数值计算实验报告

姓名 唐超

班级 智能 16

学号 201600181073

Chapter 1 误差理论

Problem 1

构造算法和 Matlab 程序，以便精确计算所有情况下的二次方程的根，包括 $|b| \approx \sqrt{b^2-4ac}$ 的情况。

Answer

对于一元二次方程 $ax^2+bx+c=0$ ，当 $|b| \approx \sqrt{b^2-4ac}$ 时，为避免其值过小引起巨量消失，从而带来精度损失。若 $b>0$ ，用下列公式求根：

$$x_1 = \frac{-2c}{b+\sqrt{b^2-4ac}},\quad x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a} \tag{1.1}$$

若 $b<0$ ，则应该用下列公式求根：

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}, x_2 = \frac{-2c}{b-\sqrt{b^2-4ac}} \tag{1.2}$$

因此精确求解所有情况下二次方程的根的代码如下：

Code

```
function [x1,x2] = NM_1_1(a,b,c)

if b*b-4*a*c>=0
    if b>=0
        x1 = -2*c/(b+sqrt(b*b-4*a*c));
        x2 = (-b-sqrt(b*b-4*a*c))/2*a;
    else
        x1 = (-b+sqrt(b*b-4*a*c))/2*a;
        x2 = -2*c/(b-sqrt(b*b-4*a*c));
    end
else
    disp('no roots');
end
```

Problem 2

对下列 3 个差分方程计算出前 10 个数值近似值。在每种情况下引入一个小的初始误差，如果没有初始误差，则每个差分方程将生成序列$\left\{\frac{1}{2^n}\right\}_{n=1}^{\infty}$，用表和图的形式分别输出结果。

(a) $r_0 = 0.994, r_n = \frac{1}{2}r_{n-1}, n = 1, 2, \ldots$

(b) $p_0 = 1, p_1 = 0.497, p_n = \frac{3}{2}p_{n-1} - \frac{1}{2}p_{n-2}, n = 2, 3, \ldots$

(c) $q_0 = 1, q_1 = 0.497, q_n = \frac{5}{2}q_{n-1} - q_{n-2}, n = 2, 3, \ldots$

Answer

输出结果如下

表 1.1 序列$\{x_n\} = \{1/2^n\}$以及近似值$\{r_n\}, \{p_n\}$和$\{q_n\}$

n	x_n	r_n	p_n	q_n
0	1.0000000000	0.9940000000	1.0000000000	1.0000000000
1	0.5000000000	0.4970000000	0.4970000000	0.4970000000
2	0.2500000000	0.2485000000	0.2455000000	0.2425000000
3	0.1250000000	0.1242500000	0.1197500000	0.1092500000
4	0.0625000000	0.0621250000	0.0568750000	0.0276250000
5	0.0312500000	0.0310625000	0.0254375000	0.0506875000
6	0.0156250000	0.0155312500	0.0097187500	0.1835937500
7	0.0078125000	0.0077656250	0.0018593750	0.4844218750
8	0.0039062500	0.0038828125	0.0020703125	1.2207734375
9	0.0019531250	0.0019414063	0.0040351562	3.0537929688
10	0.0009765625	0.0009707031	0.0050175781	7.6324121094

表 1.2 误差序列$\{x_n - r_n\}, \{x_n - p_n\}$和$\{x_n - q_n\}$

n	$x_n - r_n$	$x_n - p_n$	$x_n - q_n$
0	0.0060000000	0.0000000000	0.0000000000
1	0.0030000000	0.0030000000	0.0030000000
2	0.0015000000	0.0045000000	0.0075000000
3	0.0007500000	0.0052500000	0.0157500000
4	0.0003750000	0.0056250000	0.0348750000

5	0.0001875000	0.0058125000	0.0819375000
6	0.0000937500	0.0059062500	0.1992187500
7	0.0000468750	0.0059531250	0.4922343750
8	0.0000234375	0.0059765625	1.2246796875
9	0.0000117188	0.0059882812	3.0557460938
10	0.0000058594	0.0059941406	7.6333886719

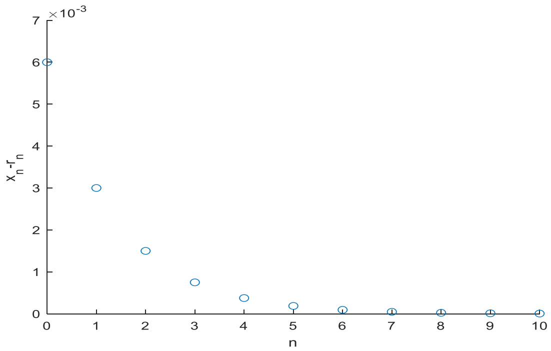


图 1.1　误差序列$\{x_n - r_n\}$

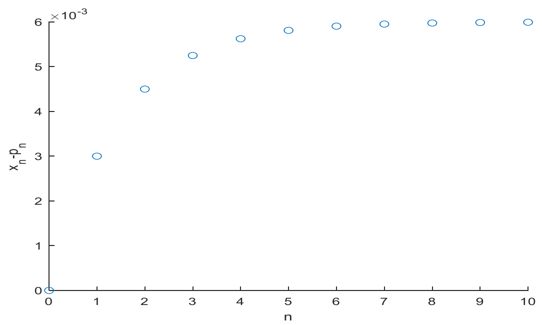


图 1.2　误差序列$\{x_n - p_n\}$

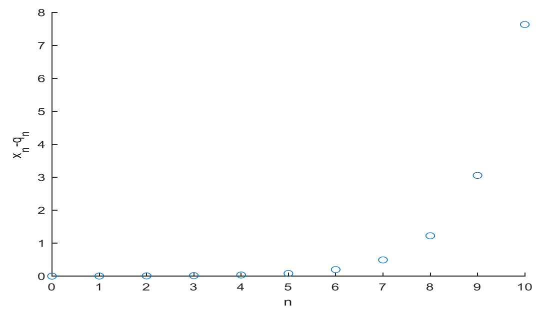


图 1.3　误差序列$\{x_n - q_n\}$

可以看出，$\{r_n\}$ 的误差是稳定的，且按指数级递减。$\{p_n\}$ 的误差也是稳定的。$\{q_n\}$ 的误差是不稳定的，且按指数级增长。

Chapter 2　非线性方程求根

Problem 1

利用牛顿法求解方程

$$\frac{1}{2}+\frac{1}{4}x^2-x\sin x-\frac{1}{2}\cos 2x=0$$

分别取 $x_0=\frac{\pi}{2},5\pi,10\pi$，使得精度不超过$10^{-5}$，比较初值对计算结果的影响。

牛顿-拉夫森定理

设 $f\in C^2[a,b]$，且 $\exists p\in[a,b]$，满足 $f(p)=0$。若 $f'(p)\neq 0$，$\exists\delta>0$，对任意初始近似值 $p_0\in[p-\delta,p+\delta]$，使得由如下定义的序列$\{p_k\}_{k=0}^{\infty}$收敛到 p：

$$p_k=p_{k-1}-\frac{f(p_{k-1})}{f'(p_{k-1})},k=1,2,... \tag{2.1}$$

Answer

在不同初值下，牛顿迭代结果如下：

表 2.1　不同初值求解结果

初值	$\pi/2$	π	10π
零点	1.89249	1.89279	-7647350975.56381
零点函数值	0.000006	0.000005	14620494228936600000
误差	0.003026	0.002722	2227547899
迭代次数	6	10	10000

可以看出，当初值 $x_0=\frac{\pi}{2},5\pi$ 时，分别经过 6 次和 10 次迭代，零点结果已经满足精度要求。当初值 $x_0=10\pi$，此时初值偏离真实零点较大，造成迭代序列已经不收敛于真实零

点，经过最大迭代次数 1000 次迭代后，零点为-7647350975.56381，显然是错误的。

上面的例子说明了牛顿法对于初值的选择非常敏感，当初值偏离真实零点过远时，会造成迭代序列发散。

Code:

```
a = [pi/2 5*pi 10*pi];

P0 = zeros(1,length(a));

Err = zeros(1,length(a));

I = zeros(1,length(a));

Y = zeros(1,length(a));

for k = 1:length(a)
    [p0,err,i,y] = NM_2_1NEWTON('f','df',a(k),10^-5,10^-5,10000);
    P0(k) = p0;
    Err(k) = err;
    I(k) = i;
    Y(k) = y;
End

function [p0,err,i,y] = newton(f,df,p0,delta,epsilon,max_iteration)
%   delta,epsilon are the tolerances of p0 and y respectively.
    for i = 1:max_iteration
        p1 = p0 - feval(f,p0)/feval(df,p0);
        err = abs(p0-p1);
        p0 = p1;
        y = feval(f,p0);
        if (err<delta)||(abs(y)<epsilon)
            break
        end
    end
end
```

Problem 2

已知

$$f(x)=5x-e^x$$

在 $(0,1)$ 上有一个实根，试分别利用二分法、牛顿法、割线法、错位法设计相应的计算格

式，并编程求解（精确到 4 位小数）。

二分法

根据精度要求计算最大迭代次数max1 =1+round((log(b-a)-log(delta))/log(2))=18，取$a=0,b=1$，令$c=(a+b)/2$，若$f(a)f(c)<0$，令$b=c$，若$f(b)f(c)<0$，令$a=c$，重复以上步骤，直至达到迭代次数。

Answer：

经过 18 次迭代，得到 $f(x)$ 的一个根 $x_0=0.2592$， $f(x_0)=1.2020\times10^{-5}$， 误差 $err=3.8147\times10^{-6}$.

Code：

```
function [c,err,yc] = bisect(f,a,b,delta)
%Input  - f is the function input as a string 'f'
%       - a and b are the left and right  end points
%       - delta is the tolerance
%Output - c is the zero
%       - yc=f(c)
%       - err is the error estimate for c
ya=feval(f,a)
yb=feval(f,b)
if ya*yb<0
    max1 = 1+round((log(b-a)-log(delta))/log(2));  %应向上取整。round 为四舍五入取整，为避免可能“四舍”向下取整，前面加 1
    for k = 1:max1
        c = (a+b)/2;
        yc = feval(f,c);
        if yc==0
            a = c;
            b = c;
        elseif yb*yc>0
            b = c;
            yb = yc;
        else
            a = c;
            ya = yc;
        end
        if b-a<delta
            break
```

```
        end
    end
    c = (a+b)/2;
    err = abs((b-a)/2);
    yc = feval(f,c);
else
    disp('please enter a,b with f(a)*f(b)<0');
end
end
```

错位法

错位法与二分法求解方程算法不同点在于 c 不再是区间 (a,b) 的中点，而是 $(a,f(a))$ 和 $(b,f(b))$ 连线与 x 轴交点，以加快迭代的收敛速率。

$$c = b - \frac{f(b)(b-a)}{f(b)-f(a)} \tag{2.2}$$

Answer

经过 13 次迭代，得到 $f(x)$ 的一个根 $x_0 = 0.2592$，$f(x_0) = 2.2204\times10^{-16}$，误差 $err = 5.5511\times10^{-17}$。

Code

```
function [c,err,yc] = regula(f,a,b,delta,max_iteration)
%Input  - f is the function input as a string 'f'
%       - a and b are the left and right  end points
%       - delta is the tolerance
%Output - c is the zero
%       - yc=f(c)
%       - err is the error estimate for c
ya=feval(f,a);
yb=feval(f,b);
if ya*yb<0
    for k = 1:max_iteration
        c = b - feval(f,b)*(b-a)/(feval(f,b)-feval(f,a));
        yc = feval(f,c);
        if yc==0
            a = c;
```

```
            b = c;
        elseif yb*yc>0
            b = c;
            yb = yc;
        else
            a = c;
            ya = yc;
        end
        if b-a<delta
            break
        end
    end
    c = b - feval(f,b)*(b-a)/(feval(f,b)-feval(f,a));
    err = max(c-a,b-c);
    yc = feval(f,c);
else
    disp('please enter a,b with f(a)*f(b)<0');
end
end
```

牛顿法

Answer

取初值 $p_0 = 0.5$，经过 3 次迭代，得到 $x_0 = 0.2592$，$f(x_0) = -5.8371\times10^{-10}$，误差 $err = 3.0015\times10^{-5}$. 代码如 Problem 1 中所示。

割线法

割线法包含的公式与错位法的公式一样，只是在关于如何定义每个后续项的逻辑判定上不一样。需要两个靠近真实零点$(p,0)$的初始点$(p_0, f(p_0))$和$(p_1, f(p_1))$，定义 p_2 为经过两个初始点的直线与 x 轴的交点的横坐标，p_2 比 p_0 或 p_1 更接近 p。

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} \tag{2.3}$$

根据两点迭代公式可得到一般项为

$$p_{k+1} = p_k - \frac{f(p_k)(p_k - p_{k-1})}{f(p_k) - f(p_{k-1})} \tag{2.4}$$

由此得到一个收敛于真实零点 p 的迭代序列$\{p_k\}_{k=1}^{\infty}$.

Answer

初次取 $p_0=10$， $p_1=8$，经过 14 次迭代，得到 $f(x)=0$ 位于 $(0,1)$ 外的另一个零点 $x_0^{'}=2.5426$， $f(x_0^{'})=-5.6493\times10^{-6}$，误差 $err=1.7319\times10^{-4}$。

又取 $p_0=-3$， $p_1=-2$，经过 5 次迭代，得到 $f(x)=0$ 在 $(0,1)$ 上的一个零点 $x_0=0.2592$， $f(x_0)=-3.2699\times10^{-9}$，误差 $err=5.2225\times10^{-6}$。

可以看出，在一个函数有多个根的情况下，初始点的选择决定了迭代序列收敛于哪一个根。

Code

```
function [p1,err,i,y] = secant(f,p0,p1,delta,epsilon,max_iteration)
  for i = 1:max_iteration
        p2 = p1 - feval(f,p1)*(p1-p0)/(feval(f,p1)-feval(f,p0));
        err = abs(p1-p2);
        p0 = p1;
        p1 = p2;
        y = feval(f,p1);
        if (err<delta)||(abs(y)<epsilon)
            break
        end
  end
end
```

Chapter 3 线性方程组求解

Problem 1

求解线性方程组 $\begin{cases}4x-y+z=7\\4x-8y+z=-21\\-2x+y+5z=15\end{cases}$

(1) 试用 LU 分解求解此方程组

Answer

所求解的方程组可写为 $AX=b$ ，其中 $A=\begin{pmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{pmatrix}$ ，$X=\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ ，$b=\begin{pmatrix} 7 \\ -21 \\ 15 \end{pmatrix}$.

经过 LU 分解，$A=LU$，$L=\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -0.5 & -0.0714 & 1 \end{pmatrix}$，$U=\begin{pmatrix} 4 & -1 & 1 \\ 0 & -7 & 0 \\ 0 & 0 & 5.5 \end{pmatrix}$ ，因此原方程组可写为 $LUX=b$ ，不妨设 $Y=UX$ ，则 $LY=b$ ，通过简单的 forward substitution 解得 $Y=\begin{pmatrix} 7 \\ -28 \\ 16.5008 \end{pmatrix}$ ，再将 Y 带入 $Y=UX$ ，通过 backward substitution 解得 $X=\begin{pmatrix} 2 \\ 4 \\ 3.00014545 \end{pmatrix}$.

Code

```
function [L, U] = LU_factorization(n, A)
%Input  - A is the matrix to be factorized.
%       - n is the dimension of matrix A.
%Output - L is a lower-triangular matrix
%       - U is a upper-triangular matrix
L = zeros(n,n);
U = zeros(n,n);
for i=1:n
    L(i,i) = 1; %L 对角线元素为 1
end
for j=1:n
    U(1,j) = A(1,j)/L(1,1); %first row of U
    L(j,1) = A(j,1)/U(1,1); %first colum of L
en
for i=2:n-1
    U(i,i) = A(i,i);
    for k=1:i-1
        U(i,i) = U(i,i) - L(i,k)*U(k,i);
    end
    if U(i,i)==0
        disp('Factorization impossible');
        break;
    end
    for j=i+1:n
        U(i,j) = A(i,j);
```

```
        L(j,i) = A(j,i);
        for k=1:i-1
            U(i,j) = U(i,j) - L(i,k)*U(k,j); % ith row of U
            L(j,i) = L(j,i) - L(j,k)*U(k,i); % ith column of L
        end
        L(j,i) = L(j,i)/U(i,i);
    end
end
U(n,n) = A(n,n);
for k=1:n-1
    U(n,n) = U(n,n) - L(n,k)*U(k,n);
end
end
```

(2) 分别用 Jacobi, Gauss-Seidel 方法求解此方程组

由于系数矩阵 A 是严格对角占优的，因此此方程组可以用 Jacobi, Gauss-Seidel 方法求解。

采用 Jacobi 方法，设 $tol = 0.00001$ ，初始 $X^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ ，经过 11 次迭代，相对误差 $relerr = 6.5336 \times 10^{-6} < tol$ ，此时 $X = \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix}$ 为满足 tol 下的方程组的解。

Jacobi 迭代的代码如下

```
function [k,err,relerr,X] = Jacobi(A, b, P, tol, max)
% Input  - P is the initial guess.
%        - tol is the tolerance for P.
%        - max is the maximum number of iterations
% Output - X is the approximation to the solution of AX=B

n = length(b);
X = zeros(n,1);
for k=1:max
    for i=1:n
        X(i) = (b(i)-A(i,[1:i-1,i+1:n])*P([1:i-1,i+1:n]))/A(i,i);
    end
    err = abs(norm(X-P));
    relerr = err/(norm(X)+eps);
    P = X;
    if(err<tol)||(relerr<tol)
        break;
    end
```

```
end
```

采用 Gauss-Seidel 方法，仍设 $tol = 0.00001$ ，初始 $X^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ ，仅经过 7 次迭代，相对误差 $relerr = 3.6162 \times 10^{-6} < tol$ ，此时 $X = \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix}$ 为满足 tol 下的方程组的解。两种迭代求解方法结果一致，但显然 Gauss-Seidel 方法的效率更高。

Gauss-Seidel 方法的代码如下

```
function [k,err,relerr,X] = Gauss_Seidel(A, b, P, tol, max)
% Input  - P is the initial guess.
%        - tol is the tolerance for P.
%        - max is the maximum number of iterations
% Output - X is the approximation to the solution of AX=B

n = length(b);
X = zeros(n,1);
for k=1:max
    for i=1:n
        if i==1
            X(1) = (b(1)-A(1,2:n)*P(2:n))/A(1,1);
        elseif i==n
            X(n) = (b(n)-A(n,1:n-1)*X(1:n-1))/A(n,n);
        else
            X(i) = (b(i)-A(i,1:i-1)*X(1:i-1)-A(i,i+1:n)*P(i+1:n))/A(i,i);
        end
    end
    err = abs(norm(X-P));
    relerr = err/(norm(X)+eps);
    P = X;
    if(err<tol)||(relerr<tol)
        break
    end
end
end
```

Problem 2

设有如下三角线性方程组，而且系数矩阵具有严格对角优势：

$$AX = b$$

其中系数矩阵 $A=\begin{pmatrix} d_1 & c_1 & & & & & \\ a_1 & d_2 & c_2 & & & & \\ & a_2 & d_3 & c_3 & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & a_{N-2} & d_{N-1} & c_{N-1} \\ & & & & & a_{N-1} & d_N \end{pmatrix}$, $X=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ ， $b=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}$

(1) 设计一个算法来求解上述方程组，算法必须有效利用系数矩阵的稀疏性。

Answer

为有效利用系数矩阵的稀疏性，不再将 A 整体作为高斯-赛德尔迭代法的输入，而是仅输入每个方程的 2 或 3 个系数。

Algorithm

Input:

the number of equations and unknowns n ;

the entries $a_i, c_i, i=1,2,...,N-1,\ d_j, j=1,2,...,N$ of A ;

the entries $b_i, i=1,2,...,N$ of b ;

an initial approximation $x^{(0)}$;

tolerance TOL ;

maximum number of iterations max1.

Output:

the approximate solution x or a message that the number of iterations was exceeded.

Step1: Set k = 1.

Step2: While ($k \le \max 1$), do steps 3~5.

Step3: Set $x_1^{(1)} = \dfrac{b_1 - c_1 x_2^{(0)}}{d_1}$.

For i = 2:N-1, set $x_i^{(1)} = \dfrac{b_i - a_{i-1} x_{i-1}^{(1)} - c_i x_{i+1}^{(0)}}{d_i}$.

Set $x_N^{(1)} = \dfrac{b_N - a_{N-1} x_{N-1}^{(1)}}{d_N}$.

Step4: If $\left\| x^{(1)} - x^{(0)} \right\| < TOL$, then output $x^{(1)} = (x_1, x_2, ..., x_N)^T$. STOP.

Step5: Set $k = k+1, x^{(0)} = x^{(1)}$.

Step6: Output ('Maximum number of iterations exceeded').

(2) 根据(1)中设计的算法构造一个 MATLAB 程序，并求解下列三角线性方程组。

(a) $a_i = c_i = 1, b_i = 3, i = 1, 2, ..., N-1$， $d_j = 4, j = 1, 2, ..., N$.

Answer

将 a,b,c,d 作为参数带入（1）中的算法，初始 $x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{50\times 1}$ ， $tol = 0.001$ ，经过 6 次迭代， $relerr = 3.6093\times 10^{-4} < tol$ ，得到方程组的解

$$x = \begin{pmatrix} 0.6340 & 0.4641 & 0.5097 & 0.4975 & 0.5007 & 0.4998 & 0.5001 & 0.5 & \ldots & 0.5 & 0.5001 & 0.4998 & 0.5007 & 0.4974 & 0.5096 & 0.4641 & 0.6340 \end{pmatrix}^T$$

（"…"代表的值均为 0.5000）.

(b) $a_i = c_i = 1$， $b_i = 1, i$ 为奇数， $b_i = 2, i$ 为偶数， $i = 1, 2, ..., N-1$， $d_j = 4, j = 1, 2, ..., N$.

Answer

将 a,b,c,d 作为参数带入（1）中的算法，初始 $x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{50\times 1}$ ， $tol = 0.001$ ，经过 6 次迭代， $relerr = 7.3603\times 10^{-4} < tol$ ，得到方程组的解

$$x = \begin{pmatrix} 0.1548 & 0.3812 & 0.3206 & 0.3368 & 0.3325 & 0.3336 & 0.3333 & 0.3334 & \ldots & 0.3334 & 0.3331 & 0.3339 & 0.3316 & 0.3398 & 0.3094 & 0.4227 \end{pmatrix}^T$$

（"…"代表的值均为 0.3334）.

Code

```
max = 100;  %the maximum number of iterations
n = 50;
a = ones(n-1,1);
```

```
b = ones(n,1);
for i=2:50
    b(i)=2;
end
c = ones(n-1,1);
d = 4*ones(n,1);
P = ones(n,1);  %the initial guess
X = zeros(n,1);
max = 1000;  %the maximum number of iterations
tol = 10^-3;    %tol is the tolerance for P
for k=1:max
    for i=1:n
        if i==1
            X(1) = (b(1)-c(1)*P(2))/d(1);
        elseif i==n
            X(n) = (b(n)-a(n-1)*X(n-1))/d(n);
        else
            X(i) = (b(i)-a(i-1)*X(i-1)-c(i)*P(i+1))/d(i);
        end
    end
    err = abs(norm(X-P));
    relerr = err/(norm(X)+eps);
    P = X;
    if(err<tol)||(relerr<tol)
        break
    end
end
```

Problem 3

利用高斯-赛德尔迭代求解带状方程组。

Answer

该带状方程组可写为： $AX = b$

其中 $A = \begin{pmatrix} 12 & -2 & 1 & & & & & & \\ -2 & 12 & -2 & 1 & & & & & \\ 1 & -2 & 12 & -2 & 1 & & & & \\ & 1 & -2 & 12 & -2 & 1 & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & 1 & -2 & 12 & -2 & 1 \\ & & & & & 1 & -2 & 12 & -2 \\ & & & & & & 1 & -2 & 12 \end{pmatrix}_{50\times 50}$ ， $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{50} \end{pmatrix}$ ， $b = \begin{pmatrix} 5 \\ 5 \\ \vdots \\ 5 \end{pmatrix}_{50\times 1}$.

系数矩阵 A 是具有严格对角优势的，因此可以用高斯-赛德尔迭代法求解该方程组。

Answer

采用 3.2 中的算法求解，只对 Step3 做如下修改：

Step3: Set $x_1^{(1)}=\dfrac{b_1-c_1x_2^{(0)}-x_3^{(0)}}{d_1}$，$x_2^{(1)}=\dfrac{b_2-a_1x_1^{(1)}-c_2x_3^{(0)}-x_4^{(0)}}{d_2}$.

For i = 3:N-2, set $x_i^{(1)}=\dfrac{b_i-x_{i-2}^{(1)}-a_{i-1}x_{i-1}^{(1)}-c_ix_{i+1}^{(0)}-x_{i+2}^{(0)}}{d_i}$.

Set $x_{N-1}^{(1)}=\dfrac{b_{N-1}-x_{N-3}^{(1)}-a_{N-2}x_{N-2}^{(1)}-c_{N-1}x_N^{(0)}}{d_{N-1}}, x_N^{(1)}=\dfrac{b_N-x_{N-2}^{(1)}-a_{N-1}x_{N-1}^{(1)}}{d_N}$.

初始 $x^{(0)}=\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}_{50\times1}$ ，$tol=0.001$ ，经过 4 次迭代，$relerr=7.3603\times10^{-4}<tol$，得到方程组的解：

$x=\begin{pmatrix}0.4638 & 0.5373 & 0.5091 & 0.4983 & 0.4990 & 0.5 & 0.5001 & 0.5 & \ldots & 0.5 & 0.5003 & 0.4997 & 0.4991 & 0.4985 & 0.5091 & 0.5373 & 0.4638\end{pmatrix}^T$

（“…”代表的值均为 0.5000）.

Chapter 4　插值多项式

Problem 1

在区间$[-5,5]$上，生成 11 个等距插值节点 $x_i, i=0,1,...,10.$ 在相应插值节点上计算函数

$$y(x)=1/(1+x^2)$$

的函数值作为观测值，$y(x_i), i=0,1,2,...,10.$

(1) 利用这 11 个数据点，生成一个 10 次拉格朗日插值多项式 $P_{10}(x)$，并做出插值函数与原函数的对比结果图。

(2) 利用此多项式近似计算

$$\int_{-5}^{5}\frac{1}{1+x^2}dx\approx\int_{-5}^{5}P_{10}(x)dx$$

与解析解比较，分析误差产生的原因。

(3) 利用$\left\{\left(x_i, y\left(x_i\right)\right)\right\}_{i=0}^{10}$构造分片线性插值多项式$P(x)$，并利用此分片插值多项式近似计算积分

$$\int_{-5}^{5}\frac{1}{1+x^2}dx \approx \int_{-5}^{5}P(x)dx$$

与解析解比较，分析误差产生的原因。

(4) 若希望提高积分的计算精度，试提出你自己的建议，并进行实验测试验证。

拉格朗日插值多项式

对于$(n+1)$个插值点$x_i, i=0,1,\ldots,n$，有n次拉格朗日插值多项式：

$$P_n(x)=\sum_{k=0}^{n}L_{n,k}(x)f(x_k)$$

其中$L_{n,k}(x)=\prod_{i=0,i\neq k}^{n}\frac{x-x_i}{x_k-x_i}$为基函数，满足$L_{n,k}(x_i)=\begin{cases}0,\mathrm{i}\neq \mathrm{k}\\ 1,\mathrm{i}=\mathrm{k}\end{cases}$.

$f(x)=P(x)+R(x)$，其中$R(x)=\frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$为拉格朗日余项，用于误差估计。

分片线性插值

对于$(n+1)$个插值点$x_i, i=0,1,\ldots,n$，有分片线性插值公式：

$$P(x)=\sum_{i=0}^{n}l_i(x)f(x_i)$$

其中

$$l_0(x)=\begin{cases}\frac{x-x_1}{x_0-x_1}, x_0\leq x\leq x_1\\ 0, \quad\quad others\end{cases},\ l_i(x)=\begin{cases}\frac{x-x_{i-1}}{x_i-x_{i-1}}, x_{i-1}\leq x\leq x_i\\ \frac{x-x_{i+1}}{x_i-x_{i+1}}, x_i< x\leq x_{i+1}\\ 0, \quad\quad others\end{cases},\ l_n(x)=\begin{cases}\frac{x-x_{n-1}}{x_n-x_{n-1}}, x_{n-1}\leq x\leq x_n\\ 0, \quad\quad others\end{cases}$$

Answer

(1) 生成的 10 次拉格朗日插值多项式为

$$P_{10}(x) = -2.26\times10^{-5}x^{10} - 1.90\times10^{-19}x^{9} + 0.013x^{8} - 1.08\times10^{-17}x^{7} - 0.0244x^{6}$$
$$+1.0061\times10^{-16}x^{5} + 0.197x^{4} - 1.08\times10^{-17}x^{3} - 0.674x^{2} - 1.67\times10^{-16}x + 1$$

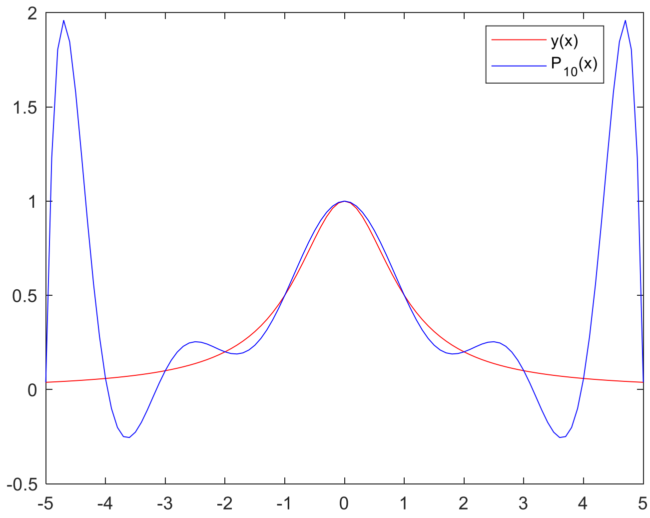


图 4.1 10 次拉格朗日插值多项式插值结果

(2) 经计算

$$\int_{-5}^{5} P_{10}(x)dx = 4.6733, \quad \int_{-5}^{5}\frac{1}{1+x^2}dx = \arctan x\,|_{-5}^{5} = 2.7468$$

相对误差 $r_1 = \dfrac{|2.7468 - 4.6733|}{2.7468} = 0.7014$.

误差原因分析：由于所取插值点较少而插值多项式的次数较大，在差值区间的边界部分插值函数会出现很大的波动，明显偏离原始函数（龙格现象）；另外，从舍入误差看，高次插值由于计算量大，可能造成严重的误差累计。

Code

```
%计算插值多项式并绘制对比结果

x = linspace(-5,5,11);

y = 1./(1+x.^2);

x1=-5:0.1:5;

y1=1./(1+x1.^2);
```

```
plot(x1,y1,'r');

w = length(x);
n = w-1;
L = zeros(w,w);
for k=1:n+1
    V=1;
    for j=1:n+1
        if j~=k
            V=conv(V,poly(x(j)))/(x(k)-x(j));
        end
    end
    L(k,:)=V;
end
C = y*L;

x2=-5:0.1:5;
p = zeros(11,length(x2));
for i=1:length(x2)
    for j=1:11
        p(j,i)=x2(i)^(11-j);
    end
end

y2 = C*p;
hold on
plot(x2,y2,'b')
legend("y(x)", "P_1_0(x)")

%计算[-5,5]上的积分值
f1 = @(xx) 1./(1+xx.^2);

I1 = integral(f1,-5,5)

q = polyint(C);

I2 = diff(polyval(q,[-5,5]))
```

(3) 利用$\left\{\left(x_i, y\left(x_i\right)\right)\right\}_{i=0}^{10}$构造的分片线性插值结果如下

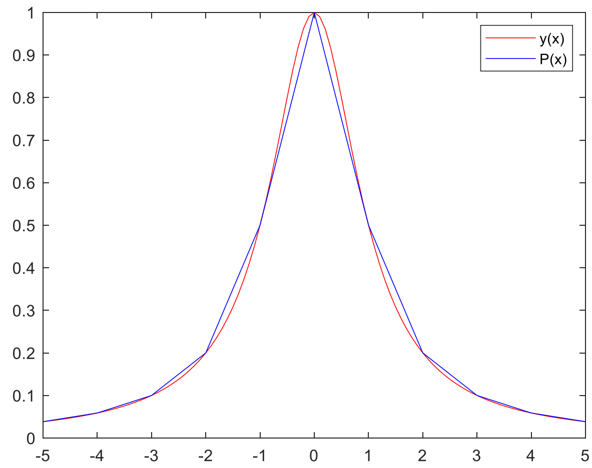


图 4.2 分片线性插值结果

经计算 $\int_{-5}^{5} P(x)dx$=2.7561，与 $\int_{-5}^{5}\frac{1}{1+x^2}dx$=2.7468 的相对误差为

$$r_2=\frac{|2.7468-2.7561|}{2.7468}=0.0034$$

相比上述 10 次插值多项式的积分结果极大地减小了误差。

误差原因分析：一般分片线性插值多项式在插值点都不是连续的，即插值函数不够平滑。当插值点较多时，可以减小因此而造成的误差。

Code

```
%插值节点
x = linspace(-5,5,11);
y = 1./(1+x.^2);

x1=-5:0.1:5;
y1=1./(1+x1.^2);
plot(x1,y1,'r');
% 计算每个子插值区间的线性插值多项式，并绘图
V = zeros(10,2);
for i=1:10
    V(i,1)=y(i+1)-y(i);
    V(i,2)=y(i)*x(i+1)-y(i+1)*x(i);
    hold on
    plot(x(i:i+1),V(i,1).*x(i:i+1)+V(i,2),'b')
end
legend("y(x)", "P(x)")
%计算积分
sumI = 0;
```

```
for k=1:10
    q = polyint(V(k,:));
    sumI = sumI+diff(polyval(q,[k-6,k-5]));
end
```

(4) 我的建议：在区间$[-5,5]$上生成更多的等距插值节点作分段线性插值并计算该积分值，以提高积分的计算精度。

采用 21 个插值节点的结果如下：

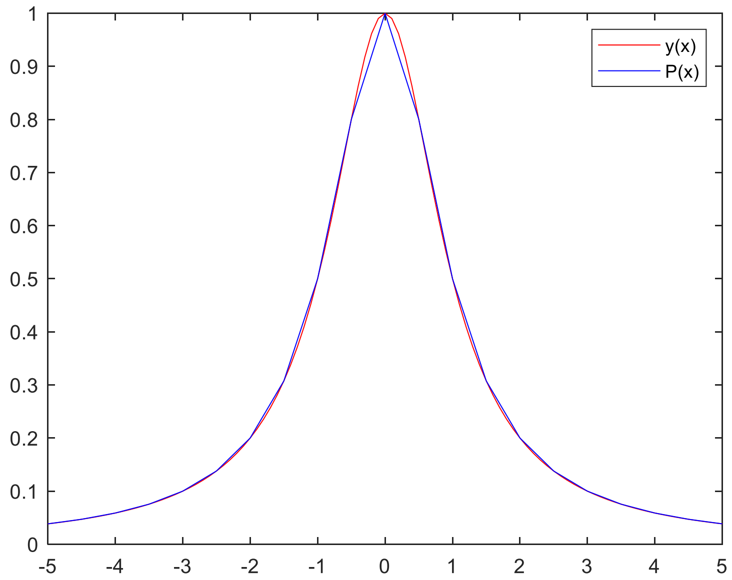


图 4.3　21 个插值节点

此时，$\int_{-5}^{5} P(x)dx=2.7462$，相对误差进一步减小为$r_3 = \dfrac{|2.7468-2.7462|}{2.7468} = 0.00022$.

Chapter 5　最小二乘拟合

Problem 1

根据下列数据，使用幂曲线拟合求解重力常量 g。

(a)

时间t_{k1}	距离d_{k1}
0.200	0.1960

(b)

时间t_{k2}	距离d_{k2}
0.200	0.1965

0.400	0.7835		0.400	0.7855
0.600	1.7630		0.600	1.7675
0.800	3.1345		0.800	3.1420
1.000	4.8975		1.000	4.9095

幂函数拟合

设有 N 个点$\{(x_k, y_k)\}_{k=1}^{N}$，其中横坐标时确定的。最小二乘幂函数拟合曲线$y = Ax^M$的系数 A 为

$$A=\left(\sum_{k=1}^{N} x_k^M y_k\right)\Big/\left(\sum_{k=1}^{N} x_k^{2M}\right)$$

Answer:

经过最小二乘拟合得到：$d_{k1} = 4.8975t_{k1}^2$，从而$g_1 = 9.7950$；$d_{k1} = 4.9095t_{k1}^2$，从而$g_2 = 9.8190$。

Code

```
%幂函数拟合 y=A*x^M
function A = chap5_1(x,y,M)
%x,y 均为列向量
A = ((x.^M)'*y)/((x.^M)'*(x.^M));
end

t = [0.200 0.400 0.600 0.800 1.000]';

d1 = [0.1960 0.7835 1.7630 3.1345 4.8975]';

d2 = [0.1965 0.7855 1.7675 3.1420 4.9095]';

g1 = chap5_1(t,d1,2)*2;

g2 = chap5_1(t,d2,2)*2;
```

Problem 2

根据点 $(0,1),(1,0),(2,0),(3,1),(4,2),(5,2),(6,1)$，求 5 种不同的三次样条插值，其中 $S'(0)=-0.6, S'(6)=-1.8, S''(0)=1, S''(6)=-1$。在同一坐标系中，画出这 5 个三次样条插值和这些数据点。

三次样条插值

定义：设有 N 个点 $\{(x_k, y_k)\}_{k=1}^{N}$，其中 $a=x_0<x_1<...<x_N=b$。如果存在 N 个三次多项式 $S_k(x)$，系数为 $s_{k,0}, s_{k,1}, s_{k,2}$ 和 $s_{k,3}$，满足如下性质：

$a.\ S(x)=S_k(x)=s_{k,0}+s_{k,1}(x-x_k)+s_{k,2}(x-x_k)^2+s_{k,3}(x-x_k)^3,\ x\in[x_k, x_{k+1}], k=0,1,...,N-1$

$b.\ S(x_k)=y_k,\ k=0,1,...,N$

$c.\ S_k(x_{k+1})=S_{k+1}(x_{k+1}),\ k=0,1,...,N-2$

$d.\ S_k^{'}(x_{k+1})=S_{k+1}^{'}(x_{k+1}),\ k=0,1,...,N-2$

$e.\ S_k^{''}(x_{k+1})=S_{k+1}^{''}(x_{k+1}),\ k=0,1,...,N-2$

则称函数 $S(x)$ 为三次样条函数。

根据三次样条函数的定义，其满足下列关系式：

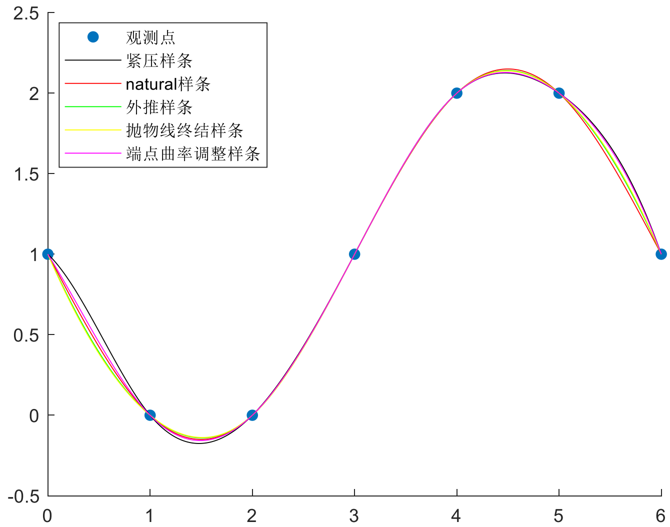
$$h_{k-1}m_{k-1}+2(h_{k-1}+h_k)m_k+h_k m_{k+1}=u_k$$

其中 $m_k=S''(x_k), h_k=x_{k+1}-x_k, d_k=\dfrac{y_{k+1}-y_k}{h_k}, u_k=6(d_k-d_{k-1}),\ k=1,2,...,N-1$。

根据不同的端点约束，可以构造以下 5 中不同的样条：

- **紧压样条**：存在唯一的三次样条曲线，其一阶导数边界条件是 $S'(a)=d_0, S'(b)=d_N$
- **natural 样条：**存在唯一的三次样条曲线，$S''(a)=0,\ S''(b)=0$。
- **外推样条：**存在唯一的三次样条曲线，其中通过对点 x_1 和 x_2 进行外推得到 $S''(a)$，同时通过对点 x_{N-1} 和 x_{N-2} 进行外推得到 $S''(b)$。
- **抛物线终结样条：**存在唯一的三次样条曲线，其中在 $[x_0, x_1]$ 内 $S'''(x)\equiv 0$，而在 $[x_{N-1}, x_N]$ 内 $S'''(x)\equiv 0$。
- **端点曲率调整样条：**存在唯一的三次样条曲线，其中二阶导数的边界条件 $S''(a)$ 和 $S''(b)$ 是确定的。

Answer



从图中可以看出，5 种样条在[2,4]上几乎重合，另外，紧压样条曲线与其他样条曲线相差较远，端点曲率调整样条与外推样条最为接近。

Code

```
function S = spline(method,X,Y,dx0,dxn)
N=length(X)-1;
H=diff(X);
D=diff(Y)./H;
A=H(2:N-1);
B=2*(H(1:N-1)+H(2:N));
C=H(2:N);
U=6*diff(D);

%clamped 样条
if method=='cl'
    B(1)=B(1)-H(1)/2;
    U(1)=U(1)-3*(D(1)-dx0);
    B(N-1)=B(N-1)-H(N)/2;
    U(N-1)=U(N-1)-3*(dxn-D(N));
    for k=2:N-1
    temp=A(k-1)/B(k-1);
    B(k)=B(k)-temp*C(k-1);
    U(k)=U(k)-temp*U(k-1);
    end
    M(N)=U(N-1)/B(N-1);
    for k=N-2:-1:1
    M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end
    M(1)=3*(D(1)-dx0)/H(1)-M(2)/2;
    M(N+1)=3*(dxn-D(N))/H(N)-M(N)/2;
```

```
end
%natural 样条
if method=='na'
    for k=2:N-1
    temp=A(k-1)/B(k-1);
    B(k)=B(k)-temp*C(k-1);
    U(k)=U(k)-temp*U(k-1);
    end
    M(N)=U(N-1)/B(N-1);
    for k=N-2:-1:1
        M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end
    M(1)=0;
    M(N+1)=0;
end
%extrapolated 样条
if method=='ex'
    B(1)=B(1)+H(1)+(H(1))^2/H(2);
    B(N-1)=B(N-1)+H(N)+(H(N))^2/H(N-1);
    A(N-2)=(H(N-1)-(H(N))^2/H(N-1));
    C(1)=H(2)-(H(1))^2/H(2);
    for k=2:N-1
        temp=A(k-1)/B(k-1);
        B(k)=B(k)-temp*C(k-1);
        U(k)=U(k)-temp*U(k-1);
    end
    M(N)=U(N-1)/B(N-1);
    for k=N-2:-1:1
        M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end
    M(1)=M(2)-H(1)*(M(3)-M(2))/H(2);
    M(N+1)=M(N)+H(N)*(M(N)-M(N-1))/H(N-1);
end
%抛物线终结样条
if method=='pt'
    B(1)=B(1)+H(1);
    B(N-1)=B(N-1)+H(N);
    for k=2:N-1
        temp=A(k-1)/B(k-1);
        B(k)=B(k)-temp*C(k-1);
        U(k)=U(k)-temp*U(k-1);
    end
    M(N)=U(N-1)/B(N-1);
    for k=N-2:-1:1
        M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end
    M(1)=M(2);
    M(N+1)=M(N);
end
```

```
%端点曲率调整样条
if method=='ep'
    U(1)=U(1)-H(1)*dx0;
    U(N-1)=U(N-1)-H(N)*dxn;
    for k=2:N-1
        temp=A(k-1)/B(k-1);
        B(k)=B(k)-temp*C(k-1);
        U(k)=U(k)-temp*U(k-1);
    end
    M(N)=U(N-1)/B(N-1);
    for k=N-2:-1:1
        M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end
    M(1)=dx0;
    M(N+1)=dxn;
end

S=zeros(N,4);
for k=0:N-1
    S(k+1,1)=(M(k+2)-M(k+1))/(6*H(k+1));
    S(k+1,2)=M(k+1)/2;
    S(k+1,3)=D(k+1)-H(k+1)*(2*M(k+1)+M(k+2))/6;
    S(k+1,4)=Y(k+1);
end
end

clear,clc

x=[0 1 2 3 4 5 6];

y=[1 0 0 1 2 2 1];

scatter(x,y,'o','filled')

%依次做紧压、natural、外推、抛物线终结、端点曲率调整样条
S1=spline('cl',x,y,-0.6,-1.8);
S2=spline('na',x,y,-0.6,-1.8);
S3=spline('ex',x,y,-0.6,-1.8);
S4=spline('pt',x,y,-0.6,-1.8);
S5=spline('ep',x,y,-0.6,-1.8);
for i=1:6
    x1=i-1:0.01:i;
    y1=polyval(S1(i,:),x1-x(i));
    y2=polyval(S2(i,:),x1-x(i));
    y3=polyval(S3(i,:),x1-x(i));
    y4=polyval(S4(i,:),x1-x(i));
    y5=polyval(S5(i,:),x1-x(i));
    hold on
    plot(x1,y1,'k',x1,y2,'r',x1,y3,'g',x1,y4,'y',x1,y5,'m');
```

```
end
legend('观测点','紧压样条','natural 样条','外推样条','抛物线终结样条','端点曲率调整样条')
```

Chapter 6 数值微分

Problem

求解函数 $f(x)=\tan\left(\cos\left(\frac{\sqrt{5}+\sin(x)}{1+x^2}\right)\right)$ 在 $x=\frac{1+\sqrt{5}}{3}$ 处的导数近似值，精度为小数点后 13 位。

Answer

使用极限的微分求解，$f'(x)\approx\frac{f(x+10^{-n}h)-f(x-10^{-n}h)}{2\cdot(10^{-n}h)}$，其中 n 为足够大的正整数。

经计算，当 $n=7$ 时，求得导数近似值满足精度要求，$f'(x)=1.228597423658$。

Code

```
function [L,n]=difflim(f,x,tol)
max1=100;
h=1;
H(1)=h;
D(1)=(feval(f,x+h)-feval(f,x-h))/(2*h);
E(1)=0;

for n=1:2
    h=h/10;
    H(n+1)=h;
    D(n+1)=(feval(f,x+h)-feval(f,x-h))/(2*h);
    E(n+1)=abs(D(n+1)-D(n));
end
n=2;
while E(n)-E(n+1)>tol&&n<max1
    h=h/10;
    H(n+2)=h;
    D(n+2)=(feval(f,x+h)-feval(f,x-h))/(2*h);
```

```
    E(n+2)=abs(D(n+2)-D(n+1));
    n=n+1;
end
n=length(D)-1;
L=[H' D'];
end
```

Chapter 7　数值积分

Problem 1

用组合辛普森公式求习题 2 中的定积分 $I=\int_a^b \sqrt{1+(f'(x))^2}dx$，精确到小数点后 11 位。

(a) $f(x)=x^3, a=0, b=1$

(b) $f(x)=\sin(x), a=0, b=\frac{\pi}{4}$

(c) $f(x)=e^{-x}, a=0, b=1$

Answer

组合辛普森公式： 通过 $g(x)$ 的 $2M+1$ 个等步长采样点 $x_k=a+kh, k=0,1,...,2M$ 逼近积分

$$\int_a^b g(x)dx \approx \frac{h}{3}(g(a)+g(b))+\frac{2h}{3}\sum_{k=1}^{M-1} g(x_{2k})+\frac{4h}{3}\sum_{k=1}^{M} g(x_{2k-1})$$

(a) $f'(x)=3x^2$，被积函数 $g(x)=\sqrt{1+9x^4}$， $M=5$，则 $h=\frac{b-a}{2M}$=0.1，将相关参数带入组合辛普森公式得到 $I_a=1.54786419399$。

(b) $f'(x)=\cos(x)$，被积函数 $g(x)=\sqrt{1+\cos^2(x)}$， $h=\frac{b-a}{2M}=\frac{\pi}{40}$，将相关参数带入组合辛普森公式得到 $I_b=1.05809581807$。

(c) $f'(x)=-e^{-x}$，被积函数 $g(x)=\sqrt{1+e^{-2x}}$， $h=\frac{b-a}{2M}$=0.1，将相关参数带入组合辛普森公式得到 $I_c=1.19270185509$。

Code

```
function s=simpr1(f,a,b,M)
h=(b-a)/(2*M);
s1=0; s2=0;
for k=1:M
    x=a+h*(2*k-1);
    s1=s1+feval(f,x);
end
for k=1:M-1
    x=a+h*2*k;
    s2=s2+feval(f,x);
end
s=h*(feval(f,a)++4*s1+2*s2+feval(f,b))/3;
end
```

Problem 2

修改组合梯形公式，使之可以求只有若干点函数值已知的函数积分。将程序 7.1 修改为求区间 $[a,b]$ 上过 M 个给定点的函数 $f(x)$ 的积分逼近。利用该程序求过点 $\left\{\left(\sqrt{k^2+1},k^{1/3}\right)\right\}_{k=0}^{13}$ 的函数的积分逼近。

组合梯形公式

通过 $f(x)$ 的 $M+1$ 个等步长采样点 $x_k=a+kh,k=0,1,\ldots,M$ 逼近积分

$$\int_a^b f(x)dx \approx \sum_{k=1}^{M}\frac{h}{2}(f(x_{k-1})+f(x_k))=\frac{h}{2}(f(a)+f(b))+h\sum_{k=1}^{M-1}f(x_k).$$

当不采用等距采样时，设相邻两采样点的距离 $x_k-x_{k-1}=h_k,k=1,2,\ldots,M$，从而，非等距采样的组合梯形公式如下：

$$\int_a^b f(x)dx \approx \sum_{k=1}^{M}\frac{h_k}{2}(f(x_{k-1})+f(x_k))$$

Answer

按上述公式得到，过点 $\left\{\left(\sqrt{k^2+1},k^{1/3}\right)\right\}_{k=0}^{13}$ 的函数的积分逼近 $I=21.8411$。

Code

```
function s=fdj_trap(x,y)
M = length(x)-1;
h = diff(x);
s = 0;
for k=1:M
    s = s+h(k)/2*(y(k)+y(k+1));
end
end

clear,clc
k = 0:13;
x = (k.^2+1).^0.5;
y = k.^(1/3);
s = fdj_trap(x,y)
```

Chapter 9　微分方程数值解

Problem 1

流行病模型。流行病的数学模型描述如下：设有 L 个成员的构成的群落，其中有 P 个感染个体，Q 个未感染个体。令 $y(t)$ 表示时刻 t 感染个体的数量。对于温和的疾病，如普通感冒，每个个体保持存活，流行病从感染者传播到未感染者。由于两组间有 PQ 种可能的接触， $y(t)$ 的变化率正比于 PQ。故该问题可以描述为初值问题：

$$y' = ky(L-y) \quad y(0) = y_0$$

(a) 用 $L = 25000, k = 0.00003, h = 0.2$ 和初值条件 $y(0) = 250$，计算 $[0, 60]$ 上的欧拉近似解。

(b) 画出(a)中的近似解。

(c) 通过求(a)中欧拉方法的纵坐标平均值来估计平均感染个体的数目。

(d) 通过用曲线拟合(a)中的数据，并用积分均值定理估计平均感染个体数目。

欧拉方法

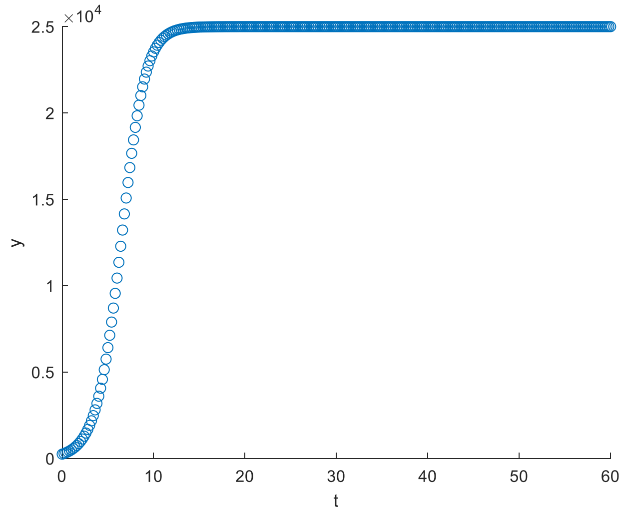
通过计算

$$y_{k+1} = y_k + hf(t_k, y_k), \quad k = 0,1,\ldots,M-1$$

求$[a,b]$上初值问题$y'=f(t,y), y(a)=y_0$的近似解。

Answer

由题意$y'=0.00003y(25000-y)=f(t,y)$，$M=\dfrac{b-a}{h}$=300，则用欧拉方法得到近似解如下图所示：



平均感染个体的数目（用上述欧拉近似解纵坐标平均值估计）为2.2302×10^4。

尝试用前面实验已经编写好的 natural 三次样条插值方法拟合这些近似解，进而利用积分中值定理计算得到平均感染个体数目为2.2334×10^4。

Code

```
function E=euler(f,a,b,ya,M)
h=(b-a)/M;
T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
    Y(j+1)=Y(j)+h*feval(f,T(j),Y(j));
end
E=[T' Y'];
End

clear,clc
f1 = @(t,y) 0.00003*y*(25000-y);
M = 60/0.2;
E=euler(f1,0,60,250,M);
```

```
scatter(E(:,1),E(:,2))
xlabel('t');ylabel('y')
% 估计均值
avg_y = sum(E(:,2))/301;
```

Problem 2

考虑一阶积分-常微分方程

$$y' = 1.3y - 0.25y^2 - 0.0001y\int_0^t y(\tau)d\tau$$

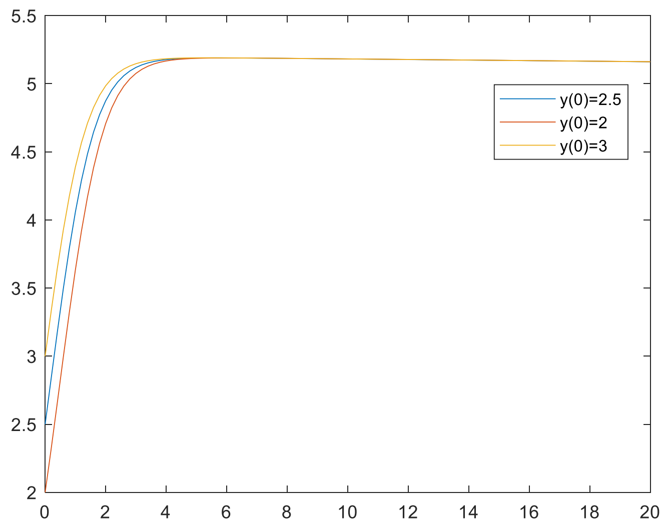
(a) 在区间 $[0,20]$，用欧拉方法和 $h=0.2, y(0)=2.5$ 以及梯形公式求方程的近似解。

(b) 用初值 $y(0)=2$ 和 $y(0)=3$ 重复(a)的计算。

(c) 在同一坐标系中画出(a)和(b)的近似解。

Answer

3 个不同初值的近似解最后都收敛到 5.16，如下



Code

```
clear,clc

h=0.2;

y(1)=2.50;

a=0;b=20;
```

```
T(1)=0;

for i=2:101
y(i)=y(i-1)+h*(1.3*y(i-1)-0.25*(y(i-1))^2-0.0001*y(i-1)*T(i-1));
    T(i)=T(i-1)+h*(y(i-1)+y(i))/2;
end
t=a:h:b;
plot(t,y)
hold on
f(1)=2.00;
R(1)=0;
for i=2:101
f(i)=f(i-1)+h*(1.3*f(i-1)-0.25*(f(i-1))^2-0.0001*f(i-1)*R(i-1));
    R(i)=R(i-1)+h*(f(i-1)+f(i))/2;
end
plot(t,f)
hold on
z(1)=3.00;
S(1)=0;
for i=2:101 z(i)=z(i-1)+h*(1.3*z(i-1)-0.25*(z(i-1))^2-0.0001*z(i-1)*R(i-1));
    S(i)=S(i-1)+h*(z(i-1)+z(i))/2;
end
plot(t,z);
```

Problem 3

用休恩方法求解微分方程

$$y'=3y+3t,\ \ y(0)=1,\ \ y(t)=\frac{4}{3}e^{3t}-t-\frac{1}{3}$$

(a) 令 $h=0.1$, 程序 9.2 执行 20 步，然后令 $h=0.05$，程序 9.2 执行 40 步。

(b) 比较(a)中的两个近似解与精确解 $y(2)$。

(c) 当 h 减半时，(a)中的最终全局误差是否和预期相符？

(d) 将两个近似解和精确解画在同一坐标系中。

休恩方法

通过计算

$$y_{k+1}=y_k+\frac{h}{2}(f(t_k,y_k)+f(t_{k+1},y_k+hf(t_k,y_k)))，\ \ k=0,1,\ldots,M-1$$

求$[a,b]$上的初值问题$y'=t-y, y(a)=y_0$的近似解。

Answer

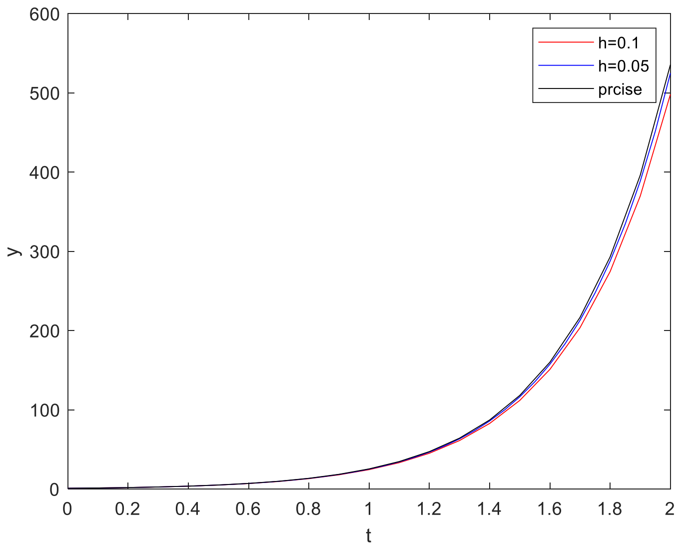
精确解$y(2)=535.5717$。

当$h=0.1$时，得到y在$t=2$处的近似解$y_1(2)$=498.1440，误差$E_1=-37.4277$。

当$h=0.05$时，得到y在$t=2$处的近似解$y_1(2)$=524.8575，误差$E_2=-10.7142$。

可以看出，当步长减半时，误差大约减小 1/4，与预期相符。

两个近似解和精确解绘制如下：



Code

```
function H=heum(f,a,b,ya,h)
M=(b-a)/h;
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
    k1 = feval(f,T(j),Y(j));
    k2 = feval(f,T(j+1),Y(j)+h*k1);
    Y(j+1)=Y(j)+h/2*(k1+k2);
end
H=[T' Y'];
End

clear,clc

f1 = @(t,y) 3*(y+t);
```

```
h1=0.1; h2=0.05;

H1=heum(f1,0,2,1,h1);

H2=heum(f1,0,2,1,h2);

t = 0:0.1:2;

y = 4/3*exp(3*t)-t-1/3;

plot(H1(:,1),H1(:,2),'r',H2(:,1),H2(:,2),'b',t,y,'k')
legend('h=0.1','h=0.05','prcise')
xlabel('t');ylabel('y')
```

Problem 4

改用 $N=4$ 的龙格-库塔方法求解 Problem 3。

4 阶龙格-库塔方法

采用公式

$$y_{k+1}=y_k+\frac{h}{6}(k_1+2k_2+2k_3+k_4)$$

求$[a,b]$上的初值问题 $y'=f(t,y),y(a)=y_0$ 的近似解。

Answer

精确解 $y(2)=535.5717$ 。

当 $h=0.1$ 时，得到 y 在 $t=2$ 处的近似解 $y_1(2)$=535.4019，误差 $E_1=-0.1578$ 。

当 $h=0.05$ 时，得到 y 在 $t=2$ 处的近似解 $y_2(2)$=535.5597，误差 $E_2=-0.0120$ 。

可以看出，当步长减半时，误差大约减小 1/16，与预期相符。

Code

```
function R=rk4(f,a,b,ya,M)
h=(b-a)/M;
T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
    k1=h*feval(f,T(j),Y(j));
    k2=h*feval(f,T(j)+h/2,Y(j)+k1/2);
    k3=h*feval(f,T(j)+h/2,Y(j)+k2/2);
    k4=h*feval(f,T(j)+h,Y(j)+k3);
    Y(j+1)=Y(j)+(k1+2*k2+2*k3+k4)/6;
end
R=[T' Y'];
end
```

Chapter 11 特征值与特征向量

Problem

已知矩阵

$$A=\begin{bmatrix}4 & -1 & 1\\ -1 & 3 & -2\\ 1 & -2 & 3\end{bmatrix}$$

是一个对称矩阵，且其特征值为$\lambda_1=6,\lambda_2=3,\lambda_3=1$. 分别用幂法、对称幂法、反幂法求其最大特征值和特征向量。

(1)幂法

设矩阵 A 有一个主特征值λ，而且对应于λ有唯一的归一化的特征向量V。通过下面的迭代过程可求出特征对λ,V。从下列向量开始：

$$X_0=[1\ 1\ \ldots\ 1]'$$

用如下递归公式递归生成序列$\{X_k\}$：

$$Y_k=AX_k\quad,\quad X_{k+1}=\frac{1}{c_{k+1}}Y_k$$

其中c_{k+1}是Y_k绝对值最大的分量。序列$\{X_k\}$和$\{c_k\}$将分别收敛到V和λ。

Answer

取 $x_0=(1\,1\,1)^T$，经过 20 次迭代，结果精度达到10^{-5}，结果为λ_1=6.0000，$V=(1\ \ -1\ \ 1)^T$。

Code

```
function [lamda,V,cnt,err]=power1(A,X,epsilon,max1)
lamda=0;
cnt=0;
err=1;
state=1;
while((cnt<=max1)&&(state==1))
    Y=A*X;
    %Normalize Y
    [m,~]=max(abs(Y));
    c1=m;
    dc=abs(lamda-c1);
    Y=(1/c1)*Y;

    dv=norm(X-Y);
    err=max(dc,dv);
    X=Y;
    lamda=c1;
    state=0;
    if(err>epsilon)
        state=1;
    end
    cnt=cnt+1;
end
V=X;
end

clear,clc
A=[4 -1 1;-1 3 -2;1 -2 3];
X=[1 1 1]';
[lamda1,V,cnt,err]=power1(A,X,10^(-5),20)
```

(2)反幂法

设$n\times n$矩阵 A 有不同的特征值$\lambda_1,\lambda_2,...,\lambda_n$。为计算特征值$\lambda_j$，可选择常量$\alpha$，使得$\mu_1=\dfrac{1}{\lambda_j-\alpha}$是$(A-\alpha I)^{-1}$的主特征值。选择适当的$X_0$，可通过下列递归公式的得到序列$\{X_k=[x_1^{(k)}\ x_2^{(k)}\ ...\ x_n^{(k)}]'\}$ 和$\{c_k\}$：

$$Y_k = (A - \alpha I)^{-1} X_k\,,\quad X_{k+1} = \frac{1}{c_{k+1}} Y_k$$

其中

$$c_{k+1} = x_j^{(k)}\quad,\quad x_j^{(k)} = \max_{1 \le j \le n} \left\{ \left| x_j^{(k)} \right| \right\}$$

这两个序列将收敛到矩阵$(A-\alpha I)^{-1}$的主特征对μ_1, V_j，从而得到$\lambda_j = \frac{1}{\mu_1} + \alpha$。

Answer

仍取$x_0 = (1\,1\,1)^T$，取α=6.2，发现经过λ_1收敛于 6.4000，而不是实际主特征值 6。又取$\alpha = 5.8$，经过 6 次迭代，结果精度达到10^{-5}，λ_1=6.0000，$V = (1\ -1\ 1)^T$。可见反幂法对于初始α的选取敏感。

Code

```
function [lamda,V,cnt,err]=invpow(A,X,alpha,epsilon,max1)
[n n]=size(A);
A=A-alpha*eye(n);
lamda=0;
cnt=0;
err=1;
state=1;
while((cnt<=max1)&&(state==1))
    Y=A\X;

    %Normalize Y
    [m , ~]=max(abs(Y));
    c1=m;
    dc=abs(lamda-c1);
    Y=(1/c1)*Y;

    dv=norm(X-Y);
    err=max(dc,dv);
    X=Y;
    lamda=c1;
    state=0;
    if(err>epsilon)
        state=1;
    end
    cnt=cnt+1;
end
lamda=alpha+1/c1;
V=X;
```

```
end

clear,clc
A=[4 -1 1;-1 3 -2;1 -2 3];
X=[1 1 1]';
[lamda,V,cnt,err]=invpow(A,X,5.8,10^-5,100);
```

(3)对称幂法

Answer

经过 18 次迭代，精度达到10^{-5}，结果为$\lambda_1=6$，$V=0.5774\cdot(1\ \ -1\ \ 1)^T$。

Code

```
function [lamda,V,k,err]=sympower(A,X,epsilon,max1)
k=1;
X=X./sqrt(X'*X);
while(k<=max1)
 Y=A*X;
 lamda=X'*Y;
 if sqrt(Y'*Y)==0
     break;
 end
 err=sqrt((X-Y./sqrt(Y'*Y))'*(X-Y./sqrt(Y'*Y)));
 X=Y./sqrt(Y'*Y);
 if err<epsilon
     lamda;
     V=X;
     break;
 end
 k=k+1;
end
end

clear,clc
A=[4 -1 1;-1 3 -2;1 -2 3];
X=[1 1 1]';
[lamda,V,k,err]=sympower(A,X,10^-9,40);
```