第一章：数值分析与科学计算引论

1.1 数值计算的误差

1.1.1 误差来源与分类

观测误差：由观测产生的误差。

截断误差：由数值方法求得的近似解与精确解之间的误差。

1.1.2 误差与有效数字

绝对误差：

$$e^* = x^* - x \tag{1}$$

相对误差：

$$e_r^* = \frac{e^*}{x} = \frac{x^* - x}{x} \approx \frac{e^*}{x^*} = \frac{x^* - x}{x^*} \tag{2}$$

绝对误差限：绝对误差绝对值的上界。

$$\varepsilon^* = |e^*| = |x^* - x| \tag{3}$$

相对误差限：相对误差绝对值的上界。

$$\varepsilon_r^* = \frac{\varepsilon^*}{|x^*|} = \frac{|x^* - x|}{|x^*|} \tag{4}$$

n位有效数字的标准形式：

$$x^* = \pm 10^m (a_1 + a_2 10^{-1} + \cdots + a_n 10^{-(n-1)}) \tag{5}$$

绝对误差限的计算：

$$|x^* - x| \leq \frac{1}{2} \times 10^{m-n+1} \tag{6}$$

相对误差的计算：

- 如果x^*具有n位有效数字，那么

$$\frac{|x^* - x|}{|x^*|} \leq \frac{1}{2a_1} \times 10^{-(n-1)} \tag{7}$$

- 如果

$$\frac{|x^* - x|}{|x^*|} \leq \frac{1}{2(a_1 + 1)} \times 10^{-(n-1)} \tag{8}$$

  那么至少x^*具有n位有效数字。

1.1.3 数值运算的误差估计

数值运算的误差估计：对于两个近似值x^*和y^*，误差限分别为$\varepsilon(x^*)$和$\varepsilon(y^*)$，那么

$$\varepsilon(x^* \pm y^*) \le \varepsilon(x^*) + \varepsilon(y^*) \tag{9}$$
$$\varepsilon(x^* y^*) \le |x^*|\varepsilon(y^*) + |y^*|\varepsilon(x^*) \tag{10}$$
$$\varepsilon(x^*/y^*) \le \frac{|x^*|\varepsilon(y^*) + |y^*|\varepsilon(x^*)}{|y^*|^2} \tag{11}$$
$$\varepsilon(f(x^*)) \le |f'(x^*)|\varepsilon(x^*) \tag{12}$$
$$\varepsilon(f(x^*, y^*)) \le |f_1'(x^*, y^*)|\varepsilon(x^*) + |f_2'(x^*, y^*)|\varepsilon(y^*) \tag{13}$$

1.2 误差定性分析与避免误差危害

稳定性：对于一个数值方法，如果输入数据有误差，且在计算过程中舍入误差不显著增长，那么称此算法为稳定的，否则称为不稳定的。

第二章：插值法

2.1 Lagrange插值

线性插值函数：

$$L_1(x)=\frac{x-x_1}{x_0-x_1}f(x_0)+\frac{x-x_0}{x_1-x_0}f(x_1) \tag{14}$$

抛物线插值函数：

$$L_2(x)=\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}f(x_1)+\frac{(x-x_3)(x-x_1)}{(x_2-x_3)(x_2-x_1)}f(x_2)+\frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}f(x_3) \tag{15}$$

Lagrange插值基函数：

$$l_i(x)=\prod_{j\neq i}\frac{x-x_j}{x_i-x_j} \tag{16}$$

Lagrange插值公式：

$$L_n(x)=\sum_{i=0}^{n}\left(\prod_{j\neq i}\frac{x-x_j}{x_i-x_j}\right)f(x_i)=\sum_{k=0}^{n}\frac{f(x_k)}{\omega'_{n+1}(x_k)}\frac{\omega_{n+1}(x)}{(x-x_k)} \tag{17}$$

$$R_n(x)=f(x)-L_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \tag{18}$$

其中

$$\omega_{n+1}(x)=\prod_{k=0}^{n}(x-x_k) \tag{19}$$

```
function fun = polynomialInterpolationFormula(x0, y0)

    % 名称:           多项式插值公式
    % 输入:
    %       x0:        插值点横坐标
    %       y0:        插值点纵坐标
    % 输出:           多项式插值公式

    %% 函数

    N = length(x0);

    % 初始化系数矩阵
    A = ones(N, N);
    for n = 2: N
        A(n, :) = x0 .^ (n-1);
    end
    A = A';

    % 求解系数
    coefficient = A \ y0';

    % 输出多项式插值函数
    syms x
    fun = coefficient(1);
    for n = 2: N
```

```
        fun = fun + coefficient(n) .* x .^ (n - 1);
    end
    fun = matlabFunction(fun);

end

```

2.2 Newton插值

差商：定义$f[x_0]=f(x_0)$为f在x_0处的0阶差商，递归的，定义如下为f在$x_0,\cdots,x_n$处的n阶均差。

$$f[x_0,\cdots,x_n]=\frac{f[x_1,\cdots,x_n]-f[x_0,\cdots,x_{n-1}]}{x_n-x_0} \tag{20}$$

- $f[x_0,\cdots,x_n]$中任意对换x_i和x_j的位置，差商不变。
- $$f[x_0,\cdots,x_n]=\sum_{k=0}^{n}\frac{f(x_k)}{\omega'_{n+1}(x_k)}=\frac{f^{(n)}(\xi)}{n!} \tag{21}$$
- $$f[x_0,\cdots,x_0]=\lim_{x_k\to x_0}f[x_0,\cdots,x_n]=\frac{f^{(n)}(x_0)}{n!} \tag{22}$$

均差表：

	0阶差商	1阶差商	2阶差商	3阶差商
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0,x_1]$		
x_2	$f[x_2]$	$f[x_1,x_2]$	$f[x_0,x_1,x_2]$	
x_3	$f[x_3]$	$f[x_2,x_3]$	$f[x_1,x_2,x_3]$	$f[x_0,x_1,x_2,x_3]$

Newton插值公式：

$$P_n(x)=f(x_0)+\sum_{i=1}^{n}f[x_0,\cdots,x_i]\prod_{j=0}^{i-1}(x-x_j) \tag{23}$$

$$R_n(x)=f(x)-P_n(x)=f[x,x_0,\cdots,x_n]\omega_{n+1}(x) \tag{24}$$

```
(* 定义函数 f *)
f[x_] := x^3

(* 定义差商函数 F *)
F[args__] := Total[f[#1] / Times @@ (#1 - Delete[{args}, Position[{args}, #1]]) & /@ {args}]

(* 调用差商函数 F *)
F[2, 3, 5]
```

```
function result = dividedDifference(fun, points)

    % 名称：差商
    % 输入：
    %       fun:    匿名函数
    %       points：需要求解差商的点
    % 输出：
    %       result：差商值

```

```
    %% 函数

    % 初始化结果
    result = 0;

    % 外层循环
    for i = 1: length(points)
        % 初始化积
        product = 1;
        % 内层循环
        for j = 1: length(points)
            if j ~= i
                product = product * (points(i) - points(j));
            end
        end
        result = result + fun(points(i)) / product;
    end

end

```

```
function NewtonInterpolationFormula(fun, a, b, points)

    % 名称:           Newton插值公式
    % 输入:
    %       fun:        匿名函数
    %       a:          插值左端点
    %       b:          插值右端点
    %       points:     插值节点
    % 输出:           插值图像

    %% 函数

    % 横坐标
    x = linspace(a, b, 1000);

    % 初始化纵坐标
    y = fun(a);
    % 求和
    for i = 1: length(points) - 1
        % 求解差商
        dividedDif = dividedDifference(fun, points(1: i + 1));
        % 初始化积
        prod = 1;
        % 求积
        for j = 0: i-1
            prod = prod .* (x - points(j + 1));
        end
        y = y + dividedDif .* prod;
    end

    % 绘图
    figure
    plot(x, y, x, fun(x))

end

```

2.3 Hermite插值

Hermite插值公式：如果要求插值函数H具有m阶导数，那么

$$H_{mn+m+n}^{(j)}(x_i)=f^{(j)}(x_i),\qquad 0\le i\le n,0\le j\le m \tag{25}$$

Taylor插值公式：

$$\begin{matrix}P^{(k)}(x_0)=f^{(k)}(x_0)\\ 0\le k\le n\end{matrix}\implies\begin{cases}P_n(x)=\sum_{k=0}^n\frac{f^{(k)}(x_0)}{k!}(x-x_0)^n\\ R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}\end{cases}$$

三点三次Hermite插值：

$$H(x_0)=f(x_0),\qquad H(x_1)=f(x_1),\qquad H(x_2)=f(x_2),\qquad H'(x_1)=f'(x_1) \tag{26}$$

$$\Downarrow \tag{27}$$

$$H_3(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1) \tag{28}$$

$$+\frac{f'(x_1)-f[x_0,x_1]-(x_1-x_0)f[x_0,x_1,x_2]}{(x_1-x_0)(x_1-x_2)}(x-x_0)(x-x_1)(x-x_2) \tag{29}$$

$$R_3(x)=\frac{f^{(4)}(\xi)}{4!}(x-x_0)(x-x_1)^2(x-x_2) \tag{30}$$

两点三次Hermite插值公式：

$$H(x_0)=f(x_0),\qquad H(x_1)=f(x_1),\qquad H'(x_0)=f'(x_0),\qquad H'(x_1)=f'(x_1) \tag{31}$$

$$\Downarrow \tag{32}$$

$$H_3(x)=f(x_0)\left(1+2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_0}{x_1-x_0}\right)^2+f(x_1)\left(1+2\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2 \tag{33}$$

$$+f'(x_0)(x-x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2+f'(x_1)(x-x_1)\left(\frac{x-x_0}{x_1-x_0}\right)^2 \tag{34}$$

$$R_3(x)=\frac{f^{(4)}(\xi)}{4!}(x-x_0)^2(x-x_1)^2 \tag{35}$$

2.4 分段低次插值

分段线性插值：

$$I_k(x)=\frac{x-x_{k+1}}{x_k-x_{k+1}}f(x_k)+\frac{x-x_k}{x_{k+1}-x_k}f(x_{k+1}) \tag{36}$$

$$|R_1(x)|=\left|\frac{f''(\xi)}{2!}(x-x_k)(x-x_{k+1})\right|\le\frac{1}{8}M_2h_k^2 \tag{37}$$

其中$h_k=x_{k+1}-x_k$，$M_2=\max|f''(x)|$。

分段Hermite插值：

$$H_3(x)=f(x_k)\left(1+2\frac{x-x_k}{x_{k+1}-x_k}\right)\left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2+f(x_{k+1})\left(1+2\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2 \tag{38}$$

$$+f'(x_k)(x-x_k)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2+f'(x_{k+1})(x-x_{k+1})\left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2 \tag{39}$$

$$|R_3(x)|=\left|\frac{f^{(4)}(\xi)}{4!}(x-x_k)^2(x-x_{k+1})^2\right|\le\frac{1}{384}M_4h_k^4 \tag{40}$$

其中$h_k=x_{k+1}-x_k$，$M_4=\max|f^{(4)}(x)|$。

2.5 三次样条插值

三次样条插值：插值函数$S(x)$在$[x_k,x_{k+1}]$上为三次函数，且$S(x_k)=f(x_k)$，同时S',S''连续。

```matlab
%% 自然边界
cubicSplineInterpolation = spline(x, [0, y, 0]);

%% 周期边界条件
% 为了满足周期边界条件，将第一个点和最后一个点连接起来
x0 = [x0, x0(1) + 2*pi];
y0 = [y0, y0(1)];
% 进行三次样条插值，使用周期边界条件
cubicSplineInterpolation = csape(x0, y0, 'periodic');
```

第三章：函数逼近

3.1 基本概念

范数：

$$
\begin{aligned}
\|f\|_1 &= \int_a^b |f| && (41)\\
\|f\|_2 &= \left(\int_a^b |f|^2\right)^{1/2} && (42)\\
\|f\|_\infty &= \sup_{[a,b]} |f| && (43)
\end{aligned}
$$

内积：

$$
(f,g) = \int_a^b fg \tag{44}
$$

Gram矩阵：$\varphi_1,\cdots,\varphi_n$线性无关，当且仅当$\det G \neq 0$，其中

$$
G = \begin{pmatrix} (\varphi_1,\varphi_1) & \cdots & (\varphi_1,\varphi_n) \\ \vdots & \ddots & \vdots \\ (\varphi_n,\varphi_1) & \cdots & (\varphi_n,\varphi_n) \end{pmatrix} \tag{45}
$$

权函数：称$[a,b]$上的非负函数ρ为权函数，如果满足如下性质。

- $\left|\int_a^b x^k \rho(x)\mathrm{d}x\right| < \infty, \qquad \forall k \in \mathbb{N}$
- $f \in C[a,b], \int_a^b |f|\rho = 0 \implies f = 0$

3.2 正交多项式

3.2.1 正交多项式

正交：称函数f与g关于权ρ正交，如果

$$
(f,g) = \int_a^b \rho f g = 0 \tag{46}
$$

正交函数系：称函数系$\varphi_0,\varphi_1,\cdots$为关于权$\rho$的正交函数系，如果

$$
(\varphi_i,\varphi_j) = \int_a^b \rho\varphi_i\varphi_j = \begin{cases} A_i, & i = j \\ 0, & i \neq j \end{cases} \tag{47}
$$

标准正交函数系：称函数系$\varphi_0,\varphi_1,\cdots$为关于权$\rho$的标准正交函数系，如果

$$
(\varphi_i,\varphi_j) = \int_a^b \rho\varphi_i\varphi_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{48}
$$

正交多项式：给定权函数ρ，构造幂函数系$1,x,x^2,\cdots$的正交多项式序列

$$
\begin{aligned}
\varphi_0 &= 1 && (49)\\
\varphi_n &= x^n - \sum_{k=0}^{n-1} \frac{(x^n,\varphi_k)}{(\varphi_k,\varphi_k)}\varphi_k && (50)
\end{aligned}
$$

或

$$\varphi_0 = 1 \tag{51}$$

$$\varphi_1 = \left(x - \frac{(x\varphi_0, \varphi_0)}{(\varphi_0, \varphi_0)}\right)\varphi_0 \tag{52}$$

$$\varphi_n = \left(x - \frac{(x\varphi_{n-1}, \varphi_{n-1})}{(\varphi_{n-1}, \varphi_{n-1})}\right)\varphi_{n-1} - \frac{(\varphi_{n-1}, \varphi_{n-1})}{(\varphi_{n-2}, \varphi_{n-2})}\varphi_{n-2} \tag{53}$$

- $[a, b]$上的n次正交多项式φ_n在(a, b)上存在n个互异零点。

```
a = -1;(*积分下限*)
b = 1;(*积分上限*)
rho[x_] := 1 + x^2 (*权函数*)

(*递推构造正交多项式*)
phi[x_, 0] := 1
phi[x_, n_] :=
 x^n - Sum[
   Integrate[rho[x] x^n phi[x, k], {x, a, b}]/
     Integrate[rho[x] phi[x, k] phi[x, k], {x, a, b}] phi[x, k], {k,
    0, n - 1}]

(*递推构造正交多项式*)
phi[x_, 0] := 1
phi[x_, 1] := (x - Integrate[x rho[x] phi[x, 0] phi[x, 0], {x, a, b}]/
    Integrate[rho[x] phi[x, 0] phi[x, 0], {x, a, b}]) psi[x, 0]
phi[x_, n_] := (x -
     Integrate[x rho[x] phi[x, n - 1] phi[x, n - 1], {x, a, b}]/
     Integrate[rho[x] phi[x, n - 1] phi[x, n - 1], {x, a, b}]) psi[x,
    n - 1] -
  Integrate[ rho[x] phi[x, n - 1] phi[x, n - 1], {x, a, b}]/
   Integrate[rho[x] phi[x, n - 2] phi[x, n - 2], {x, a, b}] phi[x,
    n - 2]
```

3.2.2 常见正交多项式

Legendre多项式：区间$[-1, 1]$上关于权$\rho = 1$的正交多项式为Legendre多项式

$$L_n = \frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n = \sum_{k=0}^{[n/2]}\frac{(-1)^k(2n-2k)!}{2^n k!(n-k)!(n-2k)!}x^{n-2k}, \qquad x \in [-1, 1], n \in \mathbb{N} \tag{54}$$

首一Legendre多项式

$$\tilde{L}_n = \frac{2^n(n!)^2}{(2n)!}L_n = \frac{n!}{(2n)!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n, \qquad n \in \mathbb{N} \tag{55}$$

- 正交性：

$$\int_{-1}^{1} L_m L_n = \begin{cases} 0, & m \neq n \\ \frac{2}{1+2n}, & m = n \end{cases} \tag{56}$$

- 奇偶性：

$$L_n(-x) = (-1)^n L_n(x) \tag{57}$$

- 递推关系：

$$(n+1)L_{n+1} = (2n+1)xL_n - nL_{n-1} \tag{58}$$

- 对于任意n次首一多项式$\tilde{P}_n$，成立

$$\|\tilde{L}_n\|_2 \le \|\tilde{P}_n\|_2 \iff \int_{-1}^{1} \tilde{L}_n^2 \le \int_{-1}^{1} \tilde{P}_n^2 \tag{59}$$

- L_n在$[-1,1]$内存在n个不同实数零点。

Chebyshev多项式：区间$[-1,1]$上关于权$\rho = 1/\sqrt{1-x^2}$的正交多项式为Chebyshev多项式

$$C_n = \cos(n\arccos x) = \frac{n}{2}\sum_{k=0}^{[n/2]}(-1)^k\frac{(n-k-1)!}{k!(n-2k)!}(2x)^{n-2k}, \qquad x\in[-1,1], n\in\mathbb{N} \tag{60}$$

令$x=\cos\theta$

$$C_n = \cos n\theta, \qquad \theta\in[0,\pi] \tag{61}$$

首一Chebyshev多项式

$$\tilde{C}_n = \frac{1}{2^{n-1}}C_n = \frac{n}{2^n}\sum_{k=0}^{[n/2]}(-1)^k\frac{(n-k-1)!}{k!(n-2k)!}(2x)^{n-2k}, \qquad x\in[-1,1], n\in\mathbb{N} \tag{62}$$

- 递推关系：

$$C_{n+1} = 2xC_n - C_{n-1} \tag{63}$$

- 正交性：

$$\int_{-1}^{1}\frac{C_mC_n}{\sqrt{1-x^2}} = \begin{cases}0, & m\ne n\\ \frac{\pi}{2}, & m=n\ne 0\\ \pi, & m=n=0\end{cases} \tag{64}$$

- C_{2n}仅含x的偶次幂，C_{2n+1}仅含x的奇次幂。
- Chebyshev零点：C_n在$[-1,1]$内存在n个零点

$$x_k = \cos\frac{2k-1}{2n}\pi, \qquad k=1,\cdots,n \tag{65}$$

- Chebyshev极值点：C_n在$[-1,1]$内存在$n+1$个零点

$$x'_k = \cos\frac{k}{n}\pi, \qquad k=0,\cdots,n \tag{66}$$

- 对于任意n次首一多项式$\tilde{P}_n$，成立

$$\|\tilde{C}_n\|_\infty \le \|\tilde{P}_n\|_\infty \iff \frac{1}{2^{n-1}} = \max_{[-1,1]}|\tilde{C}_n| \le \max_{[-1,1]}|P_n| \tag{67}$$

第二类Chebyshev多项式：区间$[-1,1]$上关于权$\rho=\sqrt{1-x^2}$的正交多项式为第二类Chebyshev多项式

$$C_n = \frac{\sin((n+1)\arcsin x)}{\sqrt{1-x^2}} \tag{68}$$

令$x=\sin\theta$

$$C_n = \frac{\sin(n+1)\theta}{\cos\theta}, \qquad \theta\in[-\pi/2,\pi/2] \tag{69}$$

- 递推关系：

$$C_{n+1} = 2xC_n - C_{n-1} \tag{70}$$

- 正交性：

$$\int_{-1}^{1}\sqrt{1-x^2}C_mC_n = \begin{cases}0, & m\ne n\\ \frac{\pi}{2}, & m=n\end{cases} \tag{71}$$

Laguerre多项式：区间$[0,\infty)$上关于权$\rho=\mathrm{e}^{-x}$的正交多项式为Laguerre多项式

$$L_n=\mathrm{e}^x\frac{\mathrm{d}}{\mathrm{d}x^n}x^n\mathrm{e}^{-x} \tag{72}$$

- 递推关系：

$$L_{n+1}=(1+2n-x)L_n-n^2L_{n-1} \tag{73}$$

- 正交性：

$$\int_0^{\infty}\mathrm{e}^{-x}L_mL_n=\begin{cases}0, & m\neq n\\ (n!)^2, & m=n\end{cases} \tag{74}$$

Hermite多项式：区间$(-\infty,\infty)$上关于权$\rho=\mathrm{e}^{-x^2}$的正交多项式为Hermite多项式

$$H_n=(-1)^n\mathrm{e}^{x^2}\frac{\mathrm{d}}{\mathrm{d}x^n}\mathrm{e}^{-x^2} \tag{75}$$

- 递推关系：

$$H_{n+1}=2xH_n-2nH_{n-1} \tag{76}$$

- 正交性：

$$\int_{-\infty}^{\infty}\mathrm{e}^{-x^2}H_mH_n=\begin{cases}0, & m\neq n\\ 2^nn!\sqrt{\pi}, & m=n\end{cases} \tag{77}$$

```
(* Legendre多项式 *)
L[x_, n_] := Simplify[1/(2^n n!) D[(x^2 - 1)^n, {x, n}]]

(* Chebyshev多项式 *)
Ch[x_, n_] := Simplify[TrigToExp[Cos[n ArcCos[x]]]]

(* 第二类Chebyshev多项式 *)
Ch[x_, n_] := FullSimplify[TrigToExp[Sin[(n + 1) ArcSin[x]]/Sqrt[1 - x^2]]]

(* Laguerre多项式 *)
L[x_, n_] := Simplify[Exp[x]*D[x^n*Exp[-x], {x, n}]]

(* Hermite多项式 *)
H[x_, n_] := Simplify[(-1)^n Exp[x^2] D[( Exp[-x^2])^n, {x, n}]]
```

3.2.3 Chebyshev多项式零点插值

Chebyshev多项式零点插值：在$[-1,1]$上，用Chebyshev多项式C_{n+1}的$n+1$个零点$x_k=\cos\frac{2k+1}{2(n+1)}\pi$作为插值结点，其中$k=0,\cdots,n$，由Lagrange插值公式或Newton插值公式可得插值多项式P_n，插值余项为

$$R_n(x)=f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x),\qquad \xi\in(-1,1) \tag{78}$$

因此

$$|R_n|\leq\frac{M_{n+1}}{(n+1)!}\max_{[-1,1]}|\omega_{n+1}|=\frac{M_{n+1}}{(n+1)!}\max_{[-1,1]}|\tilde{C}_{n+1}|=\frac{M_{n+1}}{2^n(n+1)!} \tag{79}$$

一般的，在$[a,b]$上，作映射$[a,b]\to[-1,1]$为$x\mapsto\frac{2x-b-a}{b-a}$，逆映射$[-1,1]\to[a,b]$为$x\mapsto\frac{b-a}{2}x+\frac{b+a}{2}$，可得

$$|R_n| \le \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!}M_{n+1} \tag{80}$$

3.3 最佳平方逼近

最佳平方逼近：对于函数$f \in C[a,b]$，称函数S为函数f在函数族$\mathrm{span}\{\varphi_k\}_{k=0}^n$中关于权$\rho$的最佳平方逼近，如果成立

$$\|f-S\|_2^2 = \inf_{g\in\mathrm{span}\{\varphi_n\}} \|f-g\|_2^2 \iff \int_a^b \rho(f-S)^2 = \inf_{g\in\mathrm{span}\{\varphi_n\}} \int_a^b \rho(f-g)^2 \tag{81}$$

记$S = \sum_{k=0}^n a_k\varphi_k$，那么

$$\begin{pmatrix} (\varphi_0,\varphi_0) & \cdots & (\varphi_0,\varphi_n) \\ \vdots & \ddots & \vdots \\ (\varphi_n,\varphi_0) & \cdots & (\varphi_n,\varphi_n) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f,\varphi_0) \\ \vdots \\ (f,\varphi_n) \end{pmatrix} \tag{82}$$

平方偏差/均方误差为

$$\|\delta\|_2^2 = \|f-S\|_2^2 = \|f\|_2^2 - \sum_{k=0}^n a_k(f,\varphi_k) \tag{83}$$

幂多项式系的最佳平方逼近：函数$f \in C[0,1]$在函数族$\mathrm{span}\{x^k\}_{k=0}^n$中关于权$\rho=1$的$n$次最佳平方逼近多项式满足

$$\begin{pmatrix} \frac{1}{1} & \frac{1}{2} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f,x^0) \\ \vdots \\ (f,x^n) \end{pmatrix} \tag{84}$$

正交函数系的最佳平方逼近：当$\{\varphi_k\}_{k=0}^n$关于权ρ正交时，最佳平方逼近为

$$S = \sum_{k=0}^n \frac{(f,\varphi_k)}{(\varphi_k,\varphi_k)}\varphi_k \tag{85}$$

平方偏差/均方误差为

$$\|\delta\|_2^2 = \|f-S\|_2^2 = \|f\|_2^2 - \sum_{k=0}^n \frac{(f,\varphi_k)^2}{(\varphi_k,\varphi_k)} \tag{86}$$

正交多项式系的最佳平方逼近：当$\{\varphi_n\}_{n=0}^\infty$为关于权$\rho$的正交多项式时，$n$次最佳平方逼近

$$S_n = \sum_{k=0}^n \frac{(f,\varphi_k)}{(\varphi_k,\varphi_k)}\varphi_k \tag{87}$$

满足

$$\|\delta_n\|_2 = \|f-S_n\|_2 \to 0 \tag{88}$$

Legendre多项式系的最佳平方逼近：考虑$f \in C[-1,1]$，权$\rho=1$，那么关于权ρ的正交多项式系为Legendre多项式系$\{L_n\}_{n=0}^\infty$，那么n次最佳平方逼近为

$$S_n = \sum_{k=0}^n \frac{2k+1}{2}\left(\int_{-1}^1 fL_k\right)L_k \tag{89}$$

平方偏差/均方误差为

$$\|\delta_n\|_2^2 = \|f - S_n\|_2^2 = \int_{-1}^{1} f^2 - \sum_{k=0}^{n} \frac{2k+1}{2} \left(\int_{-1}^{1} f L_k \right)^2 \tag{90}$$

```matlab
function c = weightedSquaresApproximatePolynomialFit(fun, rho, n, a, b)

    % 名称:   加权平方逼近多项式拟合
    % 输入:
    %       fun: 拟合函数
    %       rho: 拟合权重
    %       n: 拟合多项式次数
    %       a: 拟合左边界
    %       b: 拟合右边界
    % 输出:
    %       c: 拟合多项式系数

    %% 函数

    % 计算系数矩阵
    A = zeros(n + 1, n + 1);
    B = zeros(n + 1, 1);
    for i = 1: n + 1
        B(i) = integral(@(x) rho(x) .* fun(x) .* x .^ (i - 1), a, b);
        for j = 1: n + 1
            A(i, j) = integral(@(x) rho(x) .* x .^ (i + j - 2), a, b);
        end
    end
    % 求解多项式系数
    c = A \ B;

end

```

3.4 最小二乘法

点内积：对于点$\{x_k\}_{k=1}^n$，定义函数f与g关于权ω的点内积为

$$(f, g) = \sum_{k=1}^{n} \omega(x_k) f(x_k) g(x_k) \tag{91}$$

最小二乘法：对于函数$f \in C[a,b]$，称函数S为函数f在函数族$\mathrm{span}\{\varphi_k\}_{k=0}^n$中关于权$\omega$和点$\{x_k\}_{k=1}^n$的最小二乘拟合，如果成立

$$\sum_{k=0}^{n} \omega(x_k) |S(x_k) - f(x_k)|^2 = \inf_{g \in \mathrm{span}\{\varphi_k\}} \sum_{k=0}^{n} \omega(x_k) |g(x_k) - f(x_k)|^2 \tag{92}$$

记$S = \sum_{k=0}^{n} a_k \varphi_k$，那么

$$\begin{pmatrix} (\varphi_0, \varphi_0) & \cdots & (\varphi_0, \varphi_n) \\ \vdots & \ddots & \vdots \\ (\varphi_n, \varphi_0) & \cdots & (\varphi_n, \varphi_n) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f, \varphi_0) \\ \vdots \\ (f, \varphi_n) \end{pmatrix} \tag{93}$$

平方偏差/均方误差为

$$\|\delta\|^2 = \|f - S\|^2 = \|f\|^2 - \sum_{k=0}^{n} a_k (f, \varphi_k) \tag{94}$$

正交函数系的最佳平方逼近：当$\{\varphi_k\}_{k=0}^n$关于权ω正交时，最小二乘拟合为

$$S=\sum_{k=0}^n\frac{(f,\varphi_k)}{(\varphi_k,\varphi_k)}\varphi_k \tag{95}$$

平方偏差/均方误差为

$$\|\delta\|_2^2=\|f-S\|_2^2=\|f\|_2^2-\sum_{k=0}^n\frac{(f,\varphi_k)^2}{(\varphi_k,\varphi_k)} \tag{96}$$

```
% 最小二乘拟合
p = polyfit(x0, y0, n); % n次多项式的系数向量
```

```
function c = weightedLeastSquaresFit(x, y, w, n)

    % 名称:   加权最小二乘拟合
    % 输入:
    %       x: 拟合点横坐标
    %       y: 拟合点纵坐标
    %       w: 拟合权重
    %       n: 拟合多项式次数
    % 输出:
    %       c: 拟合多项式系数

    %% 函数

    % 计算系数矩阵
    A = zeros(n + 1, n + 1);
    b = zeros(n + 1, 1);
    for i = 1: n + 1
        b(i) = sum(w .* y .* x .^ (i - 1));
        for j = 1: n + 1
            A(i, j) = sum(w .* x .^ (i + j - 2));
        end
    end
    % 求解多项式系数
    c = A \ b;

end

```

第四章：数值积分与数值微分

4.1 数值积分

类型	公式	代数精度	求积余项
左矩形公式	$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(a)$		
右矩形公式	$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(b)$		
中矩形公式	$\int_a^b f(x)\mathrm{d}x \approx (b-a)f\left(\frac{a+b}{2}\right)$	1	$R[f]=\frac{(b-a)^3}{24}f''(\xi)$
梯形公式	$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}(f(a)+f(b))$	1	$R[f]=-\frac{(b-a)^3}{12}f''(\xi)$

数值求积公式：函数f在区间$[a,b]$上关于求积节点$\{x_k\}_{k=0}^n$和权$\{A_k\}_{k=0}^n$的数值求积公式为

$$I=\int_a^b f(x)\mathrm{d}x \approx I_n=\sum_{k=0}^n A_k f(x_k) \tag{97}$$

代数精度：称求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^n A_k f(x_k)$具有$m$次代数精度，如果成立

$$\int_a^b x^k\mathrm{d}x=\sum_{i=0}^n A_i x_i^k, \qquad k=0,\cdots,m \tag{98}$$

$$\int_a^b x^{m+1}\mathrm{d}x \neq \sum_{i=0}^n A_i x_i^{m+1} \tag{99}$$

机械求积公式：如果求积节点$\{x_k\}_{k=0}^n$和权$\{A_k\}_{k=0}^n$与被积函数$f(x)$无关，那么称函数f在区间$[a,b]$上关于求积节点$\{x_k\}_{k=0}^n$和权$\{A_k\}_{k=0}^n$的数值求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^n A_k f(x_k)$为机械求积公式。

插值型数值求积公式：由Lagrange插值公式

$$f(x)=\sum_{i=0}^n f(x_i)l_i(x)+R_n(x) \tag{100}$$

其中

$$l_i(x)=\frac{\prod\limits_{j\neq i}(x-x_j)}{\prod\limits_{j\neq i}(x_i-x_j)} \tag{101}$$

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \tag{102}$$

那么插值型数值求积公式为

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^n \int_a^b l_i(x)\mathrm{d}x \cdot f(x_i) \tag{103}$$

> 求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^n A_k f(x_k)$为插值型数值求积公式$\iff$求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^n A_k f(x_k)$具有$n$次代数精度。

求积余项：如果求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k)$具有$m$次代数精度，那么求积余项为

$$R[f] = \int_a^b f(x)\mathrm{d}x - \sum_{k=0}^{n} A_k f(x_k) = K f^{(m+1)}(\xi), \qquad \xi \in (a,b) \tag{104}$$

$$K = \frac{1}{(m+1)!}\left(\frac{b^{m+2} - a^{m+2}}{m+2} - \sum_{k=0}^{n} A_k x_k^{m+1}\right) \tag{105}$$

收敛性：记$h = \max\limits_{1\le k\le n}\{x_k - x_{k-1}\}$，称求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k)$具有收敛性，如果

$$\lim_{\substack{n\to\infty \\ h\to 0}} \sum_{k=0}^{n} A_k f(x_k) = \int_a^b f(x)\mathrm{d}x \tag{106}$$

稳定性：称求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k)$具有稳定性，如果对于任意$\varepsilon > 0$，存在$\delta > 0$，成立

$$|\delta_k| \le \delta, \forall 0 \le k \le n \implies \left|\sum_{k=0}^{n} A_k \delta_k\right| < \varepsilon \tag{107}$$

> 如果求积公式$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k)$中权$\{A_k\}_{k=0}^{n}$同号，那么此求积公式具有稳定性。

4.2 Newton-Cotes公式

Cotes系数：

$$C_k^{(n)} = \frac{(-1)^{n-k}}{nk!(n-k)!}\int_0^n \prod_{\substack{0\le i\le n \\ i\ne k}} (x-i)\mathrm{d}x \tag{108}$$

```
CotesCoefficient[n_, k_] := (-1)^(n - k)/(n k! (n - k)!)*
  Integrate[Product[(x - i), {i, 0, n}]/(x - k), {x, 0, n}]
```

Newton-Cotes公式：等距节点$x_k = a + \frac{b-a}{n}k$的插值型求积公式为

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)\sum_{k=0}^{n} C_k^{(n)} f(x_k) \tag{109}$$

Newton-Cotes公式的代数精度：

- 当n为奇数时，n阶Newton-Cotes公式具有n次代数精度。
- 当n为偶数时，n阶Newton-Cotes公式具有$n+1$次代数精度。

Newton-Cotes公式的稳定性：当且仅当$n \le 8$时，n阶Newton-Cotes公式具有稳定性。

阶	名称	公式	求积余项	代数精度
1	梯形公式	$T = \frac{b-a}{2}(f(a)+f(b))$	$-\frac{(b-a)^3}{12}f''(\xi)$	1
2	Simpson公式	$S = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$	$-\frac{(b-a)^5}{2880}f^{(4)}(\xi)$	3
3	四点八分之三Simpson公式	$\frac{b-a}{8}\left(f(a)+3f\left(\frac{2a+b}{3}\right)+3f\left(\frac{a+2b}{3}\right)+f(b)\right)$		3
4	Cotes公式	$C = \frac{b-a}{90}\left(7f(a)+32f\left(\frac{3a+b}{4}\right)+12f\left(\frac{a+b}{2}\right) + 32f\left(\frac{a+3b}{4}\right)+7f(b)\right)$	$-\frac{2(b-a)}{945}\left(\frac{b-a}{4}\right)^6 f^{(6)}(\xi)$	5

一阶Newton-Cotes公式　梯形公式：

$$\int_a^b f(x)\mathrm{d}x \approx T = \frac{b-a}{2}(f(a)+f(b)) \tag{110}$$

$$R[f] = -\frac{(b-a)^3}{12}f''(\xi), \qquad \xi \in (a,b) \tag{111}$$

二阶Newton-Cotes公式　Simpson公式:

$$\int_a^b f(x)\mathrm{d}x \approx S = \frac{b-a}{6}\left(f(a)+4f\left(\tfrac{a+b}{2}\right)+f(b)\right) \tag{112}$$

$$R[f] = -\frac{(b-a)^5}{2880}f^{(4)}(\xi), \qquad \xi \in (a,b) \tag{113}$$

三阶Newton-Cotes公式　四点八分之三Simpson公式:

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{8}\left(f(a)+3f\left(\tfrac{2a+b}{3}\right)+3f\left(\tfrac{a+2b}{3}\right)+f(b)\right) \tag{114}$$

四阶Newton-Cotes公式　Cotes公式:

$$\int_a^b f(x)\mathrm{d}x \approx C = \frac{b-a}{90}\left(7f\left(a\right)+32f\left(\tfrac{3a+b}{4}\right)+12f\left(\tfrac{a+b}{2}\right)+32f\left(\tfrac{a+3b}{4}\right)+7f\left(b\right)\right) \tag{115}$$

$$R[f] = -\frac{2(b-a)}{945}\left(\frac{b-a}{4}\right)^6 f^{(6)}(\xi), \qquad \xi \in (a,b) \tag{116}$$

六阶Newton-Cotes公式　Romberg公式: R

```
function result = CotesCoefficient(n, k)

    % 名称：Cotes系数
    % 输入：
    %       n
    %       k
    % 输出：
    %     result：Cotes系数C_k^n

    %% 函数
    syms x;

    result = (- 1) ^ (n - k) / (n * factorial(k) * factorial(n - k));

    % 定义被积函数
    integrand = 1;
    for i = 0: n
        if i ~= k
            integrand = integrand * (x - i);
        end
    end

    % 计算积分
    result = result * int(integrand, 0, n);

end

```

```
function result = NewtonCotesFormula(fun, n, a, b)

    % 名称：Newton-Cotes公式
    % 输入：
```

```
    %       fun:    积分函数
    %       n:      积分节点数
    %       a:      积分左边界
    %       b:      积分右边界
    % 输出:
    %     result: Newton-Cotes公式积分值

    %% 函数

    result = 0;
    for k = 0: n
        result = result + CotesCoefficient(n, k) * f(a + (b - a) * k / n);
    end
    result = (b - a) * result;

end

```

4.3 复合求积公式

名称	公式	积分余项	代数精度
复合梯形公式	$T_n = \frac{b-a}{2n}\sum_{k=1}^{n}(f(x_{k-1})+f(x_k)) = \frac{b-a}{2n}\left(f(a)+2\sum_{k=1}^{n-1}f(x_k)+f(b)\right)$	$-\frac{(b-a)^3}{12n^2}f''(\xi)$	1
复合Simpson公式	$S_n = \frac{b-a}{6n}\sum_{k=1}^{n}\left(f(x_{k-1})+4f\left(\frac{x_{k-1}+x_k}{2}\right)+f(x_k)\right) = \frac{b-a}{6n}\left(f(a)+2\sum_{k=1}^{n-1}f(x_k)+4\sum_{k=1}^{n}f\left(\frac{x_{k-1}+x_k}{2}\right)+f(b)\right)$	$-\frac{(b-a)^5}{2880n^4}f^{(4)}(\xi)$	3

复合梯形公式：等距节点$x_k = a + \frac{b-a}{n}k$的复合梯形公式为

$$\int_a^b f(x)\mathrm{d}x \approx T_n = \frac{b-a}{2n}\sum_{k=1}^{n}(f(x_{k-1})+f(x_k)) = \frac{b-a}{2n}\left(f(a)+2\sum_{k=1}^{n-1}f(x_k)+f(b)\right) \tag{117}$$

复合梯形公式具有1次代数精度、收敛性、稳定性，积分余项为

$$R[f] = -\frac{(b-a)^3}{12n^2}f''(\xi), \qquad \xi \in (a,b) \tag{118}$$

```
T[n_] := (b - a)/(2 n) (f[a] + f[b] + 2 Sum[f[a + (b - a)/n k], {k, 1, n - 1}])
```

复合Simpson公式：等距节点$x_k = a + \frac{b-a}{n}k$的复合Simpson公式为

$$\int_a^b f(x)\mathrm{d}x \approx S_n = \frac{b-a}{6n}\sum_{k=1}^{n}\left(f(x_{k-1})+4f\left(\frac{x_{k-1}+x_k}{2}\right)+f(x_k)\right) \tag{119}$$

$$= \frac{b-a}{6n}\left(f(a)+2\sum_{k=1}^{n-1}f(x_k)+4\sum_{k=1}^{n}f\left(\frac{x_{k-1}+x_k}{2}\right)+f(b)\right) \tag{120}$$

复合Simpson公式具有3次代数精度、收敛性、稳定性，积分余项为

$$R[f] = -\frac{(b-a)^5}{2880n^4}f^{(4)}(\xi), \qquad \xi \in (a,b) \tag{121}$$

```
S[n_] := (b - a)/(6 n) (f[a] + f[b] + 2 Sum[f[a + (b - a)/n k], {k, 1, n - 1}] +
   4 Sum[f[((a + (b - a)/n (k - 1)) + (a + (b - a)/n k))/2], {k, 1, n}])
```

```
function result = compoundTrapezoidalFormula(fun, n, a, b)
```

```

    % 名称：复合梯形公式
    % 输入：
    %       fun:    积分函数
    %       n:      积分节点数
    %       a:      积分左边界
    %       b:      积分右边界
    % 输出：
    %      result：复合梯形公式积分值

    %% 函数

    result = fun(a) + fun(b);
    for k = 1: n-1
        result = result + 2 * fun(a + (b - a) * k / n);
    end
    result = (b - a) / (2 * n) * result;

end

```

```
function result = compoundSimpsonFormula(fun, n, a, b)

    % 名称：复合Simpson公式
    % 输入：
    %       fun:    积分函数
    %       n:      积分节点数
    %       a:      积分左边界
    %       b:      积分右边界
    % 输出：
    %      result：复合Simpson公式积分值

    %% 函数

    result = fun(a) + fun(b);
    for k = 1: n-1
        result = result + 2 * fun(a + (b - a) * k / n);
    end
    for k = 1: n
        result = result + 4 * fun(a + (b - a) * (k - 1 / 2) / n);
    end
    result = (b - a) / (6 * n) * result;

end

```

4.4 Romberg求积公式

梯形递推公式：等距节点$x_k = a + \frac{b-a}{n}k$的梯形递推公式为

$$T_n = \frac{b-a}{2n}\left(f(a) + 2\sum_{k=1}^{n-1} f(x_k) + f(b)\right) \tag{122}$$

$$T_{2n} = \frac{1}{2}T_n + \frac{b-a}{2n}\sum_{k=1}^{n} f\left(\frac{x_{k-1}+x_k}{2}\right) \tag{123}$$

特别的

$$T_{2^0}=\frac{b-a}{2}(f(a)+f(b)) \tag{124}$$

$$T_{2^{n+1}}=\frac{1}{2}T_{2^n}+\frac{b-a}{2^{n+1}}\sum_{k=1}^{2^n}f\left(a+(2k-1)\frac{b-a}{2^{n+1}}\right) \tag{125}$$

复合求积公式关系：

$$S_n=\frac{4T_{2n}-T_n}{4-1} \tag{126}$$

$$C_n=\frac{4^2S_{2n}-S_n}{4^2-1} \tag{127}$$

$$R_n=\frac{4^3C_{2n}-C_n}{4^3-1} \tag{128}$$

4.5 Gauss型求积公式

4.5.1 Gauss型求积公式

Gauss型求积公式：称加权求积公式$\int_a^b\rho(x)f(x)\mathrm{d}x\approx\sum_{k=0}^nA_kf(x_k)$为Gauss型求积公式，如果其具有$2n+1$次代数精度。

Gauss点：称Gauss型求积公式$\int_a^b\rho(x)f(x)\mathrm{d}x\approx\sum_{k=0}^nA_kf(x_k)$的求积节点$\{x_k\}_{k=0}^n$为Gauss点。

定理：加权求积公式$\int_a^b\rho(x)f(x)\mathrm{d}x\approx\sum_{k=0}^nA_kf(x_k)$的求积节点$\{x_k\}_{k=0}^n$为Gauss点$\iff$对于任意$\le n$次多项式$p(x)$，成立

$$\int_a^b\rho(x)p(x)\omega_{n+1}(x)\mathrm{d}x=0 \tag{129}$$

Gauss型求积公式：

$$\int_a^b\rho(x)f(x)\mathrm{d}x\approx\sum_{k=0}^nA_kf(x_k) \tag{130}$$

其中求积节点$\{x_k\}_{k=0}^n$为关于权ρ的$n+1$次正交多项式的零点，权$\{A_k\}_{k=0}^n$满足

$$\int_a^b\rho(x)x^m\mathrm{d}x=\sum_{k=0}^nA_kx_k^m,\qquad 0\le m\le 2n+1 \tag{131}$$

Gauss型求积公式具有$2n+1$次代数精度、收敛性、稳定性，积分余项为

$$R_n[f]=\frac{1}{(2n+2)!}f^{(2n+2)}(\xi)\int_a^b\rho(x)\omega_{n+1}^2(x)\mathrm{d}x,\qquad \xi\in(a,b) \tag{132}$$

4.5.2 常见Gauss型求积公式

Gauss-Legendre求积公式：区间$[-1,1]$上关于权$\rho=1$的Gauss型求积公式为Gauss-Legendre求积公式

$$\int_{-1}^1f(x)\mathrm{d}x\approx\sum_{k=0}^nA_kf(x_k) \tag{133}$$

其中求积节点$\{x_k\}_{k=0}^n$为$n+1$次Legendre多项式$L_{n+1}(x)$的零点，权$\{A_k\}_{k=0}^n$满足

$$\int_{-1}^{1} x^m \mathrm{d}x = \sum_{k=0}^{n} A_k x_k^m, \qquad 0 \le m \le 2n+1 \tag{134}$$

Gauss-Legendre求积公式具有$2n+1$次代数精度、收敛性、稳定性，积分余项为

$$R_n[f] = \frac{2^{2n+3}((n+1)!)^4}{(2n+3)((2n+2)!)^3} f^{(2n+2)}(\xi), \qquad \xi \in (-1,1) \tag{135}$$

一般的

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right)\mathrm{d}t = \frac{b-a}{2}\int_{-1}^{1} g(t)\mathrm{d}t \approx \frac{b-a}{2}\sum_{k=0}^{n} A_k g(x_k) \tag{136}$$

特别的

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx 2f(0) \tag{137}$$

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx f(-1/\sqrt{3}) + f(1/\sqrt{3}) \tag{138}$$

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{5}{9}f(-\sqrt{15}/5) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{15}/5) \tag{139}$$

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{1}{36}\left(18+\sqrt{30}\right) f\left(-\sqrt{\tfrac{1}{35}\left(15-2\sqrt{30}\right)}\right) \tag{140}$$

$$+\frac{1}{36}\left(18+\sqrt{30}\right) f\left(\sqrt{\tfrac{1}{35}\left(15-2\sqrt{30}\right)}\right) \tag{141}$$

$$+\frac{1}{36}\left(18-\sqrt{30}\right) f\left(-\sqrt{\tfrac{1}{35}\left(15+2\sqrt{30}\right)}\right) \tag{142}$$

$$+\frac{1}{36}\left(18-\sqrt{30}\right) f\left(\sqrt{\tfrac{1}{35}\left(15+2\sqrt{30}\right)}\right) \tag{143}$$

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{128}{225}f(0) + \frac{1}{900}\left(322+13\sqrt{70}\right) f\left(-\tfrac{1}{3}\sqrt{\tfrac{1}{7}\left(35-2\sqrt{70}\right)}\right) \tag{144}$$

$$+\frac{1}{900}\left(322+13\sqrt{70}\right) f\left(\tfrac{1}{3}\sqrt{\tfrac{1}{7}\left(35-2\sqrt{70}\right)}\right) \tag{145}$$

$$+\frac{1}{900}\left(322-13\sqrt{70}\right) f\left(-\tfrac{1}{3}\sqrt{\tfrac{1}{7}\left(35+2\sqrt{70}\right)}\right) \tag{146}$$

$$+\frac{1}{900}\left(322-13\sqrt{70}\right) f\left(\tfrac{1}{3}\sqrt{\tfrac{1}{7}\left(35+2\sqrt{70}\right)}\right) \tag{147}$$

Gauss-Chebyshev求积公式：区间$[-1,1]$上关于权$\rho = 1/\sqrt{1-x^2}$的Gauss型求积公式为Gauss-Chebyshev求积公式

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\mathrm{d}x \approx \frac{\pi}{n}\sum_{k=0}^{n} f(\cos\frac{2k-1}{2n}\pi) \tag{148}$$

Gauss-Chebyshev求积公式具有$2n-1$次代数精度、收敛性、稳定性，积分余项为

$$R_n[f] = \frac{2\pi}{2^{2n}(2n)!} f^{(2n)}(\xi), \qquad \xi \in (-1,1) \tag{149}$$

Gauss-Laguerre求积公式：区间$[0,\infty)$上关于权$\rho = \mathrm{e}^{-x}$的Gauss型求积公式为Gauss-Laguerre求积公式

$$\int_0^{\infty} \mathrm{e}^{-x} f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k) \tag{150}$$

其中求积节点$\{x_k\}_{k=0}^{n}$为$n+1$次Laguerre多项式$L_{n+1}(x)$的零点，权$A_k=\frac{((n+1)!)^2}{x_k(L'_{n+1}(x_k))^2}$。Gauss-Laguerre求积公式具有$2n+1$次代数精度、收敛性、稳定性，积分余项为

$$R_n[f]=\frac{((n+1)!)^2}{2(n+1)!}f^{(2n+2)}(\xi),\qquad \xi\in(0,\infty) \tag{151}$$

Gauss-Hermite求积公式：区间$(-\infty,\infty)$上关于权$\rho=\mathrm{e}^{-x^2}$的Gauss型求积公式为Gauss-Hermite求积公式

$$\int_{-\infty}^{\infty}\mathrm{e}^{-x^2}f(x)\mathrm{d}x\approx\sum_{k=0}^{n}A_kf(x_k) \tag{152}$$

其中求积节点$\{x_k\}_{k=0}^{n}$为$n+1$次Hermite多项式$H_{n+1}(x)$的零点，权$A_k=\frac{2^{n+2}(n+1)!\sqrt{\pi}}{(H'_{n+1}(x_k))^2}$。Gauss-Hermite求积公式具有$2n+1$次代数精度、收敛性、稳定性，积分余项为

$$R_n[f]=\frac{((n+1)!)^2}{2(n+1)!}f^{(2n+2)}(\xi),\qquad \xi\in(-\infty,\infty) \tag{153}$$

```
function result = GaussLegendreIntegralFormula(fun, n, a, b)

    % 名称：Gauss-Legendre求积公式
    % 输入：
    %       fun:    积分函数
    %       n:      积分节点数
    %       a:      积分左边界
    %       b:      积分右边界
    % 输出：
    %       result：积分值

    %% 函数

    % 求解Legendre多项式的零点
    syms x
    L = diff((x^2-1)^n, x, n) / (2^n * factorial(n)); % Legendre多项式
    root = solve(L);                                   % Legendre多项式的根

    % 求解权重
    A = zeros(2 * n, n);
    B = zeros(2 * n, 1);
    for k = 0: 2 * n - 1
        A(k + 1, :) = transpose(root .^ k);
        B(k + 1) = int(x .^ k, -1, 1);
    end
    w = A \ B;

    % 求解积分值
    f = @(x) fun((b - a) / 2 .* x + (b + a) / 2);
    result = (b - a) / 2 * sum(w .* f(root));

end

```

```
function result = GaussLaguerreIntegralFormula(fun, n)

    % 名称：Gauss-Laguerre求积公式
```

```
    % 输入：
    %       fun:     积分函数
    %       n:       积分节点数
    % 输出：
    %       result：积分值

    %% 函数
    syms x
    L = exp(x) * diff(x^n * exp(-x), x, n); % Laguerre多项式
    root = solve(L);                         % Laguerre多项式的根
    DL = matlabFunction(diff(L, x));
    result = 0;
    for k = 1: n
        result = result + (factorial(n))^2 / root(k) / (DL(root(k)))^2 *
fun(root(k));
    end

end

```

```
function result = CompoundGaussLegendreIntegralFormula(fun, n, k, a, b)

    % 名称：复合Gauss-Legendre求积公式
    % 输入：
    %       fun:     积分函数
    %       n:       积分区间数
    %       k:       区间积分节点数
    %       a:       积分左边界
    %       b:       积分右边界
    % 输出：
    %       result：积分值

    %% 函数
    result = 0;
    x = @(i) a + (b - a) / n * i;
    for i = 1: n
        result = result + GaussLegendreIntegralFormula(fun, k, x(i - 1), x(i));
    end

end

```

4.6 数值微分

数值微分：

$$f'(x) \approx \sum_{k=0}^{n} A_k f(x_k) \tag{154}$$

$$R_n(x) = f'(x) - \sum_{k=0}^{n} A_k f(x_k) \tag{155}$$

4.6.1 差商型数值微分公式

向前差商数值微分公式:

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \tag{156}$$

$$R_n(x) = -\frac{h}{2}f''(\xi), \quad \xi \in (x, x+h) \tag{157}$$

向后差商数值微分公式:

$$f'(x) \approx \frac{f(x)-f(x-h)}{h} \tag{158}$$

$$R_n(x) = \frac{h}{2}f''(\xi), \quad \xi \in (x-h, x) \tag{159}$$

中心差商数值微分公式:

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} \tag{160}$$

$$R_n(x) = -\frac{h^2}{6}f^{(3)}(\xi), \quad \xi \in (x-h, x+h) \tag{161}$$

4.6.2 插值型求导公式

插值求导公式：以n次Lagrange插值函数$L_n(x)$近似原函数$f(x)$，其中

$$L_n(x) = \sum_{k=0}^{n} \frac{f(x_k)}{\omega'_{n+1}(x_k)} \frac{\omega_{n+1}(x)}{(x-x_k)} \tag{162}$$

$$\omega_{n+1}(x) = \prod_{k=0}^{n}(x-x_k) \tag{163}$$

截断误差为

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega'_{n+1}(x) + \frac{\omega_{n+1}(x)}{(n+1)!}\frac{\mathrm{d}}{\mathrm{d}x}f^{(n+1)}(\xi) \tag{164}$$

$$R_n(x_k) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega'_{n+1}(x_k) \tag{165}$$

两点公式：对于$x_n = x_0 + nh$

$$f'(x_0) = \frac{-f(x_0)+f(x_1)}{h} - \frac{h}{2}f''(\xi) \tag{166}$$

$$f'(x_1) = \frac{-f(x_0)+f(x_1)}{h} + \frac{h}{2}f''(\xi) \tag{167}$$

三点公式：对于$x_n = x_0 + nh$

$$f'(x_0) = \frac{-3f(x_0)+4f(x_1)-f(x_2)}{2h} + \frac{h^2}{3}f^{(3)}(\xi) \tag{168}$$

$$f'(x_1) = \frac{-f(x_0)+f(x_2)}{2h} - \frac{h^2}{6}f^{(3)}(\xi) \tag{169}$$

$$f'(x_2) = \frac{f(x_0)-4f(x_1)+3f(x_2)}{2h} + \frac{h^2}{3}f^{(3)}(\xi) \tag{170}$$

三点二阶公式:

$$f''(x) = \frac{f(x-h)-2f(x)+f(x+h)}{h^2} - \frac{h^2}{12}f^{(4)}(\xi) \tag{171}$$

第五章：解线性方程的直接法

5.1 矩阵范数

向量范数：称$\|x\|$为向量$x \in \mathbb{C}^n$的范数，如果

- $\|x\| \geq 0$，当且仅当$x = 0$时等号成立。
- $\|\lambda x\| = |\lambda| \|x\|$
- $\|x + y\| \leq \|x\| + \|y\|$

向量范数的连续性：向量范数具有连续性。

向量范数的等价性：对于$\mathbb{C}^n$上的范数$\|\cdot\|_s$与$\|\cdot\|_t$，存在$C_1, C_2 > 0$，使得对于任意$x \in \mathbb{C}^n$，成立

$$C_1 \|x\|_s \leq \|x\|_t \leq C_2 \|x\|_s \tag{172}$$

向量的收敛性：向量依范数收敛 $\Longleftrightarrow$ 依坐标收敛。

矩阵范数：称$\|A\|$为矩阵$A \in \mathbb{C}^{n \times n}$的范数，如果

- $\|A\| \geq 0$，当且仅当$A = O$时等号成立。
- $\|\lambda A\| = |\lambda| \|A\|$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|AB\| \leq \|A\| \|B\|$

矩阵的算子范数：定义关于向量范数$\|\cdot\|$的矩阵范数为

$$\|A\| = \sup_{\|x\| \leq 1} \|Ax\| \tag{173}$$

常见矩阵范数：

$$\text{矩阵范数} \begin{cases} \text{算子范数} \begin{cases} \text{行范数} & \|A\|_\infty = \max\limits_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \\ \text{列范数} & \|A\|_1 = \max\limits_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| \\ \text{谱范数} & \|A\|_2 = \max\left\{\sqrt{\lambda} : \exists x, A^T A x = \lambda x\right\} \end{cases} \\ \text{非算子范数} \quad F-\text{范数}: \|A\|_F = \left(\sum_{i,j} a_{ij}^2\right)^{1/2} \end{cases} \tag{174}$$

谱半径：定义矩阵$A \in \mathbb{C}^{n \times n}$的谱半径为

$$\rho(A) = \max\{|\lambda| : \exists x, Ax = \lambda x\} \tag{175}$$

- $$\rho(A) \leq \|A\| \tag{176}$$
- $$A^T = A \implies \rho(A) = \|A\|_2 \tag{177}$$
- $$\|A\| < 1 \implies \|(I + A)^{-1}\| \leq \frac{1}{1 - \|A\|} \tag{178}$$

5.2 误差分析

病态：称线性方程$Ax = b$为病态的，如果A或b的微小变化，引起解的巨大变化。

病态矩阵的刻画：对于非奇异矩阵A，以及非零向量b，对于线性方程$Ax = b$的解——

- 考虑$A(x + \delta_x) = b + \delta_b$，成立

$$\frac{\|\delta_x\|}{\|x\|} \leq \|A^{-1}\|\|A\|\frac{\|\delta_b\|}{\|b\|} \tag{179}$$

- 考虑$(A+\delta_A)(x+\delta_x)=b$，如果$\|A^{-1}\|\|\delta_A\|<1$，那么

$$\frac{\|\delta_x\|}{\|x\|} \leq \frac{\|A^{-1}\|\|A\|\frac{\|\delta_A\|}{\|A\|}}{1-\|A^{-1}\|\|A\|\frac{\|\delta_A\|}{\|A\|}} \tag{180}$$

- 考虑$(A+\delta_A)(x+\delta_x)=b+\delta_b$，如果$\|A^{-1}\|\|\delta_A\|<1$，那么

$$\frac{\|\delta_x\|}{\|x\|} \leq \frac{\|A^{-1}\|\|A\|}{1-\|A^{-1}\|\|A\|\frac{\|\delta_A\|}{\|A\|}}\left(\frac{\|\delta_A\|}{\|A\|}+\frac{\|\delta_b\|}{\|b\|}\right) \tag{181}$$

条件数：定义非奇异矩阵A的条件数为$\mathrm{cond}_v(A)=\|A^{-1}\|_v\|A\|_v$。

- 谱条件数：$\mathrm{cond}_2(A)=\dfrac{\max\left\{\sqrt{\lambda}:\exists x, A^TAx=\lambda x\right\}}{\min\left\{\sqrt{\lambda}:\exists x, AA^Tx=\lambda x\right\}}$
- $A^T=A$：$\mathrm{cond}_2(A)=\dfrac{\max\{|\lambda|:\exists x, Ax=\lambda x\}}{\min\{|\lambda|:\exists x, Ax=\lambda x\}}$
- $\mathrm{cond}_v(A)\geq 1$
- 如果$\lambda\neq 0$，那么$\mathrm{cond}_v(\lambda A)=\mathrm{cond}_v(A)$。
- 如果$A^TA=AA^T=I$，那么$\mathrm{cond}_2(A)=1$。
- 如果$Q^TQ=QQ^T=I$，那么$\mathrm{cond}_2(QA)=\mathrm{cond}_2(AQ)=\mathrm{cond}_2(A)$。

病态矩阵的处理：对于病态线性方程$Ax=b$，选取适当的非奇异矩阵P,Q，使得$\mathrm{cond}(PAQ)\ll\mathrm{cond}(A)$，那么

$$Ax=b \iff PAQ(Q^{-1}x)=Pb \tag{182}$$

误差分析：对于线性方程组$Ax=b$，那么

$$\frac{\|x-\tilde{x}\|}{\|x\|} \leq \mathrm{cond}(A)\frac{\|b-A\tilde{x}\|}{\|b\|} \tag{183}$$

第六章：解线性方程的迭代法

6.1 向量序列与矩阵序列

向量序列的极限：称向量序列$\{\{x_k^{(m)}\}_{k=1}^n\}_{m=1}^\infty\subset\mathbb{R}^n$收敛于$\{x_k\}_{k=1}^n\in\mathbb{R}^n$，如果对于任意$1\le k\le n$，成立$\lim\limits_{m\to\infty}x_k^{(m)}=x_k$。

- $$\lim_{n\to\infty}x_n=x\iff\lim_{m\to\infty}\|x_n-x\|=0\tag{184}$$

矩阵序列的极限：称矩阵序列$\{\{a_{i,j}^{(m)}\}_{n\times n}\}_{m=1}^\infty\subset\mathbb{R}^{n\times n}$收敛于$\{a_{ij}\}_{n\times n}\in\mathbb{R}^{n\times n}$，如果对于任意$1\le i,j\le n$，成立$\lim\limits_{m\to\infty}a_{ij}^{(m)}=a_{ij}$。

- $$\lim_{n\to\infty}A_n=A\iff\lim_{m\to\infty}\|A_n-A\|=0\tag{185}$$
- $$\lim_{n\to\infty}A_n=O\iff\lim_{m\to\infty}A_nx=0,\quad\forall x\in\mathbb{R}^n\tag{186}$$
- $$\lim_{n\to\infty}B^n=O\iff\rho(B)<1\iff\exists\|\cdot\|,\quad\|B\|<1\tag{187}$$

6.2 迭代法

一阶线性定常迭代：对于非奇异矩阵A，将线性方程$Ax=b$等价改写为$x=Bx+f$，对于任意初始向量$x^{(0)}$，构造迭代公式

$$x^{(n+1)}=Bx^{(n)}+f\tag{188}$$

误差向量：

$$\varepsilon^{(n)}=x^{(n)}-x=B^n(x^{(0)}-x)\tag{189}$$

分裂矩阵：选择分列矩阵B，使得B^{-1}易求，因此

$$Ax=b\iff x=(I-M^{-1}A)x+M^{-1}b\tag{190}$$

一阶线性定常迭代的基本定理：对于任意初始向量$x^{(0)}$，一阶线性定常迭代$x^{(n+1)}=Bx^{(n)}+f$收敛的充分必要条件为

$$\lim_{n\to\infty}B^n=O\iff\rho(B)<1\iff\exists\|\cdot\|,\quad\|B\|<1\tag{191}$$

压缩映像原理：如果存在算子范数$\|\cdot\|$，使得成立$\|B\|=q<1$，那么一阶线性定常迭代$x^{(n+1)}=Bx^{(n)}+f$收敛，且成立

$$\left\|x^{(n)}-x^*\right\|\le q^n\left\|x^{(0)}-x^*\right\|\tag{192}$$
$$\left\|x^{(n)}-x\right\|\le\frac{q}{1-q}\left\|x^{(n)}-x^{(n-1)}\right\|\tag{193}$$
$$\left\|x^{(n)}-x\right\|\le\frac{q^n}{1-q}\left\|x^{(1)}-x^0\right\|\tag{194}$$

迭代次数：若要求$\left\|x^{(n)}-x\right\|\le\varepsilon$时迭代结束，那么迭代次数为

$$N=\left\lceil\frac{\ln\varepsilon(1-q)-\ln\left\|x^{(1)}-x^{(0)}\right\|}{\ln q}\right\rceil\tag{195}$$

渐进收敛速度：$R(B)=-\ln\rho(B)$，$\rho(B)$越小，收敛速度越快。

6.3 常用一阶定常迭代

名称	迭代矩阵
Jacobi迭代	$I-D^{-1}A$
Gauss-Seidel迭代	$I-(D-L)^{-1}A$
SOR迭代	$I-w(D-wL)^{-1}A$

对于线性方程$Ax=b$，将$A=\{a_{ij}\}_{n\times n}\in\mathbb{R}^{n\times n}$分裂为$D-L-U$：

$$\begin{pmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & \ddots & & \\ & & & a_{n-1,n-1} & \\ & & & & a_{nn} \end{pmatrix} - \begin{pmatrix} 0 & & & & \\ -a_{21} & 0 & & & \\ \vdots & \vdots & \ddots & & \\ -a_{n-1,1} & a_{n-1,2} & \cdots & 0 & \\ -a_{n1} & -a_{n2} & \cdots & -a_{n,n-1} & 0 \end{pmatrix} - \begin{pmatrix} 0 & -a_{21} & \cdots & -a_{1,n-1} & -a_{1n} \\ & 0 & \cdots & -a_{2,n-1} & -a_{2n} \\ & & \ddots & \vdots & \vdots \\ & & & 0 & -a_{n-1,n} \\ & & & & 0 \end{pmatrix} \tag{196}$$

Jacobi迭代：如果$\det D\neq 0$，那么

$$\begin{aligned} Ax=b \iff x=(I-D^{-1}A)x+D^{-1}b \iff x=B_Jx+f_J \\ x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j\neq i}a_{ij}x_j^{(k)}\right), \qquad 1\leq i\leq n, k\in\mathbb{N} \end{aligned} \tag{197}$$

Gauss-Seidel迭代：如果$\det D\neq 0$，那么

$$\begin{aligned} Ax=b \iff x=(I-(D-L)^{-1}A)x+(D-L)^{-1}b \iff x=B_Gx+f_G \\ x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right), \qquad 1\leq i\leq n, k\in\mathbb{N} \end{aligned} \tag{198}$$

逐次超松弛迭代(SOR)迭代：选择松弛因子$w>0$，那么

$$\begin{aligned} Ax=b \iff x=(I-w(D-wL)^{-1}A)x+w(D-wL)^{-1}b \iff x=B_wx+f_w \\ x_i^{(k+1)}=x_i^k+\frac{w}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i}^{n}a_{ij}x_j^{(k)}\right), \qquad 1\leq i\leq n, k\in\mathbb{N} \end{aligned} \tag{199}$$

$w>1$时称为超松弛法，$w<1$时称为低松弛法，$w=1$即为G-S迭代。

定理：如果线性方程$Ax=b$的系数矩阵A为正定三对角矩阵，那么Jacobi迭代中$\rho(B_J)<1$，SOR迭代中松弛因子的最佳选择为

$$w_{\text{opt}}=\frac{2}{1+\sqrt{1-\rho^2(B_J)}} \tag{200}$$

此时$\rho(B_{w_{\text{opt}}})=w_{\text{opt}}-1$。

定理：对于非齐次线性方程$Ax=b$，其中$\det A\neq 0$，如果$A^T=A$，且A的对角线元素$a_{ii}>0$，那么

- Jacobi迭代收敛 $\iff$ $A, 2D-A$为正定矩阵。
- A为正定矩阵 $\Longrightarrow$ G-S迭代收敛。
- A为正定矩阵，且$0<w<2$ $\Longrightarrow$ SOR迭代收敛。

严格对角占优矩阵：称矩阵A为严格对角占优矩阵，如果对于任意i，成立

$$|a_{ii}|>\sum_{j\neq i}|a_{ij}| \tag{201}$$

弱对角占优矩阵：称矩阵A为弱对角占优矩阵，如果对于任意i，成立

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \tag{202}$$

且存在i_0，使得成立

$$|a_{i_0 i_0}| > \sum_{j \neq i_0} |a_{i_0 j}| \tag{203}$$

可约矩阵：称矩阵A为可约矩阵，如果存在置换矩阵P，使得成立

$$P^T A P = \begin{pmatrix} A_{11} & A_{12} \\ O & A_{22} \end{pmatrix} \tag{204}$$

定理：对于非齐次线性方程$Ax = b$，如果A为严格对角占优矩阵，或弱对角占优不可约矩阵，那么

- Jacobi迭代收敛，G-S迭代收敛。
- $0 < w \leq 1 \implies$ SOR迭代收敛。

```
function [D, L, U] = DLUDecomposition(A)

    % 名称：
    % 输入：
    %       A: 欲分解矩阵
    % 输出：
    %       D: 对角矩阵
    %       L: 下三角矩阵
    %       U: 上三角矩阵

    %% 函数

    order = size(A, 1);
    D = zeros(size(A));
    L = zeros(size(A));
    U = zeros(size(A));
    for i = 1: order
        D(i, i) = A(i, i);
        for j = 1: order
            if i > j
                L(i, j) = -A(i, j);
            elseif i < j
                U(i, j) = -A(i, j);
            end
        end
    end

end

```

```
function [judge, root] = JacobiIteration(A, b, x0, n)

    % 名称：      Jacobi迭代
    % 输入：
    %       A:      系数矩阵
    %       b:      右侧矩阵
    %       x0:     初始解
    %       n:      迭代次数

```

```
    % 输出：
    %       judge：是否收敛
    %       root：  迭代解

    %% 函数

    % DLU分解
    D = DLUDecomposition(A);

    % Jacobi矩阵
    BJ = eye(size(A)) - D \ A;

    % 计算特征值
    eigenvalues = eig(BJ);

    % 判断是否收敛
    if max(abs(eigenvalues)) < 1
        judge = 1;
        root = x0;
        for k = 1: n
            root = BJ * root + D \ b;
        end
    else
        judge = 0;
        root = [];
    end

end

```

```
function [judge, root] = GaussSeidelIteration(A, b, x0, n)

    % 名称：      Gauss-Seidel迭代
    % 输入：
    %       A:      系数矩阵
    %       b:      右侧矩阵
    %       x0:     初始解
    %       n:      迭代次数
    % 输出：
    %       judge：是否收敛
    %       root：  迭代解

    %% 函数

    % DLU分解
    [D, L, ~] = DLUDecomposition(A);

    % Gauss-Seidel矩阵
    BG = eye(size(A)) - (D - L) \ A;

    % 计算特征值
    eigenvalues = eig(BG);

    % 判断是否收敛
    if max(abs(eigenvalues)) < 1
        judge = 1;
        root = x0;
```

```
        for k = 1: n
            root = BG * root + (D - L) \ b;
        end
    else
        judge = 0;
        root = [];
    end

end

```

```
function [judge, root] = SORIteration(A, b, w, x0, n)

    % 名称:       SOR迭代
    % 输入:
    %       A:      系数矩阵
    %       b:      右侧矩阵
    %       w:      松弛因子
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       judge:  是否收敛
    %       root:   迭代解

    %% 函数

    % DLU分解
    [D, L, ~] = DLUDecomposition(A);

    % 松弛矩阵
    Bw = eye(size(A)) - (D - w * L) \ A * w;

    % 计算特征值
    eigenvalues = eig(Bw);

    % 判断是否收敛
    if max(abs(eigenvalues)) < 1
        judge = 1;
        root = x0;
        for k = 1: n
            root = Bw * root + (D - w * L) \ b * w;
        end
    else
        judge = 0;
        root = [];
    end

end

```

6.4 共轭梯度法

对于线性方程$Ax=b$，其中A为对称正定矩阵，定义

$$\varphi(x)=\frac{1}{2}(Ax,x)-(b,x) \tag{205}$$

那么记$x^*=A^{-1}b$，成立

$$\nabla\varphi(x) = Ax - b \tag{206}$$

$$\varphi(x+\lambda y) = \varphi(x) + \lambda(Ax-b, y) + \frac{\lambda^2}{2}(Ay, y) \tag{207}$$

$$\varphi(x^*) = -\frac{1}{2}(Ax^*, x^*) \tag{208}$$

$$\varphi(x) - \varphi(x^*) = \frac{1}{2}(A(x-x^*), x-x^*) \tag{209}$$

梯度下降定理：对于线性方程$Ax=b$，其中A为对称正定矩阵，那么

$$Ax^* = b \iff \varphi(x^*) = \inf_x \varphi(x) \tag{210}$$

最速下降法：

$$x^{(n+1)} = x^{(n)} + \alpha_n r^{(n)}, \qquad \alpha_n = \frac{(r^{(n)}, r^{(n)})}{(Ar^{(n)}, r^{(n)})}, \qquad r^{(n)} = b - Ax^{(n)} \tag{211}$$

$$\|x^{(n)} - x^*\|_A \le \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^n \|x^{(0)} - x^*\|_A \tag{212}$$

其中$\lambda_{\max}, \lambda_{\min}$分别为$A$的最大最小特征值，$\|x\|_A = \sqrt{(Ax, x)}$。

特别的，当α为常数时

$$x^{(n+1)} = (I - \alpha A)x^{(n)} + \alpha b \tag{213}$$

收敛的充要条件为

$$\rho(I - \alpha A) < 1 \iff 0 < \alpha < \frac{2}{\lambda_{\max}} \tag{214}$$

α的最佳取值为$\alpha = 2/(\lambda_{\max} + \lambda_{\min})$，此时

$$\rho(I - \alpha A) = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \tag{215}$$

共轭梯度法(CG方法)：

$$\begin{cases} p^{(0)} = r^{(0)} = b - Ax^{(0)} \\ \rho^{(0)} = (r^{(0)}, r^{(0)}) \\ \alpha_0 = \frac{\rho^{(0)}}{(Ap^{(0)}, p^{(0)})} \\ x^{(1)} = x^{(0)} + \alpha_0 p^{(0)} \end{cases}, \quad \begin{cases} r^{(n)} = b - Ax^{(n)} \\ \rho^{(n)} = (r^{(n)}, r^{(n)}) \\ \beta_n = \frac{\rho^{(n)}}{\rho^{(n-1)}} \\ p^{(n)} = r^{(n)} + \beta_n p^{(n-1)} \\ \alpha_n = \frac{\rho^{(n)}}{(Ap^{(n)}, p^{(n)})} \\ x^{(n+1)} = x^{(n)} + \alpha_n p^{(n)} \end{cases} \tag{216}$$

$$\|x^{(n)} - x^*\|_A \le 2\left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^n \|x^{(0)} - x^*\|_A \tag{217}$$

其中$\lambda_{\max}, \lambda_{\min}$分别为$A$的最大最小特征值，$\|x\|_A = \sqrt{(Ax, x)}$。

```
function root = conjugateGradient(A, b, x0, n)

    % 名称:        共轭梯度算法
    % 输入:
    %       A:      系数矩阵
    %       b:      右侧矩阵
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

```

```
    %% 函数

    p = b - A * x0;
    r = b - A * x0;
    rho = dot(r, r);
    alpha = rho / dot(A * p, p);
    root = x0 + alpha * p;
    if n >= 2
        for k = 2: n
            r = b - A * root;
            rho0 = rho;
            rho = dot(r, r);
            beta = rho / rho0;
            p = r + beta * p;
            alpha = rho / dot(A * p, p);
            root = root + alpha * p;
        end
    end

end
```

第七章：非线性方程的数值解

7.1 二分法

二分法：

$$|x^* - x_n| \leq \frac{b-a}{2^{n+1}} \tag{218}$$

```
clear; clc

% 设置函数
f = @(x) x^3 - 2*x - 5;
a = 2;
b = 3;

%设置误差
error = 0.5e-4;

% 迭代
while (b - a) / 2 > error
    c = (a + b) / 2;
    if f(c) == 0
        break
    elseif f(a) * f(c) < 0
        b = c;
    else
        a = c;
    end
end

% 输出根
root = (a + b) / 2;
disp(root)
```

7.2 不动点迭代

不动点迭代：

1.对于方程$x = \varphi(x)$，如果$\varphi([a,b]) \subset [a,b]$，且存在$L < 1$，使得对于任意$x, y \in [a,b]$，成立

$$|\varphi(x) - \varphi(y)| < L|x - y| \tag{219}$$

那么方程$x = \varphi(x)$在$[a,b]$内存在且存在唯一解x^*。

$$|x_n - x^*| < \frac{L^n}{1-L}|x_1 - x_0|, \qquad |x_n - x^*| < \frac{L}{1-L}|x_{n+1} - x_n| \tag{220}$$

2.对于方程$x = \varphi(x)$，如果$\varphi([a,b]) \subset [a,b]$，且对于任意$x \in [a,b]$，成立$|\varphi'(x)| < 1$，那么方程$x = \varphi(x)$在$[a,b]$内存在且存在唯一解$x^*$。

3.对于方程$x = \varphi(x)$，如果$\varphi([a,b]) \subset [a,b]$，且$|\varphi'(x_0)| < 1$，那么方程$x = \varphi(x)$在$x_0$附近存在且存在唯一解$x^*$。

收敛阶：对于不动点迭代$x_{n+1} = \varphi(x_n)$，如果$(x_{n+1} - x^*)/(x_n - x^*)^p \to C \neq 0$，那么称该迭代为$p$阶收敛的。

- 如果$p = 1, |C| < 1$，那么称该迭代为线性收敛。

- 如果$p>1$，那么称该迭代为超线性迭代。
- 如果$p=2$，那么称该迭代为平方收敛。

定理：对于不动点迭代$x_{n+1}=\varphi(x_n)$，以及$p\in\mathbb{N}^*$，如果$\varphi^{(p)}$在x^*邻域内连续，且

$$\varphi^{(k)}(x^*)=0,\quad 1\leq k\leq p-1,\qquad \varphi^{(p)}(x^*)\neq 0 \tag{221}$$

那么该迭代为p阶收敛，且

$$\lim_{n\to\infty}\frac{x_{n+1}-x^*}{(x_n-x^*)^p}=\frac{\varphi^{(p)}(x^*)}{p!} \tag{222}$$

```
function root = fixedPointIteration(phi, x0, n)

    % 名称:        不动点迭代
    % 输入:
    %       phi:    迭代函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

    %% 函数
    root = x0;
    for k = 1: n
        root = phi(root);
    end

end

```

7.3 迭代收敛的加速方法

Aitken迭代：

$$x_{n+1}=\varphi(x_n),\qquad \overline{x}_{n+1}=x_n-\frac{(x_{n+1}-x_n)^2}{x_{n+2}-2x_{n+1}+x_n} \tag{223}$$

Steffensen迭代：对于不动点迭代$x_{n+1}=\varphi(x_n)$，定义

$$y_n=\varphi(x_n),\qquad z_n=\varphi(y_n),\qquad x_{n+1}=x_n-\frac{(y_n-x_n)^2}{z_n-2y_n+x_n} \tag{224}$$

```
function root = SteffensenIteration(phi, x0, n)

    % 名称:        Steffensen迭代
    % 输入:
    %       phi:    迭代函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

    %% 函数
    root = x0;
    for k = 1: n
        y = phi(root);
        z = phi(y);
```

```
        root = root - (y - z)^2 / (z - 2 * y + root);
    end

end

```

7.4 Newton法

名称	迭代方程	收敛阶
Newton法	$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$	平方收敛
简化Newton法/平行弦法	$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_0)}$	
Newton下山法	$x_{n+1}=x_n-\lambda_n\frac{f(x_n)}{f'(x_n)}$ $\lambda_n=\max\left\{\frac{1}{2^r}:\left\lvert f\left(x_n-\frac{f(x_n)}{2^r f'(x_n)}\right)\right\rvert<\lvert f(x_n)\rvert,r\in\mathbb{N}\right\}$	
重根Newton法	$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$	线性收敛
含参m的Newton迭代法	$x_{n+1}=x_n-m\frac{f(x_n)}{f'(x_n)}$	平方收敛
改进Newton迭代法	$x_{n+1}=\varphi(x_n),\varphi(x)=x-\frac{\mu(x)}{\mu'(x)},\mu(x)=\frac{f(x)}{f'(x)}$	平方收敛

Newton法：方程$f(x)=0$的迭代

$$x_{n+1}=\varphi(x_n),\qquad \varphi(x)=x-\frac{f(x)}{f'(x)} \tag{225}$$

如果$f(x^*)=0$且$f'(x^*)\neq 0$，那么Newton迭代在x^*附近为平方收敛，且

$$\lim_{n\to\infty}\frac{x_{n+1}-x^*}{(x_n-x^*)^2}=\frac{f''(x^*)}{2f'(x^*)} \tag{226}$$

简化Newton法/平行弦法：方程$f(x)=0$的迭代

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_0)} \tag{227}$$

Newton下山法：方程$f(x)=0$的迭代

$$x_{n+1}=x_n-\lambda_n\frac{f(x_n)}{f'(x_n)} \tag{228}$$

其中下山因子

$$\lambda_n=\max\left\{\frac{1}{2^r}:\left\lvert f\left(x_n-\frac{f(x_n)}{2^r f'(x_n)}\right)\right\rvert<\lvert f(x_n)\rvert,r\in\mathbb{N}\right\} \tag{229}$$

重根Newton法：如果x^*为方程$f(x)=0$的m重根，那么迭代

$$x_{n+1}=\varphi(x_n),\qquad \varphi(x)=x-\frac{f(x)}{f'(x)} \tag{230}$$

此时

$$\varphi'(x^*)=1-\frac{1}{m} \tag{231}$$

因此该迭代为线性收敛。

含参m的Newton迭代法：如果x^*为方程$f(x)=0$的m重根，那么迭代

$$x_{n+1}=\varphi(x_n),\qquad \varphi(x)=x-m\frac{f(x)}{f'(x)} \tag{232}$$

此时

$$\varphi'(x)=1-m+m\frac{f(x)f''(x)}{(f'(x))^2},\qquad \varphi'(x^*)=0 \tag{233}$$

因此该迭代为平方收敛。

改进Newton迭代法：如果x^*为方程$f(x)=0$的m重根，那么迭代

$$x_{n+1}=\varphi(x_n),\qquad \varphi(x)=x-\frac{\mu(x)}{\mu'(x)},\qquad \mu(x)=\frac{f(x)}{f'(x)} \tag{234}$$

此时x^*为方程$\mu(x)=0$的单根，因此该迭代为平方收敛。

```
function root = NewtonIteration(fun, x0, n)

    % 名称:        Newton迭代
    % 输入:
    %       fun:    函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

    %% 函数
    syms x
    phi = matlabFunction(x - fun(x) ./ diff(fun(x)));
    root = x0;
    for k = 1: n
        root = phi(root);
    end

end

```

```
function root = NewtonDescentIteration(fun, x0, n)

    % 名称:        Newton下山迭代
    % 输入:
    %       fun:    函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

    %% 函数
    syms x
    phi = matlabFunction(fun(x) ./ diff(fun(x)));
    root = x0;
    for k = 1: n
        lambda = 1;
        A = abs(fun(root - phi(root) / 2^lambda));
        B = abs(fun(root));
        while A > B
            lambda = lambda + 1;
```

```
            A = abs(fun(root - phi(root) / 2^lambda));
            B = abs(fun(root));
        end
        root = root - lambda * phi(root);
    end

end

```

```
function root = reRootsNewtonIteration(fun, x0, n)

    % 名称:       重根Newton迭代
    % 输入:
    %       fun:    函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出:
    %       root:   迭代解

    %% 函数
    syms x
    phi = matlabFunction(x - fun(x) ./ diff(fun(x)));
    root = x0;
    for k = 1: n
        root = phi(root);
    end

end

```

```
function order = orderOfRoot(fun, x0)

    % 名称:       求解函数零点的阶
    % 输入:
    %       fun:    函数
    %       x0:     初始解
    % 输出:
    %       order: x0附近零点的阶

    %% 函数
    syms x
    % 找到最近的根
    roots = solve(fun, x);
    [~, index] = min(abs(roots - x0));
    exactRoot = roots(index);

    % 求解精确根的阶
    order = 1;
    Df = matlabFunction(diff(fun(x)));
    while abs(Df(exactRoot)) < 1e-3
        order = order + 1;
        Df = matlabFunction(diff(Df(x)));
    end

end

```

```
function root = NewtonIterationWithParameter(fun, x0, n)

    % 名称：       含参Newton迭代
    % 输入：
    %       fun:    函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出：
    %       root:   迭代解

    %% 函数
    syms x
    order = orderOfRoot(fun, x0);
    phi = matlabFunction(x - order .* fun(x) ./ diff(fun(x)));
    root = x0;
    for k = 1: n
        root = phi(root);
    end

end
```

```
function root = improvingNewtonIteration(fun, x0, n)

    % 名称：       改进Newton迭代
    % 输入：
    %       fun:    函数
    %       x0:     初始解
    %       n:      迭代次数
    % 输出：
    %       root:   迭代解

    %% 函数
    syms x
    mu = matlabFunction(fun(x) ./ diff(fun(x)));
    phi = matlabFunction(x - mu(x) ./ diff(mu(x)));
    root = x0;
    for k = 1: n
        root = phi(root);
    end

end
```

7.5 弦截法与抛物线法

弦截法：方程$f(x)=0$的迭代

$$x_{n+2}=\varphi(x_{n+1},x_n),\qquad \varphi(x,y)=x-\frac{x-y}{f(x)-f(y)}f(x) \tag{235}$$

如果f在x^*邻域内二阶连续可微，且$f'\neq 0$，那么弦截法的收敛阶为

$$p=\frac{1+\sqrt{5}}{2} \tag{236}$$

抛物线法：已知经过x_n,x_{n+1},x_{n+2}三点的抛物线为

$$p_2(x)=f(x_{n+2})+f[x_{n+1},x_{n+2}](x-x_{n+2})+f[x_n,x_{n+1},x_{n+2}](x-x_{n+1})(x-x_{n+2}) \tag{237}$$

以$p_2(x)$的零点作为x_{n+3}，方程$f(x)=0$的迭代

$$x_{n+3}=x_{n+2}-2\frac{f(x_{n+2})}{\omega_{n+3}+\operatorname{sgn}(\omega_{n+3})\sqrt{\omega_{n+3}^2-4f(x_{n+2})f[x_n,x_{n+1},x_{n+2}]}} \tag{238}$$

$$\omega_{n+3}=f[x_{n+1},x_{n+2}]+f[x_n,x_{n+1},x_{n+2}](x_{n+2}-x_{n+1}) \tag{239}$$

```
(*定义函数 f*)f[x_] := x^3 - 3 x - 1

(*定义差商函数 F*)
F[args__] :=
 Total[f[#1]/
     Times @@ (#1 - Delete[{args}, Position[{args}, #1]]) & /@ {args}]

(*定义函数 omega*)
w[x_, y_, z_] := F[y, z] + F[x, y, z] (z - y)

(*定义抛物线函数 parabola*)
parabola[x_, y_, z_] :=
 Simplify[z - (2 f[z])/(w[x, y, z] +
      Sign[w[x, y, z]] Sqrt[(w[x, y, z])^2 - 4 f[z] F[x, y, z]])]

(* 迭代求解*)
x0 = 1;
x1 = 3;
x2 = 2;
x3 = parabola[x0, x1, x2];
x4 = parabola[x1, x2, x3];
```

7.6 非线性方程组的解法

非线性方程组:

$$\begin{cases} f_1(x_1,\cdots,x_n)=0 \\ \vdots \\ f_n(x_1,\cdots,x_n)=0 \end{cases} \iff F(x)=0 \tag{240}$$

向量函数的导数：对于向量函数

$$F(x)=\begin{pmatrix} f_1(x_1,\cdots,x_n) \\ \vdots \\ f_n(x_1,\cdots,x_n) \end{pmatrix} \tag{241}$$

其导数F的Jacobi矩阵

$$F'=\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{242}$$

压缩映射原理：对于定义在区域$D\subset\mathbb{R}^n$上的向量函数Φ，如果存在闭集$D_0\subset D$和实数$0<L<1$，使得对于任意$x,y\in D_0$，成立$\Phi(x)\in D_0$，和

$$\|\Phi(x)-\Phi(y)\|\le L\|x-y\| \tag{243}$$

那么Φ在区域D中存在且存在唯一不动点x^*，且对于任意$x^{(0)}\in D_0$，由不动点迭代

$$x^{(n+1)} = \Phi(x^{(n)}) \tag{244}$$

得到的向量序列$\{x^{(n)}\}_{n=1}^{\infty}$满足$x^{(n)} \to x^*$，同时误差估计为

$$\|x^{(n)} - x^*\| \leq \frac{L^n}{1-L}\|x^{(1)} - x^{(0)}\| \tag{245}$$

局部收敛定理：如果向量函数Φ在区域$D \subset \mathbb{R}^n$内存在不动点x^*，且Φ的分量函数存在连续偏导数，且$\rho(\Phi'(x^*)) < 1$，那么存在x^*的邻域U，使得对于任意$x^{(0)} \in U$，由不动点迭代

$$x^{(n+1)} = \Phi(x^{(n)}) \tag{246}$$

得到的向量序列$\{x^{(n)}\}_{n=1}^{\infty}$满足$x^{(n)} \to x^*$。其中$\rho(\Phi'(x^*))$为向量函数$\Phi$的Jacobi矩阵的谱半径。

p**阶收敛**：称不动点迭代$x^{(n+1)} = \Phi(x^{(n)})$为$p$阶收敛，如果其得到的向量序列$\{x_n\}_{n=1}^{\infty}$满足$x_n \to x^*$，且存在$C > 0$，使得成立

$$\lim_{n\to\infty} \frac{\|x^{(n+1)} - x^*\|}{\|x^{(n)} - x^*\|^p} = C \tag{247}$$

Newton迭代：非线性方程组$F(x) = 0$的Newton迭代为

$$x^{(n+1)} = x^{(n)} - (F'(x^{(n)}))^{-1}F(x^n) \tag{248}$$

Newton迭代定理：对于区域$D \subset \mathbb{R}^n$上的向量函数$F(x)$，如果$F(x^*) = 0$，且F在x^*的开邻域$U \subset D$上存在连续导数，同时$F'(x^*)$非奇异，那么Newton迭代得到的向量序列$\{x^{(n)}\}_{n=1}^{\infty}$在$U$的闭子集$F$上超线性收敛于$x^*$。若存在$L > 0$，使得对于任意$x \in F$，成立

$$\|F'(x) - F'(x^*)\| \leq L\|x - x^*\| \tag{249}$$

那么此时Newton迭代为平方收敛。

第九章：常微分方程初值问题的数值解

9.1 简单的数值方法

初值问题：初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}x}=f(x,y)\\ y(x_0)=y_0\end{cases}\tag{250}$$

Picard定理：设初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}x}=f(x,y)\\ y(x_0)=y_0\end{cases}\tag{251}$$

其中$f(x,y)$在矩形区域$D:\begin{cases}|x-x_0|\le a\\|y-y_0|\le b\end{cases}$内连续，而且对$y$满足Lipschitz条件，则该初值问题在区间$[x_0-h,x_0+h]$上存在并且存在唯一解，其中常数

$$h=\min\left\{a,\frac{b}{M}\right\},\qquad M>\max_{(x,y)\in D}|f(x,y)|\tag{252}$$

名称	迭代公式
Euler公式	$y_{n+1}=y_n+hf(x_n,y_n)$
后退Euler公式	$y_{n+1}=y_n+hf(x_{n+1},y_{n+1})$
中心Euler公式	$y_{n+1}=y_{n-1}+hf(x_n,y_n)$
梯形公式	$y_{n+1}=y_n+\int_{x_n}^{x_{n+1}}f(x,y)\mathrm{d}x$
积分利用左矩形公式	$y_{n+1}=y_n+hf(x_n,y_n)$
积分利用右矩形公式	$y_{n+1}=y_n+hf(x_{n+1},y_{n+1})$
积分利用梯形公式	$\begin{cases}y_p=y_n+hf(x_n,y_n)\\y_c=y_n+hf(x_{n+1},y_p)\\y_{n+1}=\dfrac{1}{2}(y_p+y_c)\end{cases}$

```matlab
function matrix = EulerFormula(fun, h, x0, xend, y0)

    % 名称:          Euler公式
    % 输入:
    %       fun:      函数
    %       h:        步长
    %       x0:       初始x值
    %       xend:     终止x值
    %       y0:       初始y值
    % 输出:
    %       matrix:   近似解

    %% 函数
    n = length(x0: h: xend);
    matrix = [x0: h: xend; y0, zeros(1, n-1)];
    for k = 1: n-1
```

```
        matrix(2, k+1) = matrix(2, k) + h * fun(matrix(1, k), matrix(2, k));
    end

end

```

9.2 单步法的局部阶段误差与阶

显示单步法：

$$y_{n+1}=y_n+h\varphi(x_n,y_n,h) \tag{253}$$

隐式单步法：

$$y_{n+1}=y_n+h\varphi(x_n,y_n,y_{n+1},h) \tag{254}$$

局部截断误差：设$y(x)$为初值问题的精确解，那么定义显示单步法在x_{n+1}处的局部阶段误差为

$$T_{n+1}=y(x_{n+1})-y(x_n)-h\varphi(x_n,y(x_n),h) \tag{255}$$

精度：设$y(x)$为初值问题的精确解，如果存在最大整数p使得单步法的局部阶段误差满足

$$T_{n+1}=O(h^{p+1}) \tag{256}$$

那么称该方法为p阶精度。

9.3 Runge-Kutta方法

r**级Runge-Kutta方法**：

$$\begin{cases}y_{n+1}=y_n+h\varphi(x_n,y_n,h)\\ \varphi(x_n,y_n,h)=\sum\limits_{i=1}^{r}c_iK_i\\ K_1=f(x_n,y_n)\\ K_i=f\left(x_n+\lambda_ih,y_n+h\sum\limits_{j=1}^{i-1}\mu_{ij}K_j\right), \qquad 2\le i\le r\end{cases} \tag{257}$$

名称	迭代公式
一级Runge-Kutta方法	$y_{n+1}=y_n+hf(x_n,y_n)$
二级Runge-Kutta方法	$\begin{cases}y_{n+1}=y_n+h((1-a)K_1+aK_2)\\ K_1=f(x_n,y_n)\\ K_2=f(x_n+h/(2a),y_n+hK_1/(2a))\end{cases}$
改进Euler法（$a=1/2$）	$y_{n+1}=y_n+\frac{h}{2}(f(x_n,y_n)+f(x_n+h,y_n+hf(x_n,y_n)))$
中点公式（$a=1$）	$y_{n+1}=y_n+hf\left(x_n+\frac{h}{2},y_n+\frac{h}{2}f(x_n,y_n)\right)$
经典三阶Runge-Kutta方法	$\begin{cases}y_{n+1}=y_n+\frac{h}{6}(K_1+4K_2+K_3)\\ K_1=f(x_n,y_n)\\ K_2=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1\right)\\ K_3=f\left(x_n+h,y_n-hK_1+2hK_2\right)\end{cases}$
经典四阶Runge-Kutta方法	$\begin{cases}y_{n+1}=y_n+\frac{h}{6}(K_1+2K_2+2K_3+K_4)\\ K_1=f(x_n,y_n)\\ K_2=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1\right)\\ K_3=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_2\right)\\ K_4=f\left(x_n+h,y_n+hK_3\right)\end{cases}$

```
function matrix = improvingEulerFormula(fun, h, x0, xend, y0)
```

```matlab

    % 名称:         改进Euler公式
    % 输入:
    %       fun:      函数
    %       h:        步长
    %       x0:       初始x值
    %       xend:     终止x值
    %       y0:       初始y值
    % 输出:
    %       matrix:   近似解

    %% 函数
    n = length(x0: h: xend);
    matrix = [x0: h: xend; y0, zeros(1, n-1)];
    for k = 1: n-1
        matrix(2, k+1) = matrix(2, k) ...
            + h * fun(matrix(1, k), matrix(2, k)) / 2 ...
            + h * fun(matrix(1, k) + h, matrix(2, k) + h * fun(matrix(1, k), matrix(2, k))) / 2;
    end

end

```

```matlab
function matrix = Classic4RungeKuttaMethod(fun, h, x0, xend, y0)

    % 名称:         经典四阶Runge-Kutta方法
    % 输入:
    %       fun:      函数
    %       h:        步长
    %       x0:       初始x值
    %       xend:     终止x值
    %       y0:       初始y值
    % 输出:
    %       matrix:   近似解

    %% 函数
    n = length(x0: h: xend);
    matrix = [x0: h: xend; y0, zeros(1, n-1)];
    for k = 1: n-1
        K1 = fun(matrix(1, k), matrix(2, k));
        K2 = fun(matrix(1, k) + h/2, matrix(2, k) + h*K1/2);
        K3 = fun(matrix(1, k) + h/2, matrix(2, k) + h*K2/2);
        K4 = fun(matrix(1, k) + h, matrix(2, k) + h*K3);
        matrix(2, k+1) = matrix(2, k) + h / 6 * (K1 + 2 * K2 + 2 * K3 + K4);
    end

end

```

9.4 单步法的收敛性与稳定性

收敛性：称数值方法为收敛的，如果对于$x_n = x_0 + nh$，成立

$$\lim_{h \to 0} y_n = y(x_n) \tag{258}$$

收敛性定理：如果单步法

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h) \tag{259}$$

具有$p \geq 1$阶精度，且增量函数φ关于y满足Lipschitz条件

$$|\varphi(x, y_1, h) - \varphi(x, y_2, h)| \leq L|y_1 - y_2| \tag{260}$$

同时$y_0 = y(x_0)$，那么

$$y(x_n) - y_n = O(h^p) \tag{261}$$

Euler方法：如果$f(x, y)$关于y满足Lipschitz条件，那么Euler方法收敛。

改进Euler法：如果$f(x, y)$关于y满足Lipschitz条件，那么改进Euler方法收敛。

相容性：称单步法与初值问题相容，如果

$$\varphi(x, y, 0) = f(x, y) \tag{262}$$

相容性定理：p阶方法与初值问题相容 $\iff p \geq 1$。

稳定性：称数值方法为稳定的，如果在节点值y_n上有大小为δ的扰动，而以后各节点y_m上产生的偏差不超过δ，其中$m > n$。

绝对稳定性：称单步法为绝对稳定的，如果解微分方程$y' = \lambda y$得到的解$y_{n+1} = E(\lambda h)y_n$满足$|E(\lambda h)| < 1$。

- Euler方法：$E(\lambda h) = 1 + \lambda h$
- 二阶Runge-Kutta方法：$E(\lambda h) = 1 + \lambda h + (\lambda h)^2/2$
- 三阶Runge-Kutta方法：$E(\lambda h) = 1 + \lambda h + (\lambda h)^2/2 + (\lambda h)^3/3!$

绝对稳定域：定义绝对稳定的单步法的绝对稳定域为

$$\{(\lambda, h) \in \mathbb{C}^2 : |E(\lambda h)| < 1\} \tag{263}$$

绝对稳定区间：定义绝对稳定的单步法的绝对稳定区间为

$$\{(\lambda, h) \in \mathbb{R}^2 : |E(\lambda h)| < 1\} \tag{264}$$

A-稳定性：称单步法为A-稳定的，如果

$$|E(\lambda h)| < 1 \implies \mathrm{Re}(\lambda h) < 0 \tag{265}$$