东南大学

**研究生《数值分析》课程**

**《数值分析》**

**课 程 设 计 报 告**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 焦蔚然 | 学号 | 192787 |

|  |  |  |  |
| --- | --- | --- | --- |
| 成绩 |  | 教师 | 曹婉容 |

**2019年11月**

**一、第一章**

记



则有



给定函数 记



称为函数的阶Taylor多项式。

计算的近似值，可以用如下三种方法：

(Ⅰ)将在处进行Taylor展开，取前10项的和 估计误差



计算出，并回答作为的近似值具有几位有效数字。

(Ⅱ)将在处进行Taylor展开，取前10项的和 估计误差



计算出，并回答作为的近似值具有几位有效数字。

(Ⅲ)将在处进行Taylor展开，取前10项的和 估计误差



计算出，并回答作为的近似值具有几位有效数字。

(Ⅳ)比较上述三种算法的计算精度，体会编程中要注意的“尽量避免大数吃小数”、“尽量避免相近数相减”的准则。

**解：(Ⅰ)MATLAB源程序：**

clc;clear;

syms x; %创建符号对象x

y1=log(1+x); %函数y1=ln(1+x)

N=input('请输入N:'); %进行taylor的N阶展开

S=taylor(y1,x,0,'Order',N); %y1在点0处进行N阶泰勒展开

dif=abs(y1-S); %对函数y1和展开式S求差值并取绝对值

data=zeros(1,10000); %定义数组data存储差值

n=1;

for z=0:0.001:1 %计算x在[0,1]精度为0.001时的误差

answer=subs( dif,x,z);

answer=single(answer);

data(n)=answer;

n=n+1;

end

result=max(data); %求泰勒展开的误差最大值

fprintf('result=%f\n',result); %输出

S1=subs(S,x,1); %求在1处泰勒展开的值

fprintf('S(1)=%f\n',S1); %输出

e1=subs( dif,x,1); %求在1处误差的大小

fprintf('e1=%f\n',e1); %输出

**运行结果：**

请输入N:10

result=0.052488

S(1)=0.745635

e1=0.052488

**即为：**

，，，

作为的近似值没有有效数字。

**(**Ⅱ**)MATLAB源程序：**

clc;clear;

syms x; %创建符号对象x

y2=log(1/(1+x)); %函数y2=ln(1/(1+x))

N=input('请输入N:'); %进行taylor的N阶展开

S=taylor(y2,x,0,'Order',N); %y2在点处，进行N阶泰勒展开

dif=abs(y2-S); %对函数y2和展开式S求差值并取绝对值

data2=zeros(1,10000); %定义数组data存储差值

n=1;

for z=-0.5:0.001:0.5 %计算x在[-0.5,0.5]精度为0.001时的误差

answer=subs( dif,x,z);

answer=single(answer);

data2(n)=answer;

n=n+1;

end

result=max(data2); %求泰勒展开的误差最大值

fprintf('result=%f\n',result); %输出

S2=subs(S,x,-0.5); %求在-0.5处泰勒展开的值

fprintf('S(-0.5)=%f\n',S2); %输出

e2=subs( dif,x,-0.5); %求在-0.5处误差的大小

fprintf('e1=%f\n',e2); %输出

**运行结果：**

请输入N:10

result=0.000180

S(-0.5)=0.692967

e2=0.000180

**即为：**

，，

，作为的近似值具有3位有效数字。

(Ⅲ)**MATLAB源程序：**

clc;clear;

syms x; %创建符号对象x

y3=log((1+x)/(1-x)); %函数y3=ln((1+x)/(1-x))

N=input('请输入N:'); %进行taylor的N阶展开

S=taylor(y3,x,0,'Order',N); %y2在点处，进行N阶泰勒展开

dif=abs(y3-S); %对函数y2和展开式S求差值并取绝对值

data3=zeros(1,10000); %定义数组data存储差值

n=1;

for z=-1/3:0.001:1/3 %计算x在[-0.5,0.5]精度为0.001时的误差

answer=subs( dif,x,z);

answer=single(answer);

data3(n)=answer;

n=n+1;

end

result=max(data3); %求泰勒展开的误差最大值

fprintf('result=%f\n',result); %输出

S3=subs(S,x,1/3); %求在1/3处泰勒展开的值

fprintf('S(1/3)=%f\n',S3); %输出

e3=subs( dif,x,1/3); %求在1/3处误差的大小

fprintf('e1=%f\n',e3); %输出

**运行结果：**

请输入N:10

result=0.000001

S(1/3)=0.693146

e1=0.000001

**即为：**

，，

，作为的近似值具有5位有效数字。

**(Ⅳ)**对于，

对于，

对于，

在中，当时，同时有“大数吃小数”和“相似数相减”；在中，当时，各项均为正数，仅有“大数吃小数”；在中，当时，各项均为正数，“大数吃小数”的程度小于。由运算结果可以看出，的误差最小，的误差最大。因此，在数值分析中，要注意尽量避免“大数吃小数”和“相近数相减”。

**二、第二章**

记



(Ⅰ)分析方程存在几个实根。给出每个根所在的区间。

(Ⅱ)研究用Newton法求方程的根，分析迭代初值在什么范围内选取时，迭代序列收敛到相应的根。用Newton法求出具有8位有效数字的近似根。

(Ⅲ)分析方程存在几个实根。给出每个根所在的区间。

(Ⅳ)研究用Newton法求方程的根，分析迭代初值在什么范围内选取时，迭代序列收敛到相应的根。用Newton法求出具有8位有效数字的近似根。

**解：(Ⅰ)MATLAB源程序：**

clc;clear;

syms x; %创建符号对象x

f=3\*x^2-exp(x); %方程f(x)

g=diff(f); %求导得g(x)

h=0.1; %区间宽度可调整

x0=0;

flag=0;

data=zeros(1,100); %定义0数组data

for i=-100:100

a=subs(g,x,x0+i\*h); %左区间端点值计算

b=subs(g,x,x0+(i+1)\*h); %右区间端点值计算

if(a\*b<0) %若一正一负，则存储数据

flag=flag+1; %标志位+1

data(flag)=i;

else

flag=flag; %若不是有根区间，则不计数

end

end

fprintf('实根个数 为%d\n',flag);

disp('根所在区间为:');

for j=1:flag %标志位控制遍历次数，即区间个数

fprintf('[%.1f,%.1f]\n',x0+data(j)\*h,x0+(data(j)+1)\*h); %输出区间

end

**运行结果：**

实根个数为2

根所在区间为:

[0.2,0.3]

[2.8,2.9]

**即为：**

方程存在2个实根**，**所在区间分别为：[0.2,0.3]，[2.8,2.9]。

(Ⅱ)**MATLAB源程序：**

clc;clear;

x0=input('迭代初值x0=');

t0=ex\_root(x0); %求迭代初值对应的值

h=0.001; %修改计算精度

i=1;j=1;

tmax=t0;tmin=t0;

if(t0<1.5) %t0<1.5则收敛区间左端点

while(ex\_root(tmax)==t0)

tmax=t0+h\*i;

i=i+1;

end

fprintf('g(x)=0的根为%.8f\n',t0);

fprintf('收敛区间为(-inf %.3f)\n',tmax);

else %t0>1.5则收敛区间右端点

while(ex\_root(tmin)==t0)

tmin=t0-h\*j;

j=j+1;

end

fprintf('g(x)=0的根为%.7f\n',t0);

fprintf('收敛区间为(%.3f inf)\n',tmin);

end

**调用求根函数：**

function [ans]=ex\_root(x0)

syms x

f=3\*x^2-exp(x); %方程f(x)

g=diff(f); %求导得g(x)

g1=diff(g); %求导得g1(x)

y=x-g/g1;

t1=x0;

t2=double(subs(y,x,x0));

m=0.5e-8; %有效位设定

while(abs(t2-t1)>m) %循环求解，直到满足有效位要求

t1=t2;

t2=double(subs(y,x,t1));

end

ans=t2; %存储函数计算结果

end

**运行结果：**

迭代初值x0=2.8

g(x)=0的根为2.8331479

收敛区间为(1.791 inf)

迭代初值x0=0.3

g(x)=0的根为0.20448145

收敛区间为(-inf 1.792)

**即为：**

|  |  |
| --- | --- |
| 的根 | 收敛区间 |
| 0.20448145 |  |
| 2.8331479 |  |

(Ⅲ)**MATLAB源程序：**

clc;clear;

syms x; %创建符号对象x

f=@(x)expression; %匿名函数定义

f=3\*x^2-exp(x); %方程f(x)

h=0.1; %调整区间宽度

x0=0;

flag=0;

data=zeros(1,100); %定义0数组data

for i=-100:100

a=subs(f,x,x0+i\*h); %左区间端点值计算

b=subs(f,x,x0+(i+1)\*h); %右区间端点值计算

if(a\*b<0) %若一正一负，则存储数据

flag=flag+1; %标志位+1

data(flag)=i;

else

flag=flag; %若不是有根区间，则不计数

end

end

fprintf('实根个数为%d\n',flag);

disp('根所在区间为:');

for j=1:flag %标志位控制遍历次数，即区间个数

fprintf('[%.1f,%.1f]\n',x0+data(j)\*h,x0+(data(j)+1)\*h); %输出由data推算的区间

end

**运行结果：**

实根个数为3

根所在区间为:

[-0.5,-0.4]

[0.9,1.0]

[3.7,3.8]

**即为：**

方程存在3个实根**，**所在区间分别为：[-0.5,-0.4]，[0.9,1.0]，[3.7,3.8]。

(Ⅳ)**MATLAB源程序：**

clc;clear;

x0=input('迭代初值x0=');

t0=ex\_rof(x0); %求迭代初值对应的值

h=0.001; %修改计算精度

i=1;j=1;

p=1;q=1;

tmax=t0;tmin=t0;

if(t0<0) %区间1内根

while(ex\_rof(tmax)==t0)

tmax=t0+h\*i;

i=i+1;

end

fprintf('f(x)=0的根为%.8f\n',t0);

fprintf('收敛区间为(-inf,%.3f)\n',tmax);

else %区间3内根

if(t0>1)

while(ex\_rof(tmin)==t0)

tmin=t0-h\*j;

j=j+1;

end

fprintf('f(x)=0的根为%.7f\n',t0);

fprintf('收敛区间为(%.3f,inf)\n',tmin);

else %区间2内根

while(ex\_rof(tmin)==t0)

tmin=t0-p\*h;

p=p+1;

end

while(ex\_rof(tmax)==t0)

tmax=t0+q\*h;

q=q+1;

end

fprintf('f(x)=0的根为%.8f\n',t0);

fprintf('收敛区间为(%.3f,%.3f)\n',tmin,tmax);

end

end

**调用求根函数：**

function [ans]=ex\_rof(x0)

syms x

f=3\*x^2-exp(x); %方程f(x)

g=diff(f); %求导得g(x)

y=x-f/g;

t1=x0;

t2=double(subs(y,x,x0));

m=0.5e-8; %有效位设定

while(abs(t2-t1)>m) %循环求解，直到满足有效位要求

t1=t2;

t2=double(subs(y,x,t1));

end

ans=t2; %存储函数计算结果

end

**运行结果：**

迭代初值x0=-0.5

f(x)=0的根为-0.45896227

收敛区间为(-inf,0.205)

迭代初值x0=0.9

f(x)=0的根为0.91000757

收敛区间为(0.310,2.470)

迭代初值x0=3.7

f(x)=0的根为3.7330790

收敛区间为(2.833,inf)

**即为：**

|  |  |
| --- | --- |
| 的根 | 收敛区间 |
| -0.45896227 |  |
| 0.91000757 |  |
| 3.7330790 |  |

**三、第三章**

已知Hilbert矩阵的元素为

(Ⅰ)编程计算的无穷范数的程序。

(Ⅱ)编写计算的无穷范数条件数的程序（可调用求逆函数，在Mathematica中为，在Matlab中为，在其他语言中请自行查找。）

(Ⅲ)对，计算的无穷范数条件数；画出的关系图。

(Ⅳ)令，，对，求解，并计算和以及它们的无穷范数。

(Ⅴ)通过以上的数值实验，你有哪些体会？

**解：(Ⅰ)MATLAB源程序：**

clc;clear;

n=input('矩阵阶数n='); %自定义矩阵阶数

H=zeros(n,n); %0矩阵数组

for i=1:n %计算每个hij

for j=1:n

H(i,j)=1/(i+j-1);

end

end

S=zeros(n,1); %定义0数组S存储行和

for m=1:n

S(m,1)=0;

end

for i=1:n

for j=1:n

S(i,1)=S(i,1)+abs(H(i,j));

end

end

A=max(S); %S的最大值即为该阶无穷范数

fprintf('%d阶Hilbert矩阵的无穷范数为%.8f\n',n,A);

**运行结果：**

矩阵阶数n=6

6阶Hilbert矩阵的无穷范数为2.45000000

**即为：**

对于的无穷范数，当n=6时，。

**(**Ⅱ**)MATLAB源程序：**

clc;clear;

n=input('矩阵阶数n='); %自定义矩阵阶数

H=zeros(n,n); %0矩阵数组

for i=1:n %计算每个hij

for j=1:n

H(i,j)=1/(i+j-1);

end

end

H1=inv(H); %矩阵H求逆

S=zeros(n,1);

S1=zeros(n,1);

for m=1:n

S(m,1)=0;

S1(m,1)=0;

end

for i=1:n

for j=1:n

S(i,1)=S(i,1)+abs(H(i,j)); %存储H行和

S1(i,1)=S1(i,1)+abs(H1(i,j)); %存储H逆H1行和

end

end

A=max(S)\*max(S1); %求cond

fprintf('%d阶Hilbert矩阵的无穷范数条件数为%f\n',n,A);

**运行结果：**

矩阵阶数n=6

6阶Hilbert矩阵的无穷范数条件数为29070279.001185

**即为：**

对于的无穷范数条件数，当n=6时，。

**(**Ⅲ**)MATLAB源程序：**

clc;clear;

n=zeros(10,1);

x=zeros(10,1);

y=zeros(10,1);

for i=1:10

n(i,1)=10\*i; %求10,20,…,100的cond

x(i,1)=inf\_cond(n(i,1));

y(i,1)=log(x(i,1));

end

plot(x, y, 'r+'); %画图

grid on;

**调用无穷范数条件函数：**

function answer=inf\_cond(n)

H=zeros(n,n); %0矩阵数组

for i=1:n %计算每个hij

for j=1:n

H(i,j)=1/(i+j-1);

end

end

H1=inv(H); %矩阵H求逆

S=zeros(n,1);

S1=zeros(n,1);

for m=1:n

S(m,1)=0;

S1(m,1)=0;

end

for i=1:n

for j=1:n

S(i,1)=S(i,1)+abs(H(i,j)); %存储H行和

S1(i,1)=S1(i,1)+abs(H1(i,j)); %存储H逆H1行和

end

end

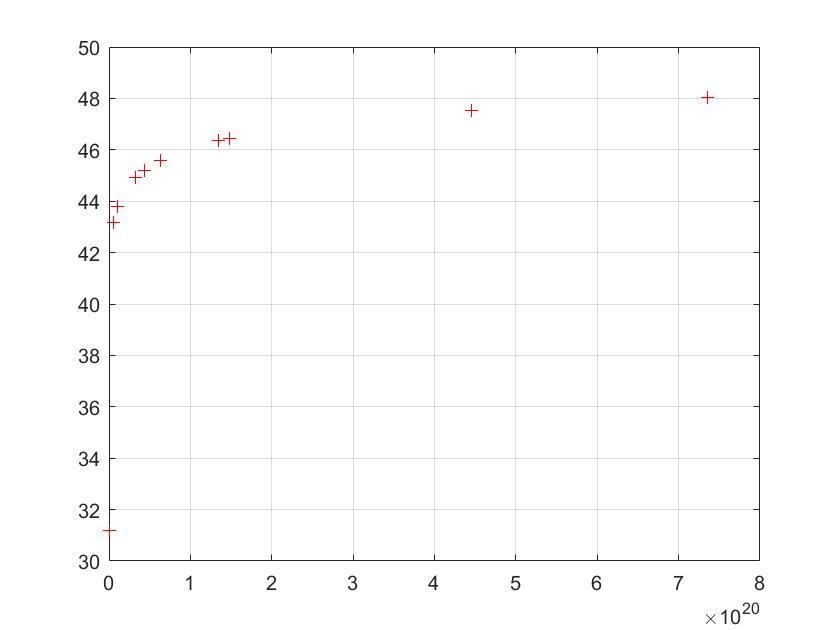
answer=max(S)\*max(S1); %求cond

end

**运行结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| n |  | n |  |
| 10 | 31.1964 | 60 | 45.2136 |
| 20 | 43.7960 | 70 | 47.5446 |
| 30 | 43.1681 | 80 | 46.4422 |
| 40 | 46.3531 | 90 | 48.0468 |
| 50 | 44.9171 | 100 | 49.0898 |

的关系图如下：



**(**Ⅳ**)MATLAB源程序：**

clc;clear;

n=input('矩阵阶数n='); %自定义矩阵阶数

H=zeros(n,n); %0矩阵数组

for i=1:n %计算每个hij

for j=1:n

H(i,j)=1/(i+j-1);

end

end

x=zeros(n,1);

for m=1:n

x(m,1)=1;

end

b=H\*x; %求解

x0=H\b;

c1=x-x0;

c2=b-H\*x0;

c11=max(abs(c1));

c22=max(abs(c2));

**运行结果：**

和的值存储于工作区c11和c22。由于是向量，这里就不予列出，仅列出其无穷范数。

|  |  |  |
| --- | --- | --- |
| n |  |  |
| 10 |  |  |
| 50 |  |  |
| 100 |  |  |

**(Ⅴ)**对于高阶Hilbert矩阵而言，其无穷范数条件数远大于1，是病态方程组。由第(Ⅳ)问可知，无论多少阶数的Hilbert矩阵，尽管都很小，但随着Hilbert矩阵阶数增加，其无穷范数条件数越来越大，解向量的相对误差仍会变得非常大。由此，用余量的大小来检验近似解的准确程度仅对良态方程组适用，对于病态方程组是不可靠的。