SI 211: Numerical Analysis

Homework 2

朱佳会\_hw2\_2018233141

1. Solution:

Given f (0) = 1, f (1) = 3, and f (2) = 19, so the three points are:

, , 

Construct a lagrange polynomial to pass through these three points. The corresponding lagrange polynomials are:







Thus, 





The form p(x) = a0 + a1x + a2x2 such that p interpolates f at x ∈ {0, 1, 2}. So, the results are 

1. Solution:

Given f (0, 0) = 1, f (0, 1) = 3, f (0, 2) = 19, f (1, 0) = 3, f (2, 0) = 19, f (1, 1) = 0, so the six points are:

, , , , , 

Use the Newton’s polynomials:





a)When , we have three points , , .





So, we can interpolate the  at (0,1), (1,3), (2,19). It’s easy to use the divided differences to get that:



b) When , we have two points , .





Due to , we need to calculate the  at .



So, we can interpolate the  at (0,0), (1, -3). It’s easy to get that:



c) When , we have one point .





In a similar way as above, we can calculate the  at .



So, we can interpolate the .

Finally, we get the 



Thus, 

1. Solution:
2. The plot of the functionand its interpolating polynomial are shown in Figure 1.

For the functionall derivatives are uniformly bounded by 1 on the interval .Thus, we have



So the approximation error is 2.50521083854×103 and we also can see the error in Figure 2 which is approximately equal to 6.5×10-3.

|  |
| --- |
| Figure_3.1 |
| Figure 1 |
| Figure_3.1.1 |
| Figure 2 |

1. The plot of the function  and its interpolating polynomial are shown in Figure 3. The approximation error is 1.9 as shown in Figure 4.

|  |
| --- |
| Figure_3.2 |
| Figure 3 |
| Figure_3.2.1 |
| Figure 4 |

Here is the code of Julia.

julia> mutable struct DDTable

x

d

end

function addData!(T,x,y)

n = length(T.x);

append!(T.x,x);

append!(T.d,y);

for i=1:n

nom = T.d[end]-T.d[end-n];

den = T.x[end]-T.x[end-i];

append!(T.d,nom/den);

end

end

T = DDTable(zeros(0),zeros(0));

function getNewtonCoefficients(T)

n = length(T.x);

c = zeros(0);

k = 0;

for i=1:n

k+=i;

append!(c,T.d[k]);

end

return c;

end

function getPolynomial(T)

n = length(T.x);

c = getNewtonCoefficients(T);

return function p(x)

b = c[end];

for i=1:n-1

b = c[n-i]+b\*(x-T.x[n-i]);

end

return b;

end

end

p = getPolynomial(T)

function interpolate(range,f)

table = DDTable(zeros(0),zeros(0));

for i=1:length(range)

addData!(table,range[i],f(range[i]));

end

return getPolynomial(table);

end

function f(x)

return sin(x);

end

range = [-5,-4,-3,-2,-1,0,1,2,3,4,5]

p = interpolate(range,f);

x=(-6:0.01:6);

y1=zeros(0);

y2=zeros(0);

for i=1:length(x)

append!(y1,f(x[i]));

append!(y2,p(x[i]));

end

using PyPlot

plot(x,y1,"b--")

plot(x,y2,"r-.")

x=(-5:0.01:5);

y1=zeros(0);

y2=zeros(0);

y=zeros(0);

for i=1:length(x)

append!(y1,f(x[i]));

append!(y2,p(x[i]));

append!(y,abs(f(x[i])-p(x[i])));

end

using PyPlot

plot(x,y,"b")

function g(x)

return 1/(1+x\*x);

end

range = [-5,-4,-3,-2,-1,0,1,2,3,4,5]

p = interpolate(range,g);

x=(-5:0.01:5);

y1=zeros(0);

y2=zeros(0);

y=zeros(0);

for i=1:length(x)

append!(y1,g(x[i]));

append!(y2,p(x[i]));

append!(y,abs(g(x[i])-p(x[i])));

end

using PyPlot

plot(x,y1,"b--")

plot(x,y2,"r-.")

plot(x,y,"b")