```
close all
clear all
clc

% Define the number of elements to use:
Ns = [2 4 8 16 32 64 128 256 512] ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%   Piecewise-linear elements
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% From my analytical derivation, the elemental stiffness matrix is:
Ke = @(e,h) [13*h/3-1/h+(e^2-e)*h 5/3*h+1/h-(e^2-e)*h;
             5/3*h+1/h-(e^2-e)*h  13/3*h-1/h+(e^2-e)*h] ;

% Setup figures for plotting
figure(1)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
xlabel('Position','FontSize',14)
ylabel('Displacement','FontSize',14)
title('Piecewise-Linear Basis Functions','FontSize',14)

figure(2)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
xlabel('Position','FontSize',14)
ylabel('Slope','FontSize',14)
title('Piecewise-Linear Basis Functions','FontSize',14)

errE = zeros(length(Ns)-1,1) ;
errL = zeros(length(Ns)-1,1) ;
for cntN = 1:length(Ns)
  N = Ns(cntN) ;        % Number of elements
  h = 1/N ;             % Element size
  K = zeros(N+1,N+1) ; % Setup the global stiffness matrix
  for cnt = 1:N         % The assembly procedure:
    K(cnt:cnt+1,cnt:cnt+1) = Ke(cnt,h)+K(cnt:cnt+1,cnt:cnt+1) ;
  end
  % Apply BC
  % u(0) = 0
  % u(1) = 1
  K = K(2:end,2:end) ;
  f = -1*K(1:end-1,end) ;
  K = K(1:end-1,1:end-1) ;
```

```
  alpha = K\f ;
  % The first and last alpha are prescribed by the BC
  alpha = [0; alpha; 1] ;

  % Calculate the displacements for plotting of the results
  x = linspace(0,1,10001) ;  % Setup a horizontal axis
  y = zeros(length(x),1) ;   % Initialize the displacement vector
  v = zeros(length(x),1) ;   % Initialize the slope vector
  % Next, handle the boundaries.  I'm assuming that alpha is set up such
  % that alpha(1) = 0, the x=0 boundary condition, and that
  % alpha(N+1) = the right boundary condition.
  y(1) = alpha(1) ;
  for cntr = 2:length(x)-1
    idx = ceil(x(cntr)/h) ; % Determine the element that x(cntr) is in
    y(cntr) = alpha(idx)*(1-(x(cntr)-(idx-1)*h)/h)+ ...
              alpha(idx+1)*(x(cntr)-(idx-1)*h)/h ;
    v(cntr) = alpha(idx)*(-1/h)+alpha(idx+1)*1/h ;
  end
  v(1) = v(2) ; % We can get away with this due to it being a
                % piecewise-linear set of basis functions
  v(end) = v(end-1) ;
  y(end) = alpha(N+1) ;

  % Plot the displacements
  figure(1)
  hold all
  plot(x,y)
  % Plot the slopes
  figure(2)
  hold all
  plot(x,v)
  if cntN > 1
    errL(cntN-1) = sqrt(sum((y-yold).^2)*x(2)) ; % Riemann definition of an
                                                  % integral.
    % For the energy norm of the error, either the slope calculated above,
    % (v), or a differentiation of the displacemet (diff(y)), can be used.
    errE(cntN-1) = sqrt(1/2*(sum((1-(x(1:end-1)+x(2)/2).^2).* ...
           ((diff(y)-diff(yold))./x(2)).^2')+12*sum((y-yold).^2))*x(2)) ;
  end
  yold = y ;
end
% Print error results
fprintf('\nFor piecewise-linear basis functions,\n')
NS = zeros(length(Ns)-1,1) ;
fprintf('1/h      L2 norm    Energy norm\n')
fprintf('________________________________\n')
for cntr = 1:length(Ns)-1
  NS(cntr) = Ns(cntr+1) ;
  fprintf('%3d %12.7f %14.7f',NS(cntr),errL(cntr),errE(cntr)) ;
  fprintf('\n')
end
```

```
% Plot the errors
figure(3)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
plot(NS,errL,'b')
hold on
plot(NS,errE,'r')
set(gca,'XScale','log','YScale','log')
xlabel('Log (1/h)','FontSize',14)
ylabel('Log(Error)','FontSize',14)
title('Piecewise-Linear Basis Functions','FontSize',14)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%   Piecewise-quadratic elements
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Use Matlab to determine the elemental stiffness matrix for the
% piecewise-quadratic basis functions:

% syms xi e h
% % x_i = (e-1)*h
% psi1 = 2/h^2*(xi-h/2)*(xi-h);
% psi2 = 4/h^2*xi*(xi-h);
% psi3 = 2/h^2*xi*(xi-h/2);
% dpsi1 = 4/h^2*xi-3/h;
% dpsi2 = 8/h^2*xi-4/h;
% dpsi3 = 4/h^2*xi-1/h;
% K11 = int(((xi+(e-1)*h)^2-1)*dpsi1*dpsi1+12*psi1*psi1,xi,0,h);
% K12 = int(((xi+(e-1)*h)^2-1)*dpsi1*dpsi2+12*psi1*psi2,xi,0,h);
% K13 = int(((xi+(e-1)*h)^2-1)*dpsi1*dpsi3+12*psi1*psi3,xi,0,h);
% K22 = int(((xi+(e-1)*h)^2-1)*dpsi2*dpsi2+12*psi2*psi2,xi,0,h);
% K23 = int(((xi+(e-1)*h)^2-1)*dpsi2*dpsi3+12*psi2*psi3,xi,0,h);
% K33 = int(((xi+(e-1)*h)^2-1)*dpsi3*dpsi3+12*psi3*psi3,xi,0,h);
% K = [K11 K12 K13; K12 K22 K23; K13 K23 K33] ;

% The above K yields the local stiffness matrix:
Ke = @(e,h) [h*((7*e^2)/3-(11*e)/3+47/15)-7/(3*h), ...
                          h*((8*e^2)/3-4*e+14/15)-8/(3*h), ...
                                          -h*(-e^2/3+e/3+1/5)-1/(3*h);
             h*((8*e^2)/3-4*e+14/15)-8/(3*h), ...
                          h*((16*e^2)/3-(16*e)/3+128/15)-16/(3*h), ...
                                  -h*(-(8*e^2)/3+(4*e)/3+2/5)-8/(3*h);
             -h*(-e^2/3+e/3+1/5)-1/(3*h), ...
                          -h*(-(8*e^2)/3+(4*e)/3+2/5)-8/(3*h), ...
                                          h*((7*e^2)/3-e+9/5)-7/(3*h)] ;
```

```
% Setup figures for plotting
figure(4)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
xlabel('Position','FontSize',14)
ylabel('Displacement','FontSize',14)
title('Piecewise-Quadratic Basis Functions','FontSize',14)

figure(5)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
xlabel('Position','FontSize',14)
ylabel('Slope','FontSize',14)
title('Piecewise-Quadratic Basis Functions','FontSize',14)

errE = zeros(length(Ns)-1,1) ;
errL = zeros(length(Ns)-1,1) ;
for cntN = 1:length(Ns)
  N = Ns(cntN) ;
  h = 1/N ;
  K = zeros(2*N+1,2*N+1) ;
  % Other than the local stiffness matrix, its assembly into the global
  % matrix, and calculating the displacement y, the two sections of code
  % are virtually the same
  for cnt = 1:N
    K(2*(cnt-1)+1:2*(cnt-1)+3,2*(cnt-1)+1:2*(cnt-1)+3) = ...
             Ke(cnt,h)+K(2*(cnt-1)+1:2*(cnt-1)+3,2*(cnt-1)+1:2*(cnt-1)+3) ;
  end
  % Apply BC
  % u(0) = 0
  % u(1) = 1
  K = K(2:end,2:end) ;
  f = -1*K(1:end-1,end) ;
  K = K(1:end-1,1:end-1) ;
  alpha = K\f ;
  % The first and last alpha are prescribed by the BC
  alpha = [0; alpha; 1] ;

  % Plot results
  x = linspace(0,1,10001) ;
  y = zeros(length(x),1) ;
  % Again, handle the boundaries first.  I'm assuming that alpha is set up
  % such that alpha(1) = 0, the x=0 boundary condition, and that
  % alpha(N+1) = the right boundary condition.
  y(1) = alpha(1) ;
  for cntr = 2:length(x)-1
```

```
    idx = ceil(x(cntr)/h) ; % Determine the element that x(cntr) is in
    % As a reminder to myself:
    % psi1 = 2/h^2*(xi-h/2)*(xi-h);
    % psi2 = 4/h^2*xi*(xi-h);
    % psi3 = 2/h^2*xi*(xi-h/2);
   y(cntr) = alpha(2*(idx-1)+1)*2/h^2*(x(cntr)-(idx-1)*h-h/2) ...
                                          *(x(cntr)-(idx-1)*h-h) ...
    +alpha(2*(idx-1)+2)*4/h^2*(x(cntr)-(idx-1)*h)*(x(cntr)-(idx-1)*h-h) ...
    +alpha(2*(idx-1)+3)*2/h^2*(x(cntr)-(idx-1)*h)*(x(cntr)-(idx-1)*h-h/2) ;
  end
  y(end) = alpha(end) ;

  % Plot the displacements
  figure(4)
  hold all
  plot(x,y)
  % Plot the slopes
  figure(5)
  hold all
  % The lazy but effective way of calculating slopes:
  v = diff(y)/x(2) ;
  plot(x(1:end-1)+x(2)/2,v)
  % Calculate error norms
  if cntN > 1
    errL(cntN-1) = sqrt(sum((y-yold).^2)*x(2)) ;
    errE(cntN-1) = sqrt(1/2*(sum((1-(x(1:end-1)+x(2)/2).^2).* ...
             ((diff(y)-diff(yold))./x(2)).^2')+12*sum((y-yold).^2))*x(2)) ;
  end
  yold = y ;
end

% Print error results
fprintf('\nFor piecewise-quadratic basis functions,\n')
NS = zeros(length(Ns)-1,1) ;
fprintf('1/h      L2 norm    Energy norm\n')
fprintf('_______________________________\n')
for cntr = 1:length(Ns)-1
  NS(cntr) = Ns(cntr+1) ;
  fprintf('%3d %12.7f %14.7f',NS(cntr),errL(cntr),errE(cntr)) ;
  fprintf('\n')
end

% Plot errors
figure(6)
clf
set(gca,'FontSize',14)
set(gcf,'color','white')
box on
plot(NS,errL,'b')
hold on
plot(NS,errE,'r')
```

```
set(gca,'XScale','log','YScale','log')
xlabel('Log (1/h)','FontSize',14)
ylabel('Log(Error)','FontSize',14)
title('Piecewise-Quadratic Basis Functions','FontSize',14)
```