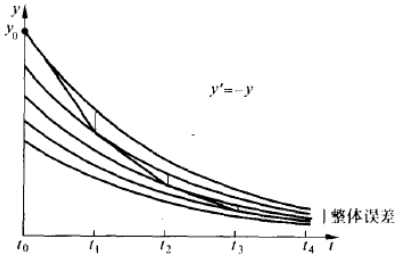


图 9.7 求解 $y'=-y$ 的欧拉方法的局部误差和整体误差

The global error is not simply the summation or integration of local error:

- IF divergence, sum of the local error $<$ total error
- IF convergence, total error $<$ sum of the local error

Explicit Runge-Kutta

The s-stage explicit Runge-Kutta scheme could be written as

$$\begin{aligned} y_{m+1} &= y_m + h(b_1 k_1 + \cdots + b_s k_s), \\ k_1 &= f(x_m, y_m), \\ k_2 &= f(x_m + c_2 h, y_m + h a_{2,1} k_1), \\ k_3 &= f(x_m + c_3 h, y_m + h(a_{3,1} k_1 + a_{3,2} k_2)), \\ &\cdots\cdots \\ k_s &= f(x_m + c_s h, y_m + h(a_{s,1} k_1 + \cdots + a_{s,s-1} k_s - 1)), \end{aligned}$$

where $a_{i,j}, c_i (i - 2, 3, \ldots, s; j < i)$ and $b_i (i = 1, 2, \ldots, s)$ be real parameters.

- the parameters are determined by the methods of Taylor's expansion
- to promise the accuracy, avoid to utilize the high-order derivatives
- the weights are not unique, and Heun's and Gill's formulas are helpful

Explicit Runge-Kutta

The s-stage explicit Runge-Kutta scheme could be written as

$$\begin{aligned} y_{m+1} &= y_m + h(b_1 k_1 + \cdots + b_s k_s), \\ k_1 &= f(x_m, y_m), \\ k_2 &= f(x_m + c_2 h, y_m + h a_{2,1} k_1), \\ k_3 &= f(x_m + c_3 h, y_m + h(a_{3,1} k_1 + a_{3,2} k_2)), \\ &\cdots\cdots \\ k_s &= f(x_m + c_s h, y_m + h(a_{s,1} k_1 + \cdots + a_{s,s-1} k_s - 1)), \end{aligned}$$

where $a_{i,j}, c_i (i - 2, 3, \ldots, s; j < i)$ and $b_i (i = 1, 2, \ldots, s)$ be real parameters.

- the parameters are determined by the methods of Taylor's expansion
- to promise the accuracy, avoid to utilize the high-order derivatives
- the weights are not unique, and Heun's and Gill's formulas are helpful

Performing R-K Calculation ...

Solve the following ODE numerical via Runge-Kutta method:

$$y' = -2ty^2, y(0) = 1$$

Starting from $t_0 = 0$ to $t_1 = 0.25$ with time-step length $h = 0.25$,

1. we have $k_1 = f(t_0, y_0) = 0, k_2 = f(t_0 + h, y_0 + hk) = -0.5$. ...
2. the analytic solution for the current problem is $y(t) = \frac{1}{1+t^2}$,
3. one can evaluate at the specific time, for example $y(0.25) = 0.9412, y(0.5) = 0.8$ then calculate the numerical errors.

Performing R-K Calculation ...

Solve the following ODE numerical via Runge-Kutta method:

$$y' = -2ty^2, y(0) = 1$$

Starting from $t_0 = 0$ to $t_1 = 0.25$ with time-step length $h = 0.25$,

1. we have $k_1 = f(t_0, y_0) = 0, k_2 = f(t_0 + h, y_0 + hk) = -0.5$. ...
2. the analytic solution for the current problem is $y(t) = \frac{1}{1+t^2}$,
3. one can evaluate at the specific time, for example $y(0.25) = 0.9412, y(0.5) = 0.8$ then calculate the numerical errors.

Classical 4*th* order R-K Scheme

The most famous numerical methods for ordinary differential equation is the forth order Runge-Kutta formula, which is

$$y_{k+1} = y_k + \frac{h_k}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{3}$$

with the definition of momentum

$$\begin{aligned} k_1 &= f(t_k, y_k), \\ k_2 &= f(t_k + \frac{1}{2}h_k, y_k + \frac{1}{2}h_k k_1), \\ k_3 &= f(t_k + \frac{1}{2}h_k, y_k + \frac{1}{2}h_k k_2), \\ k_4 &= f(t_k + h_k, y_k + h_k k_3), \end{aligned}$$

Classical 4*th* order R-K Scheme

The most famous numerical methods for ordinary differential equation is the forth order Runge-Kutta formula, which is

$$y_{k+1} = y_k + \frac{h_k}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{3}$$

with the definition of momentum

$$\begin{aligned} k_1 &= f(t_k, y_k), \\ k_2 &= f(t_k + \frac{1}{2}h_k, y_k + \frac{1}{2}h_k k_1), \\ k_3 &= f(t_k + \frac{1}{2}h_k, y_k + \frac{1}{2}h_k k_2), \\ k_4 &= f(t_k + h_k, y_k + h_k k_3), \end{aligned}$$

Example on explicit Runge-Kutta method - conti.

Let us plot the numerical solution as well as it's error

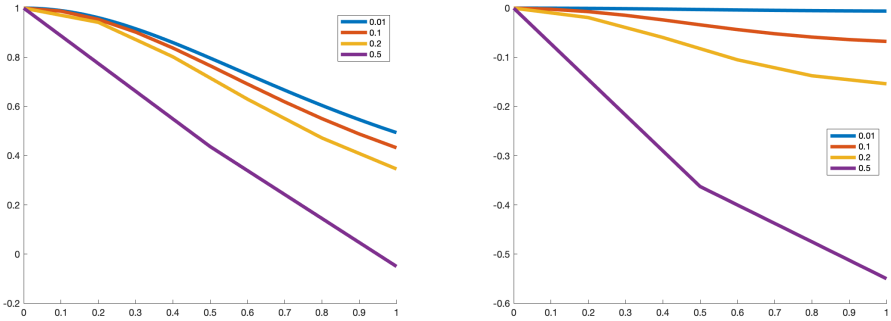


Figure: (Left)The Runge-Kutta solution with Heun's formula, (Right) Errors

Illustration on MATLAB solver: ode45

Example (Logistic Equation)

To modeling the growth of population via

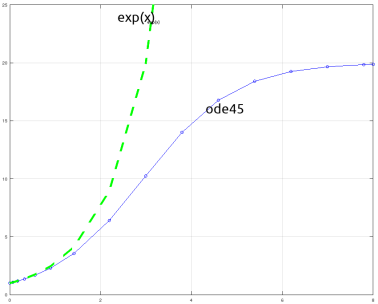
$$y_t' = \kappa y(1 - \frac{y}{\mu})$$

and given $y(0) = \eta$, analytic solution is

$$y(t) = \frac{\mu\eta e^{\kappa t}}{\eta e^{\kappa t} + \mu - \eta}.$$

It means that $y(t) \to \mu(capacity)$ if $t \to \infty$.

```
1  kappa = 1; eta = 1; mu = 20;
2  f = @(t,y) kappa*y*(1-y/mu);
3  [t,y] = ode45(f, [0, 8], eta);
4  plot(t,y,'o-b',t, exp(t),'g--', '
       linewidth',5)
5  axis([0,8, 0,25]); grid on;
```



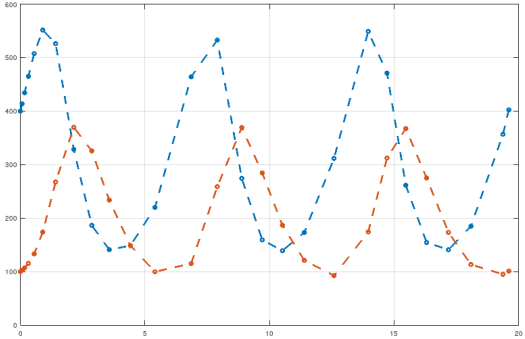
Example (Predator-Prey Model)

$$
\begin{aligned}
y_1' &= \left(1 - \frac{y_2}{\mu_2}\right) y_1, \\
y_2' &= -\left(1 - \frac{y_1}{\mu_1}\right) y_2
\end{aligned}
$$

with initial population state

$y_1(0) = \eta_1, y_2(0) = \eta_2$.

```
1  mu = [300, 200]'; eta = [400, 100]';
2  f = @(t,y) [(1-y(2)/mu(2))*y(1), ...
3              -(1-y(1)/mu(1))*y(2)]';
4  T = 6.5357;
5  [t,y] = ode45(f, [0, 3*T], eta);
6  plot(t,y,'--o','linewidth',3, ...
7           'markersize',5);
8  grid on; axis tight;
```



Stiffness

Eigenvalues of The Jacobi matrix of ODE are very different from each other

Example (Stiff ODE)

$$y' = -100y + 100x + 101, y(0) = 1.$$

Solve it with the Euler's scheme with noisy initial data:

x	0	0.1	0.2	0.3	0.4
Analytic Solution	1	1.1	1.2	1.3	1.4
Explicit Euler Method	1	1.1	1.2	1.3	1.4
Explicit Euler Method	0.99	1.19	0.39	8.59	-64.21
Explicit Euler Method	1.01	1.01	2.01	-5.99	67.01

Stiffness

Eigenvalues of The Jacobi matrix of ODE are very different from each other

Example (Stiff ODE)

$$y' = -100y + 100x + 101, y(0) = 1.$$

Solve it with the Euler's scheme with noisy initial data:

x	0	0.1	0.2	0.3	0.4
Analytic Solution	1	1.1	1.2	1.3	1.4
Explicit Euler Method	1	1.1	1.2	1.3	1.4
Explicit Euler Method	0.99	1.19	0.39	8.59	-64.21
Explicit Euler Method	1.01	1.01	2.01	-5.99	67.01

Supress the Stiffness

相反，用向后欧拉方法求解这个问题就比较合适. 向后欧拉方法的解对初始值极不敏感，这一点从下表以及图 9.10 可以看出. 由于用的是下一点的导数而不用当前点，所以即使初始值带有很大的扰动，退化也会很快消除，并且仅经过几步，向后欧拉方法的解就会收敛到所需的解. 这种现象符合稳定问题的向后欧拉方法的无条件稳定特性.

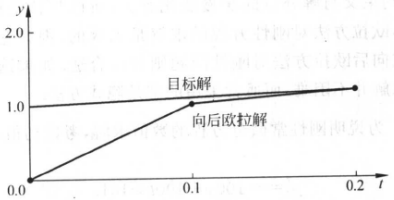


图 9.10 刚性常微分方程的向后欧拉方法得到的稳定解

t	0.0	0.1	0.2	0.3	0.4
精确解	1.00	1.10	1.20	1.30	1.40
向后欧拉解	0.00	1.01	1.19	1.30	1.40
向后欧拉解	2.00	1.19	1.21	1.30	1.40

Implicit Schemes

To compute k_i with all values of k

$$y_{m+1} = y_m + h(b_1 k_1 + \cdots + b_s k_s),$$

where k_i satisfies

$$k_i = f(x_m + c_i h, y_m + h(a_{i,1} k_1 + \cdots + a_{i,s} k_s)).$$

$$\begin{array}{c|cccc} c_1 & 0 & & & \\ c_2 & a_{21} & \ddots & 0 & \\ \vdots & \vdots & \ddots & \ddots & \\ c_s & a_{s1} & \cdots & a_{ss-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

$$\begin{array}{c|cccc} \frac{1}{5} & \frac{1}{5} & & & \\ \frac{7}{18} & \frac{7}{648} & \frac{245}{648} & & \\ \frac{19}{20} & \frac{2489}{1400} & -\frac{5567}{1360} & \frac{155439}{47600} & \\ 1 & \frac{19659}{6650} & -\frac{1181}{170} & \frac{1527174}{300475} & -\frac{176}{1919} \\ \hline & \frac{97}{798} & 0 & \frac{4374}{7777} & \frac{4000}{5757} & -\frac{25}{66} \end{array}$$

Figure: (left) Butcher table, (middle) Explicit RK4, (right) an implicit RK4.

Example (Implicit Euler Scheme)

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_k)$$

unconditional stable, however, be first order method

$$\left|\frac{1}{1-h\lambda}\right| = 1 + h\lambda + (h\lambda)^2 + \cdots < 1$$

Example (Mid-Point Scheme)

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})(y'(x_{k-1}) + y'(x_k))/2$$

unconditional stable, second order method

$$\left|\frac{1 + h\lambda/2}{1 - h\lambda/2}\right| = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{4} + \cdots < 1$$

Example (Implicit Euler Scheme)

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})y'(x_k)$$

unconditional stable, however, be first order method

$$\left|\frac{1}{1-h\lambda}\right| = 1 + h\lambda + (h\lambda)^2 + \cdots < 1$$

Example (Mid-Point Scheme)

$$y(x_k) = y(x_{k-1}) + (x_k - x_{k-1})(y'(x_{k-1}) + y'(x_k))/2$$

unconditional stable, second order method

$$\left|\frac{1+h\lambda/2}{1-h\lambda/2}\right| = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{4} + \cdots < 1$$

多保留一项，可得到二阶方法

$$\boldsymbol{y}_{k+1}=\boldsymbol{y}_k+h_k\boldsymbol{y}'_k+\frac{h_k^2}{2}\boldsymbol{y}''_k.$$

需要注意，这种方法需要计算 $\boldsymbol{y}$ 的高阶导数. 我们可以通过链式法则对 $\boldsymbol{y}'=\boldsymbol{f}(t,\boldsymbol{y})$ 求导，即

$$\boldsymbol{y}''=\boldsymbol{f}_t(t,\boldsymbol{y})+\boldsymbol{f}_{\boldsymbol{y}}(t,\boldsymbol{y})\boldsymbol{y}'=\boldsymbol{f}_t(t,\boldsymbol{y})+\boldsymbol{f}_{\boldsymbol{y}}(t,\boldsymbol{y})\boldsymbol{f}(t,\boldsymbol{y}),$$

其中的下标表示的是对给定变量求偏导. 随着阶数的增加，导数的表达式将变得非常复杂而难于求得，所以高阶泰勒级数方法并不实用. 但近年来由于符号操作以及自动求导系统的产生使得这类方法更加可行.

例 9.11　泰勒级数法　为说明泰勒级数法，我们用它来求解一维非线性常微分方程

$$y'(t)=f(t,y)=-2ty^2,$$

初值 $y(0)=1$. 对 f 求导，得

$$y''=f_t(t,y)+f_y(t,y)f(t,y)=-2y^2+(-4ty)(-2ty^2)=2y^2(4t^2y-1).$$

从 $t_0=0$ 开始，取 $h=0.25$，则 $t_1=0.25$，得

$$y_1=y_0+hy'_0+\frac{h^2}{2}y''_0=1+0-0.0625=0.9375.$$

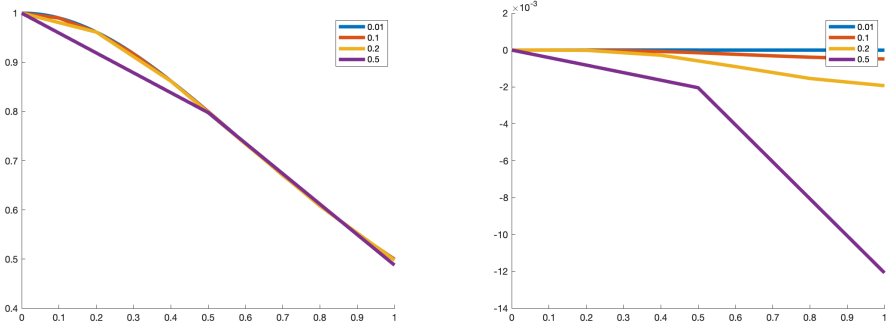
继续下一步，从 $t_1=0.25$ 到 $t_2=0.5$，得

$$y_2=y_1+hy'_1+\frac{h^2}{2}y''_1=0.9375-0.1099-0.0421=0.7856.$$

Example (implicit Runge-Kutta method)

Using Hammer method to solve the following ODE:

$$y' = -2ty^2, y(0) = 1$$



Comments on one-step method

$$y_{m+1} = y_m + h\varphi(x_m, y_m, h),$$

- Compatibility: The one-step methods is compatible if the increment function φ satisfies:

$$\varphi(x, y, 0) = f(x, y).$$

- Stability: The one-step methods is stable if for any $(x, y) \in \Omega$ and $h < H$, φ satisfies y's Lipschitz continuous condition.
- Convergence:The one-step methods is convergent if there satisfies

$$\lim_{h \to 0} y_m = y(x), x = x_m,$$

moreover, the convergence is equal to compatibility if φ satisfies Lipschitz condition about x and h.

Comments on one-step method

$$y_{m+1} = y_m + h\varphi(x_m, y_m, h),$$

- Compatibility: The one-step methods is compatible if the increment function φ satisfies:

$$\varphi(x, y, 0) = f(x, y).$$

- Stability: The one-step methods is stable if for any $(x, y) \in \Omega$ and $h < H$, φ satisfies y's Lipschitz continuous condition.
- Convergence:The one-step methods is convergent if there satisfies

$$\lim_{h \to 0} y_m = y(x), x = x_m,$$

moreover, the convergence is equal to compatibility if φ satisfies Lipschitz condition about x and h.

Comments on one-step method

$$y_{m+1} = y_m + h\varphi(x_m, y_m, h),$$

- Compatibility: The one-step methods is compatible if the increment function φ satisfies:

$$\varphi(x, y, 0) = f(x, y).$$

- Stability: The one-step methods is stable if for any $(x, y) \in \Omega$ and $h < H$, φ satisfies y's Lipschitz continuous condition.
- Convergence:The one-step methods is convergent if there satisfies

$$\lim_{h \to 0} y_m = y(x), x = x_m,$$

moreover, the convergence is equal to compatibility if φ satisfies Lipschitz condition about x and h.

Comments on one-step method

$$y_{m+1} = y_m + h\varphi(x_m, y_m, h),$$

- Compatibility: The one-step methods is compatible if the increment function φ satisfies:

$$\varphi(x, y, 0) = f(x, y).$$

- Stability: The one-step methods is stable if for any $(x, y) \in \Omega$ and $h < H$, φ satisfies y's Lipschitz continuous condition.
- Convergence:The one-step methods is convergent if there satisfies

$$\lim_{h \to 0} y_m = y(x), x = x_m,$$

moreover, the convergence is equal to compatibility if φ satisfies Lipschitz condition about x and h.

Linear Multi-step Method

To solve $y' = f(x, y)$, let $y_m = y(x_m), f_m = f(x_m, y_m)$ and then

$$y_{m+1} = \sum_{i=1}^{k} \alpha_i y_{m-i+1} + h\Phi(x_{m+1}, x_m, \ldots, x_{m-k+1}, y'_{m+1}, \ldots, y'_{m-k+1}; h),$$

where $k \in N^+$, $\{\alpha_i\}$ is a given real number, h is the steplength. Then the *truncation error* is

$$R(x_m, y_m, h) = y_{m+1} - \sum_{i=1}^{k} \alpha_i y_{m-i+1} - h\Phi(\cdots; h)$$

- p*th*-order : the most large number to make $O(h^{p+1})$ established
- Adams extrapolation : explicit
- Adams interpolation : implicit

Linear Multi-step Method

To solve $y' = f(x, y)$, let $y_m = y(x_m), f_m = f(x_m, y_m)$ and then

$$y_{m+1} = \sum_{i=1}^{k} \alpha_i y_{m-i+1} + h\Phi(x_{m+1}, x_m, \ldots, x_{m-k+1}, y'_{m+1}, \ldots, y'_{m-k+1}; h),$$

where $k \in N^+$, $\{\alpha_i\}$ is a given real number, h is the steplength. Then the *truncation error* is

$$R(x_m, y_m, h) = y_{m+1} - \sum_{i=1}^{k} \alpha_i y_{m-i+1} - h\Phi(\cdots; h)$$

- p*th*-order : the most large number to make $O(h^{p+1})$ established
- Adams extrapolation : explicit
- Adams interpolation : implicit

Linear Multi-step Method

To solve $y' = f(x, y)$, let $y_m = y(x_m), f_m = f(x_m, y_m)$ and then

$$y_{m+1} = \sum_{i=1}^{k} \alpha_i y_{m-i+1} + h\Phi(x_{m+1}, x_m, \ldots, x_{m-k+1}, y'_{m+1}, \ldots, y'_{m-k+1}; h),$$

where $k \in N^+$, $\{\alpha_i\}$ is a given real number, h is the steplength. Then the *truncation error* is

$$R(x_m, y_m, h) = y_{m+1} - \sum_{i=1}^{k} \alpha_i y_{m-i+1} - h\Phi(\cdots; h)$$

- p*th*-order : the most large number to make $O(h^{p+1})$ established
- Adams extrapolation : explicit
- Adams interpolation : implicit

Linear Multi-step Method

To solve $y' = f(x, y)$, let $y_m = y(x_m), f_m = f(x_m, y_m)$ and then

$$y_{m+1} = \sum_{i=1}^{k} \alpha_i y_{m-i+1} + h\Phi(x_{m+1}, x_m, \ldots, x_{m-k+1}, y'_{m+1}, \ldots, y'_{m-k+1}; h),$$

where $k \in N^+$, $\{\alpha_i\}$ is a given real number, h is the steplength. Then the *truncation error* is

$$R(x_m, y_m, h) = y_{m+1} - \sum_{i=1}^{k} \alpha_i y_{m-i+1} - h\Phi(\cdots; h)$$

- p*th*-order : the most large number to make $O(h^{p+1})$ established
- Adams extrapolation : explicit
- Adams interpolation : implicit

Example (p-order k-step linear multi-step method)

For the scheme

$$y_m = \sum_{i=1}^{k-1} \alpha_i y_{m-i} + h \sum_{i=0}^{k} \beta_i y'_{m-i},$$

we have if $\beta_0 = 0$, the scheme is explicit while $\beta_0 \neq 0$ the scheme is implicit. Let us consider the explicit scheme first, there are $2k-1$ parameters so that we need $\phi_j(x) = x^{j-1}, j = 1, 2, \ldots, 2k-1$.

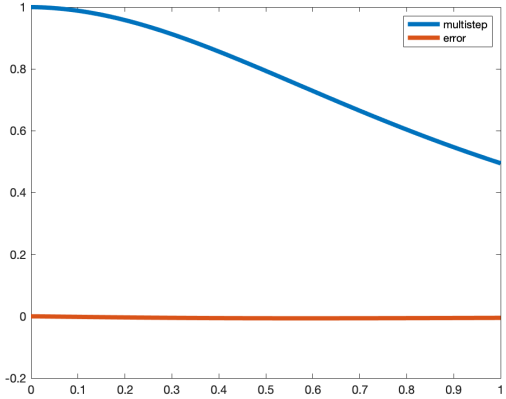
$$x_m^{j-1} = \sum_{i=1}^{k-1} \alpha_i x_{m-i}^{j-1} + h \sum_{i=1}^{k} \beta_i x_{m-i}^{j-2}, j = 1, 2, \ldots, 2k-1,$$

the matrix form can be written as $Au = b$

Example (Linear multi-step method)

To solve the following ODE with linear multi-step method

$$y' = -2xy^2, \quad y(0) = 1 \quad \Rightarrow \quad y_{k+1} = y_k + \frac{h}{2}(3y'_k - y'_{k-1}), \quad y_0 = 1.$$



M-code for Linear multi-step method

```
f = @(x,y)-2*x.*y.^2;y0 = 1;y1 = 0.9999;
h = 0.01; x = 0:h:1; temp1 = y0; temp2 = y1;
y = zeros(1,length(x)); y(1) = y0; y(2) = y1;
for j = 3:length(x)
    y(j) = temp2 + h/2*(3*f(x(j),temp2) - f(x(j-1),
           temp1));
    temp1 = temp2; temp2 = y(j);
end
ye = 1./(1+x.^2);
plot(x,y,'linewidth',4);
hold on
plot(x,y-ye,'linewidth',4);
legend('multistep','error');
```

10. Quadrature

11. ODE-based Initial Value Problem

12. ODE-based Boundary Value Problem

ODE-based Boundary Value Problems(BVPs)

Let us now consider simple form of BVPs, i.e.,

$$u'' = f(x, u, u'), a < x < b. \tag{4}$$

- Dirichlet boundary condition $u(a) = \alpha, u(b) = \beta$ is engaged for convenience
- Choice of β will play an important role on the uniqueness and stability
- Rigorous proof is complecated, let us explain with a simple example

ODE-based Boundary Value Problems(BVPs)

Let us now consider simple form of BVPs, i.e.,

$$u'' = f(x, u, u'), a < x < b. \tag{4}$$

- Dirichlet boundary condition $u(a) = \alpha, u(b) = \beta$ is engaged for convenience
- Choice of β will play an important role on the uniqueness and stability
- Rigorous proof is complecated, let us explain with a simple example

ODE-based Boundary Value Problems(BVPs)

Let us now consider simple form of BVPs, i.e.,

$$u'' = f(x, u, u'), a < x < b. \tag{4}$$

- Dirichlet boundary condition $u(a) = \alpha, u(b) = \beta$ is engaged for convenience
- Choice of β will play an important role on the uniqueness and stability
- Rigorous proof is complecated, let us explain with a simple example

ODE-based Boundary Value Problems(BVPs)

Let us now consider simple form of BVPs, i.e.,

$$u'' = f(x, u, u'), a < x < b. \tag{4}$$

- Dirichlet boundary condition $u(a) = \alpha, u(b) = \beta$ is engaged for convenience
- Choice of β will play an important role on the uniqueness and stability
- Rigorous proof is complecated, let us explain with a simple example

Existence & Uniqueness

Example (A benchmark problem)

$$u'' = -u, 0 < x < 1. \tag{5}$$

The Dirichlet boundary condition is $u(0) = 0, u(b) = \beta$ is considered for convenience. In this sense, the exact solution of this problem is written as $u(x) = c \sin x$, where c is any constant. β will play an important role on the uniqueness and stability.

- there are infinite solutions of this boundary problem when $\beta = 0$
- it is unsolvable when $\beta \neq 0$, if $b = n\pi$

Example (A benchmark problem)

$$u'' = -u, 0 < x < 1. \tag{5}$$

The Dirichlet boundary condition is $u(0) = 0, u(b) = \beta$ is considered for convenience. In this sense, the exact solution of this problem is written as $u(x) = c \sin x$, where c is any constant. β will play an important role on the uniqueness and stability.

- there are infinite solutions of this boundary problem when $\beta = 0$
- it is unsolvable when $\beta \neq 0$, if $b = n\pi$

Example (A benchmark problem)

$$u'' = -u, 0 < x < 1. \tag{5}$$

The Dirichlet boundary condition is $u(0) = 0, u(b) = \beta$ is considered for convenience. In this sense, the exact solution of this problem is written as $u(x) = c \sin x$, where c is any constant. β will play an important role on the uniqueness and stability.

- there are infinite solutions of this boundary problem when $\beta = 0$
- it is unsolvable when $\beta \neq 0$, if $b = n\pi$

接下来考虑扰动问题

$$\hat{\boldsymbol{y}}'=\boldsymbol{A}(t)\hat{\boldsymbol{y}}+\hat{\boldsymbol{b}}(t),\qquad a<t<b,$$

满足边界条件

$$\boldsymbol{B}_a\hat{\boldsymbol{y}}(a)+\boldsymbol{B}_b\hat{\boldsymbol{y}}(b)=\hat{\boldsymbol{c}}.$$

则解的扰动 $\boldsymbol{z}(t)=\hat{\boldsymbol{y}}(t)-\boldsymbol{y}(t)$，满足边值问题

$$\boldsymbol{z}'=\boldsymbol{A}(t)\boldsymbol{z}+\Delta\boldsymbol{b}(t),\qquad a<t<b,$$

边界条件为

$$\boldsymbol{B}_a\boldsymbol{y}(a)+\boldsymbol{B}_b\boldsymbol{y}(b)=\Delta\boldsymbol{c},$$

其中 $\Delta\boldsymbol{b}(t)=\hat{\boldsymbol{b}}(t)-\boldsymbol{b}(t)$，$\Delta\boldsymbol{c}=\hat{\boldsymbol{c}}-\boldsymbol{c}$. 由此可以得到解的扰动估计式

$$\|\boldsymbol{z}\|_\infty\leqslant\kappa\Big(|\Delta\boldsymbol{c}|+\int_a^b|\Delta\boldsymbol{b}(s)|\,\mathrm{d}s\Big).$$

这样，κ 是边值问题的关于常微分方程中非齐次项及边界条件扰动的绝对条件数.

Example

例 10.5　稳定边值问题　考虑常微分方程的初值问题

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & -\lambda \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

其中 $\mathrm{Re}\lambda \neq 0$，它的两种模式中的一个按指数增长，所以是不稳定的. 但如果 $\mathrm{Re}\lambda < 0$，并且给出在区间$[0,b]$上的边界条件

$$y_1(0) = 1, \quad y_2(b) = 1,$$

则即使 b 非常大，边值问题也是稳定的，因为解的增长将受到约束条件的限制. ■

Classical Numerical Methods

1. Finite Difference Method
2. Collocation Method
3. Galerkin Method
4. Shooting Method

Finite Difference Method

- Consider a uniform grid

$$x_i = a + kh, \quad k = 0, 1, \ldots, n+1$$

for interval $[a, b]$, where $h = (b-a)/(n+1)$ is the grid size.

- The purpose of numerical solution to two-point boundary value problem

$$u'' = -3u + 2\cos x, \quad u(0) = 1, u(\pi) = -1,$$

is to find approximations $u_k \approx u(x_k), k = 1, \ldots, n$ in the condition of given boundary $u_0 = u(a) = \alpha$ and $u_{n+1} = u(b) = \beta$.

- A center divided scheme yields

$$u_{k-1} - (2 + 3h^2)u_k + u_{k+1} = -2h^2 \cos(x_k), \quad = 1, 2, \ldots, n.$$

Finite Difference Method

- Consider a uniform grid

$$x_i = a + kh, \quad k = 0, 1, \ldots, n+1$$

for interval $[a, b]$, where $h = (b-a)/(n+1)$ is the grid size.

- The purpose of numerical solution to two-point boundary value problem

$$u'' = -3u + 2\cos x, \quad u(0) = 1, u(\pi) = -1,$$

is to find approximations $u_k \approx u(x_k), k = 1, \ldots, n$ in the condition of given boundary $u_0 = u(a) = \alpha$ and $u_{n+1} = u(b) = \beta$.

- A center divided scheme yields

$$u_{k-1} - (2 + 3h^2)u_k + u_{k+1} = -2h^2 \cos(x_k), \quad = 1, 2, \ldots, n.$$

Finite Difference Method

- Consider a uniform grid

$$x_i = a + kh, \quad k = 0, 1, \ldots, n+1$$

for interval $[a, b]$, where $h = (b-a)/(n+1)$ is the grid size.

- The purpose of numerical solution to two-point boundary value problem

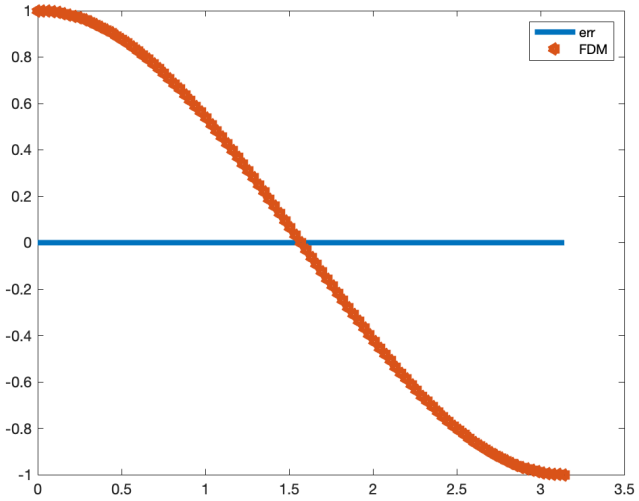
$$u'' = -3u + 2\cos x, \quad u(0) = 1, u(\pi) = -1,$$

is to find approximations $u_k \approx u(x_k), k = 1, \ldots, n$ in the condition of given boundary $u_0 = u(a) = \alpha$ and $u_{n+1} = u(b) = \beta$.

- A center divided scheme yields

$$u_{k-1} - (2 + 3h^2)u_k + u_{k+1} = -2h^2 \cos(x_k), \quad = 1, 2, \ldots, n.$$

Computational Results with $n = 20$



More Examples on Finite Difference - Eigenvalue Problem

$$u''=\lambda g(t)u, \qquad a<t<b,$$

满足齐次边值条件

$$u(a)=0, \quad u(b)=0.$$

引入分割点 $t_i=a+ih$，$i=0,\cdots,n+1$，其中 $h=(b-a)/(n+1)$，用标准的有限差分代替二阶导数，得到代数方程组

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}=\lambda g_i y_i, \qquad i=1,\cdots,n,$$

其中 $y_i=u(t_i)$，$g_i=g(t_i)$. 利用边界条件，有 $y_0=u(a)=0$，$y_{n+1}=u(b)=0$. 如果对 $i=1,\cdots,n$，$g_i\neq 0$，则可用 g_i 去除第 i 个方程，得到标准的代数特征值问题 $\boldsymbol{A}\boldsymbol{y}=\lambda\boldsymbol{y}$，其中 $\boldsymbol{A}$ 具有三对角形式

$$\boldsymbol{A}=\frac{1}{h^2}\begin{bmatrix} -2/g_1 & 1/g_1 & 0 & \cdots & 0 \\ 1/g_2 & -2/g_2 & 1/g_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1/g_{n-1} & -2/g_{n-1} & 1/g_{n-1} \\ 0 & \cdots & 0 & 1/g_n & -2/g_n \end{bmatrix}.$$

这样可以用第 4 章中讨论的方法求解. 另外，也可以直接从离散化方程组的原始形式出发，将其看成广义的特征值问题(见 4.6 节).

Collocation Method

The fudamental idea of collocation method is to find a solution $u(x) = \sum_{i=1}^{n} u_i \phi_i(x)$, such that

$$u'' = f(x, u, u'), u(a) = \alpha, u(b) = \beta, a < t < b,$$

where ϕ_i is basic function in interval $[a, b]$, U_i is undetermined coefficient.

It yields

$$\sum_{i=1}^{n} u_i \phi_i''(x_i) = f\left(x_i, \sum_{i=1}^{n} u_i \phi_i(x_i), \sum_{i=1}^{n} u_i \phi_i'(x_i)\right), i = 2, \ldots, n-1.$$

Ffollowing the boundary condition, there yields

$$\sum_{i=1}^{n} u_i \phi_i(a) = \alpha, \sum_{i=1}^{n} u_i \phi_i(b) = \beta$$

.

Collocation Method

The fudamental idea of collocation method is to find a solution $u(x) = \sum_{i=1}^{n} u_i \phi_i(x)$, such that

$$u'' = f(x, u, u'), u(a) = \alpha, u(b) = \beta, a < t < b,$$

where ϕ_i is basic function in interval $[a, b]$, U_i is undetermined coefficient.

It yields

$$\sum_{i=1}^{n} u_i \phi_i''(x_i) = f\left(x_i, \sum_{i=1}^{n} u_i \phi_i(x_i), \sum_{i=1}^{n} u_i \phi_i'(x_i)\right), i = 2, \ldots, n-1.$$

Ffollowing the boundary condition, there yields

$$\sum_{i=1}^{n} u_i \phi_i(a) = \alpha, \sum_{i=1}^{n} u_i \phi_i(b) = \beta$$

.

Collocation Method - conti.

Rewriting the problem as: $\mathcal{L}u = g(x)$, the matrix form is $Aw = b$, where

$$A = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \mathcal{L}\phi_1(x_2) & \mathcal{L}\phi_2(x_2) & \dots & \mathcal{L}\phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{L}\phi_1(x_{n-1}) & \mathcal{L}\phi_2(x_{n-1}) & \dots & \mathcal{L}\phi_n(x_{n-1}) \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_n(x_n) \end{bmatrix}, w = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, b = \begin{bmatrix} \alpha \\ g(x_2) \\ \vdots \\ g(x_{n-1}) \\ \beta \end{bmatrix},$$

and the solution is computed as $u = Kw$ with

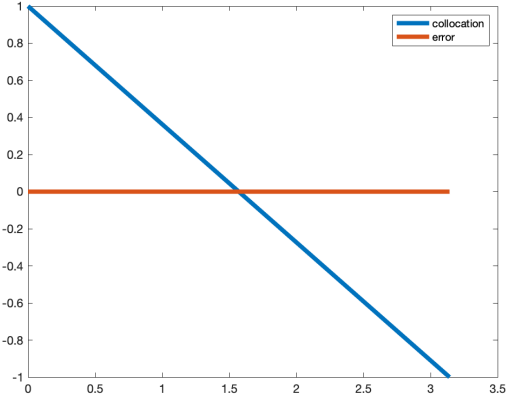
$$K = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_n(x_n) \end{bmatrix},$$

Example (collocation)

Using the method of collocation so solve the following boundary problem:

$$u'' = -3u + 2\cos t, u(0) = 1, u(\pi) = -1.$$

Choosing $h = \frac{\pi}{2}$, then the solution are shown as following:



A example in the textbook

例 10.8　配置法　用例 10.6 和例 10.7 的两点边值问题

$$u''=6t,\qquad 0<t<1$$

满足边值条件 $u(0)=0,u(1)=1$ 来说明配置法.

为减少计算量，只取一个内部配置点，即区间$[a,b]$的中点，加上边界点，共有三个配置点 $t_1=0,t_2=0.5,t_3=1$，所以要确定 3 个参数. 用前三个单项式作为基函数，近似解的形式为

$$v(t,\boldsymbol{x})=x_1+x_2t+x_3t^2.$$

结果与前面用打靶法(例 10.6)和有限差分法(例 10.7)得到的结果是一致的. 通常，这三种方法得到的结果不一定完全相同，但由于这个特殊问题的性质，它们的结果是一致的. 容易看出这个问题的真解为 $u(t)=t^3$，所以值 $u(0.5)=(0.5)^3=0.125$ 实际上是精确的. 注意到由配置法得到的二次多项式在 $t_1=0,t_2=0.5,t_3=1$ 三个点与真解取值相同，但在其他点上却不相同(为什么?)，图 10.3 画出了近似解和真解的图象.

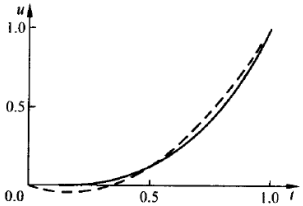


图 10.3　例 10.8 的真解(实线)和配置解(虚线)

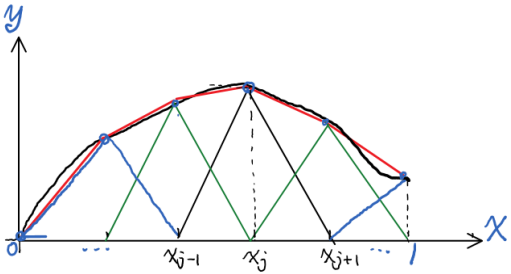
Galerkin Method

Let us show it by solving a two-point boundary value problem

$$-\frac{d^2}{dx^2}u(x) = f(x), x \in (0,1); \quad \text{with} \quad u(0) = u(1) = 0. \tag{6}$$

- nonuniform grid $0 = x_0 < x_1 < \cdots < x_{n+1} = 1$ with $h_i = (x_i - x_{i-1})$
- basis functions $\phi_k, k = 1, 2, \cdots, n$, practically,

$$\phi_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}}, & x \in (x_{k-1}, x_k], \\ \frac{x - x_{k+1}}{x_k - x_{k+1}}, & x \in (x_{k-1}, x_k), \\ 0 & \textit{otherwise}. \end{cases}$$



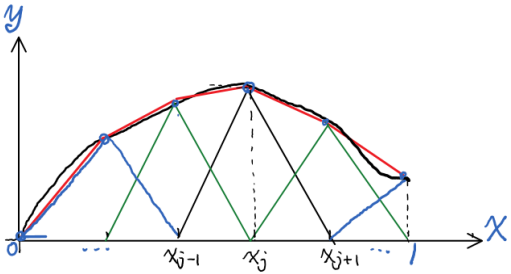
Galerkin Method

Let us show it by solving a two-point boundary value problem

$$-\frac{d^2}{dx^2}u(x) = f(x), x \in (0,1); \quad \text{with} \quad u(0) = u(1) = 0. \tag{6}$$

- nonuniform grid $0 = x_0 < x_1 < \cdots < x_{n+1} = 1$ with $h_i = (x_i - x_{i-1})$
- basis functions $\phi_k, k = 1, 2, \cdots, n$, practically,

$$\phi_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}}, & x \in (x_{k-1}, x_k], \\ \frac{x - x_{k+1}}{x_k - x_{k+1}}, & x \in (x_{k-1}, x_k), \\ 0 & \textit{otherwise}. \end{cases}$$



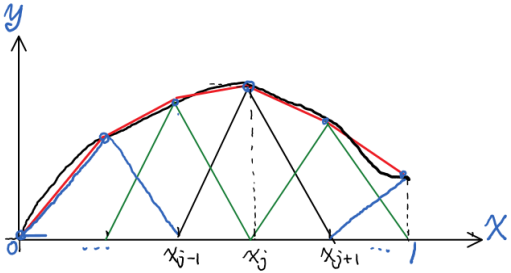
Galerkin Method

Let us show it by solving a two-point boundary value problem

$$-\frac{d^2}{dx^2}u(x) = f(x), x \in (0,1); \quad \text{with} \quad u(0) = u(1) = 0. \tag{6}$$

- nonuniform grid $0 = x_0 < x_1 < \cdots < x_{n+1} = 1$ with $h_i = (x_i - x_{i-1})$
- basis functions $\phi_k, k = 1, 2, \cdots, n$, practically,

$$\phi_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}}, & x \in (x_{k-1}, x_k], \\ \frac{x - x_{k+1}}{x_k - x_{k+1}}, & x \in (x_{k-1}, x_k), \\ 0 & \textit{otherwise}. \end{cases}$$



Galerkin Method - conti.

Any function u_h belongs to space V_h can be expressed with combination

$$u_h = \sum_{j=1}^{n} u_j \phi_j(x). \tag{7}$$

The Galerkin method for (6) is: To find $u_h \in V_h$, such that

$$\left(\frac{d}{dx}u_h, \frac{d}{dx}\phi_i\right) = (f, \phi_i), \forall i = 1, 2, \cdots, n.$$

where $(\cdot, \cdot)$ mean the inner-product defined on V_k. Finally

$$\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \vdots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{8}$$

$K_{ij} = \left(\frac{d}{dx}\phi_i, \frac{d}{dx}\phi_j\right) := \int_0^1 \frac{d}{dx}\phi_i(x)\frac{d}{dx}\phi_j(x)dx, \quad f_i = (f, \phi_i) := \int_0^1 f(x)\phi_i(x)dx.$

Galerkin Method - conti.

Any function u_h belongs to space V_h can be expressed with combination

$$u_h = \sum_{j=1}^{n} u_j \phi_j(x). \tag{7}$$

The Galerkin method for (6) is: To find $u_h \in V_h$, such that

$$\left(\frac{d}{dx}u_h, \frac{d}{dx}\phi_i\right) = (f, \phi_i), \forall i = 1, 2, \cdots, n.$$

where $(\cdot, \cdot)$ mean the inner-product defined on V_k. Finally

$$\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \vdots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{8}$$

$$K_{ij} = \left(\frac{d}{dx}\phi_i, \frac{d}{dx}\phi_j\right) := \int_0^1 \frac{d}{dx}\phi_i(x)\frac{d}{dx}\phi_j(x)dx, \quad f_i = (f, \phi_i) := \int_0^1 f(x)\phi_i(x)dx.$$

Galerkin Method - conti.

Any function u_h belongs to space V_h can be expressed with combination

$$u_h = \sum_{j=1}^{n} u_j \phi_j(x). \tag{7}$$

The Galerkin method for (6) is: To find $u_h \in V_h$, such that

$$\left(\frac{d}{dx}u_h, \frac{d}{dx}\phi_i\right) = \left(f, \phi_i\right), \forall i = 1, 2, \cdots, n.$$

where $(\cdot, \cdot)$ mean the inner-product defined on V_k. Finally

$$\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \vdots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{8}$$

$K_{ij} = \left(\frac{d}{dx}\phi_i, \frac{d}{dx}\phi_j\right) := \int_0^1 \frac{d}{dx}\phi_i(x)\frac{d}{dx}\phi_j(x)dx, \quad f_i = \left(f, \phi_i\right) := \int_0^1 f(x)\phi_i(x)dx.$

MATLAB code - Galerkin Method for 1d BVPs

```
%% -u''(x) = f(x), on (0,1); with u(a)=ua, u(b)=ub.
a=0; b=1; u = @(x) sin(pi*x); f = @(x) pi*pi*sin(pi*x); ua=u(a); ub=u(b);% EXE 1
n_elem = 16; n = n_elem; x = a + (0:n)*(b - a)/n;                    % the mesh grid
xi = [-0.577350269; 0.577350269]; wi = [0.5; 0.5];                   % the quadurature
K = zeros(n+1, n+1); rhs = zeros(n+1, 1); K_temp =[2, -1; -1, 2];    % allocate memory
for e = 1:n
    hi = x(e+1)-x(e); K(e:e+1,e:e+1) = K(e:e+1,e:e+1) + K_temp/hi; % assembling
    qx = x(e)*0.5*(1-xi) + x(e+1)*0.5*(1+xi);                        % xi's real coord
    rhs(e) = rhs(e) + sum(2.0*hi*wi.*f(qx).*(x(e+1)-qx)/hi);         % force node 1
    rhs(e+1)= rhs(e+1) + sum(2.0*hi*wi.*f(qx).*(qx - x(e))/hi);      % force node 2
end
K(1,1) = 1.0; K(1,2) = 0; rhs(1) = ua;                               % apply u(a) = ua
K(n+1, n) = 0; K(n+1, n+1) = 1.0; rhs(n+1) = ub;                     % apply u(b) = ub
u_h = K\rhs;                                                         % solving Ax = b
nn = 100; xx = a+(0:nn)/nn*(b-a); plot(xx,u(xx),'k-',x,u_h,'ro'); grid on;
```

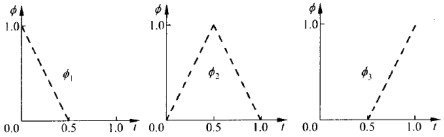
例 10.9　伽辽金方法　用例 10.6、例 10.7 和例 10.8 中的满足边界条件

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

的两点边值问题

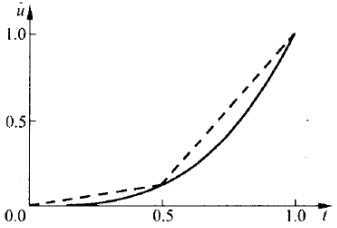
$$y''=6t,\qquad 0<t<1,$$

来说明伽辽金方法.



$$u(t)\approx v(t,x)=0.125\phi_2(t)+\phi_3(t),$$

画出了近似解及精确解的图形. 注意到 $v(0.5,x)=0.125$, 说明对这个近似解是精确的.



Remark on the Sparsity

Noticing that the basis functions $\phi_k(x), \forall k = 1, \cdots, n$ are all locally supported by their piecewise definition, it is obvious that $a_{ij} = 0$ if $|i - j| \geq 2$. So that the pattern of coefficient matrix K of the finite element system (8) will be

$$\begin{bmatrix} * & * & 0 & 0 & \cdots & 0 \\ * & * & * & 0 & \cdots & 0 \\ 0 & * & * & * & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & * & * \end{bmatrix} \tag{9}$$

- Sparsity will improve performance dramatically, as well as in Lecture 1
- It helps a lot in multi-dimensional cases

Remark on the Sparsity

Noticing that the basis functions $\phi_k(x), \forall k = 1, \cdots, n$ are all locally supported by their piecewise definition, it is obvious that $a_{ij} = 0$ if $|i - j| \geq 2$. So that the pattern of coefficient matrix K of the finite element system (8) will be

$$\begin{bmatrix} * & * & 0 & 0 & \cdots & 0 \\ * & * & * & 0 & \cdots & 0 \\ 0 & * & * & * & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & * & * \end{bmatrix} \tag{9}$$

- Sparsity will improve performance dramatically, as well as in Lecture 1
- It helps a lot in multi-dimensional cases

Shooting Method

Fundamental idea: to fully utilizing the efficiency of explicit scheme.

Considering

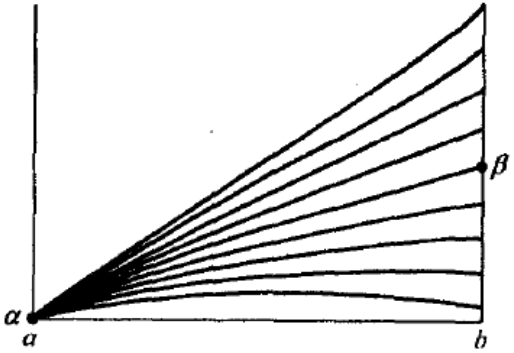
$$y' = f(x, y), a < t < b$$

s.t.

$$g(y(a), y(b)) = 0,$$

which is equal to

$$h(x) := g(x, y(b; x)) = 0.$$



Example (shooting)

Using the method of shooting so solve the following boundary problem:

$$u'' = -3u + 2\cos t, u(0) = 1, u(\pi) = -1,$$

Assume that $u_1 = u', u_2 = u_1', u_2(0) = \beta$, and the first guess is $\beta = 0.1$

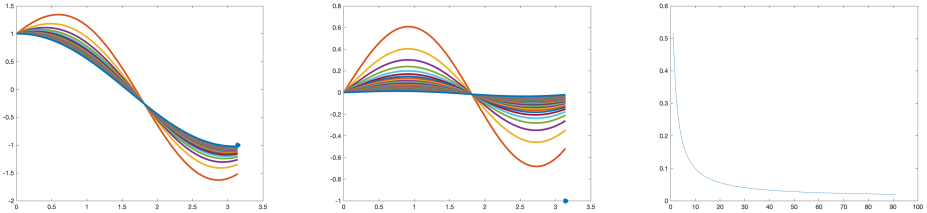


Figure: (Left)The solution, (Middle) Errors, (Right) and the objective function

A example in textbook

例 10.6　打靶法　我们用一维二阶常微分方程

$$u''=6t,\qquad 0<t<1,$$

说明两点边值问题的打靶法，边界条件为 $u(0)=0, u(1)=1$.

对每个 $u'(0)$ 的假定值，用经典的四阶龙格-库塔方法对常微分方程积分以确定数值解在 $t=1$ 处与给定值的差距. 但首先，将二阶常微分方程转化为一阶常微分方程组

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix}=\begin{bmatrix} y_2 \\ 6t \end{bmatrix},$$

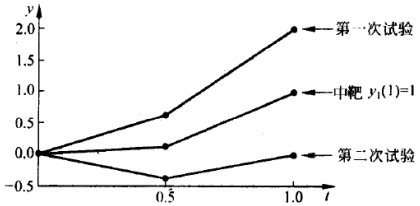


图 10.2　例 10.6 的两点边值问题的打靶法

打靶法思路简单，并且使用现有的初值问题及非线性方程的软件容易实现，但它也有严重的缺陷，最重要的一点是打靶法将继承相关初值问题的稳定（或不稳定）性

Conclusion & Homework 4

10. Quadrature

11. ODE-based Initial Value Problem

12. ODE-based Boundary Value Problem

Homework for week 4:

1. "Exercises" of Chapter **Quadrature Rules**:
   8.2、8.5、8.9、8.12、8.14
2. "Exercises" of Chapter **Ordinary Differential Equaitons**:
   9.5、9.7、9.11、10.2、10.3

Conclusion & Homework 4

10. Quadrature

11. ODE-based Initial Value Problem

12. ODE-based Boundary Value Problem

Homework for week 4:

1. "Exercises" of Chapter **Quadrature Rules**:
   8.2、8.5、8.9、8.12、8.14
2. "Exercises" of Chapter **Ordinary Differential Equaitons**:
   9.5、9.7、9.11、10.2、10.3