CS 6220 - Assignment 4

Ally Warner

April 21, 2016

1. Consider the boundary value ODE

$$-(au')' = q, \quad 0 < x < 1,$$

$$u(0) = 0, \quad u'(1) = 0,$$

where $a(x) > 0$ and $q(x)$ are known, smooth functions. It is well known (recall Section 3.1.3) that u also minimizes,

$$T = \int_0^1 [a(u')^2 - 2uq]dx$$

over all functions with bounded first derivatives that satisfy the essential BC $u(0) = 0$. Consider next discretizing the integral on a nonuniform mesh,

$$0 = x_0 < x_1 < \cdots < x_J = 1,$$

as in Exercises 2 and 3.

(a) Show that, applying the midpoint rule for the first term and the trapezoidal rule for the second term in T for each subinterval, one obtains the problem of minimizing

$$T_h = \sum_{i=0}^{J-1} a(x_{i+\frac{1}{2}}) \frac{(v_{i+1} - v_i)^2}{h_i} - h_i(q(x_i)v_i + q(x_{i+1})v_{i+1})$$

with $v_0 = 0$. Thus, $T_h(u) = T(u) + \mathcal{O}(h^2)$, where $h = \max_i h_i$.

By separating the integral, applying the midpoint rule to the first term and the trapezoidal rule for the second term and evaluating the second integral, we get the following,

$$T = \int_0^1 a(x_{i+\frac{1}{2}}) \left(\frac{v_{i+1} - v_i}{h_i} \right)^2 dx - 2 \left(\frac{h_i}{2}(v_i q(x_i) + v_{i+1} q(x_{i+1})) \right).$$

Since we are evaluating at every subinterval and since we are finding area under a curve (we are evaluating an integral), we need to include the width with the area that we already have the height of which is h_i. So evaluating the first integral gives the following,

$$T = \sum_{i=0}^{J-1} h_i a(x_{i+\frac{1}{2}}) \left(\frac{v_{i+1} - v_i}{h_i} \right)^2 - h_i(v_i q(x_i) + v_{i+1} q(x_{i+1}))$$

and gives us the end result of

$$T_h = \sum_{i=0}^{J-1} a(x_{i+\frac{1}{2}}) \frac{(v_{i+1} - v_i)^2}{h_i} - h_i(v_i q(x_i) + v_{i+1} q(x_{i+1})).$$

(b) Obtain the necessary conditions

$$a(x_{j+\frac{1}{2}}) \frac{v_j - v_{j+1}}{h_j} + a(x_{j-\frac{1}{2}}) \frac{v_j - v_{j-1}}{h_{j-1}} = \frac{h_j + h_{j-1}}{2} q(x_j)$$

for $j = 1, \ldots, J$, where we set $v_{J+1} = v_J$.

Since we are minimizing the equation in part a, we will take the derivative of T_h and equal it to zero. Since T_h involves a sum, we will need to look at different instances of the index i.
First let $i = j - 1$.

$$T_h = a(x_{j-\frac{1}{2}}) \frac{(v_j - v_{j-1})^2}{h_{j-1}} - h_{j-1}(v_{j-1} q(x_{j-1}) + v_j q(x_j))$$

$$\frac{dT_h}{dv_j} = a(x_{j-\frac{1}{2}}) 2 \frac{v_j - v_{j-1}}{h_{j-1}} - h_{j-1} q(x_j)$$

Then let $i = j$.

$$T_h = a(x_{j+\frac{1}{2}}) \frac{(v_{j+1} - v_j)^2}{h_j} - h_j(v_j q(x_j) + v_{j+1} q(x_{j+1}))$$

$$\frac{dT_h}{dv_j} = a(x + \frac{1}{2})(-2) \frac{v_{j+1} - v_j}{h_j} - h_j q(x_j)$$

Adding these derivate terms together and setting them equal to zero yields the desired equation.

$$a(x_{j+\frac{1}{2}}) \frac{v_j - v_{j+1}}{h_j} + a(x_{j-\frac{1}{2}}) \frac{v_j - v_{j-1}}{h_{j-1}} = \frac{h_j + h_{j-1}}{2} q(x_j)$$

(c) Show that upon writing the above as a linear system of equations $A\mathbf{v} = \mathbf{q}$, the matrix A is tridiagonal, symmetric, and positive definite despite the arbitrary nonuniformity of the mesh.

From the equation in part b, we group together the terms corresponding to v_{i-1}, v_i, and v_{i+1} and obtain the following,

$$\frac{h_i + h_{i-1}}{2} q(x_i) = \left(\frac{-a(x_{i-\frac{1}{2}})}{h_{i-1}} \right) v_{i-1} + \left(\frac{a(x_{i+\frac{1}{2}})}{h_i} + \frac{a(x_{i-\frac{1}{2}})}{h_{i-1}} \right) v_i + \left(\frac{-a(x_{i+\frac{1}{2}})}{h_i} \right) v_{i+1}$$

which gives the coefficients to build the matrix A. We know that A is tridiagonal because it only has three terms that shift in every row. We can show that it is symmetric by checking the $i-1$ and $i+1$ rows and comparing terms.
Row $i-1$:

$$\frac{h_{i+1} + h_i}{2} q(x_{i+1}) = \left(\frac{-a(x_{i-\frac{3}{2}})}{h_{i-2}} \right) v_{i-2} + \left(\frac{a(x_{i-\frac{1}{2}})}{h_{i-1}} + \frac{a(x_{i-\frac{3}{2}})}{h_{i-2}} \right) v_{i-1} + \left(\frac{-a(x_{i-\frac{1}{2}})}{h_{i-1}} \right) v_i$$

Row $i+1$:

$$\frac{h_{i-1} + h_{i-2}}{2} q(x_{i-1}) = \left(\frac{-a(x_{i+\frac{1}{2}})}{h_i} \right) v_i + \left(\frac{a(x_{i+\frac{3}{2}})}{h_{i+1}} + \frac{a(x_{i+\frac{1}{2}})}{h_i} \right) v_{i+1} + \left(\frac{-a(x_{i+\frac{3}{2}})}{h_{i+1}} \right) v_{i+2}$$

Since the ith-1 row has the same coefficient for v_i as the v_{i-1} term in the ith row, and the ith$+1$ row has the same coefficient for v_i as the v_{i+1} term in the ith row, A is symmetric. This can be seen in the matrix representation of these rows below.

$$\begin{bmatrix} \frac{-a(x_{i-\frac{3}{2}})}{h_{i-2}} & \frac{a(x_{i-\frac{1}{2}})}{h_{i-1}} + \frac{a(x_{i-\frac{3}{2}})}{h_{i-2}} & \frac{-a(x_{i-\frac{1}{2}})}{h_{i-1}} & & \\ & \frac{-a(x_{i-\frac{1}{2}})}{h_{i-1}} & \frac{a(x_{i+\frac{1}{2}})}{h_i} + \frac{a(x_{i-\frac{1}{2}})}{h_{i-1}} & \frac{-a(x_{i+\frac{1}{2}})}{h_i} & \\ & & \frac{-a(x_{i+\frac{1}{2}})}{h_i} & \frac{a(x_{i+\frac{3}{2}})}{h_{i+1}} + \frac{a(x_{i+\frac{1}{2}})}{h_i} & \frac{-a(x_{i+\frac{3}{2}})}{h_{i+1}} \end{bmatrix}$$

To test if A is positive definite, we will use the following theorem. A symmetric matrix is positive definite if each diagonal entry is equal to the sum of the absolute values of all other entries in the row/column in which it appears with the exception of at least one row where it is greater. This is true for A due to our boundary conditions which makes the first entry in the first row 1 followed by zeros. We can also see that the rest of the rows are equal.

2. Convince yourself by running a computational example that the solution $\mathbf{v}$ is a second order accurate despite what Exercise 3 may suggest. Then try to prove it.

For your computational experiments, take $a(x) = 1.0 + e^z$ and $q(x) = \pi^2 \sin(\pi x)$. Compare using a uniform mesh with an element spacings of $\frac{1}{20}, \frac{1}{40}, \frac{1}{80}$, and $\frac{1}{160}$ and a set of four non-uniform meshes generated by golden ratio subdivision. To generate your non-uniform meshes, your base mesh should be a uniform mesh with spacing of $\frac{1}{20}$. Let a depth-1 subdivision be the splitting of these 20 elements into 40 elements using the golden ratio partitioning on each element. Recursively design your four non-uniform meshes from depth-1 to depth-4. Comment on your results.

The following figure shows the solution $\mathbf{v}$ on a uniform mesh at different mesh refinements.

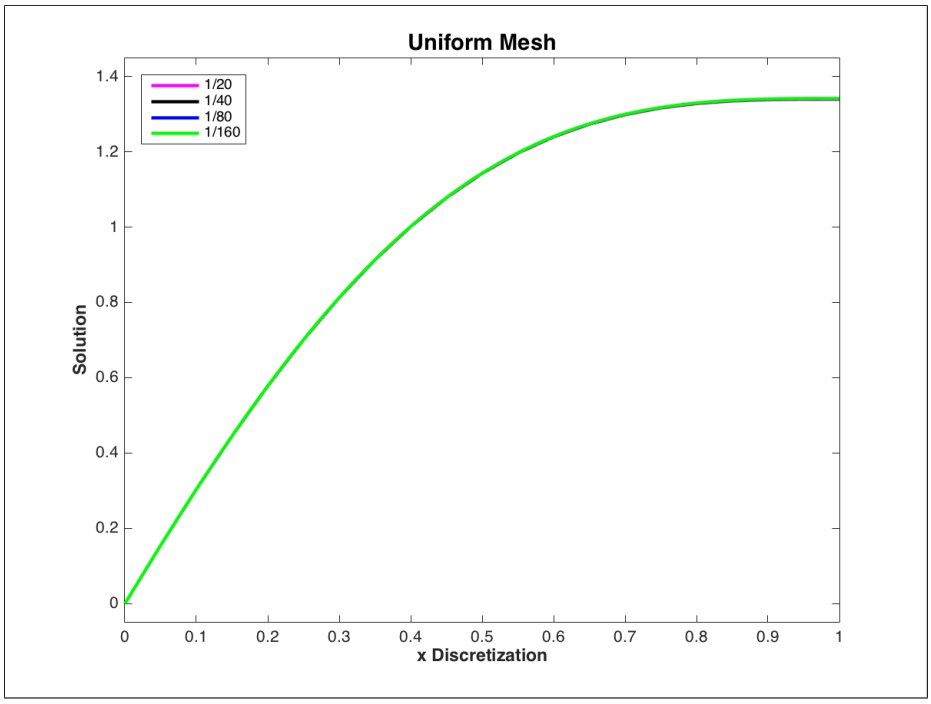

Figure 1. Solution on a Uniform Mesh

The next figure shows the solution $\mathbf{v}$ on a nonuniform mesh generated by golden ratio subdivision with a uniform base mesh with spacing of $h = \frac{1}{20}$ with different depth refinements.

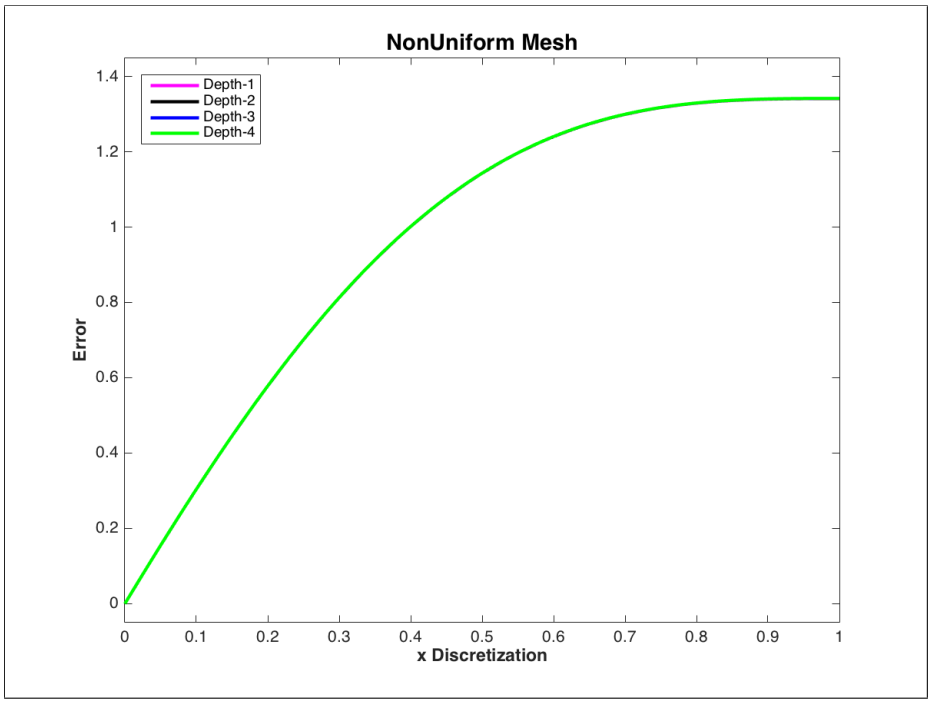

Figure 2. Solution on a Nonuniform Mesh

We can see that both types of meshes at any depth or step size produce precise and smooth solutions.

To convince myself that the solution $\mathbf{v}$ is second order accurate, I calculated the solution $\mathbf{v}$ with a uniform mesh with a step size of $h = \frac{1}{640}$ and took that to be the *true* solution. I compared the coarser meshes at each point to their respective points on the fine mesh. I took the maximum of this error and have shown it in the figures below. I plotted this data on a log log plot and to show that it is second order accurate and compared it to step size squared. We can see that these lines clearly have the same slope. It also has the shape we would expect. As the discretization gets larger, so does the error. So the solution $\mathbf{v}$ is second order accurate. The same is also true for our nonuniform mesh. I compared the nonuniform mesh error to the uniform step sizes squared as well, even though it doesn't make as much sense, but it does have the same shape and slope which shows that it also confirms that $\mathbf{v}$ is second order accurate.

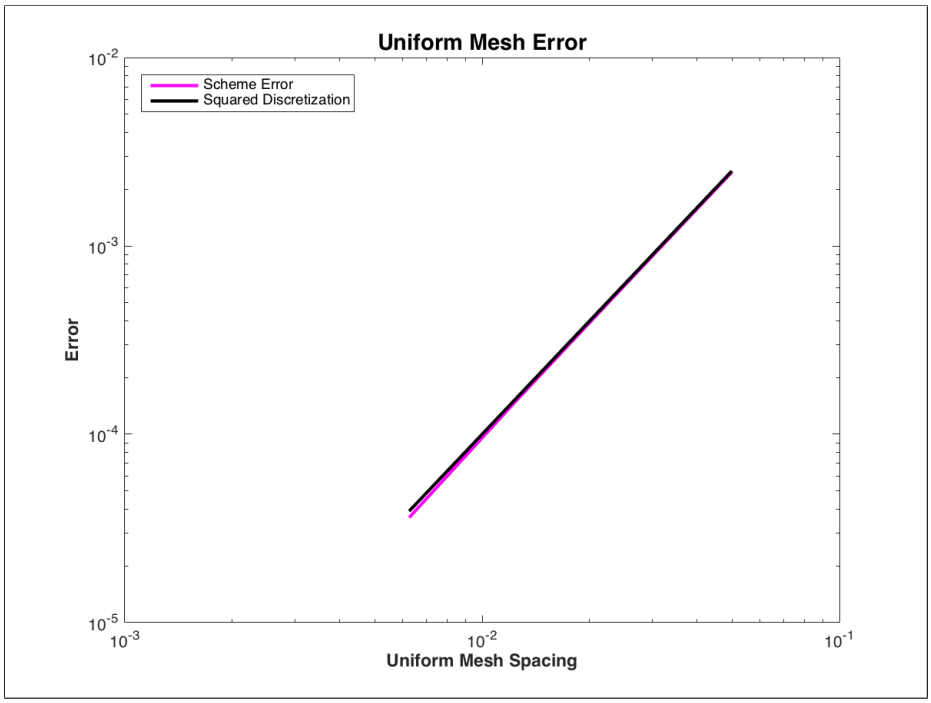

Figure 3. Uniform Mesh Error

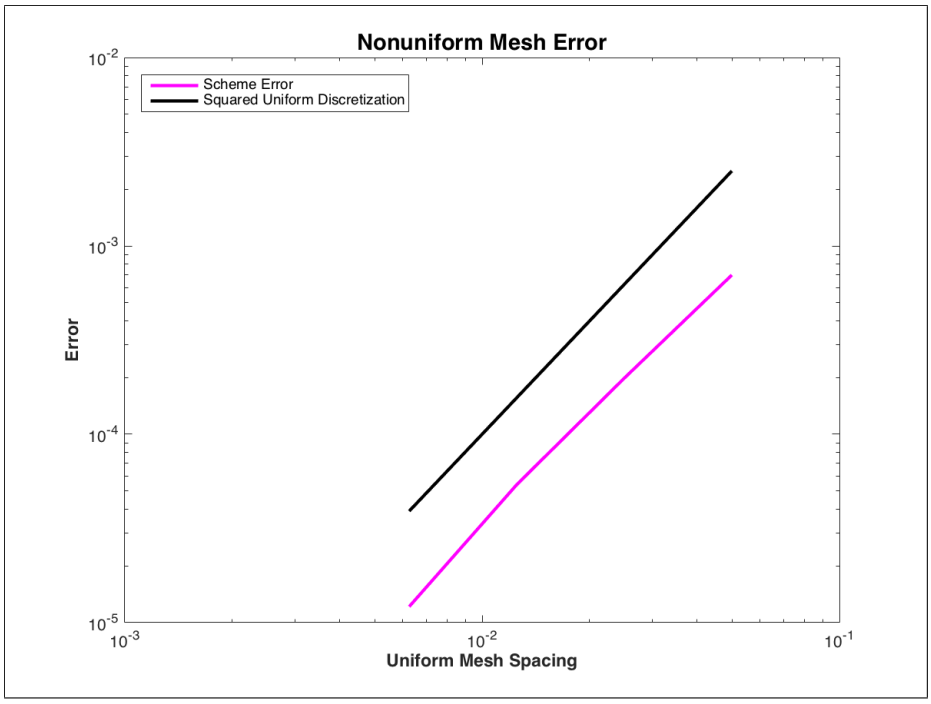

Figure 4. NonUniform Mesh Error

To *try* to prove that the solution $\mathbf{v}$ is second order accurate, we must recall from problem 1, part a that we discretized the first part of the integral with the midpoint rule which has linear error and equates to the following,

$$u'_{\text{midpoint}} = \frac{u_{j+1} - u_j}{h_j} + \mathcal{O}(h)$$

which is squared within the integral and therefore squares the error resulting in a second order accurate solution.

The MATLAB code for this problem is shown below.

```
function problem2_main
%Runs the uniform and nonuniform meshes and caluclates error

%close any figures
close all

%uniform mesh first

%Different levels of mesh spacing and colors for plotting
uniformMeshSpacing = [20,40,80,160];
colors = ['m','k','b','g'];

% "True solution"
[solution_640,x_640] = problem2_uniform(640);

uniform_error = zeros(length(uniformMeshSpacing),1);

%error stuff
for j = 1:length(uniformMeshSpacing)
    [solution,x] = problem2_uniform(uniformMeshSpacing(j));
    subVec = zeros(length(solution),1);
    for i = 1:length(solution)
        if i == 1
            subVec(1) = solution_640(1);
        elseif i == length(solution)
            subVec(end) = solution_640(end);
        else
            subVec(i) = solution_640(x_640 == x(i));
        end
    end
    uniform_error(j) = max(abs(subVec - solution));
    plot(x,solution,colors(j),'LineWidth',2)
    hold on
end

%plotting and saving
```

```
legend('1/20','1/40','1/80','1/160','Location','NorthWest')
title('Uniform Mesh','FontSize',14,'FontWeight','bold')
ylim([-0.05,1.45])
xlabel('x Discretization','FontWeight','bold')
ylabel('Solution','FontWeight','bold')
saveas(gcf,'uniform.png')

figure
loglog(1./uniformMeshSpacing,uniform_error,'m','LineWidth',2)
hold on
loglog(1./uniformMeshSpacing,(1./uniformMeshSpacing).^2,'k','LineWidth',2)
legend('Scheme Error','Squared Discretization','Location','NorthWest')
title('Uniform Mesh Error','FontWeight','bold','FontSize',14)
xlabel('Uniform Mesh Spacing','FontWeight','bold')
ylabel('Error','FontWeight','bold')
saveas(gcf,'uniform_error.png')

%nonuniform mesh
depth = 1:4;

%"True solution'
[solution_5,x_5] = problem2_nonuniform(5);

nonuniform_error = zeros(length(depth),1);

%error stuff
figure
for j = 1:length(depth)
    [solution_non,x_non] = problem2_nonuniform(depth(j));
    %error(j) = max(abs(solution(j) - solution_320(1:2^(5-j):end)));
    subVec = zeros(length(solution_non),1);
    for i = 1:length(solution_non)
        if i == 1
            subVec(1) = solution_5(1);
        elseif i == length(solution_non)
            subVec(end) = solution_5(end);
        else
            subVec(i) = solution_5(x_5 == x_non(i));
        end
    end
    nonuniform_error(j) = max(abs(subVec - solution_non));
    plot(x_non,solution_non,colors(j),'LineWidth',2)
    hold on
end

%plotting and saving
title('NonUniform Mesh','FontSize',14,'FontWeight','bold')
legend('Depth-1','Depth-2','Depth-3','Depth-4','Location','NorthWest')
ylim([-0.05,1.45])
xlabel('x Discretization','FontWeight','bold')
```

```
ylabel('Solution','FontWeight','bold')
saveas(gcf,'nonuniform.png')

figure
loglog(1./uniformMeshSpacing,nonuniform_error,'m','LineWidth',2)
hold on
loglog(1./uniformMeshSpacing,(1./uniformMeshSpacing).^2,'k','LineWidth',2)
legend('Scheme Error','Squared Uniform Discretization',...
    'Location','NorthWest')
title('Nonuniform Mesh Error','FontWeight','bold','FontSize',14)
xlabel('Uniform Mesh Spacing','FontWeight','bold')
ylabel('Error','FontWeight','bold')
saveas(gcf,'nonuniform_error.png')

end


function [solution,x] = problem2_uniform(uniformMeshSpacing)
%Assignment 4, problem 2, uniform mesh

%mesh discretization
h = 1/uniformMeshSpacing;
x = 0:h:1;

%functions a and q to solve ODE and build coefficient matrix
a = @(x)1.0 + exp(x);
q = pi^2 * sin(pi *x);

%coefficient matrix building
coeffMat = zeros(length(x),length(x));

coeffMat(1,1) = 1; %c_j;
coeffMat(1,2) = 0; %c_jplus1;
coeffMat(1,length(x)) = 0; %c_jmin1;

for i = 2:length(x)-1
    jminus = (x(i)+x(i-1))/2;
    jplus = (x(i+1)+x(i))/2;

    coeffMat(i,i-1) = -a(jminus)/h;
    coeffMat(i,i) = (a(jminus)/h) + (a(jplus)/h);
    coeffMat(i,i+1) = -a(jplus)/h;

    q(i) = (h+h)/2 * pi^2*sin(pi*x(i));
end

coeffMat(length(x),1) = 0; %c_jplus1;
coeffMat(length(x),length(x)-1) = -1; %c_jmin1;
coeffMat(length(x),length(x)) = 1; %c_j;
```

```
%solve for solution vector
solution = coeffMat\q';

end


function [solution,x] = problem2_nonuniform(depth)
%Assignment 4, problem 2, nonuniform mesh

%Setting the base mesh step size (h). The golden ratio is used to make
%the mesh nonuniform.
goldenRatio = (1+sqrt(5))/2;
h = 1/20;

%base mesh
x = 0:h:1;

%Nonuniform step sizing with the golden ratio
for j = 1:depth
    addedX = zeros(1,length(x)-1);
    for k = 1:length(x)-1
        addedX(k) = (goldenRatio*x(k+1) + x(k))/(1+goldenRatio);
    end
    x = sort([x,addedX]);
end

%q and a functions to solve ODE and make coefficient matrix
a = @(x)1.0 + exp(x);
q = pi^2 * sin(pi *x);

%coefficient matrix building
coeffMat = zeros(length(x),length(x));

coeffMat(1,1) = 1; %c_j;
coeffMat(1,2) = 0; %c_jplus1;
coeffMat(1,length(x)) = 0; %c_jmin1;

for l = 2:length(x)-1
    jminus = (x(l)+x(l-1))/2;
    jplus = (x(l+1)+x(l))/2;
    hminus = x(l) - x(l-1);
    hj = x(l+1) - x(l);

    coeffMat(l,l-1) = -a(jminus)/hminus;
    coeffMat(l,l) = (a(jplus)/hj) + (a(jminus)/hminus);
    coeffMat(l,l+1) = -a(jplus)/hj;

    q(l) = (hj+hminus)/2 * pi^2*sin(pi*x(l));
end
```

```
coeffMat(length(x),1) = 0; %c_jplus1;
coeffMat(length(x),length(x)-1) = -1; %c_jmin1;
coeffMat(length(x),length(x)) = 1; %c_j;

%solve for solution vector
solution = coeffMat\q';

end
```

3. ### 0.1 Purpose/Statement of Problem

For the problem and notation of the previous exercise, define the **hat function** in one dimension:

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{h_{i-1}}, & x_{i-1} \le x < x_i, \\ \frac{x_{i+1}-x}{h_i}, & x_i \le x < x_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Derive the Galerkin finite element method for the boundary value ODE. Show that the stiffness matrix A is tridiagonal, symmetric, and positive definite. How does this method relate to the method of the previous exercise?

Write a one-dimensional FEM solver based upon the procedure given in class and repeat the computational experiment above. Comment on your results (and the comparison with the previous problem).

0.2 Description of Mathematics

The ODE we are aiming to solve is

$$-(au')' = q, \quad 0 < x < 1,$$

$$u(0) = 0, \quad u'(1) = 0,$$

where $a(x) > 0$ and $q(x)$ are known, smooth functions. For our computational and mathematical purposes, $a(x) = 1 + e^x$ and $q(x) = \pi^2 \sin(\pi x)$.

To derive a Galerkin finite element method, we must multiply each side by the **hat function** and evaluate the integral within the space boundary.

$$\int_0^1 (au')'\phi_j dx = -\int_0^1 q\phi_j dx$$

Integration by parts magic gives the following,

$$au'\phi_j\big|_0^1 - \int_0^1 au'\phi_j' dx = -\int_0^1 q\phi_j dx.$$

Due to the boundary conditions, the first term equates to zero. If we divide by -1 we get the following,

$$\int_0^1 au'\phi_j' dx = \int_0^1 q\phi_j dx.$$

We now define $u = \sum_{i=0}^{J} \hat{u}_i\phi_i$ which is the basis decomposition of u with ϕ. This would result in $u' = \sum_{i=0}^{J} \hat{u}_i\phi_i'$. Due to our boundary conditions the limits must be changed resulting in

$$u = \sum_{i=1}^{J-1} \hat{u}_i\phi_i, \quad u' = \sum_{i=1}^{J-1} \hat{u}_i\phi_i'.$$

We plug this into our above equation,

$$\int_0^1 \left(\sum_{i=1}^{J-1} a\hat{u}_i\phi_i'\right)\phi_j' dx = \int_0^1 q\phi_j dx.$$

Since both the sum and the integral are finite, they can be switched.

$$\sum_{i=1}^{J-1} \left(\int_0^1 a\phi_i'\phi_j' dx\right)\hat{u}_i = \int_0^1 q\phi_j dx.$$

We can see that we now have a system of equations to solve for our solution $\hat{u}_i$. Our stiffness matrix will be the matrix with rows $\left(\int_0^1 a\phi_i'\phi_j' dx\right)$. Calculating both the stiffness matrix and the right hand side is hefty so we will do one at a time. Let's start with the right hand side.

$$\int_0^1 q\phi_j dx = \int_0^1 \pi^2 \sin(\pi x)\phi_j dx = \int_{x_{j-1}}^{x_{j+1}} \pi^2 \sin(\pi x)\phi_j dx$$

We break this integral up because of the **hat function**.

$$= \int_{x_{j-1}}^{x_j} \pi^2 \sin(\pi x)\frac{(x - x_{j-1})}{h_{j-1}} dx + \int_{x_j}^{x_{j+1}} \pi^2 \sin(\pi x)\frac{(x_{j+1} - x)}{h_j} dx$$

By integration by parts and simplification,

$$\int_0^1 q\phi_j dx = \frac{1}{h_{j-1}}(\sin(\pi x_j) - \sin(\pi x_{j-1})) + \frac{1}{h_j}(\sin(\pi x_j) - \sin(\pi x_{j+1})).$$

Now let's move on to the stiffness matrix. To get our coefficient terms for each row, there are three cases that we must consider because of the **hat function**.

(a) **Case 1:** $j = i - 1$

$$\int_0^1 a\phi_i'\phi_{i-1}'dx = \int_{x_{i-1}}^{x_i}(1+e^x)\left(\frac{1}{h_{j-1}}\right)\left(-\frac{1}{h_{j-1}}\right)dx = -\frac{1}{h_{j-1}^2}(h_{j-1}+e^{x_i}-e^{x_{i-1}})$$

(b) **Case 2:** $j = 1$

$$\int_0^1 a(\phi_i')^2dx = \int_{x_{i-1}}^{x_{i+1}} a(\phi_i')^2dx = \int_{x_{i-1}}^{x_i} a(\phi_i')^2dx + \int_{x_i}^{x_{i+1}} a(\phi_i')^2dx$$

$$= \int_{x_{i-1}}^{x_i}(1+e^x)\left(\frac{1}{h_{j-1}}\right)^2dx + \int_{x_i}^{x_{i+1}}(1+e^x)\left(\frac{1}{h_j}\right)^2$$

$$= \frac{1}{h_{j-1}^2}(h_{j-1}+e^{x_i}-e^{x_{i-1}}) + \frac{1}{h_j^2}(h_j+e^{x_{i+1}}-e^{x_i})$$

(c) **Case 3:** $j = i + 1$

$$\int_0^1 a\phi_i'\phi_{i+1}'dx = \int_{x_i}^{x_{i+1}}(1+e^x)\left(\frac{1}{h_j}\right)\left(-\frac{1}{h_j}\right)dx = -\frac{1}{h_j^2}(h_j+e^{x_{i+1}}-e^{x_i})$$

We can see that the stiffness matrix is tridiagonal because it only has three terms in each row centered around the diagonal term.

We can show that the matrix is symmetric by comparing locations at $(i, i+1)$ and $(i+1, i)$.

$$S_{(i,i+1)} = -\frac{1}{h_i^2}(h_i + e^{x_{i+1}} - e^{x_i})$$

Let $j = i + 1$ so that $S_{(i,i+1)} = S_{(j,j-1)}$.

$$S_{(j,j-1)} = -\frac{1}{h_{j-1}^2}(h_{j-1} + e^{x_j} - e^{x_{j-1}})$$

Putting this back into terms of i,

$$S_{(i,i+1)} = -\frac{1}{h_i^2}(h_i + e^{x_{i+1}} - e^{x_i}).$$

So the stiffness matrix is symmetric.

To test if the stiffness matrix is positive definite, we will use the following theorem. A symmetric matrix is positive definite if each diagonal entry is equal to the sum of the absolute values of all other entries in the row/column in which it appears with the exception of at least one row where it is greater. This is true for the stiffness matrix due to the boundary conditions which makes the first entry in the first row 1 followed by zeros. We can also see that the rest of the rows are equal.

0.3 Description of Algorithms

In my implementation of an FEM solver, I used essentially the same algorithm as the previous problem, but changed coefficients and my evaluation of the right hand side. First, I calculated the solution $\mathbf{v}$ with a uniform mesh with a step size of $h = \frac{1}{640}$ and took that to be the *true* solution. I then calculated the solution $\mathbf{v}$ at each mesh refinement. I compared the coarser meshes at each point to their corresponding points on the fine mesh. I also did this for the nonuniform mesh, but took a depth of 5 to be the *true* solution.

To compute the uniform solutions and nonuniform solutions, I wrote two separate methods to keep things simple. In both methods, I set the stiffness matrix to have the first and last row to satisfy the boundary conditions. I then set each of the tridiagonal entries to their respective coefficients outlined in the Mathematics section. The nonuniform section will have a specification of the corresponding h values. While I am setting up the stiffness matrix, I am also calculating the corresponding right hand side values. I then solve using both the stiffness matrix and the right hand side.

$$x = A^{-1}b$$

To compute the nonuniform meshes I calculated the "midpoints" using a ratio of $\phi : 1$, where ϕ is the golden ratio. This resulted in the following equation for the midpoints:

$$x_{mid} = \frac{\phi x_{i+1} + x_i}{1 + \phi}.$$

I began with a base mesh of $h = \frac{1}{20}$. Then I made a new array of the x_{mid}'s, concatenated this to the base mesh and sorted it. I do this again on the concatenated, sorted array for another level of depth.

0.4 Demonstration of the Correctness of Implementation

Since this problem was the same computational experiment as the previous problem, I compared their solution results and saw that they were the same.

0.5 Results/Analysis of Results

The following figure shows the solution $\mathbf{v}$ on a uniform mesh at different mesh refinements solved using a 1D FEM solver.

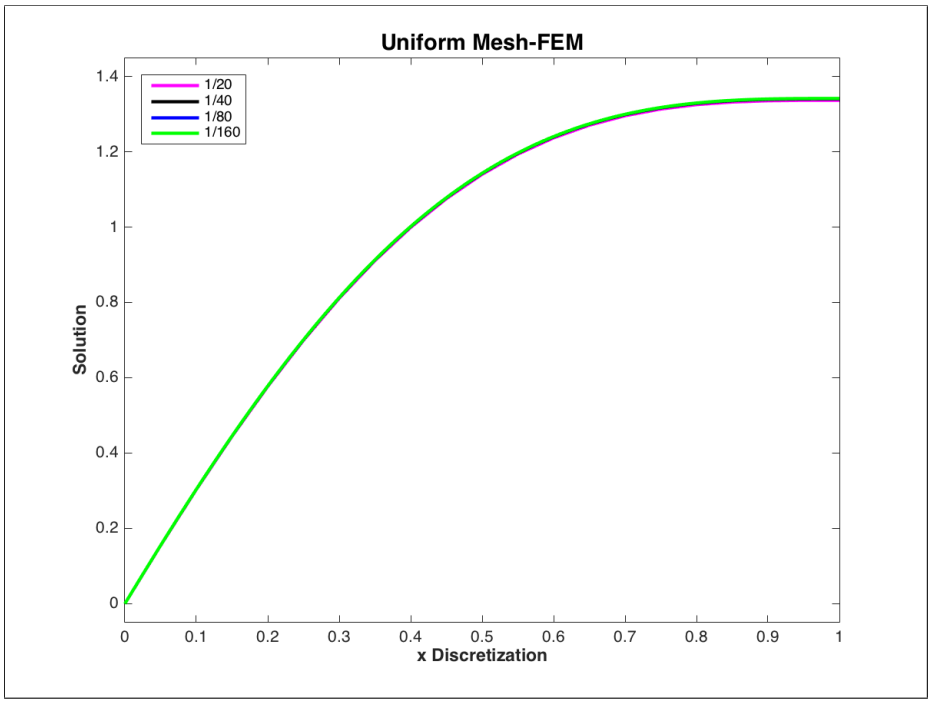

Figure 5. Solution on a Uniform Mesh with FEM

We can see that the functions are smooth and very precise throughout the different mesh refinements. This shape is what we expected to see.

The next figure shows the solution $\mathbf{v}$ on a nonuniform mesh generated by golden ratio subdivision with a uniform base mesh with spacing of $h = \frac{1}{20}$ with different depth refinements solved using a 1D FEM solver.

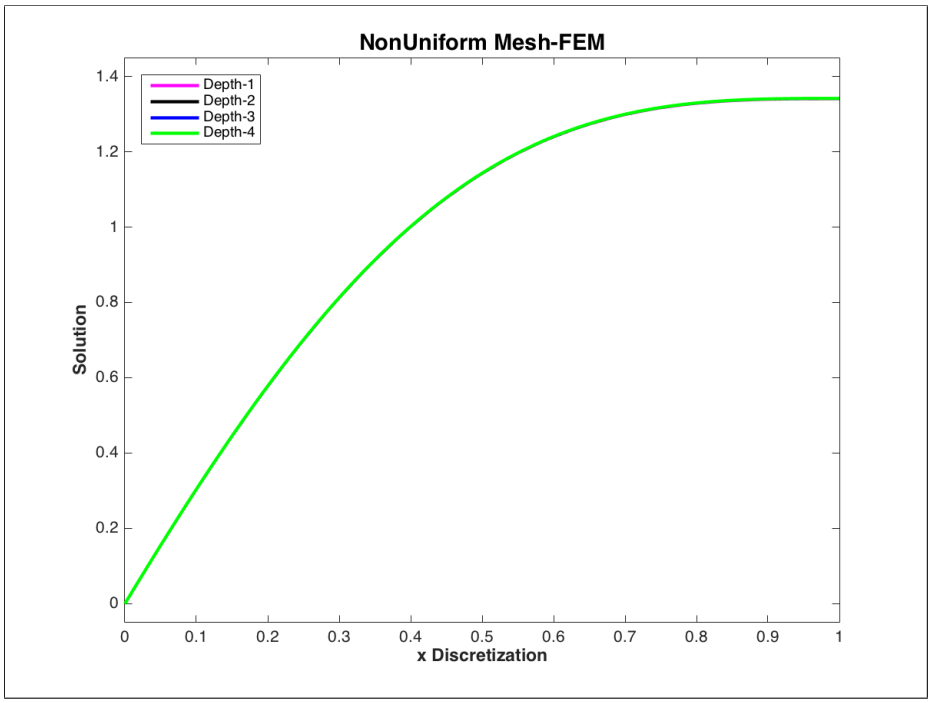

Figure 6. Solution on a Nonuniform Mesh with FEM

We can see, again, that the functions are smooth and very precise throughout the different depths. This shape is what we expected to see. The Nonuniform mesh appears to be more precise than the uniform mesh through out the different refinements.

To compare error, like in the previous problem, I took the max of the error for each mesh refinement and have shown it in the figures below. I plotted this data on a log log plot and to show that it is second order accurate, I compared it to step size squared. We can see that these lines have the same slope. It also has the shape we would expect. As the discretization gets larger, so does the error. So the solution $\mathbf{v}$ is second order accurate. The same is also true for our nonuniform mesh. I compared the nonuniform mesh error to the uniform step sizes squared as well, even though it doesn't make as much sense, but it does have a similar shape and slope. It appears that the nonuniform mesh is somewhere inbetween first and second order accurate.

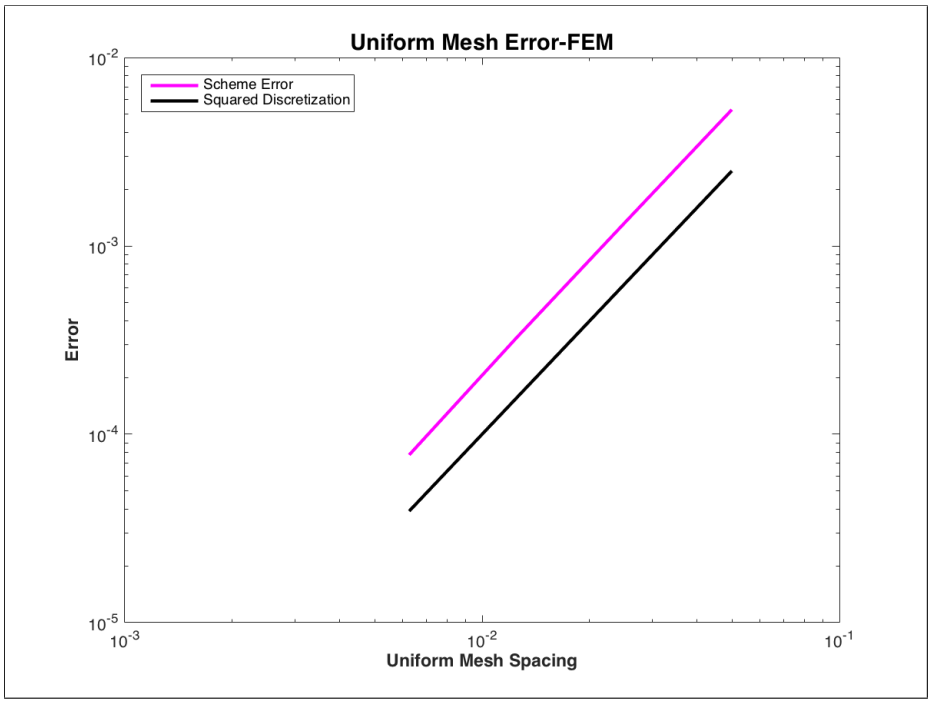

Figure 7. Uniform Mesh Error with FEM

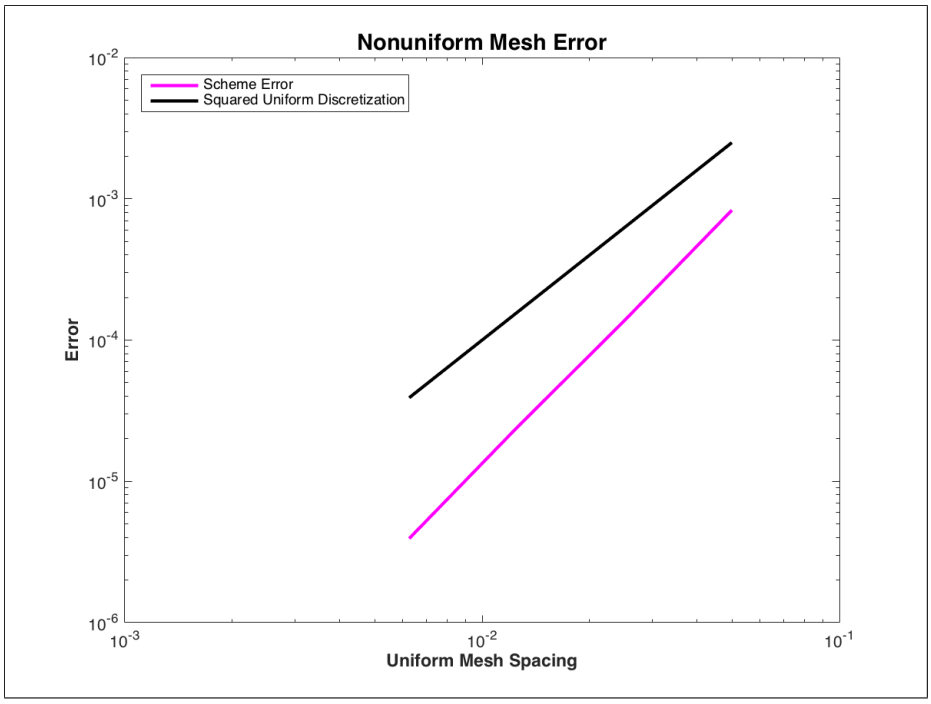

Figure 8. NonUniform Mesh Error with FEM

0.6 Summary and Conclusions

Using a 1D FEM solver did not change the shape or size of our solution, but there were changes in error. There was actually less error using an FEM solver with a uniform mesh (in comparison to the previous problem), but it was still second order accurate. Although things did get a bit wonky with the nonuniform mesh being between first and second order accurate using an FEM solver.

The MATLAB code for this problem is shown below.

```
function problem3_main
%Runs the uniform and nonuniform meshes and caluclates error with an FEM
%solver.

%close any figures
close all

%uniform mesh first

%Different levels of mesh spacing and colors for plotting
uniformMeshSpacing = [20,40,80,160];
colors = ['m','k','b','g'];

% "True solution"
[solution_640,x_640] = problem3_uniform(640);

uniform_error = zeros(length(uniformMeshSpacing),1);

%error stuff
for j = 1:length(uniformMeshSpacing)
    [solution,x] = problem3_uniform(uniformMeshSpacing(j));
    subVec = zeros(length(solution),1);
    for i = 1:length(solution)
        if i == 1
            subVec(1) = solution_640(1);
        elseif i == length(solution)
            subVec(end) = solution_640(end);
        else
            subVec(i) = solution_640(x_640 == x(i));
        end
    end
    uniform_error(j) = max(abs(subVec - solution));
    plot(x,solution,colors(j),'LineWidth',2)
    hold on
end

%plotting and saving
legend('1/20','1/40','1/80','1/160','Location','NorthWest')
```

```
title('Uniform Mesh—FEM','FontSize',14,'FontWeight','bold')
ylim([—0.05,1.45])
xlabel('x Discretization','FontWeight','bold')
ylabel('Solution','FontWeight','bold')
saveas(gcf,'uniform_FEM.png')

figure
loglog(1./uniformMeshSpacing,uniform_error,'m','LineWidth',2)
hold on
loglog(1./uniformMeshSpacing,(1./uniformMeshSpacing).^2,'k','LineWidth',2)
legend('Scheme Error','Squared Discretization','Location','NorthWest')
title('Uniform Mesh Error—FEM','FontWeight','bold','FontSize',14)
xlabel('Uniform Mesh Spacing','FontWeight','bold')
ylabel('Error','FontWeight','bold')
saveas(gcf,'uniform_error_FEM.png')

%nonuniform mesh
depth = 1:4;

%"True solution'
[solution_5,x_5] = problem3_nonuniform(5);

nonuniform_error = zeros(length(depth),1);

%error stuff
figure
for j = 1:length(depth)
    [solution_non,x_non] = problem3_nonuniform(depth(j));
    subVec = zeros(length(solution_non),1);
    for i = 1:length(solution_non)
        if i == 1
            subVec(1) = solution_5(1);
        elseif i == length(solution_non)
            subVec(end) = solution_5(end);
        else
            subVec(i) = solution_5(x_5 == x_non(i));
        end
    end
    nonuniform_error(j) = max(abs(subVec — solution_non));
    plot(x_non,solution_non,colors(j),'LineWidth',2)
    hold on
end

%plotting and saving
title('NonUniform Mesh—FEM','FontSize',14,'FontWeight','bold')
legend('Depth—1','Depth—2','Depth—3','Depth—4','Location','NorthWest')
ylim([—0.05,1.45])
xlabel('x Discretization','FontWeight','bold')
ylabel('Solution','FontWeight','bold')
saveas(gcf,'nonuniform_FEM.png')
```

```
figure
loglog(1./uniformMeshSpacing,nonuniform_error,'m','LineWidth',2)
hold on
loglog(1./uniformMeshSpacing,(1./uniformMeshSpacing).^2,'k','LineWidth',2)
legend('Scheme Error','Squared Uniform Discretization',...
    'Location','NorthWest')
title('Nonuniform Mesh Error','FontWeight','bold','FontSize',14)
xlabel('Uniform Mesh Spacing','FontWeight','bold')
ylabel('Error','FontWeight','bold')
saveas(gcf,'nonuniform_error_FEM.png')

end


function [solution,x] = problem3_uniform(uniformMeshSpacing)
%Assignment 4, problem 2, uniform mesh

%mesh discretization
h = 1/uniformMeshSpacing;
x = 0:h:1;

%functions a and q to solve ODE and build coefficient matrix
%a = @(x)1.0 + exp(x);
q = pi^2 * sin(pi*x);

%coefficient matrix building
coeffMat = zeros(length(x),length(x));

coeffMat(1,1) = 1; %c_j;
coeffMat(1,2) = 0; %c_jplus1;
coeffMat(1,length(x)) = 0; %c_jmin1;

for i = 2:length(x)-1
    %jminus = (x(i)+x(i-1))/2;
    %jplus = (x(i+1)+x(i))/2;

    coeffMat(i,i-1) = (-1/h^2)*(h + exp(x(i)) - exp(x(i-1)));
    coeffMat(i,i) = (1/h^2)*(h + exp(x(i)) - exp(x(i-1))) +...
        (1/h^2)*(h + exp(x(i+1)) - exp(x(i)));
    coeffMat(i,i+1) = (-1/h^2)*(h + exp(x(i+1)) - exp(x(i)));

    q(i) = (1/h)*(sin(pi*x(i)) - sin(pi*x(i-1)))+(1/h)*(sin(pi*x(i)) - ...
        sin(pi*x(i+1)));
end

coeffMat(length(x),1) = 0; %c_jplus1;
coeffMat(length(x),length(x)-1) = -1; %c_jmin1;
coeffMat(length(x),length(x)) = 1; %c_j;
```

```
%solve for solution vector
solution = coeffMat\q';

end


function [solution,x] = problem3_nonuniform(depth)
%Assignment 4, problem 2, nonuniform mesh

%Setting the base mesh step size (h). The golden ratio is used to make
%the mesh nonuniform.
goldenRatio = (1+sqrt(5))/2;
h = 1/20;

%base mesh
x = 0:h:1;

%Nonuniform step sizing with the golden ratio
for j = 1:depth
    addedX = zeros(1,length(x)-1);
    for k = 1:length(x)-1
        addedX(k) = (goldenRatio*x(k+1) + x(k))/(1+goldenRatio);
    end
    x = sort([x,addedX]);
end

%q and a functions to solve ODE and make coefficient matrix
%a = @(x)1.0 + exp(x);
q = pi^2 * sin(pi *x);

%coefficient matrix building
coeffMat = zeros(length(x),length(x));

coeffMat(1,1) = 1; %c_j;
coeffMat(1,2) = 0; %c_jplus1;
coeffMat(1,length(x)) = 0; %c_jmin1;

for l = 2:length(x)-1
    %jminus = (x(l)+x(l-1))/2;
    %jplus = (x(l+1)+x(l))/2;
    hminus = x(l) - x(l-1);
    hj = x(l+1) - x(l);

    coeffMat(l,l-1) = (-1/hminus^2)*(hminus + exp(x(l)) - exp(x(l-1)));
    coeffMat(l,l) = (1/hminus^2)*(hminus + exp(x(l)) - exp(x(l-1))) +...
        (1/hj^2)*(hj + exp(x(l+1)) - exp(x(l)));
    coeffMat(l,l+1) = (-1/hj^2)*(hj + exp(x(l+1)) - exp(x(l)));

    q(l) = (1/hminus)*(sin(pi*x(l)) - ...
        sin(pi*x(l-1)))+(1/hj)*(sin(pi*x(l)) - sin(pi*x(l+1)));
```

```
end

coeffMat(length(x),1) = 0; %c_jplus1;
coeffMat(length(x),length(x)-1) = -1; %c_jmin1;
coeffMat(length(x),length(x)) = 1; %c_j;

%solve for solution vector
solution = coeffMat\q';

end
```