```
format long
type = 'd1_CtsLin' ;

alpha = 0.0;
beta = 2.0;

[uErr, DuErr] = Driver_1d(11,alpha,beta,true,1,type);
disp("Part A 2bi")
disp("Given the solution in the test space u(x)=3x-3 and 11 nodes, uErr= "+
uErr +" and DuErr= " +DuErr)

nnds = 2.^(3:7)+1 ;
l = zeros(size(nnds,2),3);
for i=1:size(nnds,2)
    h = (beta - alpha) / (nnds(i) - 1) ;
    %Driver_1d(nnds,alpha,beta, uTrue_Flag,num)
    [uErr, DuErr] = Driver_1d(nnds(i),alpha,beta,true,3,type);
    l(i,:)=[DuErr, h, 0];
end
for i=1:size(nnds, 2)-1
    l(i+1,3) = log(l(i,1)/l(i+1,1))/log(l(i,2)/l(i+1,2));
end
t=table(l(:,1),l(:,2),l(:,3));
t.Properties.VariableNames=["DuErr","h","alpha DuErr"];
disp("Part A 2bii")
disp("The experimental convergence rate for x^4+1 is given in the last
column of the table")
disp(t)
disp("Thus the experimental convergence rate is about 1, which is what we
would expect since" +...
    " the error in the H_1 norm should be in the first term because we are
using continuous linear aproximations.")
disp("I believe that my program works correctly as it solves the given
problems to the error" + ...
    " that we would expect given the continuous linear approximation")

type = 'd1_CtsQuad' ;
[uErr, DuErr] = Driver_1d(11,alpha,beta,true,2,type);
disp("Part B 2bi")
disp("Given the solution in the test space u(x)=3x^2-3 and 11 nodes, uErr=
"+ uErr +" and DuErr= " +DuErr)

l = zeros(size(nnds,2),3);
for i=1:size(nnds,2)
    h = (beta - alpha) / (nnds(i) - 1) ;
    %Driver_1d(nnds,alpha,beta, uTrue_Flag,num)
    [uErr, DuErr] = Driver_1d(nnds(i),alpha,beta,true,3,type);
    l(i,:)=[DuErr, h, 0];
end
for i=1:size(nnds, 2)-1
    l(i+1,3) = log(l(i,1)/l(i+1,1))/log(l(i,2)/l(i+1,2));
end
```

```
t=table(l(:,1),l(:,2),l(:,3));
t.Properties.VariableNames=["DuErr","h","alpha DuErr"];
disp("Part B 2bii")
disp("The experimental convergence rate for x^4+1 is given in the last
column of the table")
disp(t)
disp("Thus the experimental convergence rate is about 2, which is what we
would expect since" +...
    " the error in the H_1 norm should be in the second term because we are
using continuous quadratic aproximations.")
disp("I believe that my program works correctly as it solves the given
problems to the error" + ...
    " that we would expect given the continuous quadratic approximation")
```

```
Part A 2bi
Given the solution in the test space u(x)=3x-3 and 11 nodes, uErr=
1.1139e-15 and DuErr= 2.6422e-15
Part A 2bii
The experimental convergence rate for x^4+1 is given in the last column of
the table
          DuErr              h           alpha DuErr
    ___________________    ________    ________________

     2.18746043674998         0.25                    0
     1.09502382259906        0.125     0.998294667576797
    0.547670136700825       0.0625     0.999583135965761
    0.273854733748454      0.03125     0.999896396930765
    0.136929821488749     0.015625     0.999974137905819

Thus the experimental convergence rate is about 1, which is what we would
expect since the error in the H_1 norm should be in the first term because
we are using continuous linear aproximations.
I believe that my program works correctly as it solves the given problems to
the error that we would expect given the continuous linear approximation
Part B 2bi
Given the solution in the test space u(x)=3x^2-3 and 11 nodes, uErr=
5.0925e-15 and DuErr= 1.6708e-14
Part B 2bii
The experimental convergence rate for x^4+1 is given in the last column of
the table
           DuErr                h           alpha DuErr
    ____________________     ________    _________________

      0.0912325231126511         0.25                   0
      0.0228184327995164        0.125     1.99934850778997
     0.00570523562089518       0.0625     1.99984133399132
     0.00142634786344118      0.03125     1.99996059474968
    0.000356589396762037     0.015625     1.99999016498562

Thus the experimental convergence rate is about 2, which is what we would
expect since the error in the H_1 norm should be in the second term because
we are using continuous quadratic aproximations.
I believe that my program works correctly as it solves the given problems to
the error that we would expect given the continuous quadratic approximation
```

Published with MATLAB® R2023b

```
function [localmat] = d1_ip_ten0_Dten0(isub, ...
   scal_fun, ten0a_type, ten0b_type)
%
% This function computes, for subinterval isub, the integrals
% of the (scalar) ten0a basis functions times
% the (scalar) Gradten0b basis functions multiplied by the scalar
% function scal_fun. The matrix of values is returned in localmat.
%
%


%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul
% Description of subinterval.
xleft = xpts(isub) ;
xright = xpts(isub + 1) ;
hsub = xright - xleft ;

% Evaluation of quadrature points and quadrature weights.
[quad_pts, quad_wghts] = feval(quad_rul) ;
nqpts = size(quad_pts,1) ;

% Evaluate Basis Functions and their Gradients at quad. points.
[ten0a, Gradten0a] = feval(ten0a_type, quad_pts) ;
nbas0a = size(ten0a,1) ;

[ten0b, Gradten0b] = feval(ten0b_type, quad_pts) ;
nbas0b = size(ten0b,1) ;

% Do appropriate scaling to get the true derivatives.
Gradten0b = Gradten0b / hsub ;

% Adjust points and weights to account for size of true triangle.
quad_pts = xleft + hsub* quad_pts ;
quad_wghts = hsub * quad_wghts ;

% Evaluate the scalar multiplier at the quadrature points.
sfun_vals = feval(scal_fun, quad_pts, isub) ;

% Now to do the evaluations of the integrals.
for iq = 1:nqpts
   ten0a(:,iq) = quad_wghts(iq) * sfun_vals(iq) * ten0a(:,iq) ;
end

mat1 = ten0a * Gradten0b.' ;

localmat = [ mat1 ] ;
```

Published with MATLAB® R2023b

```
function [localmat] = bfun(x_pts, isub)
%
% This function represents the b function in the
% differential equation.
%
%
%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul
%
%

nevalpts = size(x_pts,1) ;
localmat = x_pts.^2  ;
```

Published with MATLAB® R2023b

```
function [localmat] = efun(x_pts, isub)
%
% This function represents the e function in the
% differential equation.
%
%
%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul

%
%

nevalpts = size(x_pts,1) ;
localmat = 5*x_pts + 2  ;
```

Published with MATLAB® R2023b

```
function [localmat] = rhsfun(x_pts, isub, num)
%
% This function represents the rhs function in the
% differential equation.
%
%
%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul
%
%

nevalpts = size(x_pts,1) ;

if num == 1
    %3x-3
    localmat = 3*x_pts.^2+(3*x_pts-3).*(5*x_pts+2)-6;
elseif num == 2
    %3x^2-3
    localmat = 6*x_pts.^3+(3*x_pts.^2-3).*(5*x_pts+2)-6*(2*x_pts+1)-12*x_pts;
else
    %x^4+1
    localmat = 4*x_pts.^5+
(x_pts.^4+1).*(5*x_pts+2)-8*x_pts.^3-12*x_pts.^2.*(2*x_pts+1)  ;
end
```

Published with MATLAB® R2023b

```
function [localmat] = DuFun(x_pts, isub)

%
% This function computes, the current approximation for du
% at the requested x_pts points in subinterval isub.
% The vector of values is returned in localmat.
%
%


%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul

%%%%
% We firstly need to determine the corresponding points on the
% reference interval.

% Description of subinterval.
xleft = xpts(isub) ;
xright = xpts(isub + 1) ;
hsub = xright - xleft ;


% Map the points to the reference triangle.
Rx_pts = (x_pts - xleft)/ hsub ;

% Evaluate Basis Functions and their Gradients at requested points.
[basvals, Gradten1] = feval(vel_bas_type, Rx_pts) ;

% Do appropriate scaling to get the true derivatives.
Gradten1 = Gradten1 / hsub ;

% Extract from the Global solution vector the coefficient values for
% the basis functions.
if strcmp(vel_bas_type, 'd1_CtsLin') == 1
   Vel1 = Global_u([isub isub+1],1) ;

   elseif strcmp(vel_bas_type, 'd1_CtsQuad') == 1
   Vel1 = Global_u([2*isub-1 2*isub 2*isub+1],1) ;

   elseif strcmp(vel_bas_type, 'd1_CtsCub') == 1
   Vel1 = Global_u([3*isub-2 3*isub-1 3*isub 3*isub+1],1) ;

end
```

Multiply the coefficients by the basis values.

```
localmat = Vel1.' * Gradten1 ;
```

Published with MATLAB® R2023b

```
function [localmat] = uTrue(x_pts, isub, num)
%
% This function represents the diffusive function in the
% differential equation.
%
%
%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul
%
%

nevalpts = size(x_pts,1) ;

if num == 1
    %3x-3
    localmat = 3*x_pts-3 ;
elseif num == 2
    %3x^2-3
    localmat = 3*x_pts.^2-3;
else
    %x^4+1
    localmat = x_pts.^4+1 ;
end
```

Published with MATLAB® R2023b

```
function [localmat] = DuTrue(x_pts, isub, num)
%
% This function represents the diffusive function in the
% differential equation.
%
%
%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%
global xpts nnds
global Global_r  Global_s  Global_u
global rad_bas_type  str_bas_type  vel_bas_type
global quad_rul
%
%

nevalpts = size(x_pts,1) ;

if num == 1
    %3x-3
    localmat = 3*ones(1,nevalpts)'  ;
elseif num == 2
    %3x^2-3
    localmat = 6*x_pts;
else
    %x^4+1
    localmat = 4*x_pts.^3  ;
end
```

Published with MATLAB® R2023b

```
function [CQuadVal, DerivCQuadVal] = d1_CtsQuad(quad_pts)
%
% This function computes the values of the continuous
% quadratic basis functions, and of its gradient, at
% the quadrature points quad_pts --- on the reference interval (0, 1).
%

CQuadVal(1,:) = 2.0*(quad_pts-1/2).*(quad_pts-1);
CQuadVal(2,:) = -4.0*quad_pts.*(quad_pts-1);
CQuadVal(3,:) = 2.0*quad_pts.*(quad_pts-1/2);


DerivCQuadVal(1,:) = 2.0*(quad_pts-1/2)+2.0*(quad_pts-1) ;
DerivCQuadVal(2,:) = -4.0*quad_pts-4.0*(quad_pts-1)  ;
DerivCQuadVal(3,:) = 2.0*quad_pts+2.0*(quad_pts-1/2)  ;
```

Published with MATLAB® R2023b