SI 211: Numerical Analysis

Homework 3

朱佳会_hw3_2018233141

1. Solution:

According to the function of the natural cubic splines,

$$p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i \in \{1, \ldots, n\}$$

It's easy to return a natural spline that interpolates the function $f(x) = \frac{1}{1+x^2}$.

Here is the code in Julia.

```
julia> function solveC(h,r)
          alpha = [4.0]
          gamma = zeros(0)
          n = length(r)
          for i=1:n-1
            append!(gamma,1.0/alpha[i])
            append!(alpha,4.0-gamma[i])
          end
          d = zeros(n)
          d[1] = r[1]
          for i=2:n
            d[i] = r[i]-gamma[i-1]*d[i-1]
          end
          c = zeros(n)
          c[n] = d[n]/alpha[n]
          for i=1:n-1
            c[n-i] = (d[n-i]-c[n-i+1])/alpha[n-i]
          end
          return c/h
        end

      function spline(f,x0,xN,N)
          h = (xN-x0)/N
          a = zeros(N)
          x = zeros(N)
          a0=f(x0)
          for i=1:N
            x[i] = x0+i*h
            a[i] = f(x[i])
          end
          r = zeros(N)
          r[1] = 3.0/h*(a0-2.0*a[1]+a[2])
          for i=2:N-1
            r[i] = 3.0/h*(a[i-1]-2.0*a[i]+a[i+1])
```

```
    end
    c = solveC(h,r)
    d = zeros(N)
    d[1] = c[1]/(3.0*h)
    for i=2:N
      d[i] = (c[i]-c[i-1])/(3.0*h)
    end
    b = zeros(N)
    b[1] = (a[1]-a0)/h+(2.0/3.0)*c[1]*h
    for i=2:N
      b[i] = (a[i]-a[i-1])/h+(2.0*c[i]+c[i-1])*(h/3.0)
    end
    return [a,b,c,d]
    end

function f(x)
    return 1/(1+x*x)
  end

x0=-5
xN=5
N=10
h=(xN-x0)/N

T = spline(f,x0,xN,N)

A=zeros(0)
B=zeros(0)
C=zeros(0)
D=zeros(0)

for i=1:N
    append!(A,T[1][i])
    append!(B,T[2][i])
    append!(C,T[3][i])
    append!(D,T[4][i])
  end

function p(i,x)
      return   A[i]   +   B[i]*(x-x0-i*h)   +   C[i]*(x-x0-i*h)*(x-x0-i*h)   +
D[i]*(x-x0-i*h)*(x-x0-i*h)*(x-x0-i*h)
  end
```

2. Solution:

We can use the “PyPlot” to plot the function $f(x)=\dfrac{1}{1+x^2}$ and its natural spline $p(x)$ which are shown in Figure 1. It interpolates the f(x) well.

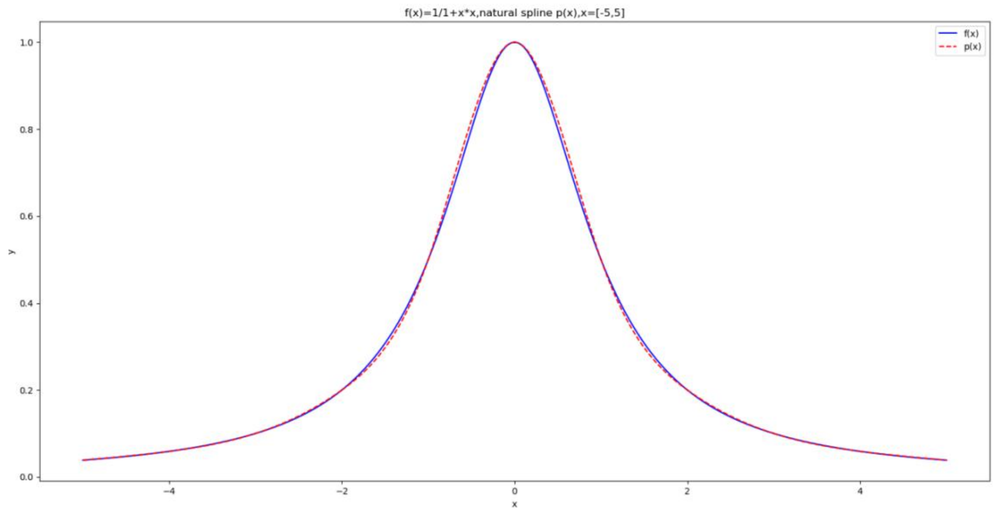

Figure 1

Here is the code in Julia.

```
julia> using PyPlot

    for i=1:N
       xmin=x0+(i-1)*h
       xmax=xmin+1*h
       x=(xmin:0.01:xmax)
       y1=zeros(0)
       y2=zeros(0)
       for j=1:length(x)
          append!(y1,f(x[j]))
          append!(y2,p(i,x[j]))
       end
       plot(x,y1,"b")
       plot(x,y2,"r--")
    end
```

3. Solution:

For $f(x)=x^2, f'(x)=2x, f''(x)=2$, so we can calculate that $\int_0^1 [f''(x)]^2 dx=\int_0^1 4dx=4$.

For $p_i(x)=a_i+b_i(x-x_i)+c_i(x-x_i)^2+d_i(x-x_i)^3, i\in\{1,\ldots,n\}$;

So, $p_i'(x)=b_i+2c_i(x-x_i)+3d_i(x-x_i)^2$, $p_i''(x)=2c_i+6d_i(x-x_i)$.

Thus, $\int_0^1 [p''(x)]^2 dx=\int_0^1 [2c_i+6d_i(x-x_i)]^2 dx=\sum_{i=1}^{N}\frac{4}{9d_i}\left[c_i^3-(c_i+3d_i*h)^3\right], h=\frac{xN-x0}{N}=\frac{1-0}{10}=0.1$.

Then we can use the Julia to compute this value and the result is:

$$\int_0^1 [p''(x)]^2 dx = 3.782389311135162$$

Comparing the values of the $\int_0^1 [f''(x)]^2 dx = 4$ and $\int_0^1 [p''(x)]^2 dx = 3.782389311135162$, we can conclude that $\int_0^1 [f''(x)]^2 dx$ is bigger than $\int_0^1 [p''(x)]^2 dx$.

Here is the code in Julia.

```
julia> function solveC(h,r)
        alpha = [4.0]
        gamma = zeros(0)
        n = length(r)
        for i=1:n-1
          append!(gamma,1.0/alpha[i])
          append!(alpha,4.0-gamma[i])
        end
        d = zeros(n)
        d[1] = r[1]
        for i=2:n
          d[i] = r[i]-gamma[i-1]*d[i-1]
        end
        c = zeros(n)
        c[n] = d[n]/alpha[n]
        for i=1:n-1
          c[n-i] = (d[n-i]-c[n-i+1])/alpha[n-i]
        end
        return c/h;
      end

      function spline(f,x0,xN,N)
        h = (xN-x0)/N
        a = zeros(N)
        x = zeros(N)
        a0=f(x0)
        for i=1:N
          x[i] = x0+i*h
          a[i] = f(x[i])
        end
        r = zeros(N)
        r[1] = 3.0/h*(a0-2.0*a[1]+a[2])
        for i=2:N-1
          r[i] = 3.0/h*(a[i-1]-2.0*a[i]+a[i+1])
        end
```

```
    c = solveC(h,r)
    d = zeros(N)
    d[1] = c[1]/(3.0*h)
    for i=2:N
      d[i] = (c[i]-c[i-1])/(3.0*h)
    end
    b = zeros(N)
    b[1] = (a[1]-a0)/h+(2.0/3.0)*c[1]*h
    for i=2:N
      b[i] = (a[i]-a[i-1])/h+(2.0*c[i]+c[i-1])*(h/3.0)
    end
    return [a,b,c,d]
    end

function f(x)
    return x*x
  end

x0=0
xN=1
N=10
h=(xN-x0)/N

T = spline(f,x0,xN,N)

C=zeros(0)
D=zeros(0)
for i=1:N
    append!(C,T[3][i])
    append!(D,T[4][i])
  end

function g(i)
      return (4.0/(9*D[i]))*((C[i])^3-(C[i]-3*D[i]*h)^3)
  end

y=zeros(0);
for i=1:N
  append!(y,g(i))
  end
```

Get the results:

```
julia> Base.sum(y)
3.782389311135162
```