First Year Computing (C++ Course)

Dr John Joss Chinn

Week 9.

Help with the Assignment

A change of plan this week. As we are rapidly approaching the Easter break and I need you all to be reasonably confident that you can do the assignment, then we simply have to skip C++ Pointers. This is unfortunate but, (A) you can do the assignment without needing to know about them and (B) they are a little complicated and would take up at least one lecture period on their own. Therefore today's lecture is about how to go about doing the assignment.

Hopefully, you are all aware that you are required to find the equations of the cubic spline curves through the set of data points given out in the assignment, on page 69, of last weeks hand out. The equations are of the form

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \qquad (i = 0 \text{ to } n-1) \qquad (1)$$

You do not need to know the mathematical derivation of the equations (which I covered anyway in Week 7). You only need to know how to use them. The equations that you need are

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 \\ 0 & 0 & h_3 & 2(h_3+h_4) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \frac{3}{h_2}(a_3-a_2) - \frac{3}{h_1}(a_2-a_1) \\ \frac{3}{h_3}(a_4-a_3) - \frac{3}{h_2}(a_3-a_2) \\ \frac{3}{h_4}(a_5-a_4) - \frac{3}{h_3}(a_4-a_4) \end{pmatrix}, \qquad (2)$$

$$d_i = \frac{(c_{i+1} - c_i)}{3h_i} \qquad (3)$$

and

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1}). \qquad (4)$$

where

$$h_i = x_{i+1} - x_i \quad \text{and} \quad a_i = y_i. \qquad (5)$$

Consider the set of simultaneous equations, from week 6,

$$\begin{aligned} 2x_1 + 8x_2 - 25x_3 &= 472 \\ 5x_1 - 20x_2 + x_3 &= -156 \\ 10x_1 + 2x_2 + 5x_3 &= -24 \end{aligned}$$

These could be written as a matrix and two column vectors

$$\begin{pmatrix} 2 & 8 & -25 \\ 5 & -20 & 1 \\ 10 & 2 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 472 \\ -156 \\ -24 \end{pmatrix}. \quad (6)$$

This is in the same form as equation (2). The augmented matrix for this system is, as we have seen before,

$$\begin{pmatrix} 2 & 8 & -25 & 472 \\ 5 & -20 & 1 & -156 \\ 10 & 2 & 5 & -24 \end{pmatrix}. \quad (7)$$

	x		y
x_0	0	y_0	0
x_1	1	y_1	2
x_2	4	y_2	1
x_3	6	y_3	4
x_4	9	y_4	5
x_5	11	y_5	4

By using the table of values on the left, given in the last two weeks notes, in equation (5), the augmented matrix of the system in equation (2) is

```
8       3       0       0       -7
3       10      2       0       5.5
0       2       10      3       -3.5
0       0       3       10      -2.5
```

(8)

This is very easy to work out; just work out the h_i's and a_i's in equation (2).

We have used the augmented matrix of equation (7) in gauss3.cpp (given in Appendix 1., for a 3 x 4 system) to solve for x_1, x_2 and x_3. The result is

$$\begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & -16 \end{pmatrix} \quad (9)$$

In other words, $x_1 = 4$, $x_2 = 8$ and $x_3 = -16$. We can also use gauss3.cpp to solve the system of equations (8) for c_1, c_2 c_3 and c_4 in equation (2). The program, gauss3a.cpp is given in Appendix 2., for this 4 x 5 system. (**Important**, try to spot the differences.) The result is

```
1       0       0       0       -1.26307
0       1       0       0       1.03485
0       0       1       0       -0.529637
0       0       0       1       -0.0911089
_
```

(10)

Therefore $c_1 = -1.26307$, $c_2 = 1.03485$, $c_3 = -0.52964$ and $c_4 = -0.09111$. We can now use these values for the c_i's in equations (3) and (4) to find the d_i's and the b_i's of equation (1). Recall, from the week 7 notes, that for a 'Natural' cubic spline we have

$$c_0 = 0 \text{ and } c_n = 0. \quad (11)$$

Therefore, for our system of six sets of data points; x_0,y_0 to x_5,y_5, we need five cubic spline curves, S_0 to S_4, and we have $c_0 = 0$ and $c_5 = 0$.

Putting all this together gives us the following cubic spline equations

$$
\begin{aligned}
S_0(x) &= 0 + 2.42102(x - x_0) + 0 - 0.42102(x - x_0)^3 \\
S_1(x) &= 2 + 1.15796(x - x_1) - 1.26307(x - x_1)^2 + 0.25532(x - x_1)^3 \\
S_2(x) &= 1 + 0.47329(x - x_2) + 1.03485(x - x_2)^2 - 0.26075(x - x_2)^3 \\
S_3(x) &= 4 + 1.48372(x - x_3) - 0.52964(x - x_3)^2 + 0.04872(x - x_3)^3 \\
S_4(x) &= 5 - 0.37852(x - x_4) - 0.09111(x - x_4)^2 + 0.01518(x - x_4)^3
\end{aligned}
\qquad (12)
$$

where x_0 to x_4 are from the table on page 76. So, finally, we have

$$S_0(x) = 0 + 2.42102(x - 0) + 0 - 0.42102(x - 0)^3 \qquad (13.0)$$

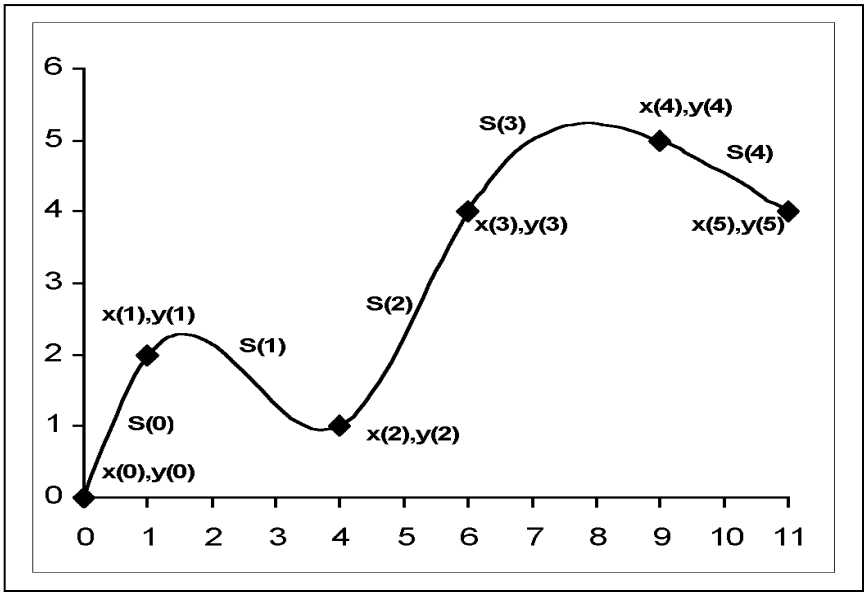
$$S_1(x) = 2 + 1.15796(x - 1) - 1.26307(x - 1)^2 + 0.25532(x - 1)^3 \qquad (13.1)$$

$$S_2(x) = 1 + 0.47329(x - 4) + 1.03485(x - 4)^2 - 0.26075(x - 4)^3 \qquad (13.2)$$

$$S_3(x) = 4 + 1.48372(x - 6) - 0.52964(x - 6)^2 + 0.04872(x - 6)^3 \qquad (13.3)$$

$$S_4(x) = 5 - 0.37852(x - 9) - 0.09111(x - 9)^2 + 0.01518(x - 9)^3 . \qquad (13.4)$$

We can use this set of equations in Microsoft Excel to plot the graphs, the curves between the data points.



To plot the graphs in Excel you will need a sufficient number of little points between each pair of consecutive given data points, x_0,y_0 and x_1,y_1 etc. I would suggest that you break up each interval into ten portions. So, you will then have one column of numbers, between $x_0 = 0$ and $x_5 = 11$. You will also have another column which contains the equations (13). So you will have equation (13.0) between $x_0 = 0$ and $x_1 = 1$, equation (13.1) between $x_1 = 1$ and $x_2 = 4$, etc. The Excel file that I used is shown over the page.

A	B	C	D	E	F	G	H	I
		Eq.(13.0)	Eq.(13.1)	Eq.(13.2)	Eq.(13.3)	Eq.(13.4)		
1	0	0						0
2	0.1	0.241681						0.241681
3	0.2	0.480836						0.480836
4	0.3	0.714938						0.714938
5	0.4	0.941463						0.941463
6	0.5	1.157883						1.157883
7	0.6	1.361672						1.361672
8	0.7	1.550304						1.550304
9	0.8	1.721254						1.721254
10	0.9	1.871994						1.871994
11	1	2	2					2
12	1.3		2.240605					2.240605
13	1.6		2.29522					2.29522
14	1.9		2.205206					2.205206
15	2.2		2.011924					2.011924
16	2.5		1.756738					1.756738
17	2.8		1.481007					1.481007
18	3.1		1.226096					1.226096
19	3.4		1.033364					1.033364
20	3.7		0.944175					0.944175
21	4		0.99989	1				1
22	4.2			1.133966				1.133966
23	4.4			1.338204				1.338204
24	4.6			1.600198				1.600198
25	4.8			1.907432				1.907432
26	5			2.24739				2.24739
27	5.2			2.607556				2.607556
28	5.4			2.975414				2.975414
29	5.6			3.338448				3.338448
30	5.8			3.684142				3.684142
31	6			3.99998	4			4
32	6.3				4.398764			4.398764
33	6.6				4.710085			4.710085
34	6.9				4.941856			4.941856
35	7.2				5.101971			5.101971
36	7.5				5.19832			5.19832
37	7.8				5.238797			5.238797
38	8.1				5.231296			5.231296
39	8.4				5.183707			5.183707
40	8.7				5.103924			5.103924
41	9				4.99984	5		5
42	9.2					4.920773		4.920773
43	9.4					4.834986		4.834986
44	9.6					4.743368		4.743368
45	9.8					4.646648		4.646648
46	10					4.545555		4.545555
47	10.2					4.440817		4.440817
48	10.4					4.333164		4.333164
49	10.6					4.223324		4.223324
50	10.8					4.112027		4.112027
51	11					4		4

You only need columns B and I. I put the rest in just to see the similarities between consecutive cubic equations at the given data points. The way that you would insert the equations (13) is, for instance, say equation (13.1) in cell D12, is;

=2+1.15796*(B12-1)-1.26307*(B12-1)^2+0.25532*(B12-1)^3

Now all that you need to do is plot column I against column B, using an “xy (scatter)” graph.

Ok, you really need to go through the motions of what I have covered above, for yourselves. In this way you will begin to understand what is needed in the assignment. Then, for your actual assignment, you need to use the data on page 69 of last weeks notes.

To give you a further hint. The example that I have discussed so far is for *six* sets of data points. Therefore *five* cubic equations are needed to plot curves between each consecutive pair of points. We then ended up having to solve a 4 x 5 augmented matrix to obtain c_1 to c_4 ($c_0 = 0$ and $c_5 = 0$). The data given out in the assignment, last week, is for *thirteen* sets of data points. You will then need to construct *twelve* cubic spline curves. Now $c_0 = 0$ and $c_{12} = 0$, for a Natural cubic Spline, so you will need to solve for c_1 to c_{11}, i.e. you will have a 11 x 12 augmented matrix system.

If you can manage to achieve the above, with a reasonable write-up, then you will pass the course. However you can do much better than this. You can do the following:-

(1) Read in the data from an external text file (see last weeks notes),
(2) Create the elements of the augmented matrix from an equation, like equation (2) (much bigger for you), in your C++ code,
(3) Run the gauss.cpp program and also produce the output, in two columns of data to an external text file
(4) Open this text file in Excel and plot the graphs using “xy (scatter)”

So, for items (1) to (4), in this list, you will need to write some C++ code for both the front end and the back end of gauss3.cpp.

Appendix 1. gauss3.cpp for a 3 x 4 matrix.

```
// gauss3.cpp
#include<iostream>
#include<conio.h>
#include<math.h>       //needed for fabs

void main()
{
        int i,j,marker,k;
        float pivot;
        float temp;

        //augmented matrix
        float A[3][4]={2,   8, -25,  472,
                       5, -20,   1, -156,
                      10,  2,   5,  -24};

for(k=0;k<3;k++) //start of big loop
{
        //need to find the biggest A[i][k]
        pivot=0;
        for(i=k;i<3;i++)
        {
                if(fabs(A[i][k])>fabs(pivot))
                {
                        pivot=A[i][k];
                        marker=i;
                }
        }

        cout<<"\npivot= "<<pivot<<" i= "<<marker<<"\n\n";
        getch();

        //routine to swap rows if A[k][k] is not the biggest
        if(marker!=k)
        {
                for(j=0;j<4;j++)
                {
                        temp=A[k][j];
                        A[k][j]=A[marker][j];
                        A[marker][j]=temp;
                }
        }

        /*just print out the augmented matrix */
        for(i=0;i<3;i++)
        {
                for(j=0;j<4;j++)
                {
                        cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }
        cout<<endl;
        getch();

        //divide 1st row in matrix or reduced matrix by pivot
        for(j=k;j<4;j++)
        {
                A[k][j]=A[k][j]/pivot;
        }

        /*for  i=1 to 3 multiply each row by -A[i][k] and add this to row i
        to make a new row i */
        for(i=k+1;i<3;i++)
        {
                if(fabs(A[i][k])>0)
                {
                        temp=-A[i][k];
                        for(j=k;j<4;j++)
                        {
                          A[i][j]=A[i][j]+A[k][j]*temp;
                        }
                }
        }

}       //end of big loop

        /*just print out the augmented matrix (again)*/
```

```
        for(i=0;i<3;i++)
        {
                for(j=0;j<4;j++)
                {
                        cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }
        getch();

   /*back substitution*/
   for(k=2;k>0;k--)
    {
          for(i=k;i>0;i--)
           {
             temp=-A[i-1][k];
             for(j=3;j>=i;j--)
                 {
                  if(A[k][j]!=0)
                   {
                     A[i-1][j]=A[i-1][j]+A[k][j]*temp;
                    }
                 }
           }
        }

   cout<<"\n\n";

        /*just print out the augmented matrix (again)*/
        for(i=0;i<3;i++)
        {
                for(j=0;j<4;j++)
                {
                        cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }

        getch();
}
```

Appendix 2. gauss3a.cpp for a 4 x 5 matrix.

```
// gauss3a.cpp
#include<iostream>
#include<conio.h>
#include<math.h>      //needed for fabs

void main()
{
       int i,j,marker,k;
       float pivot;
       float temp;

       //augmented matrix
       float A[4][5]={8,   3,   0,  0,  -7,
                      3,  10,   2,  0,   5.5,
                      0,   2,  10,  3,  -3.5,
                      0,   0,   3, 10,  -2.5 };

for(k=0;k<4;k++) //start of big loop
{
       //need to find the biggest A[i][k]
       pivot=0;
       for(i=k;i<4;i++)
       {
              if(fabs(A[i][k])>fabs(pivot))
              {
                     pivot=A[i][k];
                     marker=i;
              }
       }

       cout<<"\npivot= "<<pivot<<" i= "<<marker<<"\n\n";
       getch();

       //routine to swap rows if A[k][k] is not the biggest
       if(marker!=k)
       {
              for(j=0;j<5;j++)
              {
                     temp=A[k][j];
                     A[k][j]=A[marker][j];
                     A[marker][j]=temp;
              }
       }

       /*just print out the augmented matrix */
       for(i=0;i<4;i++)
       {
              for(j=0;j<5;j++)
              {
                     cout<<A[i][j]<<"\t";
              }
              cout<<endl;
       }
       cout<<endl;
       getch();

       //divide 1st row in matrix or reduced matrix by pivot
       for(j=k;j<5;j++)
       {
              A[k][j]=A[k][j]/pivot;
       }

       /*for  i=1 to 3 multiply each row by -A[i][k] and add this to row i
       to make a new row i */
       for(i=k+1;i<4;i++)
       {
              if(fabs(A[i][k])>0)
              {
                     temp=-A[i][k];
                     for(j=k;j<5;j++)
                     {
                       A[i][j]=A[i][j]+A[k][j]*temp;
                     }
              }
       }

}      //end of big loop
```

```
        /*just print out the augmented matrix (again)*/
        for(i=0;i<4;i++)
        {
                for(j=0;j<5;j++)
                {
                        cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }

        getch();

   /*back substitution*/
   for(k=3;k>0;k--)
    {
          for(i=k;i>0;i--)
           {
             temp=-A[i-1][k];
             for(j=4;j>=i;j--)
                 {
                  if(A[k][j]!=0)
                   {
                     A[i-1][j]=A[i-1][j]+A[k][j]*temp;
                    }
                 }
           }
        }

   cout<<"\n\n";

        /*just print out the augmented matrix (again)*/
        for(i=0;i<4;i++)
        {
                for(j=0;j<5;j++)
                {
                        cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }

        getch();
}
```