Numerical Analysis Homework3

Zhang Jiyao,PB20000204

2023 年 3 月 23 日

1 Introduction

对函数

$$f(x) = e^x, x \in [0.1]$$

构造等距节点的样条插值函数,对以下两种类型的样条函数

(1):一次分片线性样条

(2):满足$S'(0) = 1$,$S'(1) = e$的三次样条

并计算如下误差

$$max_i\{\left|f(x_{i-\frac{1}{2}}) - S(x_{i-\frac{1}{2}})\right|, i = 1, ..., N\}$$

说明:其中$x_{i-\frac{1}{2}}$是每个小区间的中点.对$N = 5, 10, 20, 40$比较以上两组节点的结果。并且计算该算法的收敛阶,公式如下

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

2 Method

先讨论第一种一次分片线性样条的情况,该情况下比较简单。因为构造的一次分片线性样条在每个区间上面都是线性函数,那么只需要根据每个区间两侧端点的值,构造对应区间上的线性函数即可。为了计算一点的值,只需要利用以下公式即可

$$y = \frac{x - x_0(i+1)}{x_0(i) - x_0(i+1)} y_0(i) + \frac{x - x_0(i)}{x_0(i+1) - x_0(i)} y_0(i+1)$$

说明:x是待求节点,y是待求值,$x_0(i)$是第i个区间的左端点,$y_0(i)$是其对应的值

接下来讨论三次样条的情况。因为三次样条本身已经确定了$4n - 2$个自由度,只需要给出边界的两个自由度。而本题中$S'(0) = 1$,$S'(1) = e$满足要求。记$M_i = S''(x_i)$,则我们可以给出M_i满足的线性方程组

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & h_2 & u_3 & h_3 & \\ & ... & ... & & \\ & h_{n-3} & u_{n-2} & h_{n-2} \\ & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ ... \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ ... \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{1}$$

然后我们采用追赶法解这个线性方程组。

其中$h_i = x_{i+1} - x_i$,$u_i = 2(h_i + h_{i+1})$,$b_i = 6\frac{y_{i+1}-y_i}{h_i}$,$v_i = b_i - b_{i-1}$,再结合边界条件$S'(0) = 1$,$S'(1) = e$即可得出结论。

接下来讨论计算样条函数在某一确定点处的值。先确定x是在哪个区间中,比如说是$[x_i, x_{i+1}]$中,那么重写$S_i(x)$的表达式为

$$S_i(x) = y_i + (x - t_i)[C_i + (x - t_i)[B_i + (x - t_i)A_i]]$$

其中,$A_i = \frac{M_{i+1}-M_i}{6h_i}$,$B_i = \frac{M_i}{2}$,$C_i = -\frac{h_i M_{i+1}}{6} - \frac{h_i M_i}{3} + \frac{y_{i+1}-y_i}{h_i}$

3 Results

n	Method (1) error	order	Method (2) error	order
5	0.0123	——	1.0907e-05	——
10	0.0032	1.9288	6.9559e-07	3.9709
20	8.2853e-04	1.9642	4.3871e-08	3.9869
40	2.0973e-04	1.9820	2.7538e-09	3.9938

4 Discussion

观察数据可知，两种方法都有随着N越来越大,误差减小，收敛阶增大的趋势。

首先单独讨论。第一种方法的误差相对较大,在$N = 5$时误差已经达到了0.01,可见分片线性样条并不是一个优秀的插值函数。并且观察到N从20变到40时,误差反而增大,这说明这种算法不够稳定,可能在某些节点处会有较大的误差。

相比之下，第二种方法的误差就小了很多,误差最大也不过$1e - 5$量级.并且误差随着N的增大越来越小，较为稳定。

两者之间对比的话，无论误差还是收敛阶,第二种方法都完胜第一种。综上我们可以认为三次样条插值是更好的算法。

5 Computer Code

```
N=6;
xx=zeros(N,1);
```

```
yy=zeros(N,1);
y0=zeros(N-1,1);
error=zeros(N-1,1);

for i=1:N
    xx(i)=(i-1)/(N-1);
    yy(i)=exp(xx(i));
end

for i=1:N-1
    y0(i)=sol_linear(xx,yy,(xx(i)+xx(i+1))/2,N);
end

for i=1:N-1
    error(i)=abs(y0(i)-exp((xx(i)+xx(i+1))/2));
end


ex=zeros(N-1,1);


for i=1:N-1
    ex(i)=abs(cubicspline(xx,yy,(xx(i)+xx(i+1))/2)-exp((xx(i)+xx(i+1))/2));
end

t1=max(error);
t_1=max(ex);




N=11;
xx=zeros(N,1);
yy=zeros(N,1);
y0=zeros(N-1,1);
error=zeros(N-1,1);

for i=1:N
    xx(i)=(i-1)/(N-1);
    yy(i)=exp(xx(i));
end
```

```
for i=1:N-1
    y0(i)=sol_linear(xx,yy,(xx(i)+xx(i+1))/2,N);
end

for i=1:N-1
    error(i)=abs(y0(i)-exp((xx(i)+xx(i+1))/2));
end



ex=zeros(N-1,1);

for i=1:N-1
    ex(i)=abs(cubicspline(xx,yy,(xx(i)+xx(i+1))/2)-exp((xx(i)+xx(i+1))/2));
end

t2=max(error);
t_2=max(ex);

ord11=log(t1/t2)/log(10/5);
ord21=log(t_1/t_2)/log(10/5);



N=21;
xx=zeros(N,1);
yy=zeros(N,1);
y0=zeros(N-1,1);
error=zeros(N-1,1);

for i=1:N
    xx(i)=(i-1)/(N-1);
    yy(i)=exp(xx(i));
end

for i=1:N-1
    y0(i)=sol_linear(xx,yy,(xx(i)+xx(i+1))/2,N);
end

for i=1:N-1
    error(i)=abs(y0(i)-exp((xx(i)+xx(i+1))/2));
end
```

```
ex=zeros(N-1,1);

for i=1:N-1
    ex(i)=abs(cubicspline(xx,yy,(xx(i)+xx(i+1))/2)-exp((xx(i)+xx(i+1))/2));
end

t3=max(error);
t_3=max(ex);

ord12=log(t2/t3)/log(20/10);
ord22=log(t_2/t_3)/log(20/10);

N=41;
xx=zeros(N,1);
yy=zeros(N,1);
y0=zeros(N-1,1);
error=zeros(N-1,1);

for i=1:N
    xx(i)=(i-1)/(N-1);
    yy(i)=exp(xx(i));
end

for i=1:N-1
    y0(i)=sol_linear(xx,yy,(xx(i)+xx(i+1))/2,N);
end

for i=1:N-1
    error(i)=abs(y0(i)-exp((xx(i)+xx(i+1))/2));
end


ex=zeros(N-1,1);

for i=1:N-1
    ex(i)=abs(cubicspline(xx,yy,(xx(i)+xx(i+1))/2)-exp((xx(i)+xx(i+1))/2));
end


t4=max(error);
t_4=max(ex);
```

```
ord13=log(t3/t4)/log(40/20);
ord23=log(t_3/t_4)/log(40/20);

disp('N=5');
disp(["Max Error of Method (1) :"  t1] );
disp(["Max Error of Method (2) :"  t_1 ]);

disp('N=10');
disp(["Max Error of Method (1) :"  t2] );
disp(["Max Error of Method (2) :"  t_2 ]);
disp(["order of Method (1) :" ord11]);
disp(["order of Method (2) :" ord21]);

disp('N=20');
disp(["Max Error of Method (1) :"  t3] );
disp(["Max Error of Method (2) :"  t_3] );
disp(["order of Method (1) :" ord12]);
disp(["order of Method (2) :" ord22]);

disp('N=40');
disp(["Max Error of Method (1) :"  t4] );
disp(["Max Error of Method (2) :"  t_4 ]);
disp(["order of Method (1) :" ord13]);
disp(["order of Method (2) :" ord23]);




function y= sol_linear(x0,y0,x,n)

ss=0;

   for i=1:n-1

    if(x>=x0(i))&&(x<=x0(i+1))

        ss=(x-x0(i+1))/(x0(i)-x0(i+1))*y0(i)+(x-x0(i))/(x0(i+1)-x0(i))*y0(i+1);
    else
        continue;
    end

   end

y=ss;

end




function Sx = cubicspline(x,y,x0)

n=length(x);


for k=2:n
    h(k)=x(k)-x(k-1);
end

for i=2:n-1
    lambda(i)=h(i+1)/(h(i)+h(i+1));
    mu(i)=1-lambda(i);
    d(i)=6/(h(i)+h(i+1))*((y(i+1)-y(i))/h(i+1)-(y(i)-y(i-1))/h(i));
end


A=zeros(n,n);
A(1,1)=2;
A(n,n)=2;
for k=2:n-1
    A(k,k)=2;
    A(k,k+1)=lambda(k);
    A(k,k-1)=mu(k);
end
b=zeros(n,1);


    dy1_0 = 1;
    dy1_n = exp(1);
   A(1,2)=1;
    A(n,n-1)=1;
    b(1)=6/h(2)*((y(2)-y(1))/h(2)-dy1_0);
    b(n)=6/h(n)*(dy1_n-(y(n)-y(n-1))/h(n));


for k=2:(n-1)
    b(k,1)=d(k);
end



M = Chase_method(A,b);

for i=1:n-1
    if (x0<x(i+1))&&(x0>=x(i))

    Sx = (((x(i+1)-x0)^3)*M(i)+((x0-x(i))^3)*M(i+1))/(6*h(i+1)) + ...
    (y(i)-M(i)*(h(i+1)^2)/6)*(x(i+1)-x0)/h(i+1) + (y(i+1)-M(i+1)*(h(i+1)^2)/6)*(x0-x(i))/h(i+1);

end
end

function [ x ] = Chase_method( A, b )



T = A;
for i = 2 : size(T,1)
    T(i,i-1) = T(i,i-1)/T(i-1,i-1);
    T(i,i) = T(i,i) - T(i-1,i) * T(i,i-1);
end
L = zeros(size(T));
L(logical(eye(size(T)))) = 1;
for i = 2:size(T,1)
    for j = i-1:size(T,1)
        L(i,j) = T(i,j);
        break;
    end
end
U = zeros(size(T));
U(logical(eye(size(T)))) = T(logical(eye(size(T))));
for i = 1:size(T,1)
```

```
    for j = i+1:size(T,1)
        U(i,j) = T(i,j);
        break;
    end
end

y = L\b;
x = U\y;
end
```