MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING

ME 310 NUMERICAL METHODS

FALL 2022

PROGRAMMING PROJECT 3

Murat Berk Buzluk 2377653

Gökberk Çiçek 2377810

Contents

Abstract

ME310 lecture students create this report to develop proper programs to solve an interpolation problem with the quadratic spline interpolation method. MATLAB is used in programming because the broader application area of MATLAB makes it preferable for students to learn. We have learned how to code and deal with different problems caused by other functions. All the mistakes that we have done while writing the code led us to a better understanding of the course objectives.

1. INTRODUCTION

When a high number of data points are given, using a single high-order polynomial interpolation increases the operation number. Splines are lower-order polynomials that pass through only two data points and provide fewer number operations instead of a single high-order polynomial. In this project, a quartic spline $S_k(x)$ with given $n+1$ points will be defined, where k denotes the spline number in between two consecutive points of data points $[x_k, x_{k+1}]$. After determining the necessary equations that can construct a matrix form as $As = b$, where s is the unknown coefficients, proper computer programming will be executed to find s values.

2. HAND CALCULATIONS

For $n+1$ data points, there are n splines, and these splines include $5n$ constant coefficients. Therefore $5n$ number of equations are needed to solve $5n$ number of unknown coefficients.

Defining $5n$ equations:

- The first and last spline functions must pass through endpoints. (2 equations.)

$$s_{1,5}(x-x_1)^4 + s_{1,4}(x-x_1)^3 + s_{1,3}(x-x_1)^2 + s_{1,2}(x-x_1) + s_{1,1} = f(x_1) = y_1 \quad \textbf{(1)}$$

$x = x_1,$

$$s_{1,5}\cancel{(x_1-x_1)}^4 + s_{1,4}\cancel{(x_1-x_1)}^3 + s_{1,3}\cancel{(x_1-x_1)}^2 + s_{1,2}\cancel{(x_1-x_1)} + s_{1,1} = f(x_1) = y_1$$

$s_{1,1} = y_1$

$$s_{n+1,5}(x - x_{n+1})^4 + s_{n+1,4}(x - x_{n+1})^3 + \cdots .. + s_{n+1,1} = f(x_{n+1}) = y_{n+1} \qquad \textbf{(2)}$$

$x = x_{n+1},$

$s_{n+1,5}\cancel{(x_{n+1} - x_{n+1})}^4 + s_{n+1,4}\cancel{(x_{n+1} - x_{n+1})}^3 + \cdots .. + s_{n+1,1} = f(x_{n+1}) = y_{n+1}$

$s_{1,n+1} = y_{n+1}$

- The spline function values must be equal at interior points. (2n-2 equations.)

$$s_{k,5}(x - x_k)^4 + s_{k,4}(x - x_k)^3 \ldots .. + s_{k,1} = f(x_k) = y_k \qquad \textbf{(3)}$$

$$s_{k-1,5}(x - x_{k-1})^4 + s_{k-1,4}(x - x_{k-1})^3 \ldots .. + s_{k-1,1} = f(x_k) = y_k \qquad \textbf{(4)}$$

For instance, when $k = 2$,

$$s_{2,5}(x - x_2)^4 + s_{2,4}(x - x_2)^3 \ldots .. + s_{2,1} = f(x_2) = y_2$$

$$s_{2,5}(\cancel{x_2 - x_2})^4 + s_{2,4}(\cancel{x_2 - x_2})^3 \ldots .. + s_{2,1} = f(x_2) = y_2$$

$$s_{2,1} = y_2$$

$$s_{1,5}(x - x_1)^4 + s_{1,4}(x - x_1)^3 \ldots .. + s_{1,1} = f(x_2) = y_2$$

$$s_{1,5}(x_2 - x_1)^4 + s_{1,4}(x_2 - x_1)^3 \ldots .. + s_{1,1} = f(x_2) = y_2$$

- The first, second, and third derivatives of spline functions must be equal at interior points. ($3(n-1) = 3n-3$ equations)

$$S_k'(x_k) = S_{k-1}'(x_k) \quad \textbf{(5)}$$

$$S_2'(x_2) = S_1'(x_2) = 4s_{1,5}(x_2 - x_1)^3 + 3s_{1,4}(x_2 - x_1)^2 \ldots .= s_{2,2} + s_{2,3}(\cancel{x_2 - x_2})$$

$$S_k''(x_k) = S''_{k-1}(x_k) \quad \textbf{(6)}$$

$$S_2''(x_2) = S_1''(x_2) = 12s_{1,5}(x_2 - x_1)^2 + 6s_{1,4}(x_2 - x_1) + 2s_{1,3} = 2s_{2,3}$$

$$S'''_k(x_k) = S'''_{k-1}(x_k) \quad \textbf{(7)}$$

$$S_2'''(x_2) = S_1'''(x_2) = 24s_{1,5}(x_2 - x_1) + 6s_{1,4} = 6s_{2,4}$$

- It is reasonable to assume the highest continuous derivative of spline functions is zero at the endpoints. The resulting spline function is called ''natural spline ''. (2 equations.)

$$S'''_1(x_1) = S'''_n(x_{n+1}) = 0 \quad \textbf{(8)}$$

$$S_n^{(3)} = 6s_{n,4} + 24s_{4,5}(x - x_n) = 0 \quad \textbf{(9)}$$

$$S_1^{(3)} = 6s_{1,4} + 24s_{1,5}(x - x_1) = 0$$

- The last equation is provided by conditioning the second-order derivative at x_1.

  The second derivative of the first spline is equal to zero.

$$S_1''(x_1) = 0 \tag{10}$$

$$S_1''(x_1) = 2s_{1,3} + 6s_{1,4}(x - x_1) + 12s_{4,5}(x - x_1)^2 = 0$$

$$S_1''(x_1) = 2s_{1,3} + 6s_{1,4}(\cancel{x_1 - x_1}) + 12s_{4,5}(\cancel{x_1 - x_1})^2 = 0$$

$$s_{1,3} = 0$$

3. NUMERICAL RESULTS

Coefficient Matrix

Coefficients matrix is as above. As is can be interpreted that some values have truncation error. For example S1,4 is actually zero but is is 2.53E-17 in the above table and in the code

Table 1 Shows exact C coefficient matrix

-0.1585	0.499071	4.25E-16	2.53E-17	-2.07135
-0.1088	0.490786	-0.12428	-0.82854	0.428388
-0.0907	-0.49709	-0.27937	0.713657	-1.06147
-0.2543	-0.58667	-0.21028	-0.56011	0.421568
-0.8589	-1.12602	-0.14709	0.620281	-0.24507
-1.7568	-0.53963	0.243354	-0.35998	0.532389
-1.9775	-0.30007	0.501964	0.704796	-0.7193
-1.9589	0.370842	0.480205	-0.73381	0.634463
-1.7055	0.617922	0.331187	0.535117	-0.52076
-1.2794	1.090068	0.352728	-0.5064	0

The table above with 3 significant figures can be shown as

Table 2 Shows C coefficient values with 3 significant figure

-0.159	0.499	0.000	0.000	-2.071
-0.109	0.491	-0.124	-0.829	0.428
-0.091	-0.497	-0.279	0.714	-1.061
-0.254	-0.587	-0.210	-0.560	0.422
-0.859	-1.126	-0.147	0.620	-0.245
-1.757	-0.540	0.243	-0.360	0.532
-1.978	-0.300	0.502	0.705	-0.719
-1.959	0.371	0.480	-0.734	0.634
-1.706	0.618	0.331	0.535	-0.521
-1.279	1.090	0.353	-0.506	0.000

Inputs

Table 3 input values of the above coefficients

1	1.1	2	2.3	3	4	4.5	5	5.5	6	7
-0.1585	-0.1088	-0.0907	-0.2543	-0.8589	-1.7568	-1.9775	-1.9589	-1.7055	-1.2794	-0.343

4. GRAPHICS OF NUMERICAL RESULTS

Spline Matrix

Quadratic Spline matrix is as below. Different colors represent different splines. With the given inputs we get such a spline matrix. If we are created the spline with accurate input and output values we can get an accurate function. As we increase the number of input the quality of the spline function is increases.

Figure 1 shows the Quadratic spline of the table 3 inputs.

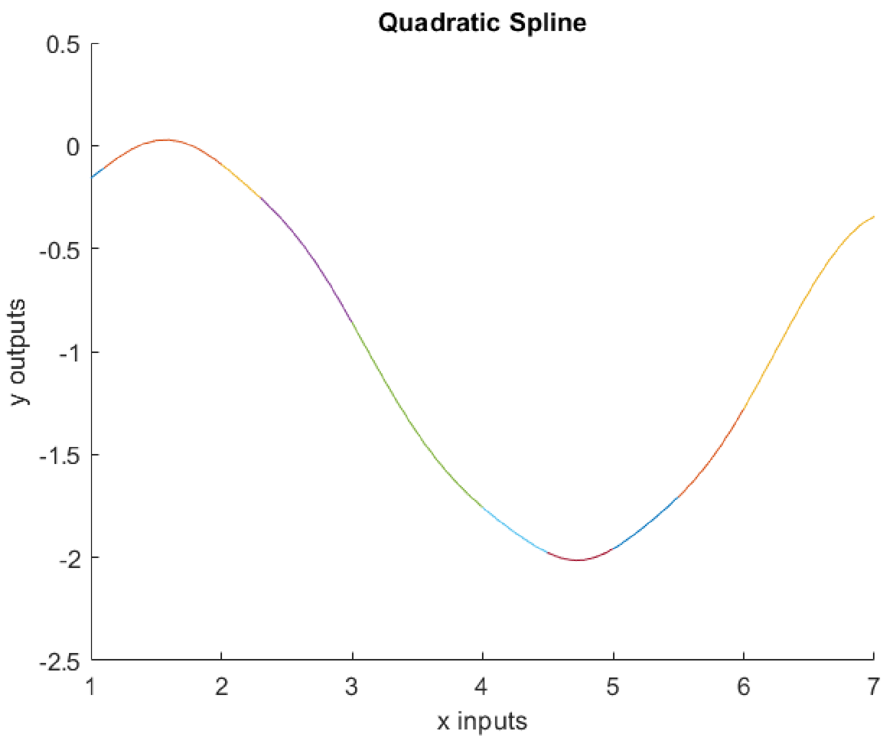

Figure 1 Shows y output values with increasing x input values.

Derivative Matrix

First derivative matrix is shown in the figure 2 below. Derivatives are accurate with the splines shown in the figure 1

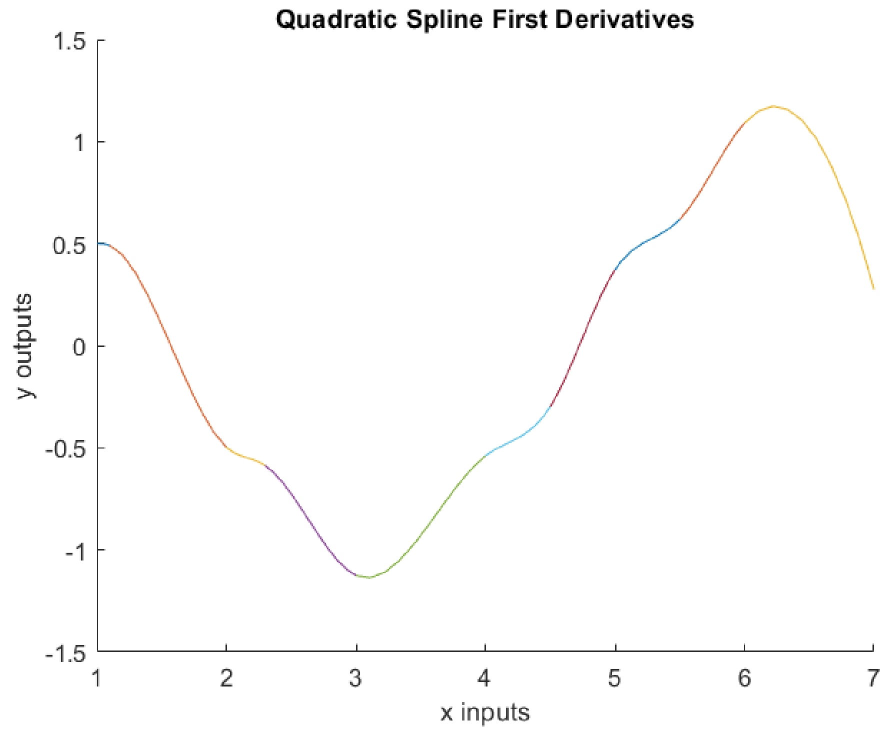

Figure 2 shows the derivative matrix of the system.

5. DISCUSSION AND CONCLUSION

In this project, a spline function $S_k(x)$ is defined, where x values are the input, and the corresponding y values are the output. Because the spline function is a quartic function, it has four unknown coefficients, and for $n + 1$ points, there are n number spline functions. That makes a total $5n\ equations$ to find unknown coefficients s_k, where k denotes the spline number. After defining the necessary $5n$ equations, a matrix system, $As = b$, is determined.

The aim of the programming is to solve the matrix system to find all unknown coefficients. After determining the coefficients of spline functions, the program will show graphs of $S(x)$ and its derivative.

7. Appendices

Code of the Program

```
clear
clc

f = fopen("input.txt");
data = textscan(f,'%s');
fclose(f);
variable = str2double(data{1}(1:1:end));

[j] = size(variable);

xx = variable(1:1:j/2);
yy = variable(j/2+1:1:end);

nn = length(xx);

A = zeros(5*(nn-1),5*(nn-1));
b = zeros(5*(nn-1),1);

for di1 = 1:nn-1
    A(di1,(di1-1)*5+1) = 1;
    A(di1+(nn-1), (di1-1)*5+(1:5)) = [1 ...
        (xx(di1+1)-xx(di1)) ...
        (xx(di1+1)-xx(di1))^2 ...
        (xx(di1+1)-xx(di1))^3 ...
        (xx(di1+1)-xx(di1))^4 ];
    b(di1) = yy(di1);
    b(di1 + (nn-1)) = yy(di1+1);
end

for di1 = 1:nn-2
    A(di1+(nn-1)*2,(di1-1)*5+(1:5)) = [0 ...
        1 ...
        2*(xx(di1+1)-xx(di1)) ...
        3*(xx(di1+1)-xx(di1))^2 ...
        4*(xx(di1+1)-xx(di1))^3];
    A(di1+(nn-1)*2,(di1+1-1)*5+(1:5)) = [0 ...
        -1 ...
        0 ...
        0 ...
        0 ];
end

for di1 = 1:nn-2
    A(di1+(nn-1)*3-1,(di1-1)*5+(1:5)) = [0 ...
        0 ...
        2 ...
        6*(xx(di1+1)-xx(di1)) ...
        12*(xx(di1+1)-xx(di1))^2 ];
    A(di1+(nn-1)*3-1,(di1+1-1)*5+(1:5)) = [0 ...
        0 ...
        -2 ...
        0 ...
        0 ];
end
```

```
for di1 = 1:nn-2
    A(di1 +(nn-1)*4-2,(di1-1)*5+(1:5)) = [0 ...
        0 ...
        0 ...
        6 ...
        24*(xx(di1+1)-xx(di1)) ];
    A(di1 +(nn-1)*4-2,(di1+1-1)*5+(1:5)) = [0 ...
        0 ...
        0 ...
        -6 ...
        0 ];
end

A(5*(nn-1)-2,3) = 1;
A(5*(nn-1)-1,4) = 1;
A(5*(nn-1),5*(nn-1)) = 1;

S = A\b;

figure
title('Quadratic Spline')
xlabel('x inputs')
ylabel('y outputs')
hold on
for di1 = 1:(nn-1)
    xnow = linspace(xx(di1),xx(di1+1),(nn-1));
    ynow = zeros(size(xnow)) ;
    for di2 = 1:length(xnow)
        ynow(di2) = transpose(S((di1-1)*5+(1:5)))* ...
            [1; ...
            (xnow(di2)-xnow(1)); ...
            (xnow(di2)-xnow(1))^2; ...
            (xnow(di2)-xnow(1))^3; ...
            (xnow(di2)-xnow(1))^4 ];
    end
    plot(xnow,ynow)
end

figure
title('Quadratic Spline First Derivatives')
xlabel('x inputs')
ylabel('y outputs')
hold on
for di1 = 1:(nn-1)
    xnow = linspace(xx(di1),xx(di1+1),(nn-1));
    ynow = zeros(size(xnow)) ;
    for di2 = 1:length(xnow)
        ynow(di2) = transpose(S((di1-1)*5+(1:5)))* ...
            [0; ...
            1; ...
            2*(xnow(di2)-xnow(1)); ...
            3*(xnow(di2)-xnow(1))^2; ...
            4*(xnow(di2)-xnow(1))^3 ];
    end
    plot(xnow,ynow)
end

x_input = input('enter the x input: ');
```

```
y_output = 0;

if x_input<xx(1) || x_input>xx(nn)
    fprintf('input is out of the interval')
end

for t = 1:(nn)
    if  x_input>=xx(t) && x_input<=xx(t+1)
        y_output =  transpose(S((t-1)*5+(1:5)))* ...
            [1; ...
            (x_input-xx(t)); ...
            (x_input-xx(t))^2; ...
            (x_input-xx(t))^3; ...
            (x_input-xx(t))^4 ];
    break
    end
end
fprintf('the y outout is: %.4f\n', y_output)
```