www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 6 June 2018, Page No. 24003-24065
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i6.07

A New Approach to Automatic Generation of All Graded Triangular and Quadrilateral Finite Element Mesh Over Polygonal Domains

H.T. Rathod[a*], Bharath Rathod[b]

[a] Department of Mathematics, Central College Campus, Bangalore University, Bangalore -560001, Karnataka State, India.

[b] Xavier Institute of Management and Entrepreneurship, Hosur Road, Electronic City Phase II, Bangalore, Karnataka 560034.

Abstract

This paper presents a new automesh generation method for a convex and cracked convex polygonal domains. We first decompose the polygon into simple sub regions in the shape of triangles. These simple regions are then triangulated to generate a mesh of graded 3-node triangular elements. We propose then an automatic 6-node triangluar conversion scheme by inserting midside nodes to these triangles. Each isolated 6-node triangle is split into four triangles according to the usual scheme,that is,by joining the midside nodes by straight lines. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given convex or cracked convex polygonal domain into all triangles, thus propagating a uniform or graded refinement. The quadrangulation of graded 3-node linear triangles is done by inserting three midside nodes and a centroidal node.Then each graded triangle is split into three quadrilaterals by joining the centroid to the midside nodes. This simple method generates a high quality mesh whose elements confirm well to the requested shape by refining the problem domain. Examples are presented to illustrate the simplicity and efficiency of the new mesh generation method for standard and arbitrary shaped triangles,convex and cracked convex polygonal domains. We have appended MATLAB programs which incorporate the mesh generation scheme developed in this paper. These programs provide valuable output on the nodal coordinates ,element connectivity and graphic display of the all graded triangular and quadrangular meshes for application to finite element analysis.

Keywords: finite elements, triangulation,quadrangulation ,all graded triangular and quadrilateral mesh generation, convex polygonal domain,cracked convex polygonal domain, uniform and graded refinement

1. Introduction

The Finite Element Method (FEM) has been invented by engineers around 1950 for solving the partial differential equations arising in solid mechanics; the main idea was to use the principle of virtual work for designing discrete approximations of the boundary value problems. The most popular reference on FEM in solid mechanics is the book of Zienckienwicz [1]. Generalizations to other fields of physics or engineering have been done by applied mathematicians through the concept of variational formulations and weak forms of the partial differential equations .

The FEM discretizes the continuous domain of the problem by means of a series of simple geometric forms called finite elements, for which the governing relations on the entire continuous domain are valid on each element. Under this assumption, the approximate solution in the entire continuous domain of the problem can be obtained by means of trial functions also called the shape functions. The FEM transforms the differential equation into an algebraic system of equations which can then be solved easily by known numerical methods.

The term Finite Element Method (FEM) appeared first in 1960, but the ideas behind it are even older and can be traced back to the engineering sciences. Being a powerful numerical tool for a variety of engineering disciplines, the FEM quickly found its way into applied mathematics where stability, discretization errors, and convergence rates are thoroughly investigated. It has been a very active field of research ever since. The standard way to gain accuracy of the approximate solution is to refine the triangulation of the computational domain, which introduces more degrees of freedom. A vast amount of publications deals with this so called h-version. An alternative way of enlarging the number of unknowns is to increase the polynomial degree of finite elements. This, so called, p-version is less common and its convergence speed strongly depends on the regularity of the solution to the PDE.

The modelling and numerical simulation of complex systems play an important role in many industrial, medical and economical applications. Very often, such systems can mathematically be described by partial differential equations (PDEs). Here, one can think for example of heat flow in materials or human tissues, aerodynamic properties of airplanes or determination of option prices in finance. In the last decades the development of efficient numerical methods to solve PDEs gave people together with the rising computing power the opportunity to simulate complex systems. Today this is done very successfully in many areas.Finite Element Analysis (FEA) is widely used in many fields including structures and optimization. The FEA in engineering applications comprises three phases: domain discretization, equation solving and error analysis. The domain discretization or mesh generation is the preprocessing phase which plays an important role in the achievement of accurate solutions.

The use of an adequate mesh is one of the main ingredients for an accurate numerical simulation. In order to obtain such a mesh, a versatile mesh generator and a mesh adaptive procedure must be available. Automatic mesh generation has received much attention from researchers on computational simulation, to minimize manual intervention, to improve mesh quality and to obtain more efficient procedures. Unstructured methodologies are becoming predominant due to the ability of modeling geometrically complex designs and because they are the natural environment for adaptivity, which may be the only hope for resolving very small scale features (e.g. boundary layers). Most finite element and finite volume codes use unstructured triangulations due to their geometrical flexibility and the low cost of linear triangular elements.

In this work, an automatic triangular mesh generator based on the advancing front technique is used as the main building block of a computational system for mesh generation that can build triangular and quadrilateral meshes over arbitrary domains. Quadrilateral elements are generated from an original mesh of triangles through a process of splitting.

The rate of convergence of the finite element method is greatly influenced by the existence of corners on the boundary. Recent investigations shows that proper refinement of the elements around the corners leads to the rate of convergence which is the same as it would be on domain with smooth boundary.

The simplified domains of many engineering problems contain sharp edges and corners, and these often pose important challenges in numerical analyses. Typical examples are reentrant corners in fluid mechanics , and crack tip problems in fracture mechanics etc.

Numerical solutions of elliptic boundary value problems defined on domains with corners have singular behaviour near the corners. This occurs even when data of the underlying problem are very smooth. Such singular behaviour affects the accuracy of the finite element method throughout the whole domain. For example, for the Poisson equation with homogeneous Dirichlet boundary conditions defined on a polygonal domain with re-entrant corners.

The precise description of an object containing *rounded and reentrant corners* leads to consider meshes with a large number of nodes in the corner neighborhood when the finite element method (FEM) is straightforwardly applied. Dealing with such meshes makes the computational work a time and resource consuming task. Instead of using a uniform mesh, one would like to use a non-uniform mesh that is denser near the singularities and coarser elsewhere to minimize the number of unknowns. In this paper, we investigate the choices of nonuniform mesh that give fast converging solutions

In section 2 of this paper, we present a novel mesh generation scheme of all graded triangular finite elements over triangular surfaces. This scheme converts the elements in background triangular mesh into triangles through the operation of graded subdivision. We first decompose the convex polygon into simple subregions in the shape of triangles. These simple subregions are then triangulated to generate a fine mesh of 3-node graded triangles. We propose then an automatic 3-node graded triangular to 3-node triangular conversion scheme in which each isolated 3-node graded triangle is split into four triangles according to the usual scheme, that is by adding three vertices in the middle of edges and then joining these three vertices by straight lines. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this fully discretizes the given polygonal domain into all 3-node triangles, thus propogating uniform graded refinement. In section 3 of this paper, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of 6- node graded triangular elements. We can then split each of these 6-node graded triangles into three quadrilaterals by joining the centroid to midside nodes of these triangles. In section 4,we have presented a method of piecing together of all triangular subregions and

eventually creating an all graded triangular and quadrilaqteral meshes for the given convex or cracked convex polygonal domain. In section 5,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for standard and arbitrary triangles,rectangles,squares, convex and cracked convex polygonal domains.

2.0 Mesh Generation

It is known that domain discretization plays an important role in finite element analysis as well as in numerous engineering analysis. As a result, various strategies and techniques for generating mesh automatically have been proposed [1-4]. The problem of meshing in two-dimensional case is defined as:
Given a 2D domain Ω together with grading functions g= g(ξ), defined over the entire domain Ω, the task of meshing is to discretize the domain Ω into triangles T or quadrilaterals Q or combination of both in consistency with the given grading functions g(ξ).
The grading functions g(ξ), which specify the element size of the discretization can be defined based on the consideration of the current analysis interests e.g. loading concentrations, boundary conditions etc. The first step in mesh generation is to divide the region into several disjoint sub-regions by openings to holes or connector to holes. For simplicity, here we assume that the region is composed of only straight lines and it is simply connected.

2.1 Division of a Standard Triangle

With reference to following figure Fig.0 we present some necessary definitions and a numerical scheme to generate graded triangles.[25-28]over an arbitrary polygonal domain.

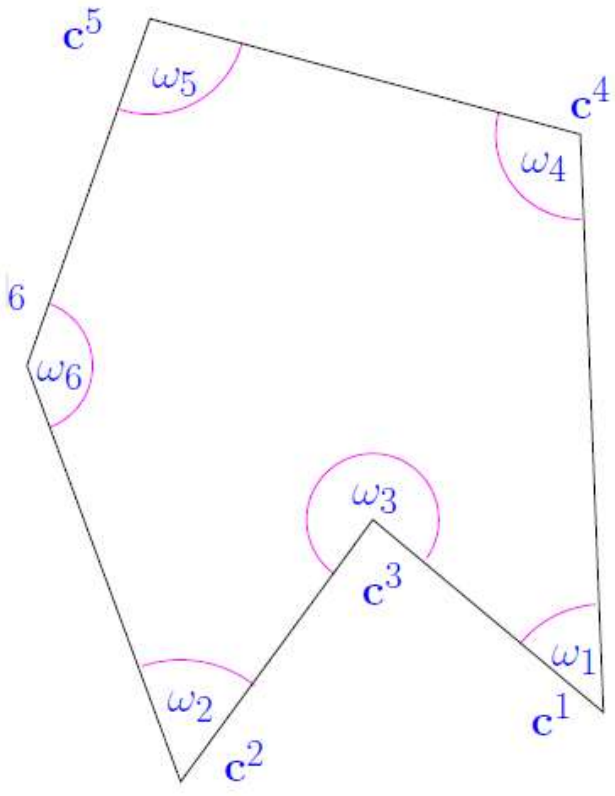


Fig.0 2D, Ω polygon, corners $\mathbf{c}^i$ with angles ω_i, $i = 1, \ldots, J$

Definition(Algebraically graded meshes in 1D):

A graded mesh of (0,1) generated by a grading function g:[0,1]→[0.1] is graded with grading factor β>0,if g(ξ)=ξ^{β}
Example:We can generate the (n+1) graded mesh points $\boldsymbol{x_k^n}$=$(\boldsymbol{k/n})^{\boldsymbol{\beta}}$,

ie $x_k^n = g(k/n)$,k=0,1,2,...n,n∈ N

Let us assume that the axes of standard triangle u and v are to be referred as $\hat{x}_1$ and $\hat{x}_2$ and c refers to a corner of the standard triangle or unit triangle.

Triangular graded meshes: β- *meshes*

Let K_0 = conv{(1, 0), (0, 0), (0, 1)} be the unit triangle. On K_0, we construct a parametric family of meshes which are graded towards the vertex (0, 0) so as to ensure an optimal rate of convergence of Lagrange interpolating Finite Elements of order $p \geq 1$.. Given an integer $m \geq 2$ and a so-called grading parameter $\beta \geq 1$, let

$$x_l^n = \left(\frac{l}{n}\right)^{\beta} \ ,l = 0,1,2, \ldots\ldots n$$

The nodes of the mesh that lie on the rectangular edges of K_0 are $(x_l^n, 0)$ and $(0, x_l^n)$, l = 0,1, . . . ,n. Then,being d_l the diagonal joining $(x_l^n, 0)$ and $(0, z)$, we divide d_l uniformly into l + 1 points. This defines all the nodes of a so-called β-graded mesh $T_{n,\beta}(K_0)$ on K_0.

If β = 1, *then* $T_{n,\beta}(K_0)$ *is quasi uniform with meshwidth* $\boldsymbol{h}$ = **1/n**.

Construction of $T_{n,\beta}(K_0)$,n∈ N on unit triangle K_0

(1) Generate 1D algebraically graded mesh on [0.1]

$0=x_0^n < x_1^n < \cdots\cdots\cdots < x_n^n = 1, x_l^n = (l/n)^{\beta} \ ,l = 0,1,2, \ldots\ldots n$

(2) Define "layers":L_j={ $\hat{x} \in K_0 : x_{j-1}^n < (\hat{x}_1 + \hat{x}_2) < x_j^n$ } $,j = 1,2,3, \ldots\ldots n$

(3) Generate Triangular mesh $T_{n,\beta}(K_0)|_{L_j}$ on each layer $j = 1,2,3, \ldots\ldots n$

(4)Union of $T_{n,\beta}(K_0)|_{L_j}$ =triangular mesh of K_0

(5) $T_{n,\beta}(K_0)|_{L_1}$ consists of a single triangle K^* adjacent to (0,0).

When $0<\boldsymbol{\beta}<1$**,we generate an algebraically graded mesh with respect to the edge** $\hat{x}_1+\hat{x}_2=1$.

Mesh points over the unit triangle domain

The layer L_j is bounded by the diagonals

$(\hat{x}_1 + \hat{x}_2)=x_{j-1}^n$……..(1 a)

$(\hat{x}_1 + \hat{x}_2)=x_j^n$ ……(1b)

Where ,it is important to note that $x_j^n=(\frac{j}{n})^{\beta}$ and $0 < x_j^n < 1, j = 1,2,3, \ldots.n$; with x_0^n =0 and x_n^n =1.

Equation (b) above refers to the diagonal joining the points $(x_j^n,0)$ and $(0, x_j^n)$,the solution is $(\hat{x}_1,\hat{x}_2)$=((j-k) x_j^n/j, kx_j^n/j) ,k=0,1,2,……..j; and the solution to Equation(a) is similar

We solve the above equations for $(\hat{x}_1,\hat{x}_2)$ and obtain (j) and (j+1) points at equal distances along the diagonals.The corner (0,0) is a single point and the diagonal joining (1,0) and (0,1) is divided into (n+1) points ((n-k)/n, k/n), k=0,1,....,n. Now, we have all the mesh points necessary for the generation of a graded triangular mesh over a standard triangle.The Cartesian mesh points for an arbitrary triangle can be computed by using affine transformation

We have used the above concepts and written the MATLAB code

[u,v,w] =triangularmeshpoints4singularcorner(p,q)

where u,v,w are area coordinates over the standard triangle T={u,v|0≤ $u, v \leq 1, u + v \leq 1$}, $u + v + w = 1$.

We also have p=n=number of divisions along each side of the standard triangle and q=β is the grading factor as input to the above MATLAB code This code denser mesh points in the neighbourhood of a corner for β> 1 and denser points in the vicinity of the edge(hypotenuse)

Clearly,β = 1,generates a uniform triangular mesh. The following outputs of the above code demonstrate this point of view.

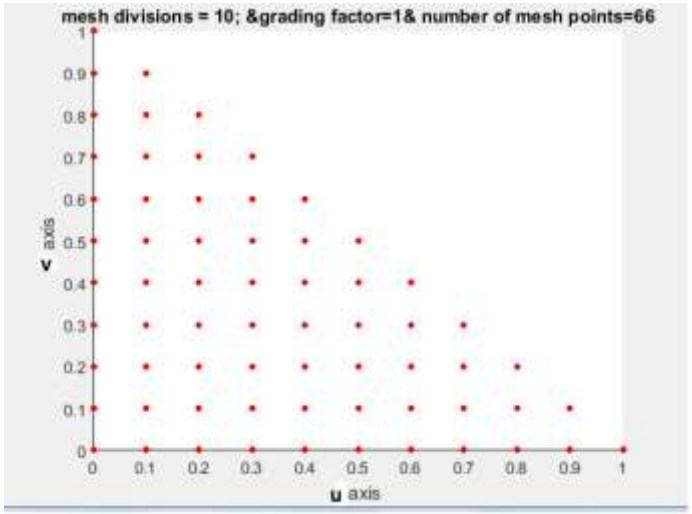


Fig.1a Distribution of mesh points for uniform mesh

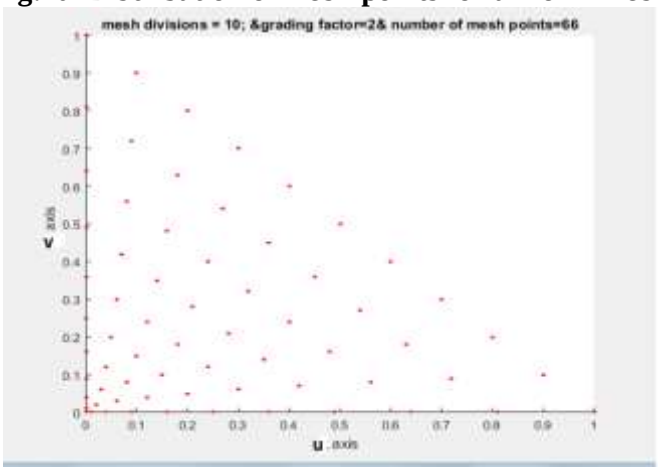


Fig.1b Distribution of mesh points for nonuniform mesh(graded mesh) denser near corner (0,0)

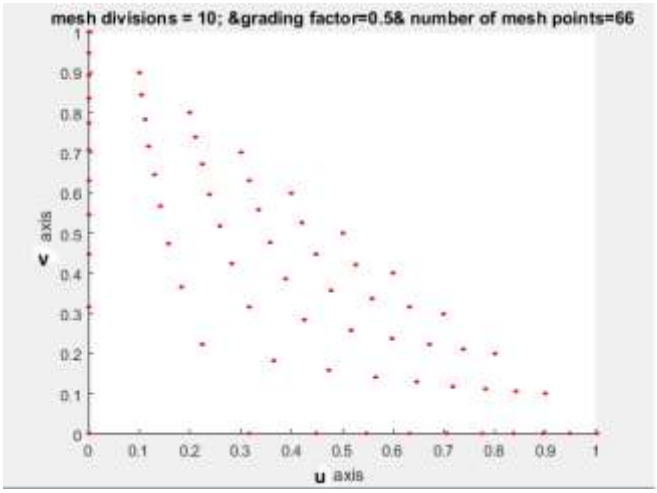


Fig.1c Distribution of mesh points for nonuniform mesh(graded mesh) denser near the edge(hypotenuse)

Graded Meshes Over A Standard Triangle

We present below a sample of graded meshes for $\beta>1$ which are denser in the vicinity of a corner.

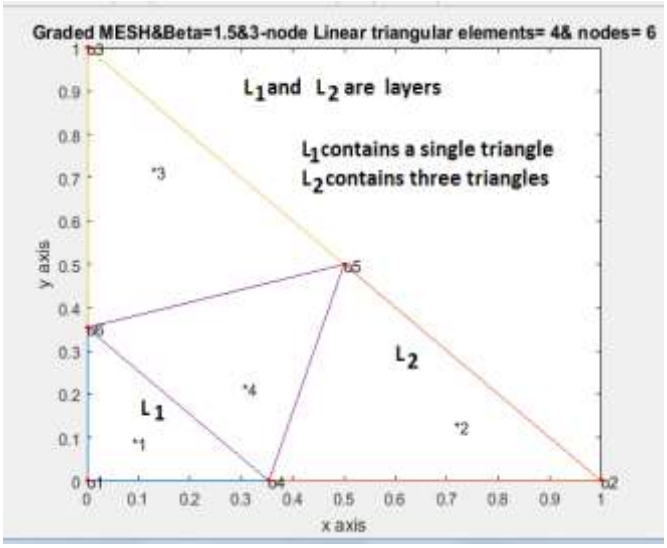


Fig.1d

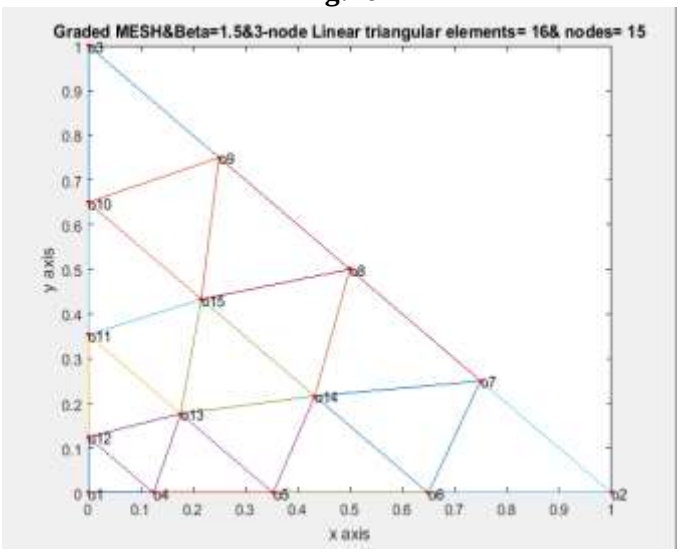


Fig.1e

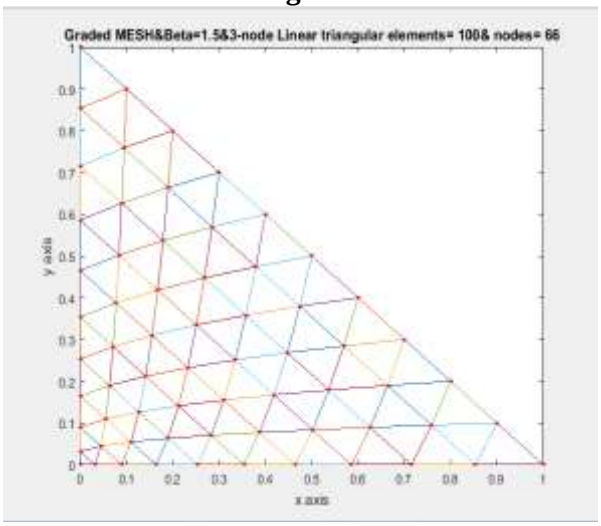


Fig.1f

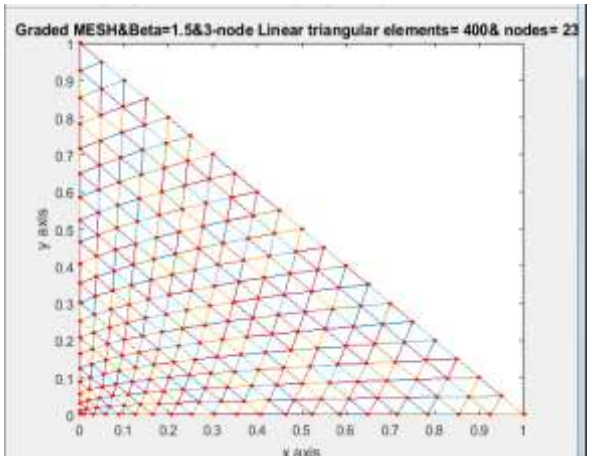


Fig.1g

The above examples indicate that for β≠1 the scheme can generate the 3-node graded triangles layer by layer only.If six node triangles are required,we can insert midside nodes to each of these.We can then insert an additional node at the centroid of these 3-node triangle and discritise this triangle into three quadrilaterals,that is by joining the centroid to the midside nodes.This will be further explained in the succeeding sections. The mesh generation is explained for uniform mesh generation,that is for β=1 which is equally valid for graded triangles β≠1,β>0

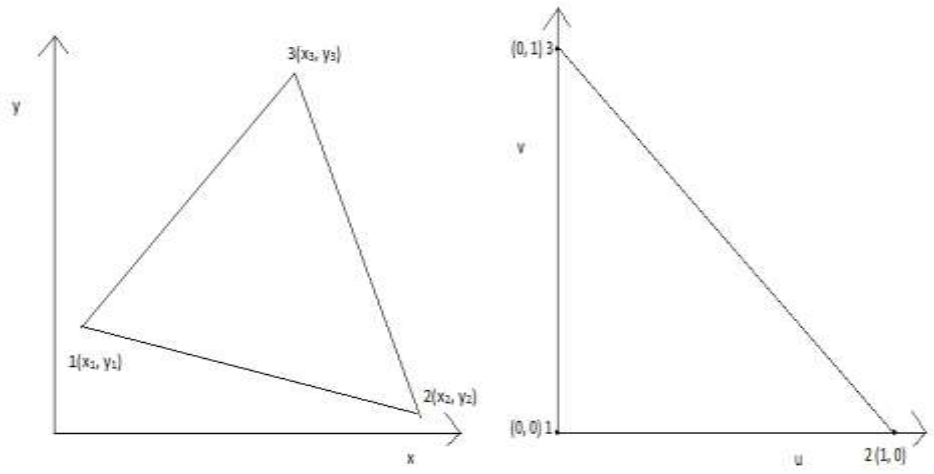
2.2 Division of an Arbitrary Triangle

We can map an arbitrary triangle with vertices ((x_i, y_i), $i = 1, 2, 3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 1h, 1i. The necessary transformation is given by the equations.

$$\boldsymbol{x = x_1 + (x_2 - x_1)u + (x_3 - x_1)v}$$

$$\boldsymbol{y = y_1 + (y_2 - y_1)u + (y_3 - y_1)v} \qquad \textbf{(2a-b)}$$

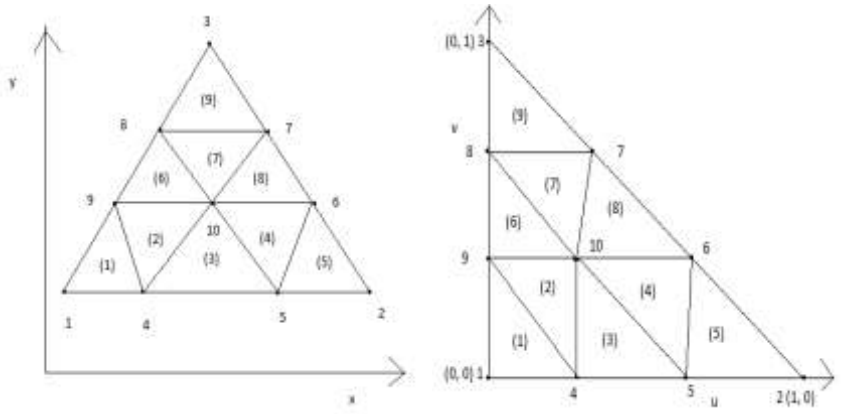
The mapping of eqn.(2a-b) describes a unique relation between the coordinate systems. This is illustrated by using the area coordinates and division of each side into three equal parts in Fig. 2a Fig. 2b. It is clear that all the coordinates of this division can be determined by knowing the coordinates ((x_i, y_i), $i = 1, 2, 3$) of the vertices for the arbitrary triangle. In general , it is well known that by making 'n' equal divisions on all sides and the concept of area coordinates, we can divide an arbitrary triangle into n^2 smaller triangles having the same area which equals Δ/n^2 where Δ is the area of a linear arbitrary triangle with vertices ((x_i, y_i), $i = 1, 2, 3$) in the Cartesian space.



1.h **1. i**

Fig. 1h An Arbitrary Linear Triangle in the (x, y) space

Fig. 1i A Right Isosceles Triangle in the (u, v)



2a **2b**

Fig. 2a Division of an arbitrary triangle into Nine triangles in Cartesian space

Fig. 2b Division of a right isosceles triangle into Nine right isosceles triangles in (u, v) space

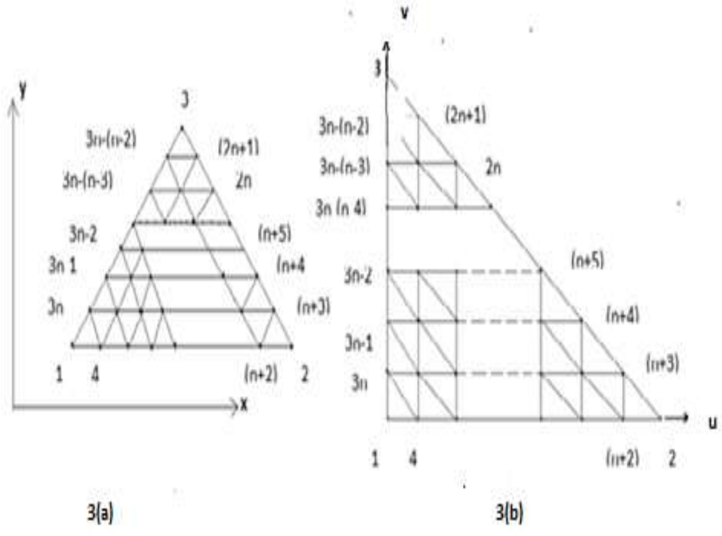


Fig. 3a Division of an arbitrary triangle with vertices$((x_i, y_i\ ), i = 1,2,3)$ into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisions of equal length

Fig. 3b Division of a right isosceles triangle with vertices{1(0,0),2(1,0),3(0,1)} into n^2 right isosceles triangle in (u, v) space, where each side is divided into n divisions of equal length

We have shown the division of an arbitrary triangle in Fig. 3a , Fig. 3b, We divided each side of the triangles (either in Cartesian space or natural space) into n equal parts and draw lines parallel to the sides of the triangles. This creates (n+1) (n+2)/2 nodes. These nodes are numbered from triangle base line l_{12} (letting l_{ij} as the line joining the vertex (x_i, y_i) and (x_j, y_j)) along the line $v = 0$ and upwards up to the line $v = 1$. The nodes 1, 2, 3 are numbered anticlockwise and then nodes 4, 5, ------, (n+2) are along line $v = 0$ and the nodes (n+3), (n+4), ------, 2n, (2n+1) are numbered along the line l_{23} i.e. $u + v = 1$ and then the node (2n+2), (2n+3), -------, 3n are numbered along the line $u = 0$. Then the interior nodes are numbered in increasing order from left to right along the line $v = \frac{1}{n}, \frac{2}{n}, ---, \frac{n-1}{n}$ bounded on the right by the line $+v = 1$. Thus the entire triangle is covered by (n+1) (n+2)/2 nodes. This is shown in the $\underline{rr}$ matrix of size $(n+1) \times (n+1)$, only nonzero entries of this matrix refer to the nodes of the triangles

$$\underline{\boldsymbol{rr}}=\begin{bmatrix} \mathbf{1}, & \mathbf{4}, & \mathbf{5}, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, & (\boldsymbol{n}+\mathbf{2}), & \mathbf{2} \\ \mathbf{3}\boldsymbol{n}, & (\mathbf{3}\boldsymbol{n}+\mathbf{1}), \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, & \mathbf{3}\boldsymbol{n}+(\boldsymbol{n}-\mathbf{2}), & (\boldsymbol{n}+\mathbf{3}), & \mathbf{0} \\ \mathbf{3}\boldsymbol{n}-\mathbf{1}, & \mathbf{3}\boldsymbol{n}+(\boldsymbol{n}-\mathbf{1}), \dots\dots\dots, & \mathbf{3}\boldsymbol{n}+(\boldsymbol{n}-\mathbf{2})+(\boldsymbol{n}-\mathbf{3}), & (\boldsymbol{n}+\mathbf{4}), \quad \mathbf{0}, & \mathbf{0} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \mathbf{3}\boldsymbol{n}-(\boldsymbol{n}-\mathbf{3}), & \frac{(\boldsymbol{n}+\mathbf{1})(\boldsymbol{n}+\mathbf{2})}{\mathbf{2}}, & \mathbf{2}\boldsymbol{n}, & \mathbf{0}, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & \mathbf{0} \\ \mathbf{3}\boldsymbol{n}-(\boldsymbol{n}-\mathbf{2}), & (\mathbf{2}\boldsymbol{n}+\mathbf{1}), & \mathbf{0}, & \mathbf{0}, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & \mathbf{0} \\ \mathbf{3}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0}, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & \mathbf{0} \end{bmatrix}$$

-----------(2c)

3. Triangulation of an Arbitrary Triangle

We now consider the quadrangulation of an arbitrary triangle. We first divide the arbitrary triangle into a number of equal size six node triangles. Let us define l_{ij} as the line joining the points (x_i, y_i) and

(x_j, y_j) in the Cartesian space (x, y). Then the arbitrary triangle with vertices at $((x_i, y_i), i = 1,2,3)$ is bounded by three lines l_{12}, l_{23}, and l_{31} . By dividing the sides l_{12}, l_{23}, l_{31} into $n = 2m$ divisions (m, an integer) creates m^2 six node triangular divisions. Then joining by straight lines the midpoints of the sides, we obtain four new triangles for each of these six node triangles. We have illustrated this process for the two and four divisions of l_{12}, l_{23}, and l_{31} sides of the arbitrary and standard triangles in Figs. 4 and 5

3.1 Two Divisions of Each side of an Arbitrary Triangle and Standard Triangle

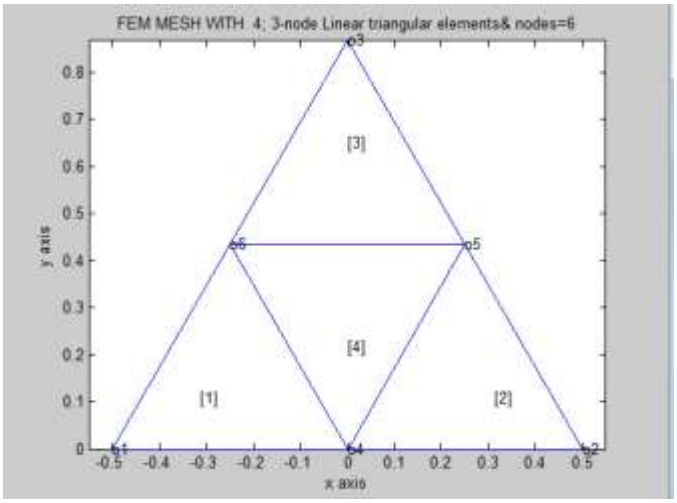


Fig 4(a). Division of an arbitrary triangle into four triangles

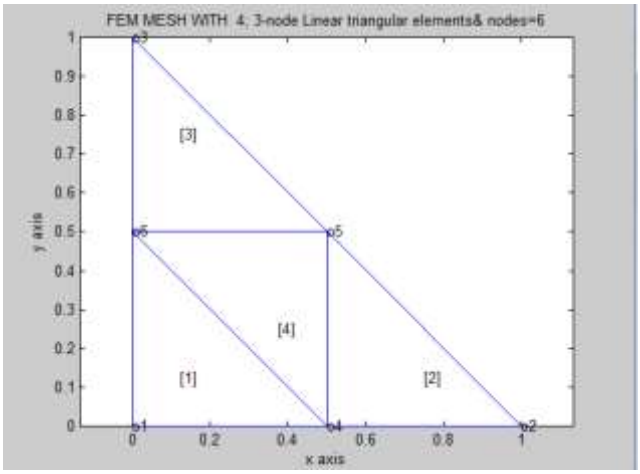


Fig.4(b) Division of a standard triangle into four triangles

3.2 Four Divisions of Each side of an Arbitrary Triangle and Standard Triangle

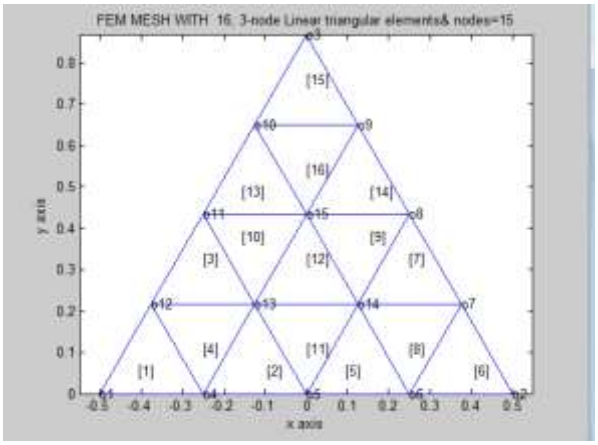


Fig 5(a). Division of an arbitrary triangle into 4 six node arbitrary triangles which give rise to 16,three node aritrary triangles

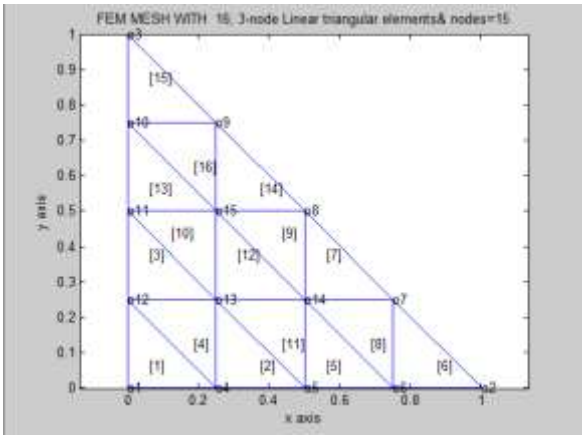


Fig.

Fig 5(b). Division of a standard triangle into 4 six node right isosceles triangles which give rise to 16,three node right isoscles triangles

In general, we note that to divide an arbitrary triangle into equal size six node triangle, we must divide each side of the triangle into an even number of divisions and locate points in the interior of triangle at equal spacing. We also do similar divisions and locations of interior points for the standard triangle. Thus n (even) divisions creates (n/2)^2 six node triangles in both the spaces. If the entries of the sub matrix $\underline{rr}\,(i\,;\,i+2,\ j\,;\,j+2)$ are nonzero then two six node triangles can be formed. If $\underline{rr}\,(i+1,\ j+2) = \underline{rr}\,\ (i+2,\ j+1;\ j+2) = 0$ then one six node triangle can be formed. If the sub matrices $\underline{rr}\,(i\,;\,i+2,\ j\,;\,j+2)$ is a (3×3) zero matrix , we cannot form the six node triangles. We now explain the creation of the six node triangles using the $\underline{rr}$ matrix of eqn.(2). We can form six node triangles by using node points of three consecutive rows and columns of $\underline{rr}$ matrix. This procedure is depicted in Fig. 6a for three consecutive rows $i\ , i+1,\ i+2$ and three consecutive columns $j\ ,\ j+1,\ j+2$ of the $\underline{rr}$ sub matrix

Formation of six node triangle using sub matrix $\underline{rr}$

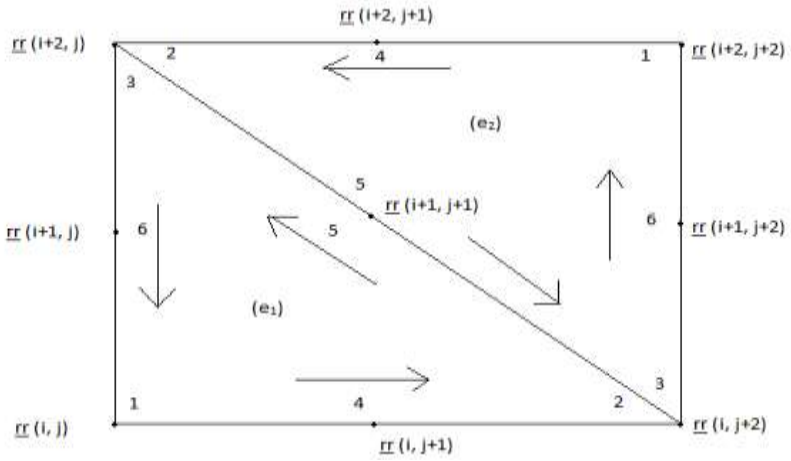


Fig. 6a Six node triangle formation for non zero sub matrix $\underline{rr}$

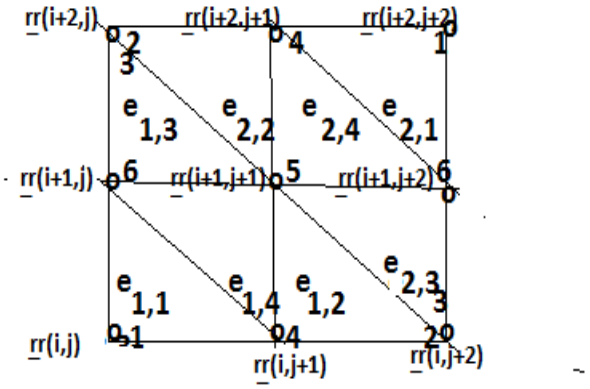


Fig.6b Formation of 3-node triangles for nonzero matrix $\underline{rr}$

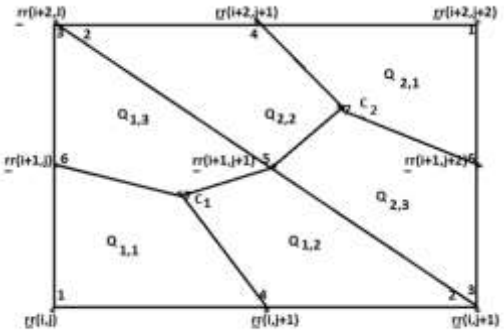


Fig.6c Formation of 4-node quadrilaterals for nonzero matrix $\underline{rr}$

If the sub matrix ($(\underline{rr}\ (\ k,l),k=i,i+1,i+2),l=j,\ j+1,\ j+2)$ is nonzero, then we can construct two six node triangles. The element nodal connectivity is then given by

(e_1) $< \underline{rr}\ (i,\ j), \underline{rr}\ (i,j+2), \underline{rr}\ (i+2,\ j), \underline{rr}\ (i,j+1), \underline{rr}\ (i+1,j+1), \underline{rr}\ (i+1,j) >$

(e_2) $< \underline{rr}\ (i+2,j+2), \underline{rr}\ (i+2,j), \underline{rr}\ (i,j+2), \underline{rr}\ (i+2,j+1), \underline{rr}\ (i+1,j+1), \underline{rr}$

$(i+1,j+2) >$ ----------(3)

If the elements of sub matrix ($(\underline{rr}\ (k,l), k = i, i+1, i+2),\ l = j,\ j+1,\ j+2)$ are nonzero, then as stated earlier, we can construct two six node triangles. We can create four 3-node tringles in each of these six node triangles (e_1) and (e_2). This procedure is depicted in Fig.6b, The nodal connectivity for the four 3- node triangles created in (e_1) and having element number n_1 are given as

$$e1,1 = T_{4n_1-3} < \underline{rr}\ (i,\ j)\ ,\ \underline{rr}\ (i,\ j+1),\ \underline{rr}\ (i+1,\ j) >$$

$$e1,2 = T_{4n_1-2} < \underline{rr}\ (i,\ j+2),\ \underline{rr}\ (i+1,\ j+1)\ ,\ \underline{rr}\ (i, j+1) >$$

$$e1,3 = T_{4n_{1-1}} < \underline{rr}\ (i+2,\ j),\ \underline{rr}\ (i+1,\ j)\ ,\ \underline{rr}\ (i+1, j+1) >$$

$$e1,4 = T_{4n_1} < \underline{rr}\ (i,\ j+1)\ ,\ \underline{rr}\ (i+1,\ j+1),\ \underline{rr}\ (i+1,\ j)\ >$$

----------(4a)

and the nodal connectivity for the 4 triangles created in (e_2) and having element number n_2 are given as

$$e2,1 = T_{4n_2-3} < \underline{rr}\ (i+2,\ j+2),\ \underline{rr}\ (i+2,\ j+1)\ ,\ \underline{rr}\ (i+1,\ j+2) >$$

$$e2,2 = T_{4n_2-2} < \underline{rr}\ (i+2,\ j),\ \underline{rr}\ (i+1,\ j+1)\ ,\ \underline{rr}\ (i+2,\ j+1) >$$

$$e2,3 = T_{4n_2-1} < \underline{rr}\ (i,\ j+2),\ \underline{rr}\ (i+1,\ j+2)\ ,\ \underline{rr}\ (i+1,\ j+1) >$$

$$e2,4 = T_{4n_2} < \underline{rr}\ (i+2,\ j+1),\ \underline{rr}\ (i+1,\ j+1)\ ,\ \underline{rr}\ (i+1, j+2) >$$

……(4b)

If the elements of sub matrix ($(\underline{rr}\ (k,l), k = i, i+1, i+2),\ l = j,\ j+1,\ j+2)$ are nonzero, then as stated earlier, we can construct two neighbouring six node triangles. We can create three 4-node quadrilaterals in each of these six node triangles (e_1) and (e_2) with centroids C_1 and C_2 respectively. This procedure is depicted in Fig.6c, The nodal connectivity for the four 4- node quadrilaterals q_ns' created in (e_1) and (e_2) and having element numbers n_1 and n_2 are given as

$$q_{3n_1-2} = <C_1, rr\ (i+1,\ j), \underline{rr}\ (i,\ j)\ ,\ \underline{rr}\ (i,\ j+1)\ >$$
$$q_{3n_1-1} = <C_1, rr\ (i,\ j+1), \underline{rr}\ (i,\ j+2)\ ,\ \underline{rr}\ (i+1,\ j+1)\ >$$
$$q_{3n_1} = <C_1, rr\ (i+1,\ j+1), \underline{rr}\ (i+2,\ j)\ ,\ \underline{rr}\ (i+1,\ j)\ >$$

----------(5a)

$$q_{3n_2-2} = <C_2, rr\ (i+1,\ j+2), \underline{rr}\ (i+2,\ j+2)\ ,\ \underline{rr}\ (i+2,\ j+1)\ >$$
$$q_{3n_2-1} = <C_2, rr\ (i+2,\ j+1), \underline{rr}\ (i+2,\ j)\ ,\ \underline{rr}\ (i+1,\ j+1)\ >$$
$$q_{3n_2} = <C_2, rr\ (i+1,\ j+1), \underline{rr}\ (i,\ j+2)\ ,\ \underline{rr}\ (i+1,\ j+2)\ >$$

----------(5b)

4. Triangulation of the Polygonal Domain

We can generate polygonal meshes by piecing together of the triangles with straight sides. We designate them as subregions(called LOOPs). The user specifies the shape of these LOOPs by designating six coordinates of each LOOP

As an example, consider the geometry shown in Fig. 7(a). This is a a square region R which is simply chosen for illustration. We divide this region into four LOOPs as shown in Fig.7(d). These

LOOPs 1,2,3 and 4 are triangles each with three sides. After the LOOPs are defined, the number of elements for each LOOP is selected to produce the mesh shown in Fig. 7(c).The complete mesh is shown in Fig.7(b)

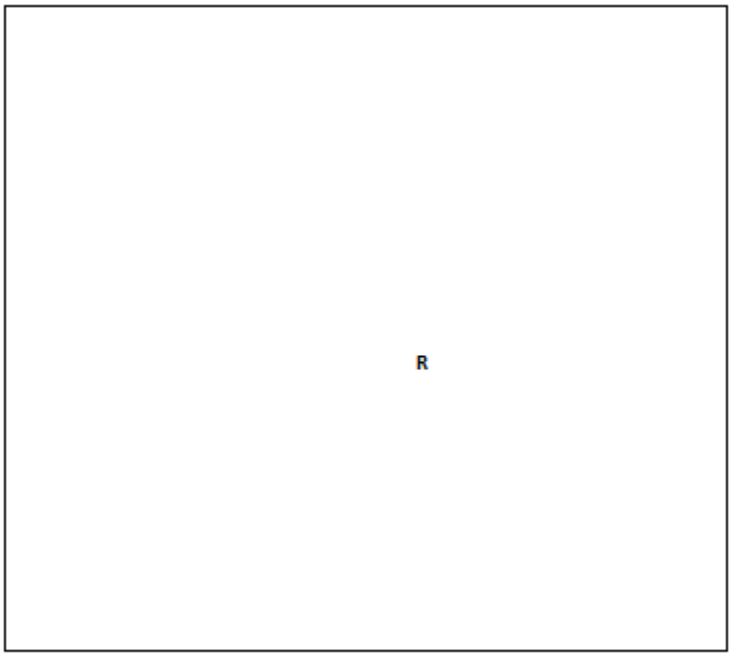


Fig.7(a) Region R to be analyzed

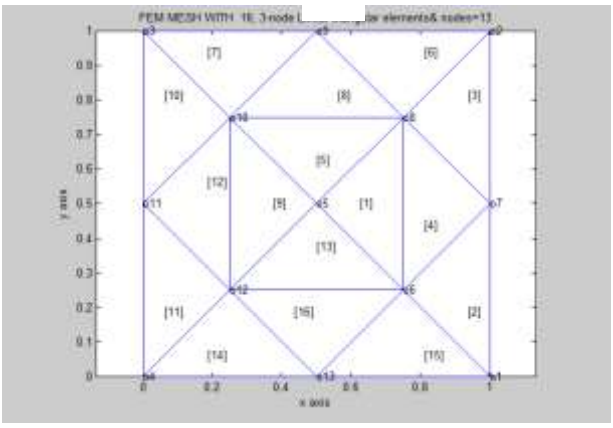
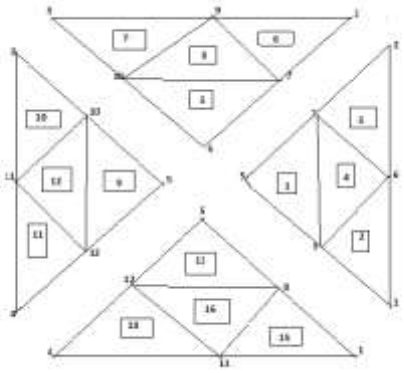
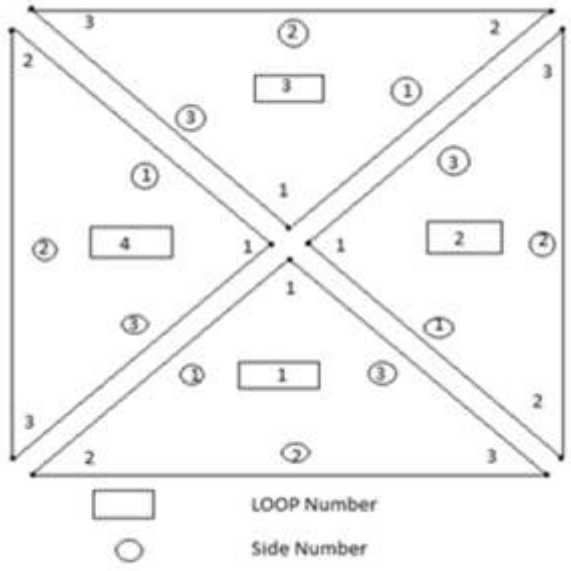


Fig.7(b) Example of completed Triangular mesh



7(c) Exploded view showing four loops



7(d) Example of a loop and side numbering scheme

By placing centroidal nodes in all the 16 triangles and joining the centroid to mid –side nodes, we can obtain the following all quadrilateral mesh containing 48 elements

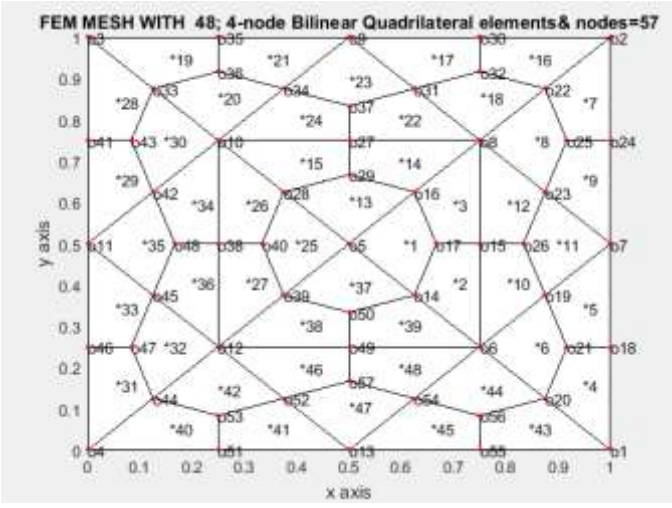


Fig.7(e) Example of completed Quadrilateral mesh

The above technique can also be applied to the graded triangular and quadrilateral finite element mesh generation.

(I)MESHES FOR CORNER SINGULARITY

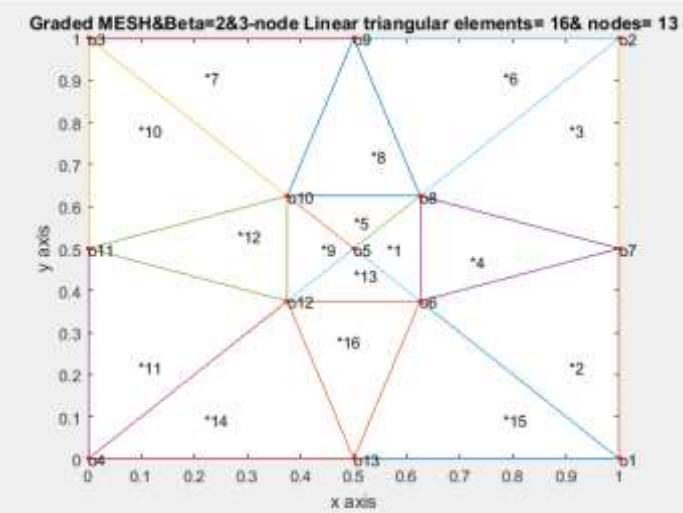


Fig.7(f)

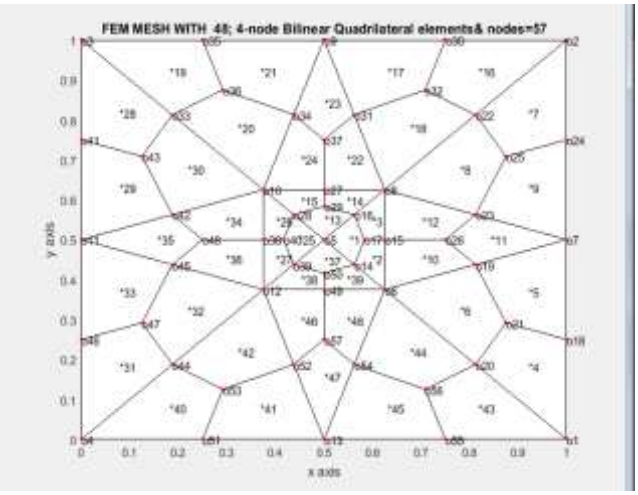


Fig.7(g)

(II)MESHES FOR EDGE SINGULARITY

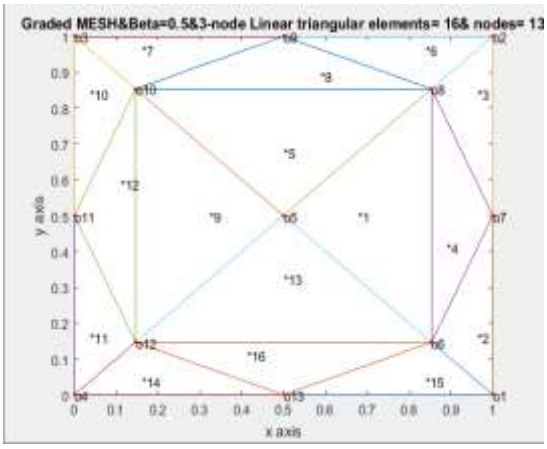


Fig.7(h)

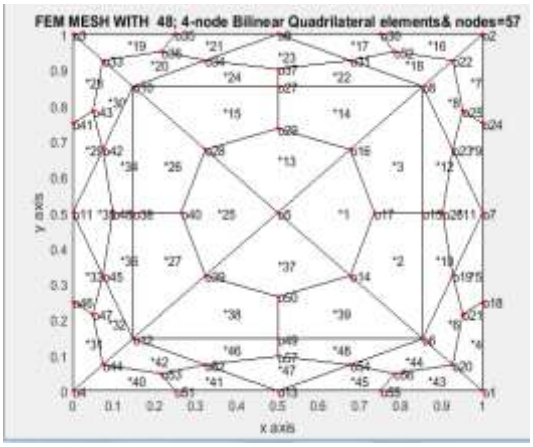


Fig.7(i)

4 Mesh Generation Scheme for Convex Polygonal Domains

How to define the LOOP geometry, specify the number of elements and piece together the LOOPs will now be explained

Joining LOOPs : A complete mesh is formed by piecing together LOOPs. This piecing is done sequentially thus, the first LOOP formed is the foundation LOOP, with subsequent LOOPs joined either to it or to other LOOPs that have already been defined. As each LOOP is defined, the user must specify for each of the three sides of the current LOOP.

In the present mesh generation code, we aim to create a convex polygon. This requires a simple procedure. We join side 3 0f LOOP 1 to side 1 of LOOP 2, side 3 of LOOP 2 will joined to side 1 of LOOP 3, side 3 of LOOP 3 will be joined to side 1 of LOOP 4. Finally side 3 of LOOP 4 will be joined to side 1 of LOOP 1.

When joining two LOOPs, it is essential that the two sides to be joined have the same number of divisions. Thus the number of divisions remains the same for all the LOOPs. We note that the sides of LOOP (i) and side of LOOP ($i+1$) share the same node numbers. But we have to reverse the sequencing of node numbers of side 3 and assign them as node numbers for side 1 of LOOP ($i+1$). This will be required for allowing the anticlockwise numbering for element connectivity

5. Application Examples

5.1 Mesh Generation Over an Arbitrary Triangle

In applications to boundary value problems due to symmetry considerations, we may have to discretize an arbitrary triangle. Our purpose is to have a code which automatically generates triangulations and quadrangulations of the domain by assuming the input as coordinates of the vertices.

5.2 Mesh Generation for Polygonal Domain

In several physical applications in science and engineering, the boundary value problem require meshes generated over polygons. Again our aim is to have a code which automatically generates a mesh of linear triangles and quadrilaterals for the complex domains such as those in [21,22]. We use the theory and

procedure developed in sections 2, 3 and 4 for this purpose. The following MATLAB codes are written for this purpose.

(1)triangular_mesh4LinearTrianglesSingularCorner_t3Nq4.m
(2) triangulation4polygonal_domain_coordinatesSingularCorner.m
(3) nodaladdresses_for4XnXn_LinearTriangles.m
(4) nodaladdresses_for4XnXn_LinearTriangles_trial.m
(5) triangularmeshpoints4singularcorner.m

We have included some meshes generated by using the above codes written in MATLAB. We illustrate the mesh generation for a standard triangle and an arbitrary triangle and polygonal domains. Some sample commands to generate these meshes are included in comment lines. In all the codes sample input data is included for easy access.

6 Conclusions

An automatic indirect triangular mesh generator which uses the splitting technique is presented for the two dimensional convex and a cracked convex polygonal domains. This mesh generation is made fully automatic and allows the user to define the problem domain with minimum amount of input such as coordinates of boundary. We first decompose the polygon into simple sub regions in the shape of triangles. These simple regions are then triangulated to generate a mesh of graded 3-node triangular elements. We propose then an automatic 6-node trianglar conversion scheme by inserting midside nodes to these triangles. Each isolated 6-node triangle is split into four triangles according to the usual scheme,that is,by joining the midside nodes by straight lines. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given polygonal domain into all triangles, thus propagating a uniform or graded refinement. The quadrangulation of graded 3-node linear triangles is done by inserting three midside nodes and a centroidal node.Then each graded triangle is split into three quadrilaterals by joining the centroid to the midside nodes. This simple method generates a high quality mesh whose elements confirm well to the requested shape by refining the problem domain.

We have also appended MATLAB programs which provide the nodal coordinates, element nodal connectivity and graphic display of the generated the graded all triangular and quadrilateral mesh for the standard triangle, an arbitrary triangle, a square and a convex or cracked convex polygonal domain. We believe that this work will be useful for various applications in science and engineering.

References

[1] Zienkiewicz. O. C, Philips. D. V, An automatic mesh generation scheme for plane and curved surface by isoparametric coordinates, Int. J.Numer. Meth.Eng, 3, 519-528 (1971)

[2] Gardan. W. J and Hall. C. A, Construction of curvilinear coordinates systems and application to mesh generation, Int. J.Numer. Meth. Eng 3, 461-477 (1973)

[3] Cavendish. J. C, Automatic triangulation of arbitrary planar domains for the finite element method, Int. J. Numer. Meth. Eng 8, 679-696 (1974)

[4] Moscardini. A. O , Lewis. B. A and Gross. M A G T H M – automatic generation of triangular and higher order meshes, Int. J. Numer. Meth. Eng, Vol 19, 1331-1353(1983)

[5] Lewis. R. W, Zheng. Y, and Usmam. A. S, Aspects of adaptive mesh generation based on domain decomposition and Delaunay triangulation, Finite Elements in Analysis and Design 20, 47-70 (1995)

[6] W. R. Buell and B. A. Bush, Mesh generation a survey, J. Eng. Industry. ASME Ser B. 95 332-338(1973)

[7] Rank. E, Schweingruber and Sommer. M, Adaptive mesh generation and transformation of triangular to quadrilateral meshes, Comn. Appl. Numer. Methods 9, 11 121-129(1993)

[8] Ho-Le. K, Finite element mesh generation methods, a review and classification, Computer Aided Design Vol.20, 21-38(1988)

[9] Lo. S. H, A new mesh generation scheme for arbitrary planar domains, Int. J. Numer. Meth. Eng. 21, 1403-1426(1985)

[10] Peraire. J. Vahdati. M, Morgan. K and Zienkiewicz. O. C, Adaptive remeshing for compressible flow computations, J. Comp. Phys. 72, 449-466(1987)

[11] George. P.L, Automatic mesh generation, Application to finite elements, New York , Wiley (1991)

[12] George. P. L, Seveno. E, The advancing-front mesh generation method revisited. Int. J. Numer. Meth. Eng 37, 3605-3619(1994)

[13] Pepper. D. W, Heinrich. J. C, The finite element method, Basic concepts and applications, London, Taylor and Francis (1992)

[14] Zienkiewicz. O. C, Taylor. R. L and Zhu. J. Z, The finite element method, its basis and fundamentals, 6th Edn, Elsevier (2007)

[15] Masud. A, Khurram. R. A, A multiscale/stabilized finite element method for the advection- diffusion equation, Comput. Methods. Appl. Mech. Eng 193, 1997-2018(2004)

[16] Johnston. B.P, Sullivan. J. M and Kwasnik. A, Automatic conversion of triangular finite element meshes to quadrilateral elements, Int. J. Numer. Meth. Eng. 31, 67-84(1991)

[17] Lo. S. H, Generating quadrilateral elements on plane and over curved surfaces, Comput. Stuct. 31(3) 421-426(1989)

[18] Zhu. J, Zienkiewicz. O. C, Hinton. E, Wu. J, A new approach to development of automatic quadrilateral mesh generation, Int. J. Numer. Meth. Eng 32(4), 849-866(1991)

[19] Lau. T. S, Lo. S. H and Lee. C. S, Generation of quadrilateral mesh over analytical curved surfaces, Finite Elements in Analysis and Design, 27, 251-272(1997)

[20] Park. C, Noh. J. S, Jang. I. S and Kang. J. M, A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints, Computer Aided Design 39, 258- 267(2007)

[21] Moin.P, Fundamentals of Engineering Numerical Analysis,second edition,Cambridge University Press(2010)

[22] Fausett.L.V, Applied Numerical Analysis Using MATLAB, second edition, Pearson Education.Inc(2008)

[23] Thompson.E.G, Introduction to the finite element method,John Wiley & Sons Inc.(2005)

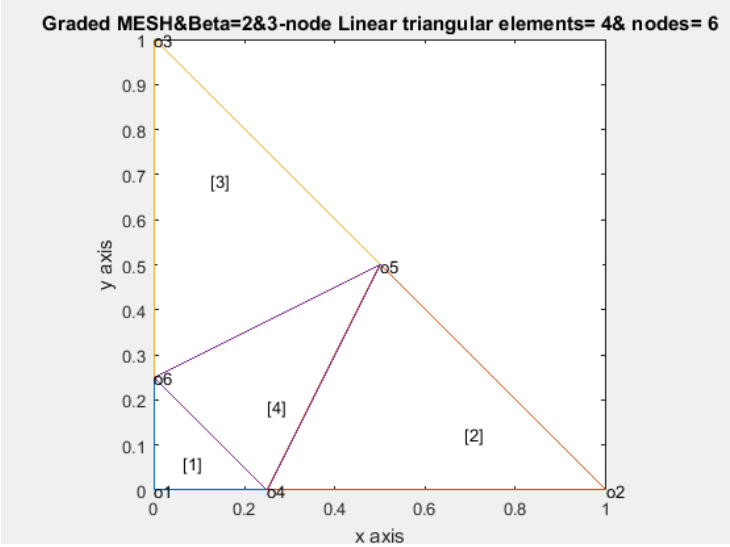
[24] ProgramMESHGEN:www.ce.memphis.edu/7111/notes/fem_code/MESHGEN_ tutorial.pdf

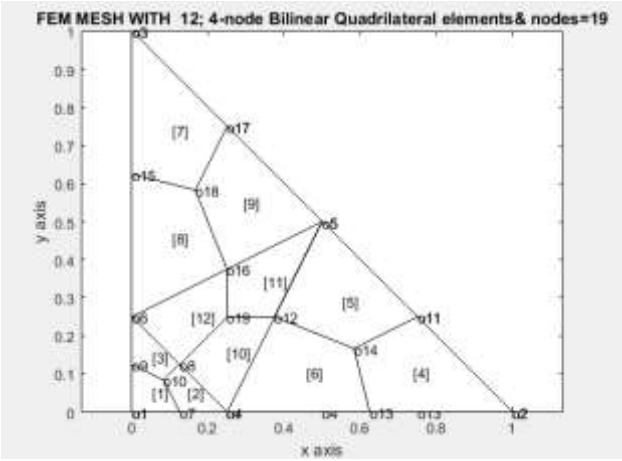
[25] Babuˇska, I., 1970. Finite element method for domains with corners. Computing (Arch. Elektron. Rechnen) 6, 264–273.

[26] F. Müller and C. Schwab,Finite Elements with mesh refinement for wave equations in polygons, Research Report No. 2013-11, April 2013 Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, CH-8092 Zürich Switzerland
[27] **Hengguang Li and Victor Nistor**, **LNG FEM: Graded Meshes on Domains of Polygonal Structures** Contemporary Mathematics Volume **586**, 2013, Pp239-246, http://dx.doi.org/10.1090/conm/586/11655
[28] Hiptmar Ralf and Schwab Christoph , Numerical Methods for Elliptic and Parabolic Boundary Value Problems Draft version December 5, 2008 **http://www.sam.math.ethz.ch/~hiptmair/tmp/NAPDE_08.pdf**

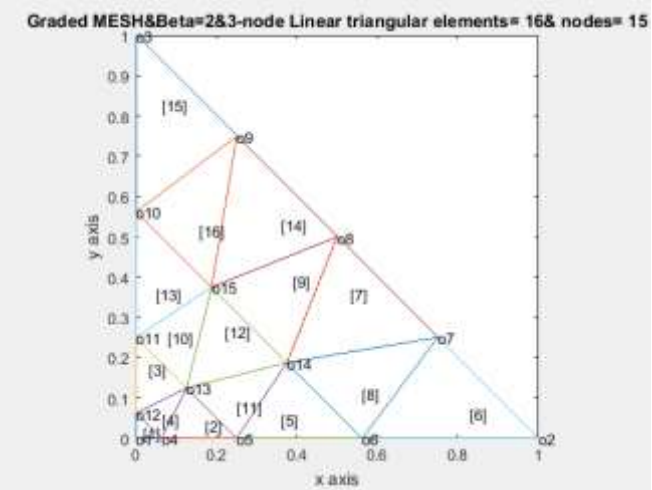
Mesh Generation Examples

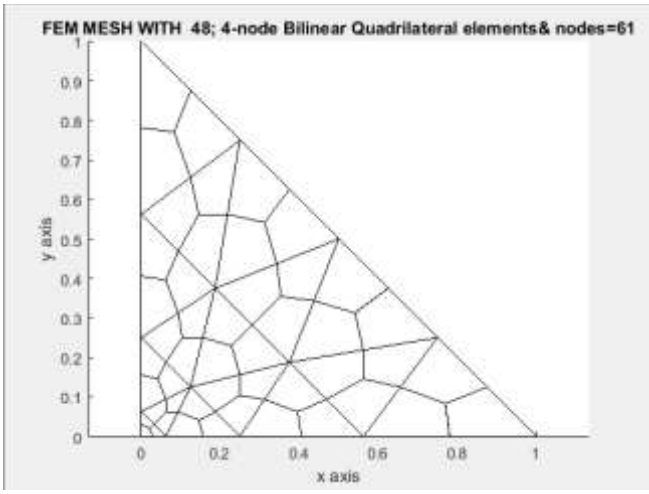
(I)FEM GRADED MESHES FOR CORNER SINGULARITY
FEM MESHES For STANDARD TRIANGLE
[1]NDIV=2;Graded FEM Meshes-β=2(For Standard Triangle)



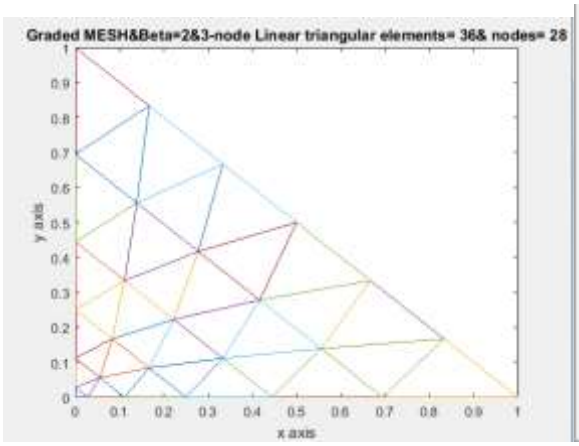


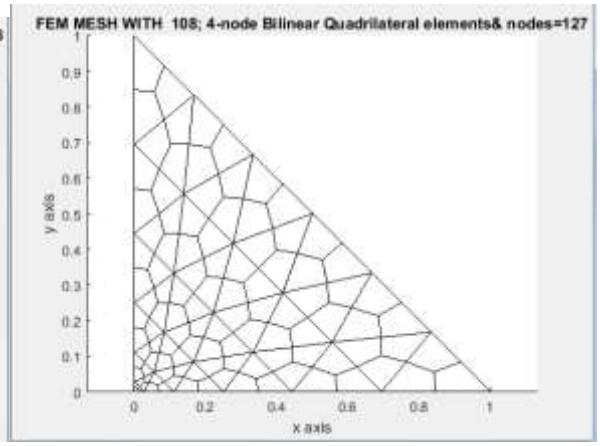
[2]NDIV=4;Graded FEM Meshes (For Standard Triangle)

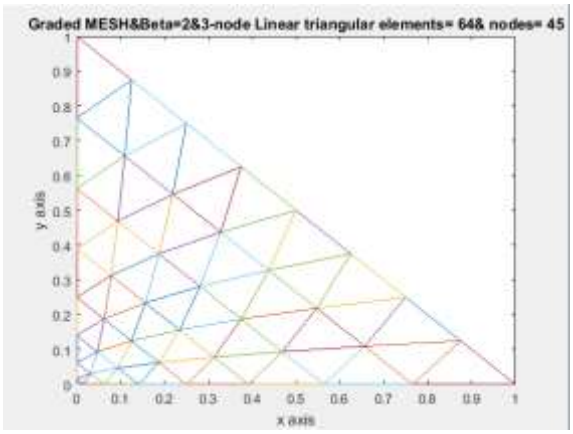




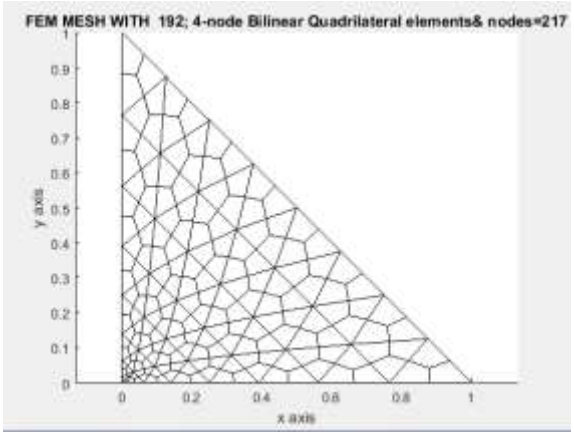
[3]NDIV=6; Graded FEM Meshes(For Standard Triangle)



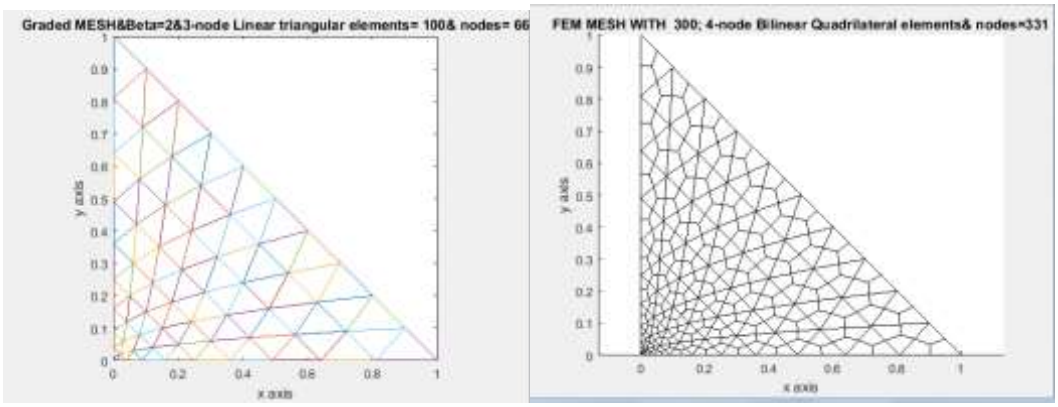




[4b]NDIV=8; Graded FEM Meshes(For Standard Triangle)

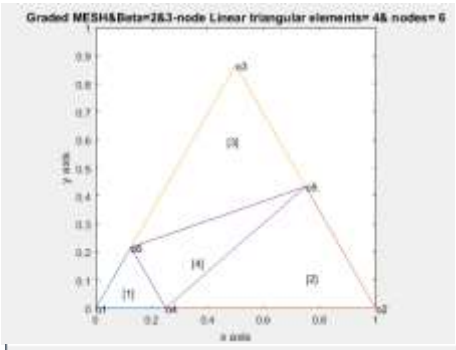


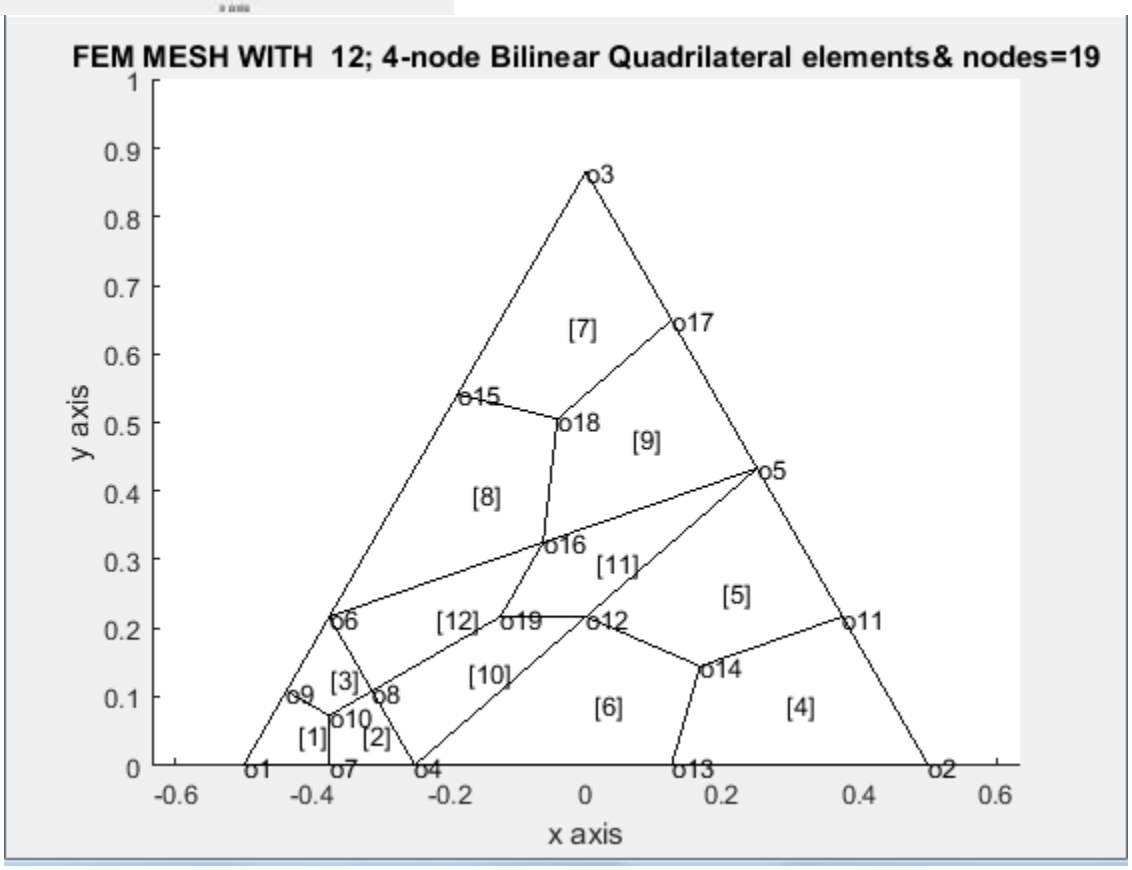
[5]NDIV=10; Graded FEM Meshes(For Standard Triangle)



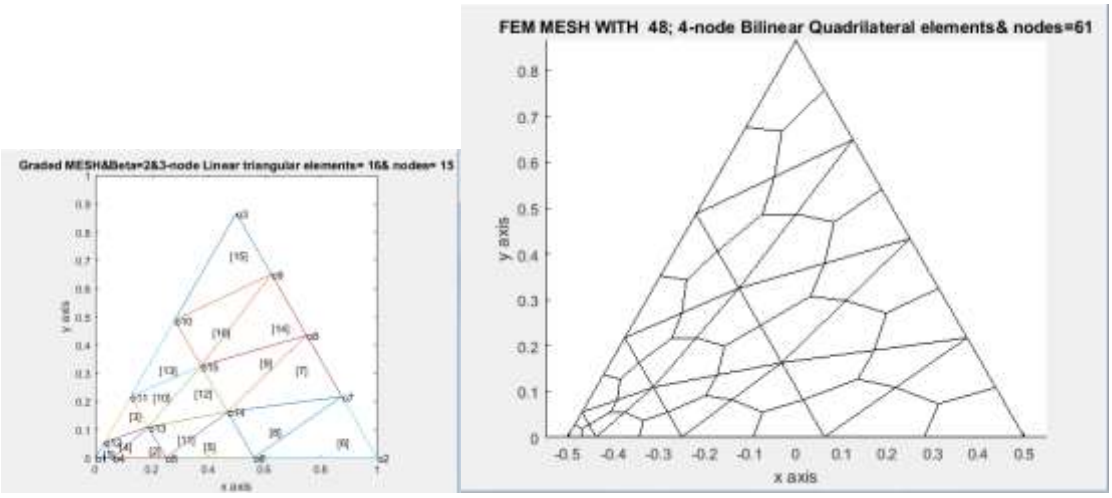
FEM MESHES For Equilateral Triangle

[1]NDIV=2;Graded FEM Meshes(For Equilateral Triangle)

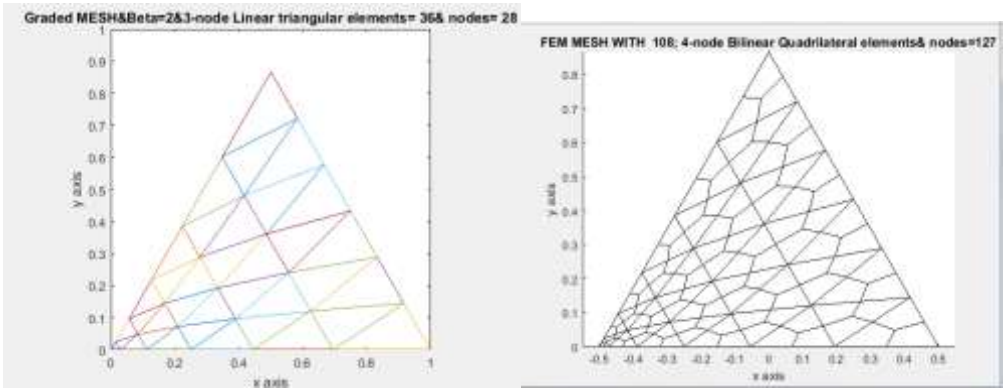




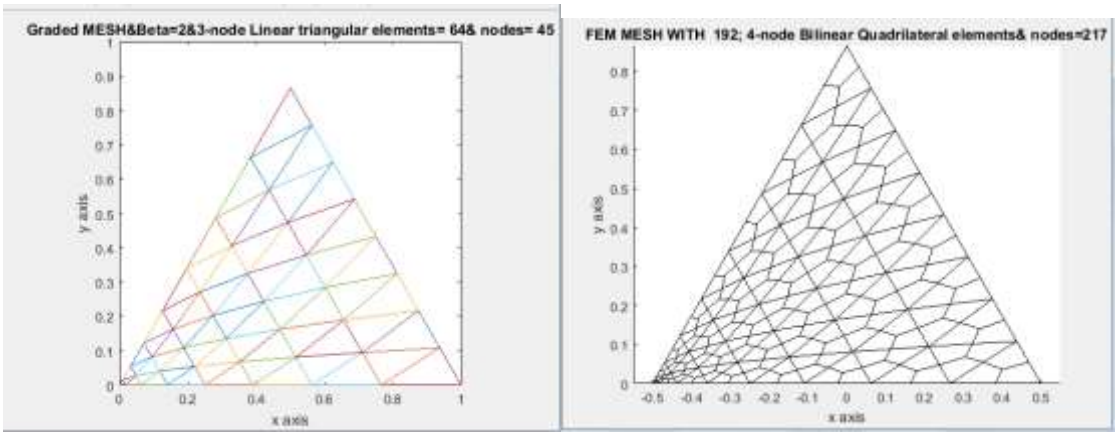
[2]NDIV=4;Graded FEM Meshes(For Equilateral Triangle)



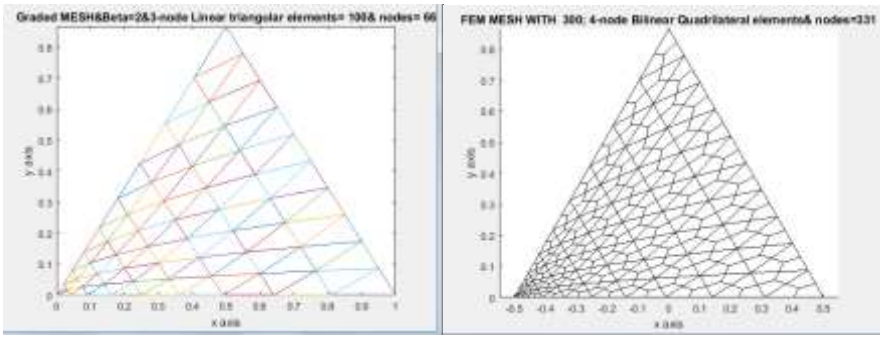
[3]NDIV=6;Graded FEM Meshes(For Equilateral Triangle)



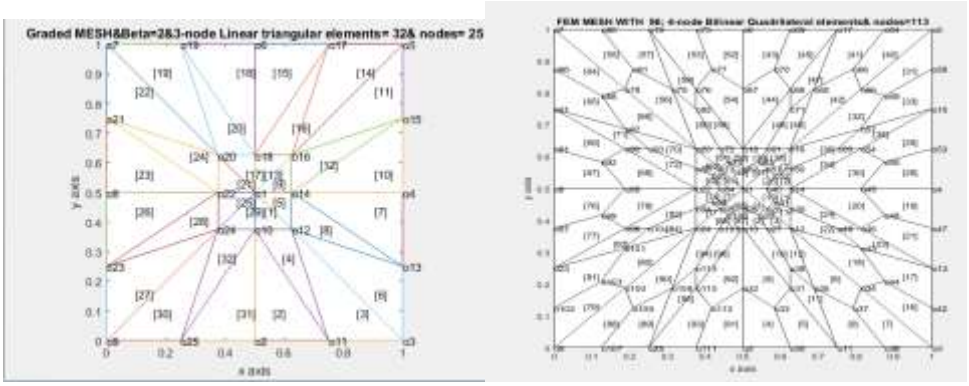
[4b]NDIV=8;Graded FEM Meshes(For Equilateral Triangle)



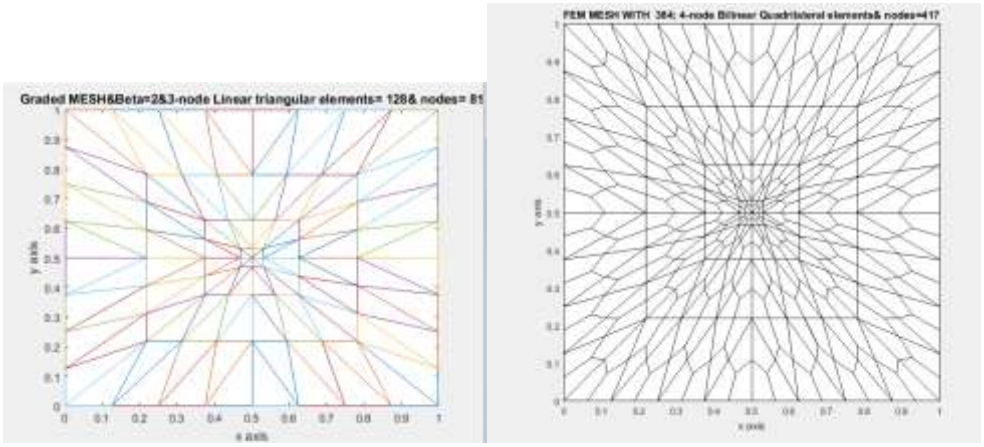
[5]NDIV=10;Graded FEM Meshes(For Equilateral Triangle)



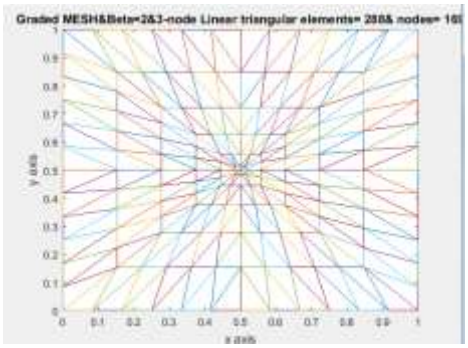
FEM GRADED MESHES β=2, 3-NODED LINEAR TRIANGLES:UNIT SQUARE,$0 \leq x, y \leq 1$
[1]NDIV=2

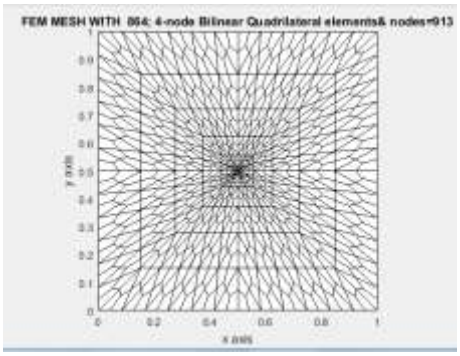


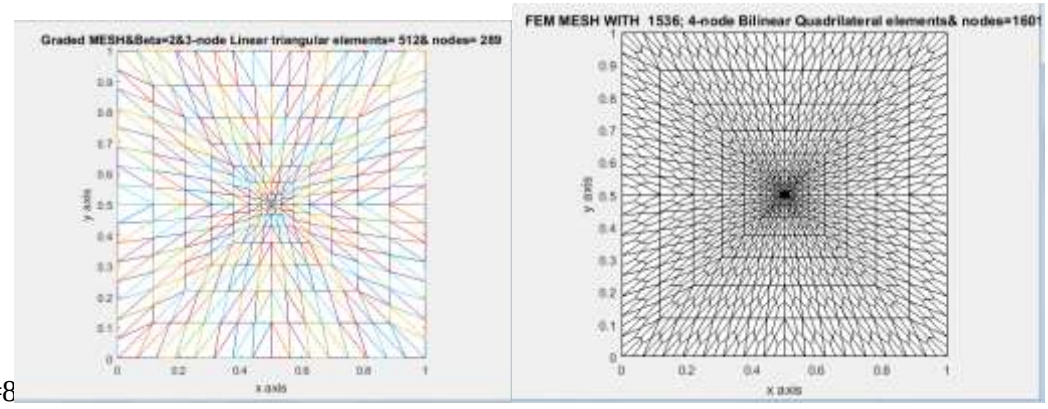
[2]NDIV=4



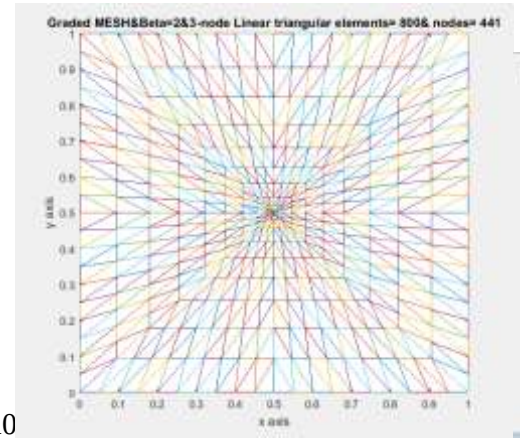
[3]NDIV=6

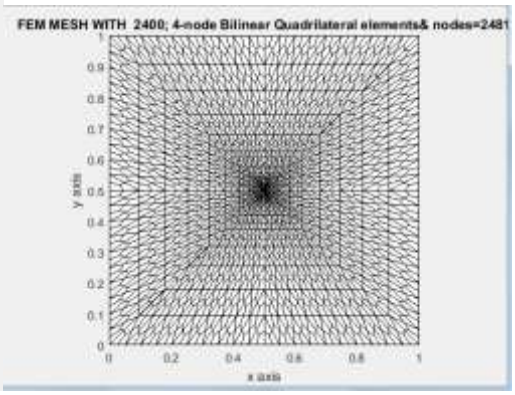






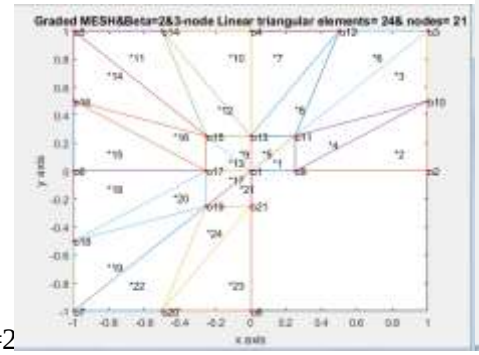
[4]NDIV=8

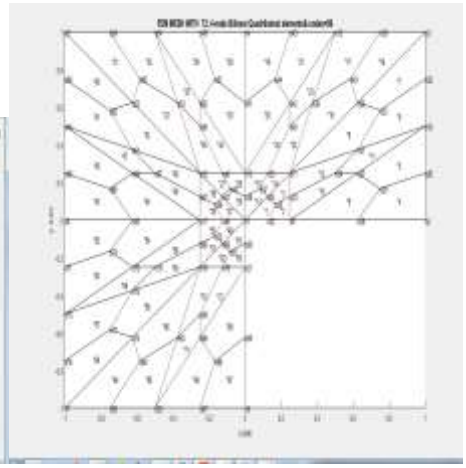




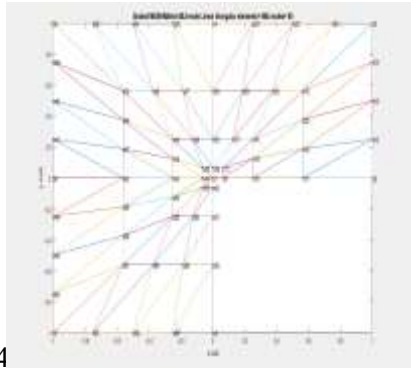
[5]NDIV=10

FEM GRADED MESHES β=2, 3-NODED LINEAR TRIANGLES:L-SHAPED DOMAIN

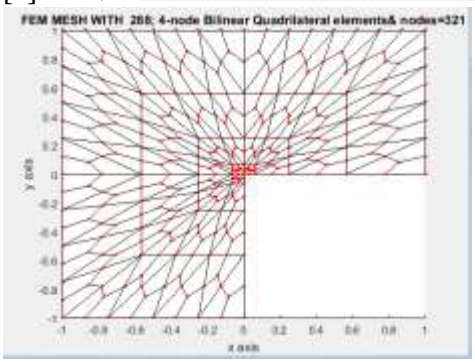


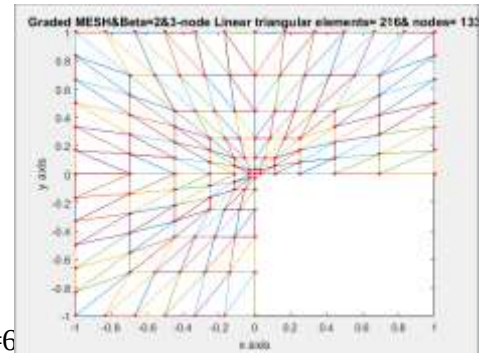


[1]NDIV=2

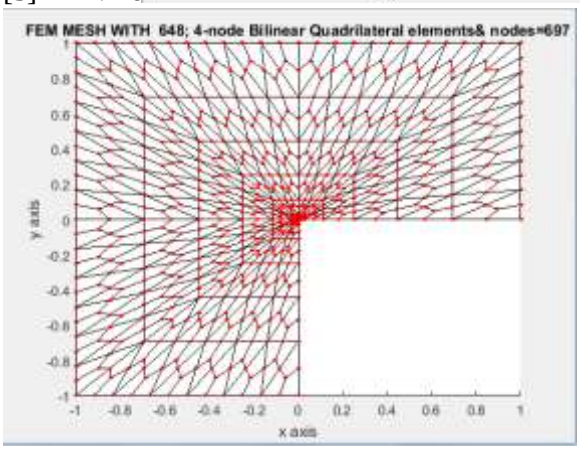


[2]NDIV=4

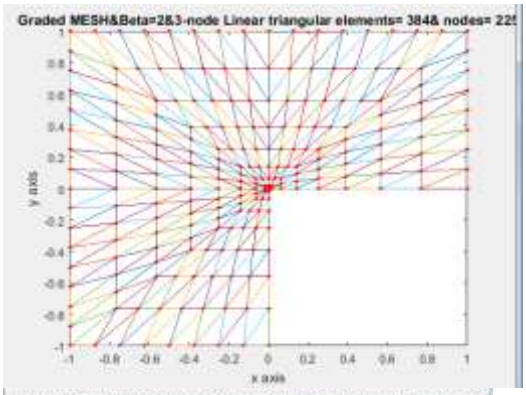


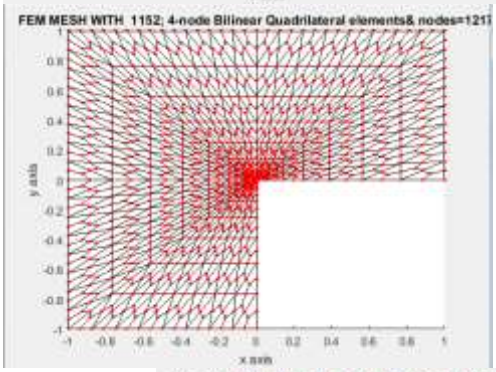


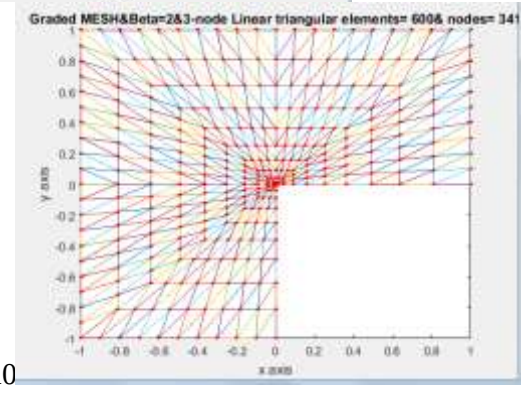
[3]NDIV=6



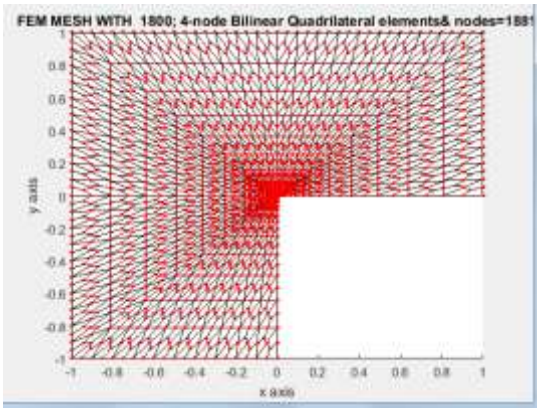
[4]NDIV=8



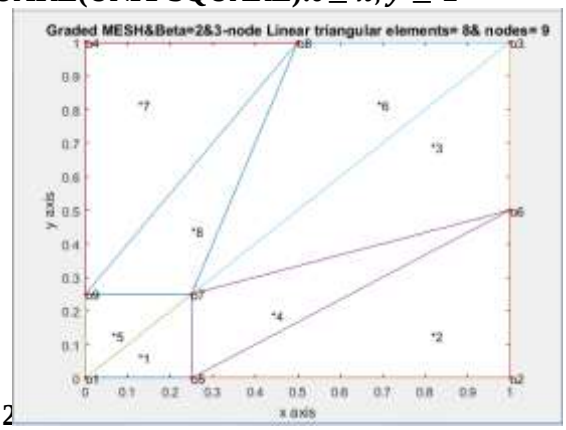




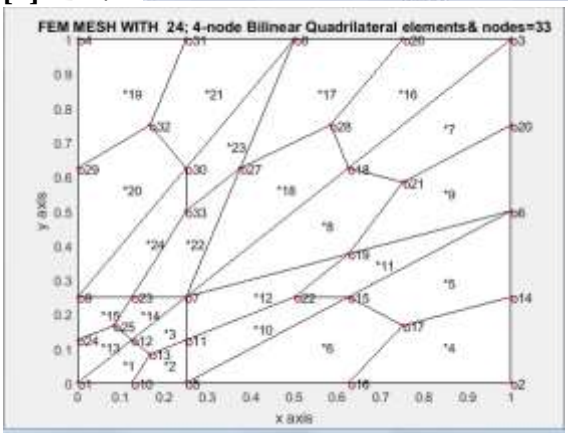
[5]NDIV=10

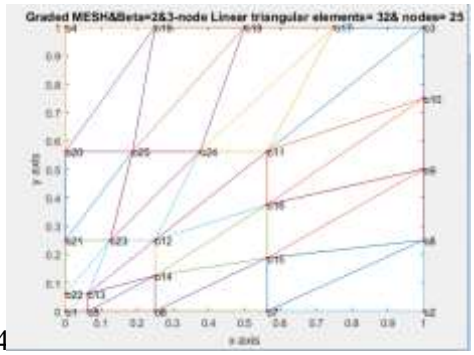


ONE SQUARE(UNIT SQUARE):$0 \leq x, y \leq 1$

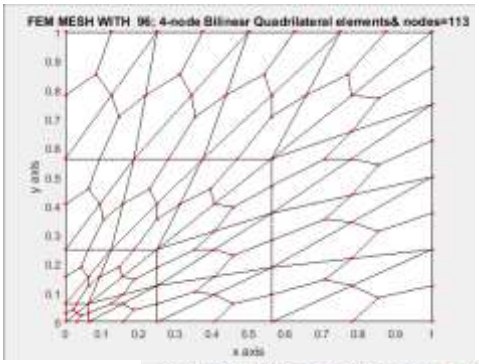


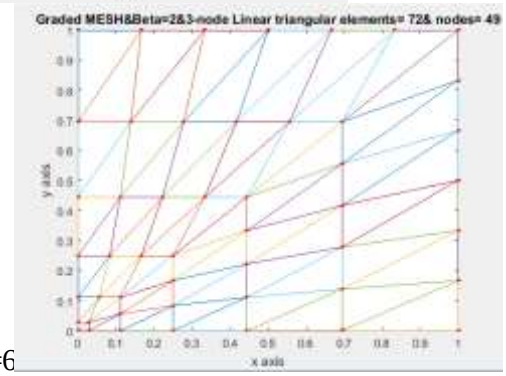
[1]NDIV=2



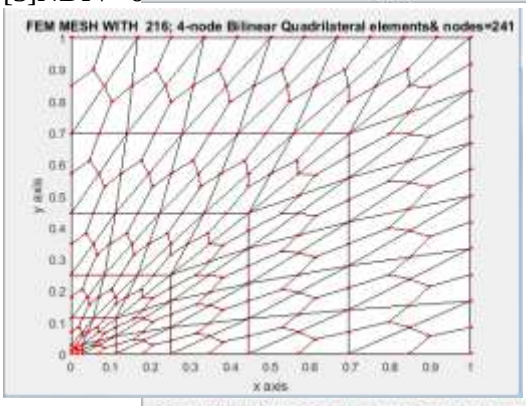


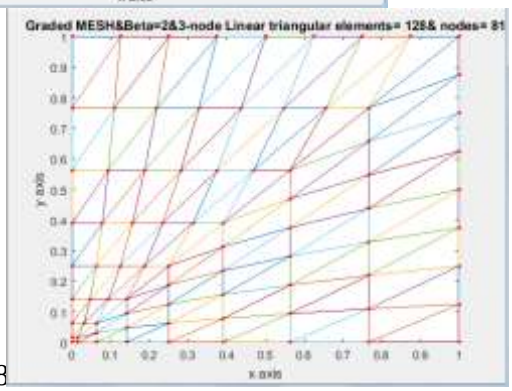
[2]NDIV=4



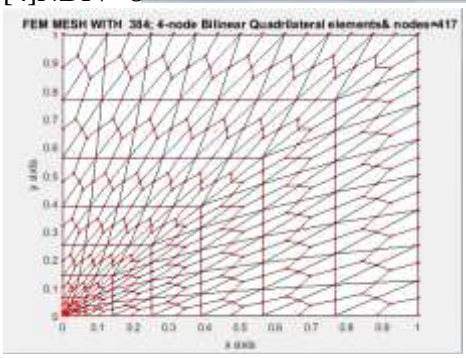


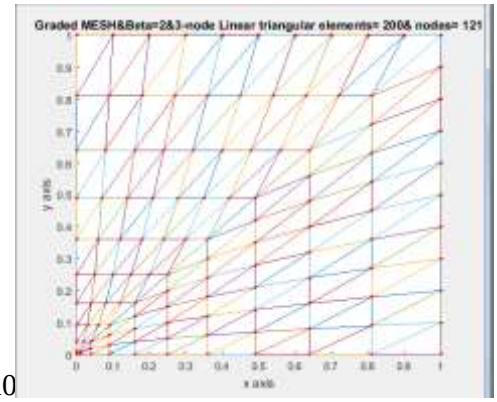
[3]NDIV=6



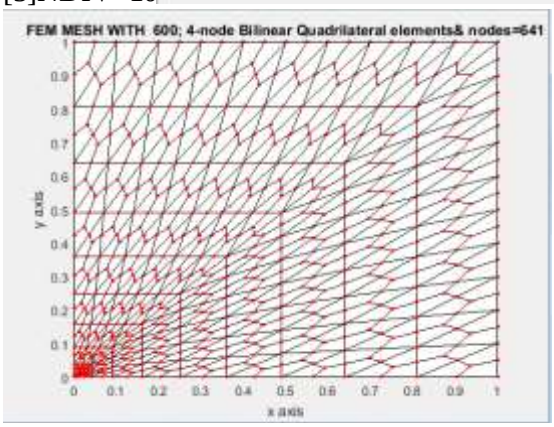


[4]NDIV=8



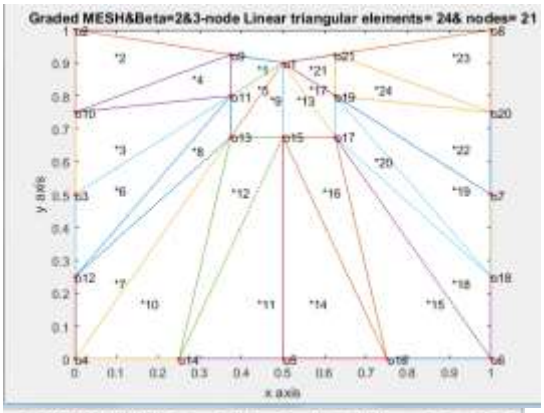


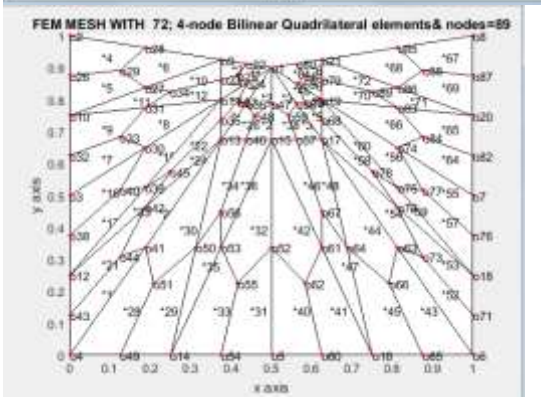
[5]NDIV=10

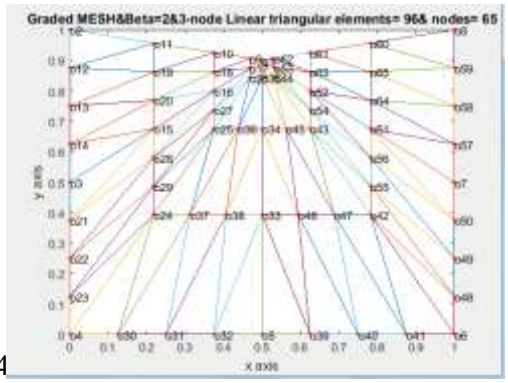


V-SHAPED GEOMETRY

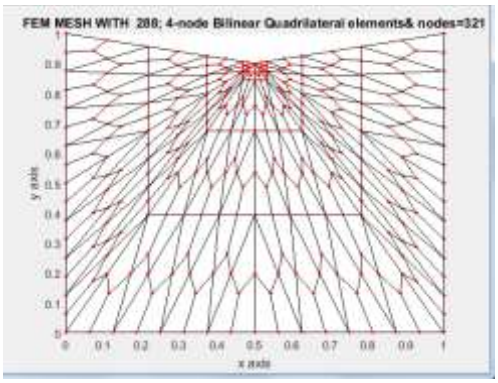
[1]NDIV=2 V-SHAPED GEOMETRY



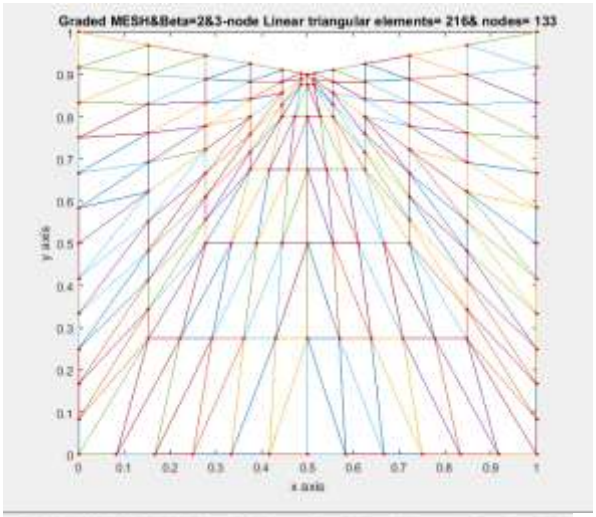


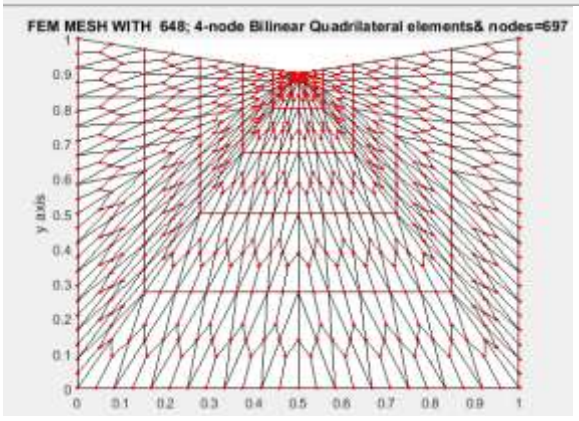


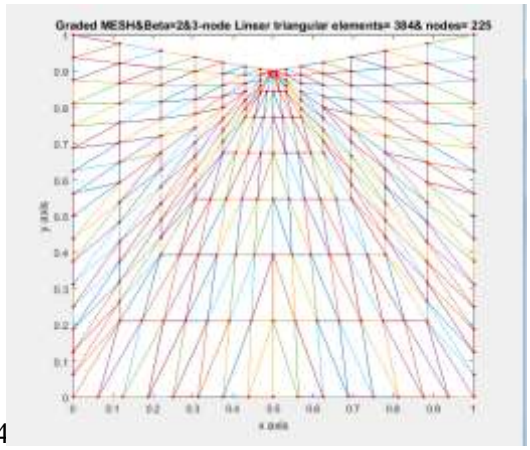
[2]NDIV=4



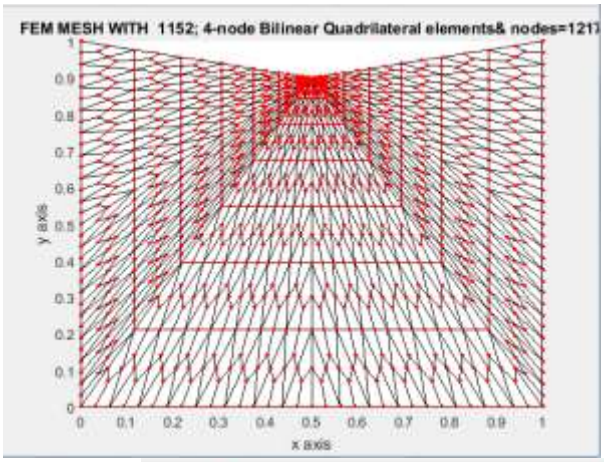
[3]NDIV=6

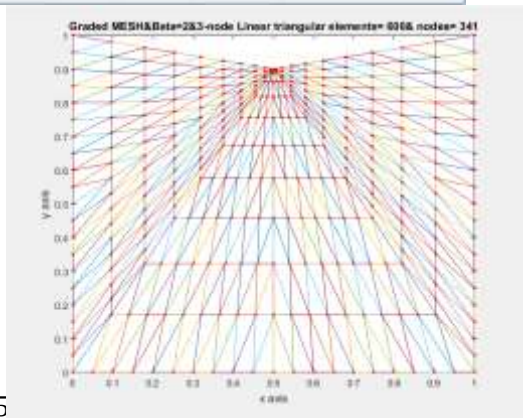




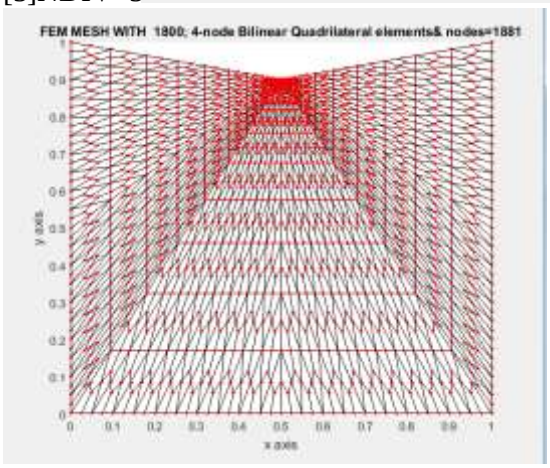


[4]NDIV=4



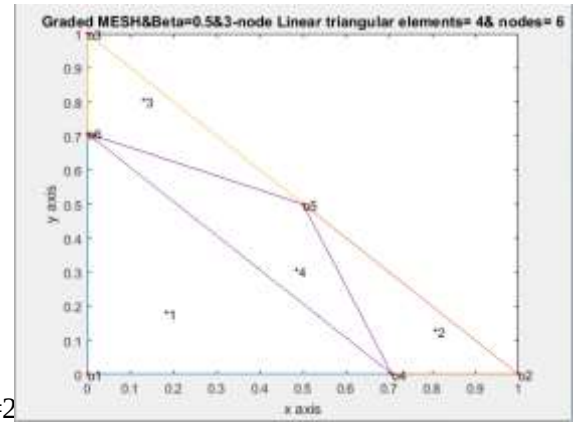


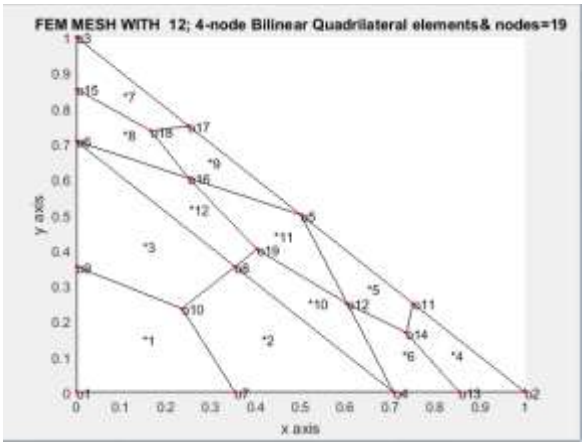
[5]NDIV=5



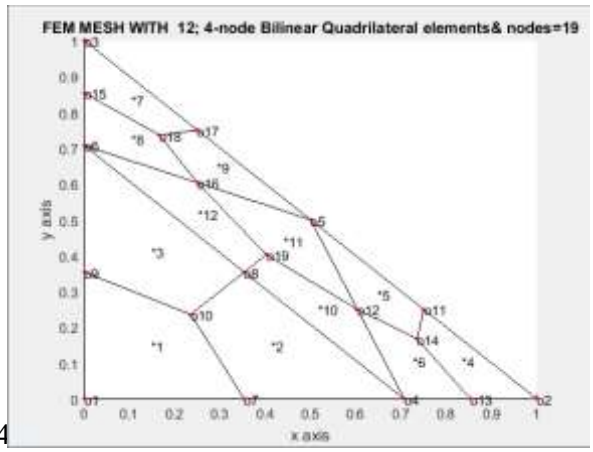
(II)FEM GRADED MESHES FOR SINGULAR EDGE:STANDARD TRIANGLE

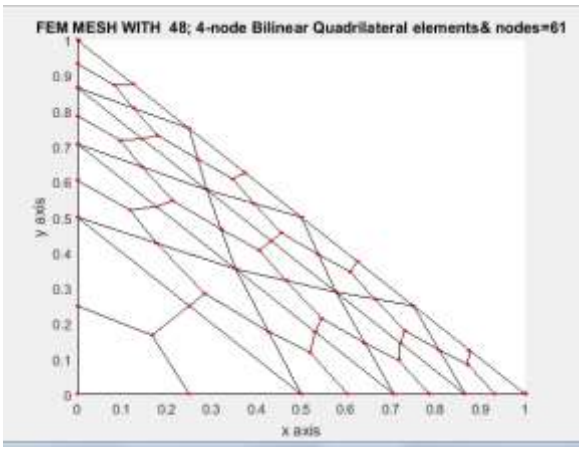
[1]NDIV=2

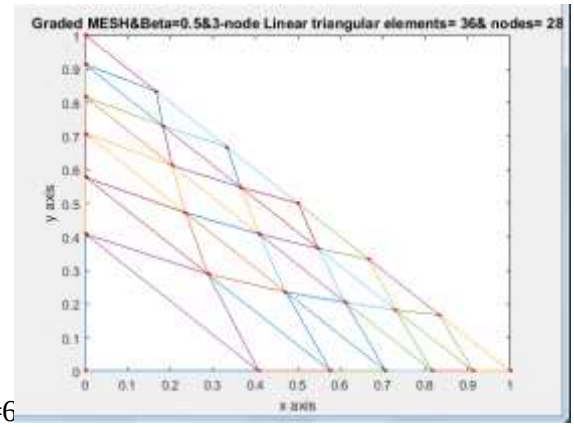




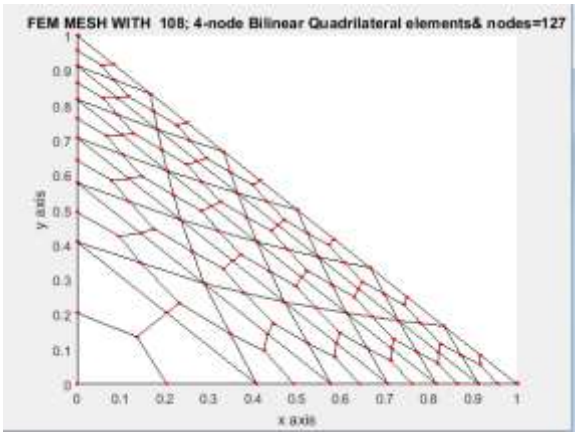
[2]NDIV=4

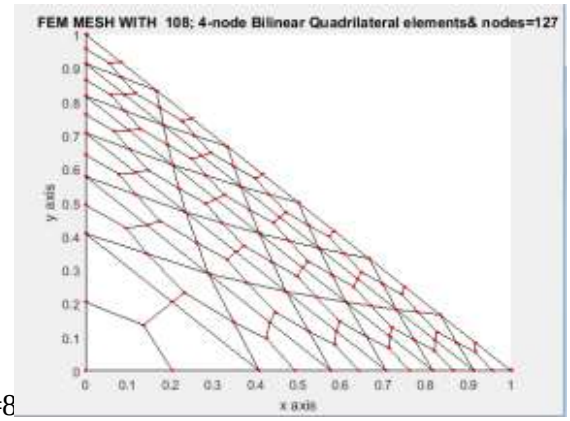




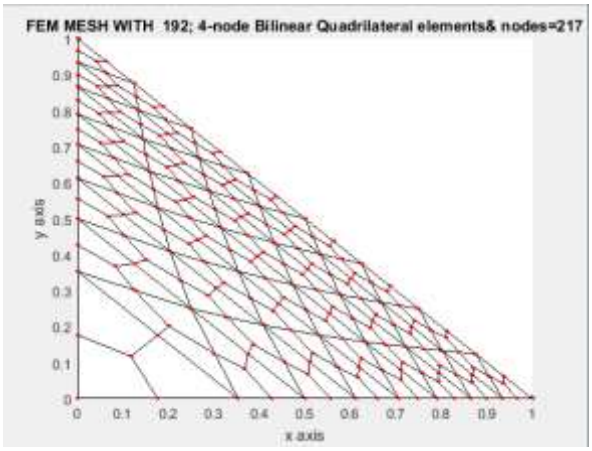


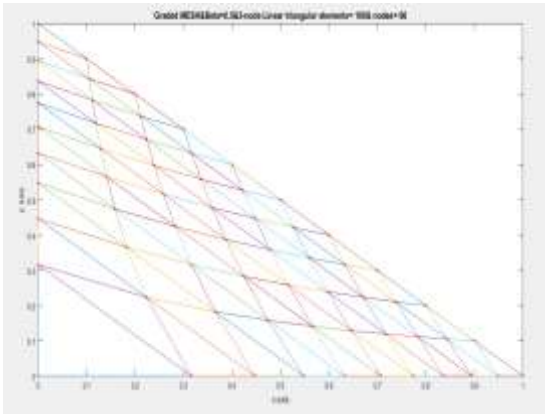
[3]NDIV=6

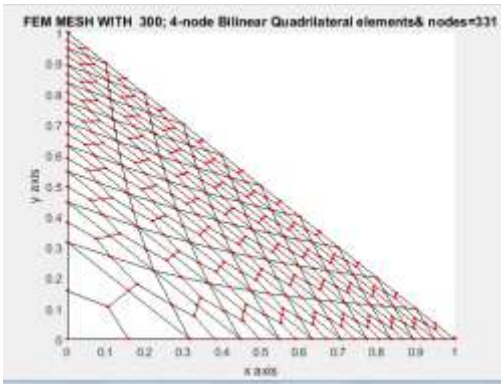


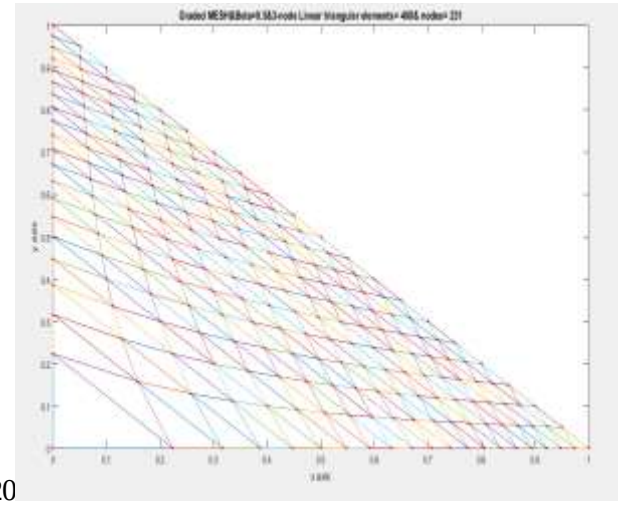


[4]NDIV=8

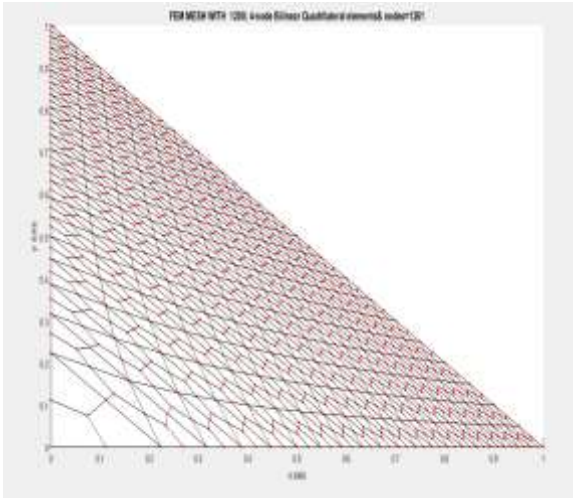




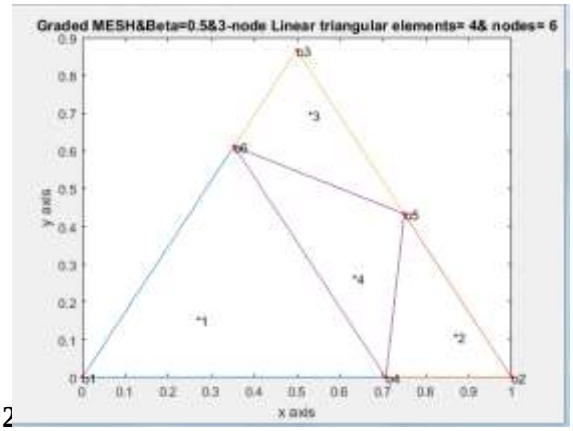




[6]NDIV=20

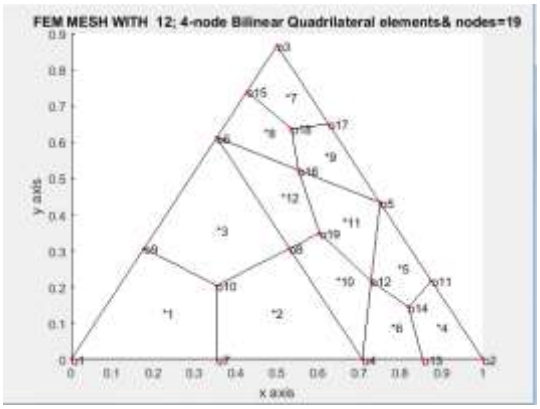


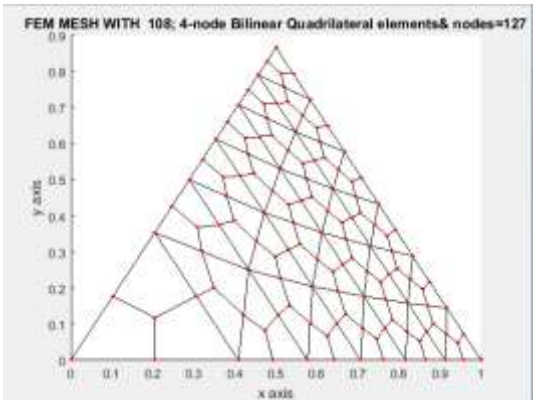
EQUILATERAL TRIANGLE

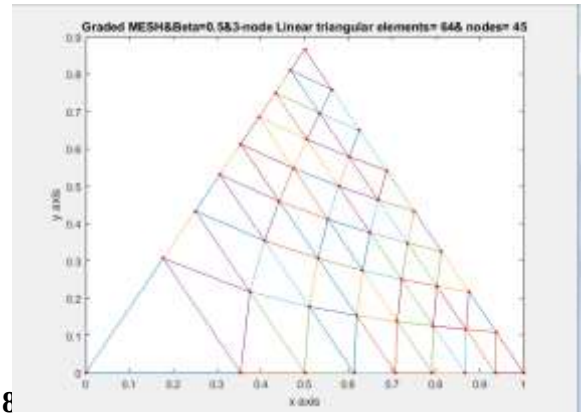


[1]NDIV=2

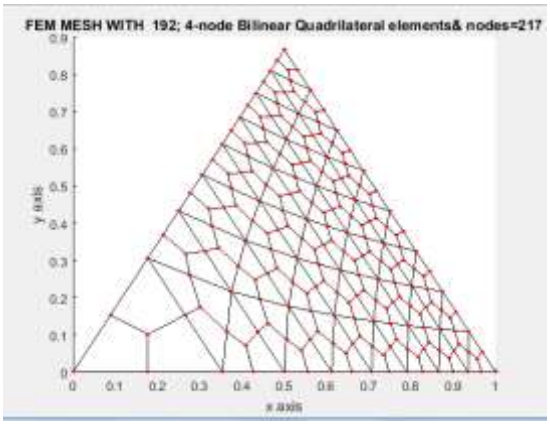
[3]NDIV=6

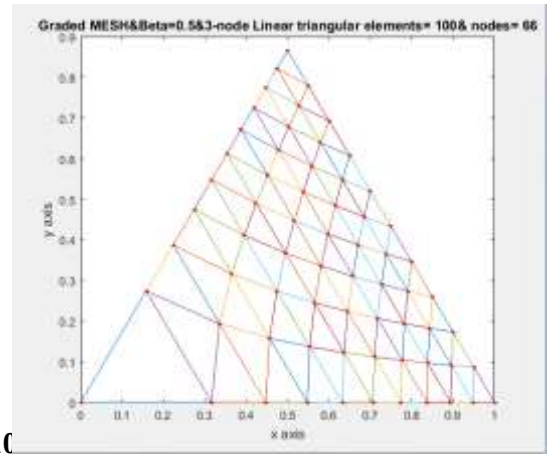




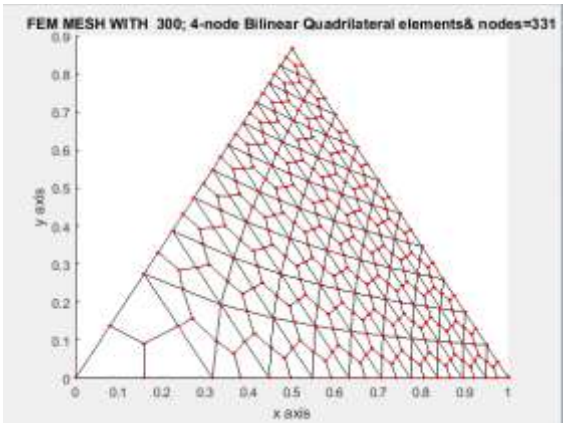


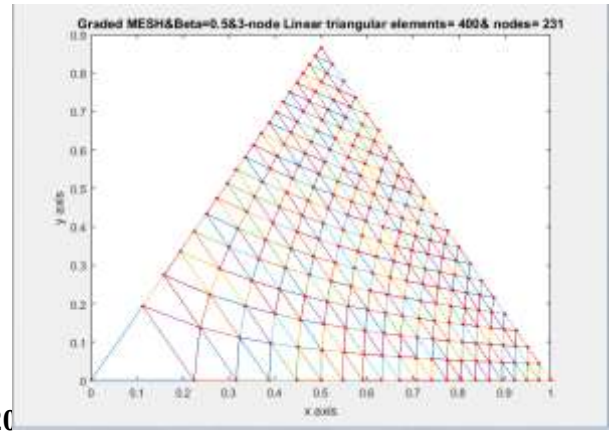
[4]NDIV=8



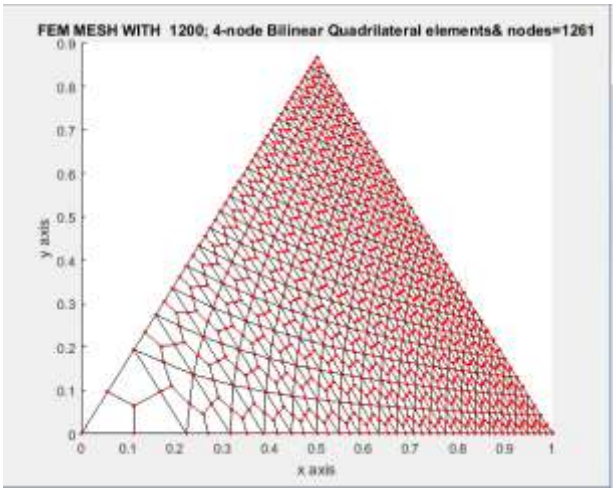


[5]NDIV=10



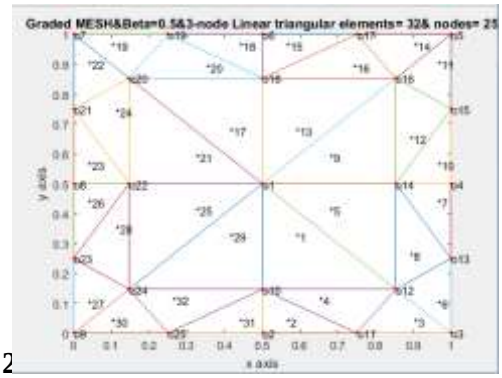


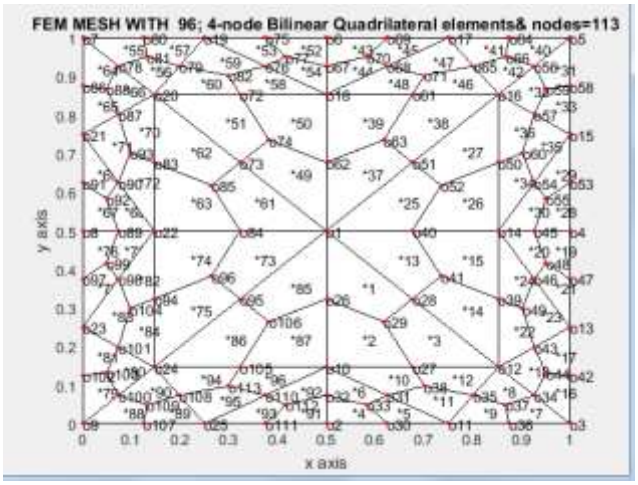
[6]NDIV=20



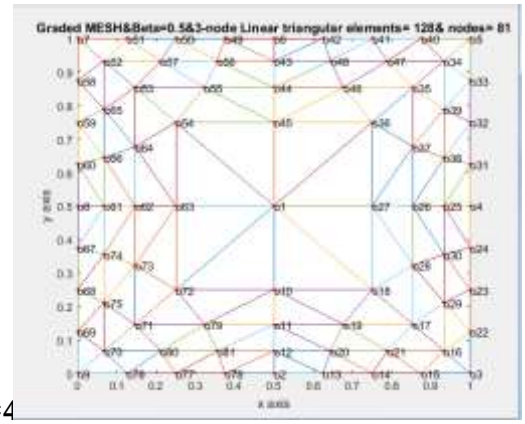
A 1-SQUARE

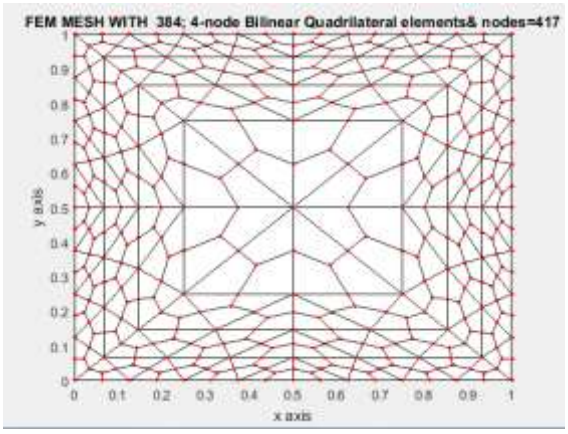
[1]NDIV=2

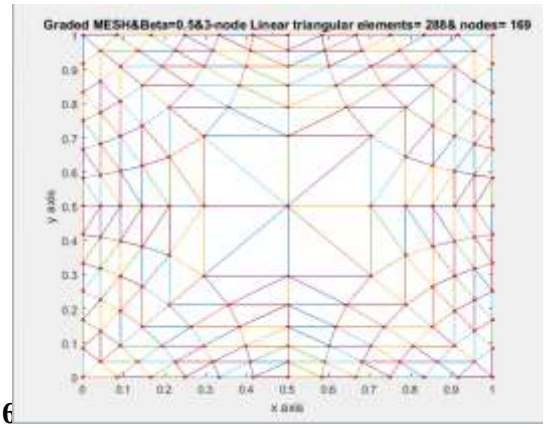




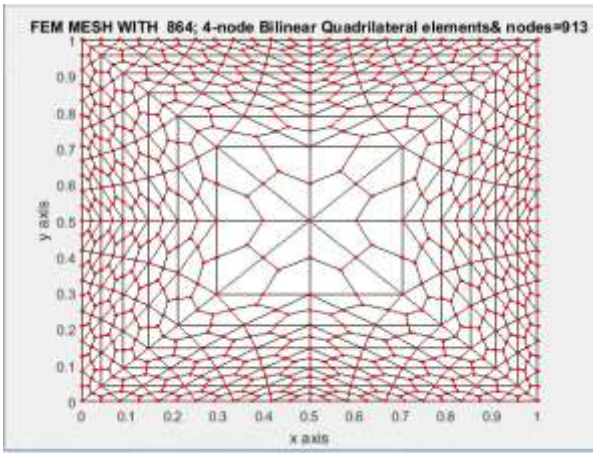
[2]NDIV=4

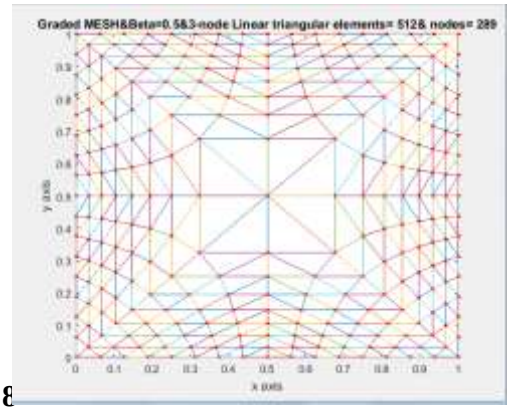




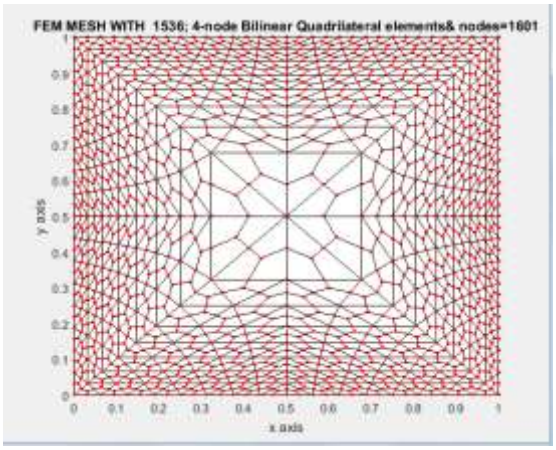


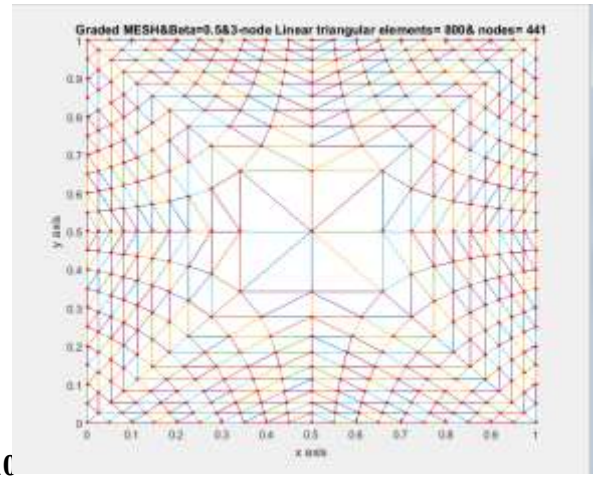
[3]NDIV=6



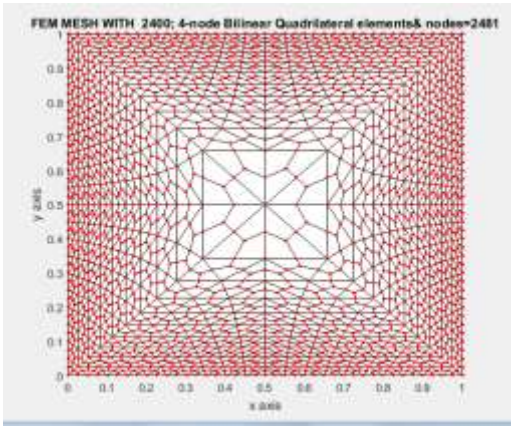


[4]NDIV=8





[5]NDIV=1(



COMPUTER PROGRAMS IN MATLAB

(1)triangular_mesh4LinearTrianglesSingularCorner_t3Nq4.m

```
function[]=triangular_mesh4LinearTrianglesSingularCorner_t3Nq4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete,beta
)
%
%maximumnumber of nodes on the actual domain
%
%clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
```

```
%type=0 for closed polygonal domain
%type=1 for open polygonal domain
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,9,6,1,1,1)
%triangular_mesh4LinearTriangles_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%triangular_mesh4LinearTriangles_t3([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,9,6,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
%quadrilateral_mesh4MOINEX_q4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,1,2,9,1,1)
%quadrilateral_mesh4MOINEX_q4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,4,4,9,1,1)
%quadrilateral_mesh4MOINEX_q4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,9,6,9,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,10,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,10,1,1)
%quadrilateral_mesh4MOINEX_q4([1],[2],[3],3,1,2,6,1,1)
%triangular_mesh4LinearTriangles_t3(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete)
%triangular_mesh4LinearTriangles_t3([1],[2],[3],3,1,2,6,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1],[2],[3],3,1,2,6,1,1,0)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,25,10,2,1,1)
%%quadrilateral_mesh4MOINEX_q4([1],[2],[3],3,1,2,17,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,9,6,21,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,9,6,22,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,23,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,25,10,24,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10],[3;4;5;6;7;8;9;10;2],10,1,2,25,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11],[3;4;5;6;7;8;9;10;11;2],11,1,2,26,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12],[3;4;5;6;7;8;9;10;11;12;2],12,9,6,27,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13],[3;4;5;6;7;8;9;10;11;12;13;2],13,9,6,2
8,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14],[3;4;5;6;7;8;9;10;11;12;13;14;2],
14,1,2,29,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15],[3;4;5;6;7;8;9;10;11;12;13;1
4;15;2],15,1,2,30,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16],[3;4;5;6;7;8;9;10;11;12
;13;14;15;16;2],16,1,2,31,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17],[3;4;5;6;7;8;9;10;
11;12;13;14;15;16;17;2],17,1,2,32,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18],[3;4;5;6;7;8;
9;10;11;12;13;14;15;16;17;18;2],18,1,2,33,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18],[3;4;5;6;7;8;
9;10;11;12;13;14;15;16;17;18;2],18,1,2,33,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19],[3;4;5;6
;7;8;9;10;11;12;13;14;15;16;17;18;19;2],19,1,2,34,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20],[3;
4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;2],20,1,2,35,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;2
1],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;2],21,1,2,36,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,51,1,1)
%triangular_mesh4LinearTriangles_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%triangular_mesh4LinearTriangles_t3([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,1,2,54,2,2)
%triangular_mesh4LinearTriangles_t3([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,1,2,55,4,2)
%triangular_mesh4LinearTriangles_t3([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,56,2,2,0)
%triangular_mesh4LinearTriangles_t3([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,56,2,2,2)
%triangular_mesh4LinearTriangles_t3([5;5;5;5],[1;2;3;4],[2;3;4;1],5,1,2,11,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;1
5;16;17;18;19;20;21],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;2],21,1,2,36,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,56,2,2,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([5;5;5;5],[1;2;3;4],[2;3;4;1],5,1,2,11,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,1,2,55,4,2,0)
%%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,1,2,54,2,2,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10],[3;4;5;6;7;8;9;10;2],10,1,2,25,1,1,0)
```

```
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,0)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,56,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1],[2],[3],3,1,2,6,1,1,0,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1],[2],[3],3,1,2,6,1,1,0,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,0,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,0,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,1,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,57,1,1,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,57,1,1,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,1,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,3,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,4,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,59,2,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,59,2,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,(ndiv/2)^2,ndiv,5
9,2,2,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1],[2],[3],3,1,2,6,1,1,0,1/2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1],[2],[3],3,1,2,7,1,1,0,1/2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1,0,1/2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,58,2,2,6,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,60,1,1,2,2)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4([5;5;5;5],[1;2;3;4],[2;3;4;1],5,1,2,11,1,1,0,1)
%triangular_mesh4LinearTrianglesSingularCorner_t3Nq4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete,beta)
[nitri,cl]=size(n1)
if(ndelete==0)&(nitri>1)
   nmax=nitri+1
end
if(ndelete==0)&(nitri==1)
   nmax=3
end
%[coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
[eln,trielm,rrr,coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesSingularCorner(n1,n2,n3,nma
x,numtri,ndiv,mesh,beta)
[nel,nnel]=size(tnodes);

disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where  i->node no. and j->x or y
%_____________________________________________________________
%
%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract  coordinates for each element
%clf
figure(2*(ndiv/2)-1)
NDEL=ndelete*4*numtri
NEL=nel-NDEL;
NNODE=max(max(tnodes(1:NEL,1:3)));
NNNODE=NNODE
if (ndelete>1)
NNNODE=NNODE-ndelete+1
end

for i=1:NEL
for j=1:nnel
x(1,j)=xcoord(tnodes(i,j),1);
y(1,j)=ycoord(tnodes(i,j),1);
end;%j loop
xvec(1,1:4)=[x(1,1),x(1,2),x(1,3),x(1,1)];
yvec(1,1:4)=[y(1,1),y(1,2),y(1,3),y(1,1)];
```

```
%axis  equal
axis  normal
rtext=0;
switch mesh
case 1
axis([0  xlength  0 ylength])
case 2
axis([0  xlength  0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl   xl  -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl   xl  -yl yl])
case 12
   axis([-xlength   xlength  -ylength ylength])

case 17
   axis([-xlength   xlength  0 ylength])
    rpoly=' one isosceles triangle in a  square';
   rtext=1;
case 18
   axis([-2*xlength   2*xlength  -ylength 2*ylength])
   rpoly='   equilateral triangle';
   rtext=1;
case 19
   axis([-xlength   xlength  -ylength/2 ylength])
    rpoly='   equilateral triangle';
   rtext=1;

 case 20
   axis([-xlength   xlength  -ylength ylength])
    rpoly='    square';
   rtext=1;
 case 21
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     pentagon';
   rtext=1;
 case 22
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     hexagon';
   rtext=1;
  case 23
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     heptagon';
   rtext=1;
   case 24
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     octagon';
   rtext=1;
    case 25
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     nonadecagon';
   rtext=1;
    case 26
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     decagon';
    rtext=1;
    case 27
   axis([-xlength   xlength  -ylength ylength])
    rpoly='     hendecagon';
   rtext=1;
    case 28
   axis([-xlength   xlength  -ylength ylength])
```

```
    rpoly='     dodecagon';
   rtext=1;
    case 29
   axis([-xlength  xlength -ylength ylength])
    rpoly='     tridecagon';
   rtext=1;
    case 30
   axis([-xlength  xlength -ylength ylength])
    rpoly='     tetradecagon';
   rtext=1;
    case 31
   axis([-xlength  xlength -ylength ylength])
    rpoly='     pentadecagon';
   rtext=1;
     case 32
   axis([-xlength  xlength -ylength ylength])
    rpoly='     hexadecagon';
   rtext=1;
     case 33
   axis([-xlength  xlength -ylength ylength])
    rpoly='     heptadecagon';
   rtext=1;
    case 34
   axis([-xlength  xlength -ylength ylength])
   rpoly='     octadecagon';
   rtext=1;
    case 35
   axis([-xlength  xlength -ylength ylength])
   rpoly='     enneadecagon';
   rtext=1;
  case 36
   axis([-xlength  xlength -ylength ylength])
   rpoly='     icosagon';
   rtext=1;

end
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<4
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['*',num2str(i)]);
end
end;%i loop

xlabel('x  axis')
ylabel('y axis')
st1='Graded  MESH&Beta=';
st2=num2str(beta);
st3='&3-node Linear triangular ';
st4='elements= ';
st5=num2str(NEL);
st6='& nodes='
st7=num2str(NNNODE);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
%if ndiv<4
for jj=1:NNODE
 if (ndelete>1)&(jj<=(nmax-ndelete+1))
 %text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
 GCOORD(jj,1:2)=gcoord(jj,1:2);
 end
```

```
 if (ndelete>1)&(jj>=(nmax+1))
% text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj-ndelete+1)]);
 GCOORD(jj-ndelete+1,1:2)=gcoord(jj,1:2);
 end
if(ndelete==0|(ndelete==1))
%text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
GCOORD(jj,1:2)=gcoord(jj,1:2);
end
end
%end




if rtext==1
  text(0.1,-1.2,rpoly)
end
%axis  off
NEL
NNNODE
GCOORD
%element  nodal connectivity matrix
if (ndelete>1)
nnmax=nmax-(ndelete-1)
for iel=1:NEL
    for jel=1:3
      if(tnodes(iel,jel)>nnmax)
         tnodes(iel,jel)=tnodes(iel,jel)-(ndelete-1);
      end
    end
end
end
tnodes;
NEL
[GROW,GCLM]=size(GCOORD)
[TROW,TCLM]=size(tnodes)
ELROW(:,1)=(1:NEL)';
[ELROW   tnodes(1:NEL,:)]
XCOORD(:,1)=GCOORD(:,1);
YCOORD(:,1)=GCOORD(:,2);
NGROW(:,1)=(1:GROW)';
[NGROW   XCOORD   YCOORD]
hold on
%TNODES(:,1:3)=tnodes(:,1:3);
if ndiv<=4
for kk=1:NNNODE
   text(GCOORD(kk,1),GCOORD(kk,2),['o',num2str(kk)]);
end
end
hold on
figure(2*(ndiv/2)-1),scatter(XCOORD,YCOORD,10,'filled','r')
hold off
xi(:,1)=GCOORD(:,1);
yi(:,1)=GCOORD(:,2);
ne=NEL;
nnd=NNNODE;
for  iii=1:ne
   for jjj=4:7
      tnodes(iii,jjj)=0;
   end
end
for inum=1:nnd
   for jnum=1:nnd
```

```
        mdpt(inum,jnum)=0;mdpt(jnum,inum)=0;
    end
end
nd=nnd;
for nnn=1:ne
    mmm1=tnodes(nnn,1);
    mmm2=tnodes(nnn,2);
    mmm3=tnodes(nnn,3);
%     xi(mmm1,1)=XCOORD(mmm1,1);xi(mmm2,1)=XCOORD(mmm2,1);xi(mmm3,1)=XCOORD(mmm3,1);
%     yi(mmm1,1)=YCOORD(mmm1,1);yi(mmm2,1)=YCOORD(mmm2,1);yi(mmm3,1)=YCOORD(mmm3,1);
%midpoint side-1 of 3-node triangle
if((mdpt(mmm1,mmm2)==0)&(mdpt(mmm2,mmm1)==0))
    nd=nd+1;
   mdpt(mmm1,mmm2)=nd;
   mdpt(mmm2,mmm1)=nd;
   mmm4=nd;
   xi(mmm4,1)=(xi(mmm1,1)+xi(mmm2,1))/2;
   yi(mmm4,1)=(yi(mmm1,1)+yi(mmm2,1))/2;
end
%midpoint side-2 of 3-node triangle
if((mdpt(mmm2,mmm3)==0)&(mdpt(mmm3,mmm2)==0))
    nd=nd+1;
   mdpt(mmm2,mmm3)=nd;
   mdpt(mmm3,mmm2)=nd;
   mmm5=nd;
   xi(mmm5,1)=(xi(mmm2,1)+xi(mmm3,1))/2;
   yi(mmm5,1)=(yi(mmm2,1)+yi(mmm3,1))/2;
end
%midpoint side-3 of 3-node triangle
if((mdpt(mmm3,mmm1)==0)&(mdpt(mmm1,mmm3)==0))
    nd=nd+1;
   mdpt(mmm3,mmm1)=nd;
   mdpt(mmm1,mmm3)=nd;
   mmm6=nd;
   xi(mmm6,1)=(xi(mmm1,1)+xi(mmm3,1))/2;
   yi(mmm6,1)=(yi(mmm1,1)+yi(mmm3,1))/2;
end
nd=nd+1;
mmm7=nd;
   tnodes(nnn,4)=mdpt(mmm1,mmm2);
   tnodes(nnn,5)=mdpt(mmm2,mmm3);
   tnodes(nnn,6)=mdpt(mmm3,mmm1);
   tnodes(nnn,7)=mmm7;
   xi(mmm7,1)=(xi(mmm1,1)+xi(mmm2,1)+xi(mmm3,1))/3;
   yi(mmm7,1)=(yi(mmm1,1)+yi(mmm2,1)+yi(mmm3,1))/3;
end
tnodes

%
%to generate special quadrilaterals
mm=0;

for iel=1:ne
   for jel=1:3
   mm=mm+1;
      switch jel
       case 1
        spqd(mm,1:4)=[tnodes(iel,7)   tnodes(iel,6) tnodes(iel,1) tnodes(iel,4)];
        nodes(mm,1:4)=spqd(mm,1:4);
        nodetel(mm,1:3)=[tnodes(iel,2)   tnodes(iel,3) tnodes(iel,1)];
        case 2
        spqd(mm,1:4)=[tnodes(iel,7)   tnodes(iel,4) tnodes(iel,2) tnodes(iel,5)];
        nodes(mm,1:4)=spqd(mm,1:4);
        nodetel(mm,1:3)=[tnodes(iel,3)   tnodes(iel,1) tnodes(iel,2)];
```

```
        case 3
        spqd(mm,1:4)=[tnodes(iel,7)   tnodes(iel,5) tnodes(iel,3) tnodes(iel,6)];
        nodes(mm,1:4)=spqd(mm,1:4);
        nodetel(mm,1:3)=[tnodes(iel,1)   tnodes(iel,2) tnodes(iel,3)];
      end%switch
    end
  end

for mmm=1:mm
   spqd(:,1:4)
end
%%*************************************************************************************

[nel,nnel]=size(nodes);

disp([xlength,ylength,nnode,nel,nnel])

figure(ndiv)
NNODE=max(max(nodes(1:nel,1:4)));
NNNODE=NNODE
nnode=max(max(tnodes));

N=(1:nnode)'
[N xi  yi]
%
%coord(:,1)=(xi(:,1));
%coord(:,2)=(yi(:,1));
 xcoords(:,1)=double(xi(:,1));
 ycoords(:,1)=double(yi(:,1));
 gcoords(:,1)=xcoords(:,1);
 gcoords(:,2)=ycoords(:,1);
%disp(gcoords)


for i=1:3:nel
for j=1:nnel
x(1,j)=xi(nodes(i,j),1);
y(1,j)=yi(nodes(i,j),1);
%
x(2,j)=xi(nodes(i+1,j),1);
y(2,j)=yi(nodes(i+1,j),1);
%
x(3,j)=xi(nodes(i+2,j),1);
y(3,j)=yi(nodes(i+2,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
%
xvec(2,1:5)=[x(2,1),x(2,2),x(2,3),x(2,4),x(2,1)];
yvec(2,1:5)=[y(2,1),y(2,2),y(2,3),y(2,4),y(2,1)];
%
xvec(3,1:5)=[x(3,1),x(3,2),x(3,3),x(3,4),x(3,1)];
yvec(3,1:5)=[y(3,1),y(3,2),y(3,3),y(3,4),y(3,1)];
axis  normal
%axis  tight
rtext=0;
switch mesh
case 1
axis([0  xlength  0 ylength])
case 2
axis([0  xlength  0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
```

```
axis([-xl  xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl  xl -yl yl])
case 12
   axis([-xlength  xlength -ylength ylength])

case 17
   axis([-xlength  xlength 0 ylength])
    rpoly=' one isosceles triangle in a  square';
   rtext=1;
case 18
   axis([-2*xlength  2*xlength  -ylength 2*ylength])
   rpoly='   equilateral triangle';
   rtext=1;
case 19
   axis([-xlength  xlength -ylength/2 ylength])
    rpoly='   equilateral triangle';
   rtext=1;

 case 20
   axis([-xlength  xlength -ylength ylength])
    rpoly='    square';
   rtext=1;
 case 21
   axis([-xlength  xlength -ylength ylength])
    rpoly='    pentagon';
   rtext=1;
 case 22
   axis([-xlength  xlength -ylength ylength])
    rpoly='    hexagon';
   rtext=1;
  case 23
   axis([-xlength  xlength -ylength ylength])
    rpoly='    heptagon';
   rtext=1;
   case 24
   axis([-xlength  xlength -ylength ylength])
    rpoly='    octagon';
   rtext=1;
    case 25
   axis([-xlength  xlength -ylength ylength])
    rpoly='    nonadecagon';
   rtext=1;
    case 26
   axis([-xlength  xlength -ylength ylength])
    rpoly='    decagon';
    rtext=1;
    case 27
   axis([-xlength  xlength -ylength ylength])
    rpoly='    hendecagon';
   rtext=1;
    case 28
   axis([-xlength  xlength -ylength ylength])
    rpoly='    dodecagon';
   rtext=1;
    case 29
   axis([-xlength  xlength -ylength ylength])
    rpoly='    tridecagon';
   rtext=1;
    case 30
   axis([-xlength  xlength -ylength ylength])
    rpoly='    tetradecagon';
   rtext=1;
```

```
    case 31
   axis([-xlength  xlength -ylength ylength])
    rpoly='    pentadecagon';
   rtext=1;
    case 32
   axis([-xlength  xlength -ylength ylength])
    rpoly='    hexadecagon';
   rtext=1;
    case 33
   axis([-xlength  xlength -ylength ylength])
    rpoly='    heptadecagon';
   rtext=1;
    case 34
   axis([-xlength  xlength -ylength ylength])
   rpoly='    octadecagon';
   rtext=1;
    case 35
   axis([-xlength  xlength -ylength ylength])
   rpoly='    enneadecagon';
   rtext=1;
 case 36
   axis([-xlength  xlength -ylength ylength])
   rpoly='    icosagon';
   rtext=1;

end

patch(xvec(1,:),yvec(1,:),'w');%plot  element
patch(xvec(2,:),yvec(2,:),'w');%plot  element
patch(xvec(3,:),yvec(3,:),'w');%plot  element
hold on;
%place element number
if ndiv<=2
midx1=mean(xvec(1,1:4))
midy1=mean(yvec(1,1:4))
text(midx1,midy1,['*',num2str(i)]);
%
midx2=mean(xvec(2,1:4))
midy2=mean(yvec(2,1:4))
text(midx2,midy2,['*',num2str(i+1)]);
%
midx3=mean(xvec(3,1:4))
midy3=mean(yvec(3,1:4))
text(midx3,midy3,['*',num2str(i+2)]);
end
end;%i loop

xlabel('x  axis')
ylabel('y  axis')
st1='FEM  MESH  WITH  ';
st2=num2str(nel);
st3='; 4-node Bilinear  ';
st4='Quadrilateral';
st5=' elements'
st6='& nodes='
st7=num2str(NNNODE);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
%**************************************
if ndiv<4
for jj=1:NNNODE
text(gcoords(jj,1),gcoords(jj,2),['o',num2str(jj)]);
end
end
```

```
hold on
figure(ndiv),scatter(xcoords,ycoords,10,'filled','r')
hold off
%TNODES
tnodes
%
NNN(:,1)=(1:NNNODE)';
[NNN  gcoords]
```

(2) triangulation4polygonal_domain_coordinatesSingularCorner.m

```
function[eln,trielm,rrr,coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesSingularCorner(n1,n
2,n3,nmax,numtri,n,mesh,beta)
%n1=node number at(0,0)for   a choosen triangle
%n2=node number at(1,0)for   a choosen triangle
%n3=node number at(0,1)for   a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e   number of divisions
%nmax=one  plus the number of segments of the polygon
%nmax=the  number of segments of the polygon plus a node interior to the polygon
%numtri=number  of T6 triangles in each segment i.e  a triangle formed  by
%joining the end poits of the segment to the interior  point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ  MOIN EXAMPLE
syms U V W xi yi

switch mesh
case 1%for  MOIN  POLYGON
x=sym([1/2;1/2;1;    1;1/2;0;   0;0])%for  MOIN  EXAMPLE
y=sym([1/2;   0;0;1/2;   1;1;1/2;0])%for  MOIN  EXAMPLE



case 2%for a unit square: 0<=x,y<=1
x=sym([1/2;1/2;1;    1;  1;1/2;0;   0;0])%FOR  UNIT  SQUARE
y=sym([1/2;   0;0;1/2;   1;  1;1;1/2;0])%FOR  UNIT  SQUARE

case 3%for  A POLYGON  like  MOIN OVER(-1/2)<=x,y<=(1/2)
   %  1    2    3   4    5   6    7   8
  x=sym([0;    0;  1/2;1/2;   0;-1/2;-1/2;-1/2])
  y=sym([0;-1/2;-1/2;    0;1/2;  1/2;    0;-1/2])

case 4%for a unit square: -0.5<=x,y<=0.5
  %       1    2   3   4   5   6    7   8
  x=sym([0;    0;  1/2;1/2;1/2;   0;-1/2;-1/2;-1/2])
  y=sym([0;-1/2;-1/2;    0;1/2;1/2;  1/2;    0;-1/2])
case 5%for a convexpolygonsixside
  %    1    2   3  4   5  6  7
x=sym([0.5;0.1;0.7;1;.75;.5;0])
y=sym([0.5;0;0.2;.5;.85;1;.25])
case 6%standard triangle
x=sym([0;1;0])
y=sym([0;0;1])
case 7%equilateral  triangle

x=sym([0;1;1/2])
y=sym([0;0;sqrt(3)/2])
%x=sym([-1/2;1/2;0])
```

```
%y=sym([0;0;sqrt(3)/2])
case 8%for a convexpolygonsixside
  %    1    2   3 4   5 6  7  8
x=sym([0.5;0.1;0.7;1;.75;.5;.25;0])
y=sym([0.5;0;0.2;.5;.85;1;.625;.25])
case 9%for a convexpolygeightside
%    1  2   3  4   5    6  7     8   9
x=([.2;0.5;.8;1.0;.75;0.5;0.25;0.0;0.5])
y=([.2;.05;.2;0.5;.85;1.0;0.90;0.6;0.5])
case 10
%    1  2   3  4   5    6  7     8   9
x=([0.5;.2;0.5;.8;1.0;.75;0.5;0.25;0.0])
y=([0.5;.2;.05;.2;0.5;.85;1.0;0.90;0.6])
 case 11 %a square
 %  1 2 3 4  5
%x=([0;1;1;0;0.5])
%y=([0;0;1;1;0.5])
 %     1 2 3 4  5
x=sym([1;1;0;0;0.5])
y=sym([0;1;1;0;0.5])
case 12  %a 2-square [-1,1]x[-1,1]
 %       1    2   3    4   5   6    7   8    9
  x=sym([0;    0;  1;   1;  1;  0;  -1;  -1; -1])
  y=sym([0;   -1; -1;   0;  1;  1;   1;   0; -1])
case 13%for  a unit square: 0<=x,y<=1
x=sym([1/2;1/2;1;    1;  1;1/2;0;   0;0])%FOR  UNIT  SQUARE
y=sym([1/2;   0;0;1/2;   1;  1;1;1/2;0])%FOR  UNIT  SQUARE
case 14%for  a unit square: 0<=x,y<=1
x=sym([1/2;1/2;1;    1;  1;1/2;0;   0;0])%FOR  UNIT  SQUARE
y=sym([1/2;   0;0;1/2;   1;  1;1;1/2;0])%FOR  UNIT  SQUARE
case 15%for  a unit square: 0<=x,y<=1
x=sym([1/2;1/2;1;    1;  1;1/2;0;   0;0])%FOR  UNIT  SQUARE
y=sym([1/2;   0;0;1/2;   1;  1;1;1/2;0])%FOR  UNIT  SQUAREpi

case 16  %a 2-square [-1,1]x[-1,1]
 %       1    2   3    4   5   6    7   8    9
  x=sym([0;    0;  1;   1;  1;  0;  -1;  -1; -1])
  y=sym([0;   -1; -1;   0;  1;  1;   1;   0; -1])

case 17%isosceles triangle(one triangle in a square-required for torsion analysis)
x=sym([0;sqrt(2)/2;   -sqrt(2)/2])
y=sym([0;sqrt(2)/2;sqrt(2)/2])
case 18%isoscles triangle(torsion of an equilateral triangle,each side=2*sqrt(3))
 x=sym([-sqrt(3);sqrt(3);   0])
 y=sym([      -1;       -1;  2])
 case 19%triangle  inscribed in a circle  of radius=1(torsion of an equilateral triangle each side=sqrt(3))
 x=sym([-sqrt(3)/2;sqrt(3)/2;   0])
 y=sym([      -1/2;       -1/2;  1])
 case 20%square inscribed in a circle  of radius=1,required  for torsion analysis
%      1      2          3       4          5
x=sym([0;-sqrt(2)/2;   sqrt(2)/2;sqrt(2)/2;-sqrt(2)/2])
y=sym([0;-sqrt(2)/2;-sqrt(2)/2;sqrt(2)/2;   sqrt(2)/2])
case 21%regular  pentagon inscribed in a circle  of radius=1,required  for torsion analysis
 %      1      2    3       4           5           6
x=sym([0;cos(pi/10);0;-cos(pi/10);-cos(3*pi/10);   cos(3*pi/10)])
y=sym([0;sin(pi/10);1;  sin(pi/10);-sin(3*pi/10);-sin(3*pi/10)])
case 22%regular  hexagon inscribed in a circle  of radius=1,required  for torsion analysis
    %      1 2   3          4        5     6          7
    x=sym([0;1;cos(pi/3);-cos(pi/3);-1;   -cos(pi/3); cos(pi/3)])
    y=sym([0;0;sin(pi/3);  sin(pi/3); 0;  -sin(pi/3);-sin(pi/3)])
    case 23%regular  heptagon inscribed in a circle  of radius=1,required  for torsion analysis
 %      1      2           3             4           5       6       7         8
x=sym([0;-cos(5*pi/14);   cos(5*pi/14); cos(pi/14);cos(3*pi/14);0;-cos(3*pi/14);-cos(pi/14)])
y=sym([0;-sin(5*pi/14);-sin(5*pi/14);-sin(pi/14);sin(3*pi/14);1;   sin(3*pi/14);-sin(pi/14)])
```

```
   case 24 %regular octagon inscribed in a circle of radius=1,required for torsion analysis
 %    1 2    3    4    5      6      7    8    9
x=sym([0;1;cos(pi/4);0;-cos(pi/4);-1;-cos(pi/4);  0; cos(pi/4)])
y=sym([0;0;sin(pi/4);1;  sin(pi/4); 0;-sin(pi/4);-1;-sin(pi/4)])
   case 25%(9-gon)nonagon
 %       1     2          3     4        5        6         7          8         9          10
  x=sym([0;cos(pi/18);  cos(5*pi/18);0;-cos(5*pi/18);-cos(pi/18);-cos(3*pi/18);-cos(7*pi/18);  cos(7*pi/18); cos(3*pi/18)])
  y=sym([0;sin(pi/18);  sin(5*pi/18);1;  sin(5*pi/18); sin(pi/18);-sin(3*pi/18);-sin(7*pi/18);-sin(7*pi/18);-sin(3*pi/18)])
   case 26%(10-gon)decagon
 %       1 2    3        4        5           6       7    8         9          10      11
  x=sym([0;1;cos(pi/5);  cos(2*pi/5);-cos(2*pi/5);-cos(pi/5);-1;-cos(pi/5);-cos(2*pi/5);  cos(2*pi/5); cos(pi/5)])
  y=sym([0;0;sin(pi/5);  sin(2*pi/5); sin(2*pi/5); sin(pi/5); 0;-sin(pi/5);-sin(2*pi/5);-sin(2*pi/5);-sin(pi/5)])
   case 27%(11-gon)hendecagon
  %     1       2       3      4       5        6          7       8        9          10           11      12
  x=sym([0;cos(3*pi/22);cos(7*pi/22);0;-cos(7*pi/22);-cos(3*pi/22);-cos(pi/22);-cos(5*pi/22);-cos(9*pi/22);  cos(9*pi/22);
cos(5*pi/22); cos(pi/22)])
  y=sym([0;sin(3*pi/22);sin(7*pi/22);1;  sin(7*pi/22); sin(3*pi/22);-sin(pi/22);-sin(5*pi/22);-sin(9*pi/22);-sin(9*pi/22);-
sin(5*pi/22);-sin(pi/22)])
  case 28%(12-gon)dodecagon
  %     1 2  3        4    5     6        7     8      9       10    11      12     13
  x=sym([0;1;cos(pi/6);cos(pi/3);0;-cos(pi/3);-cos(pi/6);-1;-cos(pi/6);-cos(pi/3);  0; cos(pi/3); cos(pi/6)])
  y=sym([0;0;sin(pi/6);sin(pi/3);1;  sin(pi/3); sin(pi/6); 0;-sin(pi/6);-sin(pi/3);-1;-sin(pi/3);-sin(pi/6)])
  case 29%(13-gon)tridecagon
  %     1       2       3         4       5     6             7         8      9             10          11          12         13          14
  x=sym([0;cos(pi/26);cos(5*pi/26);cos(9*pi/26);0;-cos(9*pi/26);-cos(5*pi/26);-cos(pi/26);-cos(3*pi/26);-cos(7*pi/26);-
cos(11*pi/26); cos(11*pi/26); cos(7*pi/26); cos(3*pi/26)])
  y=sym([0;sin(pi/26);sin(5*pi/26);sin(9*pi/26);1;  sin(9*pi/26); sin(5*pi/26); sin(pi/26);-sin(3*pi/26);-sin(7*pi/26);-
sin(11*pi/26);-sin(11*pi/26);-sin(7*pi/26);-sin(3*pi/26)])
   case 30%(14-gon)tetradecagon
  %     1 2       3        4          5       6             7         8      9      10          11          12         13         14          15
  x=sym([0;1;cos(2*pi/14);cos(4*pi/14);cos(6*pi/14);-cos(6*pi/14);-cos(4*pi/14);-cos(2*pi/14);-1;-cos(2*pi/14);-cos(4*pi/14);-
cos(6*pi/14); cos(6*pi/14); cos(4*pi/14); cos(2*pi/14)])
  y=sym([0;0;sin(2*pi/14);sin(4*pi/14);sin(6*pi/14);  sin(6*pi/14); sin(4*pi/14); sin(2*pi/14);  0;-sin(2*pi/14);-sin(4*pi/14);-
sin(6*pi/14);-sin(6*pi/14);-sin(4*pi/14);-sin(2*pi/14)])
   case 31%(15-gon)pentadecagon
  %     1       2       3          4          5      6   7             8          9      10          11          12         13         14          15
16
   x=sym([0;  cos(pi/30);cos(3*pi/30);cos(7*pi/30);cos(11*pi/30);0;-cos(11*pi/30);-cos(7*pi/30);-cos(3*pi/30);-cos(pi/30);-
cos(5*pi/30);-cos(9*pi/30);-cos(13*pi/30);  cos(13*pi/30); cos(9*pi/30); cos(5*pi/30)])
   y=sym([0;-sin(pi/30);sin(3*pi/30);sin(7*pi/30);sin(11*pi/30);1;  sin(11*pi/30); sin(7*pi/30); sin(3*pi/30);-sin(pi/30);-
sin(5*pi/30);-sin(9*pi/30);-sin(13*pi/30);-sin(13*pi/30);-sin(9*pi/30);-sin(5*pi/30)])
   case 32%(16-gon)hexadecagon
   % 1 2    3      4         5      6     7          8      9     10     11        12        13    14      15         16       17
   x=sym([0;1;cos(pi/8);cos(pi/4);cos(3*pi/8);0;-cos(3*pi/8);-cos(pi/4);-cos(pi/8);-1;-cos(pi/8);-cos(pi/4);-cos(3*pi/8);  0;
cos(3*pi/8); cos(pi/4); cos(pi/8)]);
   y=sym([0;0;sin(pi/8);sin(pi/4);sin(3*pi/8);1;  sin(3*pi/8); sin(pi/4); sin(pi/8); 0;-sin(pi/8);-sin(pi/4);-sin(3*pi/8);-1;-sin(3*pi/8);-
sin(pi/4);-sin(pi/8)]);
   case 33%(17-gon)heptadecogon
  %   1       2        3      4          5      6       7          8        9          10         11        12           13          14
15          16          17      18
   x=sym([0;cos(pi/34);cos(5*pi/34);cos(9*pi/34);cos(13*pi/34);0;-cos(13*pi/34);-cos(9*pi/34);-cos(5*pi/34);-cos(pi/34);-
cos(3*pi/34);-cos(7*pi/34);-cos(11*pi/34);-cos(15*pi/34);  cos(15*pi/34); cos(11*pi/34); cos(7*pi/34); cos(3*pi/34)]);
   y=sym([0;sin(pi/34);sin(5*pi/34);sin(9*pi/34);sin(13*pi/34);1;  sin(13*pi/34); sin(9*pi/34); sin(5*pi/34); sin(pi/34);-
sin(3*pi/34);-sin(7*pi/34);-sin(11*pi/34);-sin(15*pi/34);-sin(15*pi/34);-sin(11*pi/34);-sin(7*pi/34);-sin(3*pi/34)]);
 case 34%(18-gon)octadecogon
  % 1 2        3      4          5          6       7          8         9          10      11       12         13          14          15
16          17      18          19
 x=sym([0;1;cos(4*pi/36);cos(8*pi/36);cos(12*pi/36);cos(16*pi/36);-cos(16*pi/36);-cos(12*pi/36);-cos(8*pi/36);-cos(4*pi/36);-
1;-cos(4*pi/36);-cos(8*pi/36);-cos(12*pi/36);-cos(16*pi/36);  cos(16*pi/36); cos(12*pi/36); cos(8*pi/36); cos(4*pi/36)]);
 y=sym([0;0;sin(4*pi/36);sin(8*pi/36);sin(12*pi/36);sin(16*pi/36);  sin(16*pi/36); sin(12*pi/36); sin(8*pi/36); sin(4*pi/36); 0;-
sin(4*pi/36);-sin(8*pi/36);-sin(12*pi/36);-sin(16*pi/36);-sin(16*pi/36);-sin(12*pi/36);-sin(8*pi/36);-sin(4*pi/36)]);
case 35%(19-gon)enneadecogon
  % 1       2        3        4         5      6     7          8         9         10         11        12        13          14         15
16          17      18          19          20
```

```
x=sym([0;cos(3*pi/38);cos(7*pi/38);cos(11*pi/38);cos(15*pi/38);0;-cos(15*pi/38);-cos(11*pi/38);-cos(7*pi/38);-cos(3*pi/38);-cos(pi/38);-cos(5*pi/38);-cos(9*pi/38);-cos(13*pi/38);-cos(17*pi/38);  cos(17*pi/38); cos(13*pi/38); cos(9*pi/38); cos(5*pi/38); cos(pi/38)]);
 y=sym([0;sin(3*pi/38);sin(7*pi/38);sin(11*pi/38);sin(15*pi/38);1;  sin(15*pi/38); sin(11*pi/38); sin(7*pi/38); sin(3*pi/38);-sin(pi/38);-sin(5*pi/38);-sin(9*pi/38);-sin(13*pi/38);-sin(17*pi/38);-sin(17*pi/38);-sin(13*pi/38);-sin(9*pi/38);-sin(5*pi/38);-sin(pi/38)]);
  case 36%(20-gon)icosagon
  % 1 2    3         4          5         6     7         8          9          10        11  12     13         14          15          16     17      18          19          20     21
x=sym([0;1;cos(pi/10);cos(2*pi/10);cos(3*pi/10);cos(4*pi/10);0;-cos(4*pi/10);-cos(3*pi/10);-cos(2*pi/10);-cos(pi/10);-1;-cos(pi/10);-cos(2*pi/10);-cos(3*pi/10);-cos(4*pi/10);  0; cos(4*pi/10); cos(3*pi/10); cos(2*pi/10); cos(pi/10)]);
y=sym([0;0;sin(pi/10);sin(2*pi/10);sin(3*pi/10);sin(4*pi/10);1;  sin(4*pi/10); sin(3*pi/10); sin(2*pi/10); sin(pi/10); 0;-sin(pi/10);-sin(2*pi/10);-sin(3*pi/10);-sin(4*pi/10);-1;-sin(4*pi/10);-sin(3*pi/10);-sin(2*pi/10);-sin(pi/10)]);

case 50%isoscles triangle(torsion of an equilateral triangle,each side=1 unit)
 x=sym([-1/2;1/2;         0])
 y=sym([    0;  0;sqrt(3)/2])

 case 51%one special quadrilateral with centroid in a triangle(torsion of an equilateral triangle,each side=1 unit)
%               1        2        3        4       5
 x=sym([          0;       0;      1/4;       0;     -1/4])
 y=sym([7*sqrt(3)/24;sqrt(3)/6;sqrt(3)/4;sqrt(3)/2;sqrt(3)/4])
 case 52
 x=sym([0;1;1])
 y=sym([0;0;1])
  case 53
 x=sym([0;1;0])
 y=sym([0;1;1])
 case 54
 x=sym([0;1;1;0;-1;  0])
 y=sym([0;0;1;1;  0;-1])
  case 55
 %     1  2  3 4 5  6
 x=sym([0;-1;  1;2;0;-2])
 y=sym([0;-1;-1;0;1;  0])

case 56
 x=sym([0;1;1;0;-1])
 y=sym([0;0;1;1;-1])
case 57%for  a unit square: 0<=x,y<=1
%     1    2   3   4  5   6 7   8  9
x=sym([1/2;1/2;0;    0;  0;1/2;1;  1;1])%FOR  UNIT  SQUARE
y=sym([1/2;   1;1;1/2;  0;  0;0;1/2;1])%FOR  UNIT  SQUARE

case 58%for  a two square: -1<=x,y<=1
%     1 2 3 4  5  6  7  8  9
x=sym([0;1;1;0;-1;-1;-1;  0; 1])%FOR  A TWO SQUARE
y=sym([0;0;1;1;  1; 0;-1;-1;-1])%FOR  A TWO SQUARE

case 59%for  a two square: -1<=x,y<=1
%     1  2  3 4  5  6  7  8  9
x=sym([0;  0;  1;1;  1;  0;-1;-1;-1])%FOR  A TWO SQUARE
y=sym([0;-1;-1;0;  1;  1;  1;  0;-1])%FOR  A TWO SQUARE
case 60%for  a V SHAPED  GEOMETRY
%     1    2  3  4  5 6 7   8  9
x=sym([1/2;0;    0;0;1/2;1;   1;1;1/2])%FOR  V-SHAPED  GEOMETRY
y=sym([.9;  1;1/2;0;   0;0;1/2;1;   1])%FOR  V-SHAPED  GEOMETRY

end


if (nmax>3)
%[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles_trial(n1,n2,n3,nmax,numtri,n);
```

```
[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles_trial(n1,n2,n3,nmax,numtri,n);
end
if (nmax==3)
   [eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles(n);
end
%[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
%[U,V,W  ] =triangularmeshpoints4evendivisionsN4singularcorner(n/2,2)
[U,V,W]  =triangularmeshpoints4singularcorner(n,beta)
%[U,V,W]  =triangularmeshpoints4singularcorner(n,2)
%[U,V,W]  =triangularmeshpoints4singularcorner(n,1.5)
%[U,V,W]  =triangularmeshpoints4singularcorner(n,0.5)
ss1='number of 6-node triangles =';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
%
eln
%
ss2='number of triangular elements&nodes per element  =';
[nel,nnel]=size(trielm);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
trielm
%
nnode=max(max(trielm));
ss3='number of nodes of the triangular domain&  number of triangular elements=';
disp([ss3 num2str(nnode) ',' num2str(nel)])
 nitri=nmax-1;
 if nmax==4
    nitri=2;
 end

if (nmax==3)
   nitri=1;
end
for itri=1:nitri
disp('vertex nodes of the itri triangle')
[n1(itri,1)  n2(itri,1)  n3(itri,1)]
x1=x(n1(itri,1),1)
x2=x(n2(itri,1),1)
x3=x(n3(itri,1),1)
%
y1=y(n1(itri,1),1)
y2=y(n2(itri,1),1)
y3=y(n3(itri,1),1)
rrr(:,:,itri)
U'
V'
W'
kk=0;
for ii=1:n+1
   for jj=1:(n+1)-(ii-1)
      kk=kk+1;
      mm=rrr(ii,jj,itri);
      uu=U(kk,1);vv=V(kk,1);ww=W(kk,1);
      xi(mm,1)=x1*ww+x2*uu+x3*vv;
      yi(mm,1)=y1*ww+y2*uu+y3*vv;

%        xi(mm,1)=x1*vv+x2*ww+x3*uu;
%        yi(mm,1)=y1*vv+y2*ww+y3*uu;
   end
end
[xi  yi]
%add coordinates of centroid
 ne=(n/2)^2;
```

```
end%for itri
N=(1:nnode)'
[N xi  yi]
%
coord(:,1)=(xi(:,1));
coord(:,2)=(yi(:,1));
gcoord(:,1)=double(xi(:,1));
gcoord(:,2)=double(yi(:,1));
%disp(gcoord)
```

(3)nodaladdresses_for4XnXn_LinearTriangles.m

```
function[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles(n)
%we first generate 6-node triangles
%hence it is necessary that n is an even number,i.e:  n=2,4,6,8.....etc
%this generates (n/2)^2 triangles with 6-nodes each
%in each six node triangle we can make  4-Linear  Triangles
%standard triangle is divided into n^2 right isoscles
%triangles each of side length (1/n) units
%computes nodal connections of these right isiscles triangles
%assumes nodal addresses for the standard triangle has local nodes
%as {1,2,3}  which correspond to global nodes {1,(n+1),(n+1)*(n+2)/2}
%respectively and then generates  nodal addresses for
%six node triangles and special convex quadrilaterals
%eln=6-node triangles
%triel=3-node  linear triangular elements created in 6-node triangle
%n=number of divisions of a side and n must be even,i.e.n=2,4,6,.....
%syms mst_tri x
%disp('vertex nodes of triangle')
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2,1)=3;
%disp('vertex nodes of triangle')
%base edge
kk=3;
for k=2:n
   kk=kk+1;
   elm(k,1)=kk;
end
%disp('left edge nodes')
nni=1;
for i=0:(n-2)
   nni=nni+(n-i)+1;
   elm(nni,1)=3*n-i;
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-2)
   nni=nni+(n-i);
   elm(nni,1)=(n+3)+i;
end

%disp('interior nodes')
nni=1;jj=0;
for i=0:(n-3)
   nni=nni+(n-i)+1;
   for j=1:(n-2-i)
      jj=jj+1;
      nnj=nni+j;
      elm(nnj,1)=3*n+jj;
   end
end
%disp(elm)
%disp(length(elm))
```

```
jj=0;kk=0;
for j=0:n-1
   jj=j+1;
for k=1:(n+1)-j
   kk=kk+1;
   row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=3;
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;



rr=row_nodes;
rr
rrr(:,:,1)=rr;

%rr
%disp('element computations')
if rem(n,2)==0
ne=0;N=n+1;

for k=1:2:n-1
N=N-2;
i=k;
for j=1:2:N
   ne=ne+1;
eln(ne,1)=rr(i,j);
eln(ne,2)=rr(i,j+2);
eln(ne,3)=rr(i+2,j);
eln(ne,4)=rr(i,j+1);
eln(ne,5)=rr(i+1,j+1);
eln(ne,6)=rr(i+1,j);
end%i
%me=ne
%N-2
if (N-2)>0
for jj=1:2:N-2
ne=ne+1;
eln(ne,1)=rr(i+2,jj+2);
eln(ne,2)=rr(i+2,jj);
eln(ne,3)=rr(i,jj+2);
eln(ne,4)=rr(i+2,jj+1);;
eln(ne,5)=rr(i+1,jj+1);
eln(ne,6)=rr(i+1,jj+2);
end%jj
end
end%k
end
%
mm=0;mmm=0;
for iel=1:ne
   for jel=1:4
   %mm=mm+1;mmm=mmm+1;
     switch jel
     case 1
        mm=mm+1;mmm=mmm+1;
       trielm(mmm,1:3)=[eln(iel,1)  eln(iel,4)  eln(iel,6)];
       tnodes(mmm,1:3)=trielm(mmm,1:3);
       nodetel(mm,1:3)=[eln(iel,1)  eln(iel,2)  eln(iel,3)];
```

```
        case 2
        mm=mm+1;mmm=mmm+1;
        trielm(mmm,1:3)=[eln(iel,2)  eln(iel,5)  eln(iel,4)];
        tnodes(mmm,1:3)=trielm(mmm,1:3);
        nodetel(mm,1:3)=[eln(iel,2)  eln(iel,3)  eln(iel,1)];
        case 3
        mm=mm+1;mmm=mmm+1;
        trielm(mmm,1:3)=[eln(iel,3)  eln(iel,6)  eln(iel,5)];
        tnodes(mmm,1:3)=trielm(mmm,1:3);
        nodetel(mm,1:3)=[eln(iel,3)  eln(iel,1)  eln(iel,2)];
         case 4
         mmm=mmm+1;
         trielm(mmm,1:3)=[eln(iel,4)  eln(iel,5)  eln(iel,6)];
         tnodes(mmm,1:3)=trielm(mmm,1:3);


     end%switch
    end
  end

eln
trielm
tnodes
nodetel
```

(4) nodaladdresses_for4XnXn_LinearTriangles_trial.m

```
function[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles_trial(n1,n2,n3,nmax,numtri,n)
%function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,n)
%n1=node number at(0,0)for   a choosen triangle
%n2=node number at(1,0)for   a choosen triangle
%n3=node number at(0,1)for   a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e  number of divisions
%nmax=one  plus the number of segments of the polygon
%nmax=the  number of segments of the polygon plus a node interior to the polygon
%numtri=number  of T6 triangles in each segment i.e  a triangle formed  by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ  MOIN EXAMPLE
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8
,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8
,4,4)
%syms mst_tri x
ne=0;

%nitri=nmax-1;
[nitri,cl]=size(n1)


for itri=1:nitri
   elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1)  n2(itri,1)  n3(itri,1)]
```

```
if itri==1
kk=nmax;
for k=2:n
   kk=kk+1
   elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
   elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
   lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
   mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
   hh=hh+1;
   nni=nni+(n-i);
   elm(nni,1)=(mmax+1)+i;
   qq(hh,1)=(mmax+1)+i;

end
qq(n+1,1)=n3(itri,1);
edgen2n3(1:n+1,itri)=qq;

if itri<nitri
disp('left edge nodes')
nni=1;gg=1;pp(1,1)=n1(itri,1);
for i=0:(n-2)
   gg=gg+1;
   nni=nni+(n-i)+1;
   elm(nni,1)=lmax-i;
   pp(gg,1)=lmax-i;
end
pp(n+1,1)=n3(itri,1);
edgen1n3(1:n+1,itri)=pp
end%if itri<nitri



%if itri==n
% elm(1:n+1,1)=edgen1n2(1:n+1,1)
%end

if itri==nitri
disp('left edge nodes')
nni=1;gg=1;
for i=0:(n-2)
   gg=gg+1;
   nni=nni+(n-i)+1;
   elm(nni,1)=edgen1n2(gg,1);
end
%pp(n+1,1)=n3(itri,1);
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
```

```
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri



%elm
disp('interior nodes')
nni=1;jj=0;
for i=0:(n-3)
   nni=nni+(n-i)+1;
   for j=1:(n-2-i)
      jj=jj+1;
      nnj=nni+j;
      elm(nnj,1)=lmax+jj;
      [nnj lmax+jj];
   end
end
%disp(elm);
%disp(length(elm));

jj=0;kk=0;
for j=0:n-1
   jj=j+1;
for k=1:(n+1)-j
   kk=kk+1;
   row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=n3(itri,1);
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;
rr
rrr(:,:,itri)=rr;
disp('element computations')
if rem(n,2)==0
N=n+1;

for k=1:2:n-1
N=N-2;
i=k;
for j=1:2:N
   ne=ne+1
eln(ne,1)=rr(i,j);
eln(ne,2)=rr(i,j+2);
eln(ne,3)=rr(i+2,j);
eln(ne,4)=rr(i,j+1);
eln(ne,5)=rr(i+1,j+1);
eln(ne,6)=rr(i+1,j);
end%j
%me=ne
%N-2
if (N-2)>0
for jj=1:2:N-2
ne=ne+1
eln(ne,1)=rr(i+2,jj+2);
eln(ne,2)=rr(i+2,jj);
eln(ne,3)=rr(i,jj+2);
eln(ne,4)=rr(i+2,jj+1);
eln(ne,5)=rr(i+1,jj+1);
```

```
eln(ne,6)=rr(i+1,jj+2);
end%jj
end%if(N-2)>0
end%k

end% if rem(n,2)==0
ne
%for kk=1:ne
%[eln(kk,1:6)]
%end
%add node numbers for element centroids

nnd=max(max(eln))
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd



end%itri

%generate 4-Linear triangles in a 6-node triangles,call these as stel
mm=0;mmm=0;
for iel=1:ne
   for jel=1:4
   %mm=mm+1;mmm=mmm+1;
     switch jel
     case 1
        mm=mm+1;mmm=mmm+1;
       trielm(mmm,1:3)=[eln(iel,1)  eln(iel,4)  eln(iel,6)];
       tnodes(mmm,1:3)=trielm(mmm,1:3);
       nodetel(mm,1:3)=[eln(iel,1)  eln(iel,2)  eln(iel,3)];
       case 2
       mm=mm+1;mmm=mmm+1;
       trielm(mmm,1:3)=[eln(iel,2)  eln(iel,5)  eln(iel,4)];
       tnodes(mmm,1:3)=trielm(mmm,1:3);
       nodetel(mm,1:3)=[eln(iel,2)  eln(iel,3)  eln(iel,1)];
       case 3
       mm=mm+1;mmm=mmm+1;
       trielm(mmm,1:3)=[eln(iel,3)  eln(iel,6)  eln(iel,5)];
       tnodes(mmm,1:3)=trielm(mmm,1:3);
       nodetel(mm,1:3)=[eln(iel,3)  eln(iel,1)  eln(iel,2)];
        case 4
        mmm=mmm+1;
        trielm(mmm,1:3)=[eln(iel,4)  eln(iel,5)  eln(iel,6)];
        tnodes(mmm,1:3)=trielm(mmm,1:3);


     end%switch
   end
  end

eln
trielm
tnodes
nodetel


%
ss1='number of 6-node triangles =';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
```

```
%
eln
%
ss2='number of linear triangular elements&nodes per element =';
[nel,nnel]=size(trielm);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
nnode=max(max(trielm));
ss3='number of nodes of the triangular domain&  number of linear triangles =';
disp([ss3 num2str(nnode) ',' num2str(nel)])
```

(5)triangularmeshpoints4singularcorner.m

```
function [U,V,W]  =triangularmeshpoints4singularcorner(p,q)
%draw outline for the triangle
%triangularmeshpoints4singularcorner(7,2)
%triangularmeshpoints4singularcorner(7,1/2)
clc
syms ui vi wi U V W
syms UI VI  WI UU  VV  WW
axis  square
% x=[1;0;0;1]
% y=[0;1;0;0]
axis([0  1 0 1])
%figure(1)
% plot(x,y,'r-')
% hold on
% ui(1,1)=0;vi(1,1)=0;
% ui(1,2)=(1/p)^q;;vi(1,2)=0;
% ui(2,1)=0;;vi(2,1)=(1/p)^q;
% figure(1),scatter(ui(1,2),vi(1,2),15,'filled','g')
% figure(1),scatter(ui(2,1),vi(2,1),15,'filled','g')
kk=0;
for j=0:p
mn=j;
if j==0
jpq=0;
end
if j>0
jpq=(j/p)^q;
end
  for k=0:j
      kk=kk+1;
    %mn=m+n
    m=k;n=mn-m;
     if ((m+n)==0)
    ui(m+1,n+1)=0;
    vi(m+1,n+1)=0;
     wi(m+1,n+1)=1;
    end

    if ((m+n)>0)
    ui(m+1,n+1)=n*jpq/(m+n);
    vi(m+1,n+1)=m*jpq/(m+n);
    wi(m+1,n+1)=1-ui(m+1,n+1)-vi(m+1,n+1);
    end
%     U(kk,1)=ui(m+1,n+1);
%         V(kk,1)=vi(m+1,n+1);



  end%for k
end%for j

      % figure(1),scatter(U(:,1),V(:,1),10,'filled','r')
%   ui
%   vi
```

```
% [U(:,1)  V(:,1)]
% [ui(1,2)  vi(1,2)]
% [ui(2,1)  vi(2,1)]
UI=ui';VI=vi';WI=wi';
kk=0;n=p;
for j=1:n+1
   for i=1:(n+1)-(j-1)
       kk=kk+1;

       U(kk,1)=UI(i,j);
       V(kk,1)=VI(i,j);
       W(kk,1)=WI(i,j);

   end

end
uii= vpa(ui,5);
 vii=vpa(vi,5);
 wii=vpa(wi,5);
% disp('-----------------------------------------')
 UII=vpa(UI,5);
 VII=vpa(VI,5);
 WII=vpa(WI,5);
  kku=length(U)
  kkv=length(V)
   format short
%display the mesh points,number of divisions and grading factor
 %to display the mesh points set display=1
 display=0
 if display==1
figure(p/2),scatter(U(:,1),V(:,1),10,'filled','r')
xlabel('x  axis')
ylabel('y axis')
st1='mesh divisions = ';
st2=num2str(p);
st3='; &';
st4='grading factor=';
st5=num2str(q);
st6='& number of mesh points='
st7=num2str(kku);
title([st1,st2,st3,st4,st5,st6,st7])
 end
```