1. (**Convergence of Linear FEM**) Consider the two point BVP that we studied in class, i.e.,

   Given $f \in C^0[0,1]$, find $u \in C^2[0,1]$ such

$$\text{(D)} : \begin{cases} -\dfrac{d^2x}{du^2} = f;\ 0 < x < 1 \\ u(0) = u(1) = 0 \end{cases}$$

   where f is chosen so that the exact solution is $u(x) = \sin(\pi x)$.

   Use the linear FEM code that you have written with $h = 2^{-j}, j = 1, \dots 10$ to find the rate of convergence of the following errors (use numerical integration):

   (a) $||u - u_k||_{L^2(0,1)}$

   (b) $||u - u_k||_E$

   (c) $\max_i |u(x_i) - u_k(x_i)|$

   where $u_k \in V_k^1$ is the Galerkin solution with continuous piecewise linear finite elements and the x_i are the nodes of a partition of $[0,1]$. For each case, plot the error as a function of the mesh step size h (loglog plot). Confirm the theoretical results discussed in class.

Solution

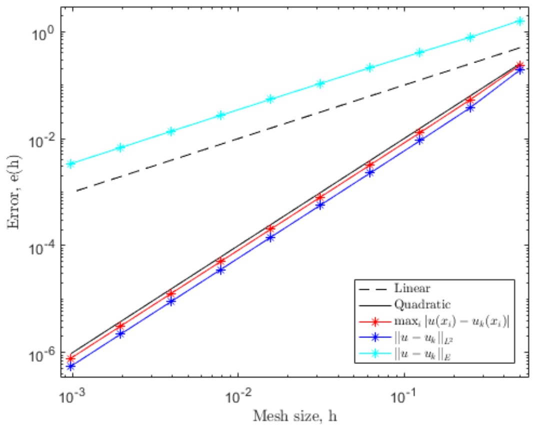


Figure 1: The FEM max and L_2 error exhibits second-order behavior while the energy norm error is first-order.

The L_2 and energy norms were computed using Simpson's 1/3 rule. Additionally, forward differences were used to calculate the derivative of $u - u_k$

for the energy norm. The convergence plots of the max, L_2, and energy errors for ten different mesh sizes are plotted in Figure 1. This plot clearly shows that the L_2 error is second-order while the energy error is instead first-order. This matches with the theoretical error estimates for linear finite elements.

2. (**Relation between Finite Difference and Finite Element Methods** Consider problem 2 from your HW1 (discretization of 1D BVP) with (1) linear finite elements and (2) quadratic finite elements. In each case, assume $f = 1$ and use a uniform (regular) mesh. On a regular mesh, the finite element approximation to the 1D BVP has the following feature: To each degree of freedom there corresponds a discrete equation. Do the following.

   (a) For the case of linear finite elements, show that the ith row of the linear system $AX = F$ is a centered finite difference discretization of the 1D BVP at the ith node of the mesh.

   (b) For the case of quadratic finite elements show that the rows of the corresponding linear system can be sorted into two types of equations. Show that we obtain a discrete equation corresponding to an endpoint x_i of subinterval I_i in the form

$$\frac{1}{h^2}\big(14u_i(t) - 8(u_{i-1/2}(t) + u_{i+1/2}(t)) + u_{i+1}(t) + u_{i-1}(t)\big) = 1,$$

   and another discrete equation corresponding to the midpoint $x_{i+1/2}$ of the interval $[x_i, x_{i+1}]$ given by

$$-\frac{4}{h^2}\big(u_i(t) - 2u_{i+1/2}(t) + u_{i+1}(t)\big) = 1,$$

   where the u_is are the degrees of freedom of the finite element solution.

Solution

(a) The centered difference formula for

$$-\frac{d^2u}{dx^2} = f$$

is given by

$$\frac{-u_{i-1}(t) + 2u_i(t) - u_{i+1}(t)}{h^2} = 1$$

The global stiffness matrix, load vector, and solution vector are given as follows for uniform mesh size, h

$$A = \begin{bmatrix} \frac{2}{h} & \frac{-1}{h} & 0 & 0 & ... & 0 & 0 \\ \frac{-1}{h} & \frac{2}{h} & \frac{-1}{h} & 0 & ... & 0 & 0 \\ 0 & \frac{-1}{h} & \frac{2}{h} & \frac{-1}{h} & ... & 0 & 0 \\ ... & ... & ... & ... & ... & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & \frac{-1}{h} & \frac{2}{h} \end{bmatrix}, \; F = \begin{bmatrix} h \\ ... \\ h \end{bmatrix}$$

$$X = \begin{bmatrix} u_0(t) \\ u_1(t) \\ ... \\ u_i(t) \\ ... \\ u_{k+1}(t) \end{bmatrix}$$

Thus, one row of this linear system may be written as

$$\frac{-u_{i-1}(t) + 2u_i(t) - u_{i+1}(t)}{h} = h$$
$$\Rightarrow \frac{-u_{i-1}(t) + 2u_i(t) - u_{i+1}(t)}{h^2} = 1$$

which is precisely the centered finite difference formula.

(b) I incorrectly stated the global stiffness matrix for quadratic finite elements in Homework 1. I restate it here along with the global load and solution vectors as

$$A = \begin{bmatrix} A^1_{22} & A^1_{23} & 0 & 0 & 0 & 0 & ... \\ A^1_{32} & A^1_{33} + A^2_{11} & A^2_{12} & A^2_{13} & 0 & 0 & ... \\ 0 & A^2_{21} & A^2_{22} & A^2_{23} & 0 & 0 & ... \\ 0 & A^3_{31} & A^2_{32} & A^2_{33} + A^3_{11} & A^3_{12} & A^3_{13} & ... \\ ... & ... & ... & ... & ... & ... & ... \end{bmatrix}, \; F = \begin{bmatrix} F^1_2 \\ F^1_3 + F^2_1 \\ F^2_2 \\ F^2_3 + F^3_1 \\ ... \end{bmatrix}$$

$$X = \begin{bmatrix} u_0(t) \\ u_{\frac{1}{2}}(t) \\ u_1(t) \\ ... \\ u_{i-\frac{1}{2}}(t) \\ u_i(t) \\ u_{i+\frac{1}{2}}(t) \\ ... \\ u_{k+1}(t) \end{bmatrix}$$

where the local stiffness matrix and load vector are given as

$$A^l = \frac{1}{3h} \begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}, \; F^l = \begin{bmatrix} \frac{h}{6} \\ \frac{2h}{3} \\ \frac{h}{6} \end{bmatrix}$$

Thus, two types of equations can be derived from the above system of equations, the first being

$$\begin{aligned}
&A_{31}u_{i-1}(t) + A_{32}u_{i-\frac{1}{2}}(t) + (A_{33} + A11)u_i(t) + A_{12}u_{i+\frac{1}{2}}(t) + A_{13}u_{i+1}(t) = F_3 + F_1 \\
&\quad\Rightarrow \frac{1}{3h}\Bigg[u_{i-1}(t) - 8u_{i-\frac{1}{2}}(t) + (7+7)u_i(t) - 8u_{i+\frac{1}{2}}(t) + u_{i+1}(t)\Bigg] = \frac{h}{3} \\
&\quad\Rightarrow \frac{1}{h^2}\Bigg[u_{i-1}(t) - 8u_{i-\frac{1}{2}}(t) + (7+7)u_i(t) - 8u_{i+\frac{1}{2}}(t) + u_{i+1}(t)\Bigg] = 1
\end{aligned}$$

the second being

$$\begin{aligned}
&A_{21}u_i(t) + A_{22}u_{i+\frac{1}{2}} + A_{23}u_{i+1} = F_2 \\
&\Rightarrow \frac{1}{3h}\Bigg[-8u_i(t) + 16u_{i+\frac{1}{2}}(t) - 8u_{i+1}(t)\Bigg] = \frac{2h}{3} \\
&\Rightarrow \frac{-4}{h}\Bigg[u_i(t) - 2u_{i+\frac{1}{2}}(t) + u_{i+1}(t)\Bigg] = 1
\end{aligned}$$

3. (**Static Condensation**) Consider problem 2 from your HW1 (discretization of 1D BVP with quadratic finite elements). Assume $f = 1$ and consider a partition of $[0, 1]$ into five elements ($k = 4$ interior nodes).

   (a) Split the Galerkin system into two sets of equations

   $$a(u_k, \phi_i) = \ell(\phi_i); 1 \leq i \leq 4 \tag{1}$$
   $$a(u_k, \phi_{i-1/2}) = \ell(\phi_{i-1/2}); 1 \leq i \leq 5 \tag{2}$$

   with u_k, the Galerkin solution, and where (1) are the equations corresponding to basis functions defined at interior nodes and (2) are the equations corresponding to basis functions defined at the midpoints of elements.

   (b) Write the system (1)-(2) in matrix/vector notation as

   $$M_{11}\mathbf{u} + M_{12}\tilde{\mathbf{u}} = \mathbf{b}_1 \tag{3}$$
   $$M_{21}\mathbf{u} + D_{22}\tilde{\mathbf{u}} = \mathbf{b}_2, \tag{4}$$

   where $\mathbf{u} = (u_1, u_2, u_3, u_4)^T$ is the vector of values of the Galerkin solution at the interior nodes, and $\tilde{\mathbf{u}} = (u_{1/2}, u_{3/2}, u_{5/2}, u_{7/2}, u_{9/2})^T$ is the vector of values of the Galerkin solution at the midpoints. Show that the matrix $M_{11} = (a(\phi_j, \phi_i))_{4\times 4}$ is a tridiagonal matrix and the matrix $D_{22} = (a(\phi_{j-1/2}, \phi_{i-1/2}))_{5\times 5}$ is a diagonal matrix. Compute the matrix $M_{12} = (a(\phi_j, \phi_i))_{4\times 5}$ and show that $M_{21} = M_{12}^T$.

   (c) Show that we can eliminate the vector $\tilde{\mathbf{u}}$ from the system (3)-(4) to obtain a system of the form

   $$M\mathbf{u} = \mathbf{b}$$

   with M a tridiagonal matrix. What is the relation of the matrix M and the vector $\mathbf{b}$ in terms of the matrices and vectors in system (3)-(4)?

   (d) Solve the system $M\mathbf{u} = \mathbf{b}$. Also, solve the original Galerkin system and comment on the two solutions.

   (e) (optional) Show that the matrix M is positive definite.

Solution

(a) For each i in $a(u_k, \phi_i) = a(u_j\phi_j, \phi_i)$, the corresponding equation is given by

$$a(u_j\phi_j, \phi_i) = \sum_{j=1}^{4} u_j(\phi_j, \phi_i) + \sum_{j=1}^{5} u_{j-\frac{1}{2}}(\phi_{j-\frac{1}{2}}, \phi_i) = l(\phi_i)$$

For each i in $a(u_k, \phi_{i-\frac{1}{2}}) = a(u_j\phi_j, \phi_{i-\frac{1}{2}})$, the corresponding equation is given by

$$a(u_j\phi_j, \phi_{i-\frac{1}{2}}) = \sum_{j=1}^{4} u_j(\phi_j, \phi_{i-\frac{1}{2}}) + \sum_{j=1}^{5} u_{j-\frac{1}{2}}(\phi_{j-\frac{1}{2}}, \phi_{i-\frac{1}{2}}) = l(\phi_{i-\frac{1}{2}})$$

(b) The matrices M_{11}, M_{12}, M_{21}, and D_{22} are found to be

$$M_{11} = \begin{bmatrix} (\phi_1,\phi_1) & (\phi_1,\phi_2) & 0 & 0 \\ (\phi_2,\phi_1) & (\phi_2,\phi_2) & (\phi_2,\phi_3) & 0 \\ 0 & (\phi_3,\phi_2) & (\phi_3,\phi_3) & (\phi_3,\phi_4) \\ 0 & 0 & (\phi_4,\phi_3) & (\phi_4,\phi_4) \end{bmatrix} = \frac{1}{3h}\begin{bmatrix} 14 & 1 & 0 & 0 \\ 1 & 14 & 1 & 0 \\ 0 & 1 & 14 & 1 \\ 0 & 0 & 1 & 14 \end{bmatrix}$$

$$M_{12} = \begin{bmatrix} (\phi_1,\phi_{\frac{1}{2}}) & (\phi_1,\phi_{\frac{3}{2}}) & 0 & 0 & 0 \\ 0 & (\phi_2,\phi_{\frac{3}{2}}) & (\phi_2,\phi_{\frac{5}{2}}) & 0 & 0 \\ 0 & 0 & (\phi_3,\phi_{\frac{5}{2}}) & (\phi_3,\phi_{\frac{7}{2}}) & 0 \\ 0 & 0 & 0 & (\phi_4,\phi_{\frac{7}{2}}) & (\phi_4,\phi_{\frac{9}{2}}) \end{bmatrix} = \frac{1}{3h}\begin{bmatrix} -8 & -8 & 0 & 0 & 0 \\ 0 & -8 & -8 & 0 & 0 \\ 0 & 0 & -8 & -8 & 0 \\ 0 & 0 & 0 & -8 & -8 \end{bmatrix}$$

$$M_{21} = \begin{bmatrix} (\phi_{\frac{1}{2}},\phi_1) & 0 & 0 & 0 \\ (\phi_{\frac{3}{2}},\phi_1) & (\phi_{\frac{3}{2}},\phi_2) & 0 & 0 \\ 0 & (\phi_{\frac{5}{2}},\phi_2) & (\phi_{\frac{5}{2}},\phi_3) & 0 \\ 0 & 0 & (\phi_{\frac{7}{2}},\phi_3) & (\phi_{\frac{7}{2}},\phi_4) \\ 0 & 0 & 0 & (\phi_{\frac{9}{2}},\phi_4) \end{bmatrix} = \frac{1}{3h}\begin{bmatrix} -8 & 0 & 0 & 0 \\ -8 & -8 & 0 & 0 \\ 0 & -8 & -8 & 0 \\ 0 & 0 & -8 & -8 \\ 0 & 0 & 0 & -8 \end{bmatrix}$$

$$D_{22} = \begin{bmatrix} (\phi_{\frac{1}{2}},\phi_{\frac{1}{2}}) & 0 & 0 & 0 & 0 \\ 0 & (\phi_{\frac{3}{2}},\phi_{\frac{3}{2}}) & 0 & 0 & 0 \\ 0 & 0 & (\phi_{\frac{5}{2}},\phi_{\frac{5}{2}}) & 0 & 0 \\ 0 & 0 & 0 & (\phi_{\frac{7}{2}},\phi_{\frac{7}{2}}) & 0 \\ 0 & 0 & 0 & 0 & (\phi_{\frac{9}{2}},\phi_{\frac{9}{2}}) \end{bmatrix} = \frac{1}{3h}\begin{bmatrix} 16 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 16 \end{bmatrix}$$

From this we can easily see that M_{11} is tridiagonal, $M_{21} = M_{12}^T$, and D_{22} is a diagonal matrix.

(c) $\tilde{\mathbf{u}}$ may be eliminated from the system as follows

$$\begin{aligned} &M_{21}\mathbf{u} + D_{22}\tilde{\mathbf{u}} = \mathbf{b}_2 \\ &\Rightarrow \tilde{\mathbf{u}} = D_{22}^{-1}(\mathbf{b}_2 - M_{21}\mathbf{u}) \end{aligned}$$

which may be substituted into 3 to

$$\begin{aligned} &M_{11}\mathbf{u} + M_{12}D_{22}^{-1}(\mathbf{b}_2 - M21\mathbf{u}) = \mathbf{b}_1 \\ &\Rightarrow (M_{11} - M_{12}D_{22}^{-1}M_{21})\mathbf{u} = \mathbf{b}_1 - M_{12}D_{22}^{-1}\mathbf{b}_2 \\ &\Rightarrow M\mathbf{u} = \mathbf{b} \end{aligned}$$

where $M = M_{11} - M_{12}D_{22}^{-1}M_{21}$ and $\mathbf{b} = \mathbf{b}_1 - M_{12}D_{22}^{-1}\mathbf{b}_2$.

(d) Using the above results, M and $\mathbf{b}$ may be computed to be

$$M = \frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

$$\mathbf{b} = \frac{5h}{6}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

(with $h = 0.2$) which gives $u = \begin{bmatrix} 0.0667 & 0.1000 & 0.1000 & 0.0667 \end{bmatrix}^T$ when solved in MATLAB. Solving the original system gives

$$u_{full} = \begin{bmatrix} 0.0667 & 0.1000 & 0.1000 & 0.0667 & 0.0383 & 0.0833 & 0.1050 & 0.0833 & 0.0383 \end{bmatrix}^T.$$

Thus, we can see that the same solution is found at the interior nodes of the quadratic finite elements. Additionally, the contributions of the basis functions defined at the midpoints of the elements are already incorporated into M and B when using static condensation to solve for the degrees of freedom at the interior nodes. Thus, this technique saves us from having to directly compute five extra degrees of freedom at the midpoints of each element. This technique will be valid so long as the midpoints basis functions are completely confined within one interval. It is this property that allowed for the formation of D_{22} which promoted the elimination of the vector $\tilde{u}$ from the original system of equations.

(e) To quickly show that M is positive definite, the eigenvalues were calculated using MATLAB and found to be $\lambda = [0.3820, 1.380, 2.6180, 3.6180]^T$. Since all of the eigenvalues of this matrix are positive and real, the matrix M is indeed positive definite.

4. Write code for linear FEM using the reference element technique. Optionally, also implement quadratic finite elements using the reference element technique.

Solution

The linear FEM with the reference element technique was coded. Setting $f = \pi^2 \sin(\pi x)$ gave the following solutions for $h = 0.5, 0.25, 0.125$.

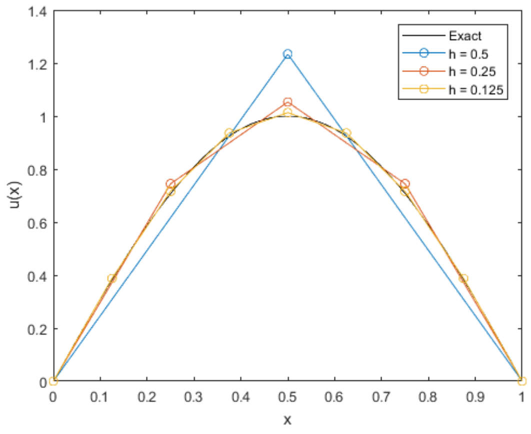


Figure 2: The linear reference element solution for $\frac{d^2u}{dx^2} = \pi^2 \sin(\pi x)$, $u(0) = u(1) = 0$.

This solution matched the solution given using the simple FEM code from Homework 1.

Finally, the quadratic FEM was implemented using the reference element technique. Figure 3 depicts the solution for $N = 2, 4, 8$ elements. The convergence of the solution for $||u-u_k||_{L^2(0,1)}$, $||u-u_k||_E$, and $\max_i |u(x_i)-u_k(x_i)|$ norms was also plotted in Figure 3. From this, we see that the energy norm exhibits second-order convergence while the L_2 and max norms exhibit fourth-order convergence.

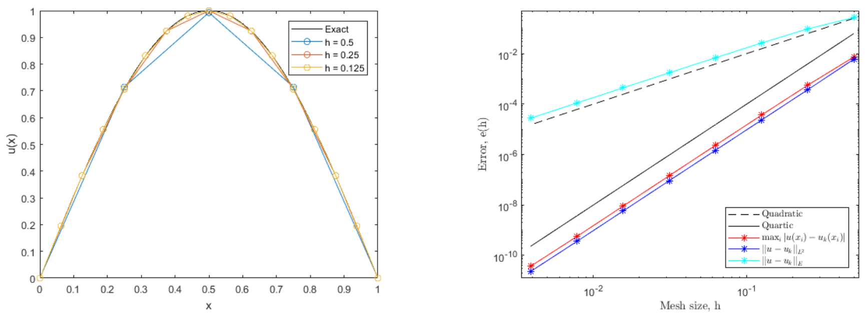


Figure 3: The quadratic reference element solution and convergence plots. The max and L_2 error exhibits fourth-order convergence while the energy norm is second-order.

5. Consider the following finite element.

$$K = [-1,1]^2, P_k = \text{span}\{1, x, y, xy\} \text{ (bilinear functions on } K),$$

$$\Sigma_K = \{l_i : P_K \to \mathbb{R}, i = 1,2,3,4 \text{ associated to the four vertices of } K\}.$$

Do the following: (i) prove or disprove that the finite element is unisolvent. (ii) In the case that it is unisolvent, compute the basis of P_K consistent with the degrees of freedom. Draw plots of the basis functions. (iii) Construct the interpolation operator on $C^0(K)$.(iv) If we move the degrees of freedom to the midpoints of edges of K, is the FEM still unisolvent?

Solution

(i) We begin by defining a general bilinear basis function

$$\psi_i(x,y) = a_i + b_i x + c_i y + d_i xy;\ \ x, y \in [-1,1]$$

From this, we define the degrees of freedom at the vertices of the square as

$$\begin{aligned} l_1(\psi_1) &= \psi_1(-1,-1) = a_1 - b_1 - c_1 + d_1 \\ l_2(\psi_2) &= \psi_2(-1,1) = a_2 - b_2 + c_2 - d_2 \\ l_3(\psi_3) &= \psi_3(1,-1) = a_3 + b_3 - c_3 - d_3 \\ l_4(\psi_4) &= \psi_4(1,1) = a_4 + b_4 + c_4 + d_4 \end{aligned}$$

this may be rewritten in matrix form as

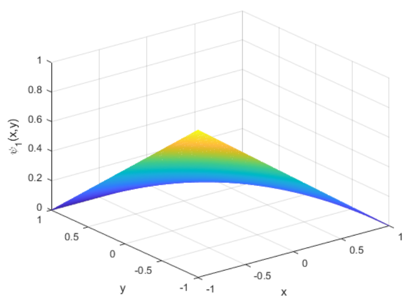
$$A = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

which has $\det(A) = -16$.

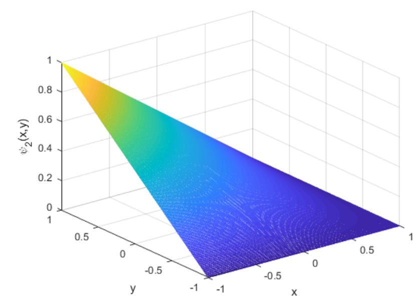
(ii) Since the determinant is nonzero, this finite element is unisolvent. From here, using the Kronecker property, the finite element can be rewritten into system form as

$$\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

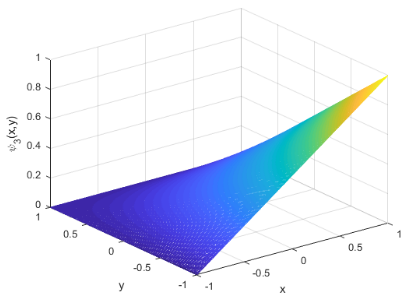
Which when solved, gives the following expressions for the four bilinear basis functions



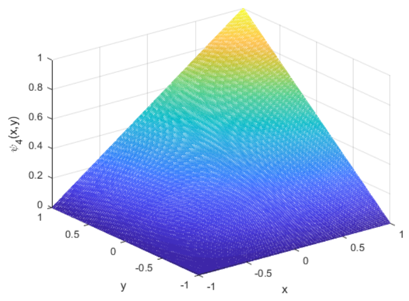
(a) $\psi_1(x, y)$



(b) $\psi_2(x, y)$



(c) $\psi_3(x, y)$



(d) $\psi_4(x, y)$

Figure 4: The four bilinear basis functions defined on the vertices of a square.

$$
\begin{aligned}
\psi_1(x, y) &= \frac{1}{4}(1 - x - y + xy) \\
\psi_2(x, y) &= \frac{1}{4}(1 - x + y - xy) \\
\psi_3(x, y) &= \frac{1}{4}(1 + x - y - xy) \\
\psi_4(x, y) &= \frac{1}{4}(1 + x + y + xy)
\end{aligned}
$$

which are plotted above in Figure 4.

(iii) The interpolation operator on $C^0(K)$ may be written as

$$
\begin{aligned}
\pi_K(v)(x, y) = \frac{1}{4}[&v(-1, 1)(1 - x - y + xy) \\
&+v(-1, 1)(1 - x + y - xy) \\
&+v(1, -1)(1 + x - y - xy) \\
&+v(1, 1)(1 + x + y + xy)]
\end{aligned}
$$

(iv) Moving the degrees of freedom to the midpoints of the edges of K gives the following expressions for the degrees of freedom

$$\begin{aligned}
l_1(\psi_1) &= \psi_1(-1,0) = a_1 - b_1 \\
l_2(\psi_2) &= \psi_2(1,0) = a_2 + b_2 \\
l_3(\psi_3) &= \psi_3(0,-1) = a_3 - c_3 \\
l_4(\psi_4) &= \psi_4(0,1) = a_4 + c_4
\end{aligned}$$

which may be rewritten into matrix form as

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

which has a determinant equal to zero. Thus, this finite element is not unisolvent.

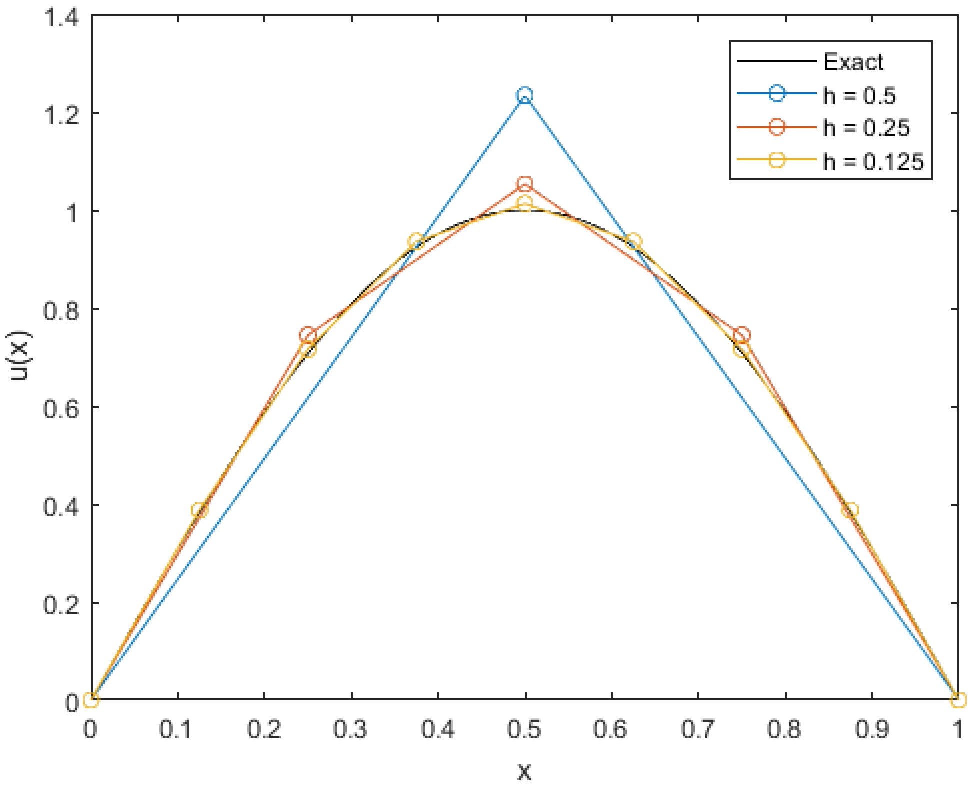
```
% MainRef.m
% Peter Ferrero, Oregon State University, MTH655, 2/17/2018
% The main file for the linear 1D reference element method to solve
 Problem 4 of Homework 2 for
% MTH 655.

clear all

n = [2,4,8];
N = length(n);
x = [0:0.001:1]';
ExactSol = Exact(x);

figure(1)
plot(x, ExactSol, 'k')
xlabel('x')
ylabel('u(x)')
hold on

for i = 1:N

    [FemSol, x] = SimpleFEM1DRef(n(i));
    ExactSol = Exact(x');
    plot(x,FemSol,'-o')

    dExactSol = pi.*cos(pi.*x');
    errorMax(i) = norm(ExactSol-FemSol, inf);
    errorL2(i) = sqrt(Simpson13Approx(n(i),x,(ExactSol-FemSol).^2));
    fL2(i) = sqrt(Simpson13Approx(n(i),x,Loadf(x).^2));

    for j = 2:length(FemSol)

        dError(j-1) = ((FemSol(j) - FemSol(j-1))/(x(j) - x(j-1))) -
 dExactSol(j-1);

    end

    errorE(i) = sqrt(Simpson13Approx(n(i),x(2:end),dError.^2));

end

a = 0; % left endpoint
b = 1; % right endpoint
h = (b-a)./n; % uniform mesh size

legend('Exact', 'h = 0.5', 'h = 0.25', 'h = 0.125')
hold off

figure(2)
loglog(h,h,'k--',h,h.^2,'k-',h,errorMax,'*-r',h,errorL2,'*-
b',h,errorE,'*-c')
xlabel('Mesh size, h', 'Interpreter', 'latex')
```

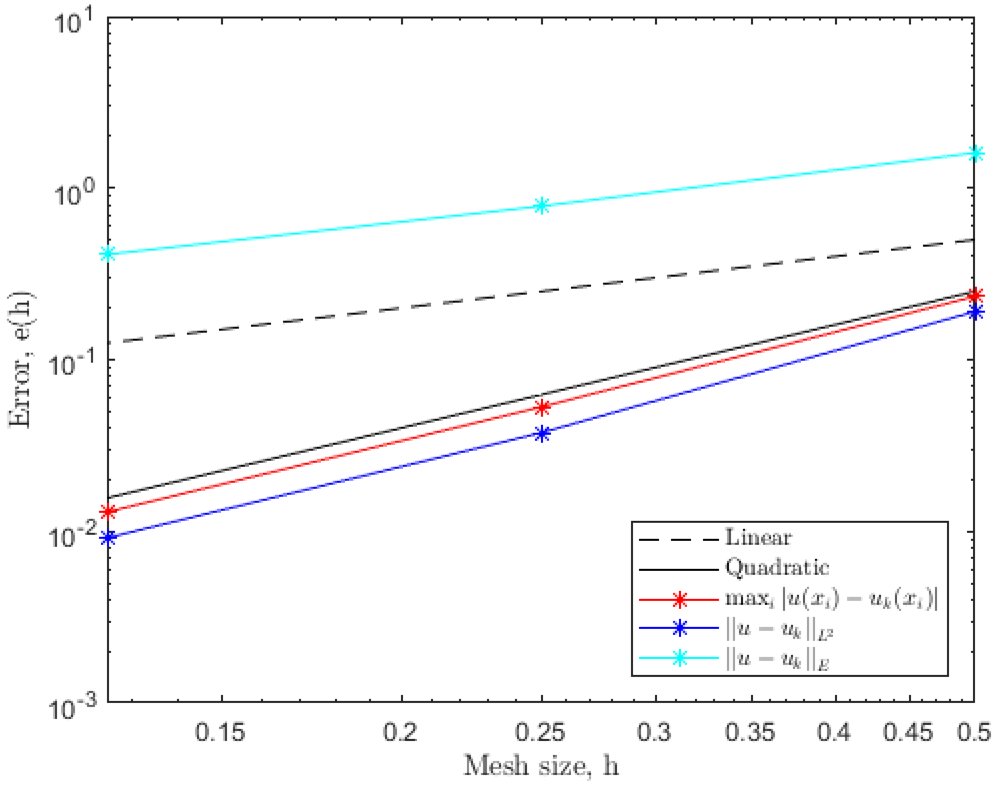
```
ylabel('Error, e(h)', 'Interpreter', 'latex')
legend({'Linear', 'Quadratic', '$\max_i |u(x_i)-u_k(x_i)|$', '$
\left \| u - u_k \right \|_{L^2}$', '$\left \| u - u_k \right \|_E
$'}, 'Interpreter', 'latex')
legend('Location', 'southeast')

% SimpleFEM1DRef.m
% Peter Ferrero, Oregon State University, MTH655, 1/31/2018
% A simple 1D reference element method to solve Problem 4 of Homework
 2 for MTH 655.

function [FemSol, x] = SimpleFEM1DRef(N)

a = 0; % left endpoint
b = 1; % right endpoint
h = (b-a)/N; % uniform mesh size
x = a:h:b; % mesh nodes

A = GStiffRef(x); % Global Stiffness matrix
F = GLoadRef(x); % Load vector

FemSol = zeros(N+1, 1); % Initialize the FEM solution
FemSol(2:N) = A(2:N,2:N)\F(2:N); % Solve the linear system
                                 % for interior nodes

end

function A = GStiffRef(x)
% ===Input: vector x of mesh nodes ===
% ===Output: Assembled stiffness matrix A ===

N = length(x) - 1; % number of elements
A = zeros(N+1, N+1); % initialize the stiffness matrix to zero

for i=1:N % loop over elements

    xi = [x(i),x(i+1)]; % element nodes
    Aloc = StiffALoc(xi); % local stiffness matrix
    n = [i i+1]; % local to global map
    A(n,n) = A(n,n) + Aloc; % Local to Global

end

end

function Aloc = StiffALoc(xi)
% ===Input xi = [x(i) x(i+1)], local nodes for element ===
% ===Output Aloc = Local stiffness matrix for element ===

Aloc = zeros(2,2); % initialize local matrix
jac = (xi(2) - xi(1))/2; % jacobian of map

[qWts, qPts] = Trapezoidal(-1,1);
```

```
for i = 1:length(qWts) % loop over quadrature points

    x = xi + (1 + qPts(1))*jac; % x on physical element = to qPts on
 ref.
    [psi, dpsi] = Reference(qPts(1)); % evaluate shape function
    Aloc = Aloc + dpsi/jac*dpsi'/jac*qWts(1)*jac;

end

end

function [psi, dpsi] = Reference(x)

psi(1,1) = 0.5 - 0.5*x;
psi(2,1) = 0.5 + 0.5*x;

dpsi(1,1) = -0.5;
dpsi(2,1) = 0.5;

end

% GLoadRef.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the 1D load vector for FEM.

function F = GLoadRef(x)

% ===Input: vector x of mesh nodes===
% ===Output: Load vector F using Trapezoidal Rule===

N = length(x)-1;
F = zeros(N+1,1);

for i = 1:N
    h = x(i+1) - x(i);
    n = [i, i+1];
    F(n) = F(n) + LoadLoc([x(i),x(i+1)]);

end

end

function Floc = LoadLoc(xi)
% ===Input xi = [x(i) x(i+1)], local nodes for element ===
% ===Output Floc = Local load vector for element ===

Floc  = zeros(2,1); % initialize local load vector
jac = (xi(2) - xi(1))/2; % jacobian of map

[qWts,qPts] = Trapezoidal(-1,1);

for i=1:length(qWts) % loop over quadrature points
```

```
    x = xi + (1+qPts(1))*jac; % x on physical element = to qPts on
 ref.
    [psi, dpsi] = Reference(qPts(i)); % evaluate shape function
    fx = feval(@Loadf,x); % Load is a user defined function for f.
    Floc = Floc + fx'.*psi*qWts(i)*jac;

end

end

% Loadf.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the local 1D load vector for FEM.

function f = Loadf(x)

% ===Input = x, mesh nodes at which to evaluate the load function f
% ===Output = f, value of load f at x

f = (pi^2)*sin(pi*x);

end

% Exact.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the exact solution for the simple FEM method.

function y = Exact(x)

% ===Input = x, mesh nodes at which to evaluate the exact solution y
% ===Output = y, the exact solution at x

y = sin(pi*x);

end

function [approx] = Simpson13Approx(n, x, Integrand)

h = (x(end)-x(1))/n; % Interval length
approx = (h/3)*(Integrand(1) + Integrand(end));

for i=2:n

    if rem(i,2) == 0

        approx = approx + 4*(h/3)*Integrand(i);

    else

        approx = approx + 2*(h/3)*Integrand(i);

    end

end
```

```
end
```





Published with MATLAB® R2017a

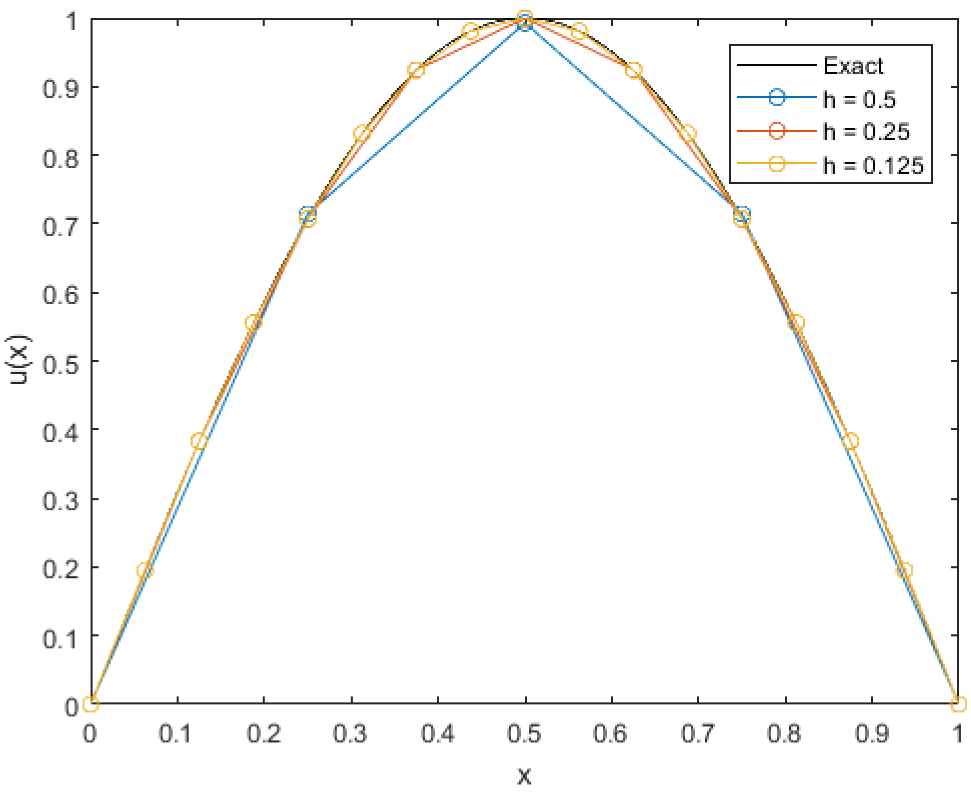
```
% MainRef.m
% Peter Ferrero, Oregon State University, MTH655, 2/17/2018
% The main file for the quadratic 1D reference element method to solve
 Problem 4 of Homework 2 for
% MTH 655.

clear all

n = [2,4,8];
N = length(n);
x = [0:0.001:1]';
ExactSol = Exact(x);

figure(1)
plot(x, ExactSol, 'k')
xlabel('x')
ylabel('u(x)')
hold on

for i = 1:N

    [FemSol, x] = SimpleFEM1DRef(n(i));
    ExactSol = Exact(x');
    plot(x,FemSol,'-o')

    dExactSol = pi.*cos(pi.*x');
    errorMax(i) = norm(ExactSol-FemSol, inf);
    errorL2(i) = sqrt(Simpson13Approx(n(i),x,(ExactSol-FemSol).^2));
    fL2(i) = sqrt(Simpson13Approx(n(i),x,Loadf(x).^2));

    for j = 3:length(FemSol)

        dError(j-1) = ((3*FemSol(j) - 4*FemSol(j-1) + FemSol(j-2))/
(2*(x(j) - x(j-1)))) - dExactSol(j);

    end

    errorE(i) = sqrt(Simpson13Approx(n(i),x(2:end),dError.^2));

end

a = 0; % left endpoint
b = 1; % right endpoint
h = (b-a)./n; % uniform mesh size

legend('Exact', 'h = 0.5', 'h = 0.25', 'h = 0.125')
hold off

figure(2)
loglog(h,h.^2,'k--',h,h.^4,'k-',h,errorMax,'*-r',h,errorL2,'*-
b',h,errorE,'*-c')
xlabel('Mesh size, h', 'Interpreter', 'latex')
```

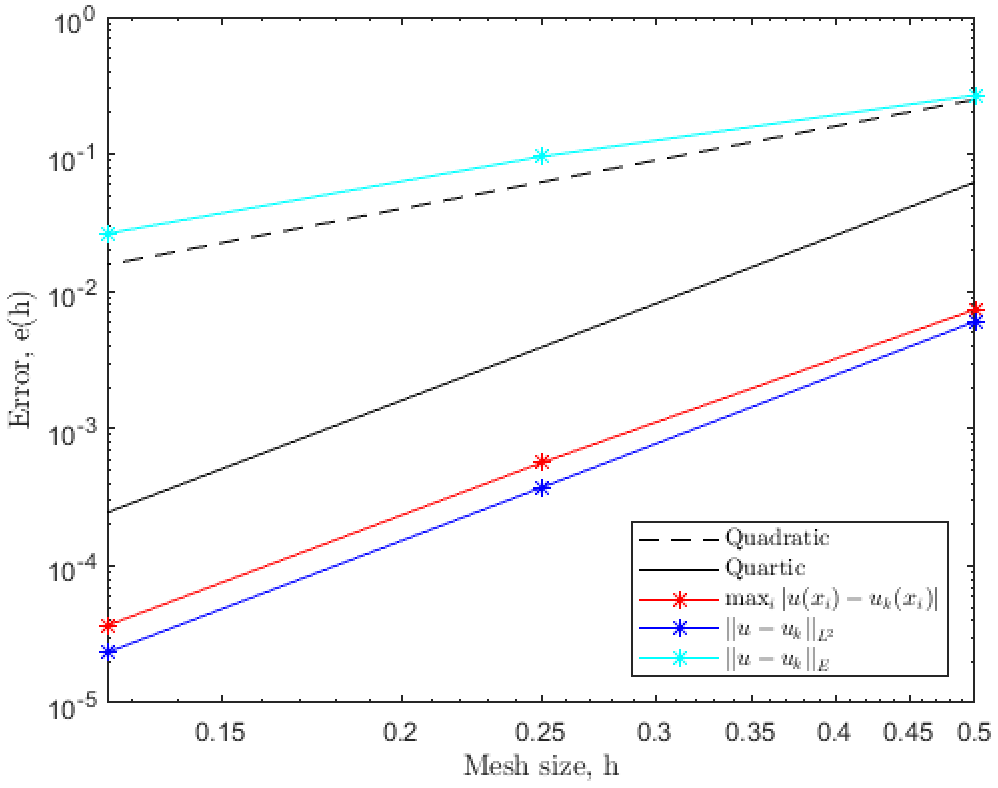
```
ylabel('Error, e(h)', 'Interpreter', 'latex')
legend({'Quadratic', 'Quartic', '$\max_i |u(x_i)-u_k(x_i)|$', '$
\left \| u - u_k \right \|_{L^2}$', '$\left \| u - u_k \right \|_E
$'}, 'Interpreter', 'latex')
legend('Location', 'southeast')

% SimpleFEM1DRef.m
% Peter Ferrero, Oregon State University, MTH655, 1/31/2018
% A simple 1D reference element method to solve Problem 4 of Homework
 2 for MTH 655.

function [FemSol, x] = SimpleFEM1DRef(N)

a = 0; % left endpoint
b = 1; % right endpoint
h = (b-a)/(2*N); % uniform mesh size
x = a:h:b; % mesh nodes

A = GStiffRef(x); % Global Stiffness matrix
F = GLoadRef(x); % Load vector

FemSol = zeros(2*N+1, 1); % Initialize the FEM solution
FemSol(2:end-1) = A(2:end-1,2:end-1)\F(2:end-1); % Solve the linear
 system
                                 % for interior nodes

end

function A = GStiffRef(x)
% ===Input: vector x of mesh nodes ===
% ===Output: Assembled stiffness matrix A ===

N = (length(x) - 1)/2; % number of elements
A = zeros(2*N+1, 2*N+1); % initialize the stiffness matrix to zero

for i=1:N % loop over elements

    xi = [x(2*i-1),x(2*i),x(2*i+1)]; % element nodes
    Aloc = StiffALoc(xi); % local stiffness matrix
    n = [2*i-1 2*i 2*i+1]; % local to global map
    A(n,n) = A(n,n) + Aloc; % Local to Global

end

end

function Aloc = StiffALoc(xi)
% ===Input xi = [x(i) x(i+1)], local nodes for element ===
% ===Output Aloc = Local stiffness matrix for element ===

Aloc = zeros(3,3); % initialize local matrix
jac = (xi(3) - xi(1))/2; % jacobian of map

[qWts, qPts] = Simpson(-1,1);
```

```
for i = 1:length(qWts) % loop over quadrature points

    x = xi + (1 + qPts(1))*jac; % x on physical element = to qPts on
 ref.
    [psi, dpsi] = Reference(qPts(i)); % evaluate shape function
    Aloc = Aloc + dpsi/jac*dpsi'/jac*qWts(i)*jac;

end

end

function [qWts, qPts] = Simpson(c,d)
% qPts are the endpoints of the reference element

qPts(1) = c;
qPts(2) = (c+d)/2;
qPts(3) = d;

qWts(1) = (d-c)/6;
qWts(2) = 4*((d-c)/6);
qWts(3) = (d-c)/6;

end

function [psi, dpsi] = Reference(x)

psi(1,1) = (x.*(x+1))./2;
psi(2,1) = -((x+1).*(x-1));
psi(3,1) = (x.*(x-1))./2;

dpsi(1,1) = (2*x + 1)./2;
dpsi(2,1) = -2*x;
dpsi(3,1) = (2*x - 1)./2;

end

% GLoadRef.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the 1D load vector for FEM.

function F = GLoadRef(x)

% ===Input: vector x of mesh nodes===
% ===Output: Load vector F using Trapezoidal Rule===

N = (length(x) - 1)/2;
F = zeros(2*N+1,1);

for i = 1:N
    n = [2*i-1, 2*i, 2*i+1];
    F(n) = F(n) + LoadLoc([x(2*i-1),x(2*i),x(2*i+1)]);

end
```

```
end

function Floc = LoadLoc(xi)
% ===Input xi = [x(i) x(i+1)], local nodes for element ===
% ===Output Floc = Local load vector for element ===

Floc  = zeros(3,1); % initialize local load vector
jac = (xi(3) - xi(1))/2; % jacobian of map

[qWts,qPts] = Simpson(-1,1);

for i=1:length(qWts) % loop over quadrature points

    x = xi + (1+qPts(1))*jac; % x on physical element = to qPts on
 ref.
    [psi, dpsi] = Reference(qPts(i)); % evaluate shape function
    fx = feval(@Loadf,x); % Load is a user defined function for f.
    Floc = Floc + fx'.*psi*qWts(i)*jac;

end

end

% Loadf.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the local 1D load vector for FEM.

function f = Loadf(x)

% ===Input = x, mesh nodes at which to evaluate the load function f
% ===Output = f, value of load f at x

f = (pi^2)*sin(pi*x);

end

% Exact.m
% Peter Ferrero, Oregon State University, MTH 655, 1/31/2018
% A function to compute the exact solution for the simple FEM method.

function y = Exact(x)

% ===Input = x, mesh nodes at which to evaluate the exact solution y
% ===Output = y, the exact solution at x

y = sin(pi*x);

end

function [approx] = Simpson13Approx(n, x, Integrand)

h = (x(end)-x(1))/n; % Interval length
approx = (h/3)*(Integrand(1) + Integrand(end));
```

```
for i=2:n
    if rem(i,2) == 0
        approx = approx + 4*(h/3)*Integrand(i);
    else
        approx = approx + 2*(h/3)*Integrand(i);
    end
end
end
```





Published with MATLAB® R2017a