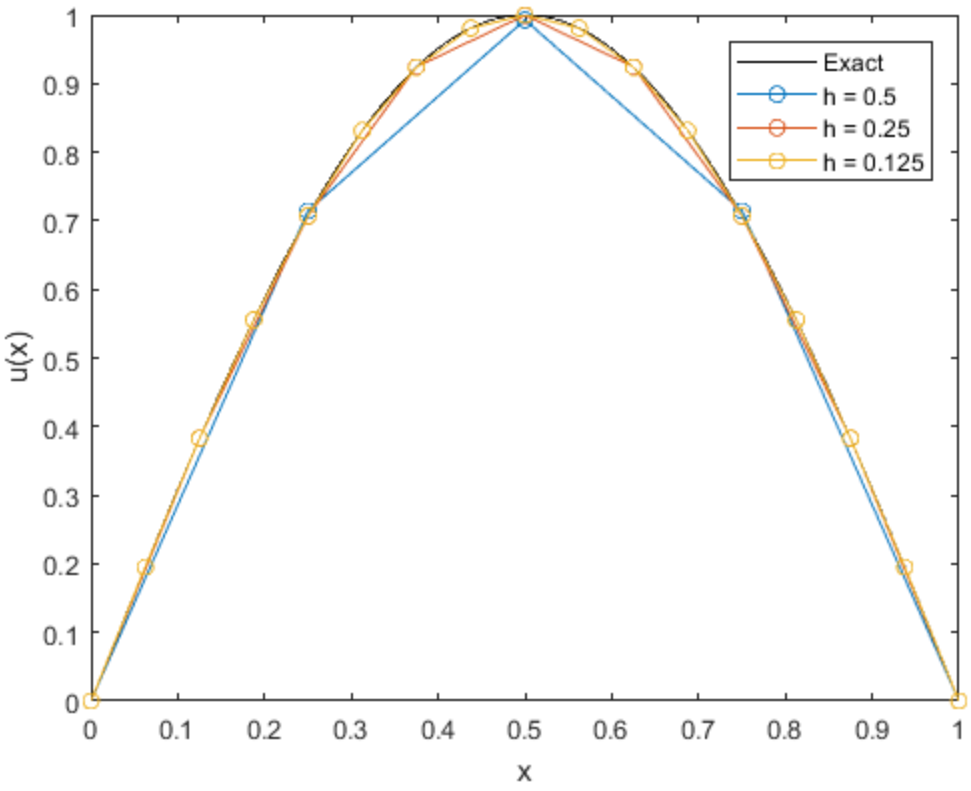
```
% MainRef.m
% Peter Ferrero, Oregon State University, MTH655, 2/17/2018
% The main file for the quadratic 1D reference element method to solve
 Problem 4 of Homework 2 for
% MTH 655.

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')
```

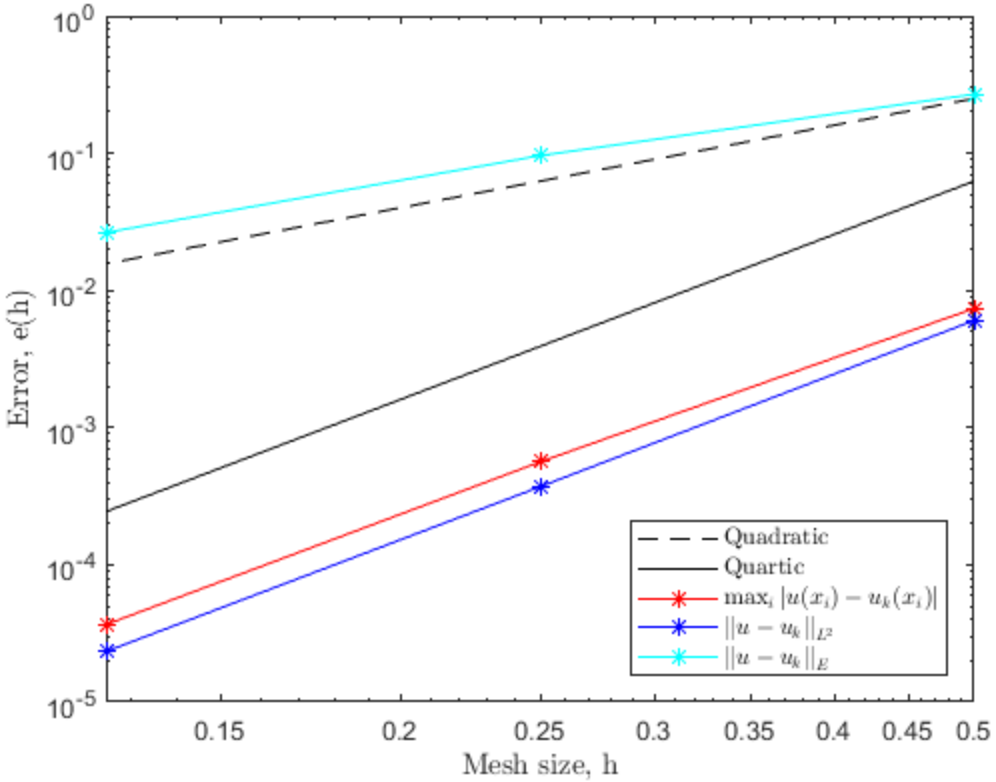
```
ylabel('Error, e(h)', 'Interpreter', 'latex')
legend({'Quadratic', 'Quartic', '$\max_i |u(x_i)-u_k(x_i)|$', '$
\left \| u - u_k \right \|_{L^2}$', '$\left \| u - u_k \right \|_E
$'}, 'Interpreter', 'latex')
legend('Location', 'southeast')

% 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