First Year Computing (C++ Course)

Dr John Joss Chinn

Week 7.

Gaussian Elimination with Back Substitution, Cubic Splines

Introduction

Last week we looked at how to code to solve a system of simultaneous equations using Gaussian Elimination with Back Substitution. This week we will go over Back Substitution using a 4 x 5 matrix rather than a 3 x 4 matrix, as the repeated part of the methodology is clearer with the bigger system of equations, bigger matrix.

For your assignment you will need to know about Natural Cubic Splines. This is a method used to draw a smooth curve through any set of data points (x,y points). It is used in drawing packages such as AutoCAD and also used in modelling simulation software, for fluid flows and structures, to help model the geometry of a component. You will learn more about such things in future years. You will also cover splines again in the third year Numerical Analysis course, run by Dr Leech.

Back Substitution

It is actually easier to understand the repeating nature of solving an upper tridiagonal matrix using backward substitution if the matrix is a 4 x 5 rather than a 3 x 4. This is because, by the time that you start to get the idea of the methodology it is over with.

Consider the following system of simultaneous equations (given in last weeks tutorial, question (4)) and their corresponding augmented matrix:-

$$\begin{aligned} x_1 - 3x_2 + 6x_3 + 2x_4 &= 39 \\ -2x_1 + 4x_2 - 7x_3 - 3x_4 &= -48 \\ 5x_1 + 2x_2 + x_3 + 4x_4 &= 37 \\ 3x_1 + 2x_2 + 6x_3 + 7x_4 &= 67 \end{aligned}$$

$$\begin{pmatrix} 1 & -3 & 6 & 2 & 39 \\ -2 & 4 & -7 & -3 & -48 \\ 5 & 2 & 1 & 4 & 37 \\ 3 & 2 & 6 & 7 & 67 \end{pmatrix}$$

This matrix has the solution $x_1 = 4$, $x_2 = 3$, $x_3 = 7$, $x_4 = 1$. Therefore the matrix that we want to end up with will be

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

The matrix below is the matrix after the row and column reduction part of the Guassian Elimination. It still needs to have the back substitution work done on it. The matrix on the right is really just a schematic to show what is happening to the elements generally. The superscripts on the augmenting elements show the iterations. In other words these elements become altered after each one of the tasks below and a change in the superscript number indicates that there has been a change.

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & -0.291\overline{6} & -6.91\overline{6} \\ 0 & 0 & 1 & 0.74359 & 7.74359 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4}^{(0)} \\ 0 & 1 & a_{1,2} & a_{1,3} & a_{1,4}^{(0)} \\ 0 & 0 & 1 & a_{2,3} & a_{2,4}^{(0)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(1a) We first eliminate the elements in column 3 (the fourth column). We first eliminate element $a_{2,3}$. Multiply row 3 (the fourth row) by $-a_{2,3}$ (i.e. -0.74359) and add this to row 2 (the third row)

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & -0.291\overline{6} & -6.91\overline{6} \\ 0 & 0 & 1 & 0.74359 & 7.74359 \\ 0 & 0 & 0 & -0.74359 & -0.74359 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & -0.291\overline{6} & -6.91\overline{6} \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4}^{(0)} \\ 0 & 1 & a_{1,2} & a_{1,3} & a_{1,4}^{(0)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(2a) We next eliminate element $a_{1,3}$. Multiply row 3 (the fourth row) by $-a_{1,3}$ (i.e. $0.291\overline{6}$) and add this to row 1 (the second row)

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & -0.291\overline{6} & -6.91\overline{6} \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 0.291\overline{6} & 0.291\overline{6} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & 0 & -6.625 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4}^{(0)} \\ 0 & 1 & a_{1,2} & 0 & a_{1,4}^{(1)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(3a) We next eliminate element $a_{0,3}$. Multiply row 3 (the fourth row) by $-a_{0,3}$ (i.e. -0.8) and add this to row 0 (the first row)

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0.8 & 7.4 \\ 0 & 1 & -1.375 & 0 & -6.625 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & -0.8 & -0.8 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0 & 6.6 \\ 0 & 1 & -1.375 & 0 & -6.625 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & a_{0,2} & 0 & a_{0,4}^{(1)} \\ 0 & 1 & a_{1,2} & 0 & a_{1,4}^{(1)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

This eliminates all of the extraneous elements in column 3 (fourth column).

(1b) We now need to do the same thing to column 2 (third column). We first eliminate element $a_{1,2}$. Multiply row 2 (the third row) by $-a_{1,2}$ (i.e. 1.375) and add this to row 1 (the second row)

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0 & 6.6 \\ 0 & 1 & -1.375 & 0 & -6.625 \\ 0 & 0 & 1.375 & 0 & 9.625 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0 & 6.6 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & a_{0,2} & 0 & a_{0,4}^{(1)} \\ 0 & 1 & 0 & 0 & a_{1,4}^{(2)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(2b) We next eliminate element $a_{0,2}$. Multiply row 2 (the third row) by $-a_{0,2}$ (i.e.-0.2) and add this to row 0 (the first row)

$$\begin{pmatrix} 1 & 0.4 & 0.2 & 0 & 6.6 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & -0.2 & 0 & -1.4 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0.4 & 0 & 0 & 5.2 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & a_{0,1} & 0 & 0 & a_{0,4}^{(2)} \\ 0 & 1 & 0 & 0 & a_{1,4}^{(2)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

This eliminates all of the extraneous elements in column 2 (third column).

(1c) We now need to do the same thing to column 1 (second column). We first eliminate element $a_{0,1}$. Multiply row 1 (the second row) by $-a_{0,1}$ (i.e. –0.4) and add this to row 0 (the first row)

$$\begin{pmatrix} 1 & 0.4 & 0 & 0 & 5.2 \\ 0 & -0.4 & 0 & 0 & -1.2 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 & a_{0,4}^{(3)} \\ 0 & 1 & 0 & 0 & a_{1,4}^{(2)} \\ 0 & 0 & 1 & 0 & a_{2,4}^{(1)} \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

This gives us the final solution matrix.

Writing the Code

We need to think through the above process in a methodical, computer-like way. This will help us to write code to perform the process. In the above tasks we have performed the following:-

(1a)	eliminate	$a_{2,3}$	modify	$a_{2,4}^{(0)}$
(2a)	eliminate	$a_{1,3}$	modify	$a_{1,4}^{(0)}$
(3a)	eliminate	$a_{0,3}$	modify	$a_{0,4}^{(0)}$
(1b)	eliminate	$a_{1,2}$	modify	$a_{1,4}^{(1)}$
(2b)	eliminate	$a_{0,2}$	modify	$a_{0,4}^{(1)}$
(1c)	eliminate	$a_{0,1}$	modify	$a_{0,4}^{(2)}$

In (1a) there are two operations along the row at i = 2. These are at j = 3 and j =4.
In (2a) there are two operations along the row at i = 1. These are at j = 3 and j =4.
In (3a) there are two operations along the row at i = 0. These are at j = 3 and j =4.

In (1b) there are two operations along the row at i = 1. These are at j = 2 and j =4.
In (2b) there are two operations along the row at i = 0. These are at j = 2 and j =4.

In (1c) there are two operations along the row at i = 0. These are at j = 1 and j =4.

There are three groups of operations here, **a**, **b** and **c**. Each group contains several i values. For each i value the tasks are performed along the rows, for different j values. We therefore need three sets of nested loops to perform these tasks. The piece of code to perform the back substitution is given below. Let us try to understand how it works.

```
/*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;
          }
        }
     }
  }
```

We need the outer loop to operate three times, to perform the group tasks, **a**, **b** and **c**. We chose an integer variable k to increment from 3 down to 1.

```
for(k=3;k>0;k--)
```

(We increment down because we have already found x_4 and we want to work back down through x_3 and x_2 down to x_1. Hence the name; “Back substitution”)
For each group task we go through several values of the integer variable i. In group task a, i starts at 2, in group task b, i starts at 1, and in group task c, i starts at 0. We can therefore use k (which is itself decrementing for each group task) to set the upper limit of i for each group task.

```
for(i=k;i>0;i--)
```

For each i value we set the floating point variable `temp` to the value of the element which we wish to eliminate (also times -1).

```
temp=-A[i-1][k]
```

We multiply the k^{th} row (i.e. decrement j) by `temp` and then add this to the $(i-1)^{th}$ row. This eliminates the desired element in the $(i-1)^{th}$ row and modifies the element on the right hand side of the $(i-1)^{th}$ row. (The k^{th} row will always contain the solution for x_k on the right hand side and one non-zero element 1, in one of the columns.)

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

Notice that we only work with the non-zero elements in the k^{th} row (`if(A[k][j]!=0)`). This would not really make any difference as something times zero is also zero.

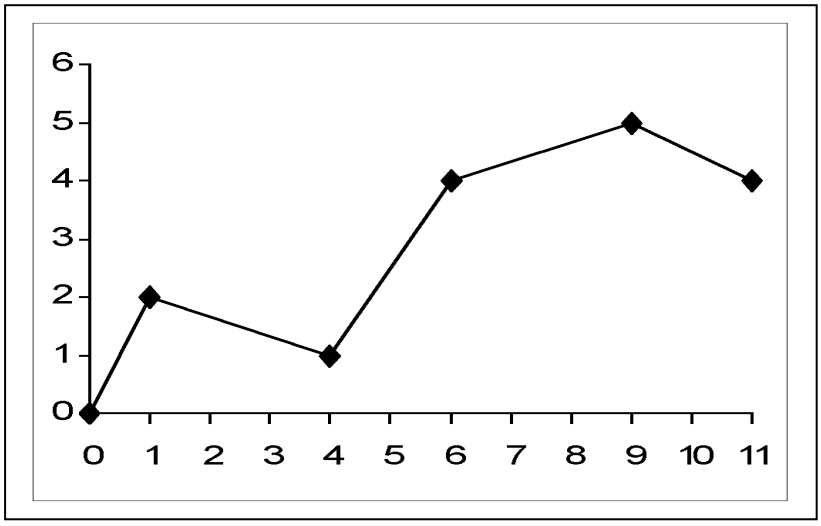
If you find all this difficult to follow (feel reassured, most people do!) then it is often a good idea to actually go through the process by hand for several iterations. Indeed, that is how anyone goes about writing code, of this sort, in the first place.

Natrural Cubic Splines

Introduction

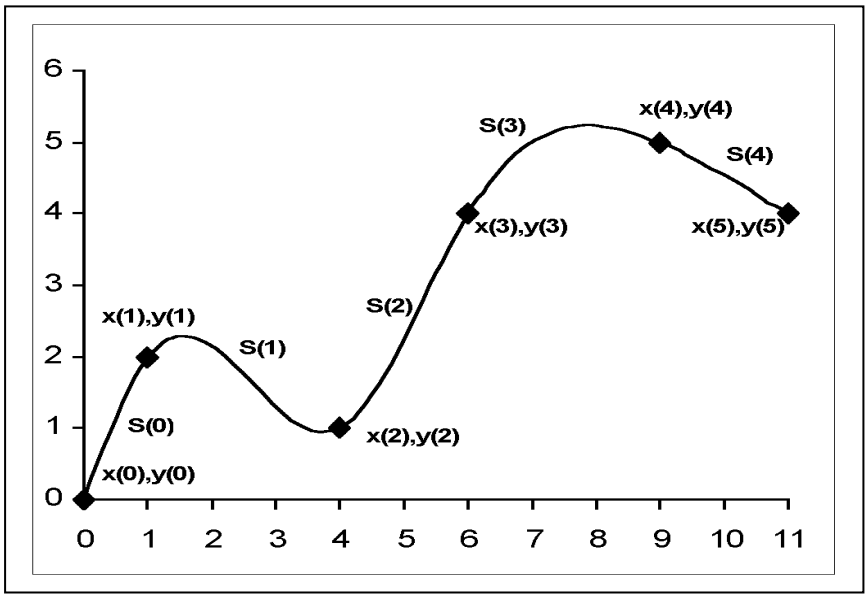
A spline curve is the curve through a set of x, y data points that a long flexible ruler would make if it were bent around the points. Such a ruler is called a *Spline*, by boatbuilders. It is used to fit the planking across the ribs of the boat. A more contemporary use of this numerical method is to fit a curve on a complicated geometry of a component, using a CAD package.

Consider the set of six data points below



	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

The six pairs of x,y data points in the chart above simply have a straight line drawn between them. So five straight lines. You will notice that there are discontinuities, i.e. sharp corners, at the points. Contrast this with the curve below, through the same set of data points.



Here you will see that there are no discontinuities. There is actually a separate curve through each two consecutive points. The form of this curve is a cubic. (Hence the name; “Cubic Splines”.) A cubic has the following form

$$y = a + bx + cx^2 + dx^3 .$$

More particularly we define the actual spline curves as

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

So S_0 goes from x_0,y_0 to x_1,y_1 and S_1 goes from x_1,y_1 to x_2,y_2 etc. So for 6 data points we have 5 curves. (See the figure above.) The reason why equation (1) is of this form is as follows. Looking at curve S_1, for example, we have

$$S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \qquad (x_0 < x < x_1) \qquad (2)$$

This means that this curve applies for all of the x values between x_0 and x_1. When $x = x_1$ then we just get

$$S_1(x_1) = a_1$$

as all of the bracketed terms disappear. As the curve S_1 must go through the point x_1,y_1 then this means that

$$a_1 = y_1. \qquad (3)$$

So this is why the cubic polynomial spline equation, between the data points, is written as it is in equation (1), i.e with $(x-x_i)$ terms instead of just x terms. In general we have

$$a_i = y_i. \qquad (4)$$

So the coefficients a_i are just the y_i ‘s. How do we find the other coefficients, b_i, c_i and d_i in equation (1)? Well there are a few rules that we can use to find them, and then, not surprisingly, we have to end up solving an augmented matrix by Gaussian Elimination with Back substitution. Let us take a look at these rules.

(I) Well we have already looked at the first rule. When $x = x_i$ then

$$S_i(x_i) = a_i = y_i. \qquad (5)$$

(II) Not only has $S_1(x_1)$ got to go through the point x_1,y_1 but, at the other end, it has to go through the point x_2,y_2. So we have

$$S_1(x_2) = a_2$$

From this and equation (5) we can say $S_1(x_2) = S_2(x_2)$. In general

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) \qquad (6)$$

(III) To preserve the continuity at the points, i.e. to not have sharp corners, then the slopes of the curves on either side of each point must be the same. By slopes we mean the same gradient, or derivative. Writing this in mathematical form we have

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) \qquad (7)$$

(IV) Also, in order to find all of the coefficients, then we must have a couple more conditions. One of them is that the second derivatives of the curves either side of a point, must be the same. This gives

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}) \qquad (8)$$

(V) Finally, for a Natural cubic Spline, we have the conditions

$$S''(x_0) = 0 \qquad (9)$$

and $S''(x_n) = 0$.

Just as a note, for a 'Clamped' cubic spline, as opposed to a 'Natural' cubic spline, the gradient of the curves at the end points, x_0,y_0 and x_n,y_n are fixed, as if they are clamped. For natural cubic splines only the positions of the endpoints are fixed, and not their slopes (as indicated by equation(9)). We are only going to look at natural cubic splines.

We can use these five conditions to help us find the coefficients in equation (1): a_i, b_i, c_i and d_i. As we have found $a_i = y_i$. From equation (5) we have

$$S_i(x_i) = a_i \qquad (5)$$

and from equation (6) we have

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) \qquad (6)$$

So, from equation (1) we have

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \qquad (1)$$

so that

$$S_i(x_{i+1}) = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = a_{i+1} \qquad (10)$$

We will use equation (10) presently. From equation (1) we get

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$

So that

$$S_i'(x_i) = b_i \quad \text{and} \quad S_{i+1}'(x_{i+1}) = b_{i+1}$$

also

$$S_i'(x_{i+1}) = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2$$

Therefore, from equation (7), $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$, we get

$$S_i'(x_{i+1}) = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 = b_{i+1} \qquad (11)$$

We will use equation (11) presently. From equation (1) we have

$$S_i''(x) = 2c_i + 6d_i(x - x_i)$$

So that

$$S_i''(x_i) = 2c_i \quad \text{and} \quad S_{i+1}''(x_{i+1}) = 2c_{i+1}$$

Also

$$S_i''(x_{i+1}) = 2c_i + 6d_i(x_{i+1} - x_i)$$

Therefore, from equation (8), $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$, we get

$$S_i''(x_{i+1}) = 2c_i + 6d_i(x_{i+1} - x_i) = 2c_{i+1} \qquad (12)$$

We will use equation (12) presently.

Let us write our equations (10), (11) and (12) again:-

$$a_{i+1} = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 \qquad (10)$$

$$b_{i+1} = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 \qquad (11)$$

$$c_{i+1} = c_i + 3d_i(x_{i+1} - x_i) \qquad (12)$$

We can manipulate these three equations to eliminate three of the variables. In that way we end up with one equation containing one unknown variable. Before we do that let us just define

$$h_i = x_{i+1} - x_i \qquad (13)$$

So that

$$a_{i+1} = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \qquad (10)$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \qquad (11)$$

$$c_{i+1} = c_i + 3d_i h_i \qquad (12)$$

From equation (12) we get

$$d_i = \frac{(c_{i+1} - c_i)}{3h_i} \qquad (14)$$

So equations (10) and (11) now become

$$a_{i+1} = a_i + b_i h_i + c_i h_i^{\,2} + \frac{h_i^{\,2}}{3}\left(c_{i+1} - c_i\right)$$

or

$$a_{i+1} = a_i + b_i h_i + \frac{h_i^{\,2}}{3}\left(2c_i + c_{i+1}\right) \qquad (15)$$

$$b_{i+1} = b_i + 2c_i h_i + h_i\left(c_{i+1} - c_i\right)$$

or

$$b_{i+1} = b_i + h_i\left(c_i + c_{i+1}\right) \qquad (16)$$

Now if we can find expressions for b_i and b_{i+1} in equation (16) then we will end up with an equation which has only the c coefficients as unknowns. We first rearrange equation (15) to make b_i the subject to give

$$b_i = \frac{1}{h_i}\left(a_{i+1} - a_i\right) - \frac{h_i}{3}\left(2c_i + c_{i+1}\right) \qquad (17)$$

Next we increase the index of equation (17) by 1 to give

$$b_{i+1} = \frac{1}{h_{i+1}}\left(a_{i+2} - a_{i+1}\right) - \frac{h_{i+1}}{3}\left(2c_{i+1} + c_{i+2}\right) \qquad (18)$$

We can now use equations (17) and (18) to replace b_i and b_{i+1} in equation (16) to give

$$\frac{1}{h_{i+1}}\left(a_{i+2} - a_{i+1}\right) - \frac{h_{i+1}}{3}\left(2c_{i+1} + c_{i+2}\right) = \frac{1}{h_i}\left(a_{i+1} - a_i\right) - \frac{h_i}{3}\left(2c_i + c_{i+1}\right) + h_i\left(c_i + c_{i+1}\right) \qquad (19)$$

We can rearrange this in to multiples of c_{i+2}, c_{i+1} and c_i to give

$$\frac{h_i}{3}c_i + \left(\frac{2}{3}h_i + \frac{2}{3}h_{i+1}\right)c_{i+1} + \frac{h_{i+1}}{3}c_{i+2} = \frac{1}{h_{i+1}}\left(a_{i+2} - a_{i+1}\right) - \frac{1}{h_i}\left(a_{i+1} - a_i\right)$$

Multiply this by 3 to get, finally

$$h_i c_i + 2\left(h_i + h_{i+1}\right)c_{i+1} + h_{i+1}c_{i+2} = \frac{3}{h_{i+1}}\left(a_{i+2} - a_{i+1}\right) - \frac{3}{h_i}\left(a_{i+1} - a_i\right) \qquad (20)$$

So equation (20) only has the c coefficients as unknowns, as we already know that the $a_i = y_i$ (for i = 0 to 6, for the six x,y data points).

	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

For our trial problem we have the six sets of x,y data points given in the table above, which is shown again here.

We can write out equation (20) a sufficient number of times to enable us to calculate c_0 to c_5, i.e. six values

For i = 0: $$h_0c_0 + 2(h_0 + h_1)c_1 + h_1c_2 = \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0)$$

For i = 1: $$h_1c_1 + 2(h_1 + h_2)c_2 + h_2c_3 = \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1)$$

For i = 2: $$h_2c_2 + 2(h_2 + h_3)c_3 + h_3c_4 = \frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2)$$

For i = 3: $$h_3c_3 + 2(h_3 + h_4)c_4 + h_4c_5 = \frac{3}{h_4}(a_5 - a_4) - \frac{3}{h_3}(a_4 - a_3)$$

From condition V (equation (9)) and $S_i''(x_i) = 2c_i$ (from the top of page 64)

$$S''(x_0) = 0$$

and $$S''(x_n) = 0\,.$$

In other words $c_0 = 0$ and $c_n = 0$ ($c_5 = 0$, in our case) we can rewrite the above equations as

For i = 0: $$2(h_0 + h_1)c_1 + h_1c_2 = \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0)$$

For i = 1: $$h_1c_1 + 2(h_1 + h_2)c_2 + h_2c_3 = \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1)$$

For i = 2: $$h_2c_2 + 2(h_2 + h_3)c_3 + h_3c_4 = \frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2)$$

For i = 3: $$h_3c_3 + 2(h_3 + h_4)c_4 = \frac{3}{h_4}(a_5 - a_4) - \frac{3}{h_3}(a_4 - a_3)$$

So we have eliminated one term from the left hand side of both the first equation and the last equation. These four equations can be solved using a matrix

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 \\ 0 & 0 & h_3 & 2(h_3+h_4) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2) \\ \frac{3}{h_4}(a_5 - a_4) - \frac{3}{h_3}(a_4 - a_4) \end{pmatrix}. \qquad (21)$$

We know that $h_0 = x_1 - x_0$ etc and that $a_0 = y_0$ etc. So we can fill in the left hand matrix and the right hand column vector (containing the a coefficients). We can then form an augmented matrix. We can solve the augmented matrix for the c coefficients using Gaussian Elimination. We can then use equation (14) to find the d coefficients

$$d_i = \frac{(c_{i+1} - c_i)}{3h_i} \quad (14)$$

and we can use equation (17) to find the b coefficients

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1}) \quad (17)$$

We will cover how to use these results to form the cubic equations, like equation (1), in next weeks lecture.

Tutorial

(1) Before you can do this tutorial you must have completed the tutorial questions from last week. You will need the code for the Gaussian Eliminiation in `gauss2.cpp` and `gauss3.cpp`.
(2) In question (4) of last weeks tutorial you were asked to modify `gauss3.cpp` so as to take a 4 x 5 augmented matrix. Maybe call this code `gauss4.cpp`. Try to get to that stage so that you can try this Cubic Spline work.
(3) Using the x and y values in the table, in the text, find the h's and the a's in the matrix system of equation (21). Then write out the augmented matrix for this system.
(4) Put the augmented matrix from question (3) into `gauss4.cpp` and solve it for the c values.