Examples solving an ODE using finite elements method. Mathematica and Matlab implementation and animation

Nasser M. Abbasi

Sept 28,2006 Compiled on September 9, 2023 at 3:03pm

Contents

This report is a basic review of weighted residual methods and FEM. Some basic differential equations are used to illustrate the method. Mathematica and Matlab code written to solve numerically a first and second order ODE using FEM.

1 Introduction

This is a basic review of Finite Elements Methods from Mathematical point of view with examples of how it can be used to numerically solve first and second order ODE's. Currently I show how to use FEM to solve first and second order ODE. I am also working on a detailed derivation and implementation using FEM to solve the 2D Poisson's equation but this work is not yet completed.

FEM is a numerical method for solving differential equations (ordinary or partial). It can also be used to solve non-linear differential equations but I have not yet studied how this is done. FEM is a more versatile numerical method than the finite difference methods for solving differential equations as it supports more easily different types of geometry and boundary conditions, in addition the solution of the differential equation found using FEM can be used at any point in the domain and not just on the grid points as the case is with finite difference methods. On the other hand FEM is more mathematically complex method, and its implementation is not as straight forward as with finite difference methods.

Considering only ordinary differential equations with constant coefficients over the x domain (real

line).

$$a\frac{dy}{dx} + b\ y(x) = f(x)$$
$$a\frac{dy}{dx} + b\ y(x) - f(x) = 0$$

defined over $0 \leq x \leq 1$ with the boundary condition $y(0) = y_0$. In the above, only when $y(x)$ is the exact solution, call it $y_e(x)$, do we have the above identity to be true.

In other words, only when $y = y_e$ we can write that $a\frac{dy_e}{dx} + b\ y_e(x) - f(x) = 0$. Such a differential equations can be represented as an operator L

$$L := (y, x) \rightarrow a\frac{dy}{dx} + by(x) - f(x) \tag{1}$$

If we know the exact solution, call it $y_e(x)$ then we write

$$L(y_e, x) \rightarrow a\frac{dy_e}{dx} + by_e(x) - f(x)$$
$$\rightarrow 0$$

FEM is based on the weighted residual methods (WRM) where we assume that the solution of an differential equation is the sum of weighted basis functions represented by the symbol ϕ_i in here. This is in essence is similar to Fourier series, where we represent a function as a weighted sums of series made up of the basis functions which happened to be in that case the sin and cosine functions.

So the first step in solving the differential equation is to assume that the solution, called $\tilde{y}(x)$, can be written as

$$\tilde{y}(x) = \sum_{j=1}^{N} q_j\ \phi_j(x)$$

Where q_j are unknown coefficients (the weights) to be determined. Hence the main computational part of FEM will be focused on determining these coefficients.

When we substitute this assumed solution in the original ODE such as shown in the above example, equation (1) now becomes

$$L(\tilde{y}, x) = a\frac{d\tilde{y}}{dx} + b\tilde{y}(x) - f(x)$$
$$= R(x)$$

Where $R(x)$ is called the differential equation residual, which is a function over x and in general will not be zero due to the approximate nature of our assumed solution. Our goal is to to determine the coefficients q_j which will make $R(x)$ the minimum over the domain of the solution.

The optimal case is for $R(x)$ to be zero over the domain. One method to be able to achieve this is by forcing $R(x)$ to meet the following requirement

$$\int_{\Omega} R(x)\ \ v_i(x)\ \ dx = 0$$

for all possible sets of function $v_i(x)$ which are also defined over the same domain. The functions $v_i(x)$ are linearly independent from each others. If we can make $R(x)$ satisfy the above for each one of these functions, then this implies that $R(x)$ is zero. And the solution $\tilde{y}(x)$ will be as close as possible to the exact solution. We will find out in FEM that the more elements we use, the closer to the exact solution we get. This property of convergence when it comes to FEM is important, but not analyzed here.

Each one of these functions $v_i(x)$ is called a test function (or a weight function), hence the name of this method.

In the Galerkin method of FEM, the test functions are chosen from the same class of functions as the trial functions as will be illustrated below.

By making $R(x)$ satisfy the above integral equation (called the weak form of the original differential equation) for N number of test functions, where N is the number of the unknown coefficients q_i, then we obtain a set of N algebraic equations, which we can solve for q_i.

The above is the basic outline of all methods based on the weighted residual methods. The choice of the trial basis functions, and the choice of the test functions, determine the method used. Different numerical schemes use different types of trial and test functions.

In the above, the assumed solution $\tilde{y}(x)$ is made up of a series of trial functions (the basis). This solution is assumed to be valid over the whole domain. This is called a global trial function. In methods such as Finite Elements and Finite volume, the domain itself is discretized, and the assumed solution is made up of a series of solutions, each of which is defined over each element resulting from the discretization process.

In addition, in FEM, the unknown coefficients, called q_i above, have a physical meaning, they are taken as the solution values at each node. The trial functions themselves are generated by using polynomial interpolation between the nodal values. The polynomial can be linear, quadratic or cubic polynomial or higher order.

Lagrangian interpolation method is normally used for this step. The order of the polynomial is determined by the number of unknowns at the nodes. For example, if our goal is to determine the axial displacement at each node, then we have 2 unknowns, one at each end of the element. Hence a linear interpolation will be sufficient in this case, since a linear polynomial $a_0 + a_1 x$ contain 2 unknowns, the a_1 and a_0. If in addition to the axial displacement, we wish to also solve for the rotation at each end of the element, hence we have a total of 4 unknowns, 2 at each end of the element, which are the displacement and the rotation.

Hence in this case the minimum interpolating polynomial needed will be a cubic polynomial $a_0 + a_1 x + a_2 x^2 + a_3 x^3$. In the examples below, we assume that we are only solving for axial displacement, hence a linear polynomial will be sufficient.

At first, we will work with global trial functions to illustrate how to use weighted residual method.

2 Weighted Residual method. Global trial functions.

The best way to learn how to use WRM is by working over and programming some examples.

We analyze the solution in terms of errors and the effect of changing N on the result.

2.1 First example. First order ODE

Given the following ODE

$$\frac{dy}{dx} - y(x) = 0$$

defined over $0 \leq x \leq 1$ with the boundary condition $y(0) = 1$, we wish to solve this numerically using the WRM. This ODE has an exact solution of $y = e^x$.

The solution using WRM will always follow these steps.

STEP 1

Assume a solution that is valid over the domain $0 \leq x \leq 1$ to be a series solution of trial (basis) functions. We start by selecting a trial functions. The assumed solution takes the form of

$$\tilde{y}(x) = \tilde{y}_0 + \sum_{j=1}^{N} q_j \ \phi_j(x)$$

Where $\phi_j(x)$ is the trial function which we have to choose, and q_j are the N unknown coefficients to be determined subject to a condition which will be shown below. $\tilde{y}_0$ is the assumed solution which needs to be valid only at the boundary conditions. Hence in this example, since we are given that the solution must be 1 at the initial condition $x = 0$, then $\tilde{y}_0 = 1$ will satisfy this boundary condition. Hence our trial solution is

$$\tilde{y} = 1 + \sum_{j=1}^{N} q_j \ \phi_j(x)$$

STEP 2

Now we decide on what trial function $\phi(x)$ to use. For this example, we can select the trial functions to be polynomials in x or trigonometric functions. Let us choose a polynomial $\phi_j(x) = x^j$, hence our assumed solution becomes

$$\tilde{y} = 1 + \sum_{j=1}^{N} q_j \ x^j \tag{1}$$

Now we need to determine the coefficients q_j, and then our solution will be complete. This is done in the following step.

STEP 3

Substituting the above assumed solution back into the original ODE, we obtain the residual $R(x)$

$$\frac{d\tilde{y}}{dx} - \tilde{y}(x) = R(x)$$

$R(x)$ is the ODE residual. This is the error which will result when the assumed solution is used in place of the exact solution.

Hence from (1), we find the residual to be

$$\begin{aligned} R(x) &= \frac{d}{dx}\left(1+\sum_{j=1}^{N} q_j\ x^j\right) - \left(1+\sum_{j=1}^{N} q_j\ x^j\right) \\ &= \sum_{j=1}^{N} q_j\ jx^{j-1} - \left(1+\sum_{j=1}^{N} q_j\ x^j\right) \\ &= -1+\sum_{j=1}^{N} q_j\ \left(jx^{j-1}-x^j\right) \end{aligned} \tag{2}$$

Our goal now is to reduce this residual to minimum. The way we achieve this is by requiring that the residual satisfies the following integral equation

$$\int_0^1 v_i(x)\, R(x)\, dx = 0 \qquad i = 1 \cdots N \tag{3}$$

The above is a set of N equations. The integration is carried over the whole domain, and $v_i(x)$ is a weight (test) function, which we have to also select. Depending on the numerical scheme used, the test function will assume different forms.

For the Galerkin method, we select the test function to be from the same family of functions as the trial (basis) functions. Hence in this example, let us select the test function to the following polynomial

$$v_i(x) = x^{i-1} \tag{4}$$

STEP 4

We now choose a value for N and solve the set of equations generated from (3). Let us pick $N = 3$, hence $R(x)$ becomes

$$\begin{aligned} R(x) &= -1+\sum_{j=1}^{3} q_j\left(jx^{j-1}-x^j\right) \\ &= -1+q_1\left(1-x^1\right)+q_2\ \left(2x-x^2\right)+q_3\left(3x^2-x^3\right) \end{aligned}$$

Substituting the above in (3) gives

$$\int_{x=0}^{x=1} x^{i-1} R(x)\, dx = 0 \qquad i = 1 \cdots N$$

$$\int_{x=0}^{x=1} x^{i-1}\left(-1+q\ \left(1-x^1\right)+q_2\left(2x-x^2\right)+q_3\ \left(3x^2-x^3\right)\right) dx = 0 \qquad i = 1 \cdots 3$$

The above generates $N=3$ equations to solve. They are

$$\begin{aligned}
\int_{x=0}^{x=1} \left(-1+q_1\ \left(1-x^1\right)+q_2\ \left(2x-x^2\right)+q_3\ \left(3x^2-x^3\right)\right) dx &= 0 \qquad i=1 \\
\int_{x=0}^{x=1} x\left(-1+q_1\ \left(1-x^1\right)+q_2\ \left(2x-x^2\right)+q_3\ \left(3x^2-x^3\right)\right) dx &= 0 \qquad i=2 \\
\int_{x=0}^{x=1} x^2\left(-1+q_1\ \left(1-x^1\right)+q_2\ \left(2x-x^2\right)+q_3\ \left(3x^2-x^3\right)\right) dx &= 0 \qquad i=3
\end{aligned}$$

Performing the integration above, gives the following 3 equations

$$\begin{aligned}
-1-q_1\left(-\frac{1}{2}\right)-q_2\left(-\frac{2}{3}\right)-q_3\left(-\frac{3}{4}\right) &= 0 \\
-\frac{1}{2}-q_1\left(-\frac{1}{6}\right)-q_2\left(-\frac{5}{12}\right)-q_3\left(-\frac{11}{20}\right) &= 0 \\
-\frac{1}{3}-q_1\left(-\frac{1}{12}\right)-q_2\left(-\frac{3}{10}\right)-q_3\left(-\frac{13}{30}\right) &= 0
\end{aligned}$$

Which can be written in matrix form as

$$\begin{bmatrix} \frac{1}{2} & \frac{2}{3} & \frac{3}{4} \\ \frac{1}{6} & \frac{5}{12} & \frac{11}{20} \\ \frac{1}{12} & \frac{3}{10} & \frac{13}{30} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$$

The solution is

$$q_1=\frac{72}{71}, q_2=\frac{30}{71}, q_3=\frac{20}{71}$$

Hence our assumed series solution is now complete, using the above coefficients, and from equation (1) we write

$$\begin{aligned}
\tilde{y} &= 1+\sum_{j=1}^{N} q_j\ x^j \\
\tilde{y} &= 1+q_1\ x+q_2 x^2+q_3 x^3
\end{aligned}$$

Hence

$$\tilde{y}(x)=1+\frac{72}{71}x+\frac{30}{71}x^2+\frac{20}{71}x^3$$

Let use compare the above solution to the exact solution $y=e^x$ by comparing the values of the solution over a number of points. This is done using the following small Mathematica code

```
In[77]:= Remove["Global`*"]
         exact[x_] := Exp[x]
         approx[x_] := 1 + 72/71 x + 30/71 x^2 + 20/71 x^3
         error[x_] := Abs[exact[x] - approx[x]]
         s = Table[{x, exact[x], approx[x], error[x]}, {x, 0, 1, .1}];
         TableForm[s, TableHeadings → {None, {"x", "exact", "approx", "error"}}]
```

Out[82]//TableForm=

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.10592	0.000744575
0.2	1.2214	1.22197	0.000569073
0.3	1.34986	1.34986	3.47354×10^{-7}
0.4	1.49182	1.49127	0.000557092
0.5	1.64872	1.64789	0.000833947
0.6	1.82212	1.82141	0.00071035
0.7	2.01375	2.01352	0.000231581
0.8	2.22554	2.22592	0.000374564
0.9	2.4596	2.46028	0.000678579
1.	2.71828	2.71831	0.0000280307

Figure 1: Using $N = 3$

To make this more useful, we can examine how the error changes as N changes. The following Mathematica code determines the solution and calculates the same table as above for $N = 1 \cdots 5$

```
In[615]:=
Remove["Global`*"]
ode[y_, x_] := D[y[x], x] - y[x];
sol = Flatten[DSolve[{ode[y, x] == 0, y[0] == 1}, y'[x], x]];
exactSolution[x_] = y[x] /. sol;
trialFunction[x_, j_] := x^j ;
weight[x_, j_] := x^(j-1) ;
eq[i_, residue_] := ∫_from^to weight[x, i] residue dx == 0;
solution[maxN_, from_, to_] := Module[{coeff, residue, j, a},
   coeff = Array[a, maxN];
   yapprox[x_] := 1 + ∑_(j=1)^maxN coeff[[j]] trialFunction[x, j];
   residue = ode[yapprox, x];
   s = Table[eq[j, residue], {j, 1, maxN}];
   sol = Flatten[Solve[s, coeff]]
  ];

from = 0;
to = 1;
approxSolution[x_, sol_] := 1 + ∑_(j=1)^maxN sol[[j, 2]] trialFunction[x, j];
error[x_, sol_] := Abs[exactSolution[x] - approxSolution[x, sol]];
Do[{sol = solution[maxN, from, to];
   Print["N=", maxN, " Approx Solution is ", approxSolution[x, sol]];
   s = Table[{x, exactSolution[x], approxSolution[x, sol], error[x, sol]},
     {x, 0, 1, .1}];
   Print[TableForm[s, TableHeadings → {None, {"x", "exact", "approx", "error"}}]];
  }, {maxN, 1, 5}];
```

Figure 2: Code used for $N = 5$

N=1 Approx Solution is $1+2x$

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.2	0.0948291
0.2	1.2214	1.4	0.178597
0.3	1.34986	1.6	0.250141
0.4	1.49182	1.8	0.308175
0.5	1.64872	2.	0.351279
0.6	1.82212	2.2	0.377881
0.7	2.01375	2.4	0.386247
0.8	2.22554	2.6	0.374459
0.9	2.4596	2.8	0.340397
1.	2.71828	3.	0.281718

N=2 Approx Solution is $1+\frac{6x}{7}+\frac{6x^2}{7}$

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.09429	0.0108852
0.2	1.2214	1.20571	0.0156885
0.3	1.34986	1.33429	0.0155731
0.4	1.49182	1.48	0.0118247
0.5	1.64872	1.64286	0.00586413
0.6	1.82212	1.82286	0.000738342
0.7	2.01375	2.02	0.00624729
0.8	2.22554	2.23429	0.00874479
0.9	2.4596	2.46571	0.00611117
1.	2.71828	2.71429	0.00399611

N=3 Approx Solution is $1+\frac{72x}{71}+\frac{30x^2}{71}+\frac{20x^3}{71}$

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.10592	0.000744575
0.2	1.2214	1.22197	0.000569073
0.3	1.34986	1.34986	3.47354×10^{-7}
0.4	1.49182	1.49127	0.000557092
0.5	1.64872	1.64789	0.000833947
0.6	1.82212	1.82141	0.00071035
0.7	2.01375	2.01352	0.000231581
0.8	2.22554	2.22592	0.000374564
0.9	2.4596	2.46028	0.000678579
1.	2.71828	2.71831	0.0000280307

N=4 Approx Solution is $1+\frac{1000x}{1001}+\frac{510x^2}{1001}+\frac{20x^3}{143}+\frac{10x^4}{143}$

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.10514	0.0000290599
0.2	1.2214	1.22141	7.83125×10^{-6}
0.3	1.34986	1.3499	0.0000382953
0.4	1.49182	1.49186	0.0000354422
0.5	1.64872	1.64873	5.00303×10^{-6}
0.6	1.82212	1.82209	0.0000288903
0.7	2.01375	2.01371	0.0000394208
0.8	2.22554	2.22553	0.000014455
0.9	2.4596	2.45963	0.0000242615
1.	2.71828	2.71828	1.10177×10^{-7}

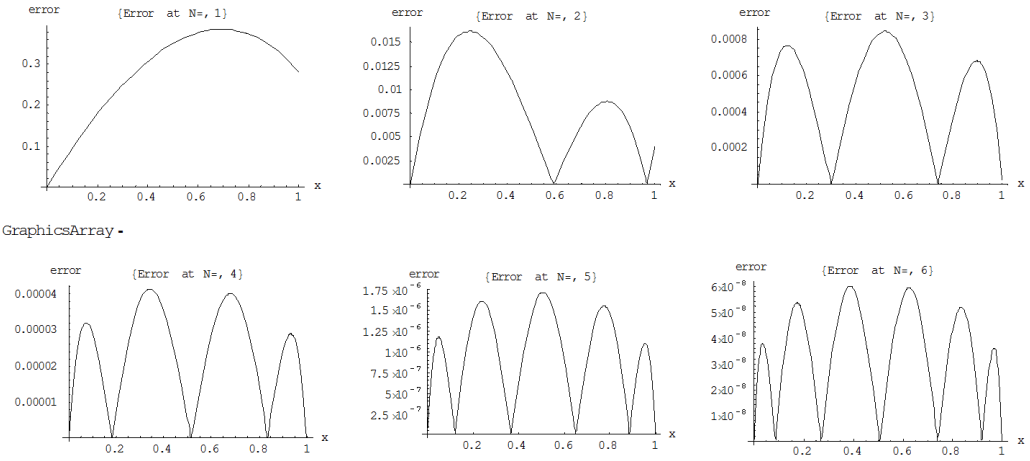
N=5 Approx Solution is $1+\frac{18090x}{18089}+\frac{9030x^2}{18089}+\frac{3080x^3}{18089}+\frac{630x^4}{18089}+\frac{252x^5}{18089}$

x	exact	approx	error
0	1	1	0
0.1	1.10517	1.10517	4.8554×10^{-7}
0.2	1.2214	1.2214	1.42918×10^{-6}
0.3	1.34986	1.34986	1.13938×10^{-6}
0.4	1.49182	1.49183	6.37093×10^{-7}
0.5	1.64872	1.64872	1.71012×10^{-6}
0.6	1.82212	1.82212	9.14353×10^{-7}
0.7	2.01375	2.01375	8.89238×10^{-7}
0.8	2.22554	2.22554	1.46473×10^{-6}
0.9	2.4596	2.4596	3.20763×10^{-7}
1.	2.71828	2.71828	2.76651×10^{-10}

Figure 3: Result for $N = 5$

This code below plots the absolute error as N changes. Notice that the number of peaks in the error plot is also N which is the polynomial order (the trial solution) used to approximate the exact solution, which is to be expected.

```
In[738]:= p = {};
Do[{sol = solution[maxN, from, to];
   p = Append[p, Plot[error[x, sol], {x, 0, 1}, PlotRange → All, AxesLabel → {"x", "error"}, PlotLabel → {"Error at N=", maxN},
      DisplayFunction → Identity]];
  }, {maxN, 1, 6}];
Show[GraphicsArray[ {p[[1]], p[[2]], p[[3]] } ]]
Show[GraphicsArray[ {p[[4]], p[[5]], p[[6]] } ]]
```




```
Out[740]= - GraphicsArray -
```


Figure 4: Error using $N = 5$

2.2 Second example. 4th order ODE

Now we will use a more complex example and repeat the above steps. We now want to numerically solve the following

$$\frac{d^4y}{dx^4} + y(x) = 1$$

defined over $0 \leq x \leq 1$ with the boundary conditions $y(0) = 0, y(1) = 0, y'(0), y'(1) = 0$. This problem is taken from Professor S.N.Atluri text book 'Methods of computer modeling in engineering and the sciences' Volume 1, page 47-50. Professor Atluri used a trigonometric functions for the trial function

$$\tilde{y} = \sum_{i=1}^{N} q_i \sin\left((2i-1)\pi x\right)$$

Which already satisfies the boundary conditions. For the test function, the same function as above is used hence the test (weight) function is

$$v_j(x) = \sin\left((2j-1)\pi x\right)$$

The book above then reduces the residual equation to a symmetric form by doing integration by parts before solving it for the coefficients. In here, we will use the unsymmetrical weak form and compare the results with those shown in the above textbook. We now start again with the same steps as we did in the above example.

step 1

Selecting the trial solution.

$$\tilde{y}(x) = \tilde{y}_0 + \sum_{j=1}^{N} q_j \ \phi_j(x)$$

$\tilde{y}_0 = 0$ as this will satisfy the boundary conditions. Hence the trial solution is

$$\tilde{y}(x) = \sum_{j=1}^{N} q_j \phi_j(x)$$

step 2

Selecting trial basis function $\phi_j(x)$. As mentioned above, we select $\phi_j(x) = \sin\left((2j-1)\,\pi x\right)$, hence the trial solution is

$$\tilde{y} = \sum_{j=1}^{N} q_j \sin\left((2j-1)\,\pi x\right)$$

step 3

Substituting the above assumed solution into the original ODE, gives the differential equation residual $R(x)$

$$\frac{d^4\tilde{y}}{dx^4} + \tilde{y}(x) - 1 = R(x)$$

$$\frac{d^4}{dx^4}\left(\sum_{j=1}^{N} q_j \ \phi_j(x)\right) + \left(\sum_{j=1}^{N} q_j \ \phi_j(x)\right) - 1 = R(x)$$

Notice the requirement above that the trial basis functions must be 4 times differentiable, which is the case here. From above we obtain

$$R(x) = \left(\sum_{j=1}^{N} \left(1 + \left((2j-1)\,\pi\right)^4\right) q_j \sin\left((2j-1)\,\pi x\right)\right) - 1$$

Our goal now is to reduce this residual to minimum. The way we achieve this is by requiring that the residual satisfies the following weak form integral equation

$$\int_{\Omega} v_i(x)\, R(x)\, dx = 0 \qquad i = 1 \cdots N \tag{3}$$

The above is a set of N equations. The integration is carried over the whole domain, and $v_i(x)$ is a weight (test) function, which we have to also select. As mentioned above, in this problem we select the test function to be

$$v_i(x) = \sin\left((2i-1)\,\pi x\right) \tag{4}$$

step 4

Deciding on a value for N and solving the set of equations generated from (3). Let us pick $N = 3$, hence $R(x)$ becomes

$$\begin{aligned} R(x) &= \left(\sum_{j=1}^{3} \left(1 + \left((2j-1)\,\pi\right)^4\right) \left(q_j \sin\left(2j-1\right)\pi x\right)\right) - 1 \\ &= \left(\left(1+\pi^4\right)\left(q_1 \sin \pi x\right) + \left(1 + (3\pi)^4\right)\left(q_2 \sin\left(3\pi x\right)\right) + \left(1 + (5\pi)^4\right)\left(q_3 \sin 5\pi x\right)\right) - 1 \end{aligned}$$

Hence (3) becomes

$$\int_{\Omega} v_i(x)\, R(x)\, dx = 0 \qquad i$$

$$\int_0^1 (\sin(2i-1)\pi x)\, R(x)\, dx = 0 \qquad i$$

$$\int_0^1 (\sin(2i-1)\pi x) \left\{ \left(\left(1+\pi^4\right)(q_1 \sin \pi x) + \left(1+(3\pi)^4\right)(q_2 \sin 3\pi x) + \left(1+(5\pi)^4\right)(q_3 \sin 5\pi x) \right) - 1 \right\} dx = 0 \qquad i$$

The above generates N equations to solve for the coefficients q_i

$$\int_0^1 (\sin \pi x) \left\{ \left(\left(1+\pi^4\right)(q_1 \sin \pi x) + \left(1+(3\pi)^4\right)(q_2 \sin 3\pi x) + \left(1+(5\pi)^4\right)(q_3 \sin 5\pi x) \right) - 1 \right\} dx = 0 \qquad i =$$

$$\int_0^1 (\sin 3\pi x) \left\{ \left(\left(1+\pi^4\right)(q_1 \sin \pi x) + \left(1+\quad(3\pi)^4\right)(q_2 \sin(3\pi x)) + \left(1+(5\pi)^4\right)(q_3 \sin 5\pi x) \right) - 1 \right\} dx = 0 \qquad i =$$

$$\int_0^1 (\sin 5\pi x) \left\{ \left(\left(1+\pi^4\right)(q_1 \sin \pi x) + \left(1+\quad(3\pi)^4\right)(q_2 \sin 3\pi x) + \left(1+(5\pi)^4\right)(q_3 \sin 5\pi x) \right) - 1 \right\} dx = 0 \qquad i =$$

Carrying the integration above and simplifying and solving for q_i gives the numerical solution.

This below is a Mathematica code which solves this problem for different N values, and compares the error as N changes. The error shown is the percentage error in the solution (approximate compared to exact) for up to $N = 10$. The result below agrees well with the result in Professor's Atluri textbook.

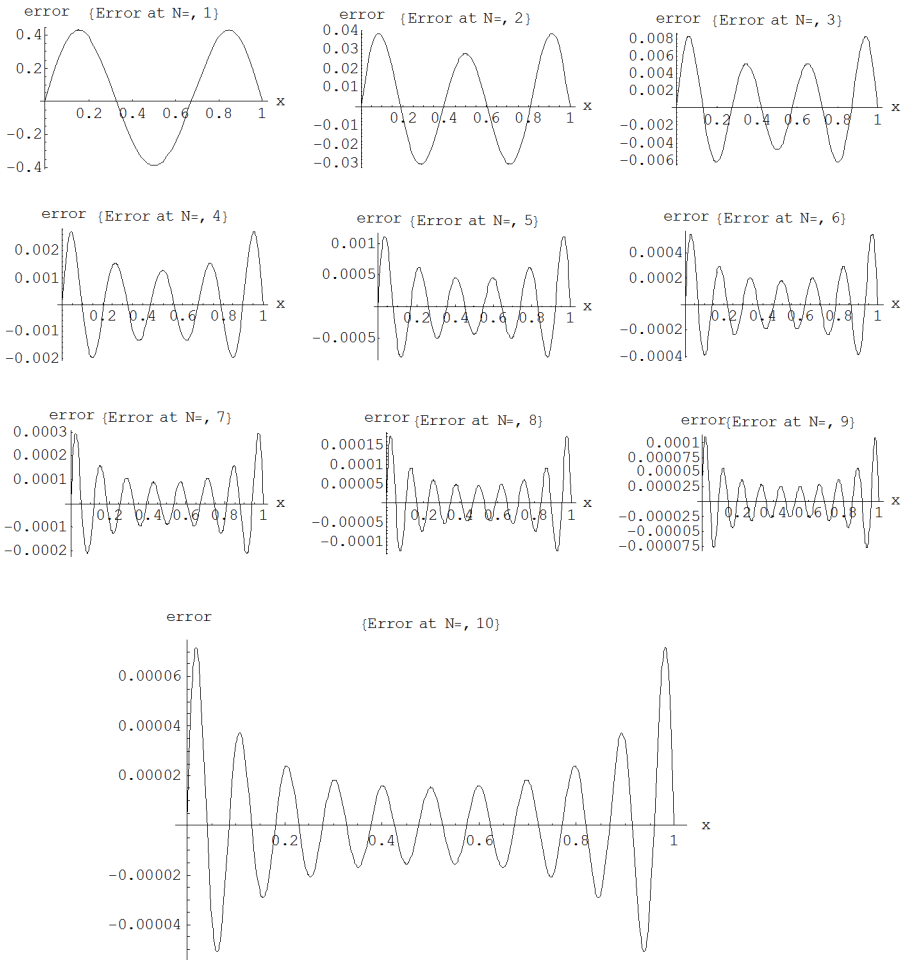


Figure 5: Error for different N using FEM for second order ODE

3 Finite element method

3.1 Example one. First order ODE, linear interpolation

Let us first summarize what we have done so far. Given a differential equation defined over domain Ω, we assume its solution to be of the form $\tilde{u}(x) = \sum_{j=1}^{N} q_j \ \phi_j(x)$.

The function $\phi_j(x)$ is called the j^{th} basis function. $q_j \ \phi_j(x)$ is called a trial function.

The function $\phi_j(x)$ is made up of functions called the shape functions N_k as they are normally called in structural mechanics books.

q_j are the unknown coefficients which are determined by solving N set of equations generated by setting N integrals of the form $\int_{\Omega} R(x) \ v_i(x) \ dx$ to zero. Where $R(x)$ is the differential equation residual. In all what follows N is the taken as the number of nodes.

In FEM, we also carry the same basic process as was described above, the differences are the following:

Now we divide the domain itself into a number of elements. Earlier we did not do this.

Next, the $\phi_j(x)$ function is found by assuming the solution to be an interpolation between the nodes of the element. The solution values at the nodes are the q_i and are of course unknown except at the boundaries as given by the problem.

We start by deciding on what interpolation between the nodes to use. We will use polynomial interpolation here. Then $q_j\phi_j(x)$ will become the interpolation function.

In addition, the coefficients q_j represent the solution at the node j. These are the unknowns, which we will solve for by solving the weak form integral equation as many times as there are unknowns to solve for.

By solving for the nodal values, we can then use the interpolating function again to find the solution at any point between the nodes.

This diagram illustrates the above, using the first example given above to solve a differential equation $\frac{du}{dx} - u(x) = 0$ with the given boundary condition of $u(0) = 1$ and defined over $0 \leq x \leq 1$

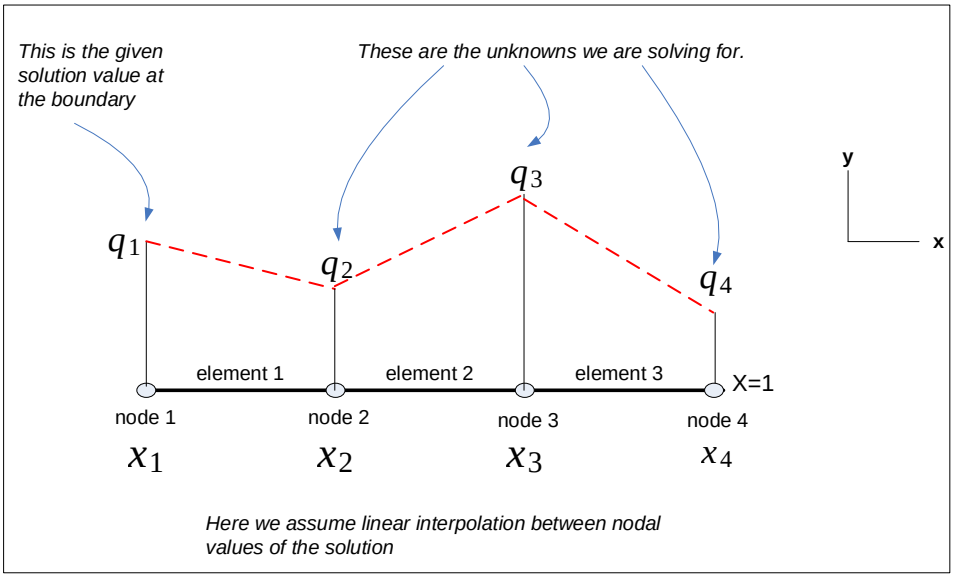


Figure 6: Finite elements with linear interpolation

Using linear interpolation, the solution $u(x)$ when x is located inside first element, is found from

$$u(x) = q_1 + \text{slope } (x - x_0)$$

But the slope of the linear interpolating line over the first element is $\frac{q_2 - q_1}{x_2 - x_1}$, hence the above becomes

$$\begin{aligned} u(x) &= q_1 + \frac{q_2 - q_1}{x_2 - x_1}(x - x_1) \\ &= q_1 \frac{(x_2 - x)}{(x_2 - x_1)} + q_2 \frac{(x - x_1)}{(x_2 - x_1)} \end{aligned}$$

The above is the linear interpolating polynomial. We could also have used the formula of Lagrangian interpolation to arrive at the same result.

The above is the approximate solution which is valid over the first element only. Using superscript to indicate the element number, and assuming we have equal division between nodes of length say h (i.e. element length is h) then we write

$$u^1(x) = q_1 \frac{(x_2 - x)}{h} + q_2 \frac{(x - x_1)}{h}$$

Again, the above is valid for $x_1 \le x \le x_2$. We now do the same for the second element

$$u^2(x) = q_2 \frac{(x_3 - x)}{h} + q_3 \frac{(x - x_2)}{h}$$

The above is valid for $x_2 \le x \le x_3$. And finally for the 3rd element

$$u^3(x) = q_3 \frac{(x_4 - x)}{h} + q_4 \frac{(x - x_3)}{h}$$

The above is valid for $x_2 \le x \le x_3$. This is now illustrated in the following diagram

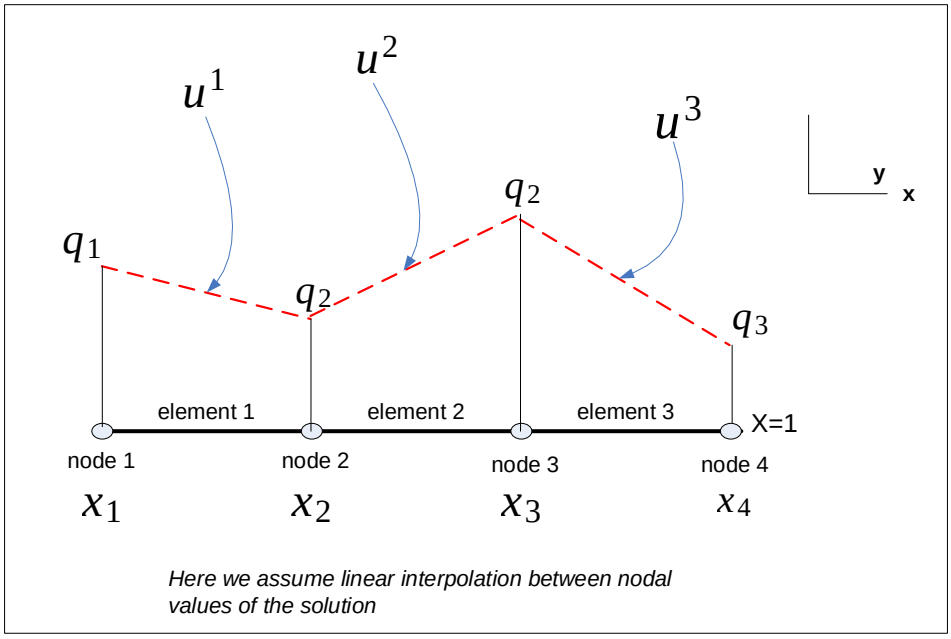


Figure 7: Finite elements with linear interpolation second diagram

Since our goal is to express the global approximate solution $u(x)$ as a series sum of basis functions each multiplied by q_j, we now rewrite each of the u^j to allow this, as follows

$$\begin{aligned} u^1(x) &= q_1 \overbrace{\frac{(x_2 - x)}{h}}^{N_1^1(x)} + q_2 \overbrace{\frac{(x - x_1)}{h}}^{N_2^1(x)} \\ &= q_1 N_1^1(x) + q_2 N_2^1(x) \end{aligned}$$

The above is valid for $x_1 \le x \le x_2$. Notice the use of the following notation: Since each element will have defined on it two shape functions, $N_1(x)$ and $N_2(x)$, one per node, then we use a superscript to indicate the element number. Hence for element 1, we will write its two shapes functions as $N_1^1(x)$ and $N_2^1(x)$.

We now do the same for the second element

$$\begin{aligned} u^2 &= q_2 \frac{\overbrace{(x_3-x)}^{N_1^2(x)}}{h} + q_3 \frac{\overbrace{(x-x_2)}^{N_2^2(x)}}{h} \\ &= q_2 N_1^2(x) + q_3 N_2^2(x) \end{aligned}$$

The above is valid for $x_2 \le x \le x_3$. And finally for the 3rd element

$$\begin{aligned} u^3 &= q_3 \frac{\overbrace{(x_4-x)}^{N_1^3(x)}}{h} + q_4 \frac{\overbrace{(x-x_3)}^{N_2^3(x)}}{h} \\ &= q_3 N_1^3 + q_4 N_2^3 \end{aligned}$$

The above is valid for $x_2 \le x \le x_3$. The global trial function is

$$\begin{aligned} u(x) &= u^1 + u^2 + u^3 \\ &= \left(q_1 N_1^1 + q_2 N_2^1\right) + \left(q_2 N_1^2 + q_3 N_2^2\right) + \left(q_3 N_1^3 + q_4 N_2^3\right) \\ &= q_1 N_1^1 + q_2\left(N_2^1 + N_1^2\right) + q_3\left(N_2^2 + N_1^3\right) + q_4\left(N_2^3\right) \end{aligned}$$

The shape function for node 1 is

$$\begin{aligned} \phi_1 &= N_1^1 \\ &= \frac{x_2 - x}{h} \end{aligned}$$

And the shape function for node 2 is

$$\begin{aligned} \phi_2 &= N_2^1 + N_1^2 \\ &= \frac{(x - x_1)}{h} + \frac{(x_3 - x)}{h} \end{aligned}$$

The shape function for node 3 is

$$\begin{aligned} \phi_3 &= N_2^2 + N_1^3 \\ &= \frac{(x - x_2)}{h} + \frac{(x_4 - x)}{h} \end{aligned}$$

The shape function for the last node is

$$\begin{aligned} \phi_4 &= N_2^3 \\ &= \frac{x - x_3}{h} \end{aligned}$$

We see that the shape function for any internal node is

$$\phi_j = \frac{x - x_{j-1}}{h} + \frac{x_{j+1} - x}{h}$$

The approximate solution is therefore

$$u(x) = \sum_{i=1}^{Number\ Nodes} q_i \phi_i$$

This completes the first part, which was to express the global approximate solution as sum of basis functions, each multiplied by an unknowns q coefficients.

The diagram below illustrates the above.

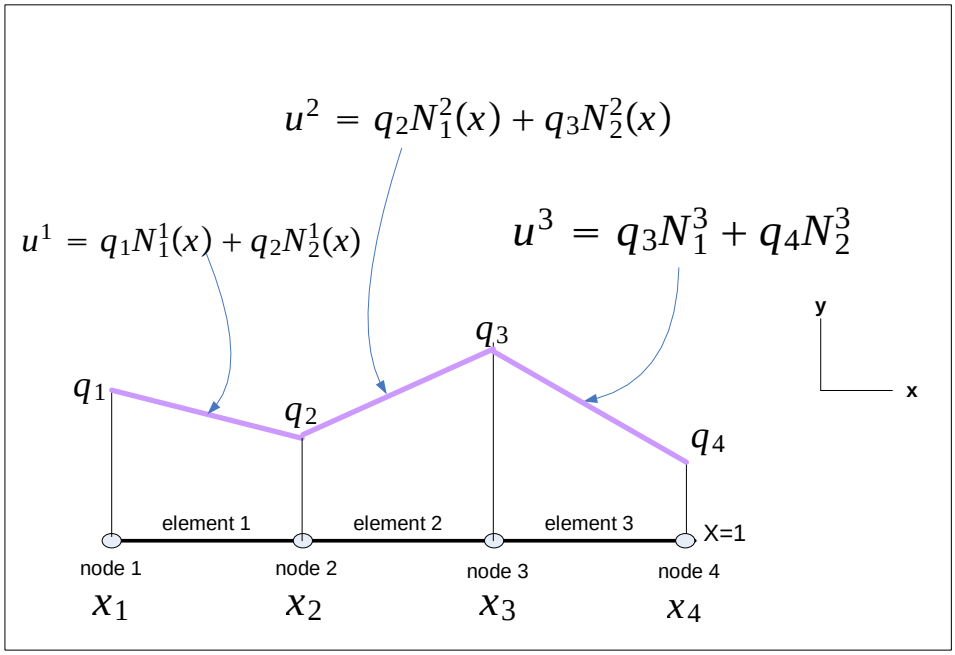


Figure 8: Showing solutions at each element

The diagram below illustrates the numbering used.

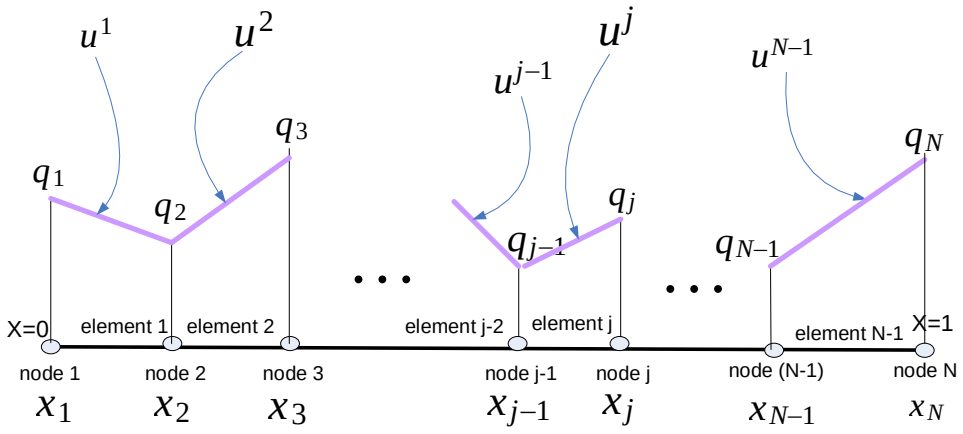


Figure 9: Numbering used for interpolation and elements

Next the residual $R(x)$ is found by substituting the above global solution into the original differential equation as we did before.

Let The differential equation to solve be

$$a\frac{du}{dx} + bu(x) = f(x)$$

Defined over $0 \leq x \leq 1$ with the initial condition $u(0) = u_0$.

N is the number of nodes, therefore the residual is

$$R(x) = a\frac{d}{dx}\sum_{i=1}^{N} q_i\phi_i + b\sum_{i=1}^{N} q_i\phi_i - f(x)$$

$$R(x) = \sum_{i=1}^{N} q_i\left(a\phi_i' + b\phi_i\right) - f(x)$$

The test functions are

$$v_j = \phi_j \qquad j = 1 \cdots N$$

The weak form of the differential equation is

$$\int_{x=x_1}^{x=x_N} \phi_j(x)\, R(x)\, dx = 0 \qquad j = 1 \cdots N$$

$$\int_{x=x_1}^{x=x_N} \phi_j(x) \left(\left[\sum_{i=1}^{N} q_i\left(a\phi_i' + b\phi_i\right) \right] - f(t) \right) dx = 0 \qquad j = 1 \cdots N$$

N equations are obtained from the above, which are solved for the q_i.

The derivatives of the shape functions are found first. Assuming in this example that the domain is divided into 3 elements results in

i	ϕ_i	ϕ_i'
1	$\frac{(x_2 - x)}{h}$	$\frac{-1}{h}$
2	$\left\{ \frac{(x - x_1)}{h}, \frac{(x_3 - x)}{h} \right\}$	$\left\{ \frac{1}{h}, \frac{-1}{h} \right\}$
3	$\frac{(x - x_2)}{h}$	$\frac{1}{h}$

In general, if there are N nodes, then for the first node

$$\phi_1 = \frac{(x_2 - x)}{h}$$

$$\phi_1' = \frac{-1}{h}$$

And for the last node

$$\phi_N = \frac{(x - x_{N-1})}{h}$$

$$\phi_N' = \frac{1}{h}$$

And for any node in the middle

$$\phi_i = \frac{(x - x_{i-1})}{h}$$
$$\phi_i' = \frac{1}{h}$$

for $x_{i-1} \le x \le x_i$ and

$$\phi_i = \frac{(x_{i+1} - x)}{h}$$
$$\phi_i' = \frac{-1}{h}$$

for $x_i \le x \le x_{i+1}$.

Looking back at the weak form integral above, it is evaluated as follows

$$\int_{x=x_1}^{x=x_N} \phi_j \left(\left[\sum_{i=1}^{N} q_i \left(a\phi_i' + b\phi_i \right) \right] - f(x) \right) dx = 0 \qquad j = 1 \cdots N$$

For the first node only, $j = 1$, the following results

$$\int_{x=x_1}^{x=x_N} \phi_1 \left(\left[\sum_{i=1}^{N} q_i \left(a\phi_i' + b\phi_i \right) \right] - f(x) \right) dx = 0$$

Since ϕ_1 is not zero only over $x_1 \le x \le x_2$, and given that $\phi_1 = \frac{x_2 - x}{h}, \phi_1' = \frac{-1}{h}, \phi_2 = \frac{(x - x_1)}{h}$ due to the range of integration limits, and that $\phi_2' = \frac{1}{h}$, then the above can simplifies to

$$\int_{x=x_1}^{x=x_2} \phi_1 \left(\left[q_1 \left(a\phi_1' + b\phi_1 \right) + q_2 \left(a\phi_2' + b\phi_2 \right) \right] - f(x) \right) dx = 0$$

$$\int_{x=x_1}^{x=x_2} \frac{(x_2 - x)}{h} \left(\left[q_1 \left(\frac{-a}{h} + b\frac{(x_2 - x)}{h} \right) + q_2 \left(a\frac{1}{h} + b\frac{(x - x_1)}{h} \right) \right] - f(x) \right) dx = 0$$

The above simplifies further to

$$-q_1 \left(\frac{(3a + 2b(x_1 - x_2)) (x_1 - x_2)^2}{6h^2} \right) - q_2 \frac{(-3a + b(x_1 - x_2)) (x_1 - x_2)^2}{6h^2} - \int_{x=x_1}^{x=x_2} \phi_1 f(x) = 0$$

$$-q_1 \left(\frac{(3a + 2b(x_1 - x_2)) (x_1 - x_2)^2}{6h^2} \right) - q_2 \frac{(-3a + b(x_1 - x_2)) (x_1 - x_2)^2}{6h^2} - \frac{1}{h} \int_{x=x_1}^{x=x_2} (x_2 - x) f(x) = 0$$

Since $x_2 - x_1 = h$ and $x_1 - x_2 = -h$ the above reduces to

$$-q_1 \left(\frac{(3a - 2bh) h^2}{6h^2} \right) - q_2 \frac{(-3a - bh) h^2}{6h^2} - \frac{1}{h} \int_{x=x_1}^{x=x_2} (x_2 - x) f(x) = 0$$

$$q_1 \left(\frac{2bh - 3a}{6} \right) + q_2 \left(\frac{bh + 3a}{6} \right) - \frac{1}{h} \int_{x=x_1}^{x=x_2} (x_2 - x) f(x) = 0$$

The above equation gives the first row in the global stiffness matrix for any first order linear ODE of the form $a\frac{du}{dx}+bu(x)=f(x)$. The above shows that numerical integration is only needed to be performed on the term $\int_{x=x_1}^{x=x_2}(x_2-x)\,f(x)$.

Next the last equation is found, which will be the last row of the stiffness matrix. For the last node only $j=N$ the following results

$$\int_{x=x_{N-1}}^{x=x_N}\phi_N\left(\left[\sum_{i=1}^{N}q_i\left(a\phi_i'+b\phi_i\right)\right]-f(x)\right)dx=0$$

Since ϕ_N domain of influence is $x_{N-1}\le x\le x_N$, the above simplifies to

$$\int_{x=x_{N-1}}^{x=x_N}\phi_N\left(\left[q_{N-1}\left(a\phi_{N-1}'+b\phi_{N-1}\right)+q_N\left(a\phi_N'+b\phi_N\right)\right]-f(x)\right)dx=0$$

Since $\phi_{N-1}=\frac{(x_N-x)}{h},\phi_{N-1}'=\frac{-1}{h},\phi_N=\frac{x-x_{N-1}}{h},\phi_N'=\frac{1}{h}$ the above becomes

$$\int_{x=x_{N-1}}^{x=x_N}\frac{x-x_{N-1}}{h}\left(\left[q_{N-1}\left(\frac{-a}{h}+b\frac{(x_N-x)}{h}\right)+q_N\left(a\frac{1}{h}+b\frac{x-x_{N-1}}{h}\right)\right]-f(x)\right)dx=0$$

Which simplifies to

$$N-1\frac{(3a+b(x_{N-1}-x_N))\left(x_{N-1}-x_N\right)^2}{6h^2}-q_N\frac{(-3a+2b(x_{N-1}-x_N))\left(x_{N-1}-x_N\right)^2}{6h^2}-\frac{1}{h}\int_{x=x_{N-1}}^{x=x_N}(x-x_{N-1})\,f(x)\,d$$

Letting $x_{N-1}-x_N=-h$ in the above becomes

$$q_{N-1}\left(\frac{bh-3a}{6}\right)+q_N\frac{(3a+2bh)}{6}-\frac{1}{h}\int_{x=x_{N-1}}^{x=x_N}(x-x_{N-1})\,f(x)=0$$

Hence the last row of the stiffness matrix can be determined directly except for the term under the integral which needs to be evaluated using integration.

Now the equation that represents any internal node is found. This will be any row in the global stiffness matrix between the first and the last row.

For any j other than 1 or N the following results

$$\int_{x=x_{j-1}}^{x=x_j}\phi_j\left(\left[\sum_{i=1}^{N}q_i\left(a\phi_i'+b\phi_i\right)\right]-f(x)\right)dx+\int_{x=x_j}^{x=x_{j+1}}\phi_j\left(\left[\sum_{i=1}^{N}q_i\left(a\phi_i'+b\phi_i\right)\right]-f(x)\right)dx=0$$

Where the integral was broken into two parts to handle the domain of influence of the shape functions.

$$\int_{x=x_{j-1}}^{x=x_j}\phi_j\left(\ q_{j-1}\left(a\phi_{j-1}'+b\phi_{j-1}\right)+q_j\left(a\phi_j'+b\phi_j\right)-f(x)\right)dx+\int_{x=x_j}^{x=x_{j+1}}\phi_j\left(q_j\left(a\phi_j'+b\phi_j\right)+q_{j+1}\left(a\phi_{j+1}'+b\phi_{j+1}\right)-f(\right.$$

For

$$x_{j-1} \le x \le x_j, \phi_j = \frac{x - x_{j-1}}{h}, \phi'_j = \frac{1}{h}, \phi_{j-1} = \frac{x_j - x}{h}, \phi'_{j-1} = \frac{-1}{h}$$
$$x_j \le x \le x_{j+1}, \phi_j = \frac{x_{j+1} - x}{h}, \phi'_j = \frac{-1}{h}, \phi_{j+1} = \frac{x - x_j}{h}, \phi'_{j+1} = \frac{1}{h}$$

Hence the weak form integral can be written as

$$\int_{x=x_{j-1}}^{x=x_j} \frac{x - x_{j-1}}{h}\left(q_{j-1}\left(\frac{-a}{h} + b\frac{x_j - x}{h}\right) + q_j\left(\frac{a}{h} + b\frac{x - x_{j-1}}{h}\right) - f(x)\right) dx$$
$$+ \int_{x=x_j}^{x=x_{j+1}} \frac{x_{j+1} - x}{h}\left(q_j\left(\frac{-a}{h} + b\frac{x_{j+1} - x}{h}\right) + q_{j+1}\left(\frac{a}{h} + b\frac{x - x_j}{h}\right) - f(x)\right) dx = 0$$

Which simplifies to

$$q_{j-1}\left(\frac{\left(-3a - b(x_{j-1} - x_j)\right)(x_{j-1} - x_j)^2}{6h^2}\right) + q_j\frac{\left(\left(3a - 2b(x_{j-1} - x_j)\right)(x_{j-1} - x_j)^2\right)}{6h^2} - \frac{1}{h}\int_{x=x_{j-1}}^{x=x_j}(x - x_{j-1})\, f(x)\, dx$$
$$q_j\left(\frac{\left(-3a - 2b(x_j - x_{j+1})\right)(x_j - x_{j+1})^2}{6h^2}\right) + q_{j+1}\left(\frac{\left(3a - b(x_j - x_{j+1})\right)(x_j - x_{j+1})^2}{6h^2}\right) - \frac{1}{h}\int_{x=x_j}^{x=x_{j+1}}(x_{j+1} - x)\, f(x)\, dx$$

For equal distance between elements, $x_j - x_{j+1} = -h, x_{j-1} - x_j = -h$ the above simplifies to

$$q_{j-1}\left(\frac{\left(-3a + bh\right)h^2}{6h^2}\right) + q_j\frac{\left(\left(3a + 2bh\right)h^2\right)}{6h^2} - \frac{1}{h}\int_{x=x_{j-1}}^{x=x_j}(x - x_{j-1})\, f(x)\, dx + q_j\left(\frac{\left(-3a + 2bh\right)h^2}{6h^2}\right)$$
$$+ q_{j+1}\left(\frac{\left(3a + bh\right)h^2}{6h^2}\right) - \frac{1}{h}\int_{x=x_j}^{x=x_{j+1}}(x_{j+1} - x)\, f(x)\, dx = 0$$

Combining gives

$$q_{j-1}\left(\frac{-3a + bh}{6}\right) + q_j\left(\frac{2bh}{3}\right) + q_{j+1}\left(\frac{3a + bh}{6}\right) - \frac{1}{h}\left(\int_{x=x_{j-1}}^{x=x_j}(x - x_{j-1})\, f(x)\, dx + \int_{x=x_j}^{x=x_{j+1}}(x_{j+1} - x)\, f(x)\, dx\right) = 0$$

The above gives the expression for any row in the stiffness matrix other than the first and the last

row. Hence the global stiffness matrix is

$$\begin{bmatrix} \frac{2bh-3a}{6} & \frac{bh+3a}{6} & 0 & 0 & 0 & \dots & 0 \\ \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & 0 & \dots & 0 \\ 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & \dots & 0 \\ 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{bh-3a}{6} & \frac{3a+2bh}{6} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ \vdots \\ \vdots \\ q_{N-1} \\ q_N \end{bmatrix} = \begin{bmatrix} \frac{1}{h}\int_{x=x_1}^{x=x_2}(x_2-x)\,f(x \\ \frac{1}{h}\left(\int_{x=x_1}^{x=x_2}(x-x_1)\,f(x)\,dx+\int_{x=x_2}^{x=x_3}(x \\ \frac{1}{h}\left(\int_{x=x_2}^{x=x_3}(x-x_2)\,f(x)\,dx+\int_{x=x_3}^{x=x_4}(x \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{j-1}}^{x=x_j}(x-x_{j-1})\,f(x)\,dx+\int_{x=x_j}^{x=x_{j+1}} \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{N-2}}^{x=x_{N-1}}(x-x_{N-2})\,f(x)\,dx+\int_{x=x_{N-1}}^{x=x_N} \\ \frac{1}{h}\int_{x=x_{N-1}}^{x=x_N}(x-x_{N-1})\,f \end{bmatrix}$$

The above shows that the stiffness matrix can be build quickly without the use of any numerical integration in this example. Integration is needed to evaluate the force (or load) vector. Depending on the forcing function, this can be simple or difficult to perform.

Once the load vector is calculated, the unknowns q_i are solved for. But before doing that, q_1 is first replaced by the initial condition given in the problem

$$\begin{bmatrix} \frac{2bh-3a}{6} & \frac{bh+3a}{6} & 0 & 0 & 0 & \dots & 0 \\ \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & 0 & \dots & 0 \\ 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & \dots & 0 \\ 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{bh-3a}{6} & \frac{3a+2bh}{6} \end{bmatrix} \begin{bmatrix} u_0 \\ q_2 \\ q_3 \\ \vdots \\ \vdots \\ \vdots \\ q_{N-1} \\ q_N \end{bmatrix} = \begin{bmatrix} \frac{1}{h}\int_{x=x_1}^{x=x_2}(x_2-x)\,f(x \\ \frac{1}{h}\left(\int_{x=x_1}^{x=x_2}(x-x_1)\,f(x)\,dx+\int_{x=x_2}^{x=x_3}(x \\ \frac{1}{h}\left(\int_{x=x_2}^{x=x_3}(x-x_2)\,f(x)\,dx+\int_{x=x_3}^{x=x_4}(x \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{j-1}}^{x=x_j}(x-x_{j-1})\,f(x)\,dx+\int_{x=x_j}^{x=x_{j+1}} \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{N-2}}^{x=x_{N-1}}(x-x_{N-2})\,f(x)\,dx+\int_{x=x_{N-1}}^{x=x_N} \\ \frac{1}{h}\int_{x=x_{N-1}}^{x=x_N}(x-x_{N-1})\,f \end{bmatrix}$$

Now the first row is removed (and remembering to multiply u_0 by the first entry in the second row) gives

$$\begin{bmatrix} \frac{-3a+bh}{6}u_0 & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & 0 & \dots & 0 \\ 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & \dots & 0 \\ 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{bh-3a}{6} & \frac{3a+2bh}{6} \end{bmatrix} \begin{bmatrix} q_2 \\ q_3 \\ \vdots \\ \vdots \\ \vdots \\ q_{N-1} \\ q_N \end{bmatrix} = \begin{bmatrix} \frac{1}{h}\left(\int_{x=x_1}^{x=x_2} (x-x_1)\, f(x)\, dx + \right. \\ \frac{1}{h}\left(\int_{x=x_2}^{x=x_3} (x-x_2)\, f(x)\, dx + \int_{x=x_3}^{x=x_4} \right. \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{j-1}}^{x=x_j} (x-x_{j-1})\, f(x)\, dx + \int_{x=x_j}^{x=x_{j+1}} \right. \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{N-2}}^{x=x_{N-1}} (x-x_{N-2})\, f(x)\, dx + \right. \\ \frac{1}{h} \int_{x=x_{N-1}}^{x=x_N} (x-x_{N-1} \end{bmatrix}$$

The first column is now removed after moving the first entry in the first row to the RHS to become part of the load vector

$$\begin{bmatrix} \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & 0 & \dots & 0 \\ \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 & \dots & 0 \\ 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} & 0 \\ 0 & 0 & 0 & \frac{-3a+bh}{6} & \frac{2bh}{3} & \frac{3a+bh}{6} \\ 0 & 0 & 0 & 0 & \frac{bh-3a}{6} & \frac{3a+2bh}{6} \end{bmatrix} \begin{bmatrix} q_2 \\ q_3 \\ \vdots \\ \vdots \\ \vdots \\ q_{N-1} \\ q_N \end{bmatrix} = \begin{bmatrix} \frac{1}{h}\left(\int_{x=x_1}^{x=x_2} (x-x_1)\, f(x)\, dx + \int_{x=x_2}^{x=x_3} (x_3-x)\, f(x)\, dx \right. \\ \frac{1}{h}\left(\int_{x=x_2}^{x=x_3} (x-x_2)\, f(x)\, dx + \int_{x=x_3}^{x=x_4} (x_4-x) \right. \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{j-1}}^{x=x_j} (x-x_{j-1})\, f(x)\, dx + \int_{x=x_j}^{x=x_{j+1}} (x_{j+1} \right. \\ \vdots \\ \frac{1}{h}\left(\int_{x=x_{N-2}}^{x=x_{N-1}} (x-x_{N-2})\, f(x)\, dx + \int_{x=x_{N-1}}^{x=x_N} (x_N \right. \\ \frac{1}{h} \int_{x=x_{N-1}}^{x=x_N} (x-x_{N-1})\, f(x) \end{bmatrix}$$

Now the system above is solved for $q_2, q_3, \cdots, q_N$. Once the q_i are found, the solution is

$$u(x) = \sum_{i=1}^{Number\ \ Nodes} q_i \phi_i$$

The Appendix shows a Mathematica code which solve the above general first order ODE. 4 first order ODE solved for illustration and the solution is compared to the exact solution. Animation

was made to help illustrate this. the RMS error was calculated. The animations can be access by clicking on the links below (shows only in the HTML version of this report).

3.2 Example Two. 2nd order ODE, Boundary value problem. Linear interpolation. Symmetric weak form.

Now the FEM formulation is given for a boundary value, second order ODE with constant coefficients of the form

$$a\frac{d^2u}{dx^2}+b\frac{du}{dx}+cu(x)=f(x)$$

with the initial conditions $u(x_0)=u_0$ (called essential Dirichlet condition), and the Neumann boundary condition $\frac{du}{dx}=\beta$ at L, for $x_0\le x\le L$

As before, the solution is assumed to be of the form

$$u=\sum_{i=1}^{N}q_i\phi_i$$

And the unknowns q_i are found by solving N algebraic equations

$$\int_{x_0}^{L}\phi_j R(x)\,dx=0 \qquad j=1\cdots N$$

Where $R(x)$ is the ODE residual obtained by substituting the assumed solution into the original ODE. Hence the above becomes (For simplicity, $j=1\cdots N$ is not written each time as it is assumed to be the case)

$$\int_{x_0}^{L}\phi_j\left(a\frac{d^2u}{dx^2}+b\frac{du}{dx}+cu(x)-f(x)\right)dx=0$$

$$\int_{x_0}^{L}\phi_j a\frac{d^2u}{dx^2}dx+\int_{x_0}^{L}\phi_j\left(b\frac{du}{dx}+cu(x)-f(x)\right)dx=0$$

Applying integration by parts on $\int_{x_0}^{L}\phi_j a\frac{d^2u}{dx^2}dx$. Since

$$\frac{d}{dx}\left(a\ \phi_j\frac{du}{dx}\right)=a\phi_j\frac{d^2u}{dx^2}+a\phi_j'\frac{du}{dx}$$

then integrating both sides of the above gives

$$\int_{x_0}^{L}\frac{d}{dx}\left(a\ \phi_j\frac{du}{dx}\right)dx=\int_{x_0}^{L}a\phi_j\frac{d^2u}{dx^2}dx+\int_{x_0}^{L}a\phi_j'\frac{du}{dx}dx$$

$$\left[a\phi_j\frac{du}{dx}\right]_{x_0}^{L}=\int_{x_0}^{L}a\phi_j\frac{d^2u}{dx^2}dx+\int_{x_0}^{L}a\phi_j'\frac{du}{dx}dx$$

Hence

$$\int_{x_0}^{L} a\phi_j \frac{d^2u}{dx^2}dx = \left[a\phi_j \frac{du}{dx}\right]_{x_0}^{L} - \int_{x_0}^{L} a\phi_j' \frac{du}{dx}dx$$

Substituting the above in equation A1 gives

$$\left[a\phi_j \frac{du}{dx}\right]_{x_0}^{L} - \int_{x_0}^{L} a\phi_j' \frac{du}{dx}dx + \int_{x_0}^{L} \phi_j \left(b\frac{du}{dx} + cu(x) - f(x)\right) dx = 0 \tag{A2}$$

Considering the term

$$\left[a\phi_j \frac{du}{dx}\right]_{x_0}^{L} = a\phi_j \frac{du}{dx}\bigg|_{x=L} - a\phi_j \frac{du}{dx}\bigg|_{x=x_0}$$

First considering the term $a\phi_j \frac{du}{dx}\big|_{x=L}$. Since $\frac{du}{dx} = \beta$ at $x = L$ then this term becomes $a\beta[\phi_j]_{x=L}$. But $[\phi_j]_{x=L}$ is non zero only for the N^{th} shape function evaluated at $x = L$. Since linear interpolation is used, the N^{th} shape function is $\phi_N = \frac{x - x_{N-1}}{h}$ which have the value of 1 at $x = x_N$. Hence

$$a\phi_j \frac{du}{dx}\bigg|_{x=L} = a\beta \qquad \text{when} j = N, \text{otherwise,} 0$$

Now considering the term $a\phi_j \frac{du}{dx}\big|_{x=x_0}$, since at $x = x_0$ all shape functions will be zero except for ϕ_1 which has the value of 1 at $x = x_0$. Hence this simplifies to $a\frac{du}{dx}\big|_{x=x_0}$

Recalling that $\frac{du}{dx} = \frac{d}{dx}\sum_{i=1}^{N} q_i\phi_i = \sum_{i=1}^{N} q_i\phi_i'$. But at $x = x_0$ only ϕ_1 is defined and its has the slope of $\frac{-1}{h}$, hence

$$a\frac{du}{dx}\bigg|_{x=x_0} = \frac{-aq_1}{h} \text{ when } j = 1, \text{ otherwise } 0$$

However, $q_1 = u_0$ since that is by definition the initial condition. Finally one obtains

$$\left[a\phi_j \frac{du}{dx}\right]_{x_0}^{L} = \left\{ \begin{array}{ll} \frac{aq_1}{h} & j = 1 \\ a\beta & j = N \end{array} \right\}$$

Hence the symmetric weak form equation (A2) can now be simplified more giving

$$\left\{ \begin{array}{ll} \frac{aq_1}{h} & j = 1 \\ a\beta & j = N \end{array} \right\} - \int_{x_0}^{L} a\phi_j' \frac{du}{dx}dx + \int_{x_0}^{L} \phi_j \left(b\frac{du}{dx} + cu(x) - f(x)\right) dx = 0 \qquad j = 1 \cdots N \tag{A3}$$

The trial function obtain by linear interpolation are used. These are shown in this table

i	ϕ_i	ϕ_i'
1	$\frac{(x_2 - x)}{h}$	$\frac{-1}{h}$
$1 < i < N$	$\left\{\frac{(x - x_{i-1})}{h}, \frac{(x_{i+1} - x)}{h}\right\}$	$\left\{\frac{1}{h}, \frac{-1}{h}\right\}$
N	$\frac{(x - x_{N-1})}{h}$	$\frac{1}{h}$

The global stiffness matrix is now constructed. For the first equation (which corresponds to the first row in the global stiffness matrix) the result is

$$\frac{aq_1}{h} - \int_{x_0}^{L} a\phi_1' \frac{du}{dx} dx + \int_{x_0}^{L} \phi_1 \left(b\frac{du}{dx} + cu(x) - f(x) \right) dx = 0$$

Since $u = \sum_{i=1}^{N} q_i \phi_i$, and since ϕ_1 domain of influence is only from x_1 to x_2 then above becomes

$$\frac{aq_1}{h} - \int_{x_1}^{x_2} a\phi_1' \frac{d}{dx}(q_1\phi_1 + q_2\phi_2)\, dx + \int_{x_1}^{x2} \phi_1 \left(b\frac{d}{dx}(q_1\phi_1 + q_2\phi_2) + c(q_1\phi_1 + q_2\phi_2) - f(x) \right) dx = 0$$

But $\phi_1 = \frac{(x_2 - x)}{h}$ and $\phi_1' = \frac{-1}{h}$ and over the domain from x_1 to x_2, $\phi_2 = \frac{(x - x_1)}{h}, \phi_2' = \frac{1}{h}$, hence the above becomes

$$\frac{aq_1}{h} - \int_{x_1}^{x_2} \frac{-a}{h}\left(\frac{-q_1}{h} + \frac{q_2}{h} \right) dx + \int_{x_1}^{x2} \frac{(x_2 - x)}{h} \left(b\left(\frac{-q_1}{h} + \frac{q_2}{h} \right) + c\left(q_1 \frac{(x_2 - x)}{h} + q_2 \frac{(x - x_1)}{h} \right) - f(x) \right) dx = 0$$

The above simplifies to

$$\frac{aq_1}{h} - \frac{q_1}{h^2}\left(ax_1 - \frac{b(x_1 - x_2)^2}{2} - \frac{c(x_1 - x_2)^3}{3} - ax_2 \right) + \frac{q_2}{h^2}\left(-ax_1 + \frac{b(x_1 - x_2)^2}{2} - \frac{c(x_1 - x_2)^3}{6} + ax_2 \right) - \frac{1}{h}\int_{x_1}^{x2} (x_2 - x)\, f(x) = 0$$

$$-\frac{q_1}{h^2}\left(-ah + ax_1 - \frac{b(x_1 - x_2)^2}{2} - \frac{c(x_1 - x_2)^3}{3} - ax_2 \right) + \frac{q_2}{h^2}\left(-ax_1 + \frac{b(x_1 - x_2)^2}{2} - \frac{c(x_1 - x_2)^3}{6} + ax_2 \right) - \frac{1}{h}\int_{x_1}^{x2} (x_2 - x)\, f(x) = 0$$

The above gives the first row in the stiffness matrix. Since it is assumed that each element will have the same length, hence $x_1 - x_2 = -h$, and the above becomes

$$-\frac{q_1}{h^2}\left(-ah + ax_1 - \frac{bh^2}{2} + \frac{ch^3}{3} - ax_2 \right) + \frac{q_2}{h^2}\left(-ax_1 + \frac{bh^2}{2} + \frac{ch^3}{6} + ax_2 \right) - \frac{1}{h}\int_{x_1}^{x2} (x_2 - x)\, f(x) = 0$$

$$-q_1\left(-\frac{a}{h} + \frac{ax_1}{h^2} - \frac{b}{2} + \frac{ch}{3} - \frac{ax_2}{h^2} \right) + q_2\left(-\frac{ax_1}{h^2} + \frac{b}{2} + \frac{ch}{6} + \frac{ax_2}{h^2} \right) - \frac{1}{h}\int_{x_1}^{x2} (x_2 - x)\, f(x) = 0$$

For the last equation, which will be the last row in the global stiffness matrix

$$a\beta - \int_{x_{N-1}}^{x_N} a\phi_N' \frac{du}{dx} dx + \int_{x_{N-1}}^{x_N} \phi_N \left(b\frac{du}{dx} + cu(x) - f(x) \right) dx = 0$$

Since $u = \sum_{i=1}^{N} q_i \phi_i$, and since ϕ_N domain of influence is only from x_{N-1} to x_N The above becomes

$$a\beta - \int_{x_{N-1}}^{x_N} a\phi_N' \frac{d}{dx}(q_{N-1}\phi_{N-1} + q_N\phi_N)\, dx + \int_{x_{N-1}}^{x_N} \phi_N \left(b\frac{d}{dx}(q_{N-1}\phi_{N-1} + q_N\phi_N) + c(q_{N-1}\phi_{N-1} + q_N\phi_N) - f(x) \right) dx$$

But $\phi_N = \frac{(x-x_{N-1})}{h}$ and $\phi'_N = \frac{1}{h}$ and over the domain from x_{N-1} to x_N, $\phi_{N-1} = \frac{(x_N-x)}{h}, \phi'_{N-1} = \frac{-1}{h}$ hence the above becomes

$$a\beta - \int_{x_{N-1}}^{x_N} \frac{a}{h}\left(q_{N-1}\left(\frac{-1}{h}\right) + q_N\frac{1}{h}\right)dx + \int_{x_{N-1}}^{x_N} \frac{(x-x_{N-1})}{h}\left(b\left(q_{N-1}\left(\frac{-1}{h}\right) + q_N\frac{1}{h}\right) + c\left(q_{N-1}\frac{(x_N-x)}{h} + q_N\frac{(x-x_N}{h}\right.\right.$$

The above simplifies to

$$a\beta - \frac{q_{N-1}}{h^2}\left(-ax_{N-1} - \frac{b(x_{N-1}-x_N)^2}{2} - \frac{c(x_{N-1}-x_N)^3}{6} + ax_N\right) + \frac{q_N}{h^2}\left(+ax_{N-1} + \frac{b(x_{N-1}-x_N)^2}{2} - \frac{c(x_{N-1}-x_N}{3}\right.$$

The above represents the last row in the global stiffness matrix. Since it is assumed that each element will have the same length, hence $x_{N-1} - x_N = -h$, and the above becomes

$$a\beta - \frac{q_{N-1}}{h^2}\left(-ax_{N-1} - \frac{bh^2}{2} + \frac{ch^3}{6} + ax_N\right) + \frac{q_N}{h^2}\left(ax_{N-1} + \frac{bh^2}{2} + \frac{ch^3}{3} - ax_N\right) - \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x) = 0$$

$$a\beta - q_{N-1}\left(-\frac{ax_{N-1}}{h^2} - \frac{b}{2} + \frac{ch}{6} + \frac{ax_N}{h^2}\right) + q_N\left(\frac{ax_{N-1}}{h^2} + \frac{b}{2} + \frac{ch}{3} - \frac{ax_N}{h^2}\right) - \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x) = 0$$

The expression for any row in between the first and the last rows is now found. For a general node j the result is

$$-\int_{x_{j-1}}^{x_{j+1}} a\phi'_j\frac{du}{dx}dx + \int_{x_{j-1}}^{x_{j+1}} \phi_j\left(b\frac{du}{dx} + cu(x) - f(x)\right)dx = 0$$

Breaking the integral into halves to make it easier to write the trial functions over the domain of influence gives

$$-\left(\int_{x_{j-1}}^{x_j} a\phi'_j\frac{du}{dx}dx + \int_{x_j}^{x_{j+1}} a\phi'_j\frac{du}{dx}dx\right) + \int_{x_{j-1}}^{x_j}\phi_j\left(b\frac{du}{dx} + cu(x) - f(x)\right)dx + \int_{x_j}^{x_{j+1}}\phi_j\left(b\frac{du}{dx} + cu(x) - f(x)\right)dx = 0$$

Considering the first domain $x_{j-1} \le x \le x_j$ gives

$$-\int_{x_{j-1}}^{x_j} a\phi'_j\frac{du}{dx}dx + \int_{x_{j-1}}^{x_j}\phi_j\left(b\frac{du}{dx} + cu(x) - f(x)\right)dx$$

Over this range, $u = q_{j-1}\phi_{j-1} + q_j\phi_j$ hence $\frac{du}{dx} = q_{j-1}\phi'_{j-1} + q_j\phi'_j$ where $\phi_{j-1} = \frac{x_j-x}{h}, \phi'_{j-1} = \frac{-1}{h}, \phi_j = \frac{x-x_{j-1}}{h}, \phi'_j = \frac{1}{h}$ hence the above becomes

$$-\int_{x_{j-1}}^{x_j} a\frac{1}{h}\left(q_{j-1}\left(\frac{-1}{h}\right) + q_j\frac{1}{h}\right)dx + \int_{x_{j-1}}^{x_j}\frac{x-x_{j-1}}{h}\left(b\left(q_{j-1}\left(\frac{-1}{h}\right) + q_j\frac{1}{h}\right) + c\left(q_{j-1}\frac{x_j-x}{h} + q_j\frac{x-x_{j-1}}{h}\right) - f(x)\right)dx \tag{A4}$$

Now considering the second domain $x_j \le x \le x_{j+1}$ gives

$$-\int_{x_j}^{x_{j+1}} a\phi_j' \frac{du}{dx} dx + \int_{x_j}^{x_{j+1}} \phi_j \left(b\frac{du}{dx} + cu(x) - f(x) \right) dx$$

Over this range, $u = q_j\phi_j + q_{j+1}\phi_{j+1}$ hence $\frac{du}{dx} = q_j\phi_j' + q_{j+1}\phi_{j+1}'$ where $\phi_j = \frac{x_{j+1}-x}{h}, \phi_j' = \frac{-1}{h}, \phi_{j+1} = \frac{x-x_j}{h}, \phi_{j+1}' = \frac{1}{h}$ hence the above becomes

$$-\int_{x_j}^{x_{j+1}} \left(\frac{-a}{h}\right)\left(\frac{-q_j}{h} + \frac{q_{j+1}}{h}\right) dx + \int_{x_j}^{x_{j+1}} \frac{x_{j+1}-x}{h}\left(b\left(\frac{-q_j}{h} + q_{j+1}\frac{1}{h}\right) + c\left(q_j\frac{x_{j+1}-x}{h} + q_{j+1}\frac{x-x_j}{h}\right) - f(x)\right) dx$$

Combine A4 and A5 and simplifying gives

$$\frac{q_{j-1}}{h^2}\left(-ax_{j-1} - \frac{b(x_{j-1}-x_j)^2}{2} - \frac{c(x_{j-1}-x_j)^3}{6} + ax_j\right) + \frac{q_j}{h^2}\left(ax_{j-1} + \frac{b(x_{j-1}-x_j)^2}{2} - \frac{c(x_{j-1}-x_j)^3}{3} - \frac{b(x_j -$$

$$+\frac{q_{j+1}}{h^2}\left(-ax_j + \frac{b(x_j-x_{j+1})^2}{2} - \frac{c(x_j-x_{j+1})^3}{6} + ax_{j+1}\right) - \int_{x_{j-1}}^{x_j} \frac{x-x_{j-1}}{h} f(x)\,dx - \int_{x_j}^{x_{j+1}} \frac{x_{j+1}-x}{h} dx = 0$$

Since we are assuming each element will have the same length, hence $x_{j-1} - x_j = -h$, $x_j - x_{j+1} = -h$ and the above becomes

$$q_{j-1}\left(-\frac{ax_{j-1}}{h^2} - \frac{b}{2} + \frac{ch}{6} + \frac{ax_j}{h^2}\right) + q_j\left(\frac{ax_{j-1}}{h^2} + \frac{2ch}{3} - \frac{ax_{j+1}}{h^2}\right)$$
$$+ q_{j+1}\left(\frac{-ax_j}{h^2} + \frac{b}{2} + \frac{ch}{6} + \frac{ax_{j+1}}{h^2}\right) - \int_{x_{j-1}}^{x_j} \frac{x-x_{j-1}}{h} f(x)\,dx - \int_{x_j}^{x_{j+1}} \frac{x_{j+1}-x}{h} f(x)\,dx = 0$$

The above gives any row in the stiffness matrix other than the first and the last row. Now we can

write the global stiffness matrix as

$$\begin{bmatrix} \left(-\frac{a}{h}+\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{3}-\frac{ax_2}{h^2}\right) & \left(-\frac{ax_1}{h^2}+\frac{b}{2}-\frac{ch}{6}+\frac{ax_2}{h^2}\right) & 0 & 0 \\ \left(-\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_2}{h^2}\right) & \left(\frac{ax_1}{h^2}+\frac{2ch}{3}-\frac{ax_3}{h^2}\right) & \left(\frac{-ax_2}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & 0 \\ 0 & \left(-\frac{ax_{j-1}}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_j}{h^2}\right) & \left(\frac{ax_{j-1}}{h^2}+\frac{2ch}{3}-\frac{ax_{j+1}}{h^2}\right) & \left(\frac{-ax_j}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_{j+1}}{h^2}\right) \\ 0 & 0 & \ddots & \cdots \\ 0 & 0 & \left(-\frac{ax_{N-1}}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_N}{h^2}\right) & \left(\frac{ax_{N-1}}{h^2}+\frac{b}{2}+\frac{ch}{3}-\frac{ax_N}{h^2}\right) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{h}\int_{x_1}^{x2}(x_2-x)\,f(x) \\ \vdots \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})\,f(x)\,dx+\frac{1}{h}\int_{x_j}^{x_{j+1}}(x_{j+1}-x)\,f(x)\,dx \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x)-a\beta \end{bmatrix}$$

We must first replace q_1 by the initial condition given in the problem, and we must remove the

first row after that as follows

$$\begin{bmatrix} \left(-\frac{a}{h}+\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{3}-\frac{ax_2}{h^2}\right) & \left(-\frac{ax_1}{h^2}+\frac{b}{2}-\frac{ch}{6}+\frac{ax_2}{h^2}\right) & 0 & 0 \\ \left(-\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_2}{h^2}\right) & \left(\frac{ax_1}{h^2}+\frac{2ch}{3}-\frac{ax_3}{h^2}\right) & \left(\frac{-ax_2}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & 0 \\ 0 & 0 & \ddots & \cdots \\ 0 & 0 & \ddots & \cdots \\ 0 & 0 & \left(-\frac{ax_{N-1}}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_N}{h^2}\right) & \left(\frac{ax_{N-1}}{h^2}+\frac{b}{2}+\frac{ch}{3}-\frac{ax_N}{h^2}\right) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{h}\int_{x_1}^{x2}(x_2-x)\,f(x) \\ \vdots \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})\,f(x)\,dx+\frac{1}{h}\int_{x_j}^{x_{j+1}}(x_{j+1}-x)\,f(x)\,dx \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x)-a\beta \end{bmatrix}$$

Multiply the first element in the second row by u_0 and removing the first row gives

$$\begin{bmatrix} \left(-\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_2}{h^2}\right)u_0 & \left(\frac{ax_1}{h^2}+\frac{2ch}{3}-\frac{ax_3}{h^2}\right) & \left(\frac{-ax_2}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & 0 \\ 0 & \left(-\frac{ax_2}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & \left(\frac{ax_2}{h^2}+\frac{2ch}{3}-\frac{ax_4}{h^2}\right) & \left(\frac{-ax_3}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_4}{h^2}\right) \\ 0 & 0 & \ddots & \ddots \\ 0 & 0 & \left(-\frac{ax_{N-1}}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_N}{h^2}\right) & \left(\frac{ax_{N-1}}{h^2}+\frac{b}{2}+\frac{ch}{3}-\frac{ax_N}{h^2}\right) \end{bmatrix} \begin{bmatrix} q \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{h}\int_{x_1}^{x_2}(x-x_2)\,f(x)\,dx + \frac{1}{h}\int_{x_2}^{x_3}(x_3-x)\,dx \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})\,f(x)\,dx + \frac{1}{h}\int_{x_j}^{x_{j+1}}(x_{j+1}-x)\,f(x)\,dx \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x) - a\beta \end{bmatrix}$$

Now moving the first element of the first row above to the RHS and removing the first column

gives

$$\begin{bmatrix} \left(\frac{ax_1}{h^2}+\frac{2ch}{3}-\frac{ax_3}{h^2}\right) & \left(\frac{-ax_2}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & 0 & 0 \\ 0 & \left(-\frac{ax_2}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_3}{h^2}\right) & \left(\frac{ax_2}{h^2}+\frac{2ch}{3}-\frac{ax_4}{h^2}\right) & \left(\frac{-ax_j}{h^2}+\frac{b}{2}+\frac{ch}{6}+\frac{ax_{j+1}}{h^2}\right) \\ \vdots & \ddots & \cdots & \ddots \\ 0 & 0 & \left(-\frac{ax_{N-1}}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_N}{h^2}\right) & \left(\frac{ax_{N-1}}{h^2}+\frac{b}{2}+\frac{ch}{6}-\frac{ax_N}{h^2}\right) \end{bmatrix} \begin{bmatrix} q_2 \\ \vdots \\ \vdots \\ q_j \\ \vdots \\ q_{N-1} \\ q_N \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{h}\int_{x_1}^{x_2}(x-x_2)\,f(x)\,dx+\frac{1}{h}\int_{x_2}^{x_3}(x_3-x)\,dx-\left(-\frac{ax_1}{h^2}-\frac{b}{2}+\frac{ch}{6}+\frac{ax_2}{h^2}\right)u_0 \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})\,f(x)\,dx+\frac{1}{h}\int_{x_j}^{x_{j+1}}(x_{j+1}-x)\,f(x)\,dx \\ \vdots \\ \vdots \\ \frac{1}{h}\int_{x_{N-1}}^{x_N}(x-x_{N-1})\,f(x)-a\beta \end{bmatrix}$$

Now we solve for the vector $[q_2, q_3, \cdots, q_N]^T$ and this completes our solution.

A Matlab implementation is below which solves any second order ODE. This code was used to generate an animation. This animation can be accessed by clicking on the link below.

4 References

1. Methods of computer modeling in engineering and the sciences. Volume 1. By Professor Satya N. Atluri. Tech Science Press.
2. Class lecture notes. MAE 207. Computational methods. UCI. Spring 2006. Instructor: Professor SN Atluri.
3. Computational techniques for fluid dynamics, Volume I. C.A.J.Fletcher. Springer-Verlag

Matlab driver file This file sets up the call to do FEM and the animation

```
function nma_fem_driver
%script to do a simulation of FEM solution of a second order ODE
%by Nasser Abbasi
%This script calls the m file called nma_fem.m
```

```
%
%this solves the ODE
%
%   a u''(t) + b u'(t) + c u(t) = f(t)
%
% with t over the range t0 to len
% and with initial conditions u(0)=u0
% and with u'(len)=beta
%
%To use this to solve an ODE, edit the values below for the variables
% a,b,c,f,len,u0,x0

close all; clear all;

    %   equation 1. u''+u'+u=sin(x)*cos(x). u(0)=1, u'(len)=beta, len=10
    %   maxy=4 miny=-1
    %   a=1, b=1, c=1, x0=0, u0=1,len=10

    a=1;
    b=1;
    c=1;
    x0=0;   % starting domain point
    u0=-10;
    len=10;
    beta=1;

    ymax=4; ymin=-2;  % for plotting only. To make the plot looks better

    %This below solves the equation exactly to compare the FEM solution
    %against.

    sol= dsolve('a*D2y+b*Dy+c*y=cos(t)*sin(t)','y(0)=-10','Dy(10)=1');
    sol=subs(sol);

    directory='matlab_ANIMATION/';
    extension='.png';

    MAX_NODES = 100;
    for n = 3:MAX_NODES
        ezplot(sol,[0:len,-ymin:ymax]);
        hold on;

        r = nma_fem(a,b,c,@f,x0,u0,beta,len,sol,n);  %SOLVES by FEM

        title(sprintf('Solving y''''+y''+y(x)=sin(x)*cos(x).  y(0)=1, y''(10)=1. by FEM. \nN=%d\nRMSER
        drawnow;

        image = getframe(gcf);
        p=frame2im(image);
        file_name=[directory num2str(n) extension] ;
        imwrite(p,file_name,'png');

        hold off;
        % pause(.5)
```

```
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%
%
% edit this below to change the forcing function.
%%%%%%%%%%%%%%%%%%%%%%%%
function v=f(x)
   v=sin(x).*cos(x);
end
```

Matlab function to solve ODE using FEM This file does FEM solution on general second order ODE

```
function rmserror=nma_fem(a,b,c,f,x0,u0,beta,len,exact_sol,nNodes)
% Solves numerically a second order ODE with constant coeff.
% using the symmetric Garlekin Finite Elements Methods.
%
% see the file nma_fem_driver.m on how to call this function.
%
% Solve a u''(t) + b u'(t) + c u(t) = f(t)
%
% with t over the range t0 to len
% and with initial conditions u(0)=u0
% and with u'(len)=beta

%by Nasser Abbasi. Sept 26,2006.

    xc=linspace(x0,len,nNodes);

    % This plots the shape functions.
    %    x=linspace(x0,len,1000);
    %    y=linspace(x0,len,1000);
    %    for i=1:nNodes
    %        for j=1:length(y)
    %            [v,d]=phi(i,x(j),nNodes,x0,len,xc);
    %            y(j)=v;
    %        end
    %        plot(x,y);
    %        hold on;
    %    end


    A    = build_stiffness_matrix(nNodes,xc,a,b,c);
    load = build_load_vector(a,u0,nNodes,xc,f,beta);

    % Now remove the first row and column from the stiffness matrix

    A(2,1)  = A(2,1)*u0;
    load(2) = load(2)-A(2,1);
    A       = A(2:end,2:end);
    load    = load(2:end);
```

```
    % SOLVE for unknowns

    q = A\load;
    q = [u0;q];

    % Plot the solution
    y = zeros(length(xc),1);
    for i=1:length(xc)
        y(i)=trial(xc(i),x0,len,xc,nNodes,q);
    end

    plot(xc,y,'ro');
    hold on;
    line(xc,y);

    % Calculate RMSerror. Use 50 points. Should be more than enough.
    NPOINTS = 50;
    x = linspace(x0,len,NPOINTS);
    rmserror = 0;
    for i = 1:length(x)
        y = trial(x(i),x0,len,xc,nNodes,q);
        t=x(i);  % USED for subs below. do not remove.
        yexact   = real(double(subs(exact_sol)));
        rmserror = rmserror+(y-yexact)^2;
    end

    rmserror = rmserror/NPOINTS;
    rmserror = sqrt(rmserror);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function v = trial(x,x0,len,xc,nNodes,q)
    if x<x0||x>len
        error('in Trial. x outside range');
    end
    v = 0;
    for i = 1:nNodes
        [s,d] = phi(i,x,nNodes,x0,len,xc); %notice ignore d here.
        v     = v+ q(i)*s;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [v,d]=phi(i,x,nNodes,x0,len,xc)
    if(i<1 || i>nNodes)
        error('node number outside range');
    end
    if(x<x0 || x>len)
        error('x outside range');
    end
```

```
    h = xc(2)-xc(1);

    if i==1
        if(x>xc(2))
            v = 0;
            d = 0;
        else
            v = (xc(2)-x)/h;
            d = -1/h;
        end
        return;
    end

    if i==nNodes
        if(x<xc(nNodes-1))
            v = 0;
            d = 0;
        else
            v = (x-xc(nNodes-1))/h;
            d = 1/h;
        end
        return;
    end

    if(x>xc(i+1) || x<xc(i-1) )
        v = 0;
        d = 0;
    else
        if(x<=xc(i))
            v = (x-xc(i-1))/h;
            d = 1/h;
        else
            v = (xc(i+1)-x)/h;
            d = -1/h;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEE my report for description of this function.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A=build_stiffness_matrix(nNodes,xc,a,b,c)

    A = zeros(nNodes,nNodes);
    h = xc(2)-xc(1);
    for i = 1:nNodes
        if i==1
            A(1,1) = -a/h+ a*xc(1)/h^2 - b/h +c*h/3 - a*xc(2)/h^2;
            A(1,2) = -a*xc(1)/h^2 +b/2 -c*h/6 + a*xc(2)/h^2;
        else
            if i==nNodes
                A(nNodes,nNodes-1) = -a*xc(nNodes-1)/h^2 -b/2 +c*h/6 +a*xc(nNodes)/h^2;
                A(nNodes,nNodes)   = a*xc(nNodes-1)/h^2 +b/2 +c*h/3 -a*xc(nNodes)/h^2;
```

```
            else
                A(i,i-1) = -a*xc(i-1)/h^2 -b/2 + c*h/6 + a*xc(i)/h^2;
                A(i,i)   = a*xc(i-1)/h^2 + 2*c*h/3 -a*xc(i+1)/h^2;
                A(i,i+1) = -a*xc(i)/h^2 + b/2 + c*h/6 + a*xc(i+1)/h^2;
            end
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Called to integrate the 'force' function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function v=integrand(x,f,xj)
    v = (x-xj).*f(x);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% SEE my report for more description of this
% This build the load vector.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function load = build_load_vector(a,u0,nNodes,xc,f,beta)

    load = zeros(nNodes,1);
    h    = xc(2)-xc(1);


    load(1) = -(1/h)* quad(@(x)integrand(x,f,xc(2)),xc(1),xc(2)) ;
    load(nNodes) = (1/h)* quad(@(x)integrand(x,f,xc(nNodes-1)),xc(nNodes-1),xc(nNodes)) - a*beta;
    for i = 2:nNodes-1
        load(i) = quad(@(x)integrand(x,f,xc(i-1)),xc(i-1),xc(i)) - ...
                  quad(@(x)integrand(x,f,xc(i+1)),xc(i),xc(i+1));
        load(i) = (1/h)*load(i) ;
    end
end
```