www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 8 Issue 3 March 2019, Page No. 24502-24574
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v8i03.4297

A New Approach to Auto generation of Finite Element Meshes Over Curved Domains with Boundary Defined by Polar Equations

H.T. Rathod[a*], Bharath Rathod[b], K. Sugantha Devi[c], K.V.Vijayakumar[d]

[a] Department of Mathematics, Jnana Bharathi Campus, Bangalore University, Bangalore -560056, Karnataka state, India.
[b] Xavier Institute of Management and Entrepreneurship, Hosur Road, Electronic City Phase II, Bangalore, Karnataka 560034.
[c] Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post, Kolar Gold Field, Kolar District, Karnataka state, Pin- 563120, India.
[d] Department of Mathematics,B.M.S.Institute of Technology,Avalahalli, Bangalore-560064, Karnataka State, India.

Abstract-

This paper presents a new mesh generation method for a simply connected curved domain of a planar region whose boundary is defined by one or more equations in polar coordinates and a positive integer parameter 'n', (n=2,4,6,8,.....etc.) **for specification of the wished degree of refinement.**We first decompose this curved domain into simple sub regions in the shape of curved triangles. These simple regions are then triangulated to generate a fine mesh of linear triangles in the interior and curved triangles or linear triangles near to the boundary of curved domain.These simple regions are then triangulated to create 6-node triangles by inserting midside nodes to these triangles. Each isolated 6-node triangle is then split into four triangles according to the usual scheme, that is, by using straight lines to join the midside nodes. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given **starshaped** curved domain or cracked convex curved domains into all triangles and then into all quadrilaterals.Thus we autogenerate a triangular and a quadrangular finite element mesh. The refinements to the mesh are obtained by increasing the number of divisions of the boundary curve.

Keywords: finite elements, triangular,quadrangular, mesh generation,analytical curved surfaces, curved triangular element, polar boundary equations,uniform refinement,starshaped curved domain.

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 use of finite element techniques for solving partial differential equations is even more suitable than other classical discretization methods for the case when the equation is defined on curved or irregular domains. This is mainly due to their versatility for fitting boundary conditions. In the case of two-dimensional domains, triangular elements enable this adjustment in a specially easy way. In general, however, it is necessary to define the triangulation of the domain empirically,which means that data related to the discretization, such as coordinates of the vertices of the triangles and their numbering, must be input when running a finite element program. This turns out to be an even more inconvenient drawback if one wishes to improve the precision of approximate solutions by using refined meshes. In fact, in this case, it would be necessary to interrupt the processing in order to redefine the partition and to input again all data mentioned above. So, it seems worthwhile to establish processes to generate partitions automatically, in which, besides the data defining the boundary, only a parameter representing the degree of refinement of the mesh would be given. In this work we present one such method for automatic generation of triangulations applied to the

case of starshaped, cracked and re-entrant two-dimensional curved domains. Let us first recall the definition of this class of domains, as it is given in [3].

Definition1.Let Ω be a bounded open set of R^n with boundary $\partial\Omega$; Ω is said to be *starshaped* if there exists a point $x_0, x_0 \in \Omega$, such that for every x,x$\in \Omega$, *the* set $\mathscr{L}$

$\mathscr{L}$**={y|y=**αx_0**+(1-** α**)x,for some** $\alpha \in [0, 1]$**}** , is a subset of Ω

It is well known that $\mathscr{L}$ represents the line segment joining x_0 to x, so otherwise stated,

Definition 1 means that for starshaped domains, every line segment joining any point x of Ω to a certain well chosen point x_0, $x_0 \in \Omega$, lies entirely in Ω, as suggested in Fig. 1.

We shall call the set of all such points x_0, the convex kernel of Ω, since it can be proved that it is always convex. We shall

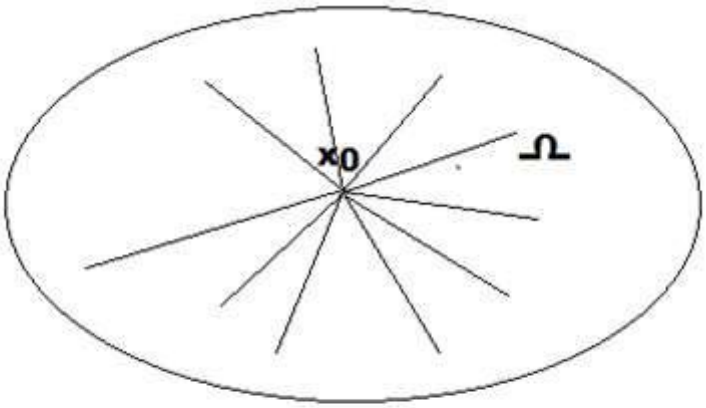


Fig.1 A star shaped domain

denote it by $\mathcal{N}$ and we clearly *have* $\mathcal{N} \subset \Omega$

The applicability of the process is related to $\mathcal{N}$ in the following sense:

If $\mu(\mathcal{N})\#0$, μ***(A)*** being the Borel-Lebesgue measure of a set **A** of R^2, the automatic triangulation method will be feasible. For instance, if Ω is convex, one can easily verify that $\mathcal{N} = \Omega$

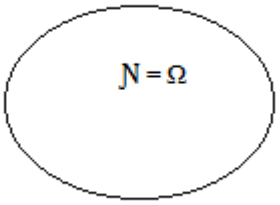


Fig. 2 The convex kernel of Ω,

. In this case the process can certainly be applied (it can be recommended if the convex domain is not too elongated).For non-convex domains whose concavities are not too sharp, the process can still be applied.Now, if the domain has an empty convex kernel, our triangulation m method could only be applied in a modified version . This is the case of the non-convex domain Another case in which the method fails to apply is the one of a non-simply connected domain, also having an empty convex kernel..The applicability of the method depends on the possibility of expressing $\partial\Omega$ by an equation in polar coordinates

$$\rho = f(\theta) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

whose origin O is suitably chosen in $\mathcal{N}$, f; being a mapping from $[0,2\pi]$ to [N, M], a bounded interval of R^+ with N > 0, where

$$N=\min_{0\le\theta\le 2\pi} (f(\theta)) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$M=\max_{0\le\theta\le 2\pi} (f(\theta)) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

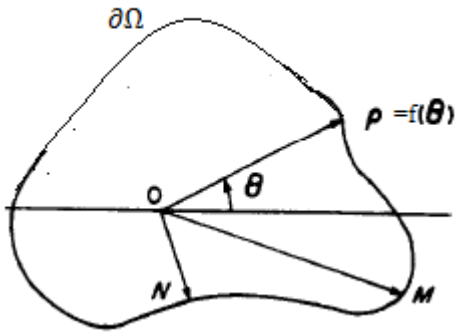


Fig.3 curved domain Ω with boundary defined by polar equation $\rho = f(\theta)$ and convex kernel Ɲ= Ω

Definition 2.A star shaped domain is said to be singular if there exists no mapping f as defined in (1),for any origin lying in Ɲ= Ω Conversely if there exists at least one point O,O∈ Ɲ ,for which the mapping f exists, Ω is said to be non-singular.This is shown in Fig.3.

In this work we present one such a method for automatic generation of triangulations applied specially to the case of star shaped two-dimensional curved domains.

2. Mesh Generation Over Starshaped Domains

Let Ω be a non-singular starshaped domain of R^2. So,we can choose a point O in convex kernel Ɲ= Ω .as the origin and eqn(1) holds.

We next choose an an angle β such that

$$\beta = \frac{2\pi}{n} \qquad .(4)$$

for a given integer n,n≥ 3

Definition 3.Given a positive integer parameter p, we define the vertices $P_{m,l}$ of the **wedge element W** subtending an angle β at (0,0) by the relations(see Fig.):

$P_{0,0} = (0,0); P_{m,l} = (\rho_{ml}, \theta_{ml})$

Where

$$\theta_{ml}=\frac{m\beta}{l}, \text{ m=1,2,3.....,l}$$

$$\rho_{ml}=\frac{lf(\theta_{ml})}{p}, \text{ l=1,2.3,.....,p} \qquad (5)$$

When Ω is a non-singular star shaped domain of R^2 and consists of 'n' **wedge elements**,we have m=1,2,3.....nl. In the following figure, Ω is shown as an assemblage of wedge elements

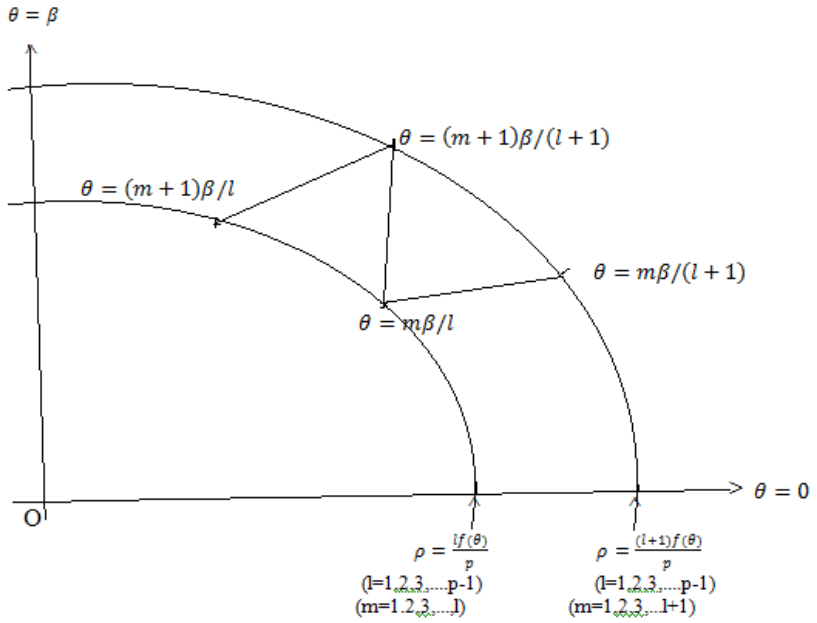
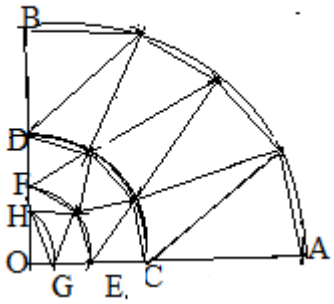


Fig.4 Two consecutive boundary curves of a wedge element

The mesh generation procedure over wedge element is illustrated in the following Fig.4
The subtending angle $\beta = \pi/2$ for this example.The automesh generation scheme will be fully discussed in the next section



,

Fig,5:Homotetically reduced mesh generation over a quarter circle B_i (i=1,2,3,4) boundaries in a quarter circle
B_1 = AB ; B_2 =CD ; B_3 =EF ;B_4 =GH.

We generalise the procedure to generate triangular mesh for an arbitrary curved domain consisting of wedge elements for which one side is the boundary curve defined by the polar equation (1) and the other two sides are straight lines. .First we depict a single wedge element in Fig.6

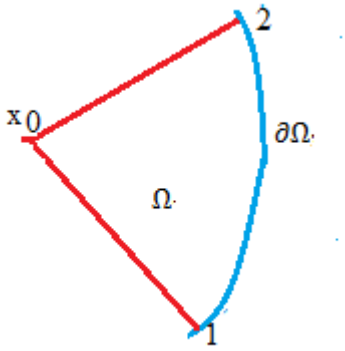


Fig.6: Domain Ω is a single wedge element emanating from x_0

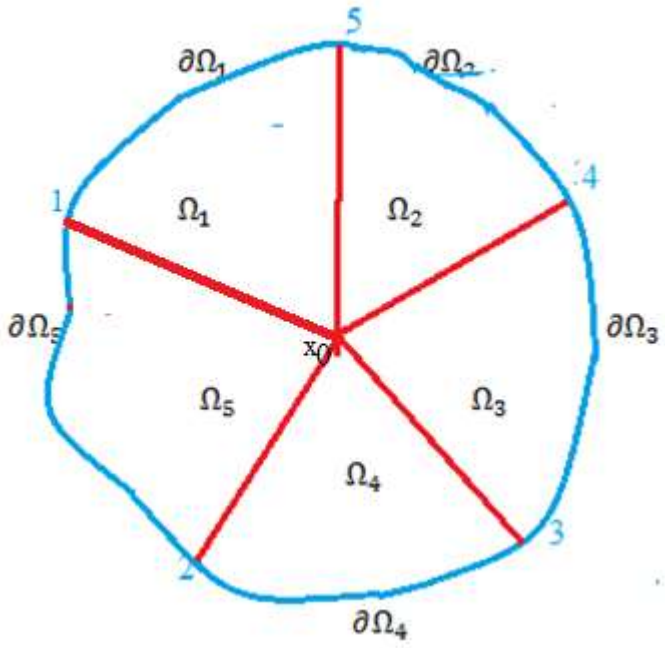


Fig.7: Domain Ω is starshaped and discretized by five wedge elements emanating from x_0
$\Omega=\sum_{i=1}^{i=5}\Omega_i$, *the domain* and $\partial\Omega=\sum_{i=1}^{i=5}\partial\Omega_i$, *the boundary*
We present below an example of non starshaped domain Ω (Fig.7) which can be discretised by three starshaped domains and $\Omega=\sum_{i=1}^{i=3}\Omega_i$,[4]

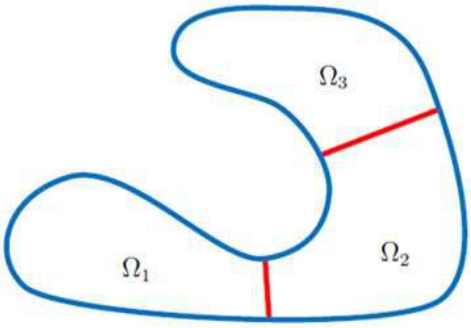


Fig. **8** : Non-star shape case of Ω.

Fig.8: Domain Ω is nonstarshaped and discretized by three starshaped domains $\Omega_i, i = 1, 2, 3$; and $\Omega=\sum_{i=1}^{i=3}\Omega_i$

2.1 Discritisation of Wedge Elements

Wedge elements are curved triangular elements having two straight sides and one curved side.When the curved side is defined by simple quadratic equations,it may possible to replace this curve by a parabolic arc passing through four points,this may be impossible with many curved boundries defined by polar equations.We now propose a method to triangulate these wedge elements or curved triangular elements.We assume that the curved boundary of the wedge element is defined by the polar equation

$$\rho = f(\theta) \qquad (1)$$

Using the above relation, we can generate the Cartesian coordinate on the curved boundary as well as the mesh points interior to the wedge element.

Next, we consider a point $(0,0)\ \epsilon\Omega$ the scaling center, and connect it with the boundary by means of rays that emanate from it.

We take the initial ray OA as $\theta = \theta_0$ and final ray OB as $\theta = \theta_N$.

Let us make 'n' divisions of the subtending angle $(\theta_N - \theta_0)$ along the boundary curve.

Let us now define on the boundary curve

$$\theta_i^{n+1}=\frac{(\theta_N-\theta_0)}{n}(i-1)+\theta_0, \text{i=1,2,3,.....,(n+1)}$$

$$\rho_i^{n+1}=\rho(\theta_i^{n+1}), \text{ i=1,2,3,.....,(n+1)} \qquad (2)$$

Then we reduce this boundary curve by a factor '1/(n+1)' and the (n-1) divisions can be then written as

$$\theta_i^{n}=\frac{(\theta_N-\theta_0)}{(n-1)}(i-1)+\theta_0, \text{i=1,2,3,.....,n}$$

$$\rho_i^{n}=\rho(\theta_i^{n})(\frac{n}{(n+1)}), \text{ i=1,2,3,.....,n} \qquad (3)$$

We shall keep reducing the boundary curve,after (n+1) steps we reach the point O.
Hence, in general we can write:

$$\rho_i^{n+2-j} = \rho(\theta_i^{n+2-j})\ (\frac{n+2-j}{(n+1)}),\ (\text{ i=1,2,3,.....,(n+1-j))};\ \ (\text{j=3,4,.....n})$$

$$\theta_i^{n+1-j}=\frac{(\theta_N-\theta_0)}{(n-j)}(i-1)+\theta_0,\ \ (\text{ i=1,2,3,.....,(n+1-j))};\ \ (\text{j=2,3,.....(n-1)}) \qquad (4)$$

$\theta_1^2=\theta_0, \theta_2^2=\theta_N$

$\rho_1^1 = 0, \theta_1^1=\theta_0$, $\theta_m^m=\theta_N$,m=2,3,.....,(n+1)
Now, we can write the Cartesian coordinate points over the wedge element and they can be computed as:

(i)On the curved boundary of the domain(say first curve)

$$x_i^{n+1} = \rho_i^{n+1}\cos(\theta_i^{n+1}),\ y_i^{n+1} = \rho_i^{n+1}\sin(\theta_i^{n+1}) \ ,$$

(5)

$\theta_i^{n+1}=\frac{(\theta_N-\theta_0)}{n}(i-1)+\theta_0, \rho_i^{n+1}=\rho(\theta_i^{n+1})=f(\theta_i^{n+1})$,

$(i=1,2,3,\dots\dots,n+1)$

(ii)On the first reduced curved boundaries of the domain(say second curve)

$x_i^n = \rho_i^n\cos(\theta_i^n), y_i^n = \rho_i^n\sin(\theta_i^n)$,

(6)

$\theta_i^n=\frac{(\theta_N-\theta_0)}{(n-1)}(i-1)+\theta_0, \rho_i^n=\rho(\theta_i^n)(\frac{n}{(n+1)})=f(\theta_i^n)(\frac{n}{(n+1)})$,
$(i=1,2,3,\dots\dots,n)$

(iii) On the succeeding(3^{rd} to n^{th})reduced curved boundaries of the domain

$x_i^{n+2-j} = \rho_i^{n+2-j}\cos(\theta_i^{n+2-j}), y_i^n=\rho_i^{n+2-j}\sin(\theta_i^{n+2-j})$,

(i=1,2,3,.....,(n+1-j)); (j=3,4,.....n)

(7)

Where,

$\rho_i^{n+2-j} = \rho(\theta_i^{n+2-j})(\frac{n+2-j}{(n+1)})$, (i=1,2,3,.....,(n+1-j)); (j=3,4,.....n)

$\theta_i^{n+1-j}=\frac{(\theta_N-\theta_0)}{(n-j)}(i-1)+\theta_0$, (i=1,2,3,.....,(n+1-j)); (j=2,3,.....(n-1))

We illustrate the above procedure for a single wedge element in Fig.9

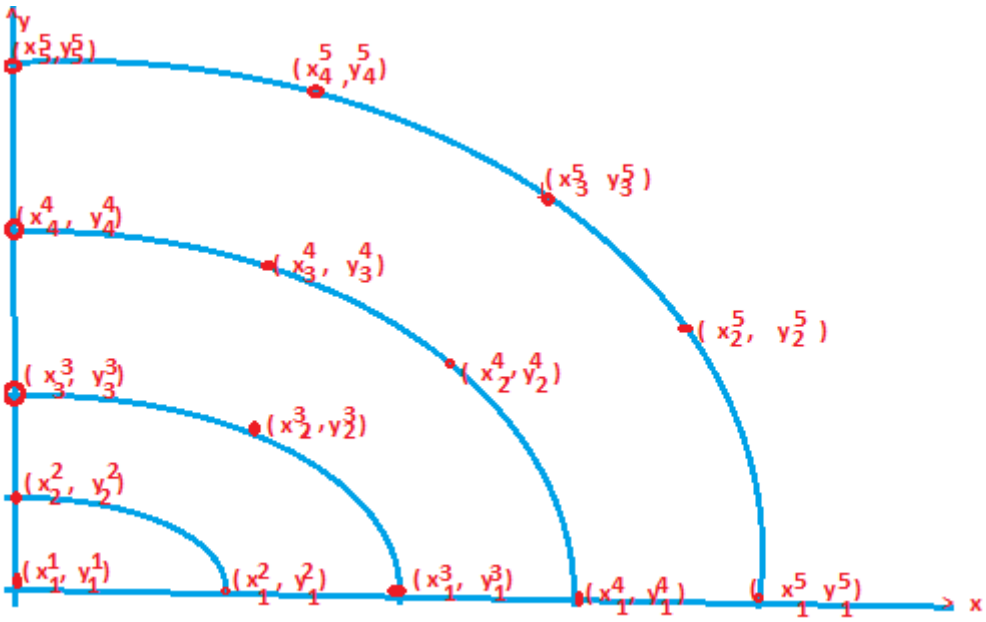


Fig.9: Mesh points over a Wedge Element for n=4

2.2 Generation of Mesh Points and Finite Element meshes over a Wedge Element

We now consider the generation of mesh points finite rlement meshes for curved domain whose boundary is defined by a polar equations.

Example 1:A smooth domain Ω given by the interior of the boundary curve $\partial\Omega$ defined by polar equation

$\rho(\theta)$=0.9+0.1*cos(4*θ) and

$x(\theta)=\rho(\theta)\cos(\theta), y(\theta)=\rho(\theta)\sin\theta\ \theta\in[0,2\pi]$

We discritise the domain Ω into four sub-domains Ω_i ,i=1,2,3,4

$\Omega=\sum_{i=1}^{4}\Omega_i$

Where Ω_i is the sub − domain over ith quadrant
We consider Ω_1 the sub − domain over first quadrant i. e $\theta \in [0, \pi/2]$
Using the above , mesh points , the following triangular and quadrangular meshes on first quadrant were created. This procedure will be further explained in the next section.

Example 2:A smooth domain Ω given by the interior of the boundary curve $\partial\Omega$ defined by polar equation
$\rho(\theta)$=1 and
$x(\theta) = \rho(\theta)\square\cos(\theta)$, $y(\theta) = \rho(\theta)\square\square\sin\square\theta\square\square\theta \in [0,2\pi]$
We discritise the domain Ω into four sub-domains Ω_i ,i=1,2,3,4
$\Omega = \sum_{i=1}^{4} \Omega_i$
Where Ω_i is the sub − domain over ith quadrant
We consider Ω_1 the sub − domain over first quadrant i. e $\theta \in [0, \pi/2]$

We next follow the procedure stated earlier and the following figures were generated by assuming n=4, for Example 2.for illustration,one can compare this with Fig.9

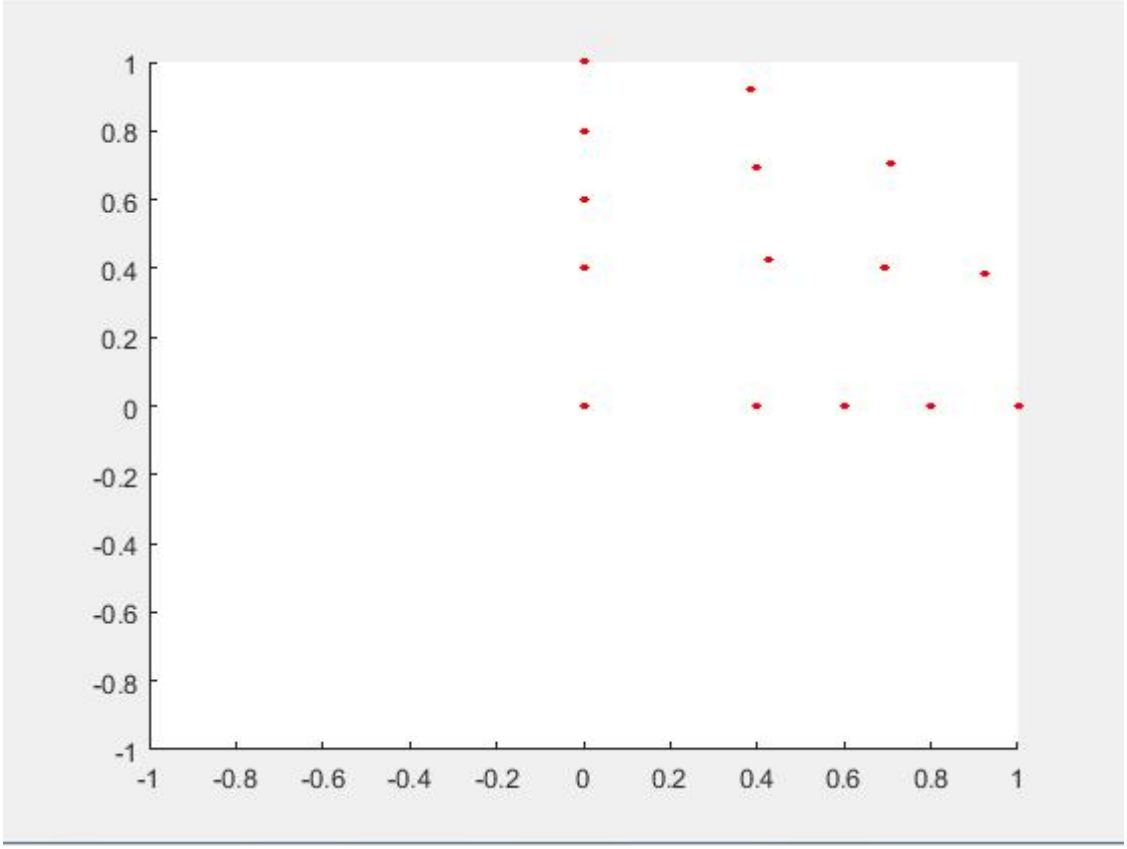


Fig.10a:SCATTERED POINTS IN 2D FOR A QUARTER OF CIRCULAR DOMAIN(n=number of divisions on boundary=4)

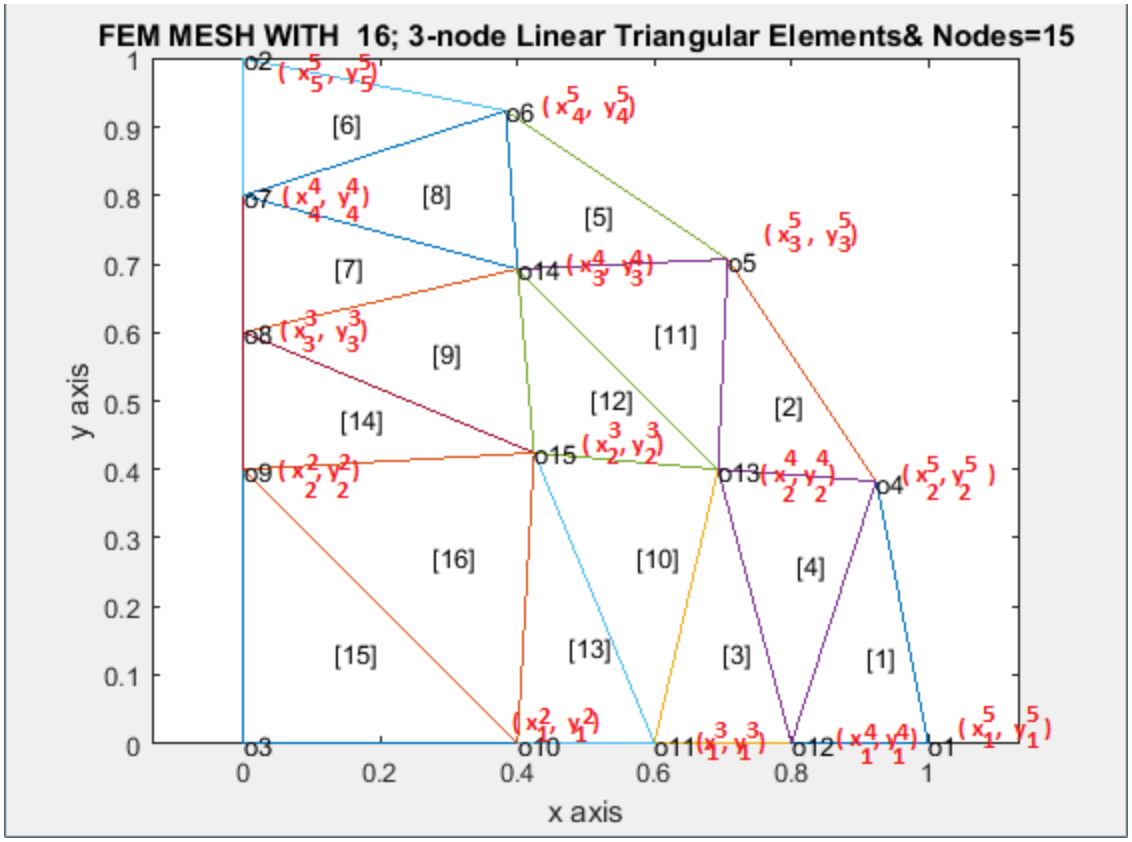


Fig.10b:Triangular Mesh generation Over a quarter circle(n=number of divisions on boundary=4)

We now follow the procedure stated earlier and the following figures were generated by assuming n=6, for Example 1

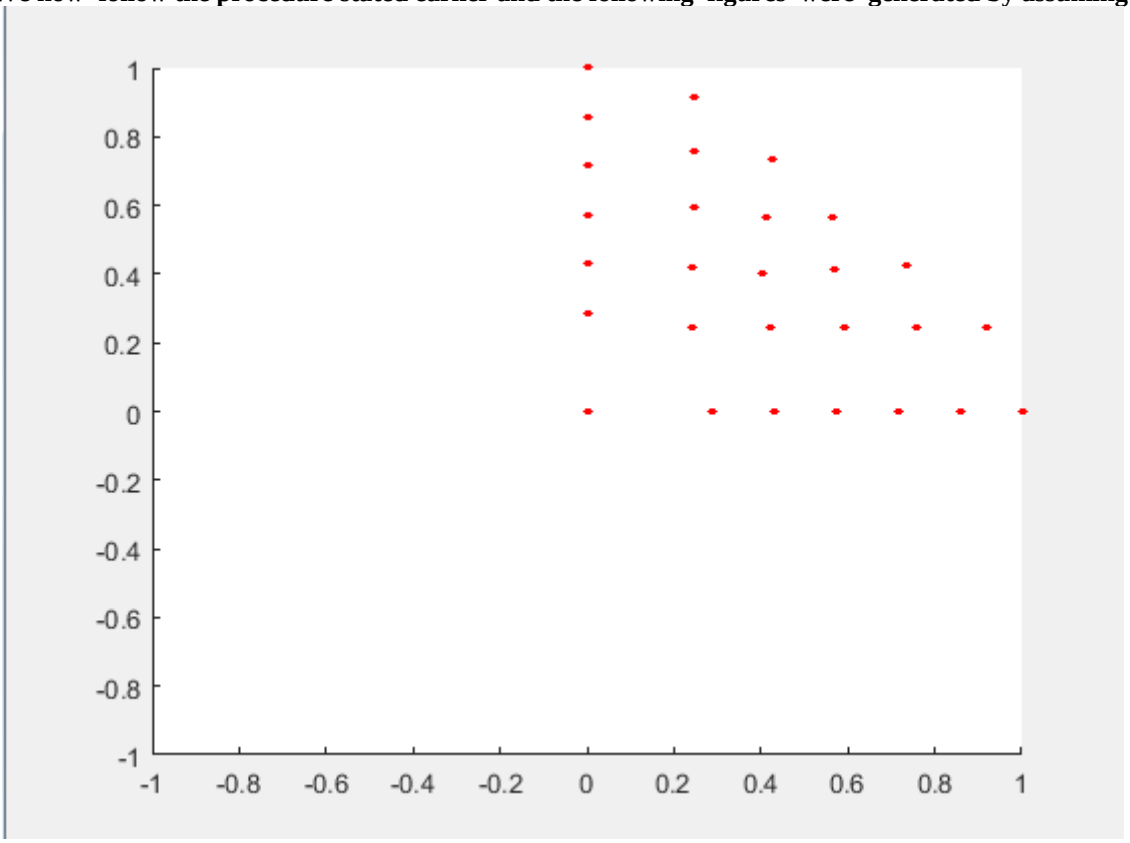


Fig.11a:SCATTERED POINTS IN 2D FOR A QUARTER OF DOMAIN(n=number of divisions on boundary=6)

FEM MESH HAVING 36 LINEAR 3-NODE TRIANGLES AND 28 NODES

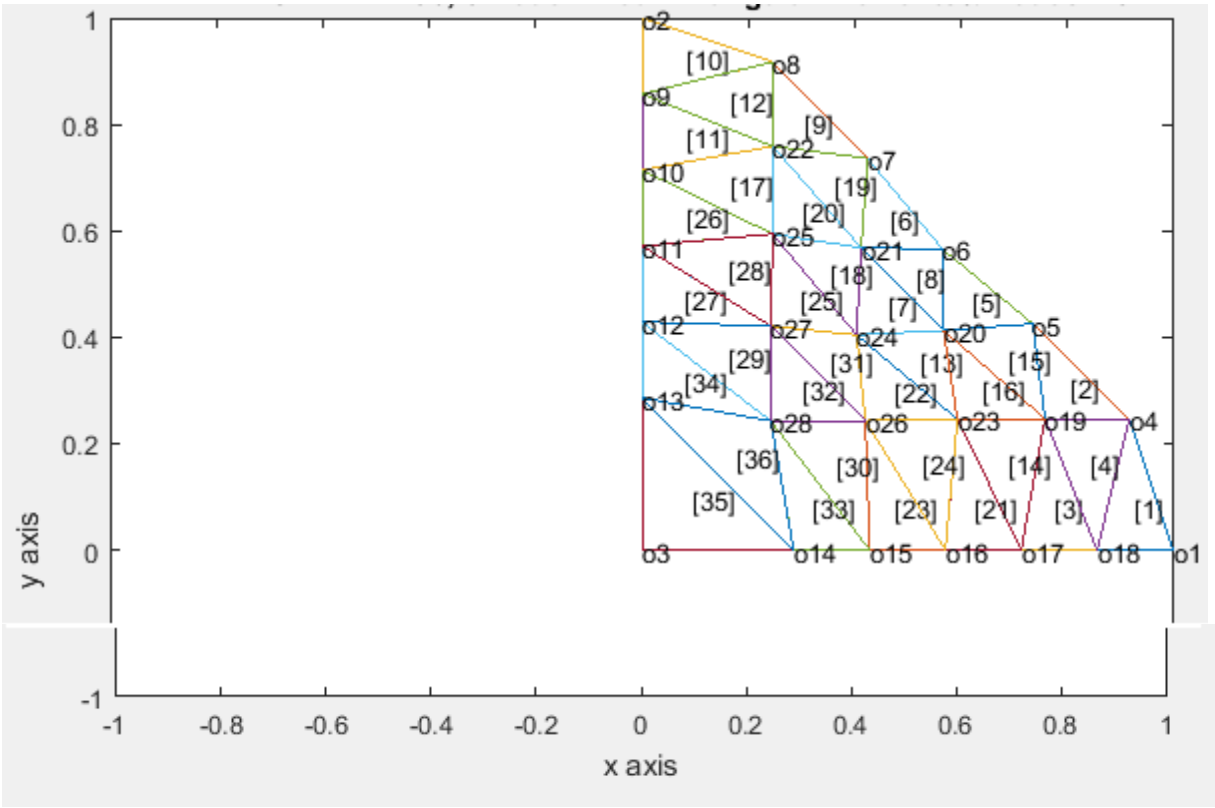


Fig.11b:Triangular Mesh generation(n=number of divisions on boundary=6)

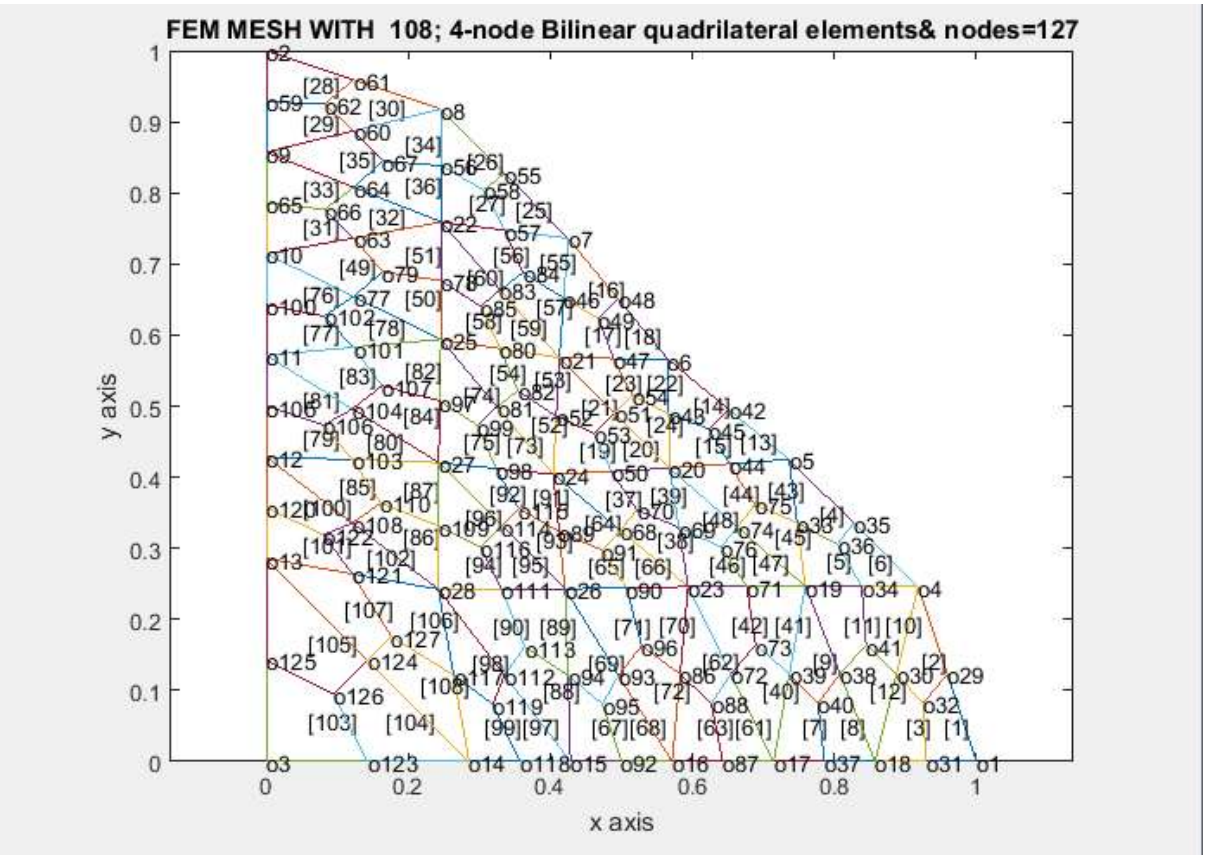


Fig.11c:Special Quadrilateral Mesh generation(n=number of divisions on boundary=6)

We next follow the procedure stated earlier and the following figure was generated by assuming n=10

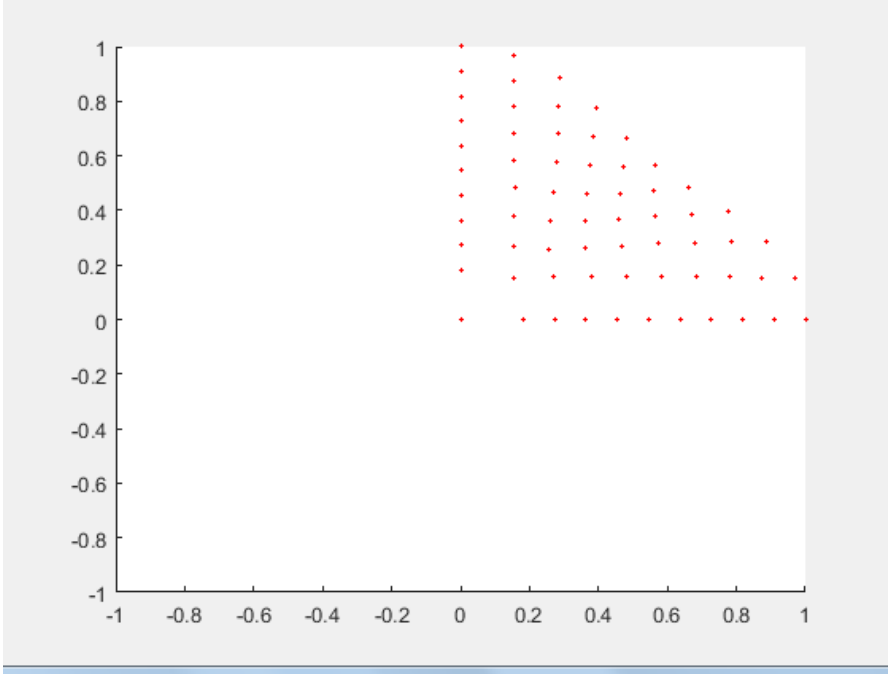


Fig.12a:SCATTERED POINTS IN 2D FOR A QUARTER OF DOMAIN(n=number of divisions on boundary=10)

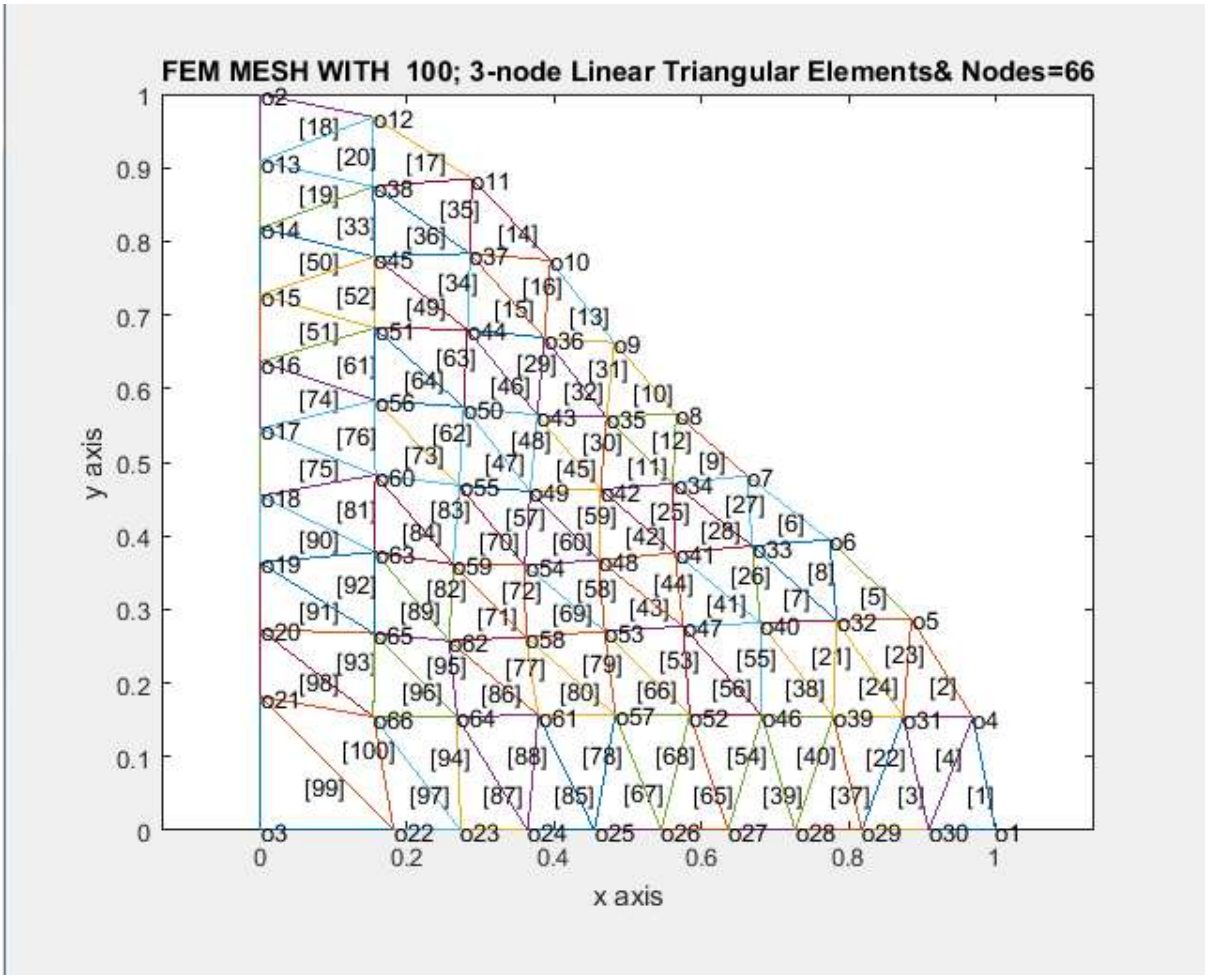


Fig.12b:Triangular Mesh generation(n=number of divisions on boundary=10)

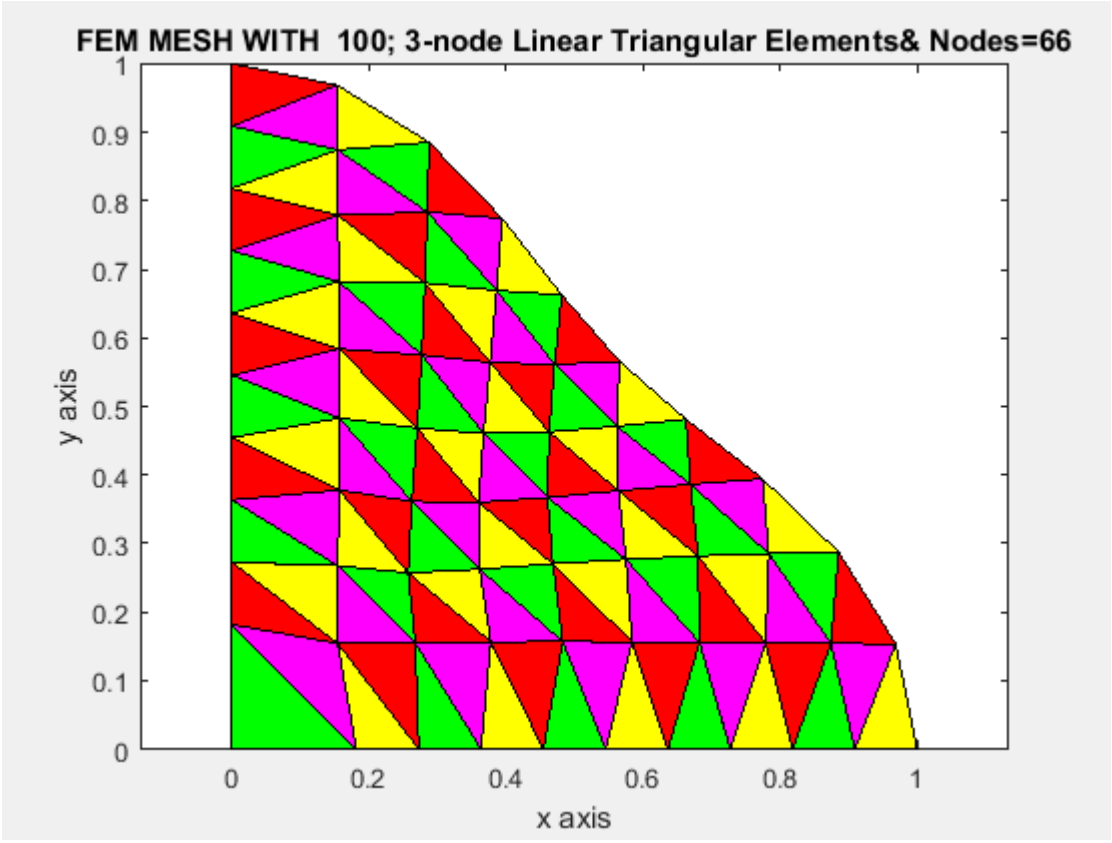


Fig.12c:Triangular Mesh generation(n=number of divisions on boundary=10)in color

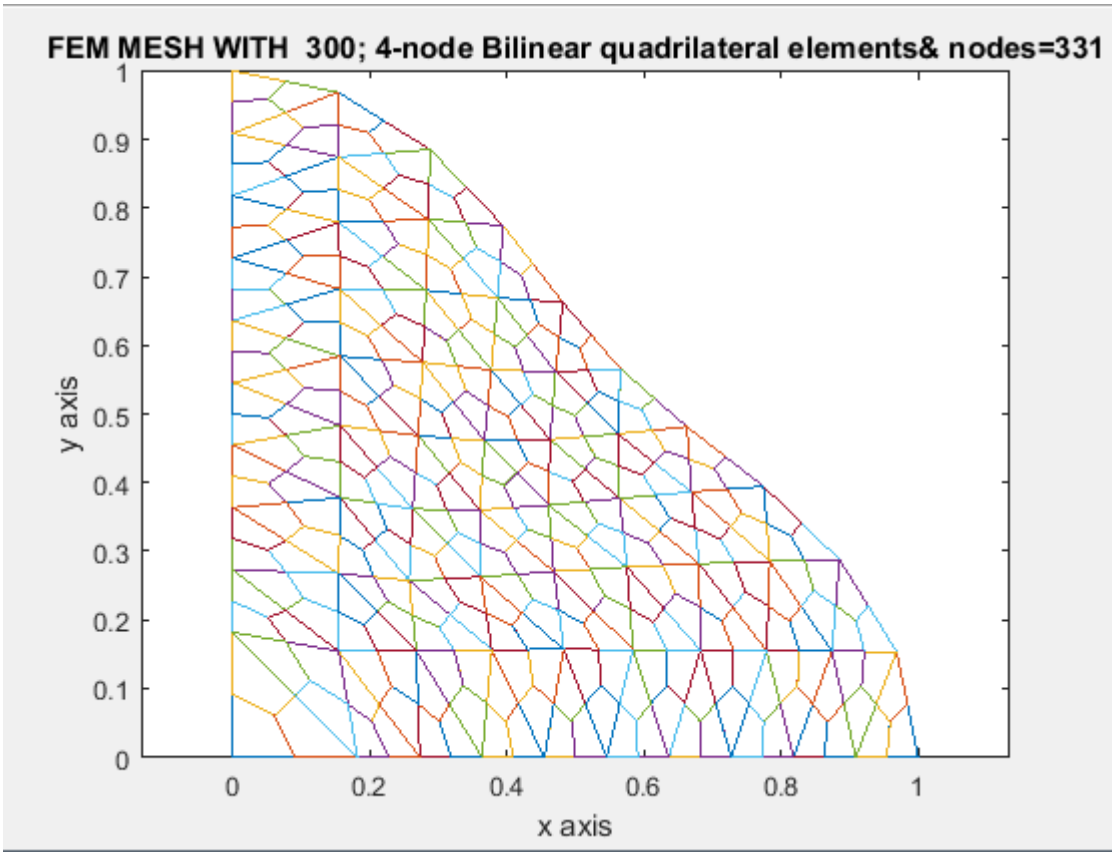


Fig.12d:Special Quadrilateral Mesh generation(n=number of divisions on boundary=10)

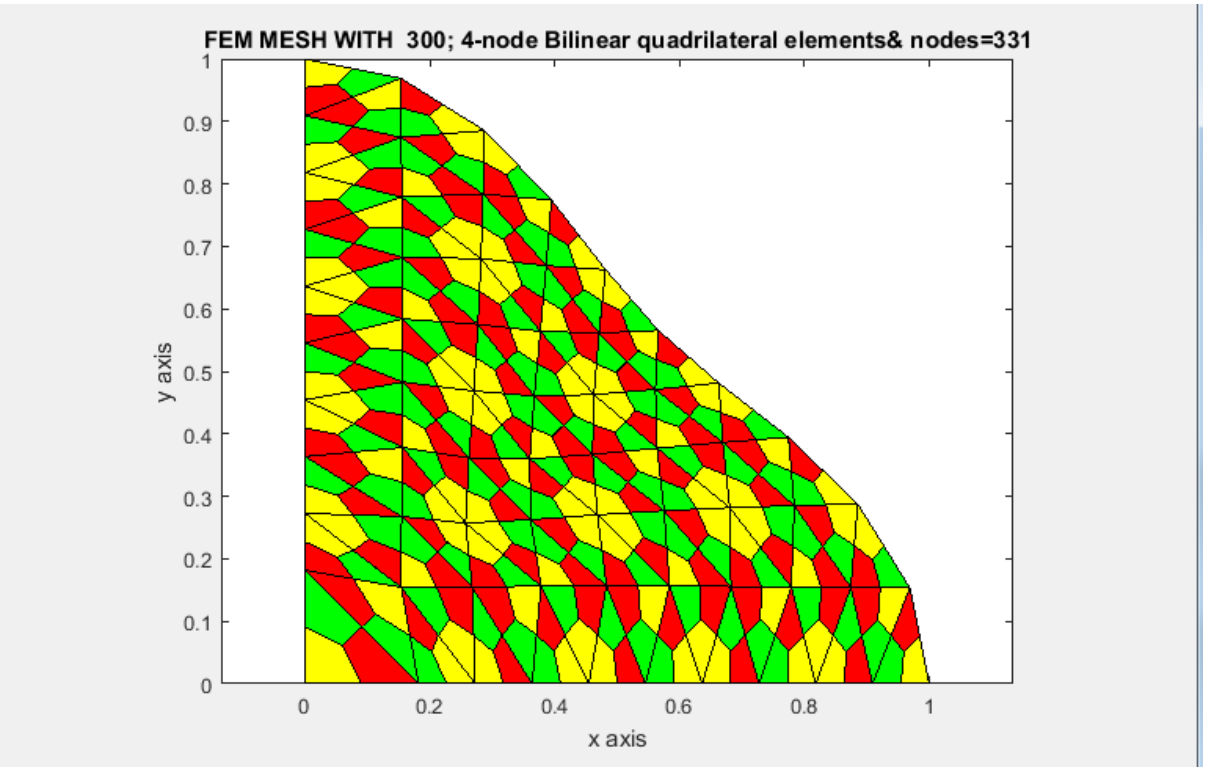


Fig.12e:Special Quadrilateral Mesh generation(n=number of divisions on boundary=10)in colour

3.1 Graded Mesh Generation Over Wedge Elements

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(θ), defined over the entire domain Ω, the task of meshing is to discretize the domain Ω into wedges *W* or quadrilaterals *Q* or combination of both in consistency with the given grading functions g(θ).
The grading functions g*(θ)*, 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. We first consider mesh generation over a single wedge element,because an assemblage of several wedge elements will create an arbitrary star shaped or nonstar shaped domain.
In the notations of the previous section,we can write the new equations by modifying these by including a parameter β ,known as grading factor, we have

$\theta_i^{n+1}=\frac{(\theta_N-\theta_0)}{n}(i-1)+\theta_0, \rho_i^{n+1}=\rho(\theta_i^{n+1})=f(\theta_i^{n+1})$, (i=1,2,3,.....,(n+1)) (8)

to describe the boundary curve

Then we reduce this boundary curve by a factor '1/(n+1)' and the (n-1) divisions of θ can be then written as

$\theta_i^{n}=\frac{(\theta_N-\theta_0)}{(n-1)}(i-1)+\theta_0$,i=1,2,3,.....,n

(9)

$\rho_i^{n}=\rho(\theta_i^{n})(\frac{n}{(n+1)})^{\beta}$, i=1,2,3,.....,n

Where β is known as a grading factor and in general the succeeding reduced boundaries are defined as

$\rho_i^{n+2-j}=\rho(\theta_i^{n+2-j})(\frac{n+2-j}{(n+1)})^{\beta}$, (i=1,2,3,.....,(n+1-j)); (j=3,4,.....n) ,$\beta>0$

$\theta_i^{n+1-j}=\frac{(\theta_N-\theta_0)}{(n-j)}(i-1)+\theta_0$, (i=1,2,3,.....,(n+1-j)); (j=2,3,.....(n-1))

(10)

$\theta_1^2=\theta_0, \theta_2^2=\theta_N$

$\rho_1^1=0, \theta_1^1=\theta_0, \theta_m^m=\theta_N$,m=2,3,.....,(n+1)

We may note that $\beta=1$,generates a uniform mesh.

We have already displayed the figures to generate the mesh points and finite element meshes on a wedge element for Examples 1 and 2.We now elaborate the algorithm.

3.2 :Nodal Connectivity over Wedge Eleements

We consider a wedge element having two straight sides and one curved side.We start node numbering first along the curved side in an anticlockwise direction.Along the curved side the starting node is '1 ' and the end node is '2',these nodes are joined to a center node '3 '.This describes the wedge element with a boundary curve joining nodes 1 and 2,and two straight sides joining nodes 2 and 3 and another straight side joining nodes 1 and 3
The mesh points are generated in (n+1) rows:
(i)The first row containing '(n+1)' mesh points are for the curved boundary joining nodes '1' and '2'.They are:
1,4,5,6,7,.....,(n-1),n,(n+1),(n+2),2
(ii) The second row containing 'n' mesh points are for the first reduced curved boundary joining nodes '3n' and '(n+3)'.They are:
3n,3n+1,.....,.....,..... $3n+(n-2)$, $(n+3)$

(iii) The third row containing '(n-1)' mesh points are for the second reduced curved boundary joining nodes '3n-1' and '(n+4)'.They are:

$$3n-1, 3n+(n-1), \ldots\ldots\ldots\ldots, 3n+(n-2)+(n-3), (n+4)$$

.....
and the '(n-1)th, 'n'th and '(n+1)'th rows will contain 3,2,and 1 mesh points

$3n-(n-3)$, $\frac{(n+1)(n+2)}{2}$, $2n$
$3n-(n-2)$, $(2n+1)$

3
respectively.

This creates (n+1) (n+2)/2 nodes. . Thus the entire wedge or curved 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 curved triangle

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

...............(10)

We have displayed the above procedure of node numbering in the following figure:

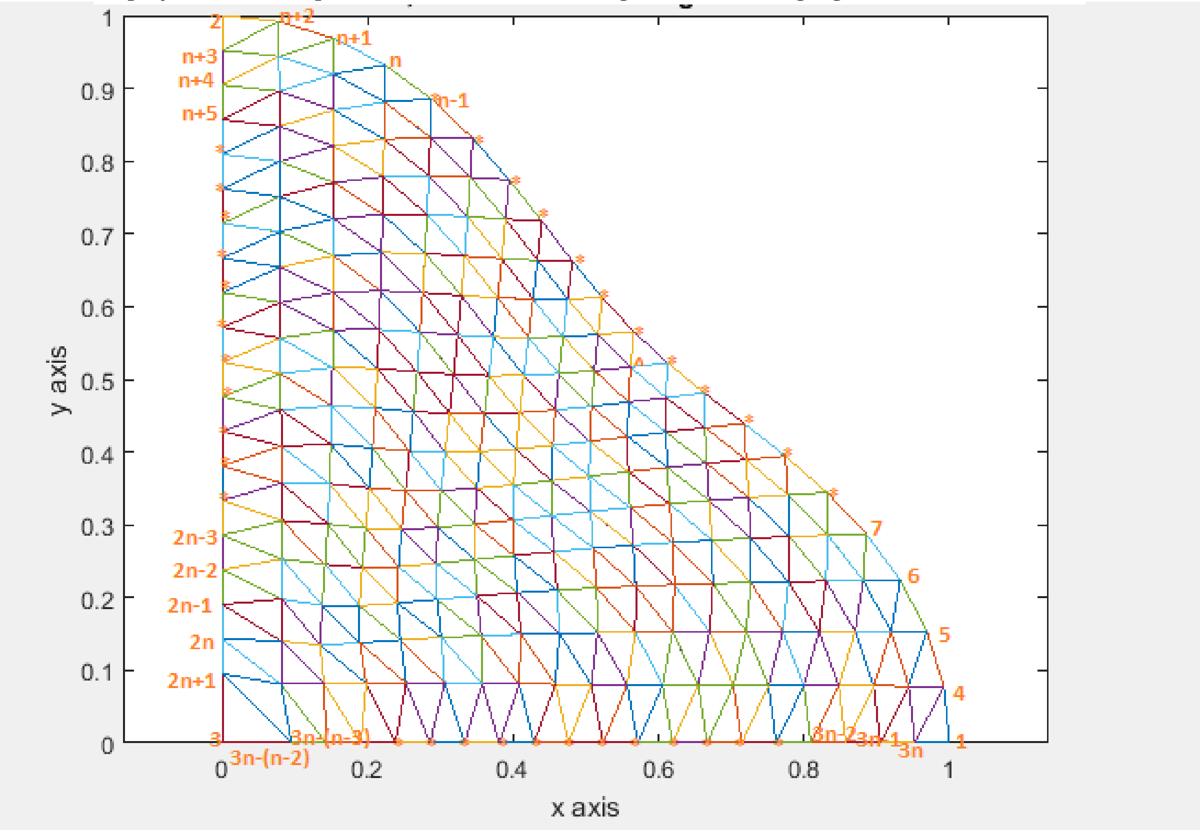


Fig.13: Node numbering for an arbitrary wedge element defined over first quadrant

In general, we note that to divide a wedge (a curved triangle with two straight sides) into six node polygons, we must divide each side of the triangle into an even number of divisions and locate points in the interior of the wedge at equal spacing. Thus n (even) divisions creates ($n/2)^2$ six node polygons.. If the entries of the sub matrix $\underline{rr}\,(i : i+2,\ j : j+2)$ are nonzero then two six node polygons can be formed. If $\underline{rr}\,(i+1,\ j+2) = \underline{rr}\ (i+2,\ j+1; j+2) = 0$ then one six node polygon 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 polygons. We now explain the creation of the six node polygons using the $\underline{rr}$ matrix of eqn.( ). We can form six node polygons by using node points of three consecutive rows and columns of $\underline{rr}$ matrix. This procedure is depicted in Fig.14 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 polygons using sub matrix $\underline{rr}$

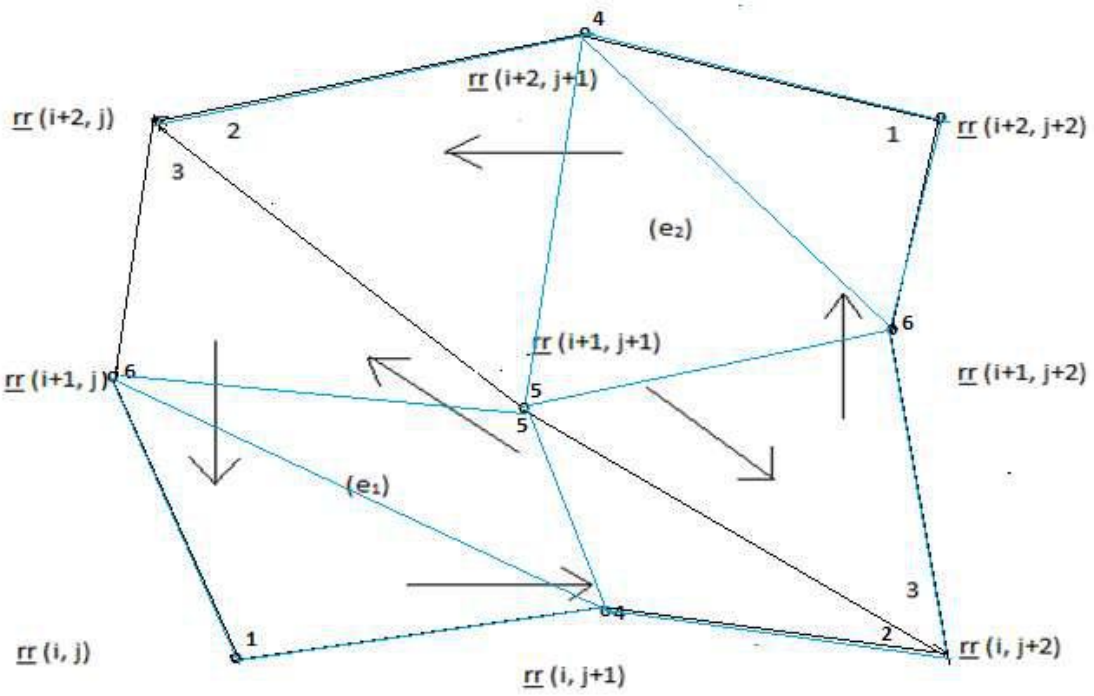


Fig.14: Six node polygon formations for non zero sub matrix $\underline{rr}$:Linear Triangles

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 polygons. The element nodal connectivity is then given by , wherein we read {... ... ...} as connecting the nodal addresses

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

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

$(i+1, j+2)$ }

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 polygons. In the present case,we cannot create three special quadrilaterals in each of these six node polygons. By special quadrilateral,we mean those quadrilateral having the Jacobian expression obtained by applying the bilinear isoparametric coordinate transformation to a straight edge quadrilateral[10].

$C(4 + \xi + \eta)$, $-1 \le \xi, \eta \le 1$, $in\ the\ natural\ space$.where C is some suitable constant.

In the above Fig. ,we have created eight linear triangles.We can divide each of these triangles into three special quadrilaterals by choosing centroidal points and inserting midside nodes to these linear triangles.This is shown in the following Fig.15 .

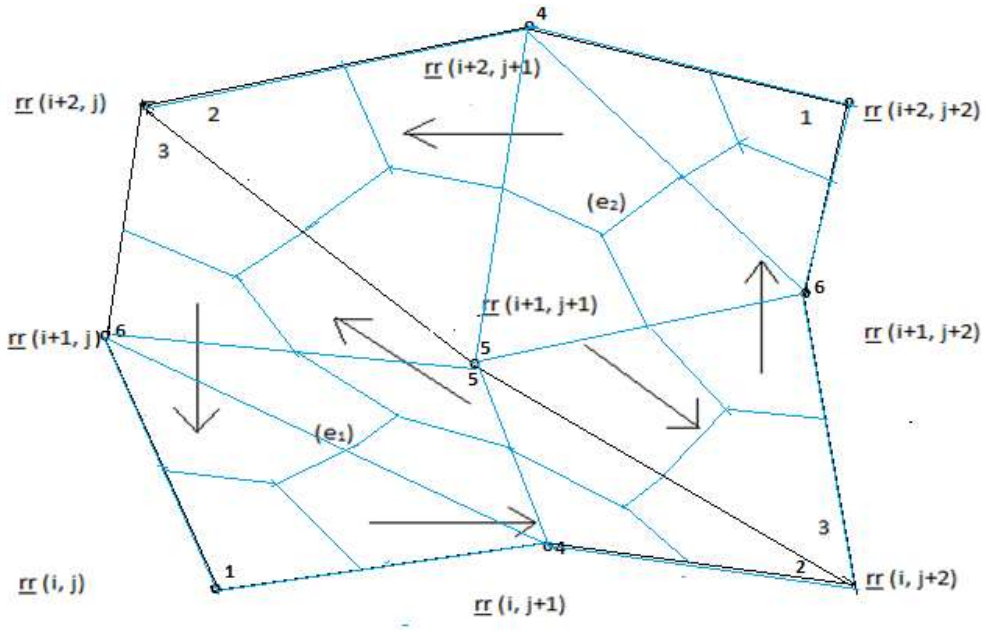
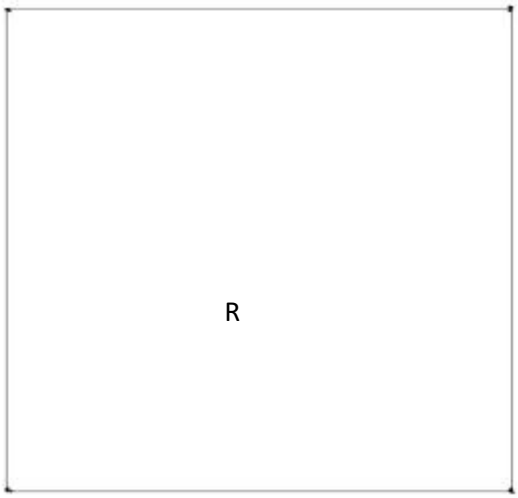


Fig.15: Six node polygon formations for non zero sub matrix $\underline{rr}$:Special Quadrilaterals

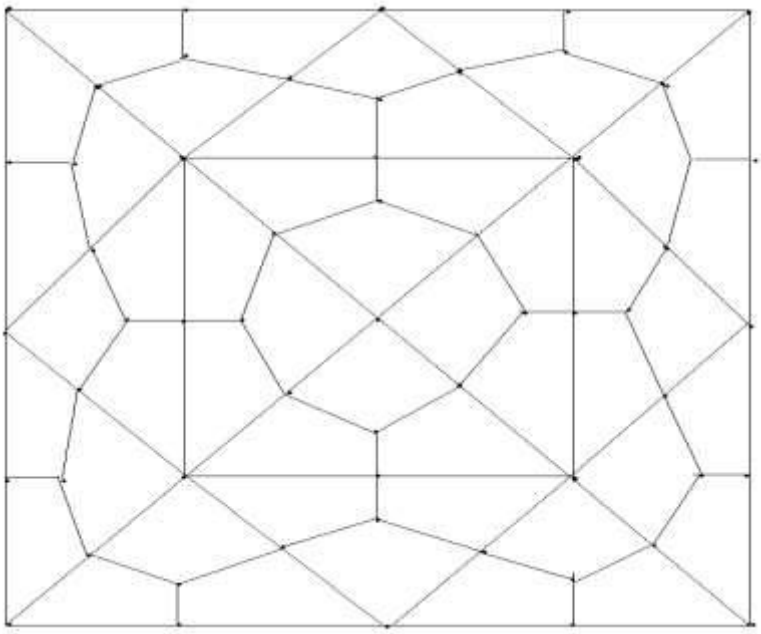
4. Triangulation and Quadrangulation of the Wedge Elements[1,2,11,12,13,14]

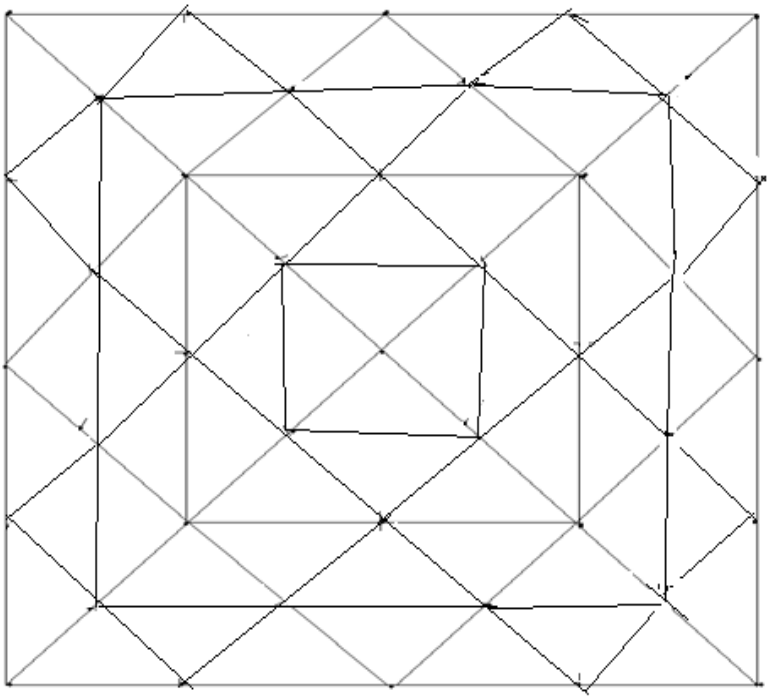
We can generate polygonal and analytical curved surface meshes by piecing together triangles with straight sides (linear triangle) and curved sides(curved triangle) respectively by using subsections (called LOOPs). The user specifies the shape of these LOOPs by designating six coordinates of each LOOP

As an example, consider the geometry shown in Fig. 16(a). This is a rectangular region which is simply chosen for illustration. We divide this region into four LOOPs as shown in Fig.16(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. 16(c).The complete mesh is shown in Fig.16(b)



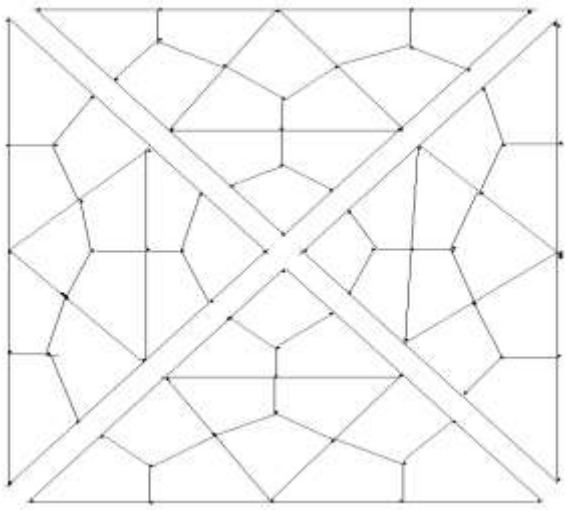
(i)Fig. 16(a) Region R to be analyzed

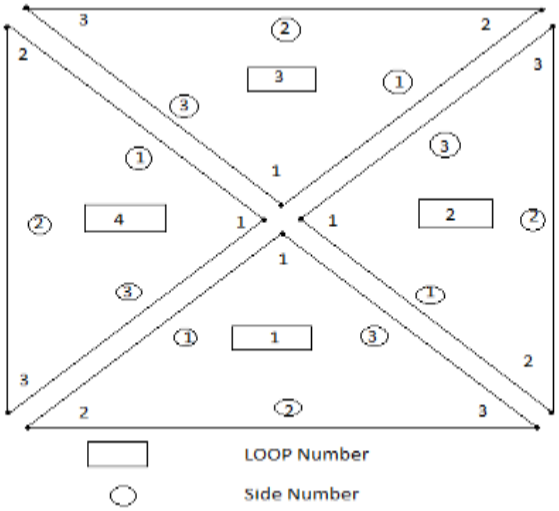




(ii) Fig. 16(b) Example of complete quadrangulated mesh and

(ii) Fig. 16(c) Example of complete triangulated mesh





(iii)Fig.16(d) Exploded view showing four loops

(iv)Fig.16(e) Example of a loop and side numbering scheme

We next illustrate the above procedure for an analytical curved surface ,this is shown in Figs.17a-17d ,with reference to an elliptical region.

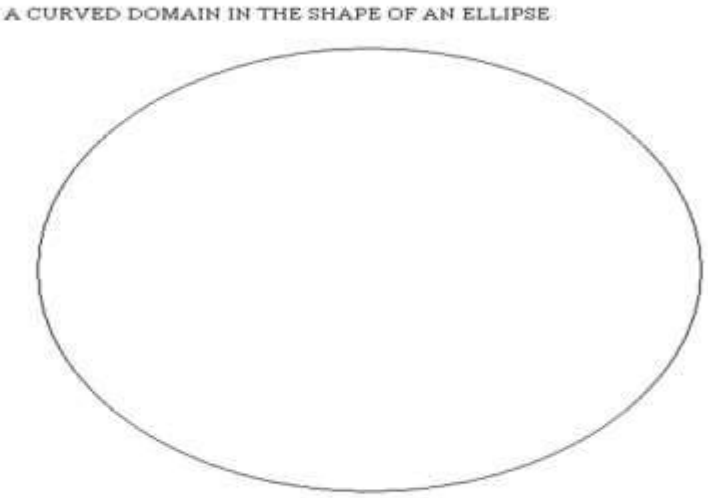


Fig.17a

AN ELLIPSE AS A CURVED DOMAIN MADE UP OF FOUR CURVED TRIANGLES

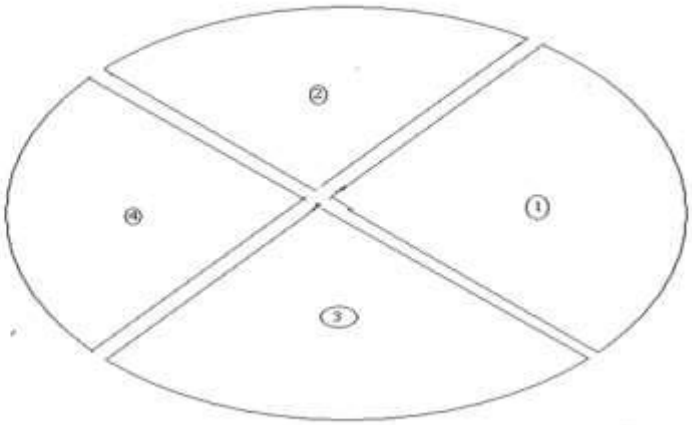


Fig.17b

ELLIPSE AS A CURVED DOMAIN MADE UP OF TWELVE QUADRILATERALS

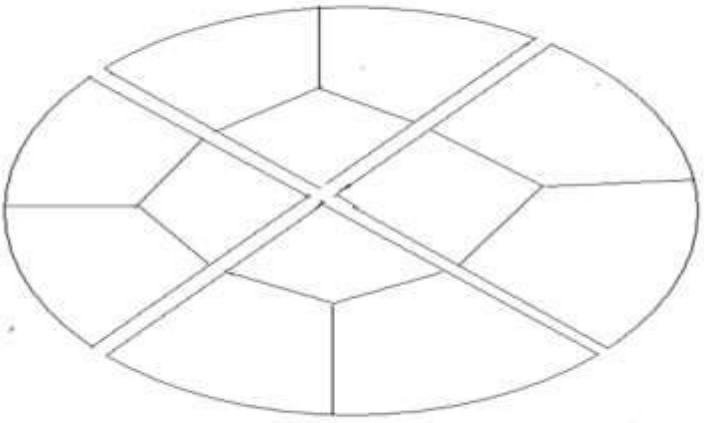


Fig. 17c

AN ELLIPSE AS CURVED DOMAIN DISCRITISED INTO TWELVE QUADRILATERALS

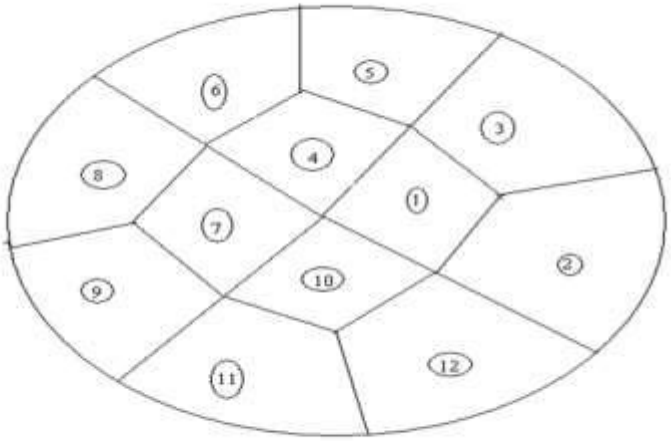


Fig.17d

AN ELLIPSE AS A CURVED DOMAIN DISCRETISED INTO SIXTEEN TRIANGLES

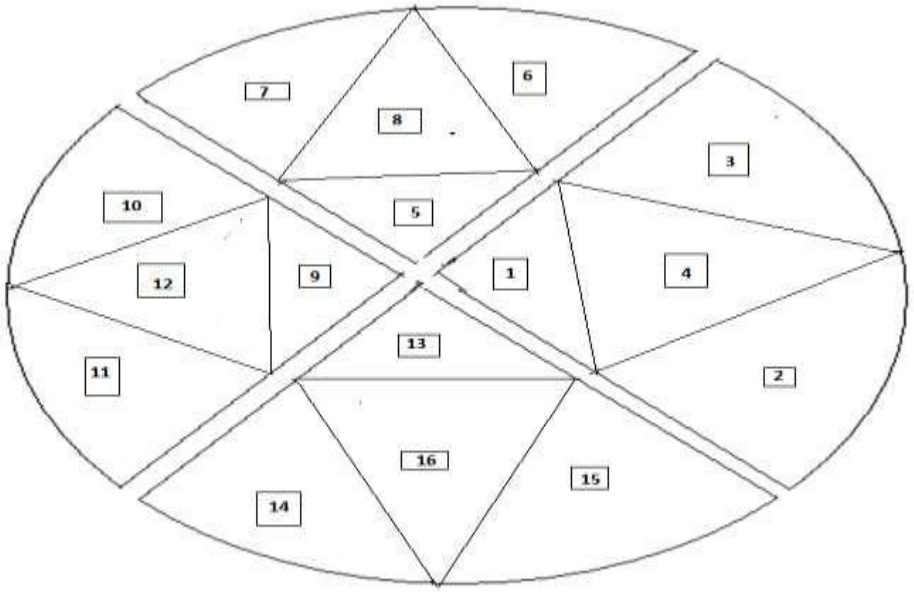


Fig 17e
COMPLETE TRIANGULATED MESH FOR AN ELLIPSE

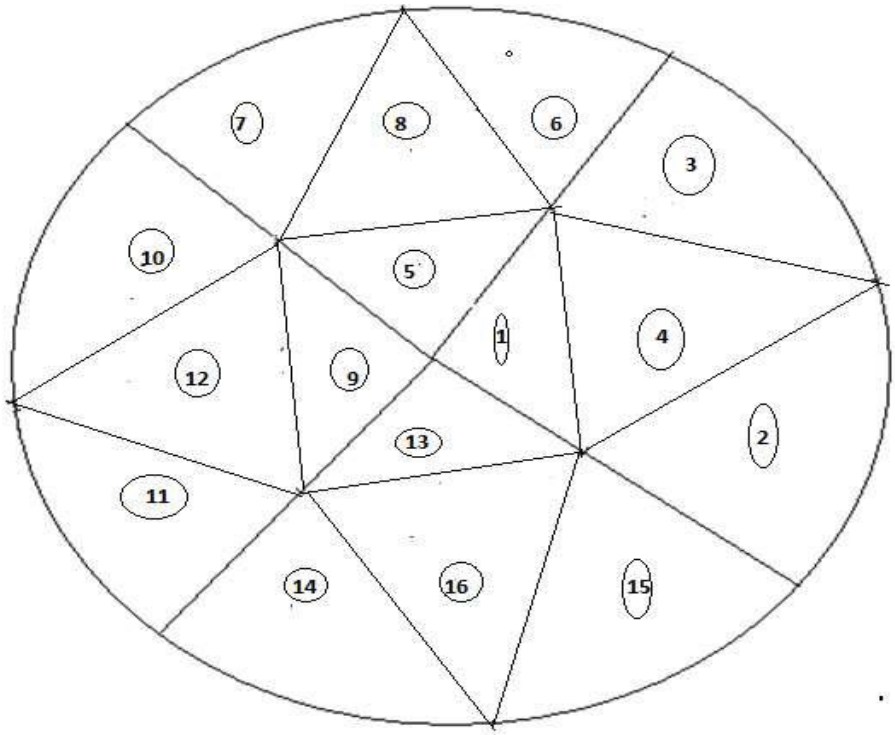


Fig.17f

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 an all quadrilateral mesh for an analytical curved surface. 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 finite element connectivity

5. Applications Of Automesh Generation Scheme

In authors' recent works[12,13,14], an automatic indirect quadrilateral mesh generator which uses the splitting technique is presented for the two dimensional convex domains.It presents the mesh generation over an arbitrary linear triangle and also the mesh generation for a convex polygonal domain and in a recent paper, a new mesh generation method for a simply connected curved domain of a planar region which has curved boundary described by one or more analytical equations in Cartesian form. This paper presents a new mesh generation method for a simply connected curved domain of a planar region whose boundary is defined by one or more equations in polar coordinates.In earlier sections,we have explained the the mesh generation scheme.The following examples are considered

5.1 Examples of Polar Equations for the Curved Domain[7,8,9]

(1)Cross-section of the bar whose outer periphery is defined by the equations
$\rho(\theta) = 0.9 + 0.1\cos(4\theta)$,
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$,-1 $\leq x, y \leq 1$
(2) A circular disk with unit radius whose outer periphery is defined by the equations
$\rho(\theta) = 1$, x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$, ,-1 $\leq x, y \leq 1$
(3)An ellipse whose outer periphery is defined by equations
x(θ) = $a\cos(\theta)$, y(θ) = $b\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$,-a $\leq x \leq a$,-b $\leq y \leq b$,a$\neq b$
where a and b are constants and known as lengths of major and minor axes
(4) Cross-section of the ***hypotrochoid bar*** whose outer periphery is defined by the equations
$\rho(\theta) = \sqrt{9.25 + 3\cos(4\theta)}$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$
(5) ***A kite-shaped bar*** defined by the equations
$\rho(\theta) = \sqrt{(0.6\cos(\theta) + 0.3\cos(2\theta) - 0.2)^2 + 0.36(sin^2(\theta)}$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$
(6) Cross-section of **the epitrochoid-shape bar** whose outer periphery is defined by the equations
$\rho(\theta) = \sqrt{5 - 4\cos(\theta)}$, x($\theta$)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
(7) Cross-section of the ***epitrochoid-shape bar*** whose outer periphery is defined by the quations
$\rho(\theta) = \sqrt{10 - 6\cos(2\theta)}$, x($\theta$)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
(8) Cross-section of the ***epitrochoid-shape bar*** whose outer periphery is defined by the equations
$\rho(\theta) = \sqrt{17 - 8\cos(3\theta)}$, x($\theta$)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
(9)An **asteroid shaped bar** whose outer periphery is defined by the equations
$x(\theta) = (3\cos(\theta) + \cos(3\theta))/4$
$y(\theta) = (3\sin(\theta) - \sin(3\theta))/4$
(10)A **cassini oval shaped bar** whose outer periphery is defined by the equations
$\rho(\theta) = \sqrt{\cos(2\theta) + \sqrt{((1.1)^4 - (\sin(2\theta))^2)}}$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$
(11)A **cranioid shaped bar** whose outer periphery is defined by the equations
$\rho(\theta) = 0.25\sin(\theta) + 0.5\sqrt{1 - 0.9(\cos(\theta))^2} + 0.5\sqrt{1 - 0.7(\cos(\theta))^2}$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$
(12) Cross-section of the ***hypotrochoid bar*** whose outer periphery is defined by the equations
$\rho(\theta) = \sqrt{(9.25 + 3\cos(4\theta))/12.25}$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
$\theta\epsilon[0\ ,2\pi]$
(13)A **rhombus shaped bar** whose outer periphery is defined by the equations
$\rho(\theta)$ =1/(cos(θ)+sin(θ)),$\theta \in [0, \pi/2]$
$\rho(\theta) = -1$/(cos(θ)-sin(θ)),$\theta \in [\pi/2, \pi]$

$\rho(\theta) = -1/(\cos(\theta)+\sin(\theta)), \theta \in [\pi, 3\pi/2]$
$\rho(\theta) = 1/(\cos(\theta)-\sin(\theta)), \theta \in [3\pi/2, 2\pi]$
$x(\theta)= \rho(\theta)\cos(\theta)$, $y(\theta)= \rho(\theta)\sin(\theta)$
and spanned by the vertices{(1 ,0), (0 ,1) , (-1 ,0) ,(0 ,-1)} in Cartesian space (x,y)
(14) A **square shaped bar** $-1 \leq x, y \leq 1$ whose outer periphery is defined by the equations:
$\rho(\theta) = 1/\cos(\theta), \theta \in [-\pi/4, \pi/4]$
$\rho(\theta) = 1/\sin(\theta), \theta \in [\pi/4, 3\pi/4]$
$\rho(\theta) = -1/\cos(\theta), \theta \in [3\pi/4, 5\pi/4]$
$\rho(\theta) = -1/\sin(\theta), \theta \in [5\pi/4, 7\pi/4]$
$x(\theta)= \rho(\theta)\cos(\theta)$, $y(\theta)= \rho(\theta)\sin(\theta)$
and spanned by the vertices{(1 ,-1), (1 ,1) , (-1 ,1) ,(-1 ,-1)} in Cartesian space (x,y)
(15) An **arbitrary shaped curved bar** $-1 \leq x, y \leq 1$ whose outer periphery is defined by the equations:
$\rho(\theta) = 0.9 + 0.1\cos(4\theta), \theta \in [0, \pi/2]$
$\rho(\theta) = 1, \theta \in [\pi/2, \pi]$
$\rho(\theta) = -1/(\cos(\theta)+\sin(\theta)), \theta \in [\pi, 3\pi/2]$
$\rho(\theta) = \sqrt{(9.25 + 3\cos(4\theta))/12.25}, \theta \in [3\pi/2, 2\pi]$
$x(\theta)= \rho(\theta)\cos(\theta)$, $y(\theta)= \rho(\theta)\sin(\theta)$
and spanned by the vertices{(1 ,0), (0 ,1) , (-1 ,0) ,(0 ,-1)} in Cartesian space (x,y)

5.2 Auto generation of Finite Element Meshes
We now display the all triangular and all quadrilateral finite element meshes for the above examples of polar equations for curved domain.The Cartesian coordinates of the mesh points can be obtained by usual equations: $x(\theta)= \rho(\theta)\cos(\theta)$, $y(\theta)= \rho(\theta)\sin(\theta)$, $\theta \epsilon [0, 2\pi]$
Where $\rho(\theta)$ will be different for each example.

Example (1) $\rho(\theta) = 0.9 + 0.1\cos(4\theta)$

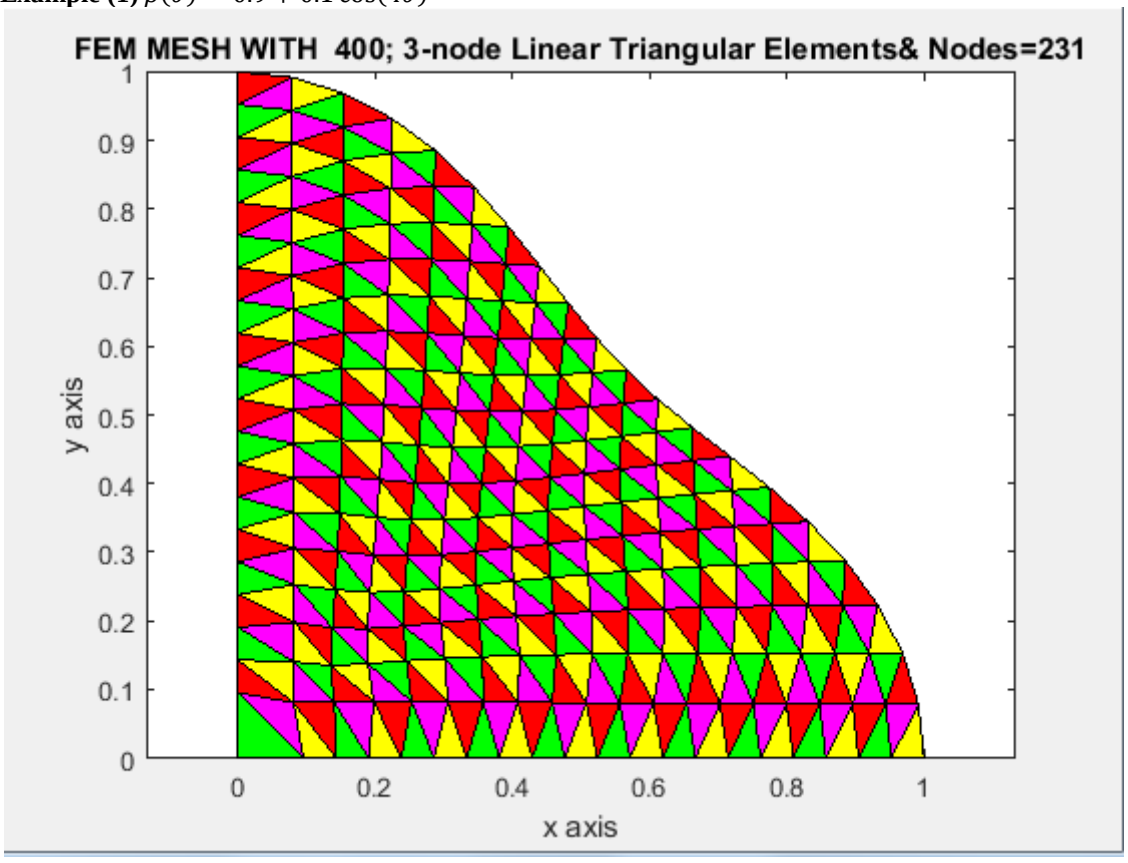


Fig.18a

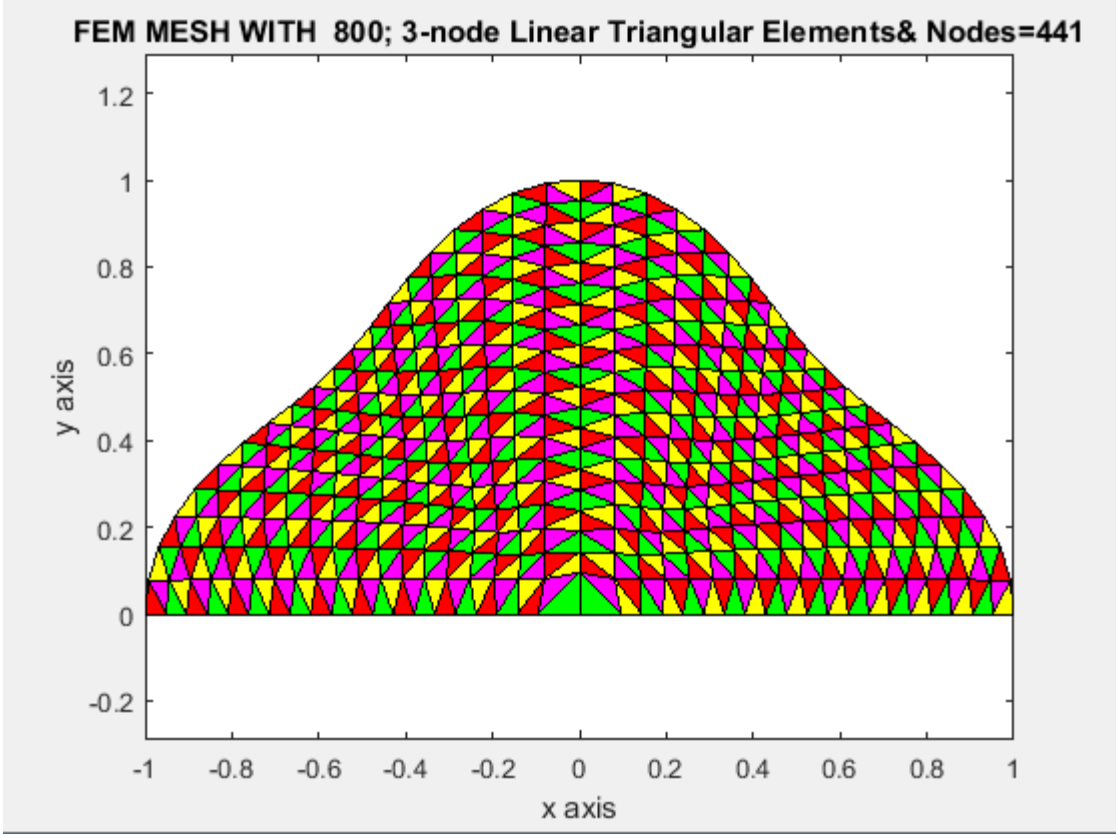


Fig.18b

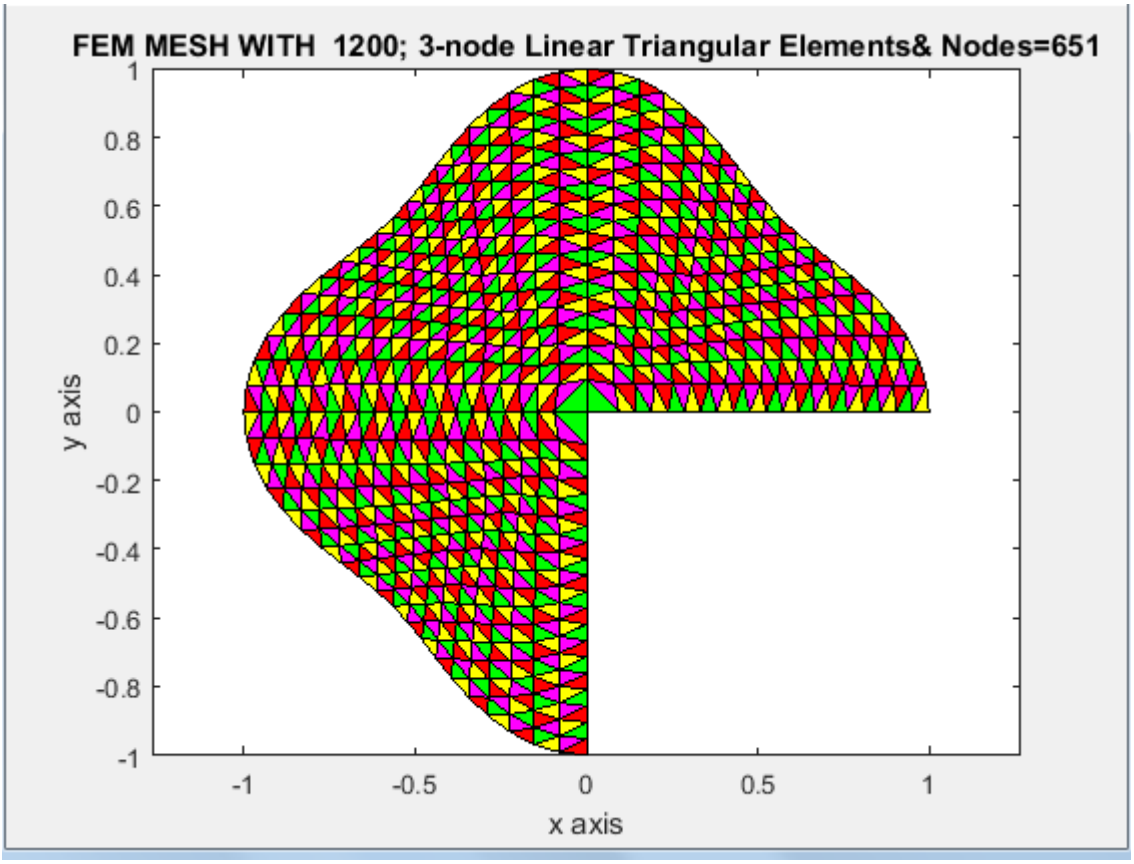


Fig.18c

FEM MESH WITH 1600; 3-NODE LINEAR TRIANGULAR ELEMENTS &NODES=860

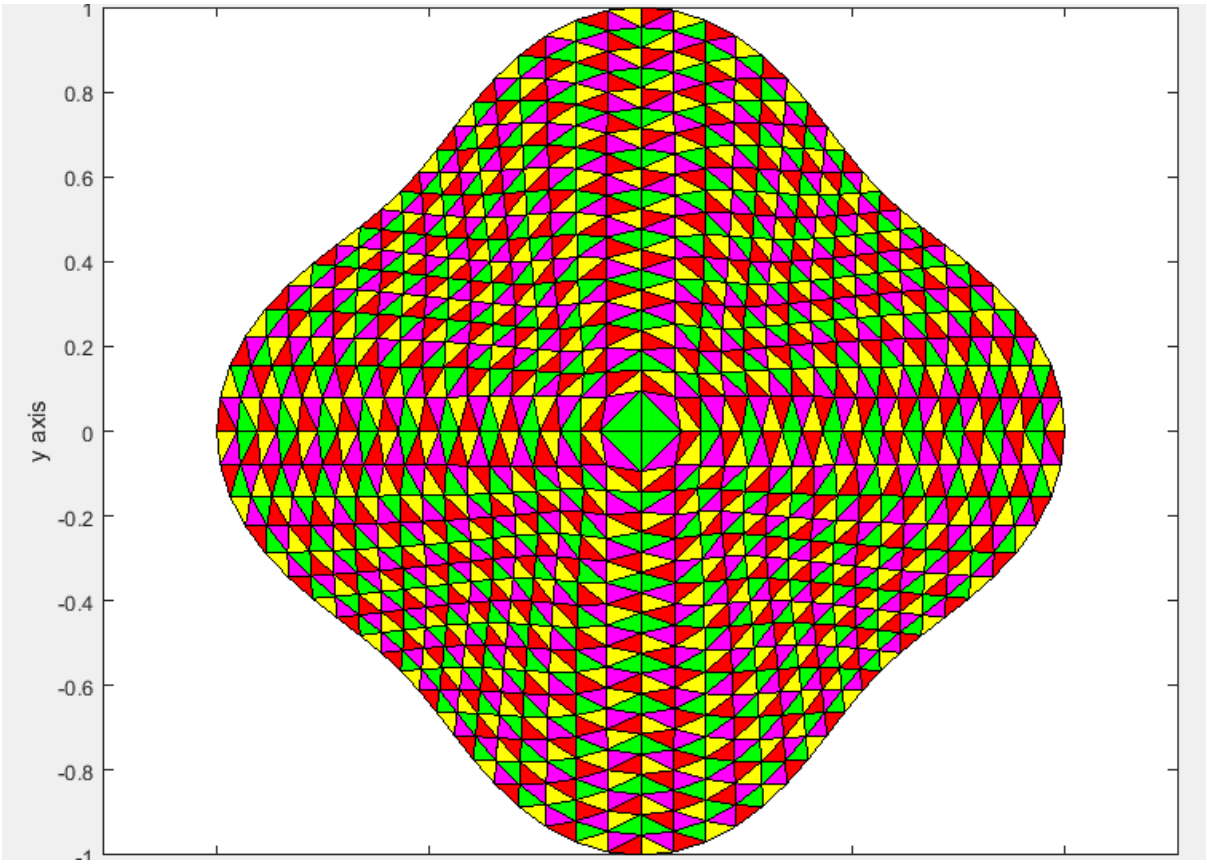


Fig.18d

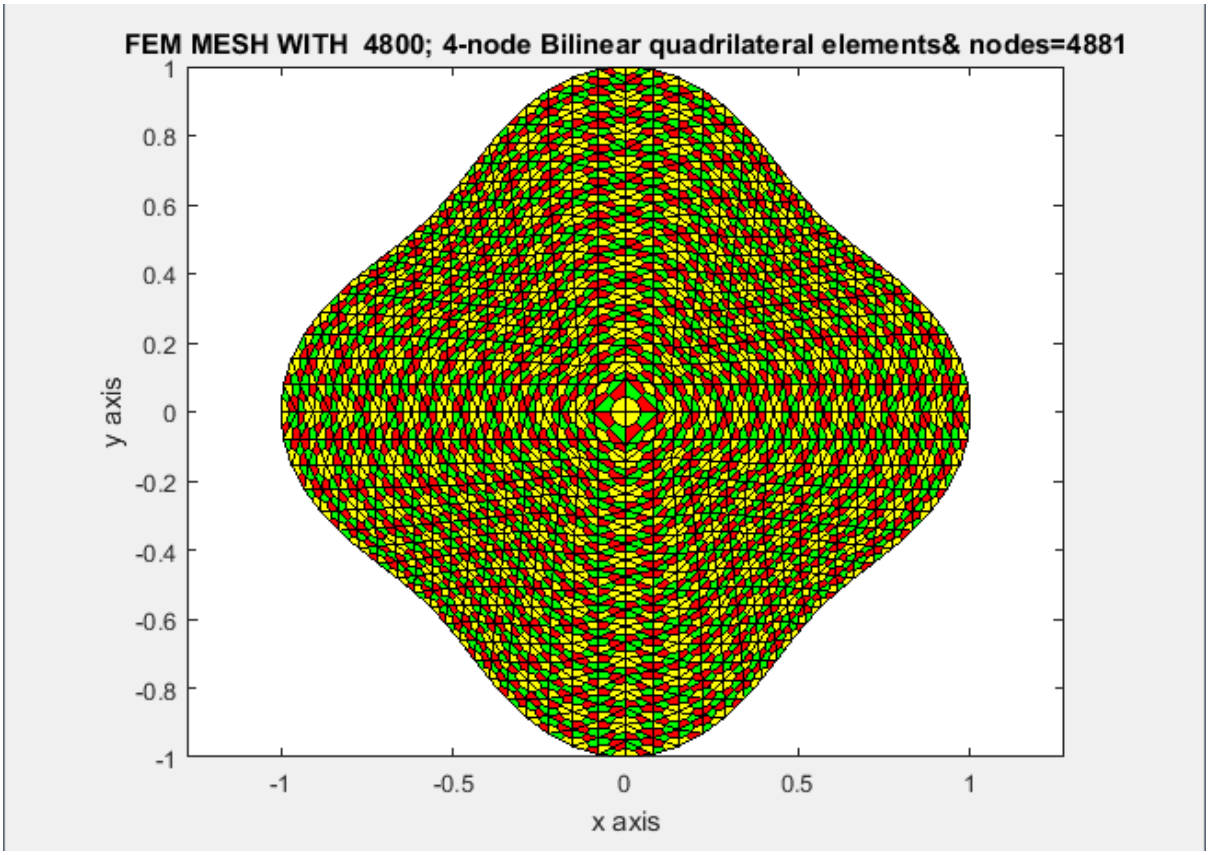


Fig.18e

Example (2) $\rho(\theta) = 1$

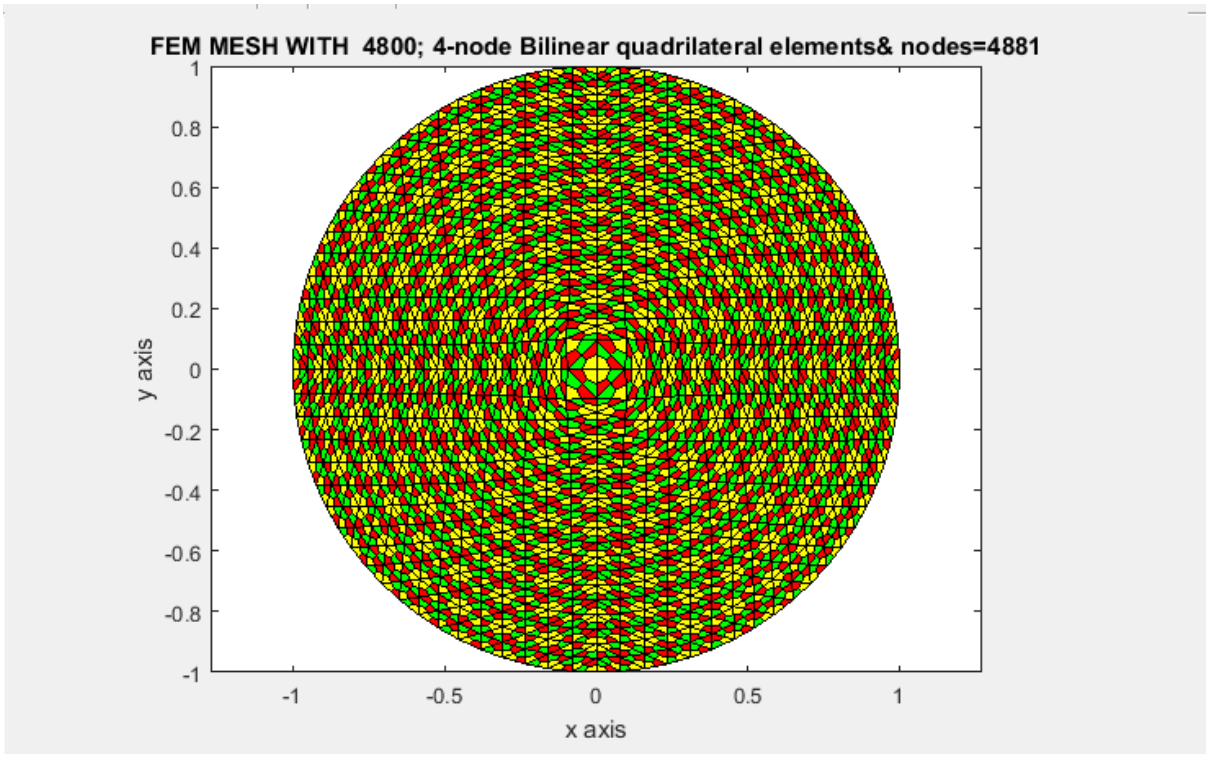


Fig.19

Example (3): $x(\theta) = a\cos(\theta), y(\theta) = b\sin(\theta)$, where a and b are constants and $a \neq b$

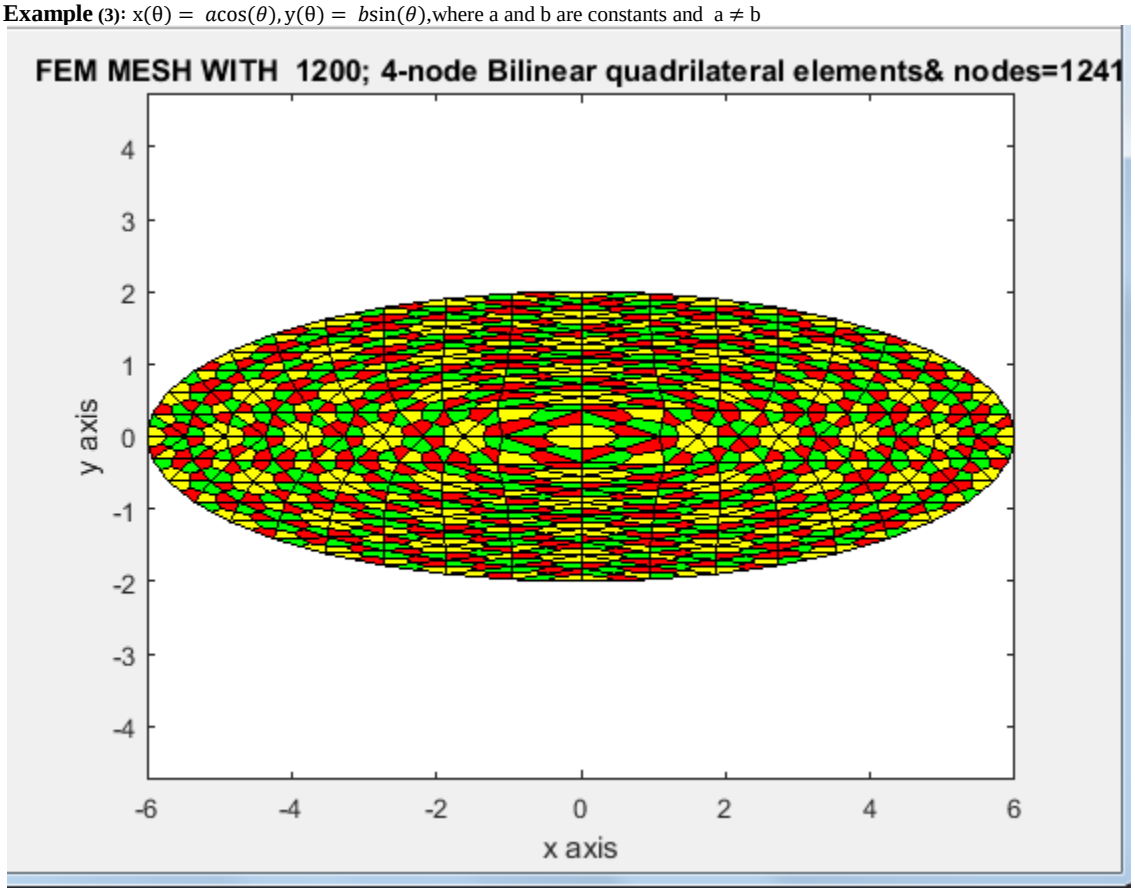


Fig.20

Example (4): $\rho(\theta) = \sqrt{9.25 + 3\cos(4\theta)}$

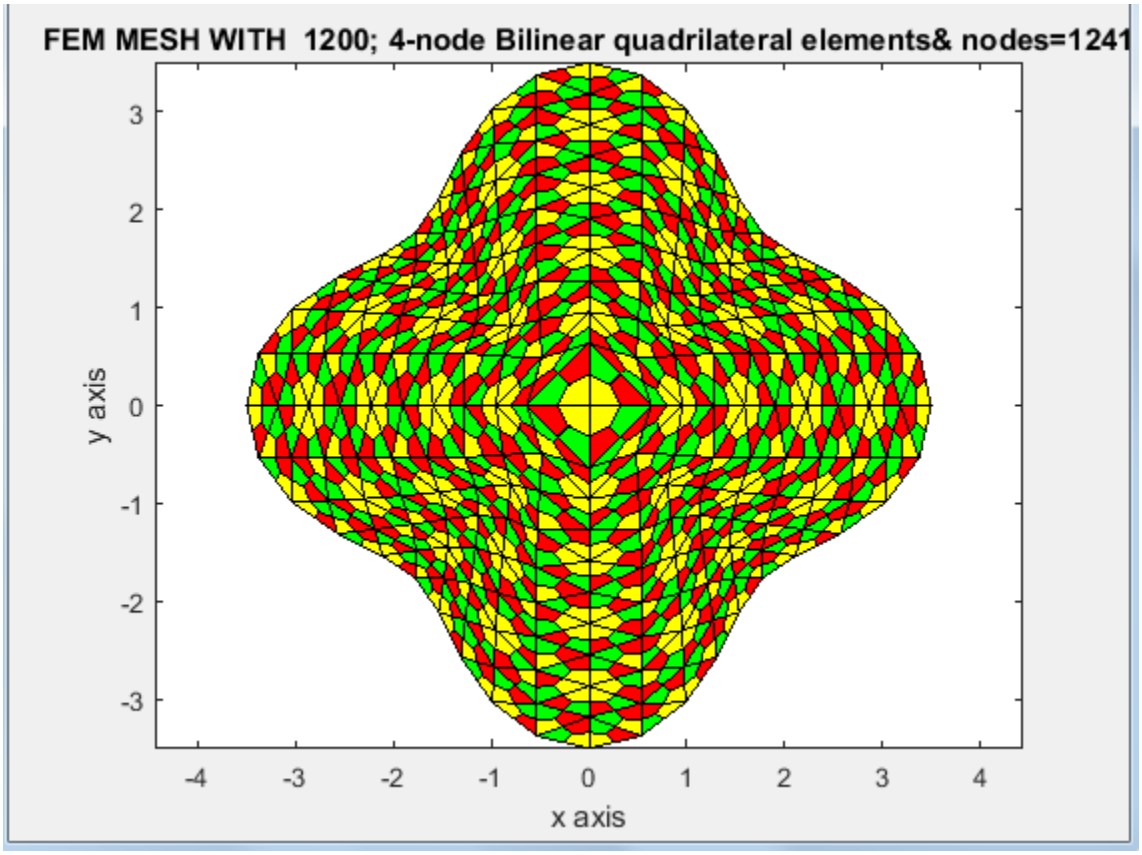


Fig.21

Example(5): $\rho(\theta) = \sqrt{(0.6\cos(\theta) + 0.3\cos(2\theta) - 0.2)^2 + 0.36(sin^2(\theta)}$

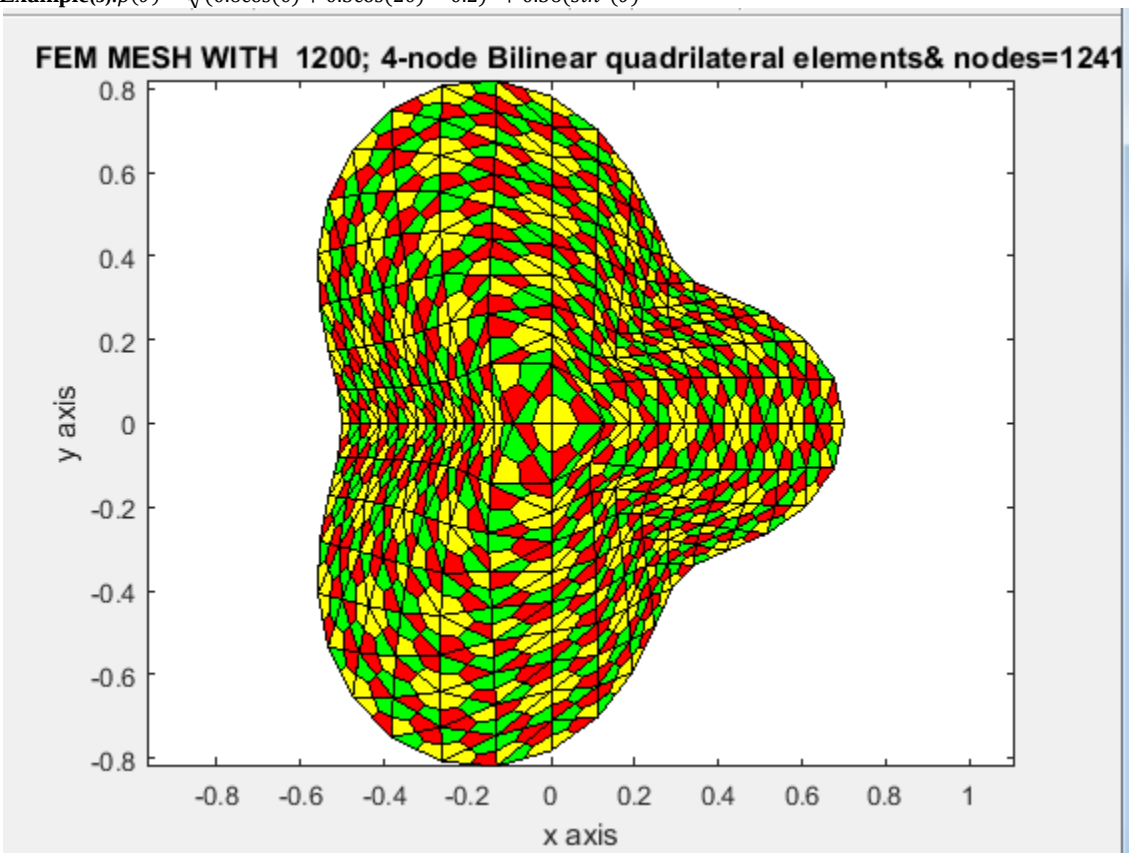


Fig.22

Example (6): $\rho(\theta) = \sqrt{5 - 4\cos(\theta)}$,

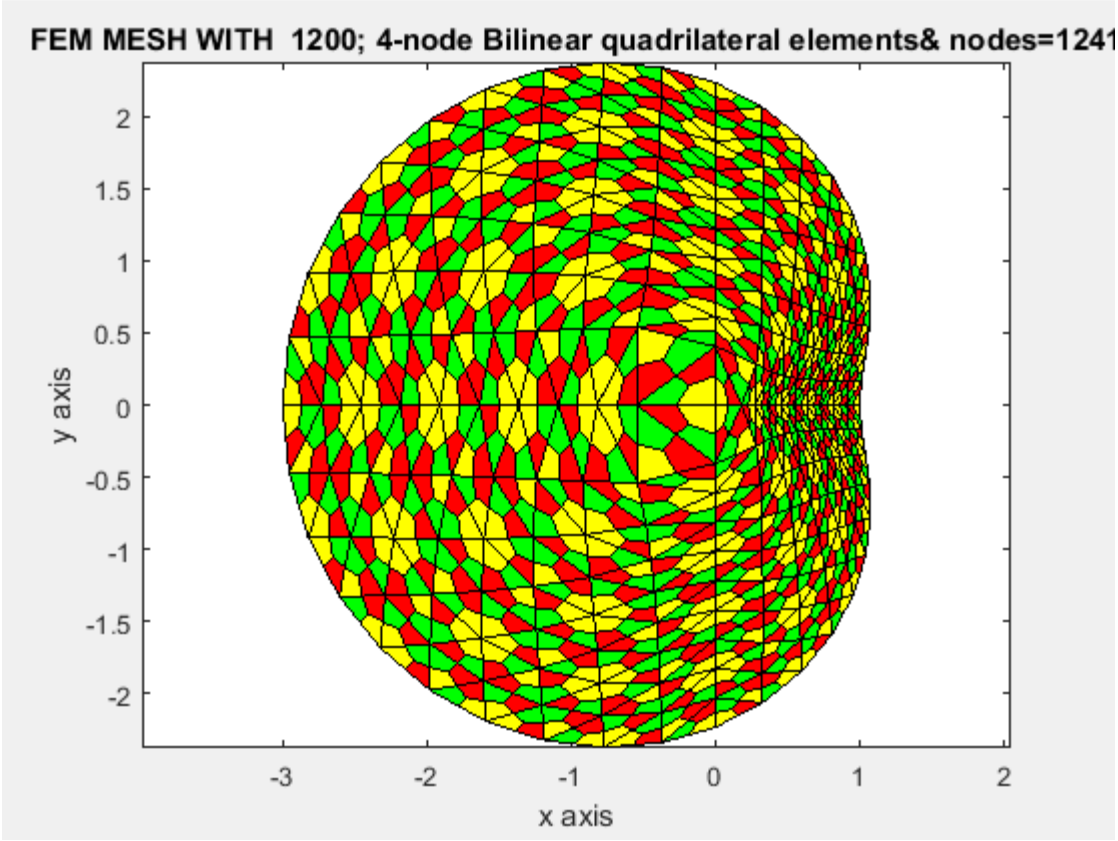


Fig.23

Example(7): $\rho(\theta) = \sqrt{10 - 6\cos(2\theta)}$

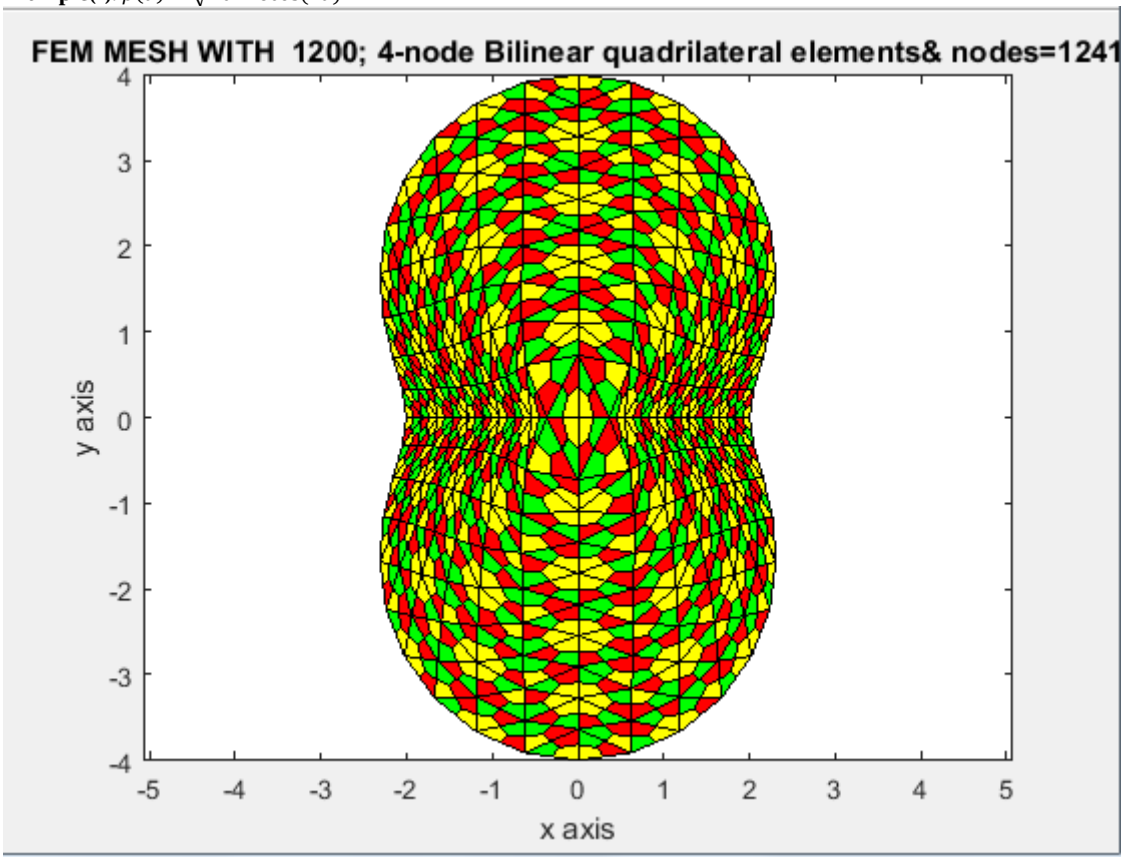


Fig.24

Example (8): $\rho(\theta) = \sqrt{17 - 8\cos(3\theta)}$

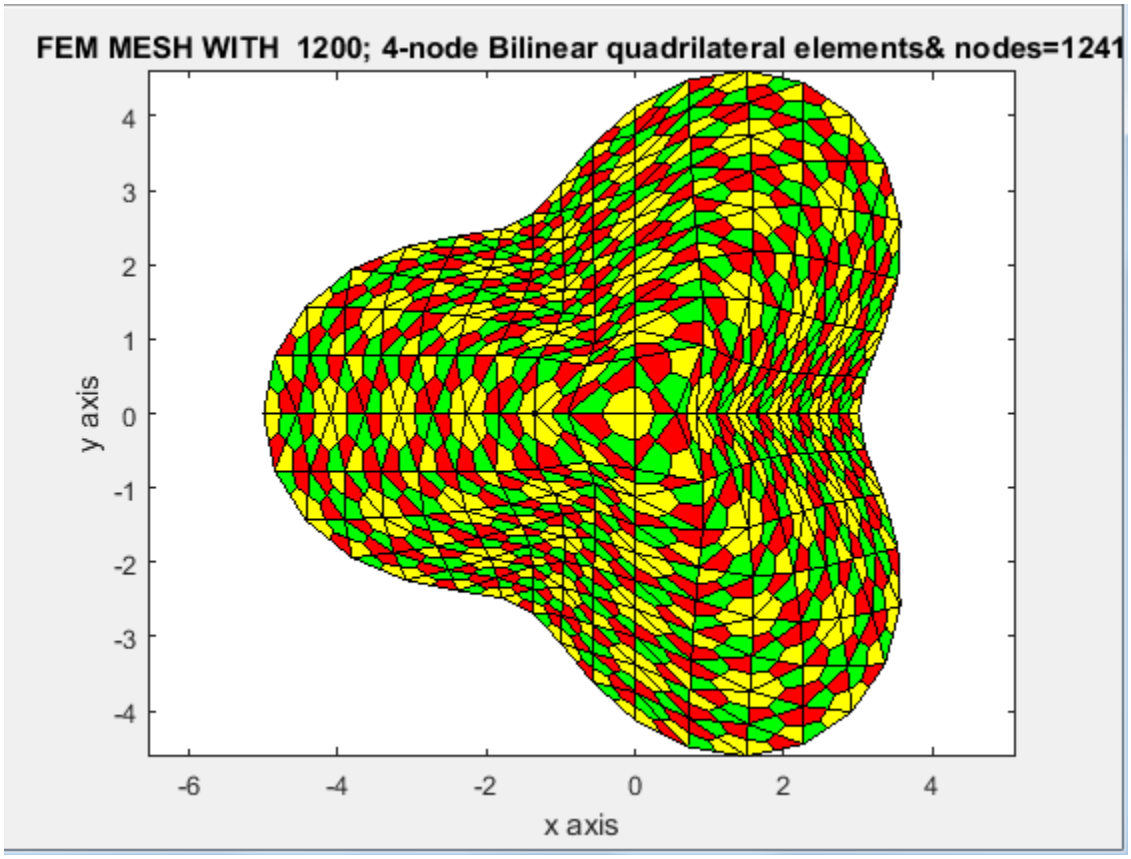


Fig.25

Example (9): $x(\theta) = (3\cos(\theta) + \cos(3\theta))/4\ ,\quad y(\theta) = (3\sin(\theta) - \sin(3\theta))/4$

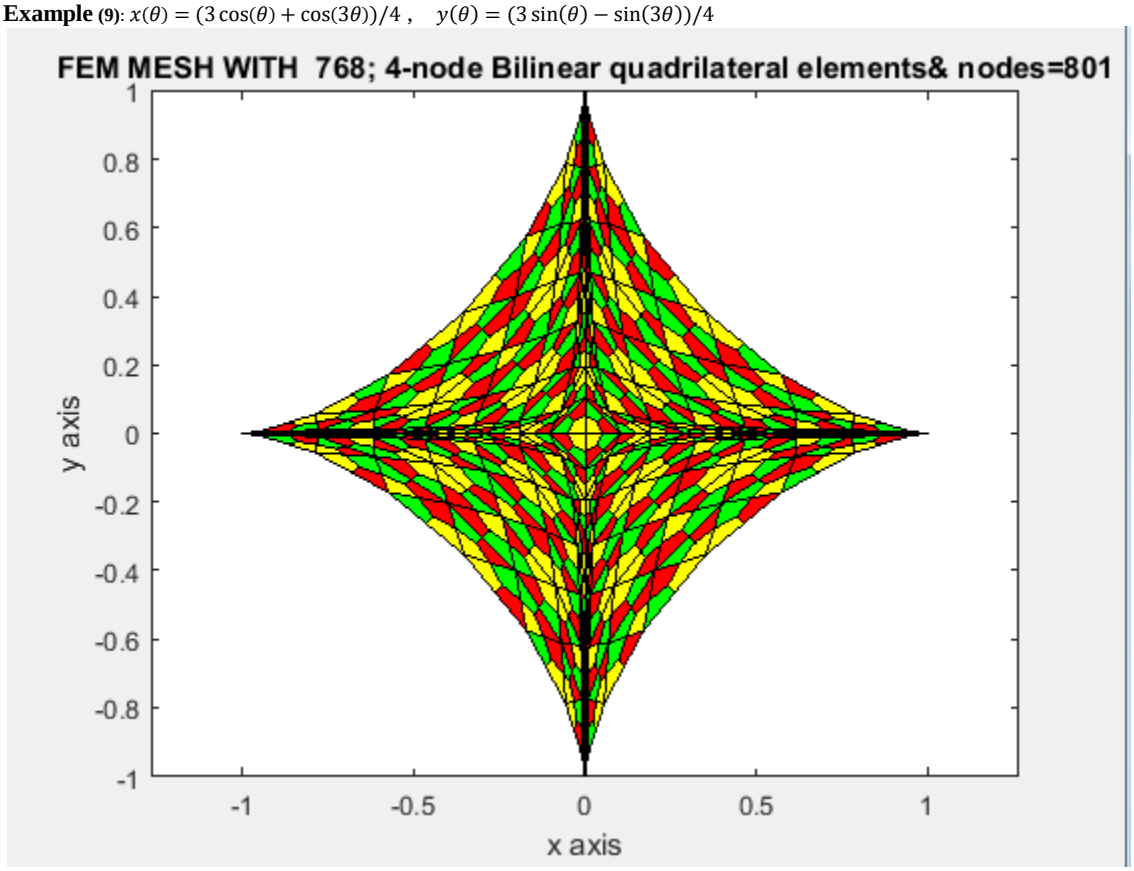


Fig.26

Example (10): $\rho(\theta) = \sqrt{\cos(2\theta) + \sqrt{((1.1)^4 - (\sin(2t))^2)}}$

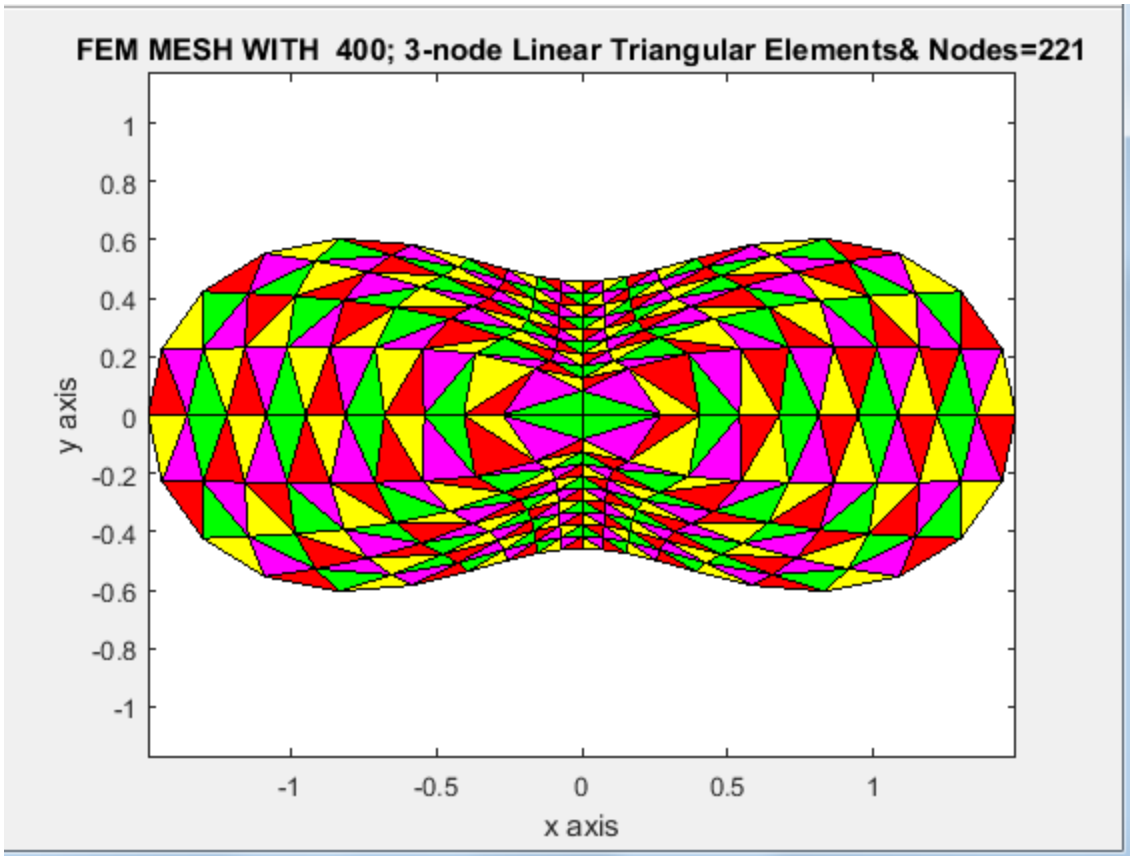


Fig.27

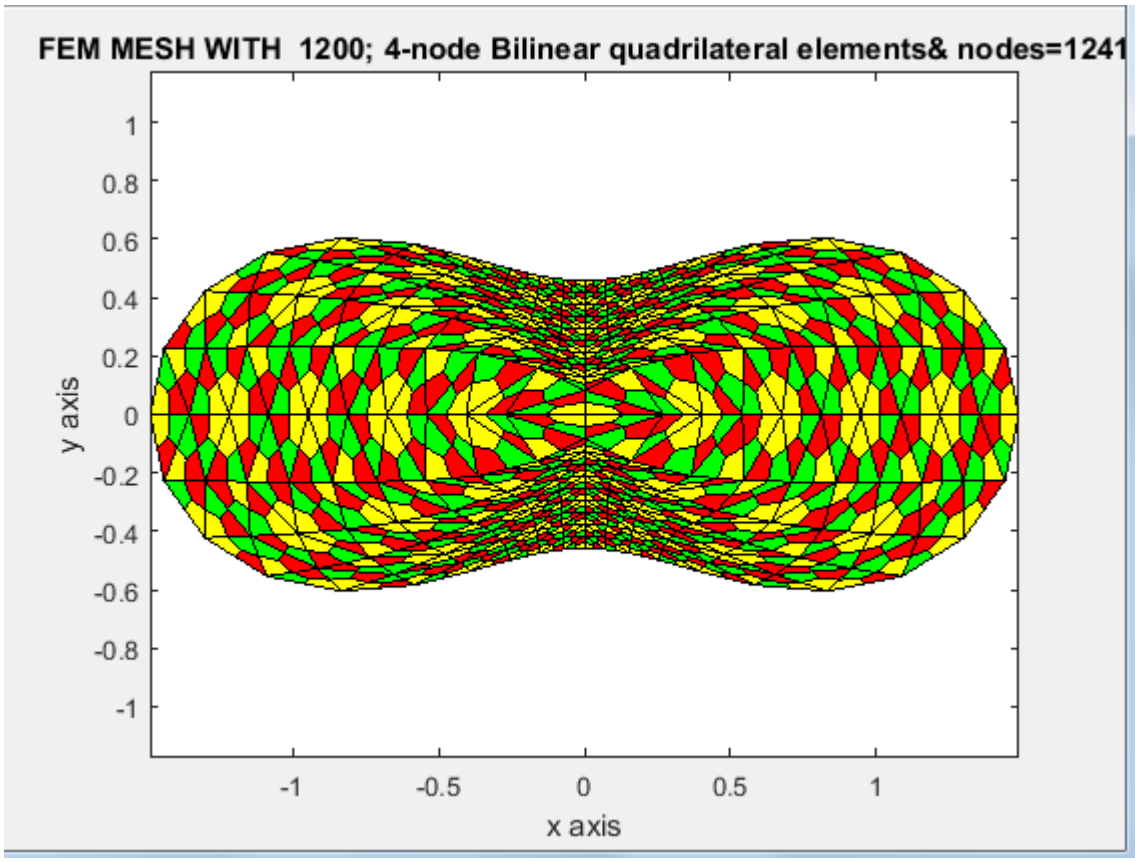


Fig.28

Example (11): $\rho(\theta) = 0.25\sin(\theta) + 0.5\sqrt{1 - 0.9(\cos(\theta))^2} + 0.5\sqrt{1 - 0.7(\cos(\theta))^2}$

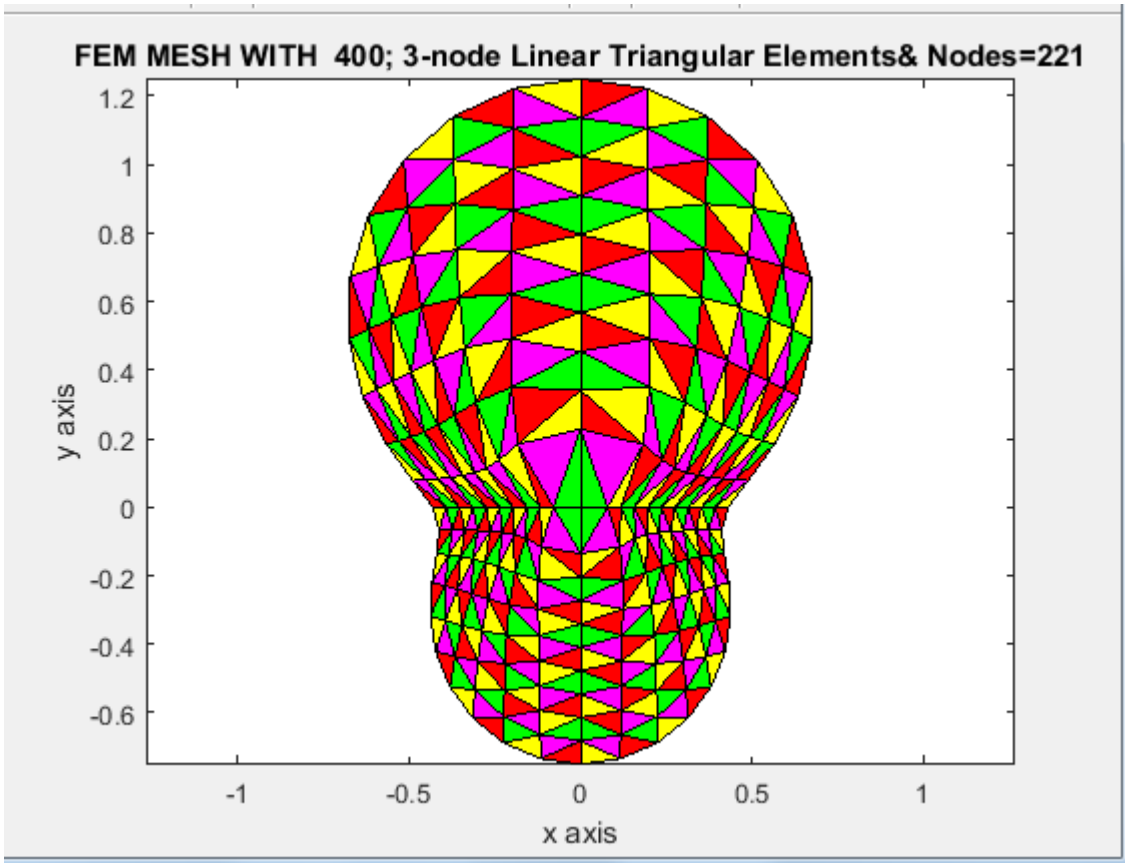


Fig.29

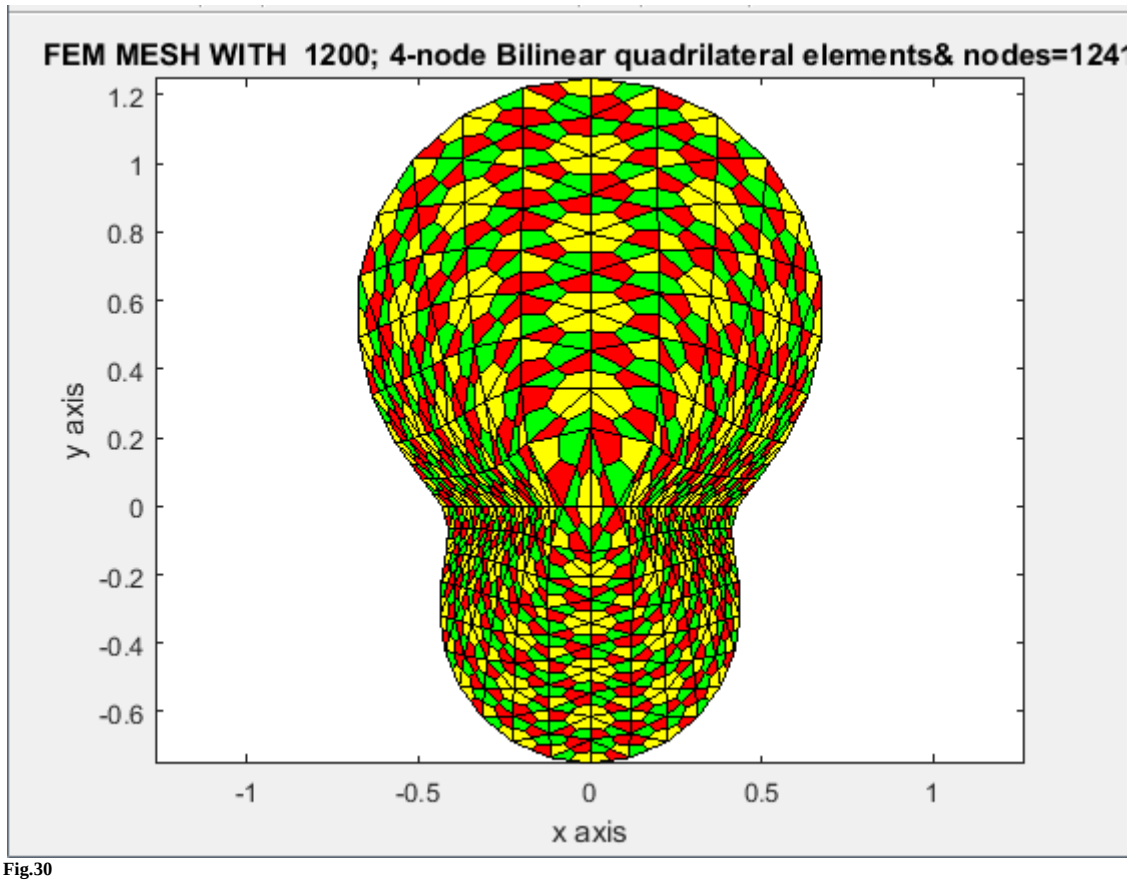


Fig.30

Example (12): $\rho(\theta) = \sqrt{(9.25 + 3\cos(4\theta))/12.25}$

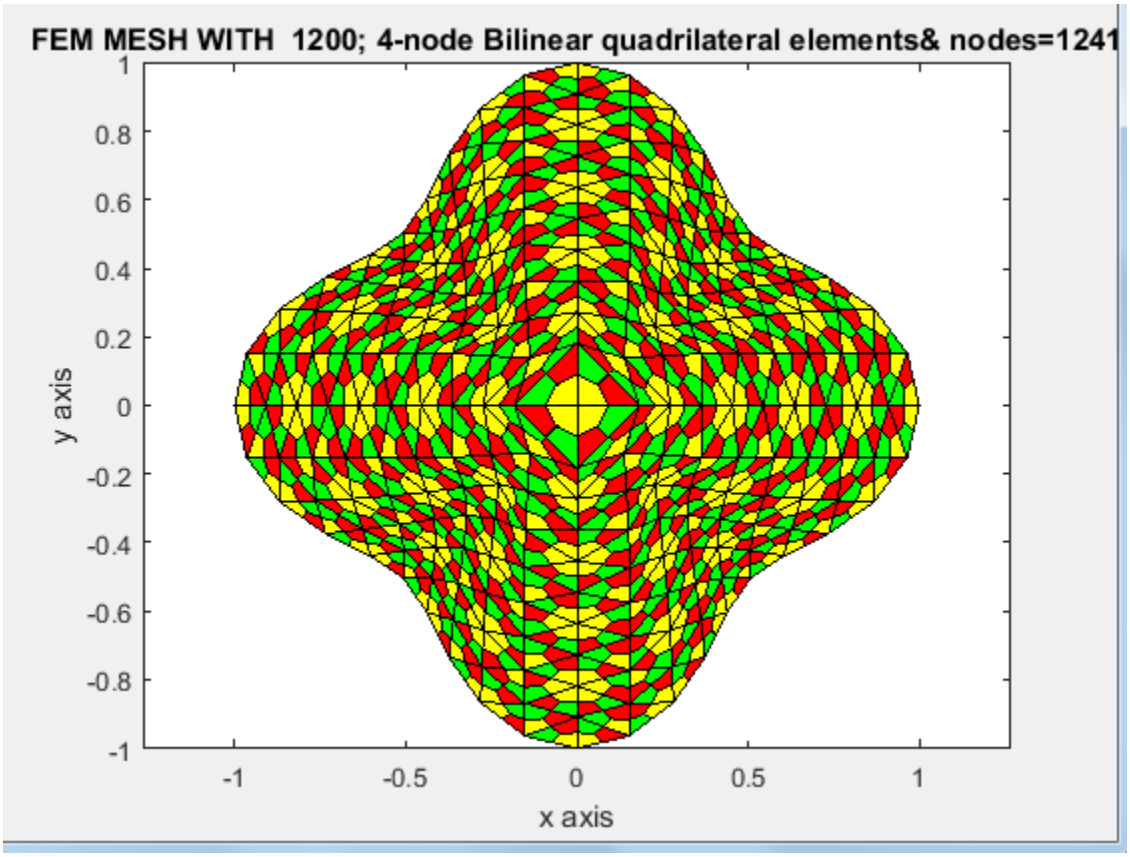


Fig.31

Example (13): A **rhombus shaped bar** whose outer periphery is defined by the equations
$\rho(\theta)$ =1/(cos(θ)+sin(θ)),$\theta \in [0, \pi/2]$
$\rho(\theta) = -1/(\cos(\theta)-\sin(\theta)), \theta \in [\pi/2, \pi]$
$\rho(\theta) = -1/(\cos(\theta)+\sin(\theta)), \theta \in [\pi, 3\pi/2]$
$\rho(\theta)$ =1/(cos(θ)-sin(θ)),$\theta \in [3\pi/2, 2\pi]$
x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$
and spanned by the vertices{(1 ,0), (0 ,1) , (-1 ,0) ,(0 ,-1)} in Cartesian space (x,y)

MATLAB commands

%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,mesh,1,1,0,0)
%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,25,10,mesh,1,1,0,1)
%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/2;pi],[pi/2;pi;3*pi/2],5,25,10,mesh,1,1,0,1)
%mesh=13;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/2;2*pi],5,25,10,mesh,1,1,0,1)

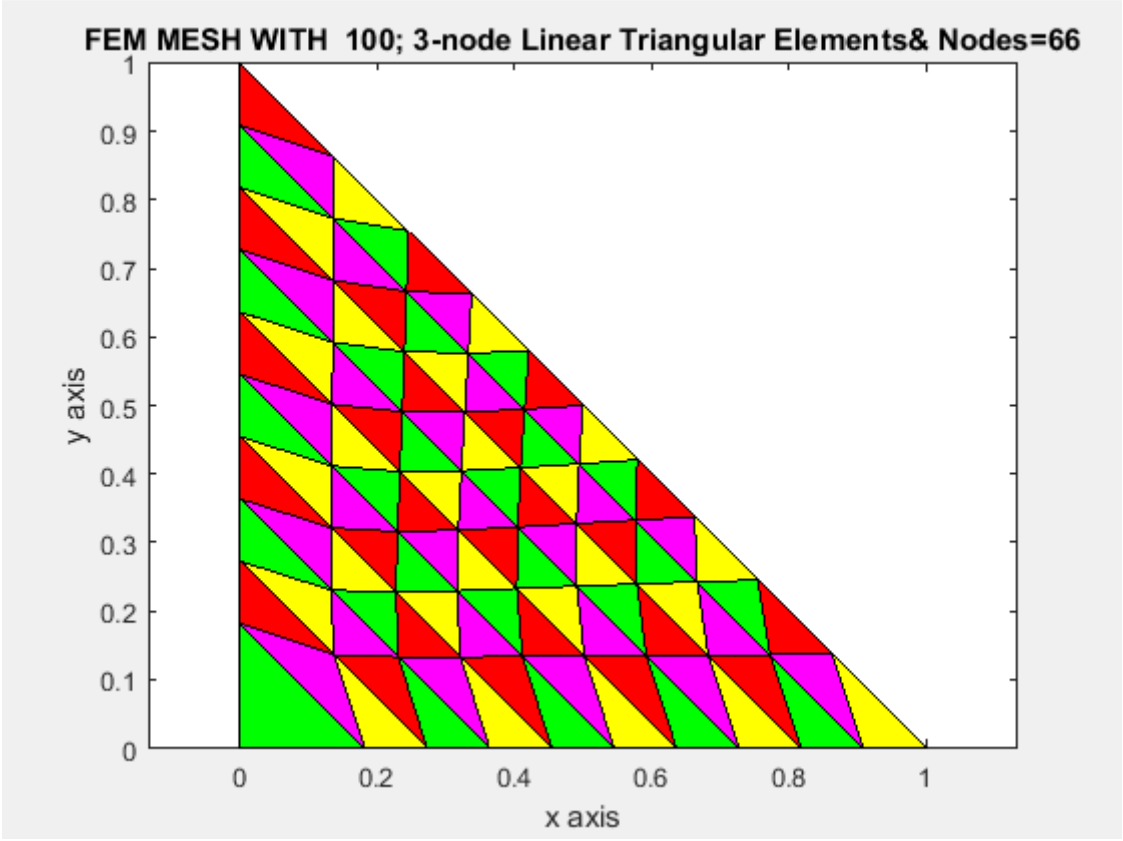


Fig.32a

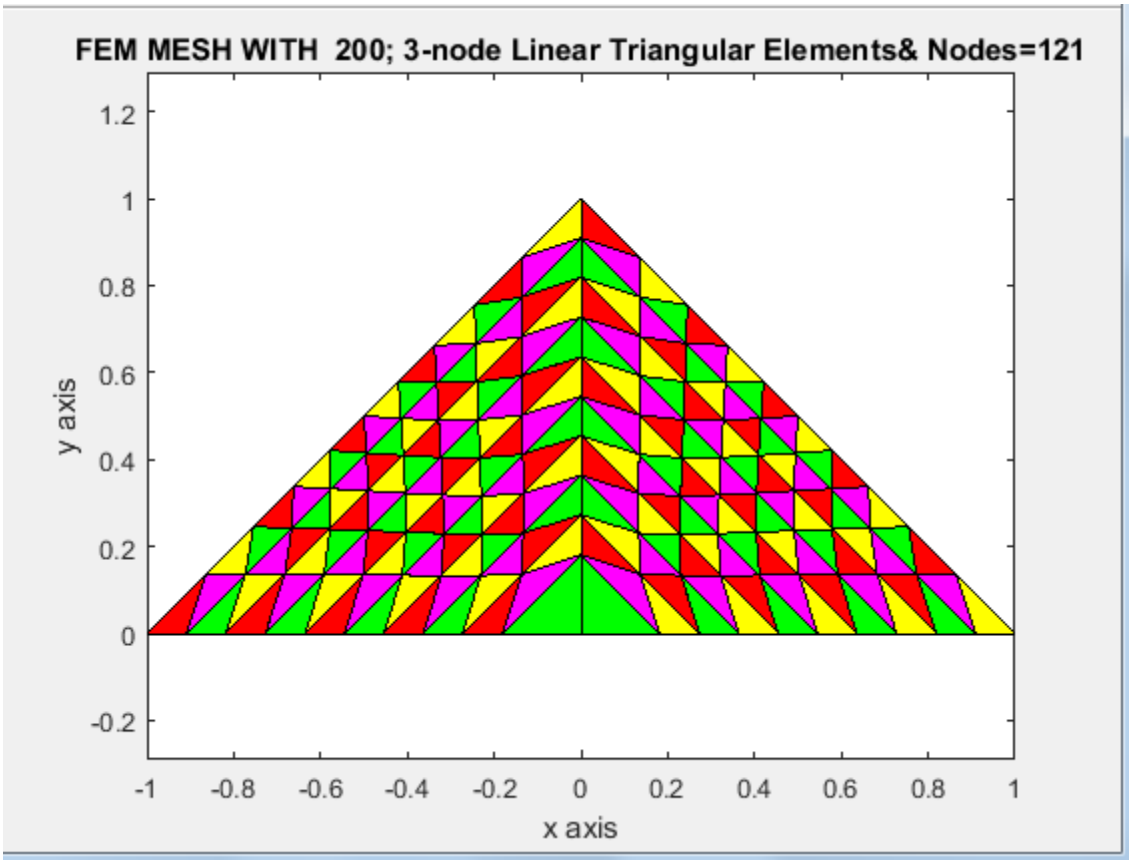


Fig.32b

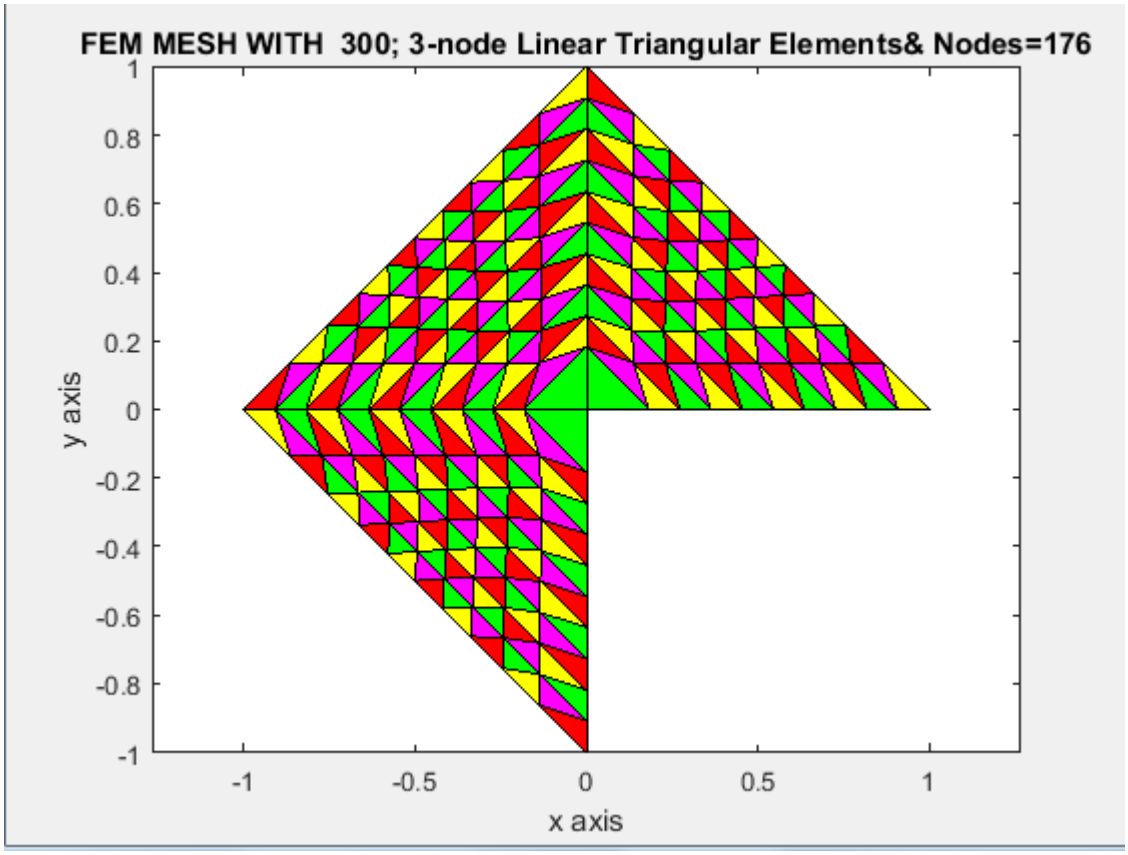


Fig.32c

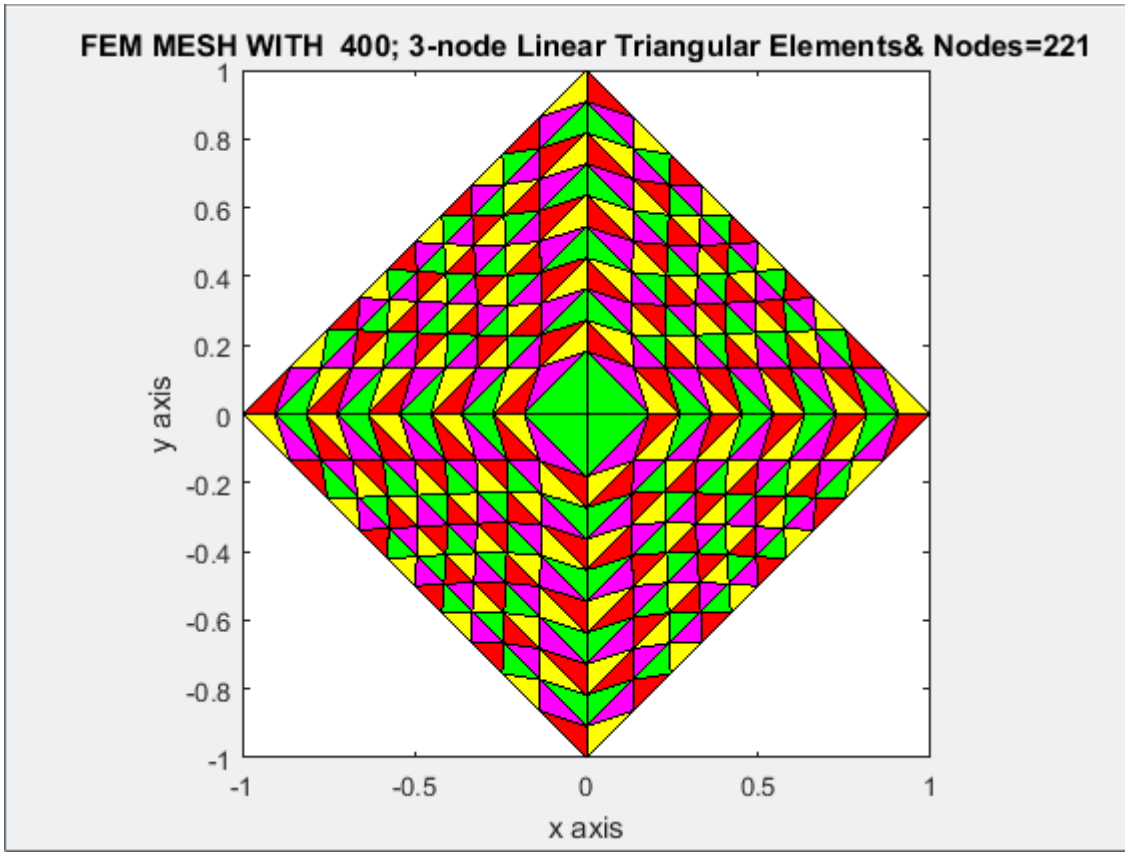


Fig.32d

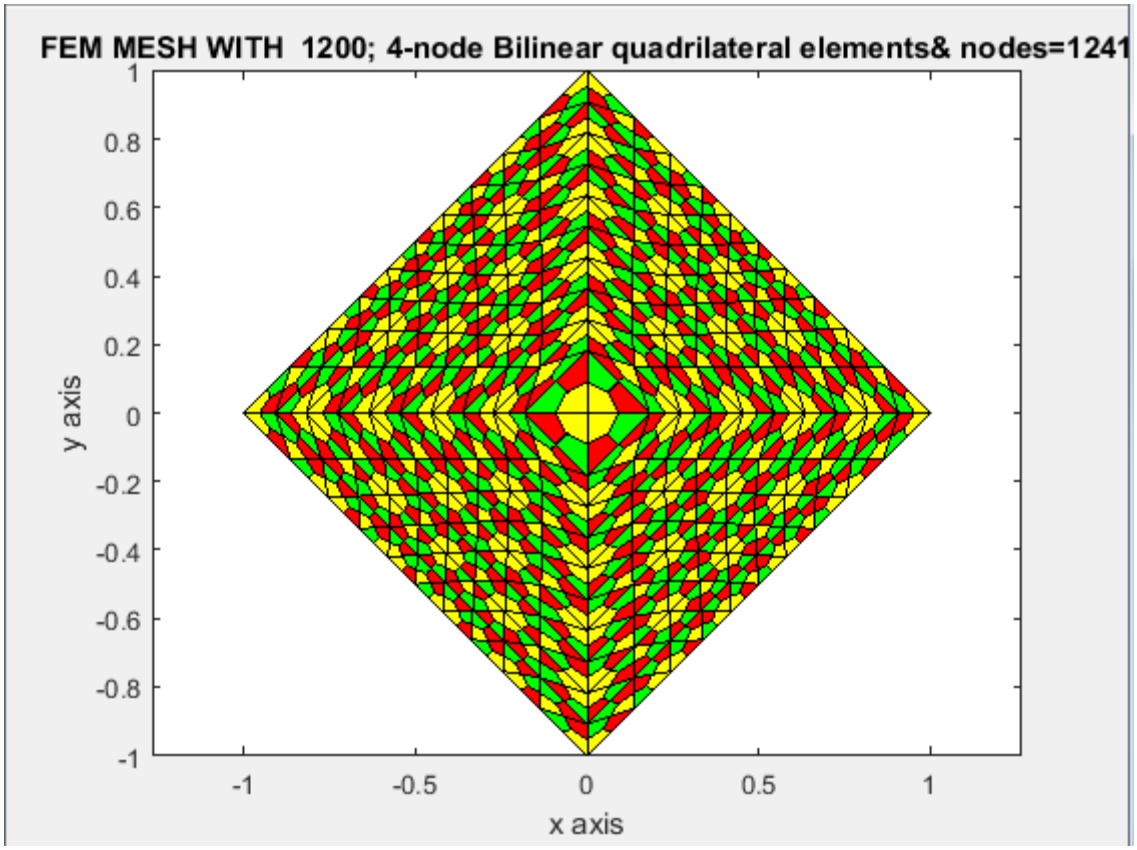


Fig.32e

Example (14): A **square shaped bar** -1≤ $x, y \leq 1$ whose outer periphery is defined by the equations:
$\rho(\theta) =$ 1/cos(θ),$\theta \in [-\pi/4, \pi/4]$
$\rho(\theta) =$ 1/sin(θ), $\theta \in [\pi/4, 3\pi/4]$
$\rho(\theta) = -1$/cos(θ),$\theta \in [3\pi/4, 5\pi/4]$
$\rho(\theta) = -1$/sin(θ),$\theta \in [5\pi/4, 7\pi/4]$
x(θ)= $\rho(\theta)$cos(θ), y(θ)= $\rho(\theta)$sin(θ)
and spanned by the vertices{(1 ,-1), (1 ,1) , (-1 ,1) ,(-1 ,-1)} in Cartesian space (x,y)
MATLAB commands

```
mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[-pi/4],[pi/4],3,25,10,mesh,1,1,0,1)
mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[-pi/4;pi/4],[pi/4;3*pi/4],4,25,10,mesh,1,1,0,1)
mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[-
pi/4;pi/4;3*pi/4],[pi/4;3*pi/4;5*pi/4],5,25,10,mesh,1,1,0,1)
mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[-
pi/4;pi/4;3*pi/4;5*pi/4],[pi/4;3*pi/4;5*pi/4;7*pi/4],5,25,10,mesh,1,1,0,1)
mesh=14;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[-
pi/4;pi/4;3*pi/4;5*pi/4],[pi/4;3*pi/4;5*pi/4;7*pi/4],5,25,10,mesh,1,1,0,1)
```

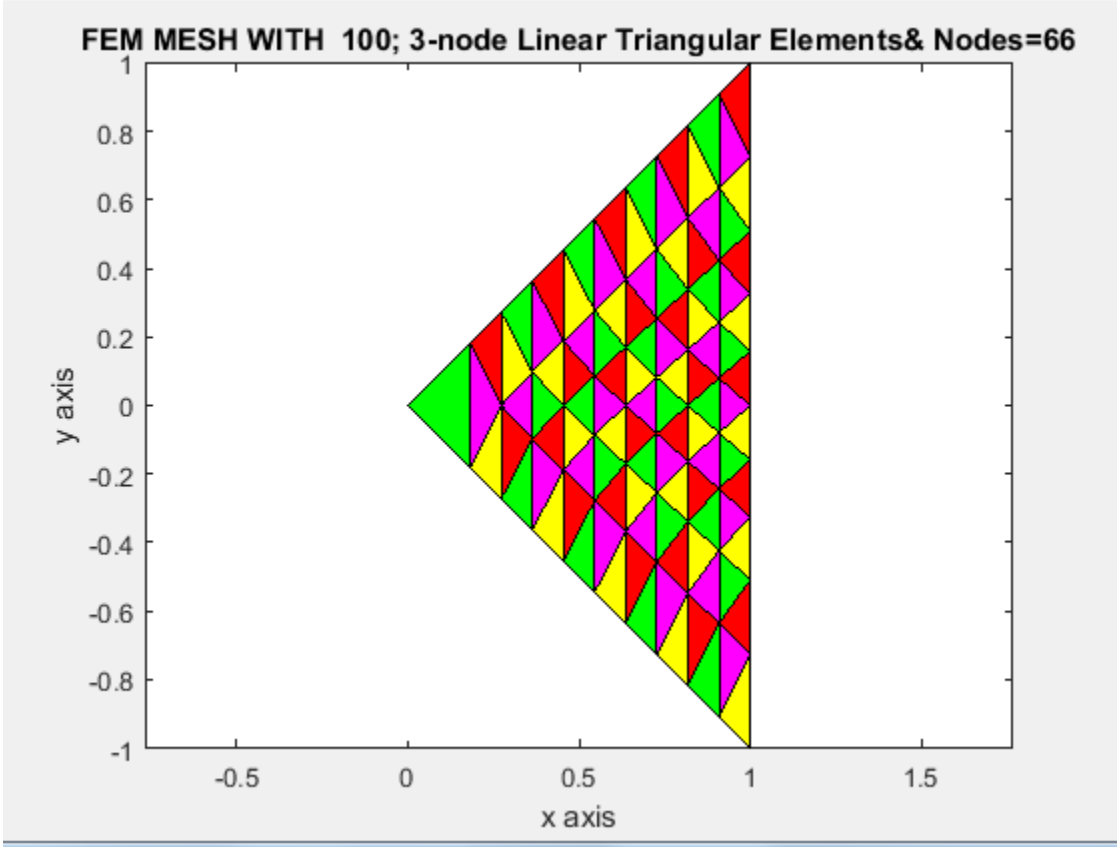


Fig.33a

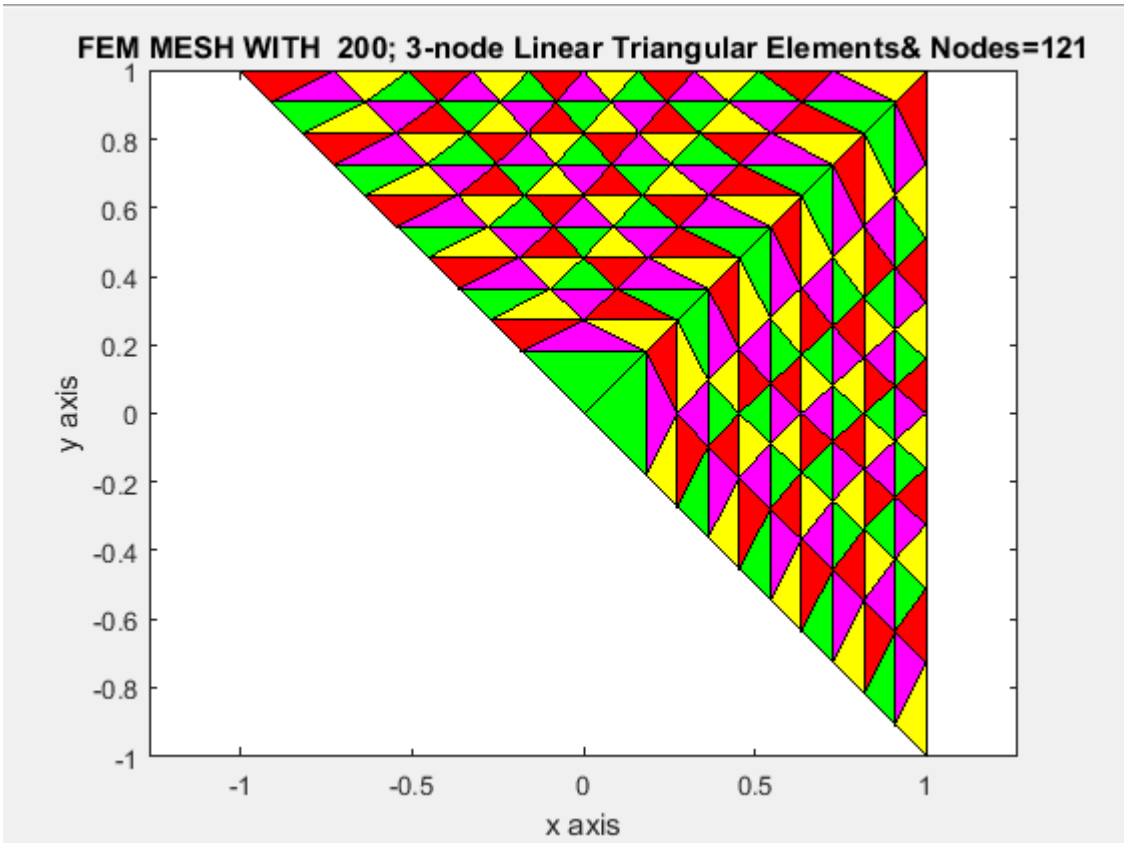


Fig.33b

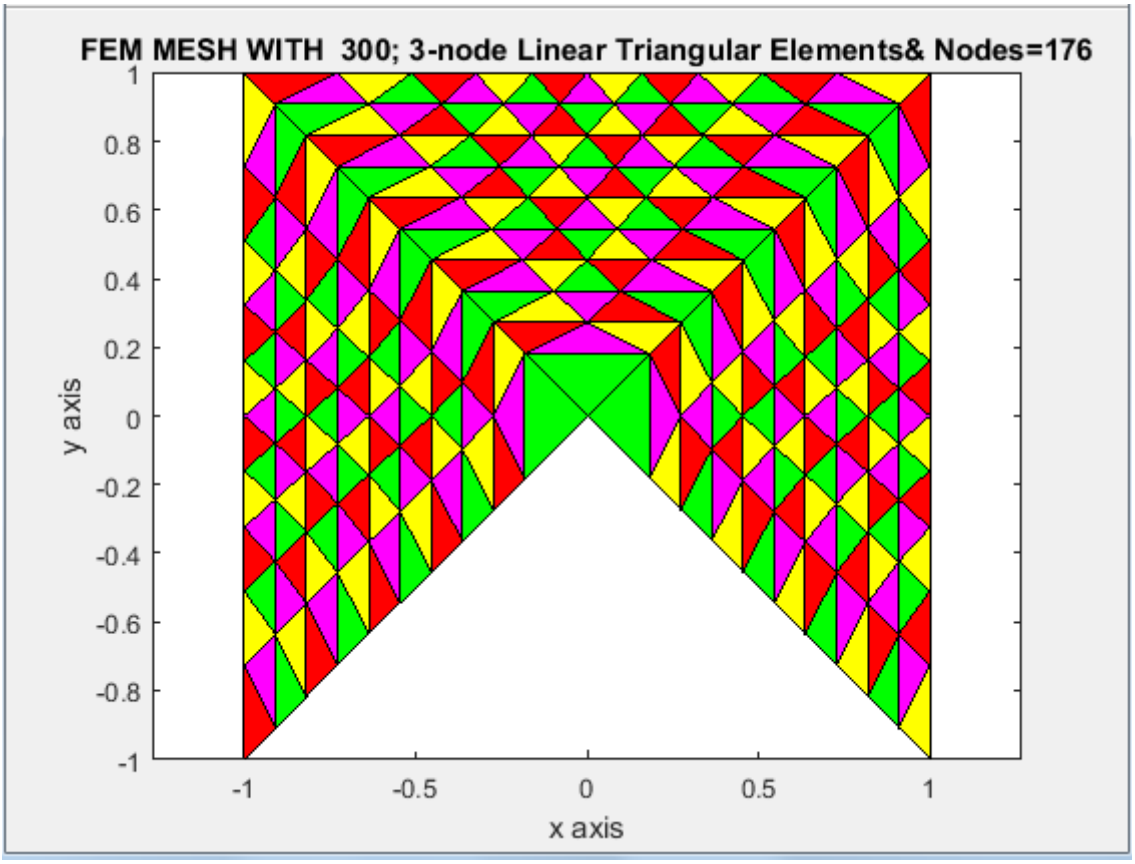


Fig.33c

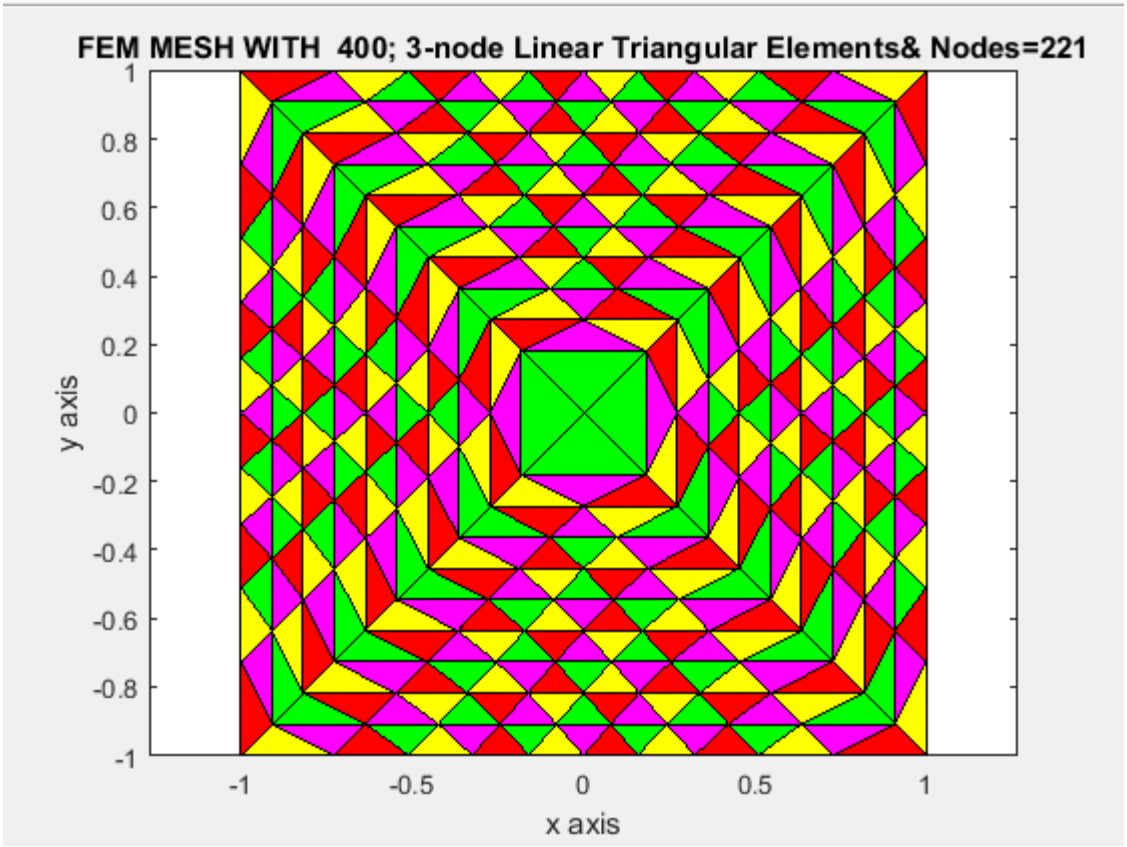


Fig.33d

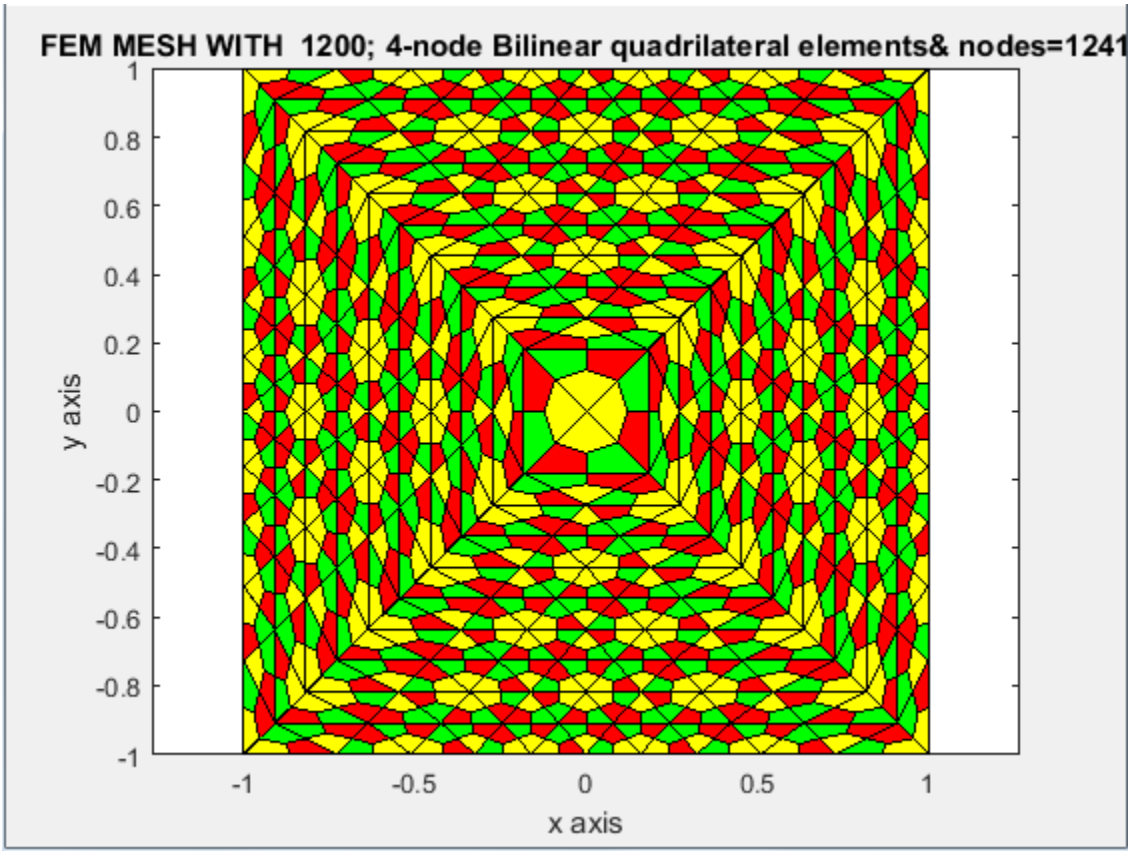


Fig.33e

Example(15): An **arbitrary shaped curved bar** -1≤ $x, y \leq 1$ whose outer periphery is defined by the equations:

$\rho(\theta) = 0.9 + 0.1\cos(4\theta)$, $\theta \in [0, \pi/2]$

$\rho(\theta) = 1, \theta \in [\pi/2, \pi]$

$\rho(\theta) = -1/(\cos(\theta)+\sin(\theta)), \theta \in [\pi, 3\pi/2]$

$\rho(\theta) = \sqrt{(9.25 + 3\cos(4\theta))/12.25}$, $\theta \in [3\pi/2, 2\pi]$

x(θ)= $\rho(\theta)\cos(\theta)$, y(θ)= $\rho(\theta)\sin(\theta)$

and spanned by the vertices{(1 ,0), (0 ,1) , (-1 ,0) ,(0 ,-1)} in Cartesian space (x,y)

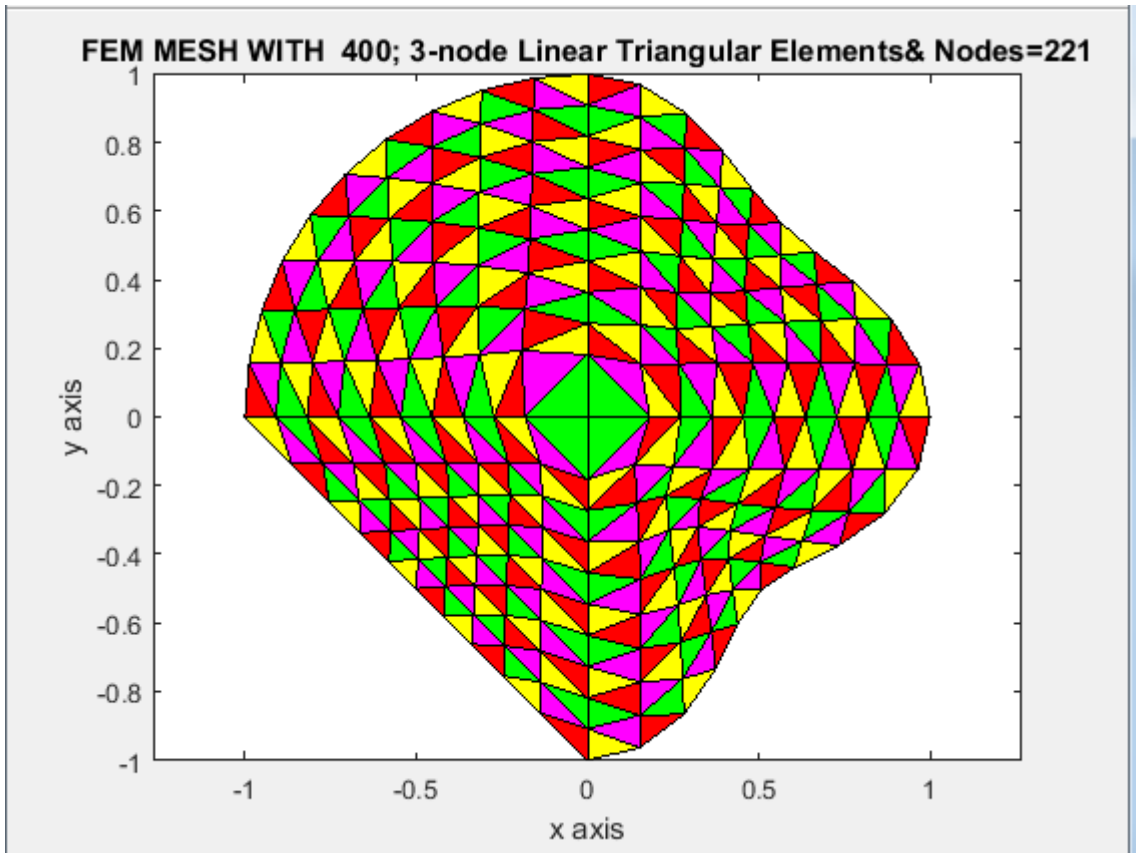


Fig.34a

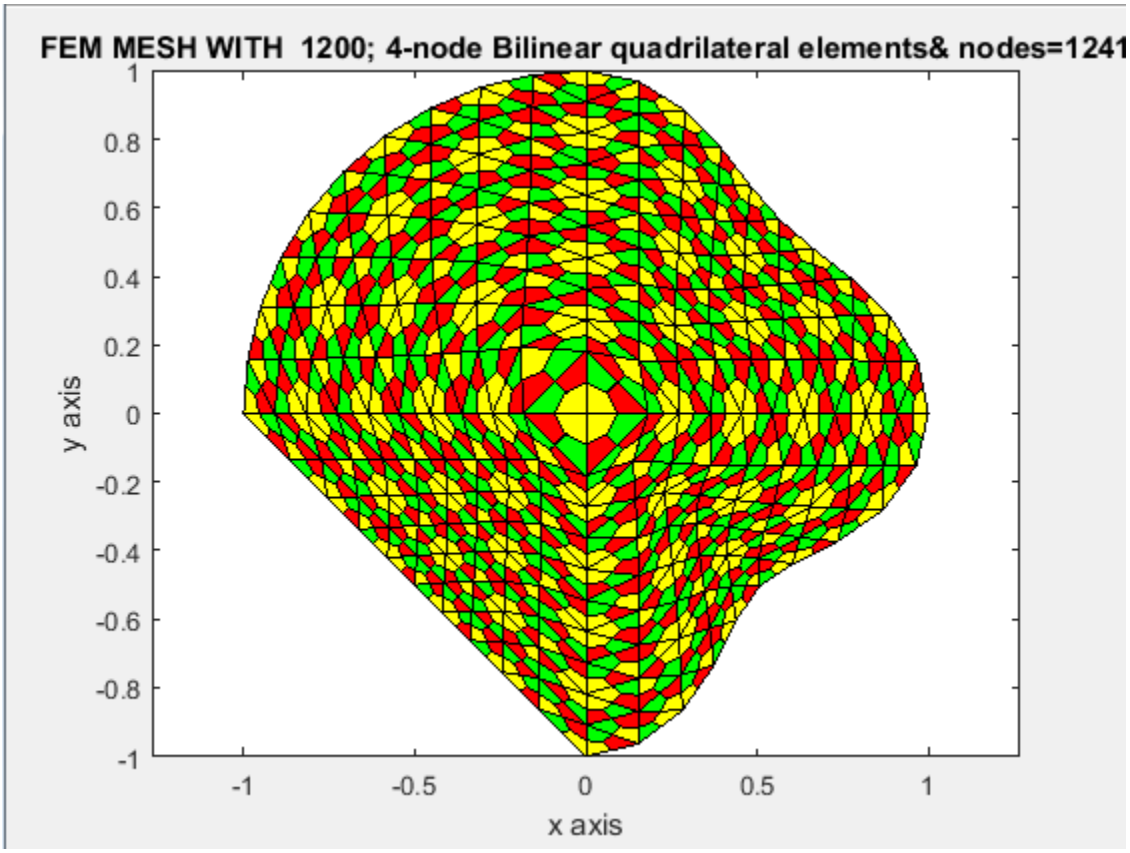


Fig.34b

5.2 Mesh Generation Over an Analytical Curved Surface

In several physical applications in science and engineering, the boundary value problem require meshes generated over curved surfaces whose boundary is defined by analytical equations of curves. Again our aim is to have a code which automatically generates a mesh of linear convex quadrilaterals in the interior of the domain and quadrilaterals with three straight edges and one curved edge near to the boundary of curved surface for the complex domains such as those in [7,8,9]. We use the theory and procedure developed in sections 2, 3 and 4. The following MATLAB codes are written for this purpose.

(I)COMPUTER PROGRAMS FOR TRIANGULATION

(1) triangular_mesh4LinearTriangles_t3curvedBoundary.m
(2) triangulation4polygonal_domain_coordinatesNcurvedBoundary.m
(3) nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary.m
(4) nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2.m

(II)COMMON PROGRAMS REQUIRED FOR TRIANGULATION AND QUADRANGULATION

(5) fcnt.m
(6) generate_polar_coordinate_over_the_standard_triangle.m

(III)COMPUTER PROGRAMS FOR QUADRANGULATION

(7) triangular_mesh4LinearTriangles_t3Nq4curvedBoundary.m
(8) triangulation4polygonal_domain_coordinatesNq4curvedboundary.m
(9) nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundaryNq4.m
(10) nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2Nq4.m

We have already included some meshes generated by using the above MATLAB codes. Some sample commands to generate the curved domains stated above also appear in the comment lines of programs. In all the cases the sample data is a part of the codes.

Conclusions

An automatic indirect quadrilateral mesh generator which uses the splitting technique is presented for the two dimensional analytical curved surfaces whose boundary is defined by a polar equation. 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. Once this input is created, by selecting an appropriate interior point of the curved domain, we form the subdomains in the shape of curved triangles. These subdomains are then triangulated to generate a fine mesh of six node polygonal elements which can be converted to form four linear 3-node triangles . We have then proposed an automatic triangular to quadrilateral conversion scheme in which each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of the

edges and a vertex at the barrycentre of the triangular element. This task is made a bit simple since a fine mesh of six node polygon is first generated. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this discretizes the given curved domain into all convex bilinear quadrilaterals in the interior of the curved domain and quadrilaterals with three straight sides and one curved side which forms part of the curved boundary, thus propogating a uniform refinement. This simple method generates 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 all quadrilateral mesh for the wedge element or curved triangular element. We believe that this work will be useful for various applications in science and engineering. The quality of the quadrilateral mesh can be subsequently enhanced by a series of mesh modifications and element shape improvement procedures.One advantage of the mesh is for applications to two dimensional boundary value problems,because the jacobian of all the interior quadrilaterals is a linear expression,as explained in our research work[10]. The elements near to the boundary are a few quadrilaterals having one curved side and three straight sides.Thus an algorithm based on the proposed mesh generation scheme has computational convenience and it can be easily coded.

References

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

[2] 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)

[3] V.Ruas, AUTOMATIC GENERATION OF TRIANGULAR FINITE ELEMENT MESHES, Comp. & Maths. with Appls.. Vol. S. PP. I25-140(1979)

[4] G. Xu, B. Mourrain, R. Duvigneau, and A. Galligo. Parametrization of computational domain in isogeometric analysis: methods and comparison.Computer Methods in Applied Mechanics and Engineering, 200(23{24):2021-2031, 2011.

[5] M. Utri and A. Brummer. Energy potential of dual lead rotors for twin screw compressors. In IOP Conference Series: Materials Science and Engineering,volume 232, page 012018. IOP Publishing, 2017.

[6] H.T. Rathod , Bharath Rathod , K.V.Vijayakumar , K. Sugantha Devi, A new automatic finite element mesh generation scheme of all quadrilaterals over an analytical curved surface by using parabolic arcs,International Journal Of Engineering And Computer Science ISSN:2319-7242,Volume - 3 Issue -8 August, 2014 Page No. 7437-7507

[7] Chein-Shan Liu, Elastic Torsion Bar with Arbitrary Cross-Section Using the Fredholm

[8] Integral Equations, CMC, vol.5, no.1, pp.31-42, 2007, Tech Science Press

[9] P. Constantinou , S. Franz , L. Ludwig , C. Xenophontos, Finite element approximation of reaction–diffusion problems using an exponentially graded mesh, Computers and Mathematics with Applications 76 (2018) 2523–2534

[10] Sudeep Nayak, Torsional rigidity in beams with arbitrary cross-sections,B.Tech thesis, Department of Mechanical Engineering, NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA ,May (2014)

[11] H. T. Rathod and Md. Shafiqul Islam, Some precomputed Universal Numeric Arrays for Linear Convex Quadrilateral Finite Elements, Finite Elements in Analysis and Design, Vol.38, pp. 113-136 (2001)

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

[13] Rathod H.T,Rathod Bharath,Shivaram.K.T,Sugantha Devi.K, A new approach to automatic generation of all quadrilateral mesh for finite analysis, International Journal of Engineering and Computer Science, Vol. 2,issue 12,pp3488-3530(2013)

[14] H.T.Rathod, Bharath Rathod, K.T.Shivaram, A.S.Hariprasad, K.V.Vijayakumar K.Sugantha Devi, A New Approach to an All Quadrilateral Mesh Generation Over Arbitrary Linear Polygonal Domains for Finite Element Analysis, International Journal of Engineering and Computer Science , vol.3,issue4(2014),pp 5224-5272

[15] H.T. Rathod , Bharath Rathod, A New Approach to Automesh Generation of $all\ \beta$ graded triangular and quadrilateral Finite Elements over Analytical Surfaces by using the Parabolic Arcs passing through four points on the Boundary Curve, International Journal Of Engineering And Computer Science ISSN:2319-7242,Volume 7 Issue 9 September 2018, Page No. 24214-24310

APPENDIX

COMPUTER PROGRAMS FOR TRIANGULATION

%PROGRAM-1**********

```
function[]=triangular_mesh4LinearTriangles_t3curvedBoundary(n1,n2,n3,th0,thn,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete
,color)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,1,1,1,0)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[pi/2],[pi],3,25,10,1,1,1,0)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[pi],[3*pi/2],3,25,10,1,1,1,0)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[3*pi/2],[2*pi],3,25,10,1,1,1,0)
% mesh=3;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,mesh,1,1,0)
%generate_triangular_mesh_over_the_standard_triangle(mesh)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,1,1,1,0,0)
%triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,1,1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,1,2,mesh,1,1,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,4,4,mesh,1,1,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/2;pi],[pi/2;pi;3*pi/2],5,1,2,mesh,1,1,0
)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/2;2*pi
],5,100,20,mesh,1,1,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,4,4,mesh,1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,4,4,mesh,1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/2;pi],[pi/2;pi;3*pi/2],5,4,4,mesh,1,1,0
,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/2;2*pi
],5,4,4,mesh,1,1,0,0)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[0],[pi/2],3,4,4,1,1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,4,4,mesh,1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/2;pi],[pi/2;pi;3*pi/2],5,4,4,mesh,
1,1,0,0)
%mesh=1;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/2;
2*pi],5,4,4,mesh,1,1,0,0)
%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,mesh,1,1,0,0)
%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,25,10,mesh,1,1,0,1)
%mesh=13;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/2;pi],[pi/2;pi;3*pi/2],5,25,10,mesh,1
,1,0,1)
%mesh=13;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/
2;2*pi],5,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[-pi/4],[pi/4],3,25,10,mesh,1,1,0,0)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[-pi/4],[pi/4],3,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[-pi/4;pi/4],[pi/4;3*pi/4],4,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[-
pi/4;pi/4;3*pi/4],[pi/4;3*pi/4;5*pi/4],5,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[-
pi/4;pi/4;3*pi/4;5*pi/4],[pi/4;3*pi/4;5*pi/4;7*pi/4],5,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[-
pi/4;pi/4;3*pi/4;5*pi/4],[pi/4;3*pi/4;5*pi/4;7*pi/4],5,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[o],[pi/4],3,25,10,mesh,1,1,0,0)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[o],[pi/4],3,25,10,mesh,1,1,0,0)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1],[2],[3],[0],[pi/4],3,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2],[2;3],[4;4],[0;pi/4],[pi/4;pi/2],4,25,10,mesh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3],[2;3;4],[5;5;5],[0;pi/4;pi/2],[pi/4;pi/2;3*pi/4],5,25,10,me
sh,1,1,0,1)
%mesh=14;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/4;pi/2;3*pi/4],[pi/4;pi/2;3
*pi/4;pi],6,25,10,mesh,1,1,0,1)
fcn=mesh;
switch fcn
case 1
axis([-1 1 -1 1])
case 2
axis([-1 1 -1 1])
case 3
axis([-6 6 -2 2])
case 4
axis([-4 4 -4 4])
case 5
axis([-1 1 -1 1])
case 6
axis([-4 3 -4 3])
```

```
case 7
axis([-4 4 -4 4])
case 8
axis([-5 5 -5 5])
case 9
axis([-1 1 -1 1])
case 10
axis([-1.5 1.5 -1 1])
case 11
axis([-1 1 -1 2])
case 12
axis([-1 1 -1 1])
case 13
axis([-1 1 -1 1])
case 14
 axis([-1 1 -1 1])
% case 15
% axis([-1 1 -1 1])
% case 16
% axis([-1 1 -1 1])
end
nitri=length(n1)

[coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesNcurvedBoundary(n1,n2,n3,th0,thn,nmax,
numtri,ndiv,mesh)
[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
if color==0
figure(2*mesh)
end
if color==1
figure(2*mesh+1)
end
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
if color==0
for i=1:NEL
for j=1:nnel-1
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

plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=6
midx=mean(xvec(1,1:3))
```

```
midy=mean(yvec(1,1:3))
text(midx,midy,['[',num2str(i),']']);
end
end;%i loop

xlabel('x axis')
ylabel('y axis')
st1='FEM MESH WITH  ';
st2=num2str(NEL);
st3='; 3-node Linear ';
st4='Triangular';
st5=' Elements'
st6='& Nodes='
st7=num2str(NNNODE);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if ndiv<=6
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 ndiv>6
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-ndelete+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



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
%
end%if color==0
%=========================================================================
if color==1
for i=1:NEL
for j=1:nnel-1
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)];
xvect(1,1:3)=[x(1,1),x(1,2),x(1,3)];
yvect(1,1:3)=[y(1,1),y(1,2),y(1,3)];
axis equal
plot(xvec,yvec);%plot element
%*************************************8

 switch tnodes(i,nnel)
case 1
    patch(xvect,yvect,'y'  )
 case 2

   patch(xvect,yvect,'r'  )
 case 3

    patch(xvect,yvect,'g'  )
 case 4

    patch(xvect,yvect,'m'  )

 end
%*******************************************88888888888




hold on;
%place element number
if ndiv<=6
midx=mean(xvec(1,1:3))
midy=mean(yvec(1,1:3))
text(midx,midy,['[',num2str(i),']']);
end
end;%i loop
hold on
xlabel('x axis')
ylabel('y axis')
st1='FEM MESH WITH  ';
st2=num2str(NEL);
st3='; 3-node Linear ';
st4='Triangular';
st5=' Elements'
st6='& Nodes='
st7=num2str(NNNODE);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if ndiv<=6
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 ndiv>6
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-ndelete+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

end%if color==1


%=========================================================================
%PROGRAM-2***********************************************************

function[coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesNcurvedBoundary(n1,n2,n3,tht0,t
htn,nmax,numtri,n,mesh)
%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,...)
fcn=mesh;
switch fcn
case 1
axis([-1 1 -1 1])
case 2
axis([-1 1 -1 1])
case 3
axis([-6 6 -2 2])
case 4
axis([-4 4 -4 4])
case 5
axis([-1 1 -1 1])
case 6
axis([-4 3 -4 3])
case 7
axis([-4 4 -4 4])
case 8
axis([-5 5 -5 5])
case 9
axis([-1 1 -1 1])
case 10
axis([-1.5 1.5 -1 1])
case 11
axis([-1 1 -1 2])
end
nitri=length(n1)
if (nitri>1)
 [eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2(n1,n2,n3,nmax,numtri,n)
end
if (nitri==1)

[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary(n1,n2,n3,nmax,numtri,n)

end
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)])
for itri=1:nitri
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
rrr(:,:,itri)

kk=0;
for ii=1:n+1
   for jj=1:(n+1)-(ii-1)
      kk=kk+1;
```

```
        mm=rrr(ii,jj,itri);
        th0=tht0(itri,1);thn=thtn(itri,1);
        [U,V]=generate_polar_coordinate_over_the_standard_triangle(th0,thn,n,fcn,itri,nitri)
        uu=U(kk,1);vv=V(kk,1);
        xi(mm,1)=uu;
        yi(mm,1)=vv;
    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)
%PROGRAM-3************************************************************
function[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary(n1,n2,n3,nmax,numtri,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,.....
nitri=length(n1)
for itri=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 triangle')
%base edge
kk=nmax;
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)=n3(itri,1);
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;



rr=row_nodes;
rr
rrr(:,:,1)=rr
 edgen1n2(1,1:n+1,1)=rr(1,1:n+1)
 edgen1n3(1:n+1,1,1)=rr(1:n+1,1)
 edgen2n3(1:n+1,1,1)=diag(rr(1:n+1,n+1:-1:1))

%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%if rem
end%for itri=1
ne1=ne;
nmax1=max(max(eln));
%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)];
       trielm(mmm,4)=1;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
       trielm(mmm,4)=2;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
       trielm(mmm,4)=3;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
        trielm(mmm,4)=4;
        tnodes(mmm,1:4)=trielm(mmm,1:4);


     end%switch
   end
 end

eln
trielm
tnodes
nodetel

%PROGRAM-4**********************************************************
function[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2(n1,n2,n3,nmax,numtri,n)
%all intermediate curved triangles can be taken up now
%note that we have computed the following for the first curved triangle
% rrr(:,:,1)=rr
%  edgen1n2(1,1:n+1,1)=rr(1,1:n+1)
%  edgen1n3(1:n+1,1,1)=rr(1:n+1,1)
%  edgen2n3(1:n+1,1,1)=diag(rr(1:n+1,n+1:-1:1))
%they must be further used in curved triangles:2 to nitri-1
%ne=(n/2)^2,the number of six node triangles
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2(n1,n2,n3,nmax,numtri,n)
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2([1;2;3;4],[2;3;4;1],[5;5;5;5],5,1,2)
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2.m
nitri=length(n1)
%if nitri==1
%function[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary(n1,n2,n3,nmax,numtri,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,.....
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary(1,2,3,3,1,2)
%syms mst_tri x
%disp('vertex nodes of triangle')
nitri=length(n1)
if nitri==1
for itri=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 triangle')
%base edge
kk=nmax;
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)=n3(itri,1);
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;
```

```
rr=row_nodes;
rr
rrr(:,:,1)=rr
 edgen1n2(1,1:n+1,1)=rr(1,1:n+1)
 edgen1n3(1:n+1,1,1)=rr(1:n+1,1)
 edgen2n3(1:n+1,1,1)=diag(rr(1:n+1,n+1:-1:1))

%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%if rem
end%for itri=1
 ne1=ne;
nmax1=max(max(eln));
Nmax(1,1)=nmax1;

%=======================================
%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)];
       trielm(mmm,4)=1;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
       trielm(mmm,4)=2;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
```

```
        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)];
        trielm(mmm,4)=3;
        tnodes(mmm,1:4)=trielm(mmm,1:4);
        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)];
         trielm(mmm,4)=4;
         tnodes(mmm,1:4)=trielm(mmm,1:4);


      end%switch
    end
  end

MM=mm;
MMM=mmm;
eln
trielm
tnodes
nodetel

return
end
nitri=length(n1)
if nitri>1
   if n2(nitri,1)~=n1(1,1)
      domain=0;%open domain
   end
   if n2(nitri,1)==n1(1,1)
      domain=1;%closed domain
   end
   ne=0;
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)
if (itri>1)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
% if (itri>1)&(itri==nitri)
%    lmax=nmax;
```

```
% end% if (itri>1)&(itri==nitri)
% mmax=nmax;
%  if itri<nitri
   mmax=max(max(edgen1n2(1:n+1,itri)))
%  end%f itri<nitri
disp('right edge nodes')
 %if itri<nitri
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;
 %end
%if (itri<nitri)|(itri==1)
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)&(itri==1)
if (itri>1)
disp('left edge nodes')
edgen1n3(1:n+1,itri)=edgen2n3(1:n+1,itri-1)
end

if (itri>1)&(domain==0)
 edgen1n3(1:n+1,nitri)=edgen2n3(1:n+1,nitri-1)
end
if (itri>1)&(domain==1)
 edgen2n3(1:n+1,nitri)=edgen2n3(1:n+1,1)
% elm(1:n+1,1)=edgen1n2(1:n+1,1)
end

if (itri==nitri)&(domain==1)
lmax=max(max(edgen1n2(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;
if (itri>1)
rr(1:n+1,1)=edgen2n3(1:n+1,itri-1)
end
 if (itri==nitri)&(domain==1)
%  rr(1:n+1,1)=edgen1n3(1:n+1,1);
for ii=1:n+1
   rr(ii,n+1-(ii-1))= edgen1n3(ii,1);
   rr(ii,1)=edgen2n3(ii,nitri-1);

end
end


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))
Nmax(itri,1)=nmax
%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)];
       trielm(mmm,4)=1;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
       trielm(mmm,4)=2;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
       trielm(mmm,4)=3;
       tnodes(mmm,1:4)=trielm(mmm,1:4);
       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)];
        trielm(mmm,4)=4;
        tnodes(mmm,1:4)=trielm(mmm,1:4);


     end%switch
   end
  end

eln
trielm
tnodes
nodetel

end
end
```

(II)COMMON PROGRAMS REQUIRED FOR TRIANGULATION AND QUADRANGULATION

```
 %PROGRAM-5**************************************************
function[fcn]=fcnt(n,t,itri,nitri)
%
switch n
case 1%
    if 1<=itri<=nitri
   fcn=(0.9+0.1*cos(4*t));
    end
case 2
```

```
      if 1<=itri<=nitri
    fcn=1;
     end

case 3
     if 1<=itri<=nitri
     a=6;b=2;
    fcn=a*b/(sqrt(a^2*(sin(t))^2+b^2*(cos(t))^2));
     end

case 4
     if 1<=itri<=nitri
    fcn=sqrt(9.25+3*cos(4*t));
     end

case 5
     if 1<=itri<=nitri
    fcn=sqrt((0.6*cos(t)+0.3*cos(2*t)-0.2)^2+0.36*(sin(t))^2);
     end

 case 6
     if 1<=itri<=nitri
    fcn=sqrt(5-4*cos(t));
     end

case 7
     if 1<=itri<=nitri
    fcn=sqrt(10-6*cos(2*t));
     end

case 8
     if 1<=itri<=nitri
    fcn=sqrt(17-8*cos(3*t));
     end

case 9
     if 1<=itri<=nitri
    fcn=1;
     end

case 10
     if 1<=itri<=nitri
    fcn=sqrt(cos(2*t)+sqrt((1.1)^4-(sin(2*t))^2));
     end

case 11
     if 1<=itri<=nitri
    fcn=0.25*sin(t)+0.5*sqrt(1-0.9*(cos(t))^2)+0.5*sqrt(1-0.7*(cos(t))^2)
     end

case 12
     if 1<=itri<=nitri
    fcn=sqrt((9.25+3*cos(4*t))/12.25);
     end
%
case 13

if 1<=itri<=nitri
     switch itri
         case 1

    fcn=1/(cos(t)+sin(t));%x+y=1
         case 2

    fcn=-1/(cos(t)-sin(t))%x-y=-1
```

```
        case 3

    fcn=-1/(cos(t)+sin(t));%x+y=-1
        case 4

    fcn=1/(cos(t)-sin(t));%x-y=1

    end
end
case 14
     switch itri
        case 1

    fcn=1/(cos(t));%x=1

        case 2

    fcn=1/(sin(t))%y=1

       case 3

    fcn=-1/(cos(t));%x=-1


        case 4

    fcn=-1/(sin(t));%y=-1
     end
end
case 15
 if 1<=itri<=nitri
     switch itri
        case 1
    fcn=(0.9+0.1*cos(4*t));
        case 2
    fcn=1;
        case 3
    fcn=-1/(cos(t)+sin(t));%x+y=-1
        case 4
     fcn=sqrt((9.25+3*cos(4*t))/12.25);
        end
 end
otherwise
   disp('something wrong')
end

end

%PROGRAM-6************************************************************

function[U,V]=generate_polar_coordinate_over_the_standard_triangle(th0,thn,n,fcn,itri,nitri)
%generate_polar_coordinate_over_the_standard_triangle(th0,thn,n)
%generate_polar_coordinate_over_the_standard_triangle(0,pi/2,20)
%generate_polar_coordinate_over_the_standard_triangle(pi/2,pi,20)
%generate_polar_coordinate_over_the_standard_triangle(0,pi/2,20,1)
%generate_polar_coordinate_over_the_standard_triangle(pi/2,pi,20,1)
%generate_polar_coordinate_over_the_standard_triangle(pi,3*pi/2,20,1)
%generate_polar_coordinate_over_the_standard_triangle(3*pi/2,2*pi,20,1)
switch fcn
case 1
axis([-1 1 -1 1])
case 2
axis([-1 1 -1 1])
case 3
axis([-6 6 -2 2])
```

```
case 4
axis([-4 4 -4 4])
case 5
axis([-1 1 -1 1])
case 6
axis([-4 3 -4 3])
case 7
axis([-4 4 -4 4])
case 8
axis([-5 5 -5 5])
case 9
axis([-1 1 -1 1])
case 10
axis([-1.5 1.5 -1 1])
case 11
axis([-1 1 -1 2])
case 12
axis([-1 1 -1 1])
case 13
axis([-1 1 -1 1])
 case 14
 axis([-1 1 -1 1])
 case 15
 axis([-1 1 -1 1])


end
ui(1:n+1,1:n+1)=zeros(n+1,n+1);vi(1:n+1,1:n+1)=zeros(n+1,n+1);


kk=0;
for j=1:n+1

if (n-j+1)>=1
dth=(thn-th0)/(n-j+1);

for k=1:n-j+2
  th(k,1)=th0+(k-1)*dth;
 end


for i=1:(n-j+2)
    kk=kk+1;
    ti=th(i,1)

if ((fcn~=3)|(fcn~=9))
fcnti=fcnt(fcn,ti,itri,nitri)
ui(i,j)=cos(ti)*fcnti*(n-j+2)/(n+1);
vi(i,j)=sin(ti)*fcnti*(n-j+2)/(n+1);
end
if fcn==3
    a=6;b=2;
fcnti=fcnt(fcn,ti,itri,nitri)
ui(i,j)=cos(ti)*a*(n-j+2)/(n+1);
vi(i,j)=sin(ti)*b*(n-j+2)/(n+1);
end
if fcn==9
    a=1;
fcnti=fcnt(fcn,ti,itri,nitri)
ui(i,j)=((3*cos(ti)+cos(3*ti))/4)*a*((n-j+2)/(n+1))^1.5;
vi(i,j)=((3*sin(ti)-sin(3*ti))/4)*a*((n-j+2)/(n+1))^1.5;
%  ui(i,j)=((3*cos(ti)+cos(3*ti))/4)*a*((n-j+2)/(n+1))^0.5;
%  vi(i,j)=((3*sin(ti)-sin(3*ti))/4)*a*((n-j+2)/(n+1))^0.5;

end
```

```
 U(kk,1)=ui(i,j);
 V(kk,1)=vi(i,j);
 N(i,j)=i;

end


end
if (n-j+1)==0
 ui(1,n+1)=0;
 vi(1,n+1)=0;
kk=kk+1;
U(kk,1)=0;
V(kk,1)=0;
end
ui
vi

  U'
  V'

  kk
 figure(2*fcn-1),scatter(U(:,1),V(:,1),10,'filled','r')
  hold on
  N
end
end

%=======================================
```

(III)COMPUTER PROGRAMS FOR QUADRANGULATION

```
%PROGRAM-7*****************************************************

function[]=triangular_mesh4LinearTriangles_t3Nq4curvedBoundary(n1,n2,n3,th0,thn,nmax,numtri,ndiv,mesh,xlength,ylength,nd
elete,color)
%triangular_mesh4LinearTriangles_t3Nq4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength,ndelete)
%triangular_mesh4LinearTriangles_t3Nq4([1],[2],[3],3,1,2,6,1,1,0)
%triangular_mesh4LinearTriangles_t3Nq4([1],[2],[3],3,1,2,7,1,1,0)
%triangular_mesh4LinearTriangles_t3([1],[2],[3],3,1,2,7,1,1,0)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[0],[pi/2],3,25,10,1,1,1,0)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[0],[pi/2],3,9,6,1,1,1,0)
%mesh=1;triangular_mesh4LinearTriangles_t3curvedBoundary([1;2;3;4],[2;3;4;1],[5;5;5;5],[0;pi/2;pi;3*pi/2],[pi/2;pi;3*pi/2;2*pi
],5,1,2,mesh,1,1,0)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[0],[pi/2],3,9,6,1,1,1,0,0)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[0],[pi/2],3,9,6,1,1,1,0,1)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[pi/2],[pi],3,9,6,1,1,1,0,1)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[pi],[3*pi/2],3,9,6,1,1,1,0,1)
%triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1],[2],[3],[3*pi/2],[2*pi],3,9,6,1,1,1,0,1)
%mesh=1;triangular_mesh4LinearTriangles_t3Nq4curvedBoundary([1;2],[2;3],[4;4],[0;pi/2],[pi/2;pi],4,4,4,mesh,1,1,0,0)
fcn=mesh;

switch fcn
case 1
axis([-1 1 -1 1])
case 2
axis([-1 1 -1 1])
case 3
axis([-6 6 -2 2])
case 4
axis([-4 4 -4 4])
case 5
axis([-1 1 -1 1])
case 6
axis([-4 3 -4 3])
```

```
case 7
axis([-4 4 -4 4])
case 8
axis([-5 5 -5 5])
case 9
axis([-1 1 -1 1])
case 10
axis([-1.5 1.5 -1 1])
case 11
axis([-1 1 -1 2])
end

nitri=length(n1)

[coord,gcoord,tnodes,spqd,nodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesNq4curvedboundary(n1,n2,n3,th0,thn,nmax,numtri,ndiv,mesh)

 [nel,nnel]=size(nodes);

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
if color==0
figure(2*mesh)
end
if color==1
figure(2*mesh+1)
end
NDEL=ndelete*4*numtri*3
NEL=nel-NDEL;
NNODE=max(max(nodes(1:NEL,1:4)));
NNNODE=NNODE
if (ndelete>1)
NNNODE=NNODE-ndelete+1
end
if color==0
for i=1:NEL
for j=1:nnel-1
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,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)];
axis equal
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=6
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='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<=6
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 ndiv>6
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-ndelete+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=1:4
     if(nodes(iel,jel)>nnmax)
        nodes(iel,jel)=nodes(iel,jel)-(ndelete-1);
     end
   end
end
end
nodes
end%if color==0
```

```
if color==1
for i=1:NEL
for j=1:nnel-1
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,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)];
xvect(1,1:4)=[x(1,1),x(1,2),x(1,3),x(1,4)];
yvect(1,1:4)=[y(1,1),y(1,2),y(1,3),y(1,4)];
axis equal
plot(xvec,yvec);%plot element
hold on;
 switch nodes(i,nnel)
case 1
    patch(xvect,yvect,'y'  )
 case 2

   patch(xvect,yvect,'r'  )
 case 3

    patch(xvect,yvect,'g'  )


 end
%*****************************************88888888888

hold on;
%place element number
if ndiv<=6
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='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<=6
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 ndiv>6
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-ndelete+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



NEL
NNNODE
GCOORD
%element nodal connectivity matrix
if (ndelete>1)
nnmax=nmax-(ndelete-1)
for iel=1:NEL
    for jel=1:4
      if(nodes(iel,jel)>nnmax)
         nodes(iel,jel)=nodes(iel,jel)-(ndelete-1);
      end
    end
end
end
nodes
end%if color==1
%PROGRAM-8**********************************************************

function[coord,gcoord,tnodes,spqd,nodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesNq4curvedboundary(n
1,n2,n3,tht0,thtn,nmax,numtri,n,mesh)
%[coord,gcoord,tnodes,nodetel,nnode,nel]=triangulation4polygonal_domain_coordinatesNcurvedBoundary(n1,n2,n3,tht0,thtn,nm
ax,numtri,n,mesh)
%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

fcn=mesh;

switch fcn
case 1
axis([-1 1 -1 1])
case 2
axis([-1 1 -1 1])
case 3
axis([-6 6 -2 2])
case 4
axis([-4 4 -4 4])
case 5
```

```
axis([-1 1 -1 1])

end
nitri=length(n1)
if (nitri>1)
%[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles_trial(n1,n2,n3,nmax,numtri,n);

%[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles_trialNq4(n1,n2,n3,nmax,numtri,n);
[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2Nq4(n1,n2,n3,nmax,nu
mtri,n)
end
if (nitri==1)
   %[eln,trielm,rrr,tnodes,nodetel]=nodaladdresses_for4XnXn_LinearTriangles(n);
   %[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNq4(n)
   [eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundaryNq4(n)
end
%[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
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==3)
   nitri=1;
end
for itri=1:nitri
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
rrr(:,:,itri)
kk=0;
for ii=1:n+1
   for jj=1:(n+1)-(ii-1)
      kk=kk+1;
      mm=rrr(ii,jj,itri);
      th0=tht0(itri,1);thn=thtn(itri,1);
      [U,V]=generate_polar_coordinate_over_the_standard_triangle(th0,thn,n,fcn,itri,nitri)
      uu=U(kk,1);vv=V(kk,1);
      xi(mm,1)=uu;
      yi(mm,1)=vv;
   end
end
[xi yi]
%add coordinates of centroid and midside nodes of a triangular element
 ne=4*(n/2)^2;
% stdnode=kk;
 for iii=1+(itri-1)*ne:ne*itri
    kk=kk+1;
   node1=trielm(iii,1);
   node2=trielm(iii,2);
   node3=trielm(iii,3);
   mm4=trielm(iii,4);
   mm5=trielm(iii,5);
   mm6=trielm(iii,6);
```

```
   mm7=trielm(iii,7);
   %
   xi(mm4,1)=(xi(node1,1)+xi(node2,1))/2;
   yi(mm4,1)=(yi(node1,1)+yi(node2,1))/2;
   %
   xi(mm5,1)=(xi(node2,1)+xi(node3,1))/2;
   yi(mm5,1)=(yi(node2,1)+yi(node3,1))/2;
   %
   xi(mm6,1)=(xi(node1,1)+xi(node3,1))/2;
   yi(mm6,1)=(yi(node1,1)+yi(node3,1))/2;
   %
   xi(mm7,1)=(xi(node1,1)+xi(node2,1)+xi(node3,1))/3;
   yi(mm7,1)=(yi(node1,1)+yi(node2,1)+yi(node3,1))/3;

 end

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)

%PROGRAM-9**********************************************************
function[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundaryNq4(n)
% nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundaryNq4.m
%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;
edgen1n2(1,1:n+1)=rr(1,1:n+1)
edgen1n3(1:n+1,1)=rr(1:n+1,1)
edgen2n3(1:n+1,1)=diag(rr(1:n+1,n+1:-1:1))

%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
%

%


mmm=0;
for iel=1:ne
   for jel=1:4
        switch jel
     case 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


ne=mmm;
%CHECK FOR TRIANGLES
if ne~=4*(n/2)^2
   disp('error in computing 3-node triangles')
end
nnd=(n+1)*(n+2)/2;
for inum=1:nnd
   for jnum=1:nnd
      mdpt(inum,jnum)=0;
   end
end
nd=nnd;
for nnn=1:ne
   mmm1=trielm(nnn,1);
   mmm2=trielm(nnn,2);
   mmm3=trielm(nnn,3);
```

```
%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;
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;
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;
end
nd=nd+1;
   trielm(nnn,4)=mdpt(mmm1,mmm2);
   trielm(nnn,5)=mdpt(mmm2,mmm3);
   trielm(nnn,6)=mdpt(mmm3,mmm1);
   trielm(nnn,7)=nd;
end
trielm

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

for mmm=1:mm
    spqd(:,1:5)
 end
ss2='number of 4-node special convex quadrilaterals =';
[p2,q2]=size(spqd);
disp([ss2 num2str(p2)])
[p2 12*(n/2)^2]
if (p2~=12*(n/2)^2)
    [p2 12*(n/2)^2]
    disp('ERROR IN ELEMENT NODAL CONNECTIVITY')
end
rr
```

```
edgen1n2(1,1:n+1)=rr(1,1:n+1)
edgen1n3(1:n+1,1)=rr(1:n+1,1)
edgen2n3(1:n+1,1)=diag(rr(1:n+1,n+1:-1:1))

%PROGRAM-10************************************************
function[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2Nq4(n1,n2,n3,n
max,numtri,n)
%[eln,trielm,rrr,tnodes,spqd,nodes,nodetel]=nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundaryNq4(n)
%all intermediate curved triangles can be taken up now
%note that we have computed the following for the first curved triangle
% rrr(:,:,1)=rr
%  edgen1n2(1,1:n+1,1)=rr(1,1:n+1)
%  edgen1n3(1:n+1,1,1)=rr(1:n+1,1)
%  edgen2n3(1:n+1,1,1)=diag(rr(1:n+1,n+1:-1:1))
%they must be further used in curved triangles:2 to nitri-1
%ne=(n/2)^2,the number of six node triangles
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2(n1,n2,n3,nmax,numtri,n)
%nodaladdresses_for4XnXn_LinearTrianglesNcurvedboundary2([1;2;3;4],[2;3;4;1],[5;5;5;5],5,1,2)
%=========================================================================

nitri=length(n1)
if nitri>1
   if n2(nitri,1)~=n1(1,1)
      domain=0;%open domain
   end
   if n2(nitri,1)==n1(1,1)
      domain=1;%closed domain
   end
   ne=0;
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)
if (itri>1)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
   mmax=max(max(edgen1n2(1:n+1,itri)))
%  end%f itri<nitri
disp('right edge nodes')
 %if itri<nitri
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;
 %end
%if (itri<nitri)|(itri==1)
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)&(itri==1)
if (itri>1)
disp('left edge nodes')
edgen1n3(1:n+1,itri)=edgen2n3(1:n+1,itri-1)
end

if (itri>1)&(domain==0)
 edgen1n3(1:n+1,nitri)=edgen2n3(1:n+1,nitri-1)
end
if (itri>1)&(domain==1)
 edgen2n3(1:n+1,nitri)=edgen2n3(1:n+1,1)
% elm(1:n+1,1)=edgen1n2(1:n+1,1)
end

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

%**************************************
%**************************************
if (itri==nitri)&(domain==1)
lmax=max(max(edgen1n2(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;
if (itri>1)
rr(1:n+1,1)=edgen2n3(1:n+1,itri-1)
end
 if (itri==nitri)&(domain==1)
%  rr(1:n+1,1)=edgen1n3(1:n+1,1);
for ii=1:n+1
   rr(ii,n+1-(ii-1))= edgen1n3(ii,1);
   rr(ii,1)=edgen2n3(ii,nitri-1);

end
end


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))
Nmax(itri,1)=nmax
end%itri
%=========================================================
%generate 4-Linear triangles in a 6-node triangles,call these as trielm
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

end
%=============================================================================
ne=mmm;
%CHECK FOR TRIANGLES
if ne~=4*nitri*(n/2)^2
   disp('error in computing 3-node triangles')
end
for inum=1:nnd
   for jnum=1:nnd
      mdpt(inum,jnum)=0;
   end
end
nd=nnd;
for nnn=1:ne
   mmm1=trielm(nnn,1);
   mmm2=trielm(nnn,2);
   mmm3=trielm(nnn,3);
```

```
%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;
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;
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;
end
nd=nd+1;
   trielm(nnn,4)=mdpt(mmm1,mmm2);
   trielm(nnn,5)=mdpt(mmm2,mmm3);
   trielm(nnn,6)=mdpt(mmm3,mmm1);
   trielm(nnn,7)=nd;
end
trielm

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

for mmm=1:mm
    spqd(:,1:5)
 end
ss2='number of 4-node special convex quadrilaterals =';
[p2,q2]=size(spqd);
disp([ss2 num2str(p2)])

%=========================================================================
end
```