Numerical Analysis Homework5

Zhang Jiyao,PB20000204

2023 年 4 月 15 日

1 Introduction

问题1

分别编写一个用复化$Simpson$积分公式和复化梯形积分公式计算积分的通用程序,并且利用上述的程序计算积分

$$I(f) = \int_0^4 sin(x)dx$$

取节点$x_i, i = 0, ..., N$,其中$N = 2^k, k = 1, ..., 12$.并分析误差,计算算法的收敛阶.

$$Ord = \frac{ln(Error_{ord}/Error_{now})}{ln(N_{now}/N_{old})}$$

问题2

对函数

$$f(x) = \frac{1}{1 + 25x^2}, x \in [-1, 1]$$

构造Lagrange插值多项式$p_L(x)$,插值节点取为:

$$1.x_i = 1 - \frac{2}{N}i, i = 0, 1, ..., N$$

$$2.x_i = -cos(\frac{i+1}{N+2}\pi), i = 0, 1, ..., N$$

并且利用$\int_{-1}^{1} p_L(x)dx$计算积分$\int_{-1}^{1} f(x)dx$的近似值,计算如下误差

$$|\int_{-1}^{1} p_L(x)dx - \int_{-1}^{1} f(x)dx|$$

并且对$N = 5, 10, 15, 20, 25, 30, 35, 40$比较以上两组节点的结果.

2 Method

对问题1,考虑对应的复化梯形公式和复化$Simpson$公式,直接代入计算即可.

复化梯形公式,对任意的区间$[a, b]$,以及节点x_k,我们有

$$T_n = \frac{b-a}{2n}[f(a) + 2\sum_{k=1}^{n-1} f(x_k) + f(b)]$$

复化$Simpson$公式,对任意的区间$[a,b]$,以及节点x_k,我们有

$$S_n = \frac{b-a}{6n}[f(a) + 4\sum_{k=1}^{n-1} f(x_{k+\frac{1}{2}}) + 2\sum_{k=1}^{n-1} f(x_k) + f(b)]$$

对于问题2,考虑对应的$Lagrange$插值函数,定义

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

则在以上结点插值f的次数最多为n的多项式为

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

则我们就近似的以

$$\int_a^b p(x)dx$$

作为f的积分值

3 Results

第一个问题的结果如下图所示

N	复化Simpson error	order	复化梯形error	order
2	0.2666	0	0.5918	0
4	0.0104	4.6789	0.1402	2.0782
8	0.00059	4.1367	0.0346	2.0184
16	3.6155e-05	4.0327	0.0086	2.0045
32	2.2470e-06	4.0081	0.0022	2.0011
64	1.4025e-07	4.0020	0.0005	2.0003
128	8.7623e-09	4.0005	0.0001	2.0000
256	5.4759e-10	4.0001	3.3643e-05	2.0000
512	3.4224e-11	4.0000	8.4109e-06	2.0000
1024	2.1388e-12	4.0001	2.1027e-06	2.0000
2048	1.3400e-13	3.9965	5.2568e-07	2.0000
4096	8.1046e-15	4.0474	1.3142e-07	2.0000

第二个问题,若选用第一组节点如下图所示

N	$\int_{-1}^{1} p_L(x)dx$	$\int_{-1}^{1} f(x)dx$	$\lvert \int_{-1}^{1} p_L(x)dx - \int_{-1}^{1} f(x)dx \rvert$
5	0.4615	0.54936	0.0878
10	0.9347	0.54936	0.3853
15	0.8311	0.54936	0.2818
20	-5.3699	0.54936	5.9193
25	-5.3999	0.54936	5.9492
30	153.7979	0.54936	153.2485
35	173.8803	0.54936	173.3309
40	-4912.4114	0.54936	4.9130e+03

若选用第二组节点则如下图所示

N	$\int_{-1}^{1} p_L(x)dx$	$\int_{-1}^{1} f(x)dx$	$\lvert \int_{-1}^{1} p_L(x)dx - \int_{-1}^{1} f(x)dx \rvert$
5	0.4811	0.54936	0.0682
10	0.5541	0.54936	0.0047
15	0.5476	0.54936	0.0018
20	0.5500	0.54936	6.5049e-04
25	0.5494	0.54936	3.9432e-07
30	0.5493	0.54936	3.8668e-05
35	0.5494	0.54936	3.2346e-06
40	0.5494	0.54936	4.5084e-06

4 Discussion

关于第一个问题,可以看出复化Simpson公式计算出的值无论是误差还是收敛阶数都要优于复化梯形公式。我们可以认为复化Simpson公式是一个更优良的算法。

关于第二个问题,可以直接看出第二组节点是远远优于第一组节点的。回忆之前的作业,采用第一组这种等距节点进行插值,往往就可能会在某些点处产生较大的误差。而采用第二组节点这种得到的误差就很小,基本能得到近似的积分值。

5 Computer Code

```
function y=f(x)
y=sin(x);
end

T1=zeros(12,1);
Sn1=zeros(12,1);
error1=zeros(12,1);
error2=zeros(12,1);
```

```
order1=zeros(12,1);
order2=zeros(12,1);
f=@(x)sin(x);

for k=1:12
    N=2^k;
    x=linspace(0,4,N+1);
    T1(k)=T(f,0,4,2^k);

    Sn1(k)=Sn(f,x);
    error1(k)=abs(T1(k)-1+cos(4));
    error2(k)=abs(Sn1(k)-1+cos(4));
    if(k>1)
      order1(k)=log(error1(k-1)/error1(k))/log(2);
      order2(k)=log(error2(k-1)/error2(k))/log(2);
    end

end

for k=1:12
    t=2^k;
    disp(["N is:" t]);
    disp(["error of trapezium error is:" error1(k)]);
    disp(["order of trapezium error is:" order1(k)]);
    disp(["error of Simpson error is:" error2(k)]);
    disp(["order of Simpson error is:" order2(k)]);
end

syms x;
int1=zeros(8,1);
int2=zeros(8,1);
error1=zeros(8,1);
error2=zeros(8,1);
for k=1:8
    N=5*k;
    sym1=0;
    sym2=0;
    xi1=zeros(N+1,1);
    xi2=zeros(N+1,1);
    yi1=zeros(N+1,1);
    yi2=zeros(N+1,1);
```

```
poly1=zeros(N+1,1);
poly2=zeros(N+1,1);

for j=0:N
  xi1(j+1)=1-2*j/N;
  xi2(j+1)=-cos(pi*(j+1)/(N+2));
  yi1(j+1)=1/(1+25*xi1(j+1)^2);
  yi2(j+1)=1/(1+25*xi2(j+1)^2);
end

for i=1:N+1
 t=1;
 q=1;
for p=1:N+1
    if p~=i
        t=t*(x-xi1(p))/(xi1(i)-xi1(p));
        q=q*(x-xi2(p))/(xi2(i)-xi2(p));
    end
end



g = matlabFunction(t);
ab= matlabFunction(q);


sum11 = 0;
sum12 = 0;
sum21 = 0;
sum22 = 0;
a=-1;
b=1;
h=0.002;
for c = 0:999
    z=a+h*(c+1/2);
    sum11 = sum11 + g(z);
    sum12 = sum12 + ab(z);
end
for j = 1:999
    y=a+h*j;
   sum21 = sum21 + g(y);
```

```
        sum22 = sum22 + ab(y);
    end
     poly1(i) = h/6*(g(a)+4*sum11+2*sum21+g(b));
     poly2(i) = h/6*(ab(a)+4*sum12+2*sum22+ab(b));

     sym1=sym1+poly1(i)*yi1(i);
     sym2=sym2+poly2(i)*yi2(i);

     end

     int1(k)=sym1;
     int2(k)=sym2;
     error1(k)=abs(int1(k)-0.4*atan(5));
     error2(k)=abs(int2(k)-0.4*atan(5));


     disp(["N is:" N]);
     disp(["integral of first nodes is:" int1(k)]);
     disp(["integral of second nodes is:" int2(k)]);
     disp(["integral of the function is:" 0.4*atan(5)]);
     disp(["error of first nodes is:" error1(k)]);
     disp(["error of second nodes is:" error2(k)]);


end



function int=Sn(f,x)
N=size(x,2);
N=N-1;
h=x(2)-x(1);
x2=x(2:2:N); %系数为4
x3=x(3:2:N-1); %系数为2
int=h*(f(x(1))+f(x(N+1))+2*sum(f(x3))+4*sum(f(x2)))/3;

function Tn=T(f,a,b,n)
    h=(b-a)/n;
    sum=0;
    for k=1:n-1
        x=a+h*k;
        sum=sum+feval('f',x);
```

```
    end

    format long

    Tn=(feval('f',a)+2*sum+feval('f',b))*h/2;
end
```