

Due Monday, October 5th, 12:00 midnight

This homework is considering the analysis of 1DBVP as discussed in class. A list of MatLab programs is provided together with an example problem.

Problem 1 – Boundary value problems (BVPs) with non-smooth solutions – Error analysis and higher order (quadratic and cubic) elements (MatLab)

Consider the following BVP

$$-\frac{d}{dx}\left(k(x)\frac{du}{dx}\right) = f(x), 0 \leq x \leq 1$$

where $k(x) = \frac{1}{a} + a(x - x_0)^2$ and

$$f(x) = 2\{1 + a(x - x_0)[\arctan[a(x - x_0)] + \arctan(ax_0)]\}.$$

The analytical solution to this problem (check it out!) is:

$$u_{ex}(x) = (1 - x)[\arctan a(x - x_0) + \arctan(ax_0)]$$

and $\frac{du_{ex}}{dx} = -\{[\arctan a(x - x_0)] + [\arctan(ax_0)]\} + \frac{(1 - x)a}{1 + a^2(x - x_0)^2}$

The boundary conditions are $u(0) = u(1) = 0$.

- Plot the analytical solution for different values of a and $x_0 = 0.5$ to realize that the function $u_{ex}(x)$ exhibits behavior which ranges from very smooth to almost discontinuous near $x = x_0$.
- For both a 'smooth' problem (small a , e.g. $a = 5$) and a 'rough' problem (large a , e.g. $a = 50$), study the convergence rate of linear, quadratic and cubic elements. Use uniform meshes of 2, 4, 8 and 16 elements ($h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$).

Study the convergence in both the L_2 and energy norms discussed in lecture. Plot $\log E \equiv \log \|u - u^h\|$ versus $\log h$ and determine the experimental convergence rates for the different element types. Compare these with the theoretical values shown in class, e.g for the L_2 norm

$$\|E\|_{L_2} = \left(\int_0^1 (E(x))^2 dx\right)^{1/2} \leq C_1 h^{k+1},$$

C_1 being a constant independent of h , and for finite elements employing complete polynomials of degree k .

Similarly, we have the following result for the energy norm:

$$\|E\|_{energy} = \left(\frac{1}{2} \int_0^1 k(x) \left(\frac{dE}{dx} \right)^2 dx \right)^{1/2} \leq C_2 h^k,$$

C_2 being a constant independent of h , and for finite elements employing complete polynomials of degree k .

In your results, please normalize these computed errors with the corresponding norms of the exact solution, i.e. use the following:

$$\bar{e}_{L_2} = \frac{\left(\int_0^1 (u_{ex}(x) - u^h(x))^2 dx \right)^{1/2}}{\left(\int_0^1 (u_{ex}(x))^2 dx \right)^{1/2}}$$

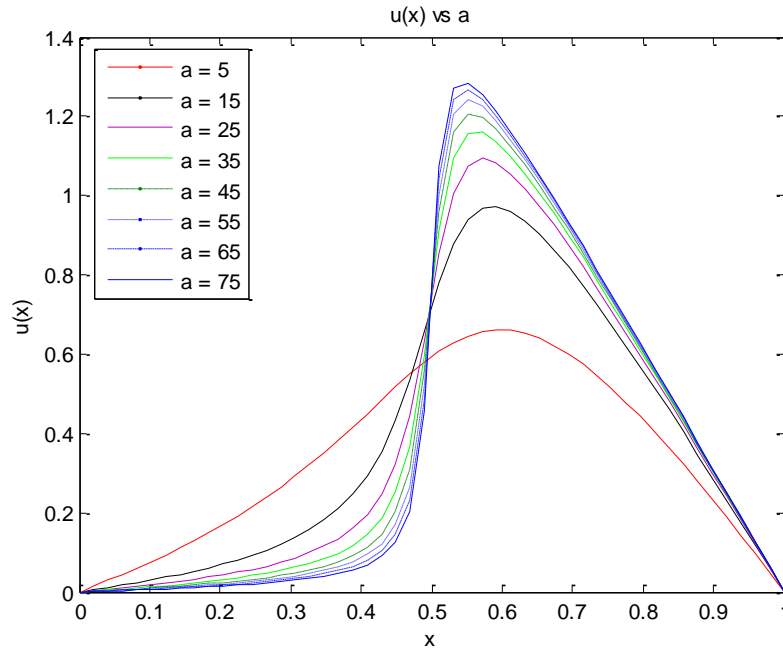
and

$$\bar{e}_{energy} = \frac{\left(\frac{1}{2} \int_0^1 k(x) \left(\frac{du_{ex}(x)}{dx} - \frac{du^h(x)}{dx} \right)^2 dx \right)^{1/2}}{\left(\frac{1}{2} \int_0^1 k(x) \left(\frac{du_{ex}(x)}{dx} \right)^2 dx \right)^{1/2}}$$

The above theoretical error estimates are called asymptotic since they exhibit increasing accuracy as $h \rightarrow 0$. How small does h in your error plots need to be to achieve asymptotic convergence?

Solution:

(a) The figure is plotted as follows:



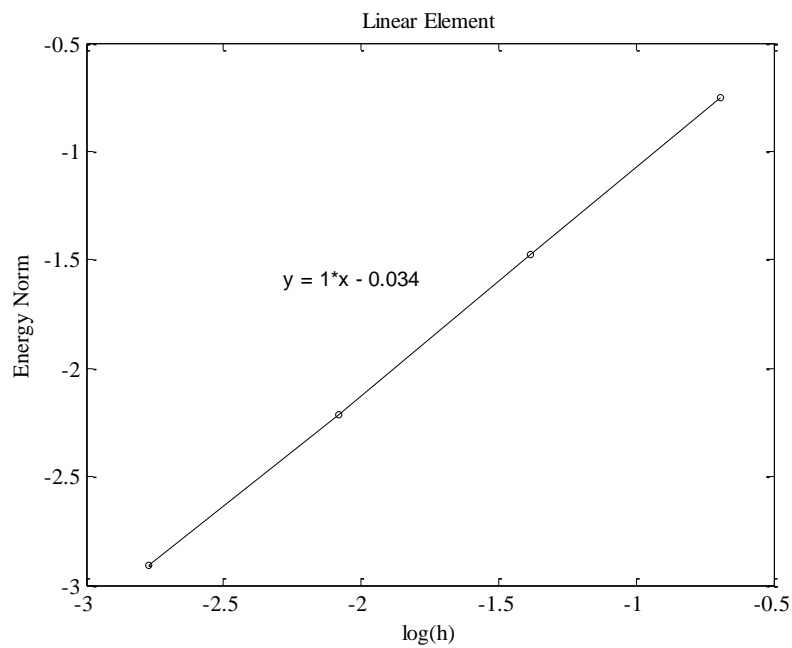
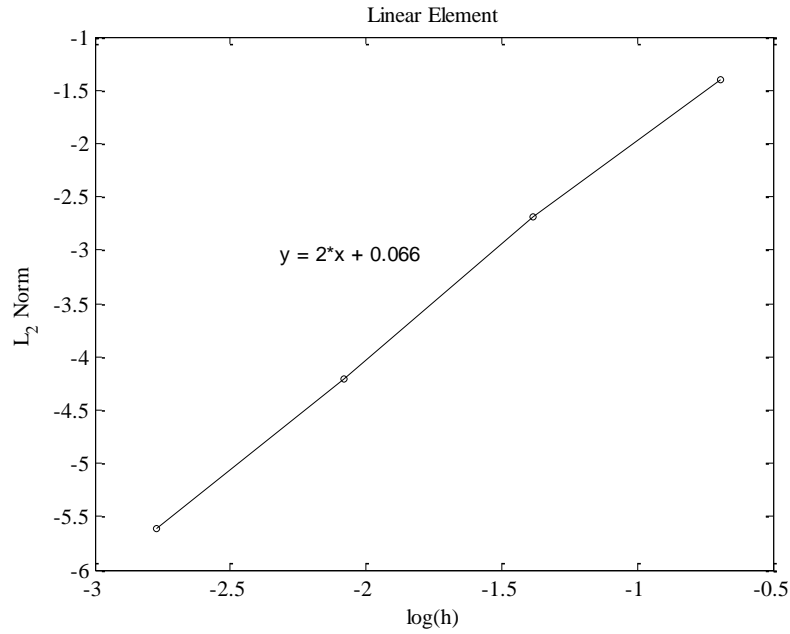
So it is obvious that the function exhibits behavior which ranges from very smooth to almost discontinuous near middle point with increase of a.

(b) Let us first tabulated the error for different parameters:

(1) **a = 5 :**

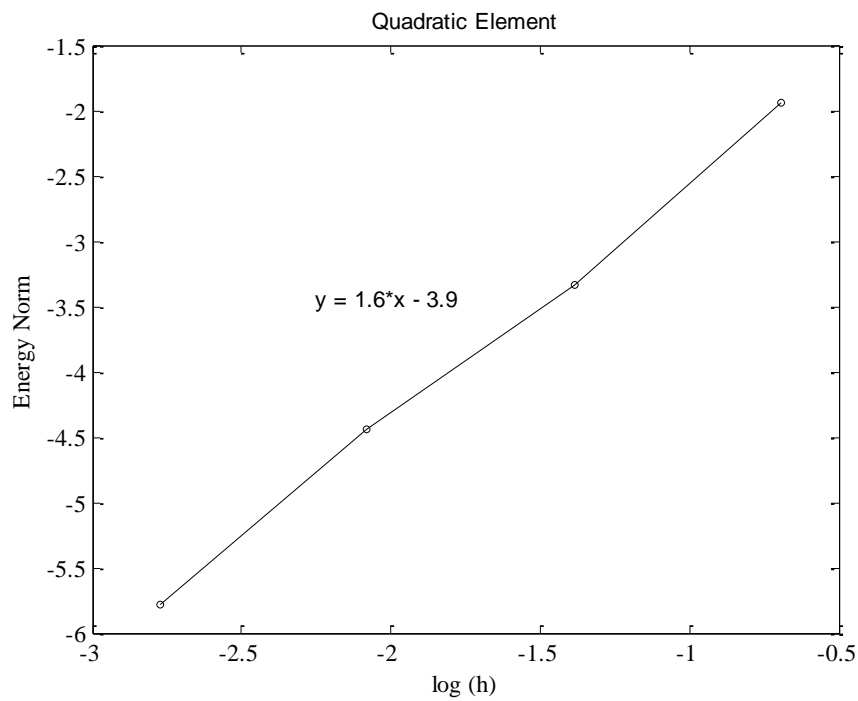
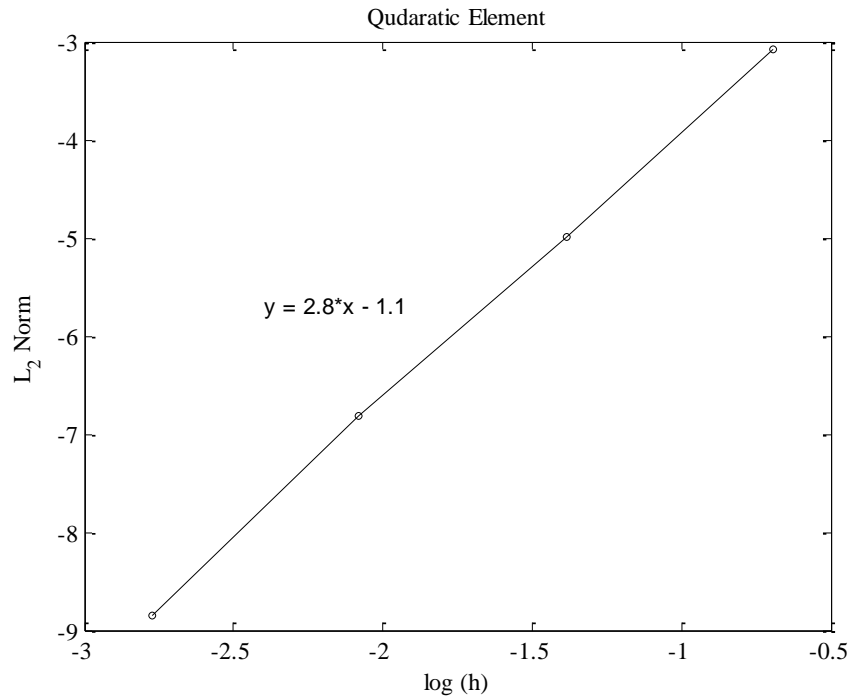
Linear elements:

h	L2 Norm	Energy Norm
1/2	2.470096e-001	4.729150e-001
1/4	6.785376e-002	2.282863e-001
1/8	1.476291e-002	1.093434e-001
1/16	3.629805e-003	5.462090e-002



Quadratic elements:

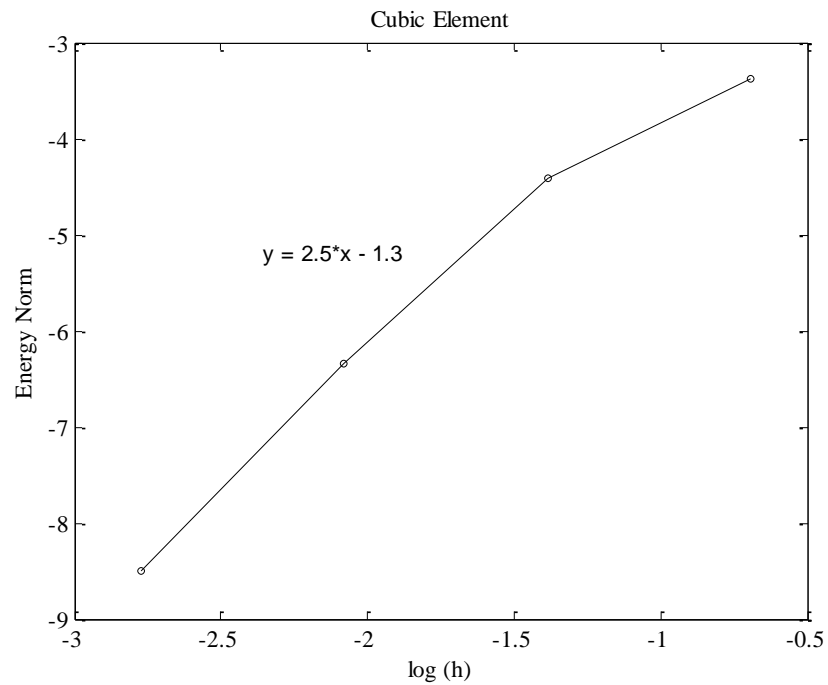
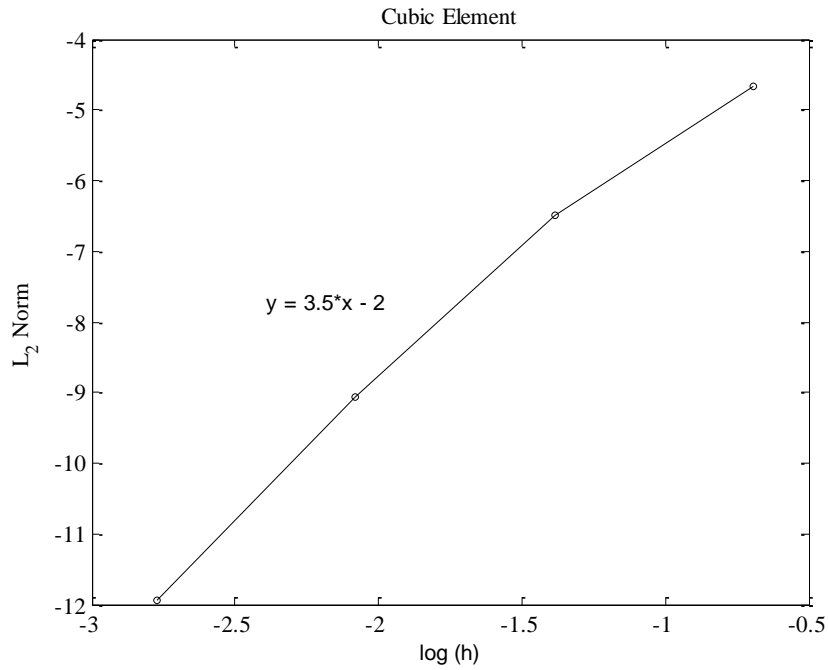
h	L2 Norm	Energy Norm
1/2	4.614893e-002	1.428571e-001
1/4	6.781632e-003	3.538466e-002
1/8	1.097329e-003	1.177542e-002
1/16	1.433696e-004	3.074287e-003



Cubic elements:

h	L2 Norm	Energy Norm
1/2	9.317924e-003	3.427679e-002
1/4	1.516161e-003	1.208630e-002
1/8	1.144852e-004	1.745773e-003

1/16	6.539205e-006	2.030547e-004
------	---------------	---------------

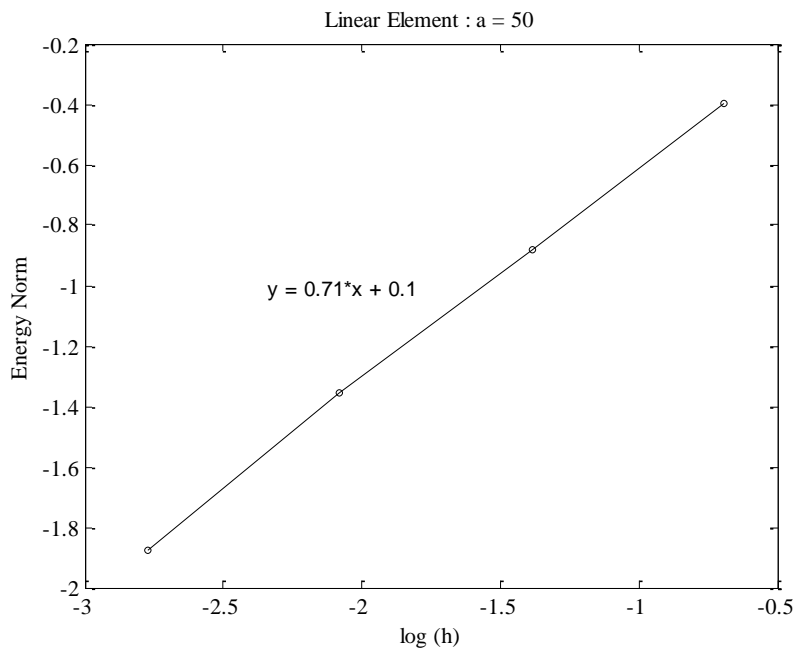
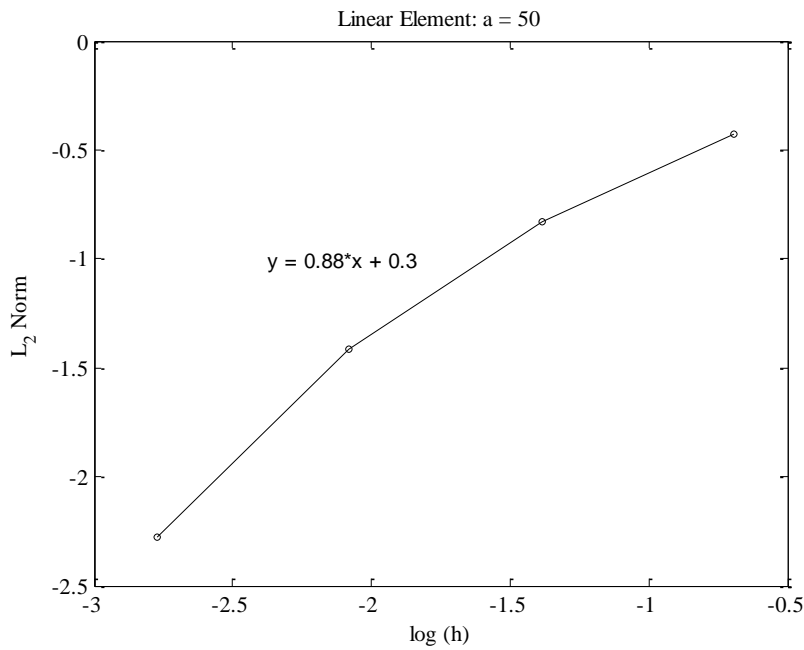


(2) a = 50 :

Linear elements:

h	L2 Norm	Energy Norm
1/2	6.529922e-001	6.722299e-001

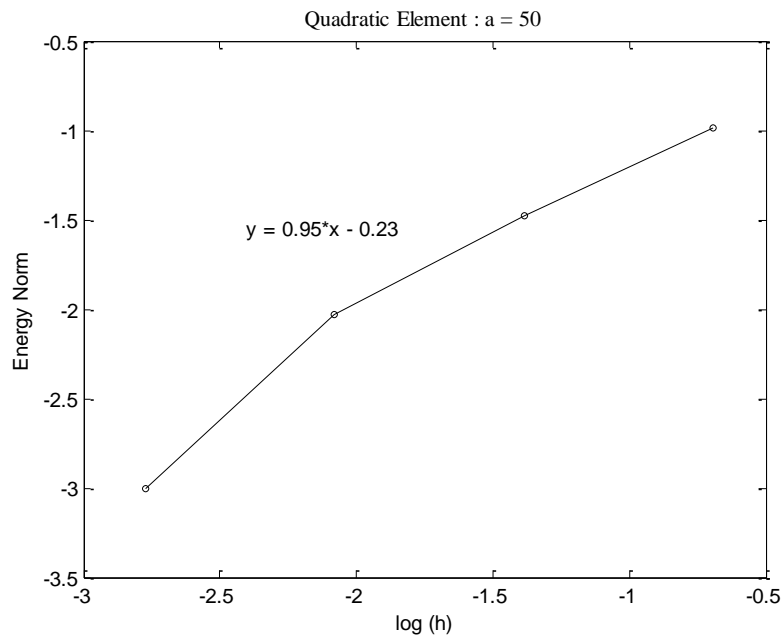
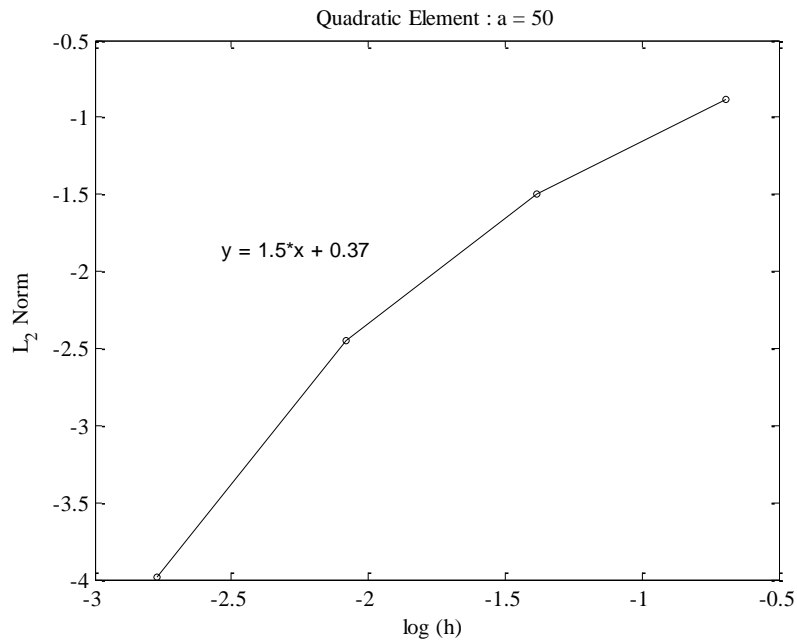
1/4	4.370131e-001	4.135555e-001
1/8	2.433105e-001	2.579198e-001
1/16	1.029235e-001	1.529029e-001



Quadratic elements:

h	L2 Norm	Energy Norm
1/2	4.148505e-001	3.719197e-001
1/4	2.241999e-001	2.281659e-001
1/8	8.638589e-002	1.307904e-001

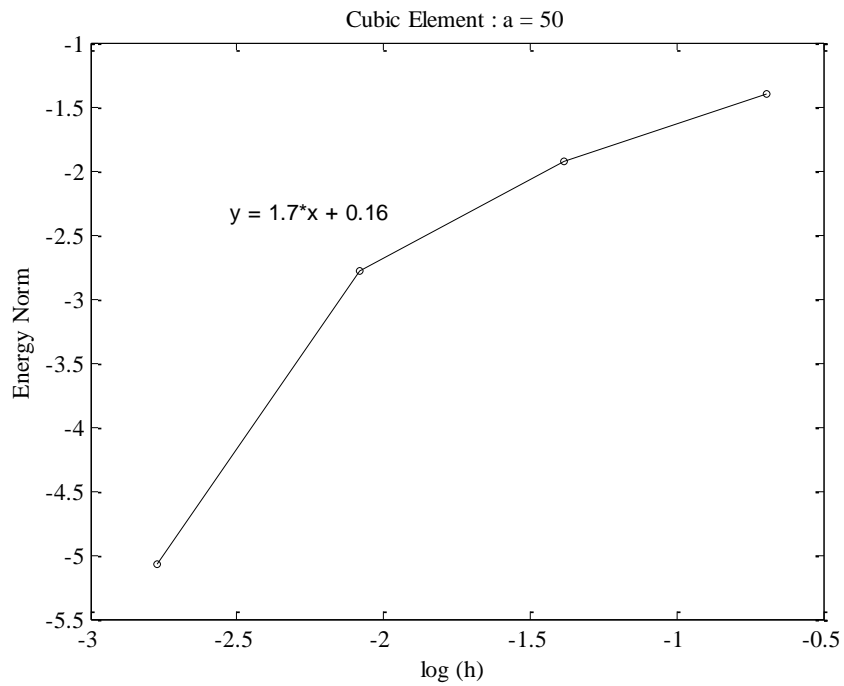
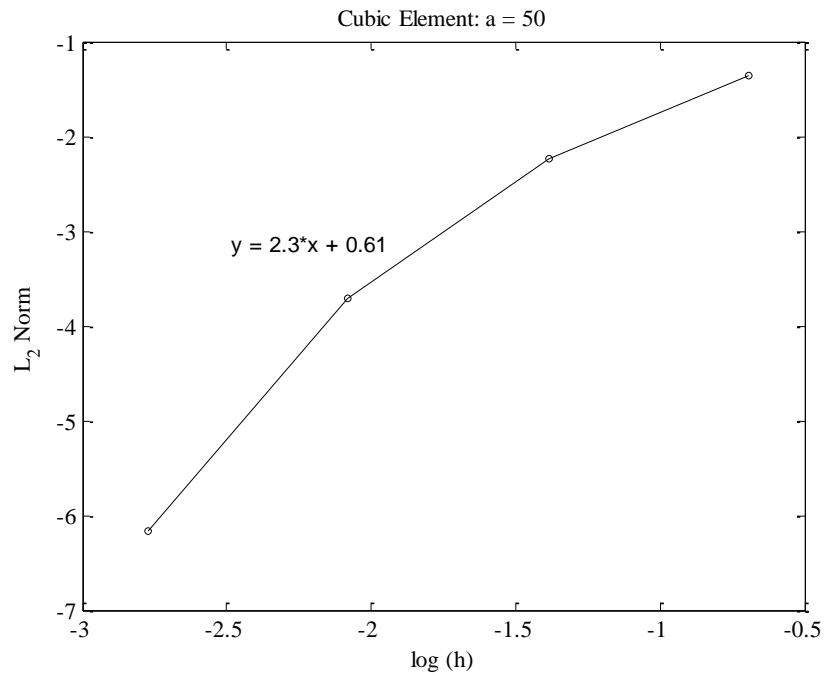
1/16	1.851009e-002	4.969547e-002
------	---------------	---------------



Cubic elements:

h	L2 Norm	Energy Norm
1/2	2.583418e-001	2.449109e-001
1/4	1.059886e-001	1.449958e-001
1/8	2.429702e-002	6.174534e-002

1/16	2.104420e-003	6.288979e-003
------	---------------	---------------



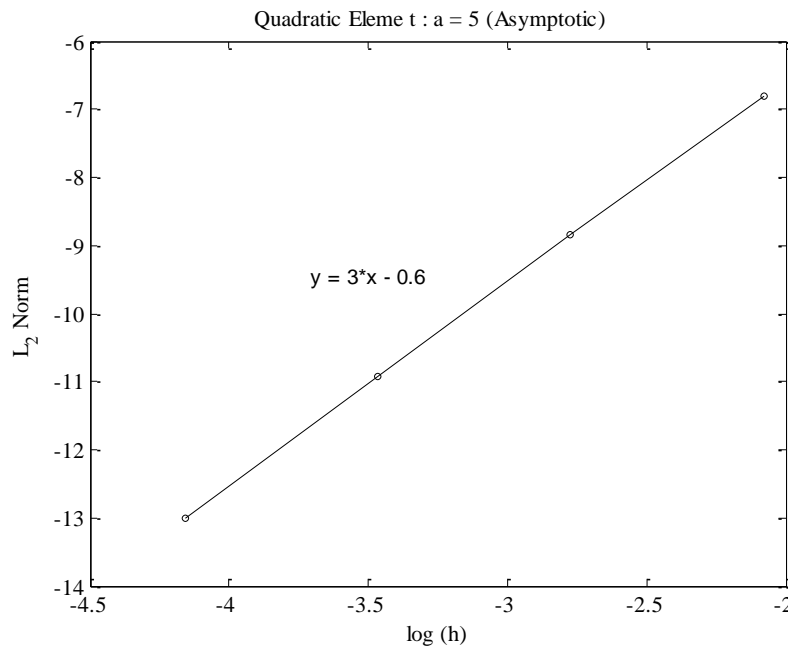
The theoretic result for L_2 norm can be expressed as

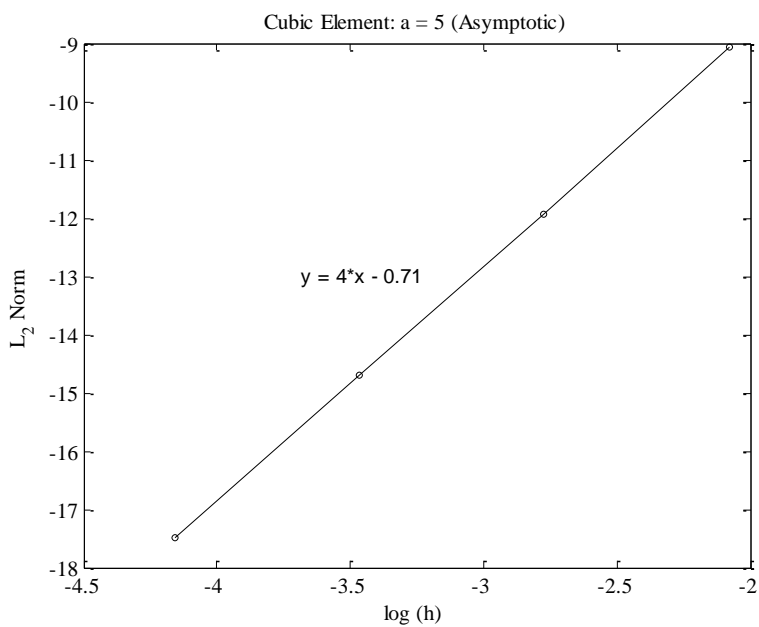
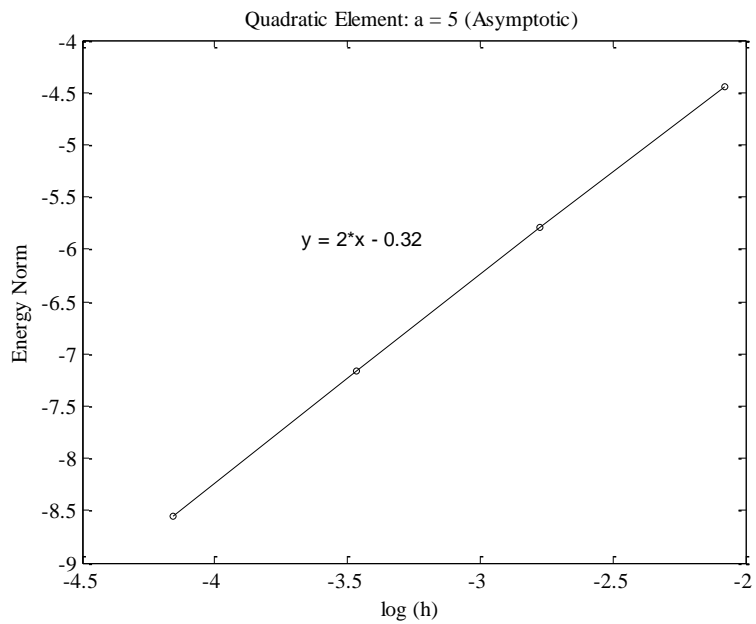
$$\log(\| E \|_{L_2}) = \log(C_1) + (k + 1)\log(h),$$

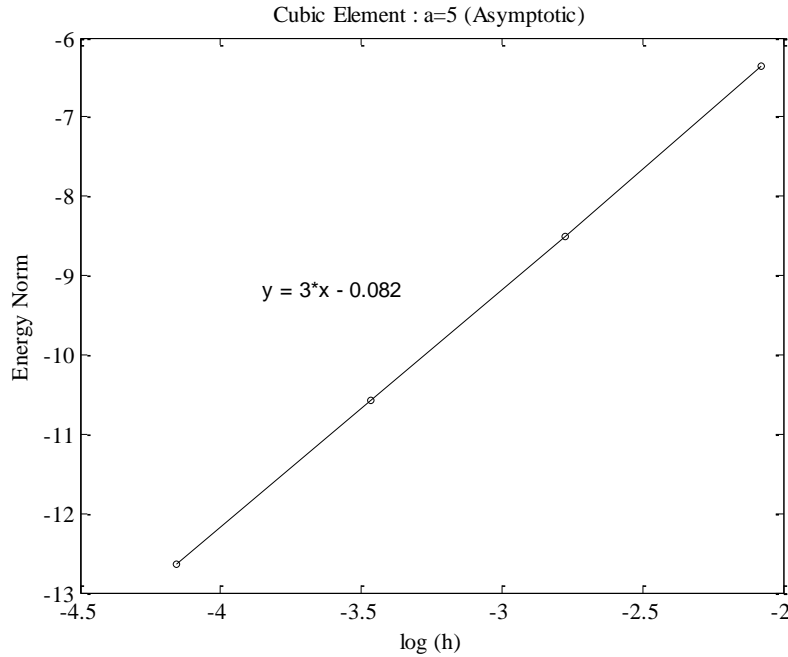
So if we plot the log error vs log h, the plot is a straight line, where the slope ($k+1$) is the rate of convergence of the element. Therefore, for linear elements, the slope is 2. The slope is 3 for quadratic elements and 4 for cubic elements if the asymptotic convergence is achieved. The energy norm can also be expressed as

$$\log(\|E\|_{energy}) = \log(C_2) + k \log(h),$$

Thus the accuracy in the derivative is one order less than the accuracy in the function. If the solution is smooth ($a=5$), the convergence rate doesn't deviate too much from the theoretic value. In particular, when using linear element, it achieves the asymptotic convergence. For higher order elements, the asymptotic convergence has not been achieved yet. More elements are needed. From the error plots, it seems h needs to be $\frac{1}{8}$ for quadratic and $\frac{1}{8}$ for cubic element.



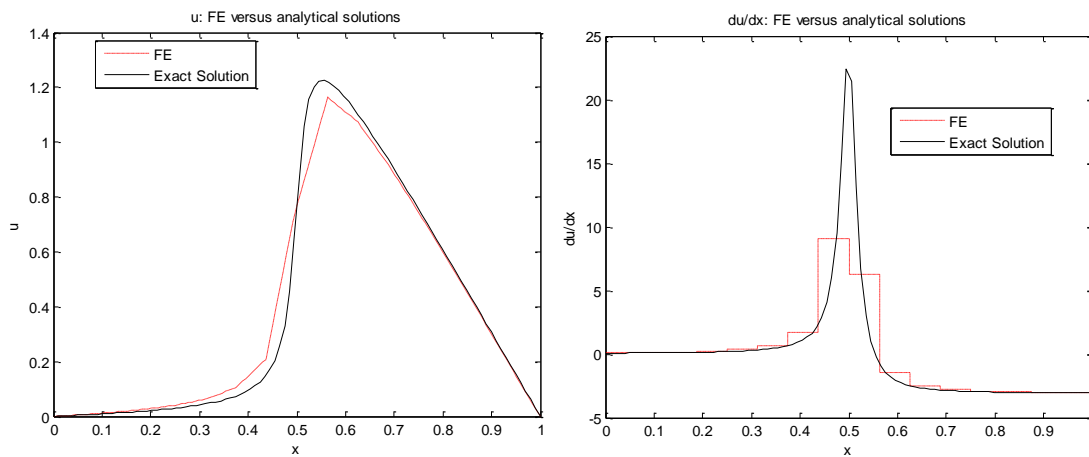




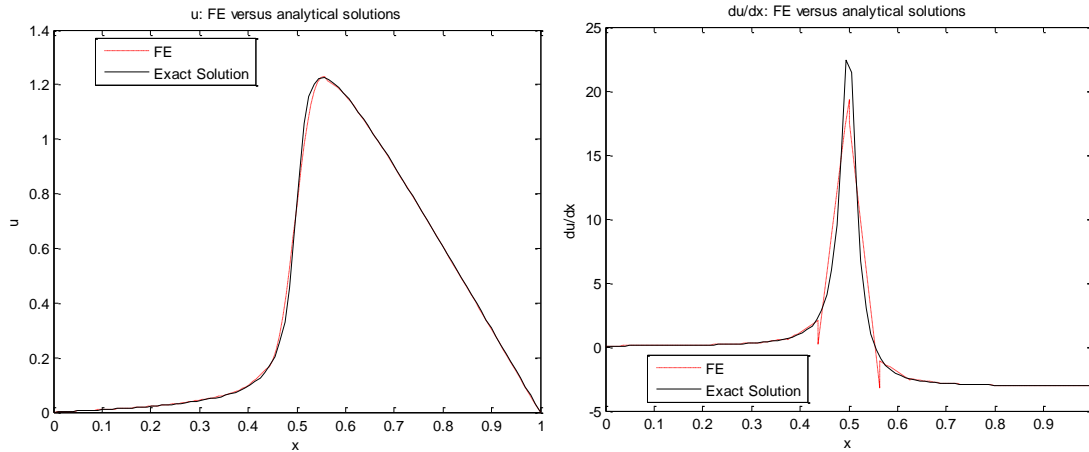
If the solution exhibits some discontinuous behavior, the experimental convergence rates are even lower than the theoretic rates. This is because near the discontinuous region, the derivative of the solution changes rapidly. Thus, more elements are needed in this region. A better idea is to use adaptive mesh where put more elements around the discontinuous region while putting less elements in the smooth region.

Let's see the plots using 16 elements when a = 50:

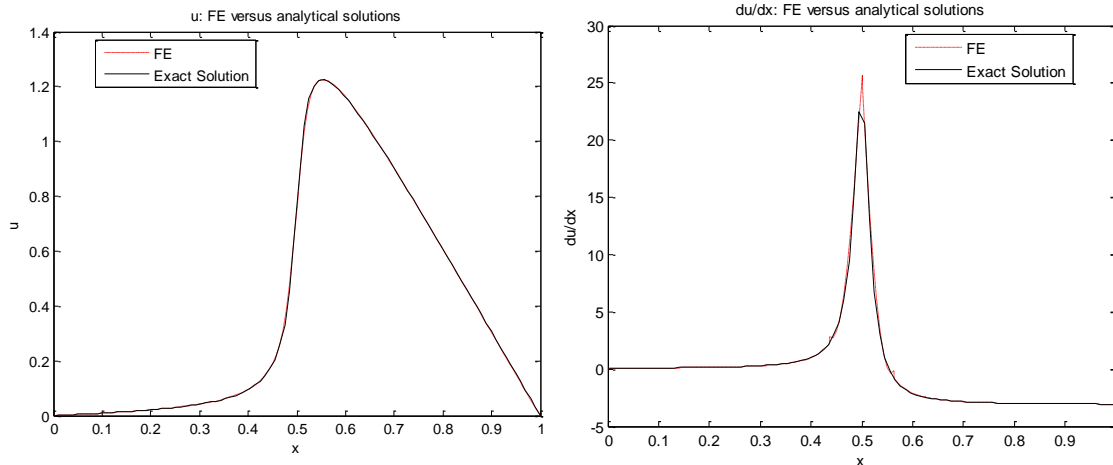
Linear elements:



Quadratic Elements:

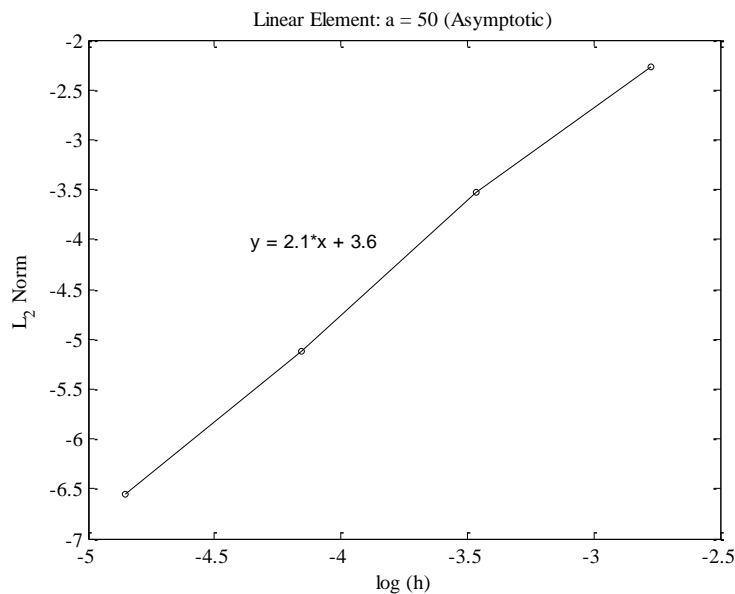


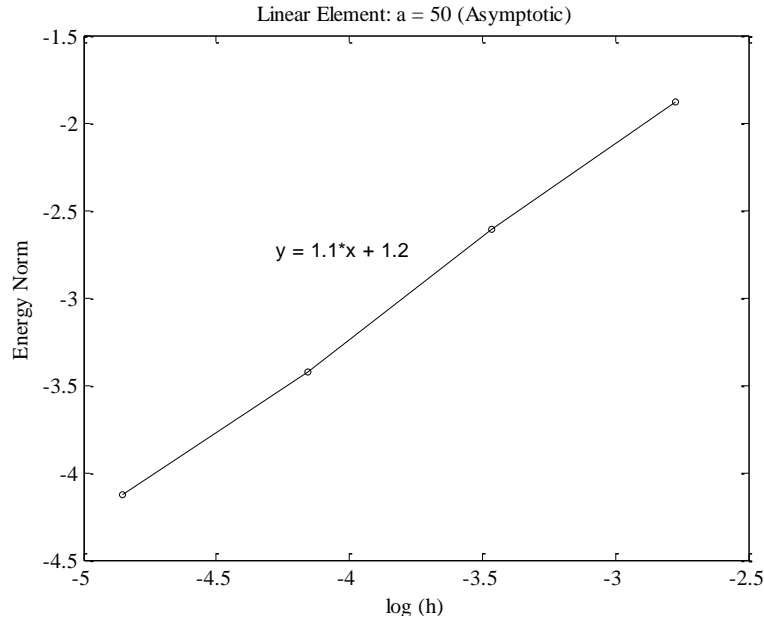
Cubic Elements:



Therefore, it is seen that the largest error of the derivative occurred at the tip of the nearly discontinuous region. To achieve the asymptotic convergence rate,

Linear Element:





Therefore, h needs to be at least $1/16$ to achieve the asymptotic convergence rate.

Problem 2 – Boundary value problems (BVPs) with non-smooth solutions – Adaptive finite element (MatLab programming required)

From Problem 1, it is seen that when the solution exhibits non-smooth behavior, the convergence of the solution is significantly reduced. A possible remedy is to adaptively refine the finite element mesh around the singular region.

Basic Ideas of Adaptive Algorithms:

Suppose we can compute the exact error e_K in element K . If the overall target error for the computation is γ , we want that $\sum_K e_K \leq \gamma$. Having a finite element mesh, we would like to construct a new mesh, with the minimal number of nodes, such that $\sum_K e_K$ meets the target error. This is actually a constrained nonlinear optimization problem. To facilitate its solution, we can introduce an iteration over successively finer grids. First, we need to *estimate* or bound the true error. The error bound in such estimates normally have the form $\varepsilon = \sqrt{\sum_K \varepsilon_K^2}$, where ε_K^2 is the contribution from element K . The estimates are only valid as the sum, but it is common to use the local components ε_K^2 of the sum as indicators for the refinement of individual elements. That means that we locally employ of the form $e_K^2 \leq \varepsilon_K^2$. Of course, ε_K should be a computationally attractive formula.

We let T^j be the mesh in the iteration j , that is, after j refinements of the initial mesh. Furthermore, let $m(T^j)$ be the number of element in T^j . We can then apply the values of

ε_K in the current mesh T^j to select the elements to be refined. Applying a mesh refinement algorithm yields a new mesh T^{j+1} . The selection of elements to be refined is often based on the principle that the error should be uniformly distributed throughout the mesh. Thus, we aim at having $\varepsilon_K^2 \leq \gamma^2 / m_{opt}$, where m_{opt} is the number of element in the final (optimal) mesh. A natural consequence is that an element K is marked for refinement if $\varepsilon_K^2 > \gamma^2 / m_{opt}$. The iteration is stopped when the total estimated error ε is less than the target error γ , or when the number of refinements exceeds a prescribed maximum level. For practical computations we often choose $\gamma = \eta \|u_h\|_e$, where η is given tolerance for the relative error in the global energy norm, if the energy norm is used for the estimate.

Adaptive finite element computation:

Choose initial mesh T^0

Compute u_h using the current mesh T^0 .

$j = 1$.

While $j \leq j_{max}$

$j = j + 1$

Compute estimator ε_K in each element

If the total error $\varepsilon = \sqrt{\sum_K \varepsilon_K^2} > \eta \|u_h\|_e$ then

Split the elements K for which $\varepsilon_K > \eta \|u_h\|_e / \sqrt{m(T^j)}$ equally into two smaller elements

Compute u_h using the new mesh T^{j+1}

Endwhile

Here the energy norm can be computed in the following way:

$$\|u_h\|_e = \left(\int_0^1 \nabla \mathbf{u}_h k(x) \nabla \mathbf{u}_h dx \right)^{\frac{1}{2}} = (\mathbf{u}_h^T \mathbf{K} \mathbf{u}_h)^{\frac{1}{2}}$$

where \mathbf{K} is the stiffness matrix.

In this problem, ε_K is chosen to be the ZZ (Zienkiewicz-Zhu) estimator, which is a popular indicator for low-order elements. It is defined as

$$\varepsilon_K = \left(\int_{\Omega^k} (\mathbf{q}^* - \mathbf{q}_h) k(x) (\mathbf{q}^* - \mathbf{q}_h) dx \right)^{\frac{1}{2}}$$

where $q_h = \frac{du_h}{dx}$. A very simply choice of \mathbf{q}^* is the smoothed version of q_h at the nodal point, using the Galerkin, least-squares, or L^2 projection methods. A MATLAB file *makeGradient.m* is provided to compute the \mathbf{q}^* given the solution vector \mathbf{u} .

Implementing the Adaptive finite element algorithm with linear finite element by modifying the 1DBVP MATLAB code. Resolve the problem 1 with $a=50$ and three different $x_0=0.2, 0.5, 0.8$. Give the final errors comparing with analytical solution and plot the adaptive mesh. Verify that the obtained mesh is refined near x_0 . Compare your solutions with that of uniform mesh with the same number of elements in the final mesh. Consider $\eta=5\%$ and $\eta=1\%$ for each case to check the convergence of the algorithm. The initial mesh consists of 2 elements and the maximum number of refinement is 15.

Solution:

We first modify the *main.m*:

```
% Initialize the first solver
Solve;
fprintf(1, 'Now, it is in iteration %d:\n', 1);
fprintf(1, 'The number of elements is %d\n', nel);
ErrorAnalysis;

nlevels = 15;
Is_Refine = 1;
grid_level = 1;
% Iteratively refine the grid
while ( Is_Refine && grid_level < nlevels)
    grid_level = grid_level + 1;
    Is_Refine = ErrorEstimator;
    if (Is_Refine)
        Solve;
        fprintf(1, 'Now, it is in iteration %d:\n', grid_level);
        fprintf(1, 'The number of elements is %d\n', nel);
        ErrorAnalysis;
    end
end

postprocessor;      % Postprocessor
```

The function *ErrorEstimator.m* is

```
function Is_Refine = ErrorEstimator;
include_variables;      % include global variables

Is_Refine = 0;          % Initialize the refinement flag

% Calculate the global energy norm
energy_norm = sqrt(d'*K * d);
clear global K

% Calculate the smoothing gradient at the nodes
gradient = makeGradient(d);

% Loop over all the elements to find the local error;
Local_error = zeros(nel,1);
```

```

for elmID = 1 : nel

    [gp,w] = gauss(ngp);           % extract gauss points and weights
    glb = Elems(:,elmID);         % Extract the global number of
                                % the element nodes
    coord = Nodes(glb)';         % Extract the global coordinates
                                % of the element nodes
    de = d(glb);                 % Extract the nodal value
    qe = gradient(glb);          % Extract the smooth gradient
    for i = 1:length(gp)         % loop over all the gauss points

        [n,dndxi] = FiniteElement_1D(gp(i)); % get basis functions and
                                                % derivatives at Gauss
points

        x = n*coord;             % Calculate the global
                                % coordinates of the
                                % current integration point
        dxdxi = dndxi*coord;     % Calculate the components of
                                % the Jaccobian matrix
        Jacc = dxdxi;           % Formulate the Jaccobian matrix

        detJxW = Jacc * w(i);    % Calculate integration weight

        dN = dndxi/Jacc;         % Calculate the derivatives of
                                % the basis function with
                                % respect to x direction in
                                % physical coordinates
        qh = n *qe;              % smooth gradient at the gauss points
        duh = dN*de;            % derivative at the gauss points

        Local_error(elmID) = Local_error(elmID) + pp(x)*(qh-
duh)^2*detJxW;
    end
    Local_error(elmID) = sqrt(Local_error(elmID));
end

sum_error = sqrt(norm(Local_error));

yita = 1e-2;                    % error threshold

if sum_error > yita*energy_norm % Refine the element
    newx = [];
    for elmID = 1 : nel
        if Local_error(elmID) > yita*energy_norm/sqrt(nel)
            Is_Refine = 1;
            glb = Elems(:,elmID);
            coord = Nodes(glb)';
            newx = [newx (coord(1)+coord(2))/2]; % split the element
into 2
        end
    end
end

if (Is_Refine)

```

```
nno = nno + length(newx); % update the number of points
nel = nel + length(newx); % update the number of elements

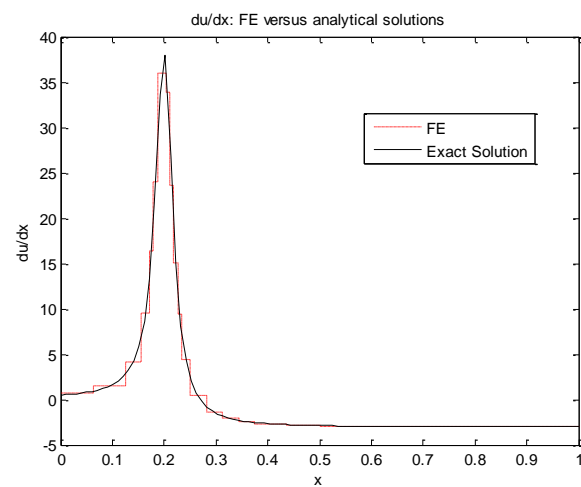
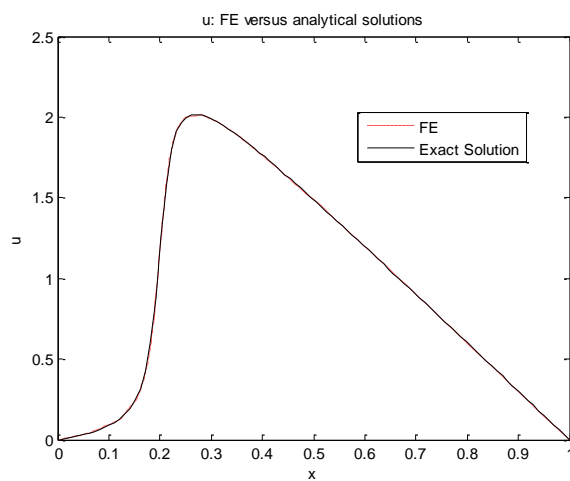
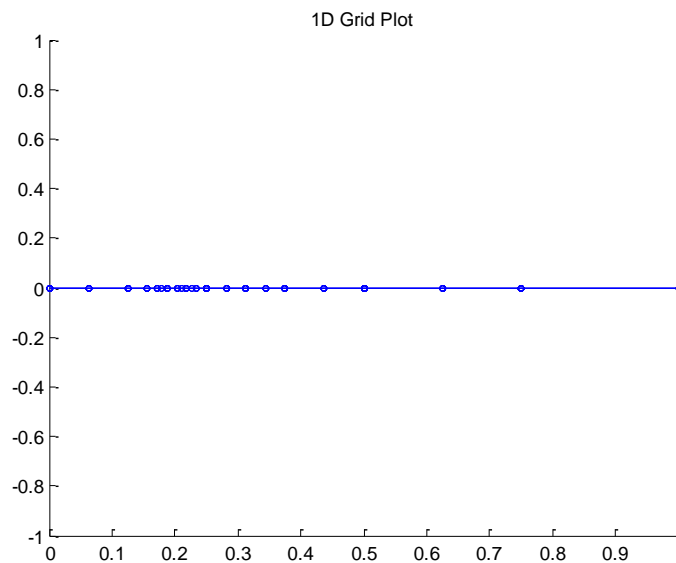
Nodes = sort([Nodes newx]); % Add the new nodes and sort them from
left to right

Elems = [1:(nno-1);      % Update the element connectivity
         2: nno ];
end
```

The results are

$v_0 = 0.2, \eta = 5\%$:

Adaptive Grid:



Now, it is in iteration 7:

The number of elements is 21

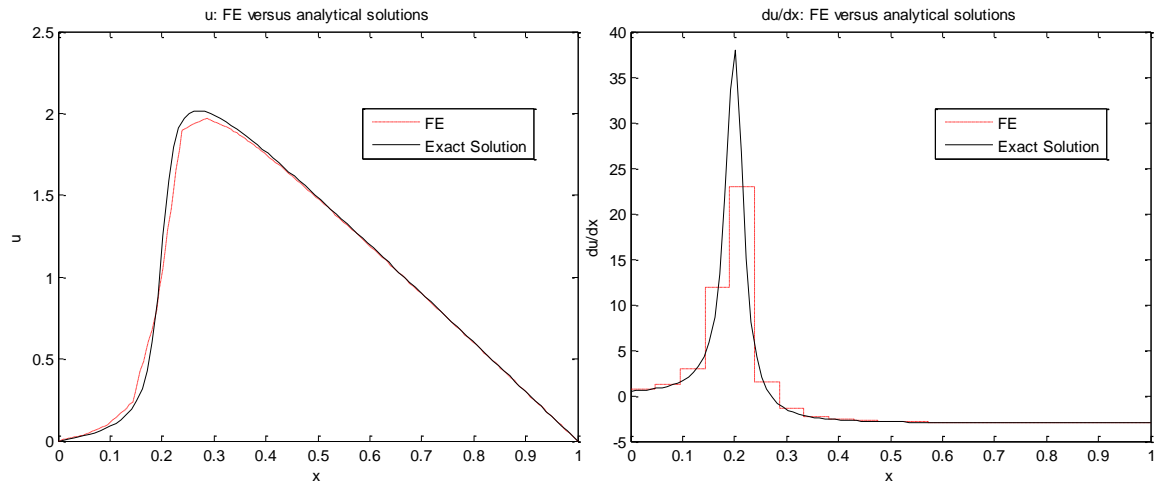
Error Reports:

The maximum norm error is 1.275768e-002

The L₂ norm error is 2.166001e-003

The energy norm error is 2.423451e-002

Uniform Grid with 21 elements :



Error Reports:

The maximum norm error is 2.033999e-001

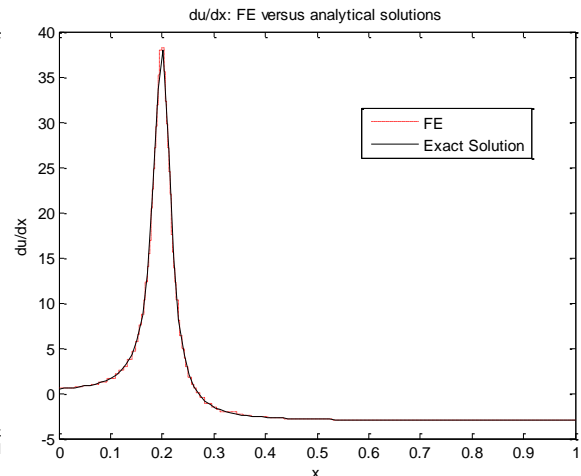
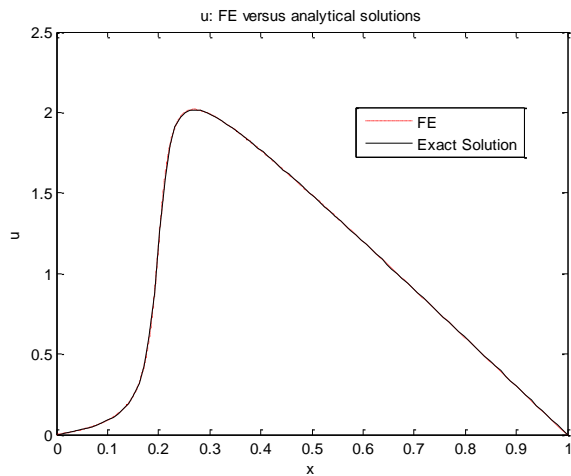
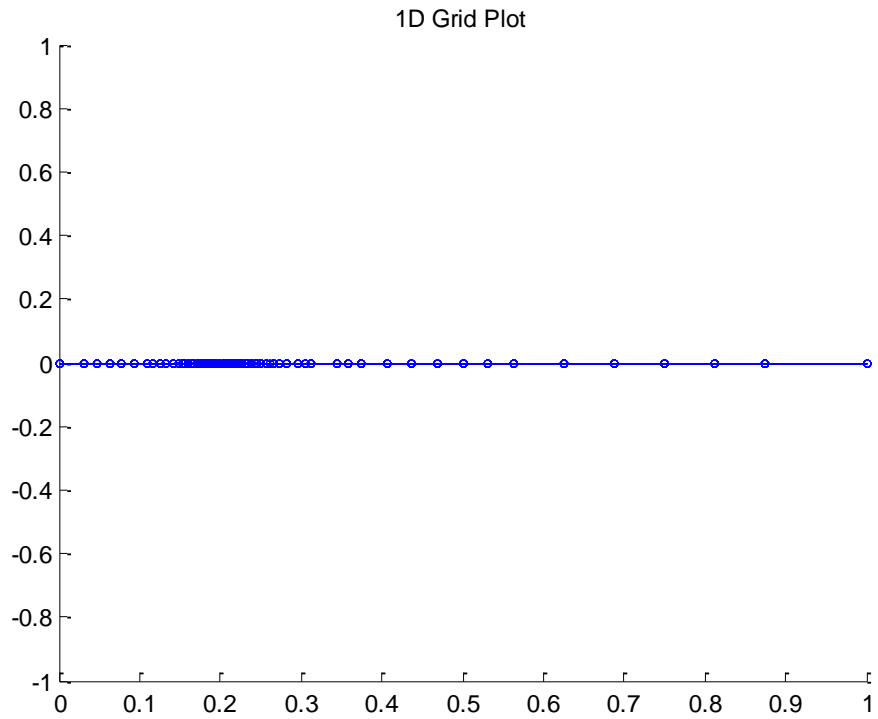
The L₂ norm error is 4.324413e-002

The energy norm error is 8.563105e-002

Therefore, it is obvious that using adaptive finite element method achieves much higher accuracy.

$v_0 = 0.2, \eta = 1\%$:

Adaptive Grid:



Now, it is in iteration 10:

The number of elements is 70

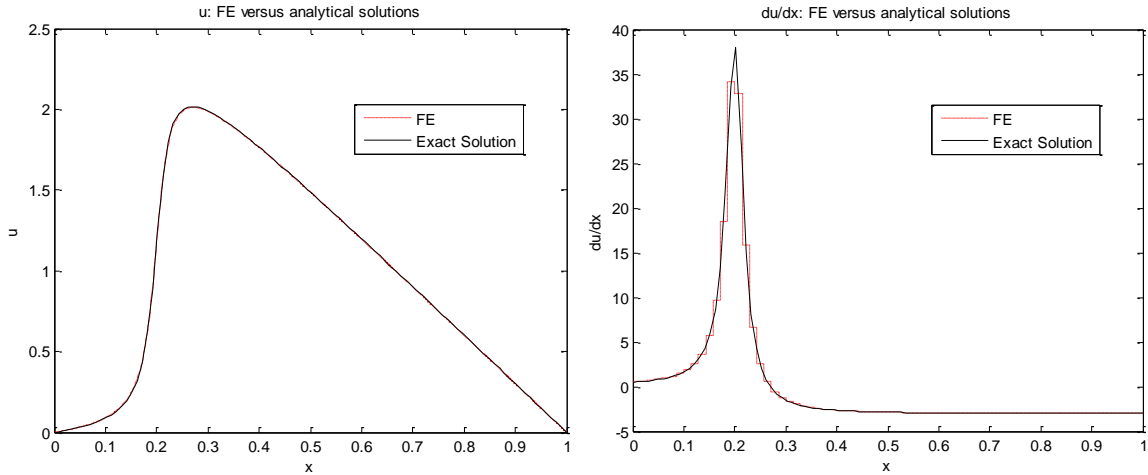
Error Reports:

The maximum norm error is 1.201122e-003

The L₂ norm error is 2.447796e-004

The energy norm error is 6.948799e-003

Uniform Grid with 70 elements :

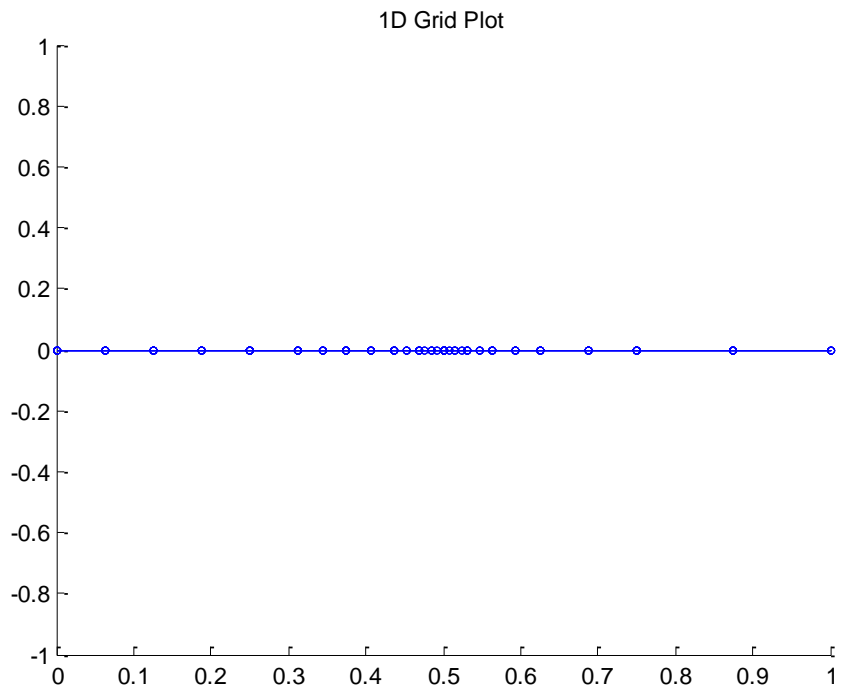


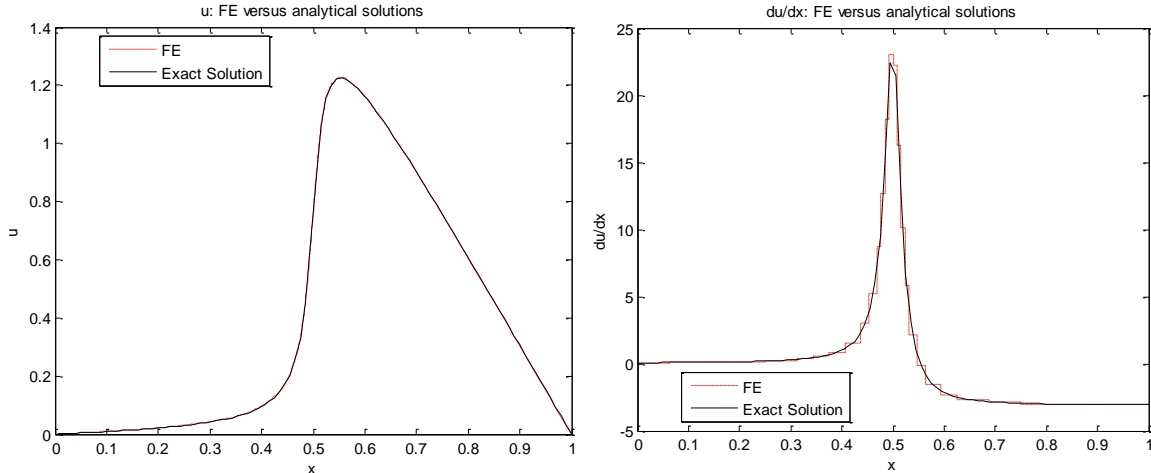
Error Reports:

The maximum norm error is 1.809538e-002
 The L₂ norm error is 3.811093e-003
 The energy norm error is 2.361601e-002

$v_0 = 0.5, \eta = 5\%$:

Adaptive Grid:





Now, it is in iteration 7:

The number of elements is 27

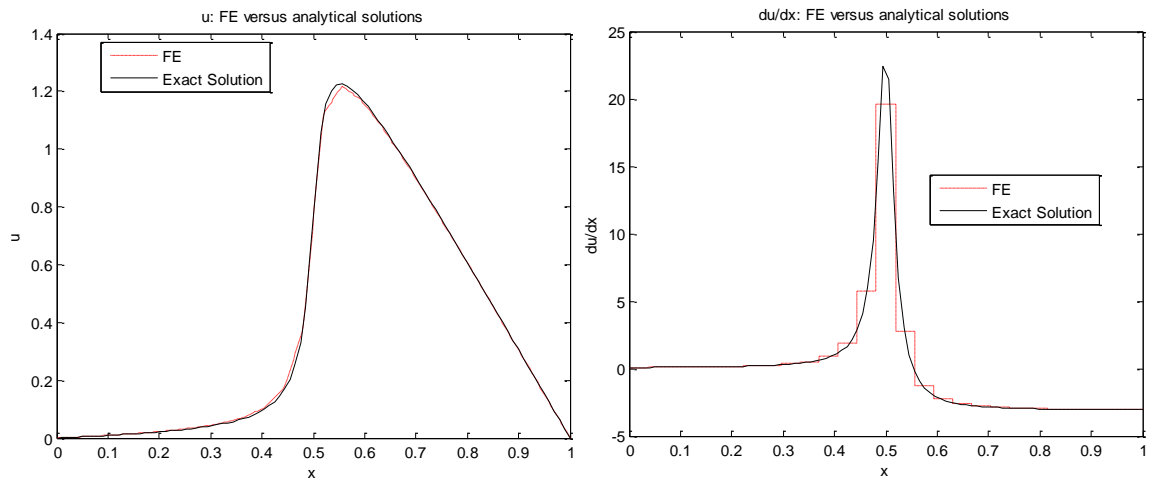
Error Reports:

The maximum norm error is 3.058346e-003

The L₂ norm error is 1.266543e-003

The energy norm error is 2.524421e-002

Uniform Grid with 27 elements :



Error Reports:

The maximum norm error is 2.636907e-002

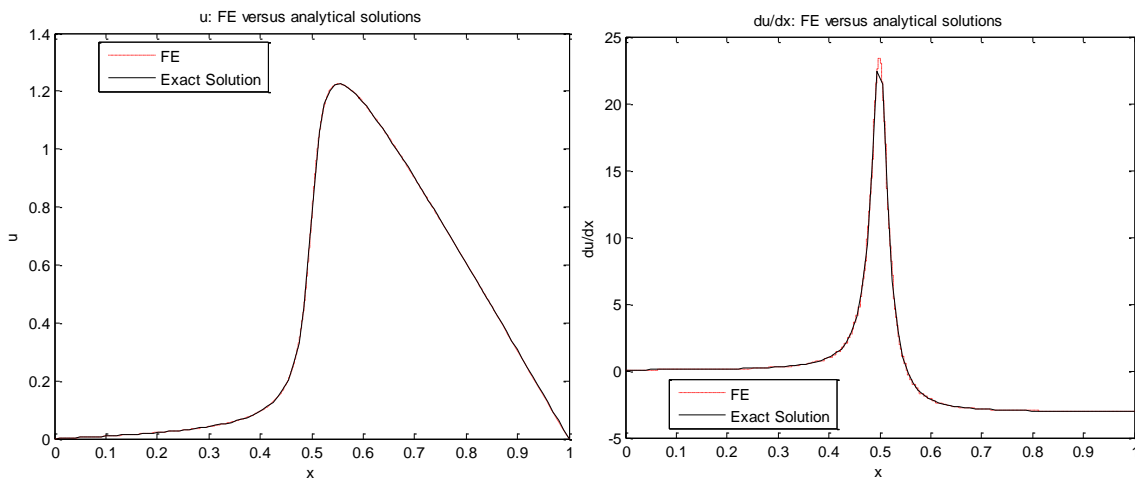
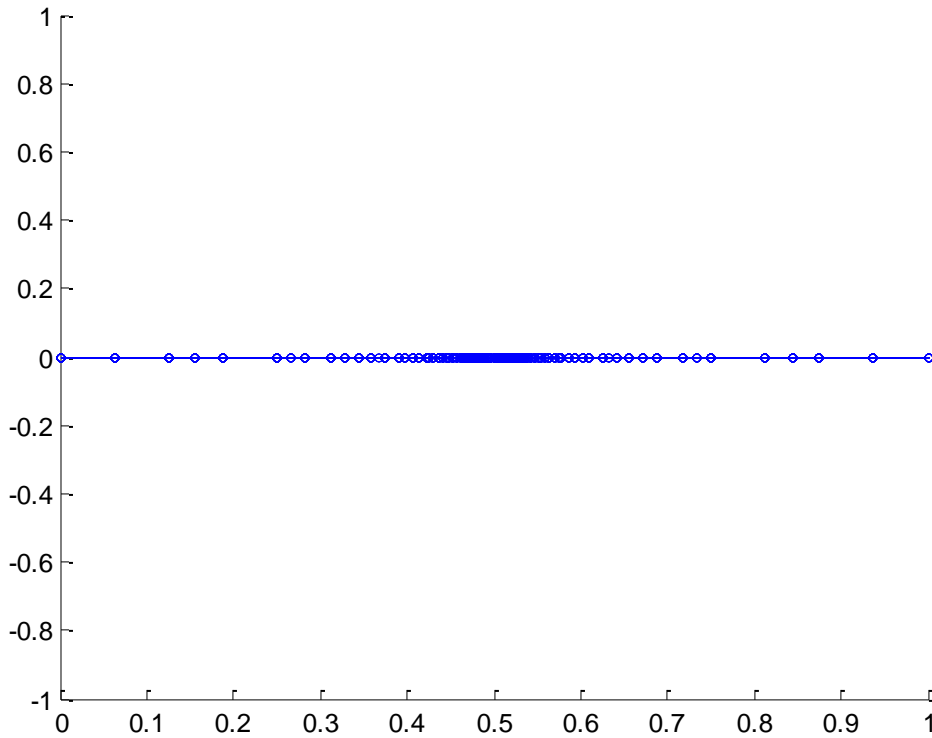
The L₂ norm error is 1.522892e-002

The energy norm error is 5.315527e-002

$v_0 = 0.5, \eta = 1\%$:

Adaptive Grid:

1D Grid Plot



Now, it is in iteration 12:

The number of elements is 91

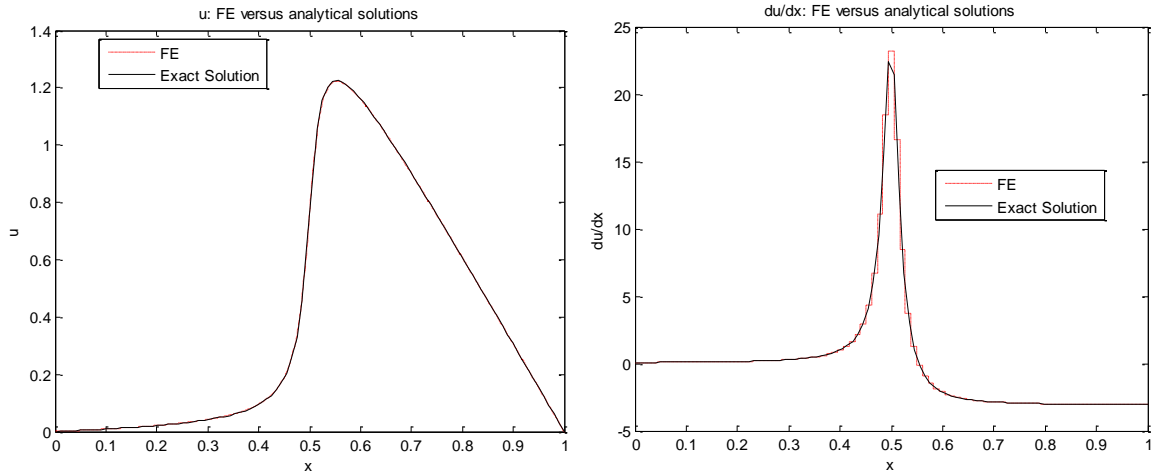
Error Reports:

The maximum norm error is 3.874715e-004

The L₂ norm error is 2.331656e-004

The energy norm error is 7.043511e-003

Uniform Grid with 91 elements :



Error Reports:

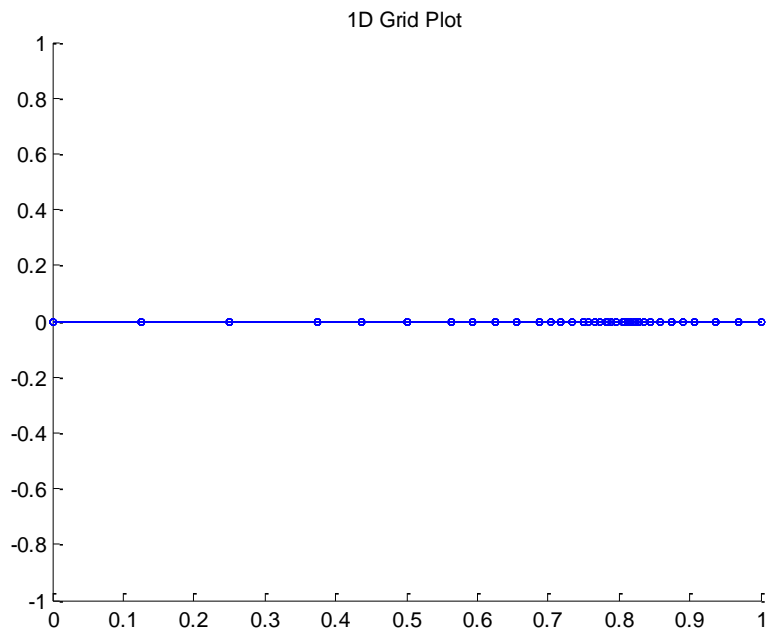
The maximum norm error is 7.307524e-003

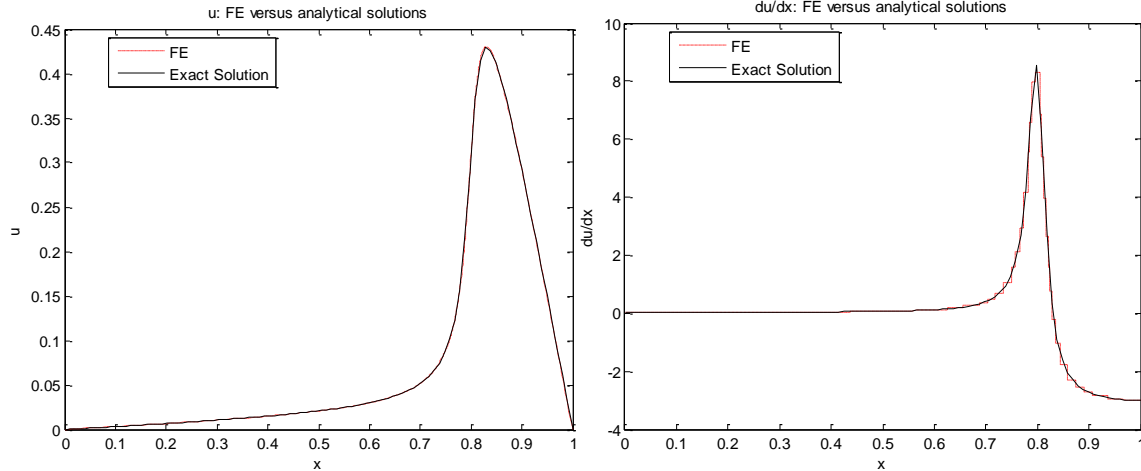
The L₂ norm error is 2.815145e-003

The energy norm error is 2.254971e-002

$\nu_0 = 0.8, \eta = 5\%$:

Adaptive Grid:





Now, it is in iteration 9:

The number of elements is 37

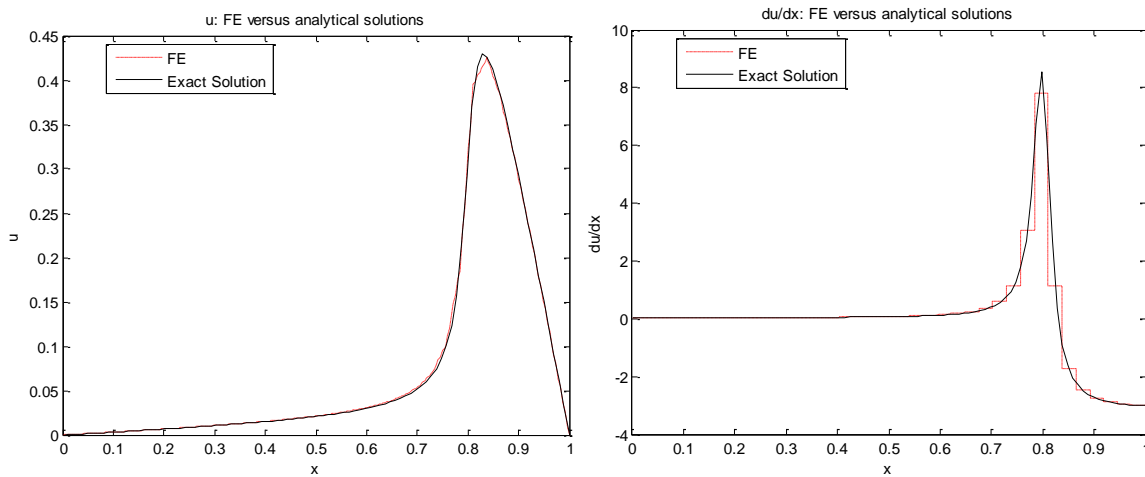
Error Reports:

The maximum norm error is 7.720785e-004

The L₂ norm error is 1.044899e-003

The energy norm error is 2.602892e-002

Uniform Grid with 37 elements :



Error Reports:

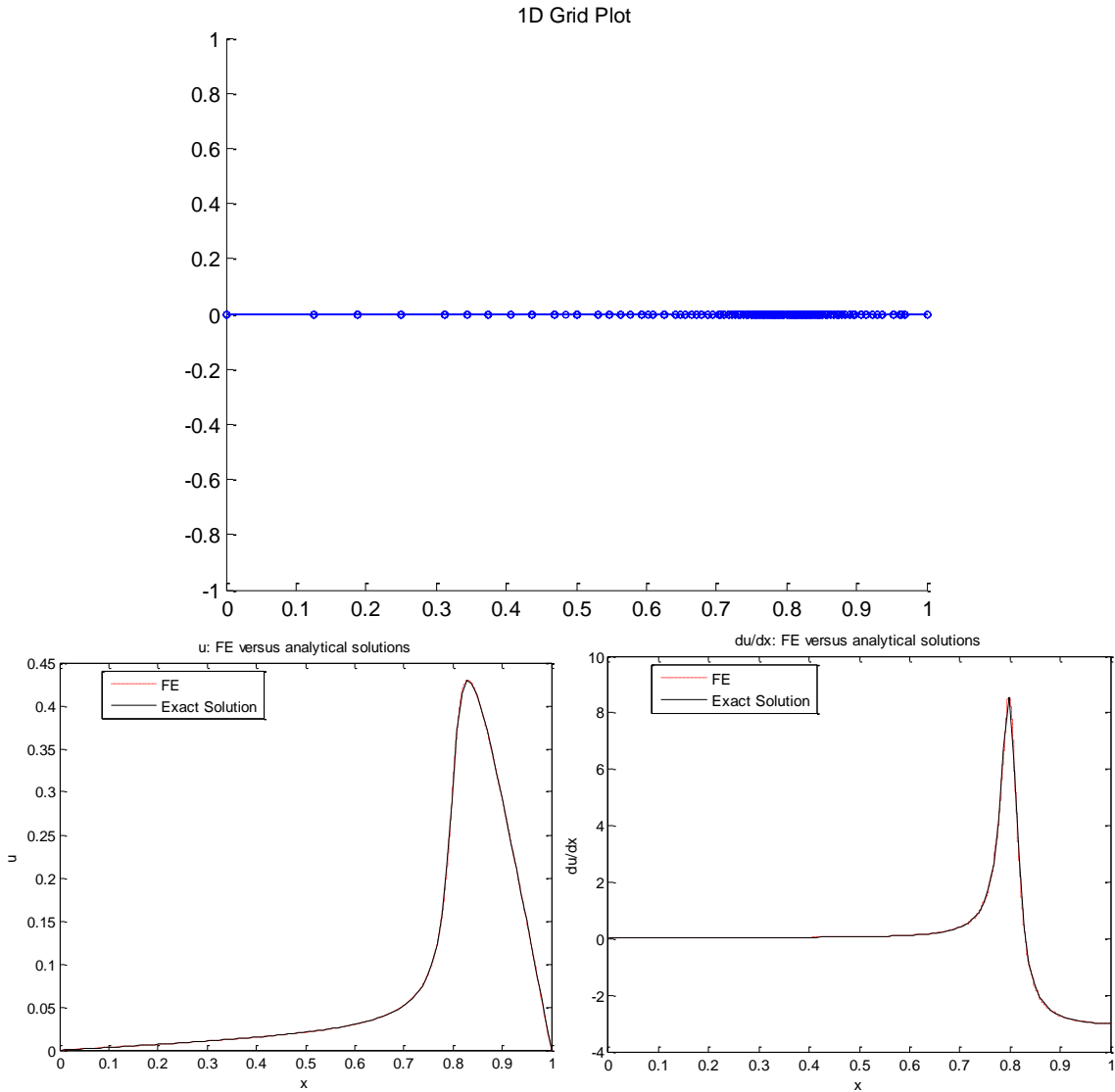
The maximum norm error is 1.063826e-002

The L₂ norm error is 1.864349e-002

The energy norm error is 7.696202e-002

$v_0 = 0.8, \eta = 1\%$:

Adaptive Grid:



Now, it is in iteration 12:

The number of elements is 133

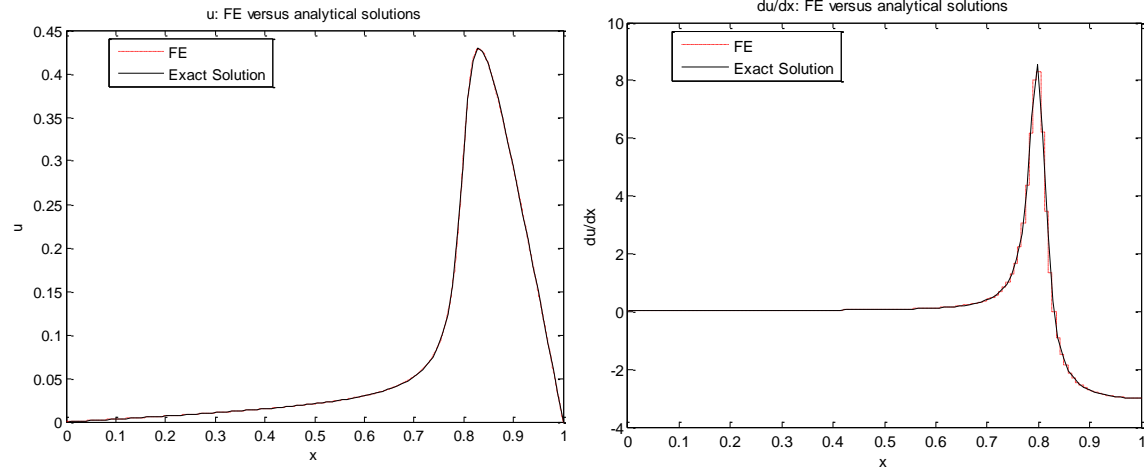
Error Reports:

The maximum norm error is 9.164394e-005

The L_2 norm error is 1.907806e-004

The energy norm error is 7.322681e-003

Uniform Grid with 133 elements :

**Error Reports:**

The maximum norm error is $1.320462e-003$

The L_2 norm error is $2.156525e-003$

The energy norm error is $2.502188e-002$

Overall, the developed algorithm can effectively capture the discontinuity in the solution and put more elements around that region. Comparing with that of uniform grid with same number of elements, the error of adaptive grid is nearly one order lower than that of uniform grid. Therefore, adaptive finite element method is more efficient in this case.