第四次作业

李阳 515072910019 黄弘毅 515072910038 王潇卫 515072910032

1

First, we need write a function to realize the Lagrange interpolation.

Code lagran.m

```
function C=lagran(X,Y)
%Input: X and Y are corresponding dataset.
%Output:
% C is a vector that contains the Lagrang coefficients from n-1 to zero
order.
w=length(X);%the number of data points.
n=w-1;% the max order of polynomial.
L=zeros(w,w);
%Form the Lagrange coefficient polynomials
for i=1:n+1
   V=1;
   for j=1:n+1
      if i~=j
         V=conv(V,poly(X(j)))/(X(i)-X(j));
      end
   end
   L(i,:)=V;%V is a vector
end
%Determine the coefficients of the Lagrange interpolator
C=Y*L;% the coefficients of simplifed polynomials.
```

Now, let me explain the for-loop in the code. According to theory, $L_i(x) = \prod_{k=1,k\neq i}^{n+1} \frac{x-x_i}{x_i-x_k}$ $P(x) = \sum_{i=1}^{n+1} f_i l_i(x)$ 【conv(u,v)】means polynomials multiply polynomials. for example,if we want to calculate $(s^2+2s+2)*(s+4)*(s+1)$

```
>> w=conv([1, 2, 2], conv([1, 4], [1, 1]))

w =

     1     7    16    18     8

>> P=poly2str(w,'s')

P =

    's^4 + 7 s^3 + 16 s^2 + 18 s + 8'

>>
```

So in the code, for each i,

L(i,:)=V means the coefficients of L_i from high order to low order.

Now, let's solve this question;

```
%% (1)
clear all;clc;
X=[1,2,2.5];
Y=X+2./X;
C=lagran(X,Y);
f11=polyval(C,1.5);
f12=polyval(C,1.2);

%% (2)
X=[0.5 1 2 2.5];
Y=X+2./X;
C=lagran(X,Y);
f21=polyval(C,1.5);
f22=polyval(C,1.2);

f1=1.5+2/(1.5);
r11=(f11-f1)/f1;
r21=(f21-f1)/f1;

f2=1.2+2/(1.2);
r12=(f12-f2)/f2;
r22=(f22-f2)/f2;
```

\begin{array}{c|lcr} x & \text{f(x)} & \text{quadratic} & \text{cubic} & \text{rrq} &\text{rrc} \ \hline 1.5 & 2.8333 & 2.9000 & 2.7000 & 0.0235 & -0.0471\ 1.2 & 2.8667 & 2.9360 & 2.7696 & 0.0242 & -0.0339\ \end{array} 'rrq' means relative error of quadratic; 'rrc' means relative error of cubic; So we can see that in this question, quadratic Lagrange interpolation has a better approximation. I think this results from the function $f(x) = x + \frac{2}{x}$, for the interval between[1.2,1.5], it seems like quadratic function than the cubic function.

2

(a)

according to the formula of up-limit of error $|\Delta f(x)| \leq \frac{\gamma}{4(n+1)} h^{n+1}$ $\gamma = max|f^{n+1}(x)|$ $h = max(x_{i+1} - x_i)$ here $n = 1, \gamma = max(\sin(x) \leq \sin(1))$ we calculate $h = 0.00218$ Since it's a sufficient condition, larger h may also allow $|\Delta f(x)|$ to smaller than the requirement, we decide to search for h in a more accurate way:

Since in each segment we have $|\Delta f(x)| \leq \frac{\gamma}{8} h^2$, where the second derivative of $\sin(x)$ is $-\sin(x)$, it's reasonable to **assume that largest error always occur at segment [1-h,1]**, so we firstly set h, and then calculate 1000 points in segment [1-h,1] both for $\sin(x)$ and Lagrange interpolation and find the largest error among them. If the largest error is still smaller than the requirement, we increase h further, until some h result in a out-of-requirement error(here we just increase h by hand for simplicity, without any iteration method).

Code 1: for Lagrange interpolation

```
function yi = LagrangeInterp(x,y,xi)
n = length(x);
L = zeros(1,n);
for i = 1:n,
    L(i) = 1;
    for j = 1:n,
        if j ~= i,
            L(i) = L(i)* (xi - x(j))/(x(i) - x(j));
        end
    end
end
yi = sum(y.* L);
```

Code 2: search for a more accurate h

```
X=[1-h,1];
Y=[sin(1-h),sin(1)];
Err=zeros(1,1000);
for i=1:1000
    Err(i)=sin(1-h+i/1000*h)-LagrangeInterp(X,Y,1-h+i/1000*h);
end
Errmax=max(abs(Err))
```

And we find h is in **somewhere between 0.002181 and 0.002182** (the Errmax of 0.002181 is 4.9998e-07, and of 0.002182 is 5.0044e-07). So it seems that the estimation from the formula is already good in this case.

(b)

In this case $n = 2, \gamma = max(\cos(x)) \leq 1$, we calculate $h = 0.018171$ Again we apply the same method to search for a more accurate h(only with difference that we assume the largest error always occur at [0,2h])

```
%Lagrange Interpolation of sin(x) in [0,1] to 2th order
h=0.0198;
X=[0,h,2*h];
Y=[0,sin(h),sin(2*h)];
Err=zeros(1,1000);
for i=1:1000
    Err(i)=sin(i/1000*2*h)-LagrangeInterp(X,Y,i/1000*2*h);
end
Errmax=max(abs(Err))
```

And we find h is in **somewhere between 0.0198 and 0.0199** (the Errmax of 0.0198 is 4.9787e-07, and of 0.0199 is 5.0546e-07). In this case the accurate h is more larger than estimation.

(c)

In this case $n = 3, \gamma = max(\sin(x)) \leq \sin(1)$, we calculate $h = 0.05528$ Again we apply the same method to search for a more accurate h(only with difference that we assume the largest error always occur at [1-3h,1])

```
%Lagrange Interpolation of sin(x) in [0,1] to 3th order
h=0.062;
X=[1-3*h,1-2*h,1-h,1];
Y=[sin(1-3*h),sin(1-2*h),sin(1-h),sin(1)];
Err=zeros(1,1000);
for i=1:1000
    Err(i)=sin(1-3*h+i/1000*3*h)-LagrangeInterp(X,Y,1-3*h+i/1000*3*h);
```

And we find h is in **somewhere between 0.062 and 0.063** (the Errmax of 0.062 is 4.8997e-07, and of 0.063 is 5.2184e-07). In this case the accurate h is more larger than estimation.

3

We write a general code applying Newton divided difference method.

```
%Newton divided difference n order
%X, Y should be row
function [P,D]=Newtondivdif(X,Y,x)
n=length(X)-1;
D=zeros(n+1,n+1);
D(:,1)=Y';
for i=1:n
    for k=i:n
        D(k+1,i+1)=(D(k,i)-D(k+1,i))/(X(k-i+1)-X(k+1));
    end
end
XX=ones(1,n);
for p=1:n
    for q=p:n
        XX(1,q)=XX(1,q)*(x-X(p));
    end
end

P=D(1,1);
for j=1:n
    P=P+D(j+1,j+1)*XX(j);
end
```

Divided difference table:

0.6900	0	0	0	0
1.1000	0.4100	0	0	0
1.3900	0.2900	-0.0600	0	0
1.6100	0.2200	-0.0350	0.0083	0
1.9500	0.1700	-0.0167	0.0046	-0.0008

$P_4(2.4) = 1.2272$ $P_4(4.2) = 1.6487$

4

（a）

First, we use Lagrange polynomial interpolation to find I(1), the function has given before.

```
L = zeros(1,n);
for i = 1:n,
    L(i) = 1;
    for j = 1:n,
        if j ~= i,
            L(i) = L(i)* (xi - x(j))/(x(i) - x(j));
        end
    end
end
yi = sum(y.* L);
```

Through Lagrange polynomial interpolation we find I(2)= 1.571274720000000

Then, we use Newton Interpolating Polynomials and we also find I(2)=1.571274720000000

```
%Newton divided difference n order
%X, Y should be row
function [P,D]=Newtondivdif(X,Y,x)
n=length(X)-1;
D=zeros(n+1,n+1);
D(:,1)=Y';
for i=1:n
    for k=i:n
        D(k+1,i+1)=(D(k,i)-D(k+1,i))/(X(k-i+1)-X(k+1));
    end
end
XX=ones(1,n);
for p=1:n
    for q=p:n
        XX(1,q)=XX(1,q)*(x-X(p));
    end
end
```

(b)

(b)

From the table we know that I = 1.58 lies between $\alpha = [5, 10]$, so we set the step size = 0.01, error = 1e-5 when we find $abs(y(\alpha) - 1.58) \leq 1e - 5$, we return α and I

```
% 计算物理 第四次作业 5（b）
alpha = [0,5,10,15,20,25];
I = [1.57080,1.57379,1.58284,1.59814,1.62003,1.64900];

error = 1e-5;
```

```
        a
        yi
        break
    end
end
```

Result:

alpha = 8.750000000000000 I = 1.580001547851562