www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 6 June 2015, Page No. 12792-12848

A New Approach to Automatic Generation of All Quadrilateral Finite Element Mesh for Planar Multiply Connected Regions

H.T. Rathod [a*] ***, K.V.Vijayakumar*** [b] ***, A. S. Hariprasad*** [c] ***, K. Sugantha Devi*** [d] ***, C.S.Nagabhushana*** [e]

[a] Department of Mathematics, Central College Campus, Bangalore University, Bangalore -560001, Karnataka State, India.
Email: htrathod2010@gmail.com

[b] Department of Mathematics,B.M.S.Institute of Technology,Avalahalli, Bangalore-560064, Karnataka State, India.
Email: kallurvijayakumar@gmail.com

[c] Department of Mathematics,Dr.S.M.C.E,Bangalore 562132, Karnataka State, India.
Email: ashariprasad@yahoo.co.in
[d] Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post, Kolar Gold Field, Kolar District, Karnataka state, Pin- 563120, India.
Email: suganthadevik@yahoo.co.in

[e] Department of Mathematics, HKBK College of Engineering, Nagavara, Bangalore – 560045, India.
Email: csnagabhusana@gmail.com

Abstract

. A new approach for the automatic generation and refinement of finite element meshes over multiply connected planar regions has been developed. This paper represents continuation of authors research activities in that area. An algorithm for producing a triangular mesh in a convex polygon is presented in authors recent work. It is used for the finite element triangulation of a complex polygonal region of the plane decomposed into convex polygons. We decompose the convex polygonal regions into simple sub regions in the shape of triangles. These simple regions are then triangulated to generate a fine mesh of triangular elements. We then propose an automatic triangular to quadrilateral conversion scheme.In this scheme, 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 a the barycentre of the element. To preserve the mesh conformity, a similar procedure is also applied to every triangle of the domain to fully discretize the given complex polygonal domain into all quadrilaterals, thus propagating uniform refinement. This simple method generates a mesh whose elements confirm well to the requested shape by refining the problem domain. We have modified these algorithms and demonstrated their use by generating high quality meshes for some typical multiply connected regions: **square domains with regular polygonal holes inside and vice versa.** We have also made improvements and modifications to to the above triangulation algorithm of the triangle which can now triangulate a trapezium cut out of a triangle. This new algorithm on the triangulation of a trapezium cut out of a triangle is applied to quadrangulate the planar regions in the shape of a **circular annulus and square domain with a square hole inside.** We have appended MATLAB programs which incorporate the mesh generation schemes developed in this paper. These programs provide valuable output on the nodal coordinates, element connectivity and graphic display of the all quadrilateral mesh for application to finite element analysis.

Keywords: finite elements, triangulation ,quadrilateral mesh generation,convex, nonconvex, and multiply connected polygonal domains,square domain with regular polygonal holes, circular annulus

1. INTRODUCTION

The finite element method (FEM) developed in the 1950's as a method to calculate the elastic deformations in solids. Sixty years later, the point of view is more abstract which allows FEM to be used as a general purpose method applicable to all kinds of partial differential equations. The advent of modern computer technologies provided a powerful tool in numerical simulations for a range of problems in partial differential equations over arbitrary complex domains. Finite element meshes are used in the Finite Element Method (FEM), a versatile and powerful numerical procedure to analyze complex structures and continua in many scientific and engineering fields. Finite Element Analysis (FEA) is widely used for many fields including structures and optimization. The FEA in engineering applications comprises three phases: domain discretization, equation solving and error analysis. The domain discretization or mesh generation is the preprocessing phase which plays an important role in the achievement of accurate solutions.

FEM is now used as a general purpose method applicable to all kinds of partial differential equations. The advent of modern computer technologies provided a powerful tool in numerical simulations for a range of problems in partial differential equations over arbitrary complex domains. A mesh is required for finite element method as it uses finite elements of a domain for analysis. A mesh is a discretization of a geometric domain into small simple shapes, such as triangles or quadrilaterals in two dimensions and tetrahedra or hexahedra in three. Meshes find use in many application areas. In geography and cartography, meshes give compact representations of terrain data. In computer graphics, most objects are ultimately reduced to meshes before rendering. Finally, meshes are almost essential in the numerical solution of partial differential equations arising in physical simulation. We can easily distinguish two types of planar domains. For us, a simple polygon includes both boundary and interior. A polygon with holes is a simple polygon minus the interiors of some other simple polygons; its boundary has more than one connected component.

Finite Element Analysis (FEA) is widely used in many fields including structures and optimization. The FEA in engineering applications comprises three phases: domain discretization, equation solving and error analysis. The domain discretization or mesh generation is the preprocessing phase which plays an important role in the achievement of accurate solutions.

FEM requires dividing the analysis region into many sub regions. These small regions are the elements which are connected with adjacent elements at their nodes. Mesh generation is a procedure of generating the geometric data of the elements and their nodes, and involves computing the coordinates of nodes, defining their connectivity and thus constructing the elements. Hence mesh designates aggregates of elements, nodes and lines representing their connectivity. Though the FEM is a powerful and versatile tool, its usefulness is often hampered by the need to generate a mesh. Creating a mesh is the first step in a wide range of applications,including scientific and engineering computing and computer graphics. But generating a mesh can be very time consuming and prone to error if done manually. In recognition of this problem a large number of methods have been devised to automate the mesh generation task. An attempt to create a fully automatic mesh generator that is capable of generating a valid finite element meshes over arbitrary complex domains and needs only the information of the specified geometric boundary of the domain and the element size, started from the pioneering work [1] in the early 1970's. Since then many methodologies have been proposed and different algorithms have been devised in the development of automatic mesh generators [2-4]. In order to perform a reliable finite element simulation a number of researchers [5-7] have made efforts to develop adaptive FEA method which integrates with error estimation and automatic mesh modification. Traditionally adaptive mesh generation process is started from coarse mesh which gives large discretization error levels and takes a lot of iterations to get a desired final mesh. The research literature on the subject is vast and different techniques have been proposed [8,21-24], as several engineering applications to real world problems cannot be defined on a rectangular domain or solved on a structured square mesh. The description and discretization of the design domain geometry, specification of the boundary conditions for the governing state equation, and accurate computation of the design response may require the use of unstructured meshes.**We may note that there are many mesh generation methods but no one method can be considered a universal tool. Mesh generation requirements greatly depending on the specific application and its computational domain features.**

An unstructured simplex mesh requires a choice of mesh points (vertex nodes) and triangulation. Many mesh generators produce a mesh of triangles by first creating all the nodes and then connecting nodes to form triangles. The question arises as to what is the 'best' triangulation on a given set of points. One particular scheme, namely Delaunay triangulation [8], is considered by many researchers to be most suitable for finite element analysis. If the problem domain is a subset of the Cartesian plane, triangular or quadrilateral meshes are typically employed.

The method used for mesh generation can greatly affect the quality of the resulting mesh. Usually the geometry and physical problem of the domain direct the user which method to apply. The real problems in 2D and 3D involve the complex topology, and distribution of the boundary conditions. Such situation requires automatic mesh generator to reduce the user influence to this process as much as possible. The advancing front is another popular mesh generation method that can be used for adapting FE mesh strategies. Conceptually , the advancing front method is one of the simplest mesh generation processes. This element generating algorithm starts from an initial front formed from the specified boundary of the domain and then generates elements, one by one, as the front advances into the region to be discretized until the whole domain is completely covered by elements [9-10]. In general, good quality meshes of quadrilateral elements cannot be directly obtained from these meshing techniques. An additional step is therefore required to obtain quadrilateral meshes from the triangular meshes. It is generally known that FEA using quadrilateral mesh is more accurate than that of a triangular one [11-20].

In a recent works[25-26], a novel mesh generation scheme of all quadrilateral elements for convex polygonal domains is presented. This scheme converts the elements in background triangular mesh into quadrilaterals through the operation of splitting. This scheme first decompose the convex polygon into simple subregions in the shape of triangles. These simple subregions are then triangulated and assembled to form a convx polygon and thus generate a fine mesh of triangles.The scheme also proposes 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 edges and a vertex at the barrycentre of the triangular element. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this fully discretizes the given convex polygonal domain into all quadrilaterals, thus propogating uniform refinement.

In the present paper,we propose the automatic generation and refinement of finite element meshes over multiply connected planar regions. This paper represents continuation of authors research activities in that area. An algorithm for producing a triangular mesh in a convex polygon is first presented in authors earlier work[25]. It can be used for the finite element triangulation of a complex polygonal region of the plane which is decomposed into convex polygons.We can decompose the convex polygonal regions into simple sub regions in the shape of triangles. These simple regions are then triangulated to generate a fine mesh of triangular elements. We then propose an automatic triangular to quadrilateral conversion scheme. 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 element. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given complex polygonal domain into all quadrilaterals, thus propagating uniform refinement. This simple method generates a mesh whose elements confirm well to the requested shape by refining the problem domain.We have modified these algorithms and demonstrated their use by generating high quality meshes for some typical multiply connected regions: **square domains with regular polygonal holes inside and vice versa.**We have considered the regular polygonal shapes:**rhombus**, **pentagon,hexagon,heptagon and octagon**. We have also made improvements and modifications to to the above triangulation algorithm of our recent paper[25], this algorithm can now triangulate a trapezium cut out of a triangle. This new algorithm on the triangulation of a trapezium cut out of a triangle is applied to the regions in the shape of a circular annulus and a square domain with a square hole inside and the layered polygonal domains. We have appended MATLAB programs which incorporate the mesh generation schemes developed in this paper. These programs provide valuable output on the nodal coordinates ,element connectivity and graphic display of the all quadrilateral mesh for application to finite element analysis.

In section 2 of this paper, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of six node triangular elements. In section 3,we explain the procedure to split these triangles into quadrilaterals. In section 4,we have presented a method of piecing together of all triangular subregions and eventually creating an all quadrilateral mesh for the given convex polygonal domain. This is just necessary to generate an all quadrilateral mesh over a single convex polygon.In section 5,we have proposed the scheme to assemble the convex polygons to form a multiply connected domain and we have also shown that a nonconvex polygon can be formed by assembling a convx polygon and a cracked polygon. We first present the schemes to generate an all quadrilateral mesh for a convex polygon. This is necessary to understand the mesh generation of the multiply connected domains, for the cracked polygon , the nonconvex polygon and the mesh generation for square domains with regular polygon hole inside and a pentagon domain with a square hole inside which are possible by joining several convex polygonal domains and the concept of sections 2- 5. In section 6 ,the the layered domains such as circular annulus and square domain with a square hole inside are generated using the new concept of trapezium assembly proposed here. In section 7,we briefly discuss the types of meshes that can be generated using the algorithms of sections 2-6. In section 8 , we display the various meshes generated by using all the algorithms developed in this paper.

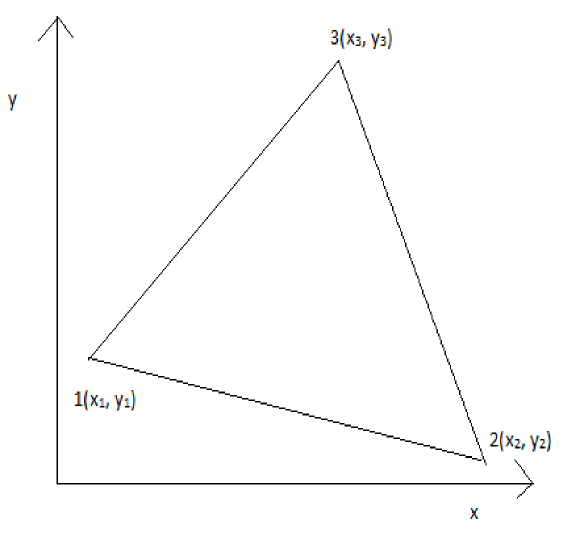
2. Division of an Arbitrary Triangle

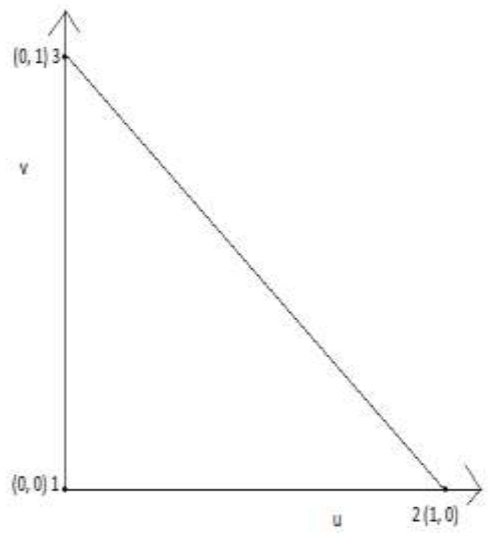
We can map an arbitrary triangle with vertices ($(x_i, y_i),\ i = 1, 2, 3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 1a, 1b. The necessary transformation is given by the equations.

$$x = x_1 + (x_2 - x_1)u + (x_3 - x_1)v$$

$$y = y_1 + (y_2 - y_1)u + (y_3 - y_1)v \qquad (1)$$

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



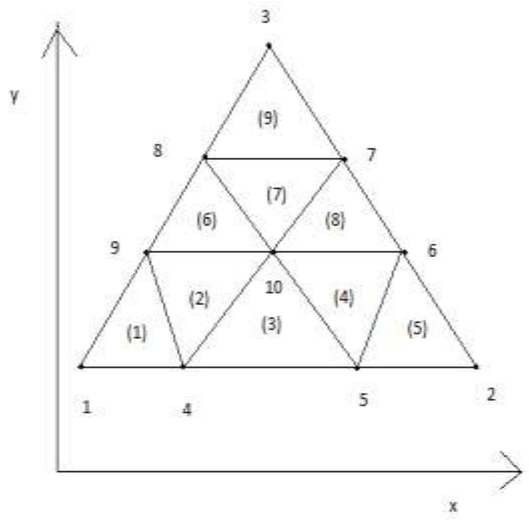


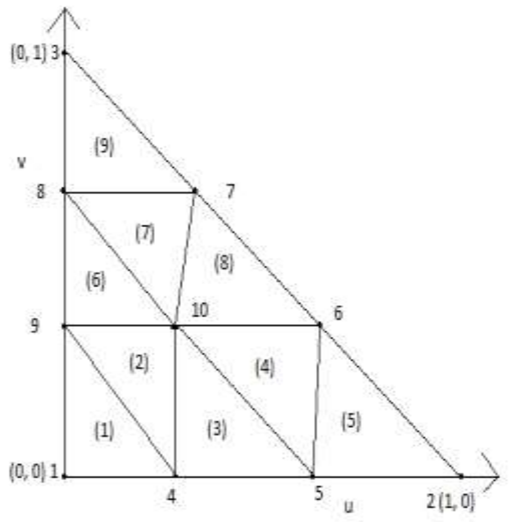
1.a 1. b

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

Fig. 1b A Right Isosceles Triangle i

n the (u, v)





2(a) 2(b)

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

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

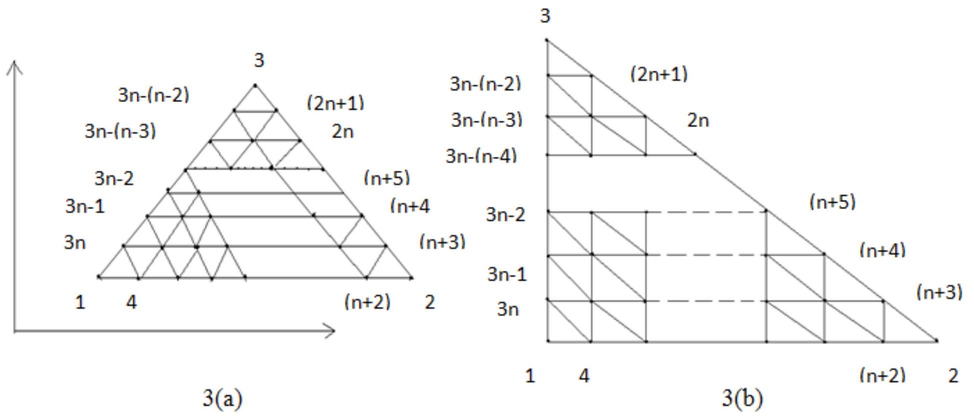


Fig. 3a Division of an arbitrary triangle into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisions of equal length

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

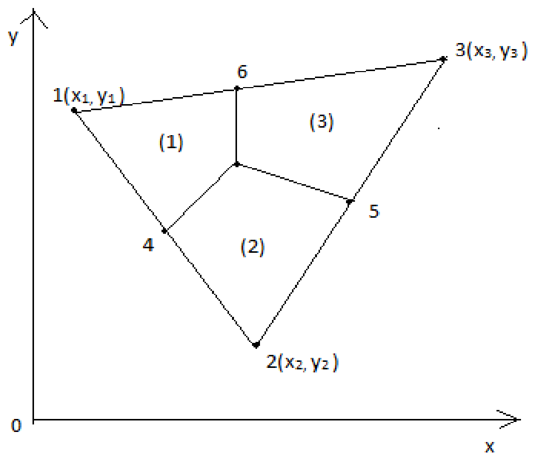
We have shown the division of an arbitrary triangle in Fig. 3a , Fig. 3b, We divided each side of the triangles (either in Cartesian space or natural space) into n equal parts and draw lines parallel to the sides of the triangles. This creates (n+1) (n+2) nodes. These nodes are numbered from triangle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

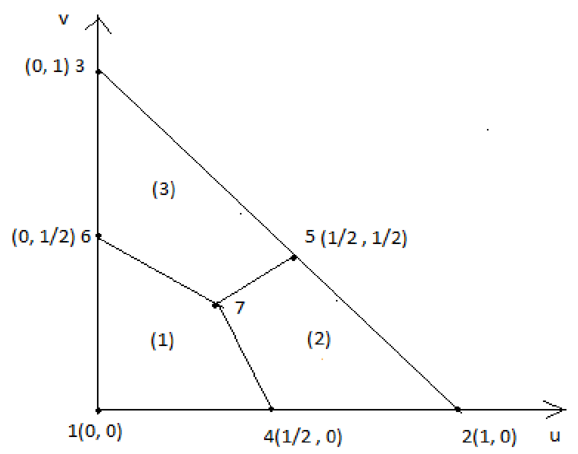
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \ldots\ldots\ldots\ldots\ldots\ldots, & & (n+2), & 2 \\ 3n, & (3n+1), \ldots\ldots\ldots\ldots\ldots\ldots & , 3n+(n-2), & & (n+3), & 0 \\ 3n-1, & 3n+(n-1), \ldots\ldots\ldots & , 3n+(n-2)+(n-3), & (n+4), & 0, & 0 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots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

3. Quadrangulation of an Arbitrary Triangle

We now consider the quadrangulation of an arbitrary triangle. We first divide the arbitrary triangle into a number of equal size six node triangles. Let us define l_{ij} as the line joining the points (x_i, y_i) and (x_j, y_j) in the Cartesian space (x, y). Then the arbitrary triangle with vertices at $((x_i, y_i), i = 1,2,3)$ is bounded by three lines l_{12}, l_{23}, and l_{31} . By dividing the sides l_{12}, l_{23}, l_{31} into $n = 2m$ divisions (m, an integer) creates m^2 six node triangular divisions. Then by joining the centroid of these six node triangles to the midpoints of their sides, we obtain three quadrilaterals for each of these triangle. We have illustrated this process for the two and four divisions of l_{12}, l_{23}, and l_{31} sides of the arbitrary and standard triangles in Figs. 4 and 5

Two Divisions of Each side of an Arbitrary Triangle



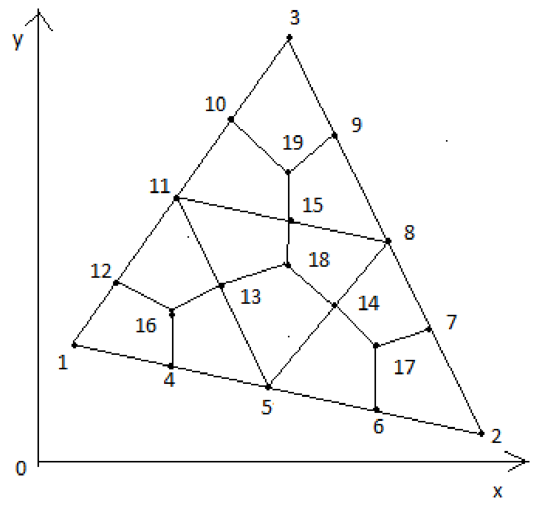


4(a) 4(b)

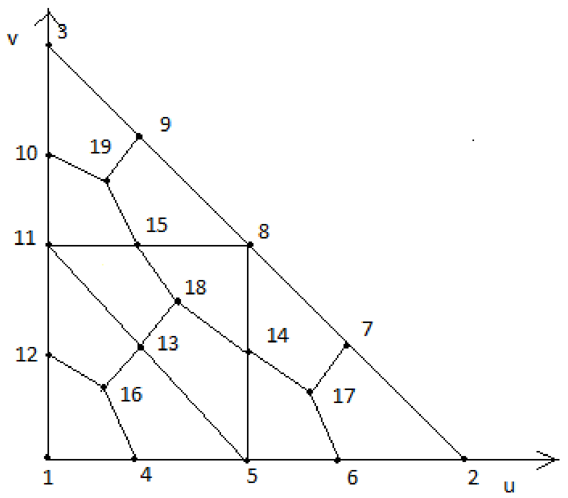
Fig 4(a). Division of an arbitrary triangle into three quadrilaterals

Fig 4(b). Division of a standard triangle into three quadrilaterals

Four Divisions of Each side of an Arbitrary Triangle



5(a)



5(b)

Fig 5(a). Division of an arbitrary triangle into 4 six node triangles

Fig 5(b). Division of a standard triangle into 4 right isosceles triangle

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

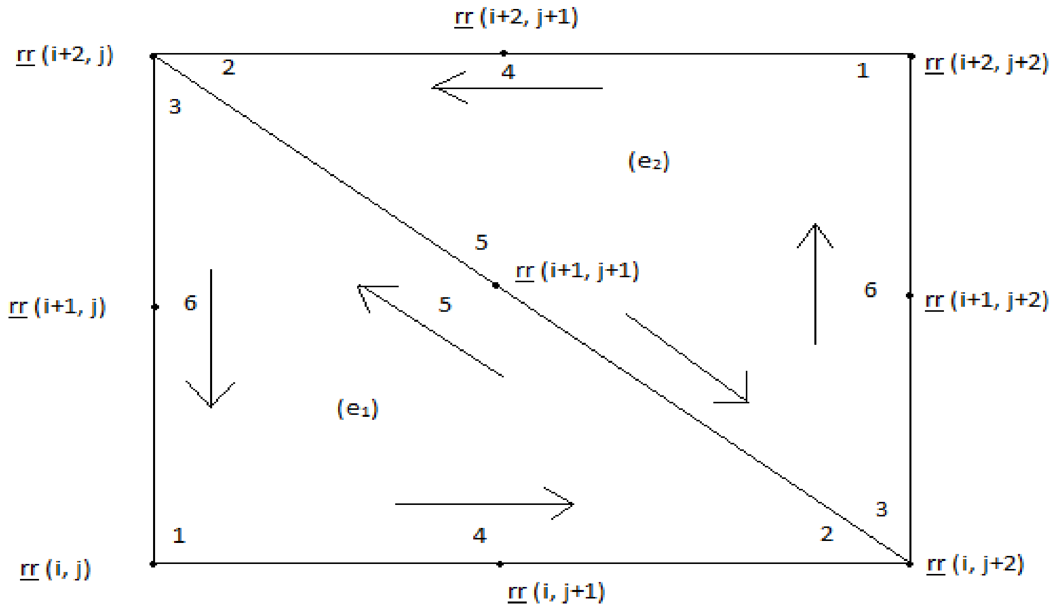


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

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

(e₁) $< \underline{rr}\ (i,\ j), \underline{rr}\ (i, i+2), \underline{rr}\ (i+2,\ j), \underline{rr}\ (i, j+1), \underline{rr}\ (i+1, j+1), \underline{rr}\ (i+1, j) >$

(e₂) $< \underline{rr}\ (i+2, j+2), \underline{rr}\ (i+2, j), \underline{rr}\ (i, j+2), \underline{rr}\ (i+2, j+1), \underline{rr}\ (i+1, j+1), \underline{rr}$

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

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 standard earlier, we can construct two six node triangles. We can create three quadrilaterals in each of these six node triangles. The nodal connectivity for the 3 quadrilaterals created in (e₁) are given as

$Q_{3n_1-2} < c_1, \underline{rr}\ (i+1,\ j),\ \underline{rr}\ (i,\ j), \underline{rr}\ (i, j+1) >$

$Q_{3n_1-1} < c_1, \underline{rr}\ (i,\ j+1),\ \underline{rr}\ (i,\ j+2), \underline{rr}\ (i+1, j+1) >$

$Q_{3n_1} < c_1, \underline{rr}\ (i+1,\ j+1),\ \underline{rr}\ (i+2,\ j), \underline{rr}\ (i+1, j) >$

and the nodal connectivity for the 3 quadrilaterals created in (e₂) are given as

$Q_{3n_2-2} < c_2, \underline{rr}\ (i+1,\ j+2),\ \underline{rr}\ (i+2,\ j+2), \underline{rr}\ (i+2,\ j+1) >$

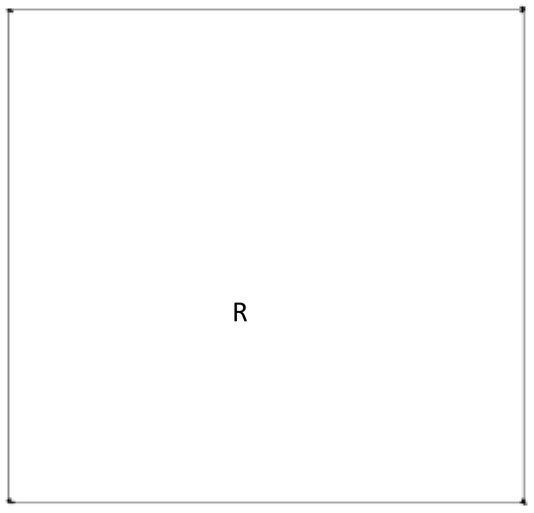
$Q_{3n_2-1} < c_2, \underline{rr}\ (i+2,\ j+1),\ \underline{rr}\ (i+2,\ j), \underline{rr}\ (i+1,\ j+1) >$

$Q_{3n_1} < c_2, \underline{rr}\ (i+1,\ j+1),\ \underline{rr}\ (i,\ j+2), \underline{rr}\ (i+1, j+2) >$ -------------------- (5)

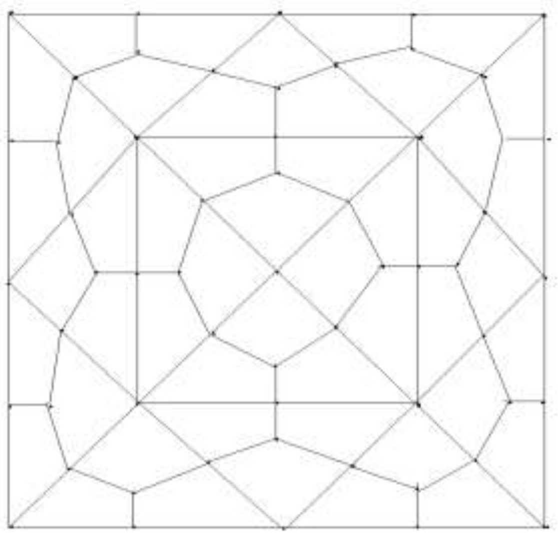
4. Quadrangulation of the Polygonal Domain

We can generate polygonal meshes by piecing together triangular with straight sides. Subsection (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. 8(a). This is a a square region which is simply chosen for illustration. We divide this region into four LOOPs as shown in Fig.8(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. 8(c).The complete mesh is shown in Fig.8(b)

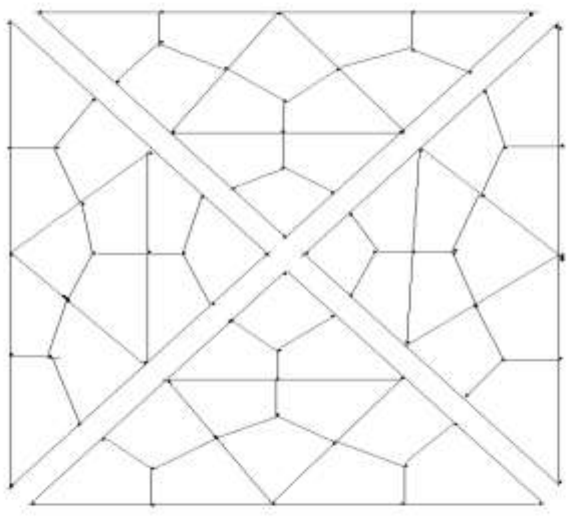


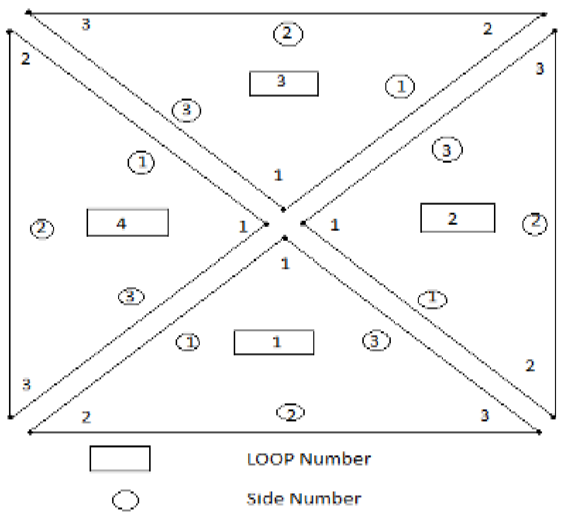
8 (a)



8 (a)

(i)Fig. 8(a) Region R to be analyzed (ii) Fig. 8(b) Example of completed mesh





8(c) 8 (d)

(iii)Fig.8(c) Exploded view showing four loops

(iv)Fig.8(d) Example of a loop and side numbering scheme

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

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

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

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

5. Mesh Generation for linear multiply connected domains

Let us consider a plane domain which is divisible into a finite number of linear polygons,say,n.
Now,Let $P_m{}^n$refer to the n-th linear polygon having m sides.Let $T_i{}^n, i = 1,2, \ldots . m$ refer to the i-th triangle of $P_m{}^n$ and $P_m{}^n$= $\sum_{i=1}^{i=m} T_i{}^n$.We can have m=3,4,.....etc.,since an arbitrary linear triangle can be further divided into three triangles.

In our recent work[ ],we considered a simple multiply connected domain with linear boundary : **a square duct which is made-up of 12 squares**. We divide each square into four triangles as shown in Fig. . Every triangle can be transformed into a right isosceles triangle which has two adjacent sides and a hypotenuse.In conformity with this notion,every triangle has two adjacent sides, say side 1-2 and side 1-3 ;and a hypotenuse,say,side 2-3. The square duct shown in Fig. 9 has 48-triangles.We can express this as a sum, $\sum_{n=1}^{n=12} P^{n}{}_{4}$=$\sum_{n=1}^{n=12} \sum_{i=1}^{i=4} T_{i}{}^{n}$. We can set up a counter on triangles,thus for a fixed m,(m=4,for the present example) :$T_{i}{}^{n}$=$T_{(n-1)m+i}$,*i=1,2,*

Mesh Generation Over a Square Duct

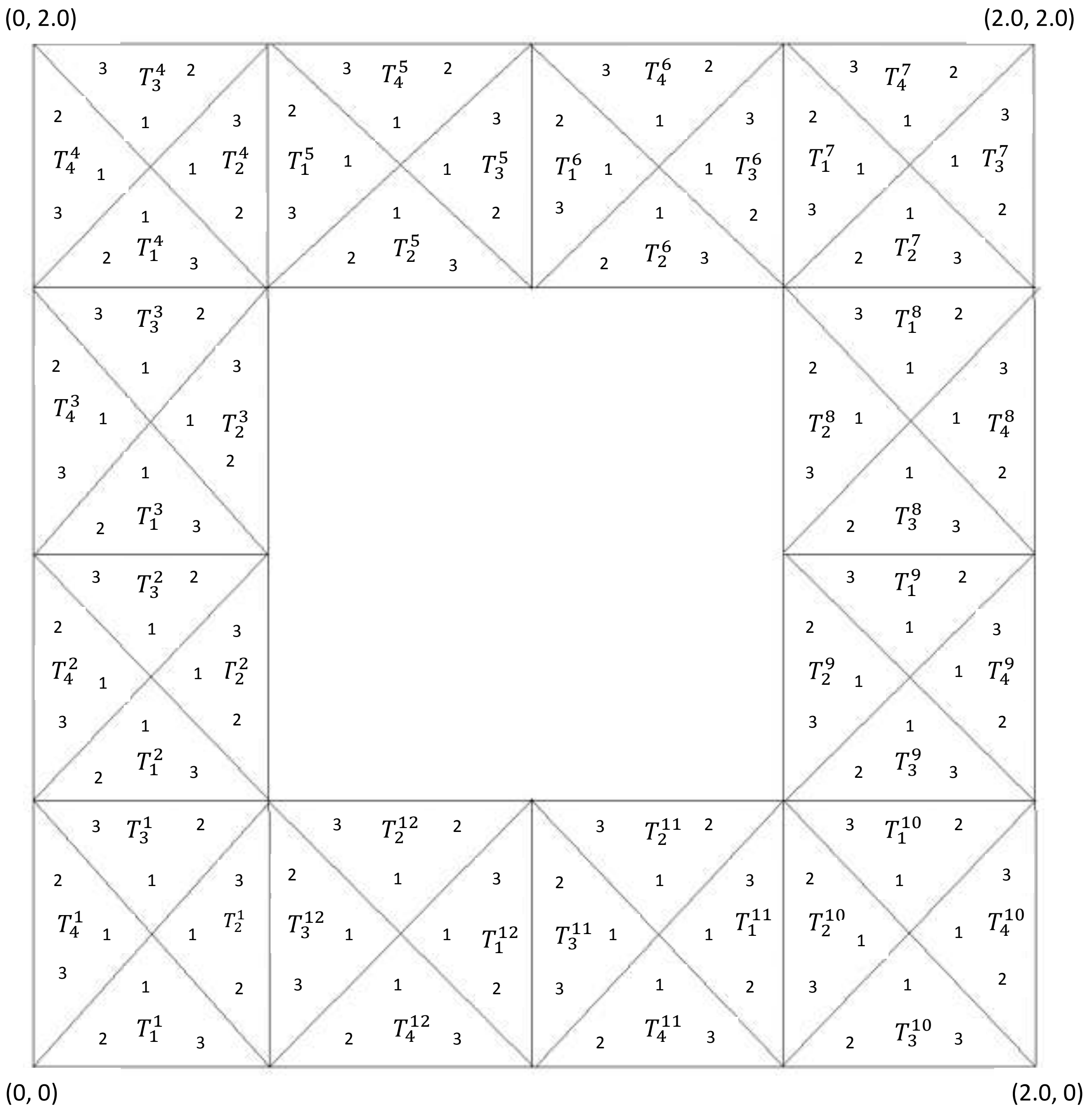

(0, 2.0)
(2.0, 2.0)
(0, 0)
(2.0, 0)
T_3^4 T_4^4 T_2^4 T_1^4
T_4^5 T_1^5 T_3^5 T_2^5
T_4^6 T_1^6 T_3^6 T_2^6
T_4^7 T_1^7 T_3^7 T_2^7
T_3^3 T_4^3 T_2^3 T_1^3
T_1^8 T_2^8 T_4^8 T_3^8
T_3^2 T_4^2 T_2^2 T_1^2
T_1^9 T_2^9 T_4^9 T_3^9
T_3^1 T_4^1 T_2^1 T_1^1
T_2^{12} T_3^{12} T_1^{12} T_4^{12}
T_2^{11} T_3^{11} T_1^{11} T_4^{11}
T_1^{10} T_2^{10} T_4^{10} T_3^{10}

Fig. 9 A Square duct made-up of 12 squares
$(\boldsymbol{P_4^n}, \boldsymbol{n = 1, 2, ----, 12})$ ***where*** $\boldsymbol{P_4^n} = \sum_{i=1}^{4} \boldsymbol{T_i^n}$ $\boldsymbol{, (n = 1, 2, ----, 12)}$

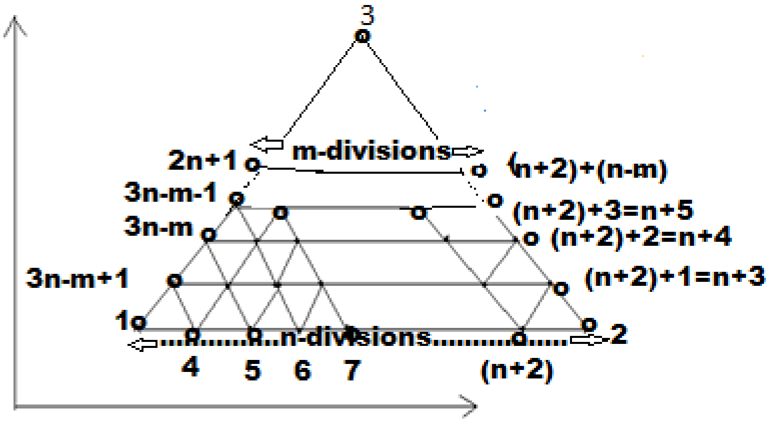
The above algorithm is already translated into computer programs which can join several convex polygons to form the simple multiply connected domain of a square duct consisting of 12-squares.
These MATLAB programs can be referred in our earlier work[ ].

6 Division of a Trapezim

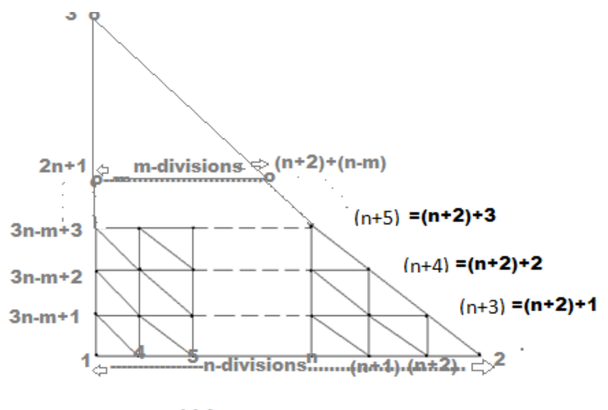
We consider a triangle and note the following important theorem on midsegment. **Definition**:The **midsegmen**t of a triangle is a segment whose endpoints are both midpoints of sides.

Theorem: Every triangle has three midsegments. The midsegment of a triangle is always parallel to the third side (the side whose midpoint it doesn't include), and half as long as the third side.

Application of the above result implies that a trapezium is created by taking a part of the triangle divided as in section 2 of this paper. This can be further explained. Let us consider an arbitrary triangle. We divide one of the sides of this triangle into **n** equal parts and the remaining other two sides into **(n-m+1)** equal divisions. This creates a trapezium cut out of a triangle.This trapezium has n divisions on the base segment and it will have **m** divisions on the top segment.The adjacent sides have (n-m) divisions. The resulting trapezium is shown in Fig.11 a-b . The triangle with n equal divisions on all sides has $\boldsymbol{n^2}$ subtriangles and $(\boldsymbol{n}+\mathbf{1})(\boldsymbol{n}+\mathbf{2})/\mathbf{2}$ nodes.This is already shown in Fig.3a-b . We have modified the existing programs to include this concept. We have generated the meshes for square domain with a hole inside and a circular annulus by application of these programs.We explain this with reference to the mesh generation for a square domain with a square hole inside. We have also generated a FEM mesh for a circular annulus whose inner and outer circular boundaries are approximated by straight line edges.The elements along the boundary can be easily converted to curved triangles with two straight sides and one curved side or curved quadrilaterals with three straight sides.



11 (a)



11(b)

Fig.11(a-b)Division of trapeziums(a partial arbitrary triangle& a partial right isoscles triangle) into n-division on bottom side,m-division on top side and (n-m) divisions on adjacent sides

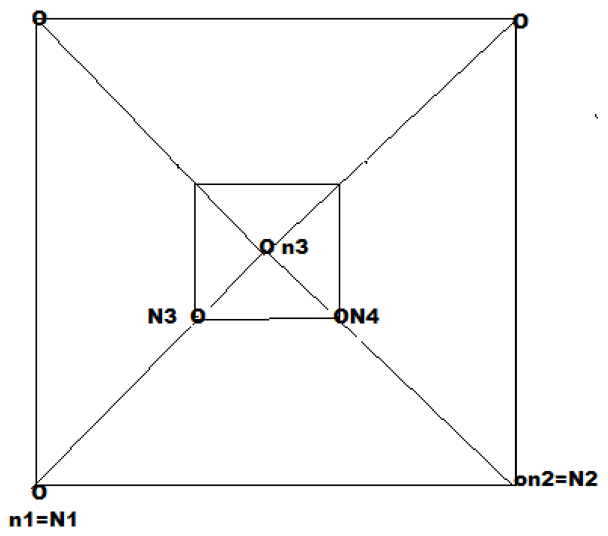
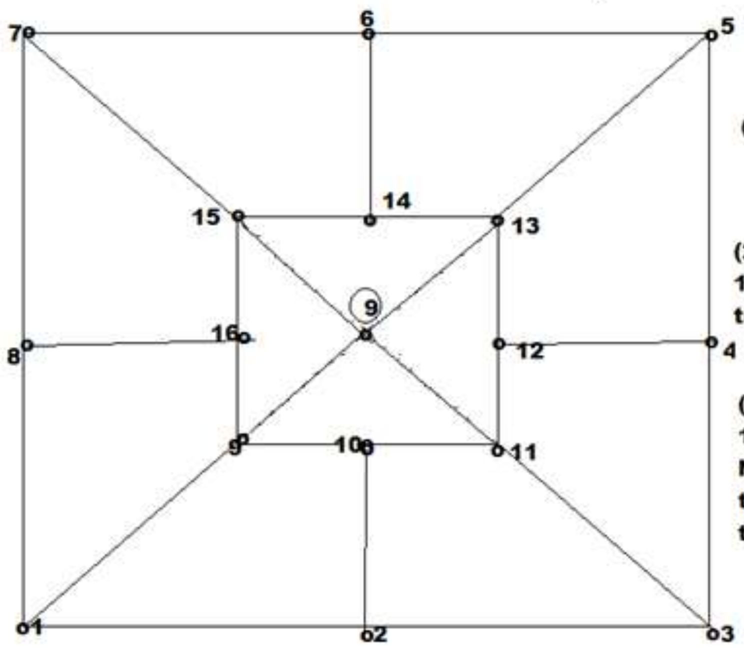


Fig.12(a)n1,n2,n3 refer to the vertex nodes of the triangle and N1,N2,N3,N4 refer to vertex nodes of the trapezium



Note:

(1) For all the covering triangles $\boxed{9}$ refers to nodal vertex n3

(2)Node numbers 1,2,3,4,5,6,7,8 refer to n1 and n2 for the covering triangles

(3)Node numbers 1,2,.....,16 refer to N1,N2,N3,N4 for the covering trapeziums

Fig.12(b)FEM COARSE MESH-TRAPEZIUM ASSEMBLY

We now explain this procedure with reference to a square domain with a square hole inside.We know that division of a triangle into n^2 subtriangles can be made by knowing its three coordinate vertices.This input is also sufficient to create the divisions of a trapezium cut out of the same triangle.But to describe a trapezium cut out,we need four vertices.Let us call the nodal addresses as n1,n2,n3 and the nodal vertices of the trapezium cut out as N1,N2,N3,N4. We use nodal addresses of n1,n2,n3 and their corresponding coordinates to create the trapezium cut out of the triangle, since the trapezium cut out of the triangle and the triangle emnate from the same point(coordinate of nodal address n3).We may further note that N1=n1,N2=n2 and n3=max(n1,n2)+1 which we have called as nmax. Whereas max(N1,N2,N3,N4) is called by a different name, say,kmax. We have the following sample input for the coarse FEM mesh of Fig.12(b) in the MATLAB program written for thispurpose writtenforthispurpose.

```
n1=[1;2;3;4;5;6;7;8];
n2=[2;3;4;5;6;7;8;1];
n3=[(9); (9); (9); (9); (9); (9); (9); (9)]
```
We have the coordinate of (9)=(0.5,0.5) for Fig.12(b) and nodal addresses of N1,N2,N3,N4 are
```
N1=[1;2;3;4;5;6;7;8];
N2=[2;3;4;5;6;7;8;1];
N3=[9;10;11;12;13;14;15;16];
N4=[10;11;12;13;14;15;16;1];
```
The MATLAB program **M-FILES** developed for this purpose are:

(1) `quadrilateral_mesh4convexpolygoneightsidesq4trapezium.m`
(2) `polygonal_domain_QUADcoordinatestrapezium.m`
(3) `nodaladdresses_special_convex_quadrilaterals4atrapezium.m`
(4) `generate_area_coordinate_over_the_standard_triangle.m`
(5) `trapezium_mesh.m`
(6) `trapezium_mesh0.m`
(7) `square_mesh.m`
(8) square_mesh1.m

7.0 APPLICATIONS OF MESH GENERATIONS

We have made further improvements in the already published programs which are covered in sections 2-5.They refer to mesh generation of planar domains which are either closed or open.They refer to a incomplete convex poygon which is generated by partial revolving around a geometric point and a convex polygon which is generated by completely revolving around a geometric pont.We have demonstrated the methods of generating multiply connected domains for these cases. We have then demonstrated the mesh generation for the planar domains which require the joining of several convex polygons.The applications are typical examples for square domain with regular polygonal holes inside.The regular polygons considered have the shape of a rhombus,pentagon,hexagon,heptagon and octagon.We then consider the mesh generation by application of the procedure explained on trapezium cut out of a triangle .We have demonstrated the procedure by generating a square domain with a square hole inside.We have also applied this technique to generate a circular annulus with linear inner and outer boundaries.The three sets of MATLAB programs are necessary for this purpose.The mesh generation example which require application of revolving the triangles partially or completely and joining several polygons together is already considered in our earlier works[].We have further demonstrated the potential of these studies in the mesh generation examples of sections 8-9.The application of section 6 is demonstrated by mesh generation examples of section 10.We now list the MATLAB PROGRAMS based on the algorithm of section 6.The following programs are developed for this purpose.

7.1 Domain combining several Linear Triangles

We use the idea that a multiply connected domain with internal hole can also be generated by the ideas proposed in section 4 ,an example of a square domain with a internal hole and a square domain with external holes are presented to demonstrate this concept

7.2 Domains Combining Several Linear Convex Polygons

We apply the algorithm of section 5 to the automesh generation for some examples of practical importance.This can be made possible by joining several convex polygons. The domain is a square with a hole

inside.The examples of such a mesh generation can appear in various complexity. Here we have considered a square domain with regular polygon hole inside and one example of a regular polygon with a square hole inside. They are:
(1)Square domain with a rhombus hole
(2) Square domain with a pentagon hole
(3) Square domain with a hexaagon hole
(4) Square domain with a heptagon hole
(5) Square domain with a octagon hole
(6)**Pentagon domain with a square hole**
The MATLAB program for sections 7.1 and 7.2 are already available in our earlier works[,   ].

7.3 Generating Layered Domains

We apply the algorithm of section 6 to generate square domain with a square hole inside and a circular annulus.This algorithm joins several trapeziums emanating from a common point. The required programs are listed as Appendix to this paper. We now list the MATLAB PROGRAMS based on the algorithm of section 6.The programs developed for this purpose are already listed.

8.0 APPLICATION EXAMPLES ON AUTOMESH GENERATION

We now present some typical automesh generation examples of all quadrilateral meshes for frinite element analysis .These examples cover the concepts proposed in sections 7.1 ,7.2 and 7.3

8.1   Examples on multiply connected FEM **Meshes: Emanating from one geometric point**

The meshes shown below emanate from one geometric point and they combining several triangles around a point. We have used the concept of section 4 in this process. We present an initial mesh and then the refinements of the same mesh.One can easily find the input data like nodal connections and cartesian coordinates of the nodes from this initial fem mesh for the domain. . The Matlab programs of the authors earlier paper[ ] is used for this purpose

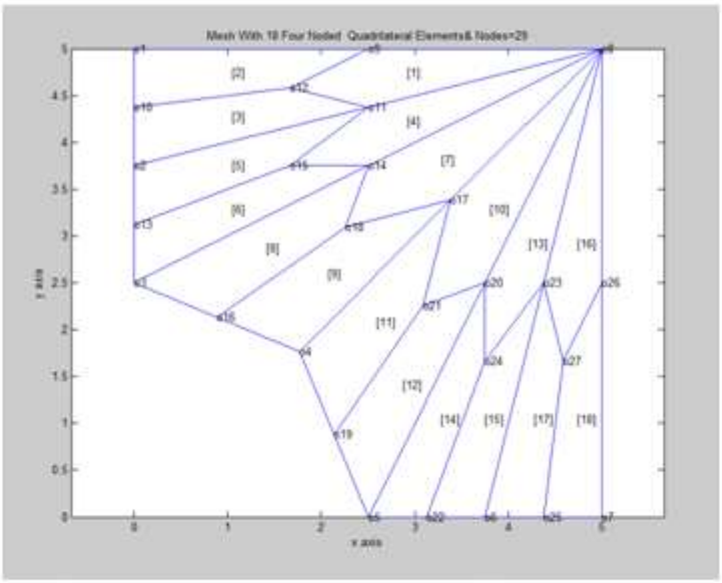


Fig. Initial FEM mesh for a square doman with a quarter of coarse circular hole

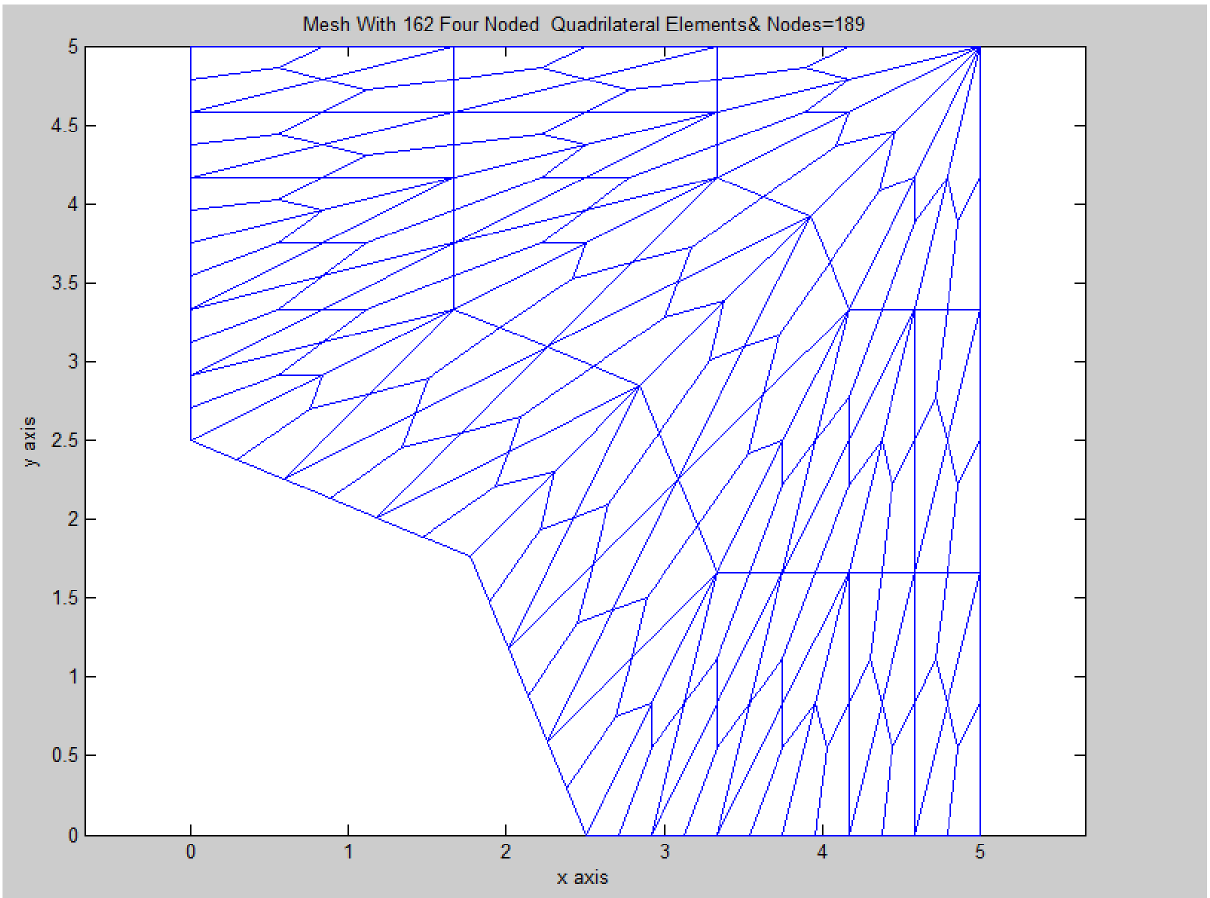


Fig. Fine Mesh For A square with a quarter of a coarse circular hole

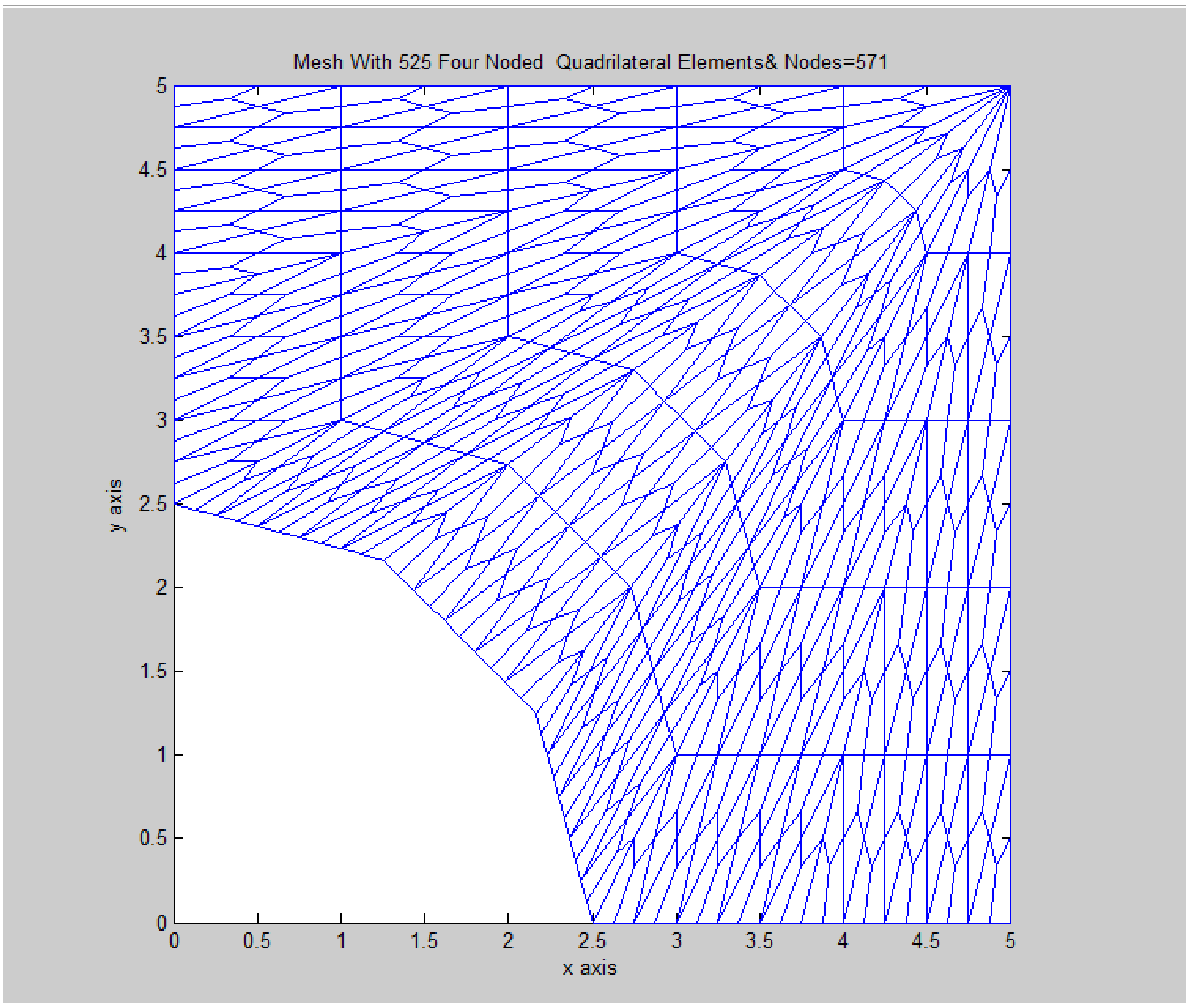


Fig. A very fine mesh for a square domain with a coarse quarter of a circular hole

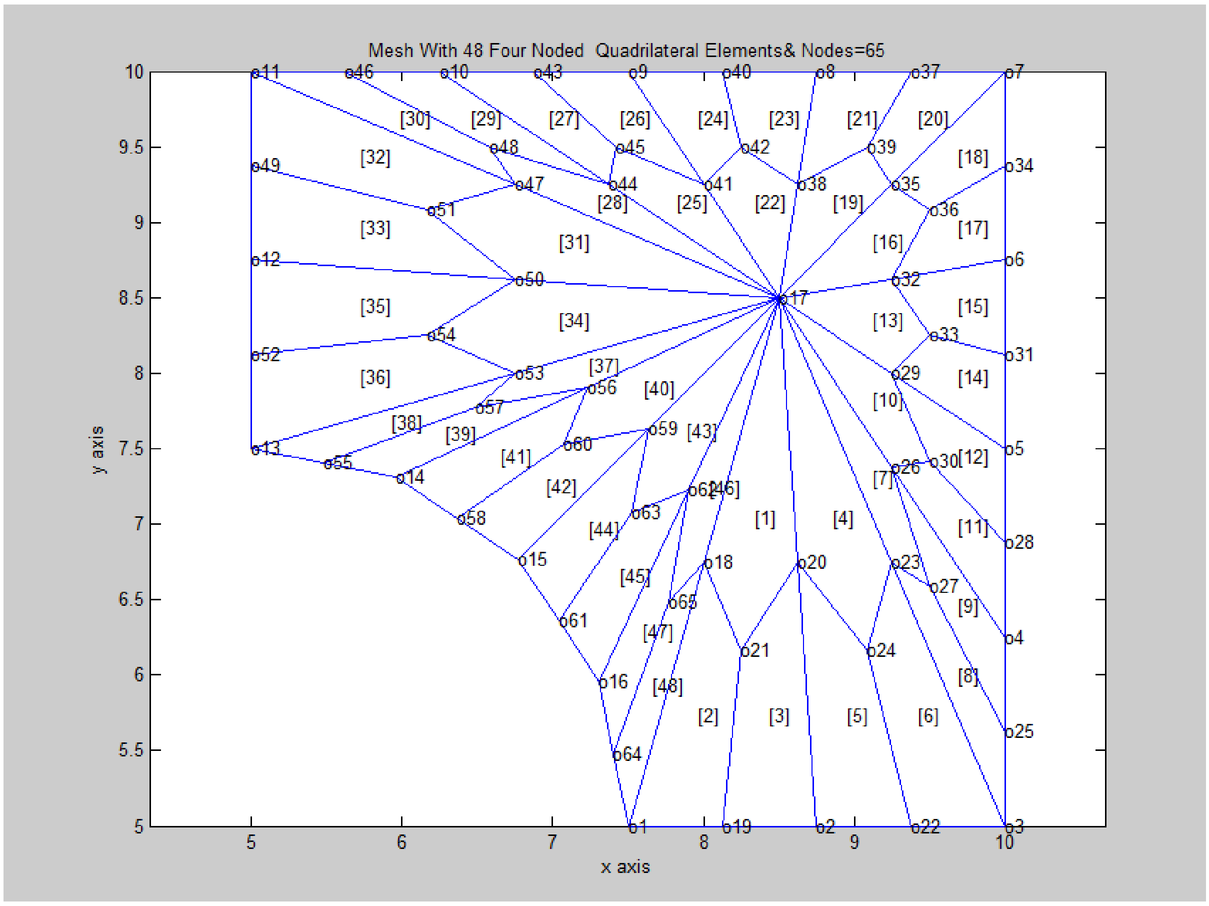


Fig. Initial Mesh for a square domain with a coarse quarter of a circular hole

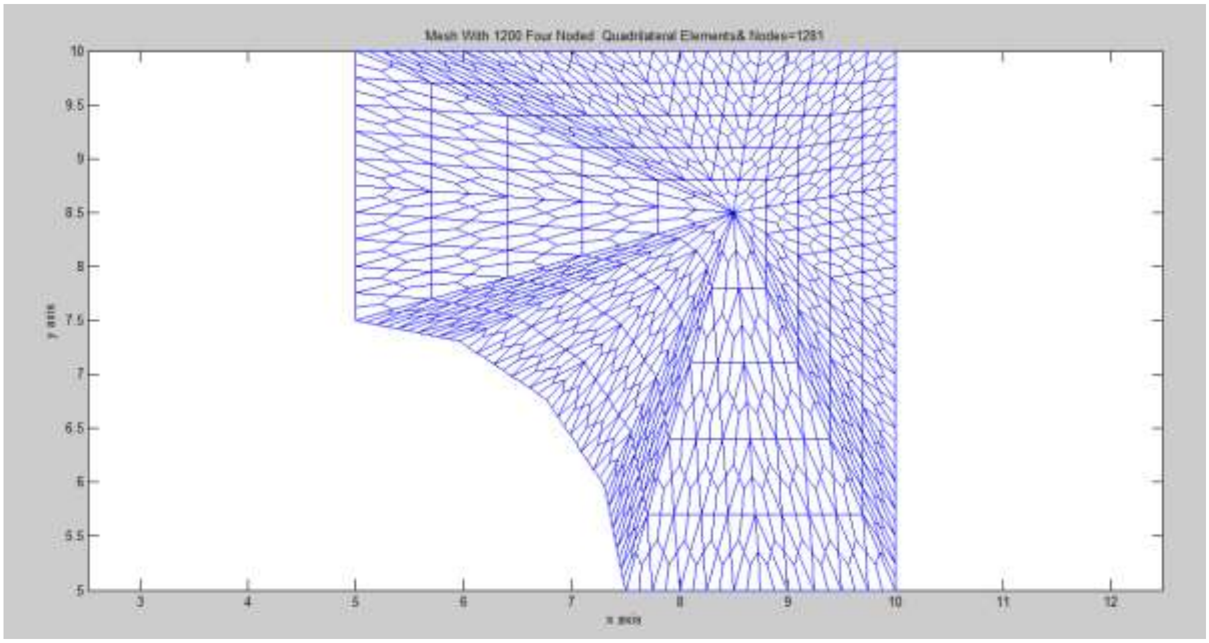

Mesh With 1200 Four Noded Quadrilateral Elements& Nodes=1281
x axis
y axis

Fig. Fine Mesh for a square domain with a coarse quarter of a circular hole

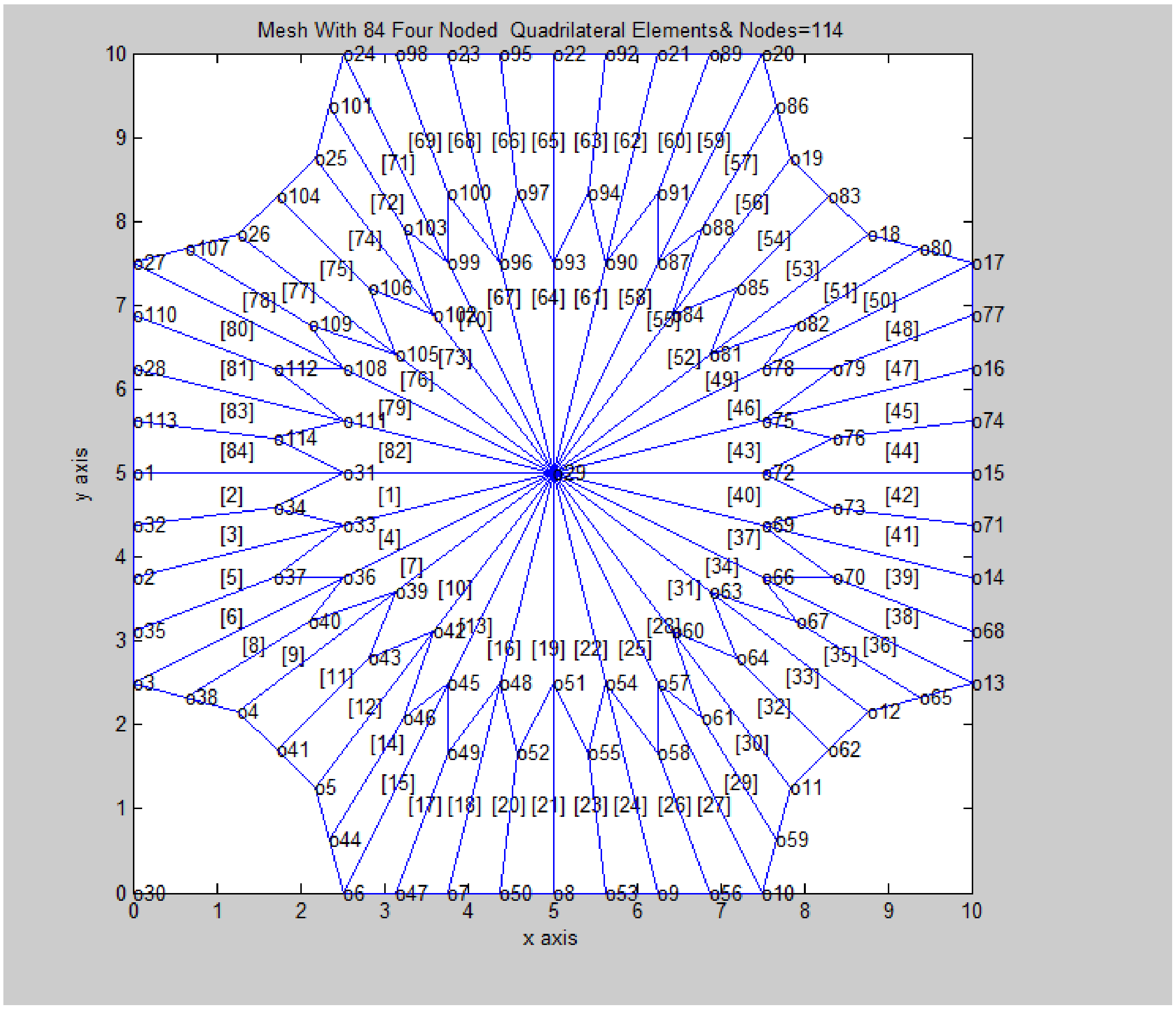


Fig. Initial FEM mesh for a Square domain with coarse quarter circles as corner cut outs

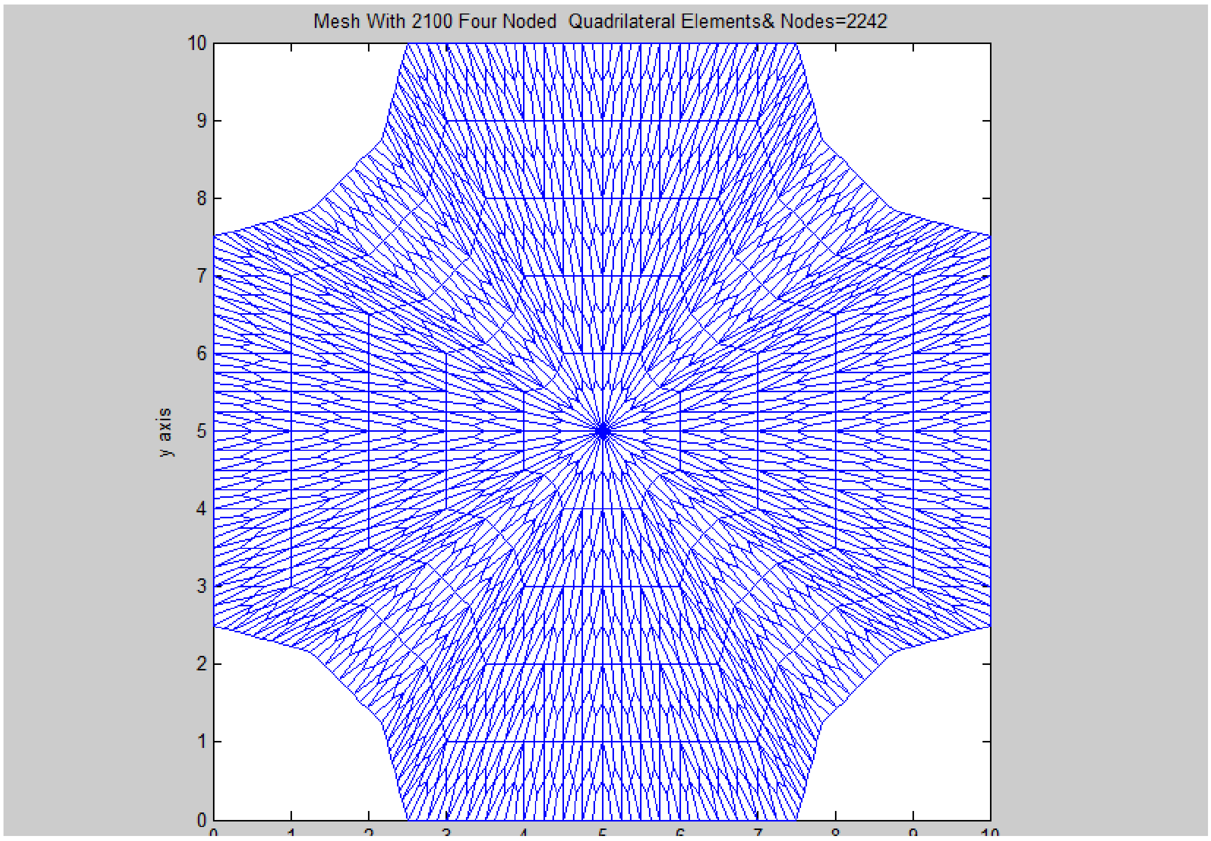

Mesh With 2100 Four Noded Quadrilateral Elements& Nodes=2242
y axis
0
1
2
3
4
5
6
7
8
9
10

Fig. Fine Mesh for Square domain with coarse quarter circles as corner cut outs

8.2 Examples on multiply connected FEM Meshes: Emanating from several geometric points

The meshes shown below emanate from several geometric points and they combining several convex polygons and each of these convex polygons is the creation of a domain surrounded by several linear triangles.We have used the concept of section 5 in this process. We present an initial mesh and then the refinements of the same mesh.One can easily find the input data like nodal connections and cartesian coordinates of the nodes from this initial fem mesh for the domain. The Matlab programs of the authors earlier paper[ ] is used for this purpose

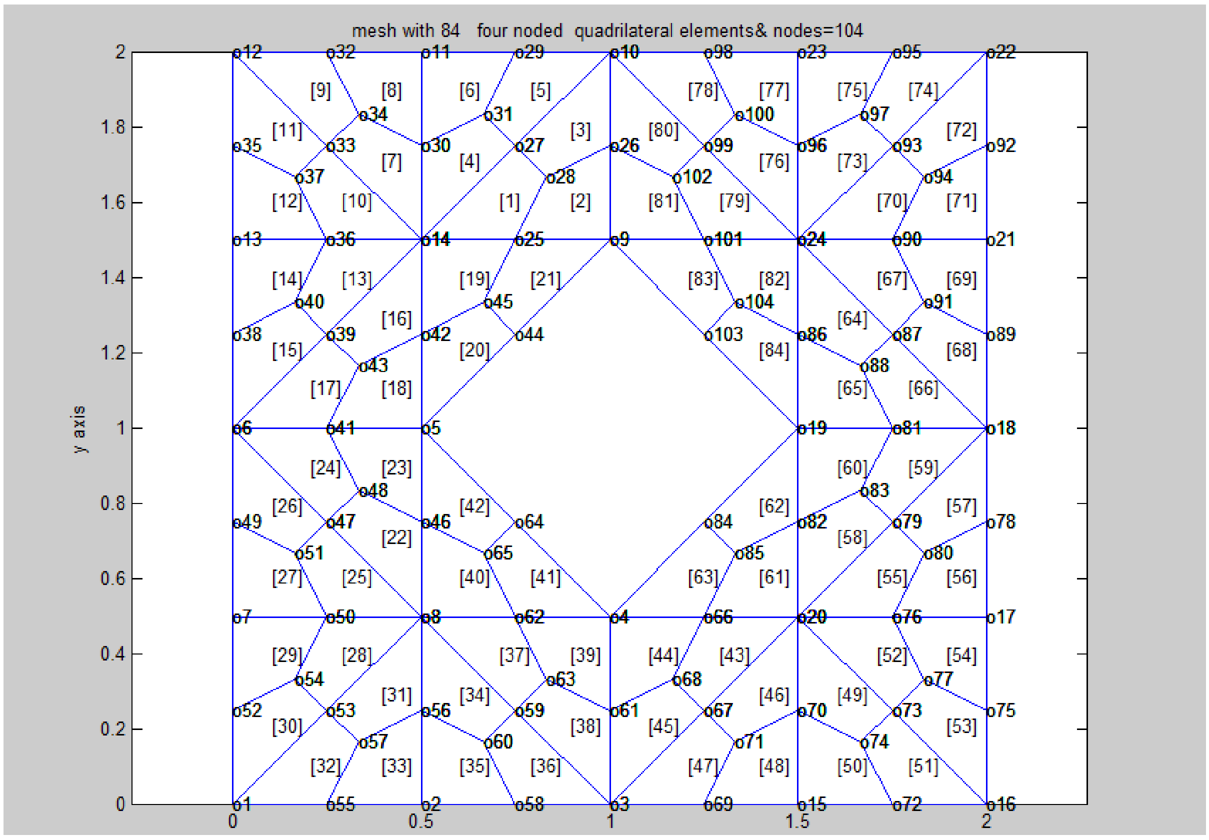


Fig. INITIAL FEM MESH FOR A SQUARE DOMAIN WITH A RHOMBUS HOLE INSIDE

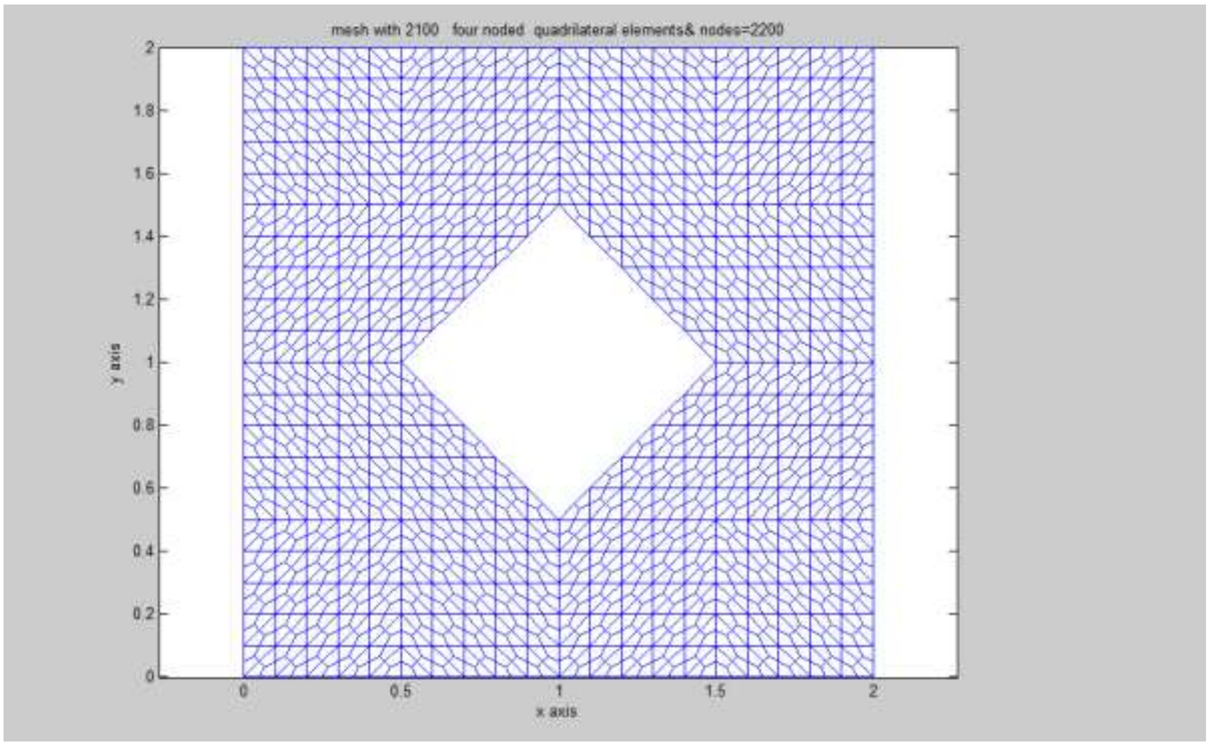

mesh with 2100 four noded quadrilateral elements& nodes=2200
y axis
x axis

Fig. FINE FEM MESH FOR A SQUARE DOMAIN WITH A RHOMBUS HOLE INSIDE

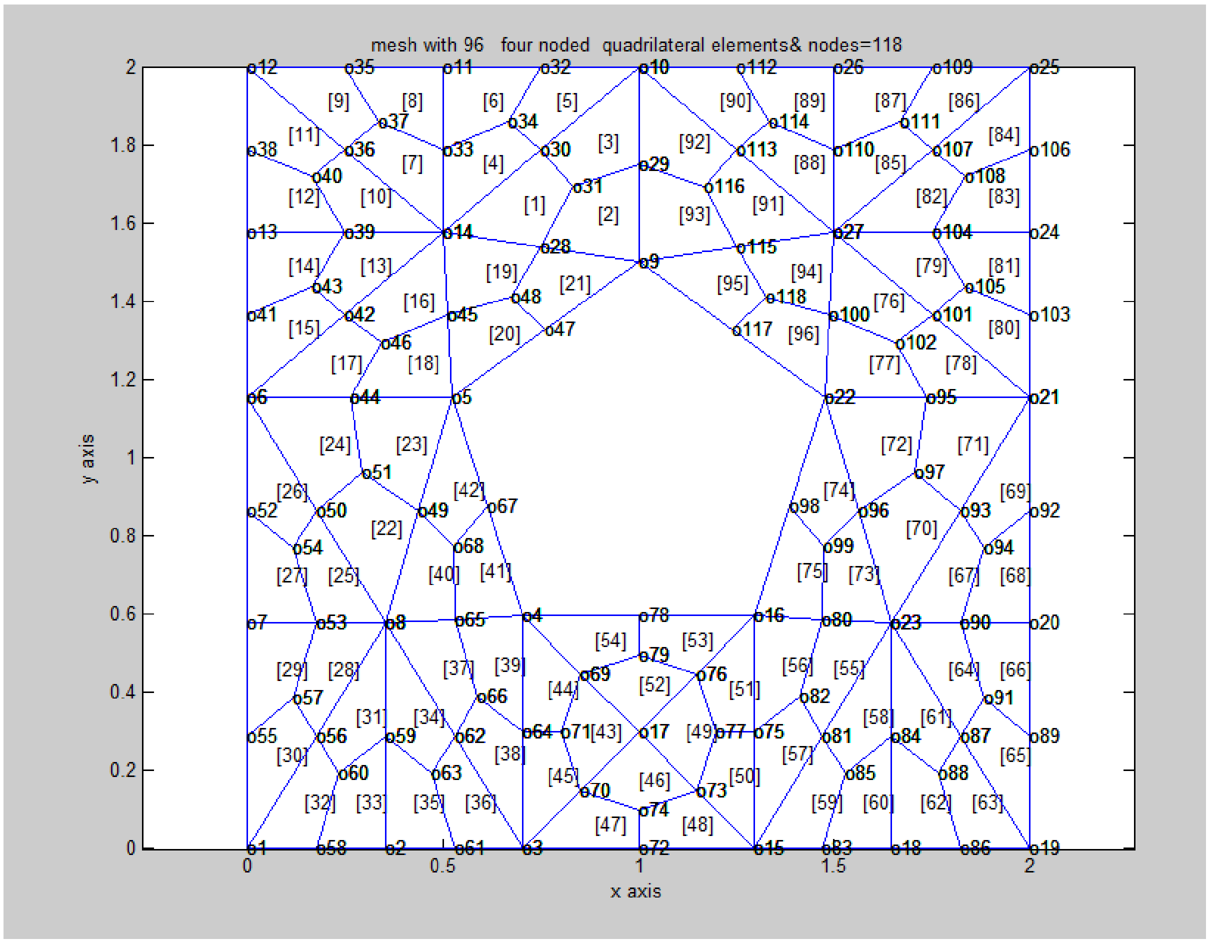


Fig. INITIAL FEM MESH FOR A SQUARE DOMAIN WITH A PENTAGON HOLE INSIDE

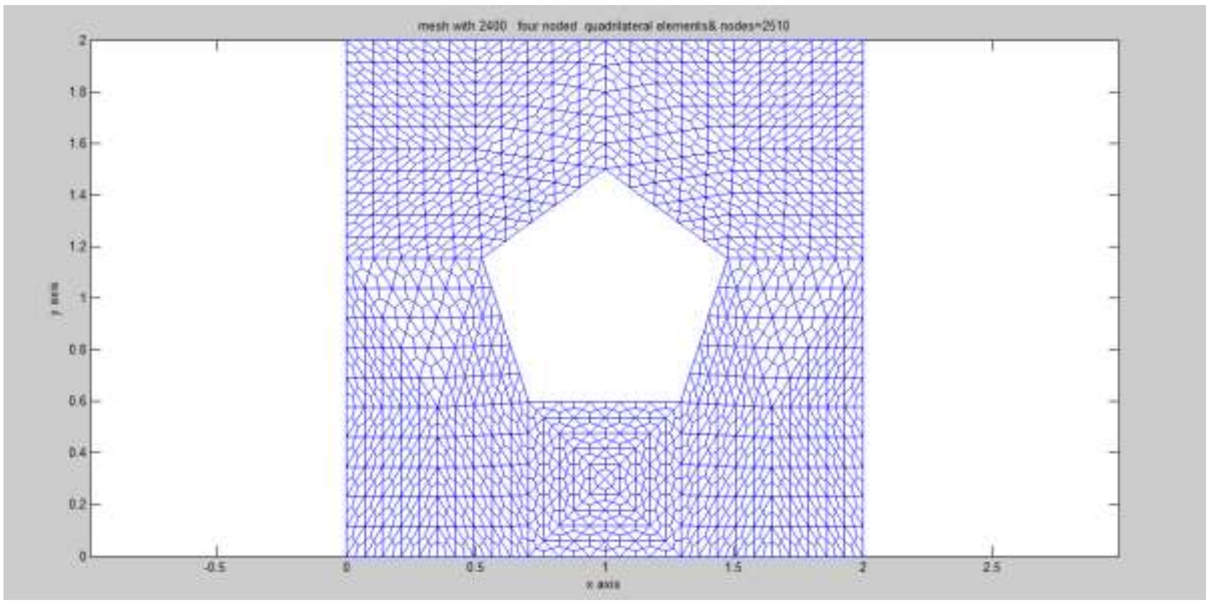

mesh with 2400 four noded quadrilateral elements& nodes=2510
x axis
y axis

Fig. FINE FEM MESH FOR A SQUARE DOMAIN WITH A PENTAGON HOLE INSIDE

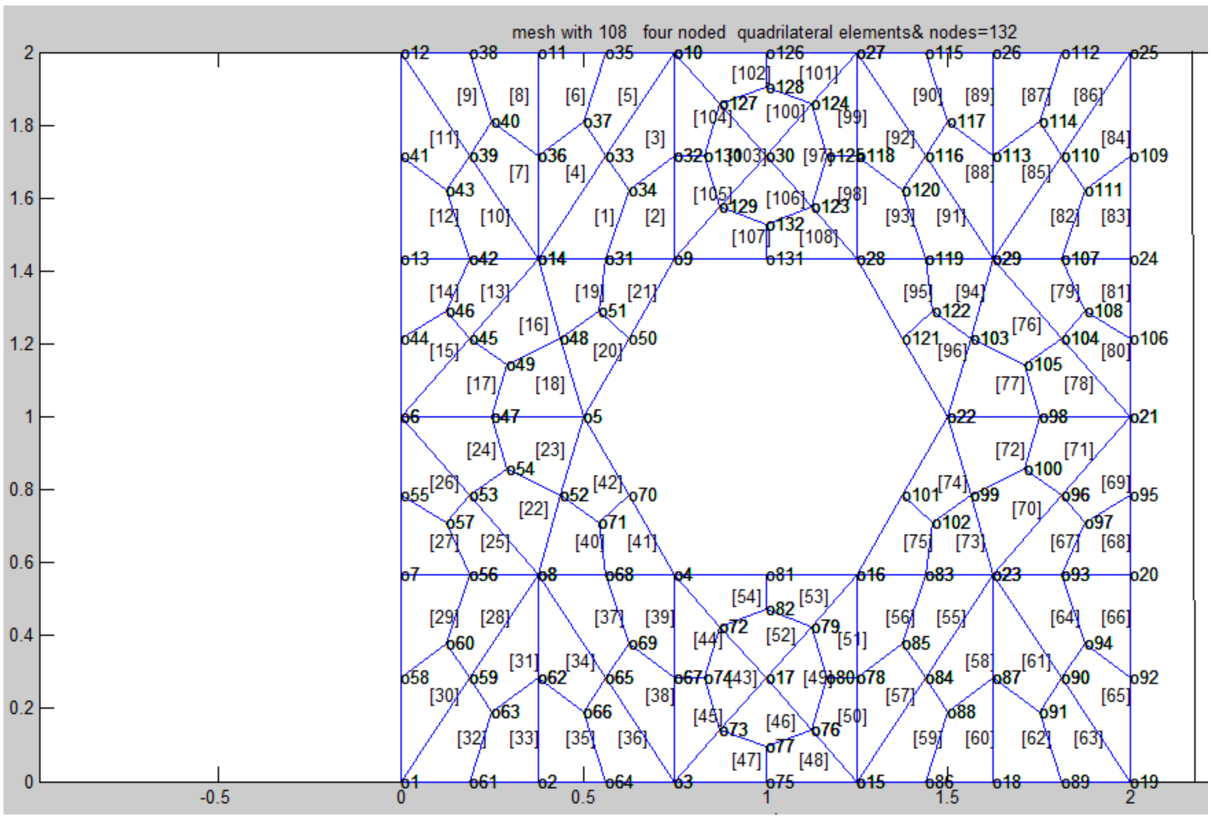
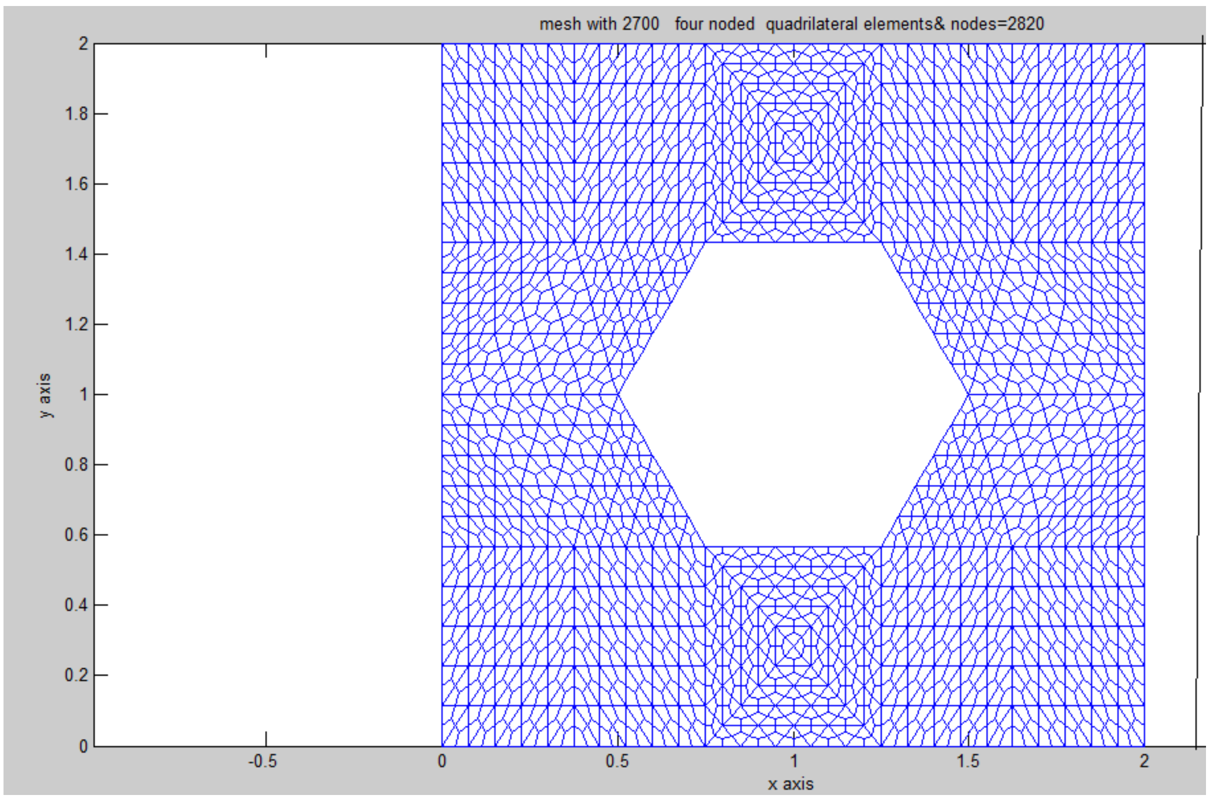


Fig. INITIAL FEM MESH FOR A SQUARE DOMAIN WITH A HEXAGON HOLE INSIDE



mesh with 2700 four noded quadrilateral elements& nodes=2820
y axis
x axis

Fig. FINE FEM MESH FOR A SQUARE DOMAIN WITH A HEXAGON HOLE INSIDE

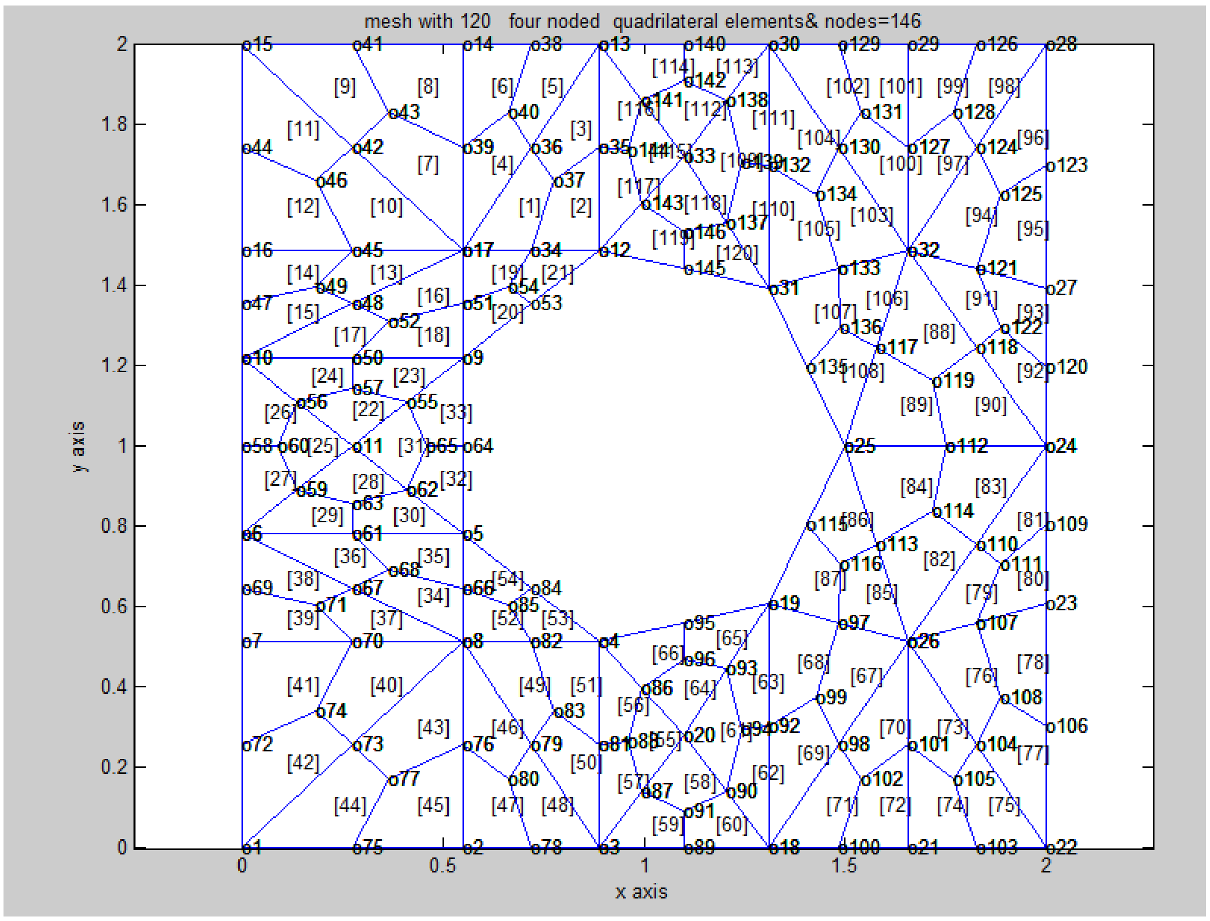


Fig. INITIAL FEM MESH FOR A SQUARE DOMAIN WITH A HEPTAGON HOLE INSIDE

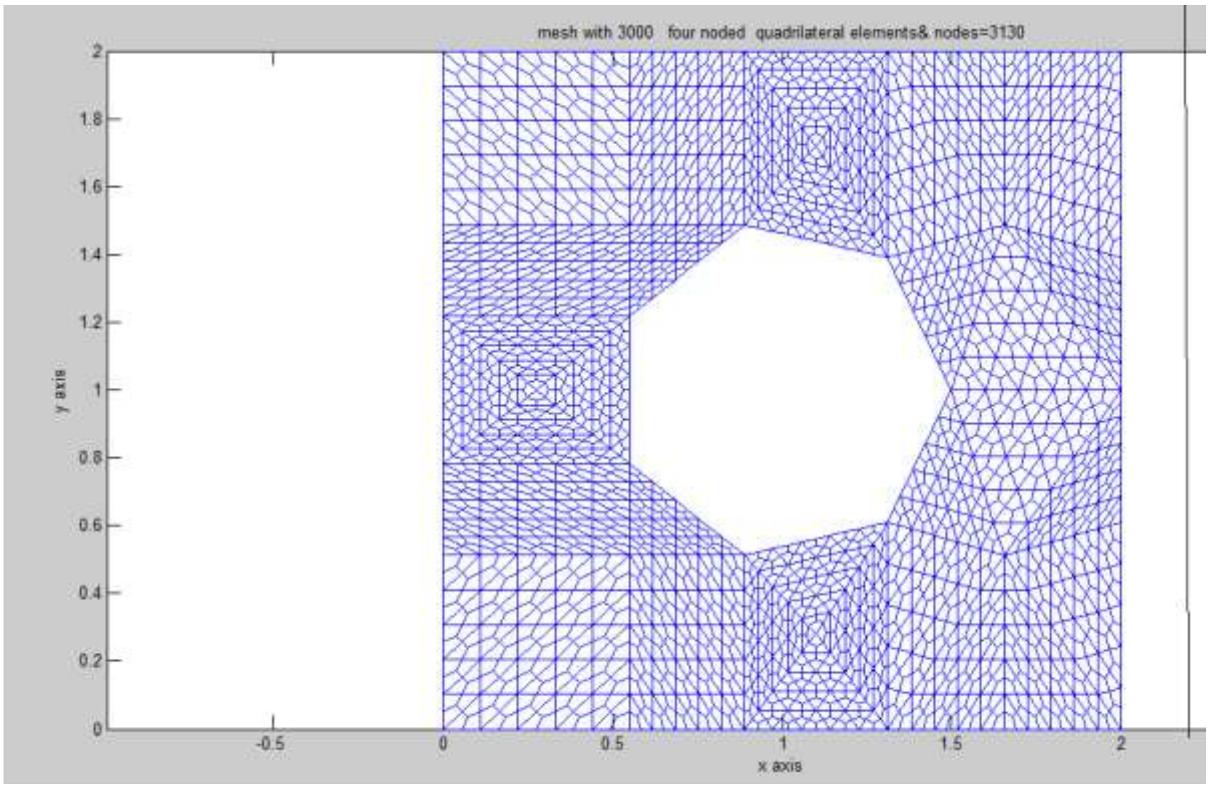

mesh with 3000 four noded quadrilateral elements& nodes=3130
x axis
y axis

Fig. FINE FEM MESH FOR A SQUARE DOMAIN WITH A HEPTAGON HOLE INSIDE

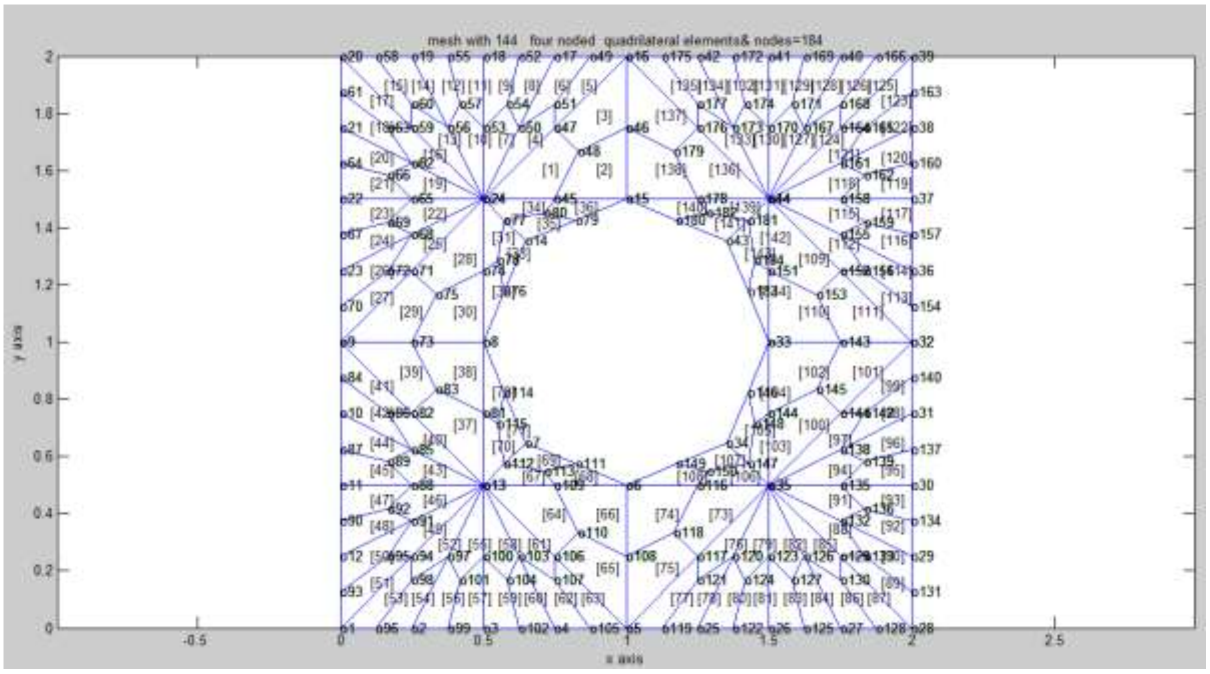


Fig. INITIAL FEM MESH FOR A SQUARE DOMAIN WITH A OCTAGON HOLE INSIDE

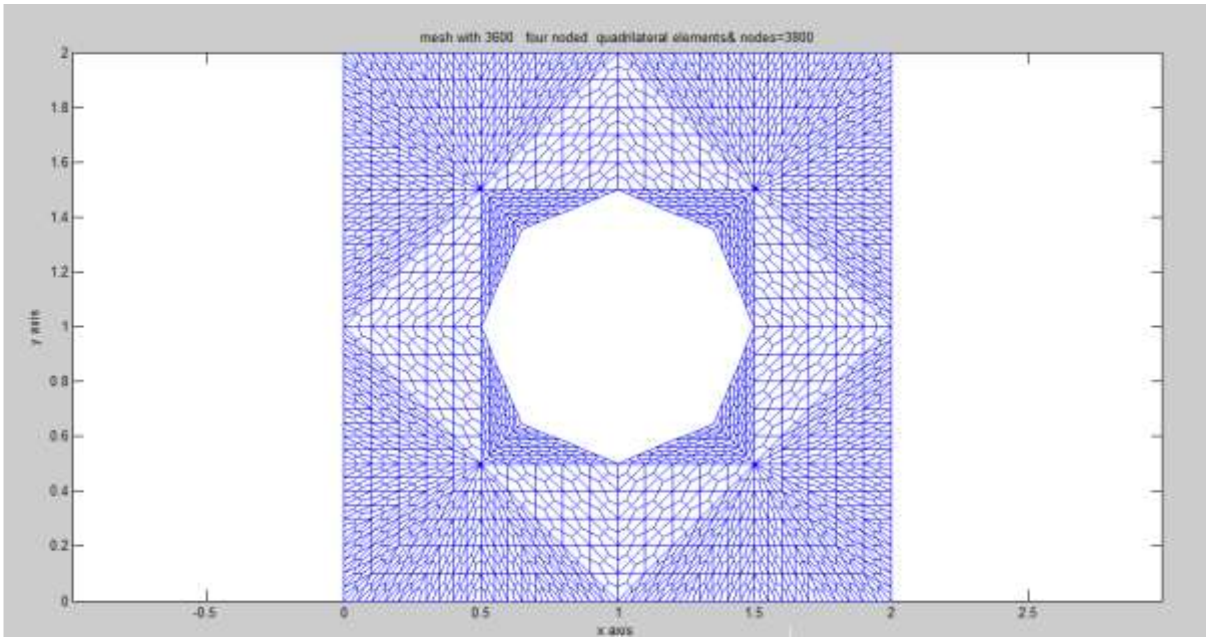

mesh with 3600 four noded quadrilateral elements& nodes=3800
x axis
y axis

Fig. FINE FEM MESH FOR A SQUARE DOMAIN WITH A OCTAGON HOLE INSIDE

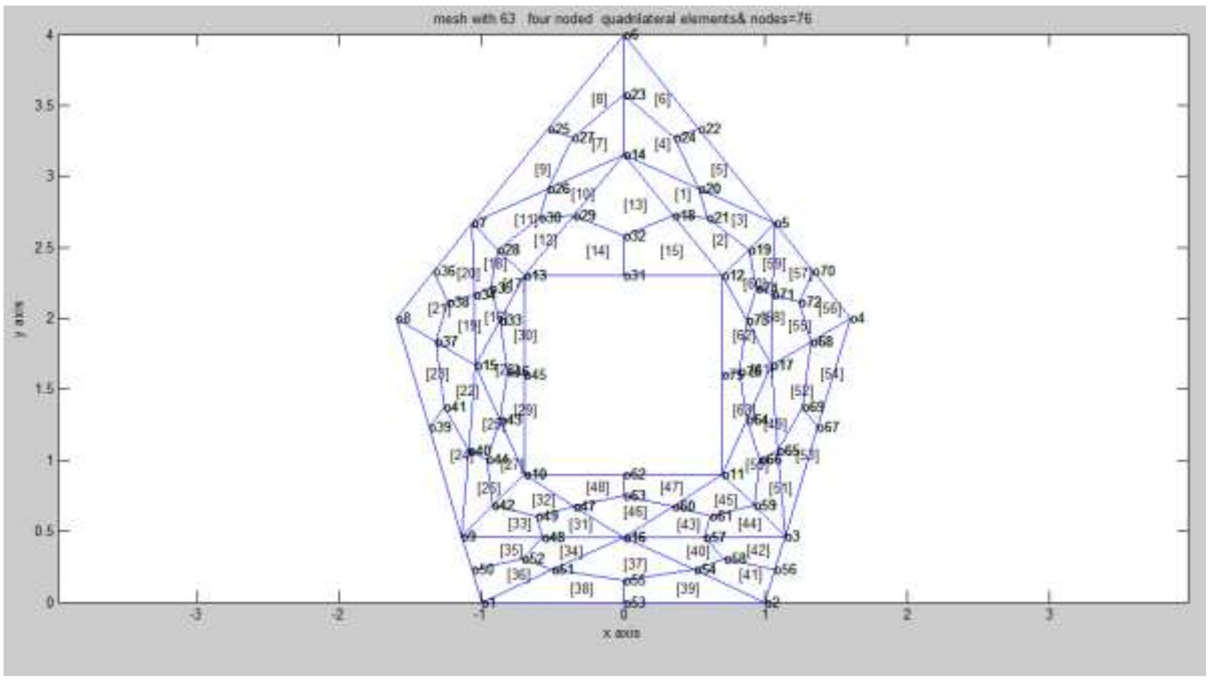


Fig. INITIAL FEM MESH FOR A PENTAGON DOMAIN WITH A SQUARE HOLE INSIDE

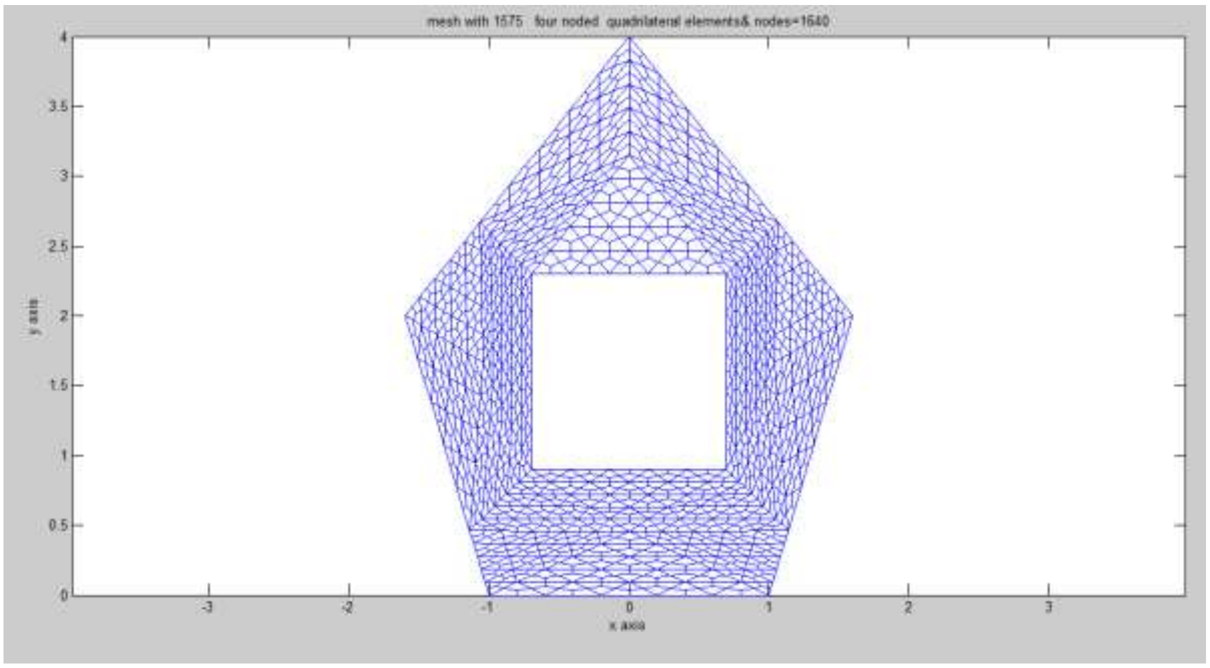

mesh with 1575 four noded quadrilateral elements& nodes=1640
x axis
y axis

Fig. FINE FEM MESH FOR A PENTAGON DOMAIN WITH A SQUARE HOLE INSIDE

8.3 Examples on multiply connected FEM **Meshes**: trapeziums emanating from a common point

We apply the algorithm of section 6 to generate square domain with a square hole inside and a circular annulus.This algorithm joins several trapeziums emanating from a common point. We have used the concept of section 6 in this process. We present an initial mesh and then the refinements of the same mesh. One can easily find the input data like nodal connections and cartesian coordinates of the nodes from this initial fem mesh for the domain.

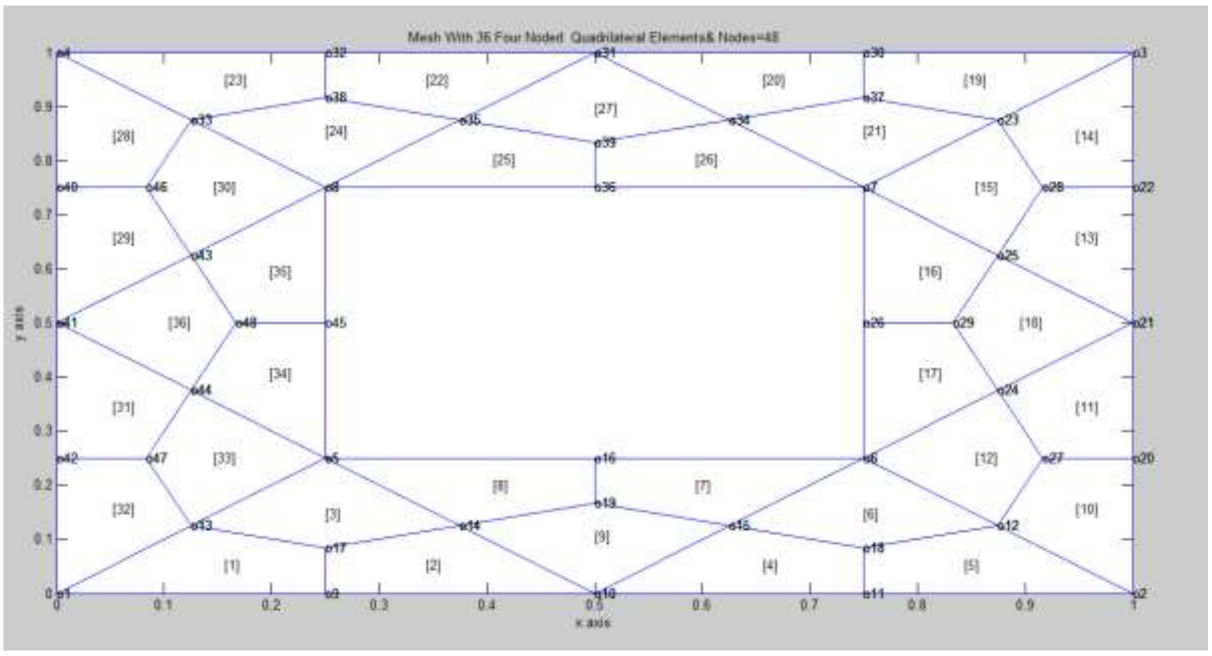


FIG. INITIAL FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-1

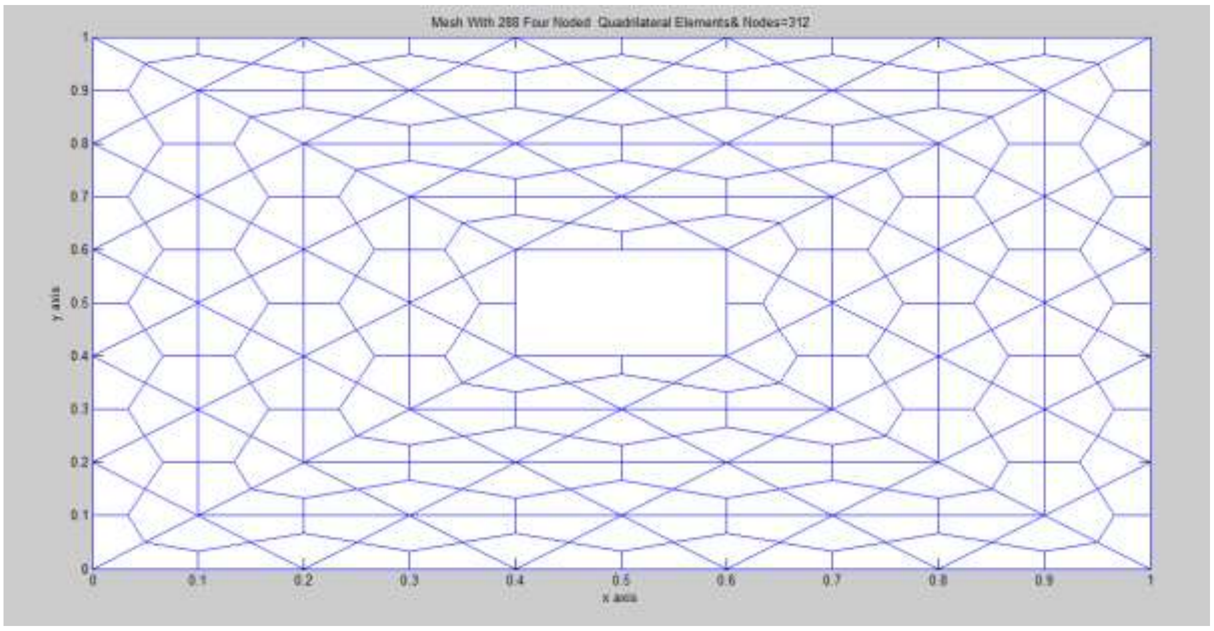


FIG. FINE FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-1

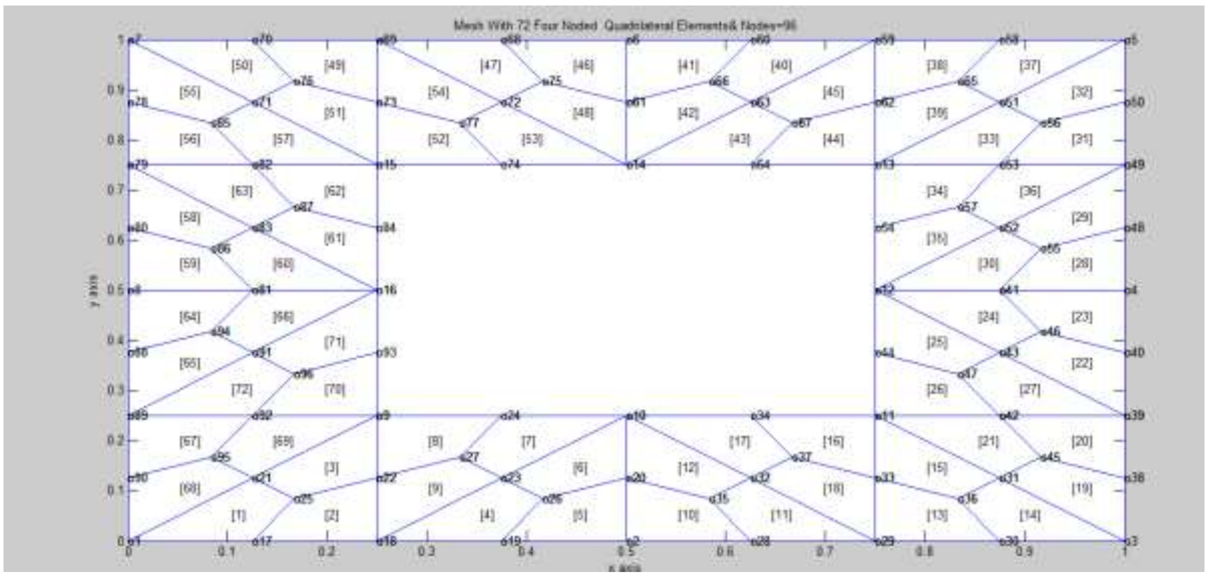


FIG. INITIAL FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-2

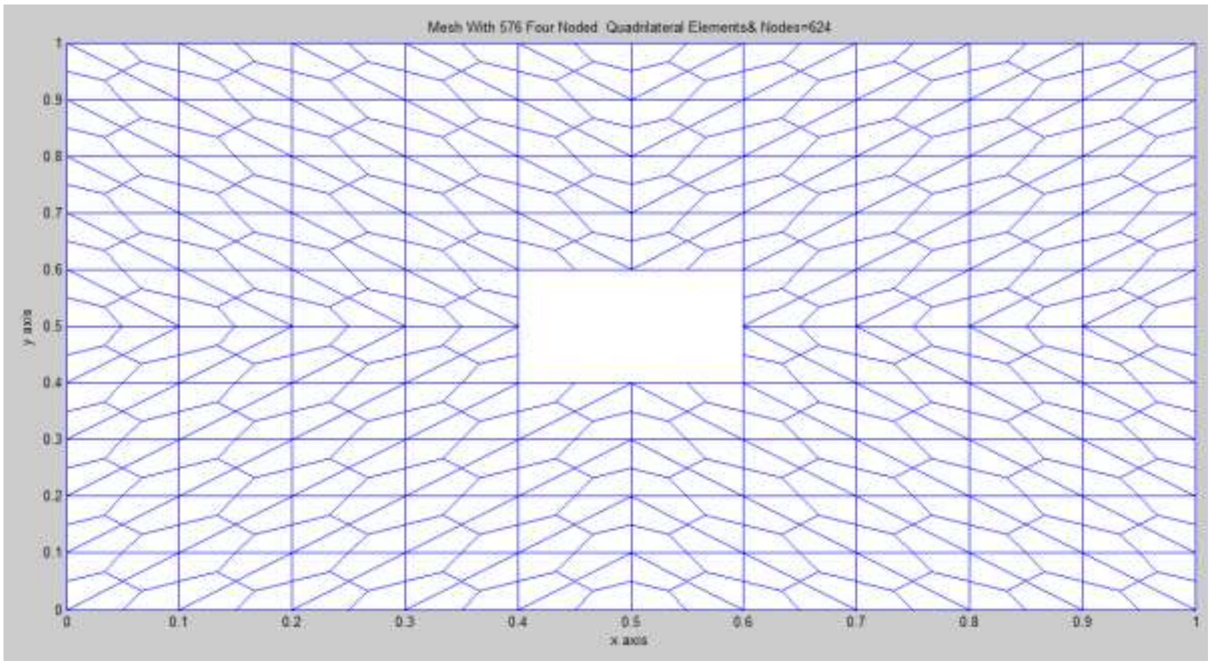


FIG. FINE FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-2

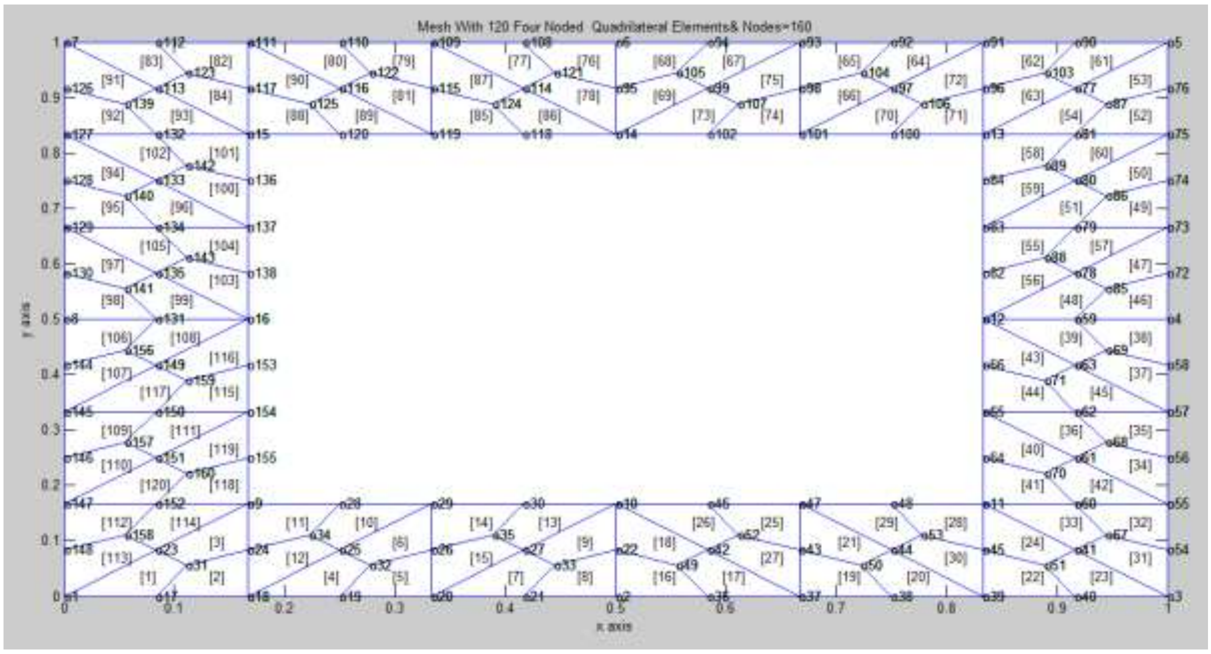


FIG. INITIAL FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-3

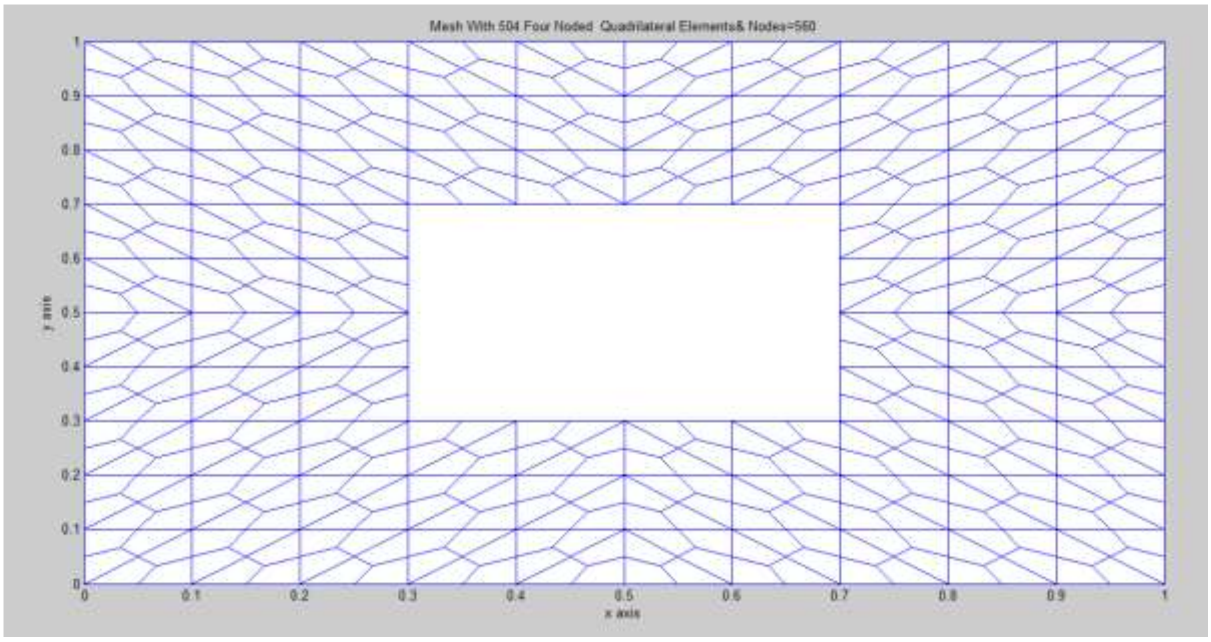


FIG. FINE FEM MESH OF SQUARE DOMAIN WITH A SQUARE HOLE INSIDE-3

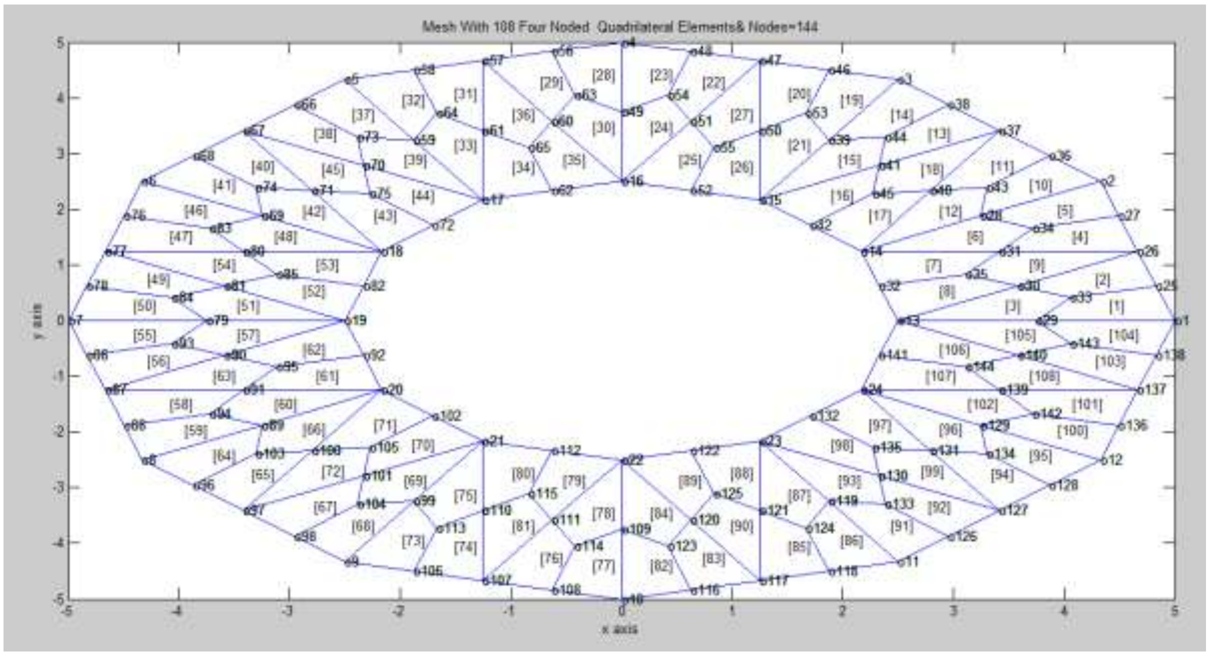


FIG INITIAL FEM MESH FOR CIRCULAR ANNULUS WITH A LINEAR BOUNDARY -1

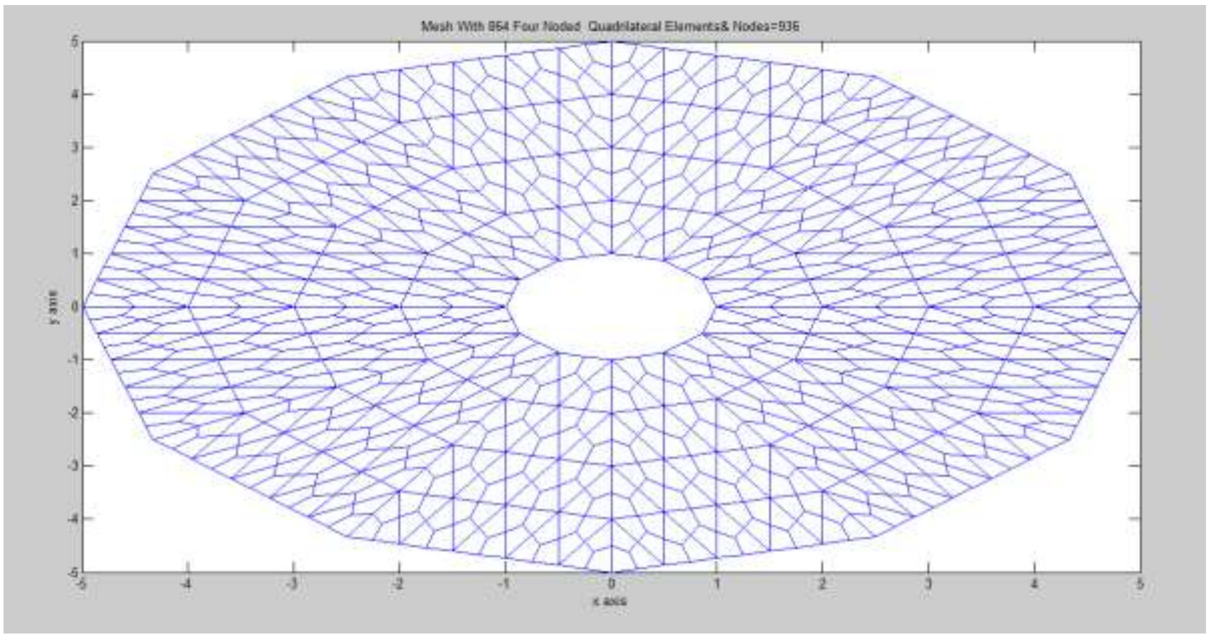


FIG FINE FEM MESH FOR CIRCULAR ANNULUS WITH A LINEAR BOUNDARY -1

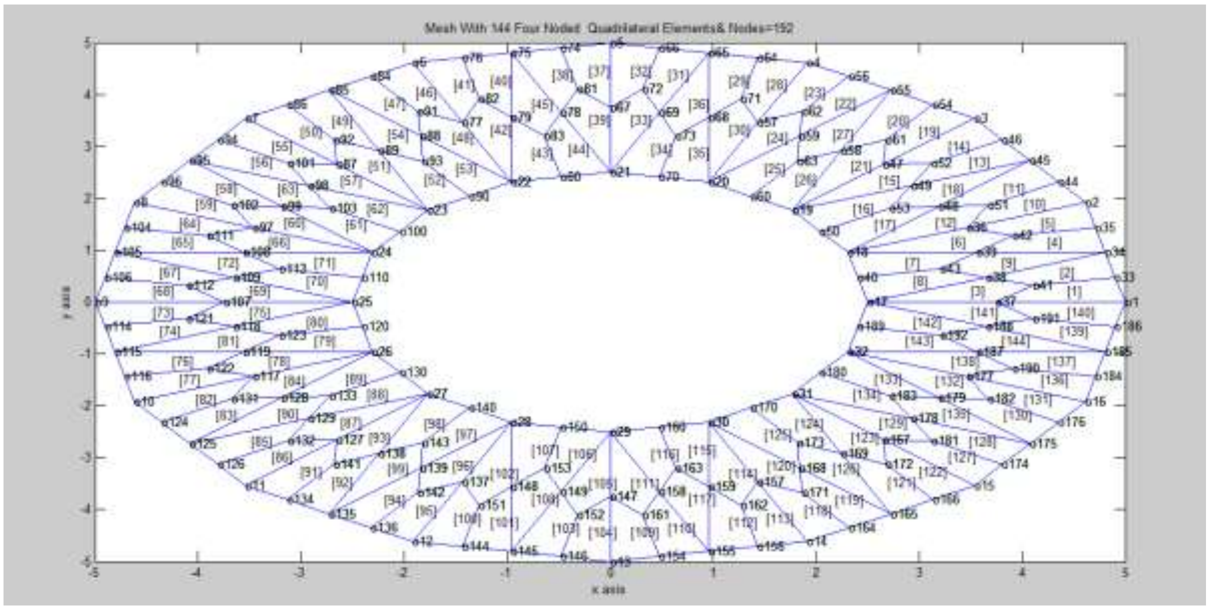
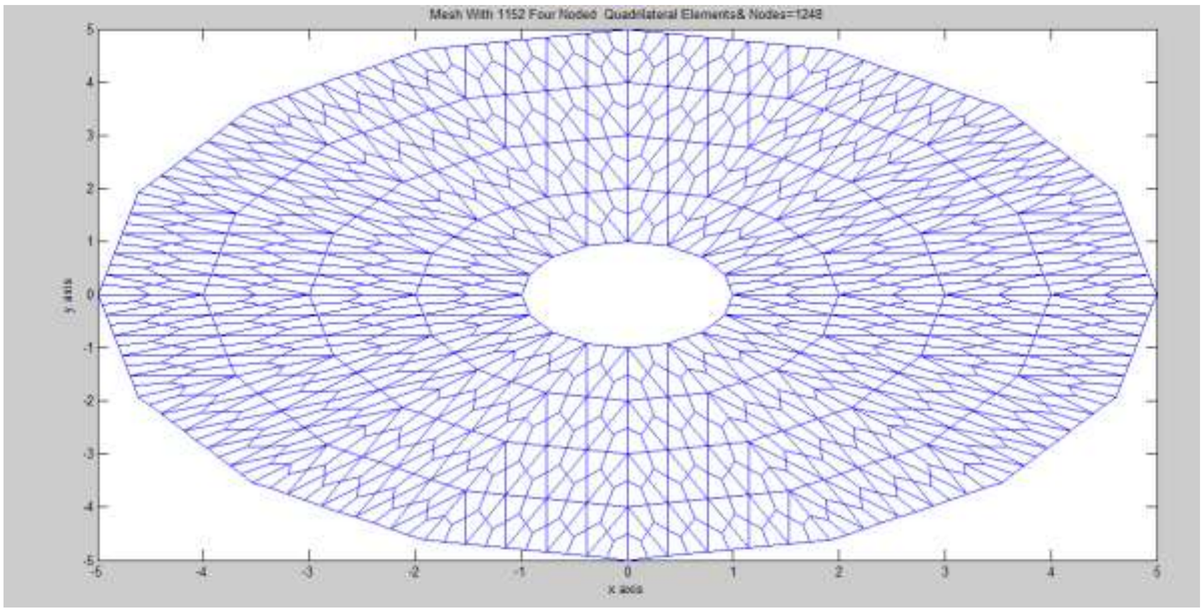


FIG INITIAL MESH FOR CIRCULAR ANNULUS WITH A LINEAR BOUNDARY-2



Mesh With 1152 Four Noded Quadrilateral Elements& Nodes=1248
x axis
y axis

FIG FINE MESH FOR CIRCULAR ANNULUS WITH A LINEAR BOUNDARY- 2

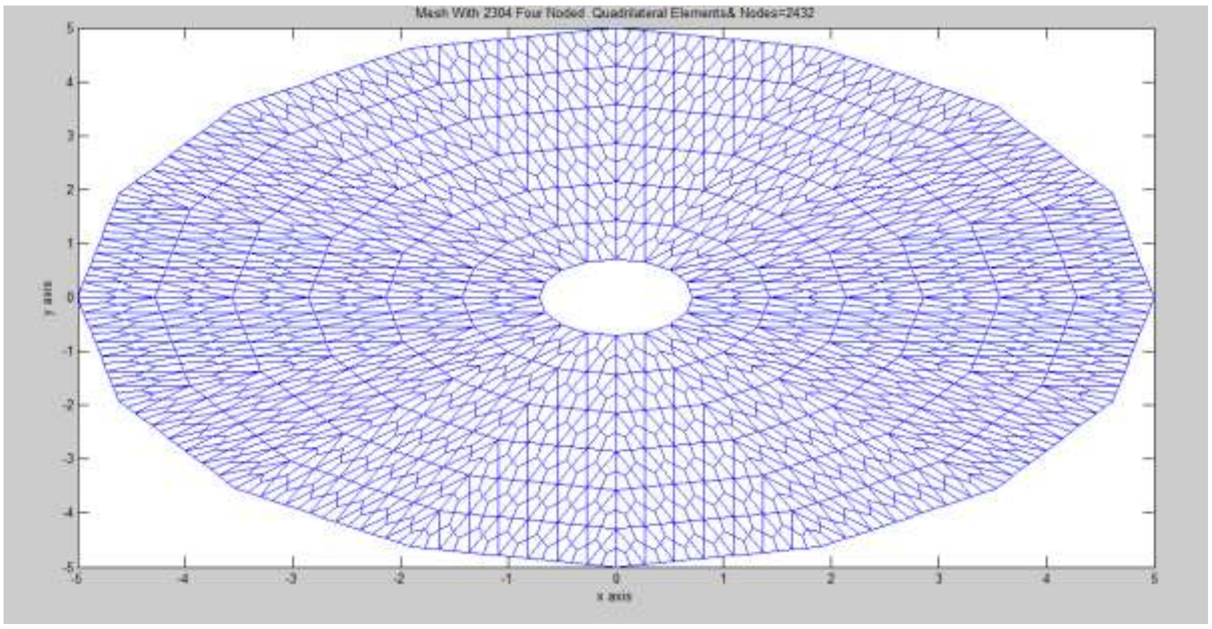

Mesh With 2304 Four Noded Quadrilateral Elements& Nodes=2432
x axis
y axis
-5
-4
-3
-2
-1
0
1
2
3
4
5

FIG FINE MESH FOR CIRCULAR ANNULUS WITH A LINEAR BOUNDARY-2

9.CONCLUSIONS

An automatic indirect quadrilateral mesh generator which uses the splitting technique is presented. This paper describes a scheme for finite element mesh generation of a convex polygon,cracked polygon, non-convex polygon and multiply connected linear polygon domains **such as polygons with holes** and layered domains**:**such as **circular annulus.** 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 and element nodal connectivity in counter clockwise orientation for the coarse discretisation. Once this input is created, by selecting appropriate number of interior points of the linear polygonal domain, we form the triangular subdomains. These subdomains are then triangulated to generate a fine mesh of six node triangular elements. 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 triangles 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 linear polygonal domain into all quadrilaterals, 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 finite element mesh for a square duct, a convex, a non-convex and cracked convex polygons. We believe that this work will be useful for various applications in science and engineering.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[25] Rathod. H.T. , Rathod. Bharath , Shivaram. K. T. and Sugantha Devi. K. A new approach to automatic generation of all quadrilateral mesh for finite element analysis,International Journal of Engineering and Computer Science,vol.2,issue12,pp 3488-3530(2013)

[26]. 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

APPENDIX:COMPUTER PROGRAMS

PROGRAM-1

```
function[]=quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,k
max,nmax,numtri,ndiv,m,mesh,xlength,ylength)

global edgen1n3 edgen2n4 edgen2n1 edgen3n4 kmax itri nmax
global XCOORD YCOORD NODES


[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_QUADcoordinatestrapezium(n1,n2,
n3,N1,N2,N3,N4,nmax,numtri,ndiv,m,mesh)
[nel,nnel]=size(nodes);
disp([xlength,ylength,nnode,nel,nnel])
NODES(1:nel,1:nnel,itri)=nodes;
%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
for i=1:nel
for j=1:nnel
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
axis tight
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
```

```
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 5
axis([0 xlength 0 ylength])

case 6
axis([0 xlength 0 ylength])
case 7
axis([0 xlength 0 ylength])


case 8
    axis([0 xlength 0 ylength])
case 9
    axis([0 xlength 0 ylength])

 case 10
    axis([0 xlength 0 ylength])
case 11

  axis([0 xlength 0 ylength])
   case 12

  axis([0 xlength 0 ylength])


  case 13

  axis([0 xlength 0 ylength])
  case 14

  axis([0 xlength 0 ylength])
  case 15

  axis([0 xlength 0 ylength])
  case 16

  axis([0 xlength 0 ylength])
  case 17

  axis([0 xlength 0 ylength])
  case 18

  axis([0 xlength 0 ylength])
   case 19

  axis([-xlength xlength -ylength ylength])
   case 20

  axis([-xlength xlength -ylength ylength])

   case 21

  axis([-xlength xlength -ylength ylength])
   case 22

  axis([-xlength xlength -ylength ylength])

   case 23
  axis([-xlength xlength -ylength ylength])

   case 24

  axis([-xlength xlength -ylength ylength])

   case 25

  axis([-xlength xlength -ylength ylength])
   case 26

  axis([-xlength xlength -ylength ylength])

   case 27

  axis([-xlength xlength -ylength ylength])
   case 28

  axis([-xlength xlength -ylength ylength])
   case 29

  axis([-xlength xlength -ylength ylength])
   case 30

  axis([-xlength xlength -ylength ylength])




end
 %
plot(xvec,yvec);%plot element
hold on;
%place element number
if (ndiv<=8)
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
if itri==1
text(midx,midy,['[',num2str(i),']']);
end
if (itri>1)
text(midx,midy,['[',num2str(i+(itri-1)*nel),']']);
end
end% if ndiv
end;%i loop

xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(itri*nel);
st3=' Four Noded  ';
st4='Quadrilateral';
st5=' Elements'
st6='& Nodes='
st7=num2str(nnode);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if (ndiv<=8)
for jj=1:nnode
%if (jj~=1)& (jj~=2)& (jj~=3)& (jj~=4)
%text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj+4)]);
%
for i=1:nel
```

```
  for j=1:nnel
    if(jj==nodes(i,j))
     text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
    end
  end
end
%
end
end
hold on
gcoord
for i=1:nel
    kode=0;
    for j=1:nnel

        if(nodes(i,j)~=0)
            kode=kode+1
        end
        if(kode>=1)
            XCOORD(nodes(i,j),1)=xcoord(nodes(i,j),1);
            YCOORD(nodes(i,j),1)=ycoord(nodes(i,j),1);
        end
    end
end

NODES
```

PROGRAM-2

```
function[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_QUADcoordinatestrapeziu
m(n1,n2,n3,N1,N2,N3,N4,nmax,numtri,n,m,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
%kmax=the number of nodes for the trapezium
%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



global edgen1n3 edgen2n4 edgen2n1 edgen3n4 kmax itri nmax


syms U V W x y

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


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

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

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

x=sym([0;1;1/2])
y=sym([0;0;sqrt(3)/2])
case 8%for a convexpolygonsixside
  %    1    2    3  4    5   6   7   8
x=sym([0.5;0.1;0.7;1;.75;.5;.25;0])
y=sym([0.5;0;0.2;.5;.85;1;.625;.25])
case 9
%    1  2   3  4    5    6  7     8   9
x=([.2;0.5;.8;1.0;.75;0.5;0.25;0.0;0.5])
y=([.2;.05;.2;0.5;.85;1.0;0.90;0.6;0.5])
case 10
%    1  2   3  4    5    6  7     8   9
x=([0.5;.2;0.5;.8;1.0;.75;0.5;0.25;0.0])
y=([0.5;.2;.05;.2;0.5;.85;1.0;0.90;0.6])
case 11%lunar model
    x=[0.5;1.0;0.5;0.0;0.5]
    y=[0.5;0.5;1.0;0.5;0.0]
case 12
%      1   2         3       4       5         6     7       8         9
   x=double([0.5;1.0;0.5+.3536;0.5;0.5-.3536;0.0;.3536;0.5;0.5+.3536])
   y=double([0.5;0.5;0.5+.3536;1.0;0.5+.3536;0.5;.3536;0.0;0.5-.3536])
 case 13
   %       1  2  3
   x=sym([0;1;0.5])
   y=sym([0;0;0.5])

 case 14
       %  1  2  3 4  5
    x=sym([0.5;0;1;1;0])
    y=sym([0.5;0;0;1;1])

 case 15
%x=sym([0;1;1/2])
%y=sym([0;0;1/2])
x=sym([0;1;1;0;1/2])
y=sym([0;0;1;1;1/2])
itri=1
kmax=8
 case 16
x=sym([0;1;1;0;1/2])
y=sym([0;0;1;1;1/2])
%x=sym([1;1;1/2])
%y=sym([0;1;1/2])
itri=2
case 17
%x=sym([1;0;1/2])
%y=sym([1;1;1/2])
x=sym([0;1;1;0;1/2])
y=sym([0;0;1;1;1/2])
itri=3
```

```
case 18

%x=sym([0;0;1/2])
%y=sym([1;0;1/2])

x=sym([0;1;1;0;1/2])
y=sym([0;0;1;1;1/2])
itri=4
% input for circular annulas%12-sectors
case 19
ro=sym(5);
x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=1
kmax=24
nmax=13
case 20

    ro=sym(5);
    x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end


itri=2

case 21

 ro=sym(5);
  x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=3
case 22
   ro=sym(5);
    x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end


itri=4
case 23
 ro=sym(5);
  x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
```

```
itri=5

case 24
 ro=sym(5);
  x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=6

case 25
ro=sym(5);
 x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=7
 case 26
ro=sym(5);
 x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=8


case 27
 ro=sym(5);
  x=zeros(13,1);y=zeros(13,1);
for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=9
case 28
    ro=sym(5);
     x=zeros(13,1);y=zeros(13,1);
 for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=10
case 29
    ro=sym(5);
     x=zeros(13,1);y=zeros(13,1);
 for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
```

```
itri=11
case 30
    ro=sym(5);
     x=zeros(13,1);y=zeros(13,1);
  for i=1:12
    ti=sym((i-1)*pi/6);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end

itri=12
% input for circular annulas%12-sectors
 case 31
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=1
kmax=32
nmax=17
case 32
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=2
case 33
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=3

case 34
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=4
case 35
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=5

case 36
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
```

```
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=6
case 37
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=7
case 38
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=8

case 39
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=9

case 40
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=10

case 41
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=11

case 42
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=12

case 43
```

```
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=13

case 44
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=14

case 45
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=15
case 46
ro=sym(5);
     x=zeros(17,1);y=zeros(17,1);
  for i=1:16
    ti=sym((i-1)*pi/8);
    x(i,1)=sym(ro*cos(ti));
    y(i,1)=sym(ro*sin(ti));
end
itri=16

    case 51
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=1
        kmax=16
    case 52
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=2

     case 53
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=3
    case 54
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=4
    case 55
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=5
    case 56
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=6

     case 57
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=7

    case 58
        %      1   2   3   4 5  6  7  8   9
        x=sym([0;0.5;1.0;1.0;1;0.5;0;0.0;0.5]);
        y=sym([0;0.0;0.0;0.5;1;1.0;1;0.5;0.5]);
        itri=8








end
x
y

[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals4atrapezium(N1
,N2,N3,N4,n,m,kmax,itri,2);

        disp('vertices of trapezium=')
        [rrr(1,1,itri) rrr(1,n+1,itri) rrr(n-m+2,1,itri) rrr(n-m+2,m,itri)]
 [U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
ss1='number of 6-node triangles with centroid=';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
%
eln
%
ss2='number of special convex quadrilaterals elements&nodes per element =';
[nel,nnel]=size(spqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
spqd
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])


%for itri=1:1
%itri=1
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
nn1=n1(itri,1)
nn2=n2(itri,1)
nn3=n3(itri,1)
x1=x(nn1,1)
x2=x(nn2,1)
x3=x(nn3,1)
%
y1=y(nn1,1)
```

```
y2=y(nn2,1)
y3=y(nn3,1)
rrr(:,:,itri)
U'
V'
W'
kk=0;
for ii=1:n-m+2
    for jj=1:(n+1)-(ii-1)
        kk=kk+1;
        mm=rrr(ii,jj,itri);
        mmm(kk,1)=mm;
        uu=U(kk,1);vv=V(kk,1);ww=W(kk,1);
        xi(mm,1)=x1*ww+x2*uu+x3*vv;
        yi(mm,1)=y1*ww+y2*uu+y3*vv;
    end
end
[xi yi]
[(1:kk)'  mmm]
rrr(:,:,itri)
%add coordinates of centroid
 %ne=(n/2)^2;
% stdnode=kk;
ne=p1
for iii=1:ne
%for iii=1:ne
     %kk=kk+1;
     node1=eln(iii,1)
     node2=eln(iii,2)
     node3=eln(iii,3)
     mm=eln(iii,7)
     xi(mm,1)=(xi(node1,1)+xi(node2,1)+xi(node3,1))/3;
     yi(mm,1)=(yi(node1,1)+yi(node2,1)+yi(node3,1))/3;
 end

%end
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)
 rrr
   %edgen2n3
```

PROGRAM-3

```
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals4atrap
ezium(N1,N2,N3,N4,n,m,kmax,itri,type)
global edgen1n3 edgen2n4 edgen2n1 edgen3n4 kmax itri nmax
 %
nnd=(n+1)*(n+2)/2-m*(m-1)/2+kmax-4
nitri=nmax-1
if type==1
    m=0
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;
%nodes along edge:n2-n3
edgen2n3(1:n+1,1)=zeros(n+1,1);
for i=1:(n+1)
        edgen2n3(i,1)=row_nodes(i,(n+1)-i+1);
end



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

nnd=(n+1)*(n+2)/2;
for kkk=1:ne
    nnd=nnd+1;
    eln(kkk,7)=nnd;
end

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




end%type==1




if type==2
```

```
    for ij=1:(n+1)*(n+2)/2-m*(m-1)/2
        elm(ij,1)=0;
    end
elm
if kmax==4
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2-m*(m+1)/2+1,1)=3
%disp('vertex nodes of triangle')
%base edge
%kk=3;
kk=4;
for k=2:n
kk=kk+1;
elm(k,1)=kk;
end

%disp('left edge nodes')
nni=1;
for i=0:(n-m-1)
nni=nni+(n-i)+1;
%elm(nni,1)=3*n-m-i;
elm(nni,1)=n+3+n-m+1+i;
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-m)

nni=nni+(n-i);
%elm(nni,1)=(n+3)+i;
if i<(n-m)
elm(nni,1)=(n+4)+i;
end
if i==(n-m)
elm(nni,1)=4
end

end
 edgen2n4(1,1)=2;
%disp('interior nodes')
nni=1;jj=0;
for i=0:(n-m)
nni=nni+(n-i)+1;
for j=1:(n-2-i)
jj=jj+1;
nnj=nni+j;
%elm(nnj,1)=3*n-m+jj;
elm(nnj,1)=n+3+n-m+n-m+jj;
end
end
end%kmax==4
%change vertices of trapezium
if kmax>4
%n1=1;n2=2;n3=5;n4=6;
elm(1,1)=N1(itri,1);
elm(n+1,1)=N2(itri,1);
elm((n+1)*(n+2)/2-m*(m+1)/2+1,1)=N3(itri,1);
%elm((n+1)*(n+2)/2-m*(m-1)/2,1)=n4;

kk=kmax;
```

```
for k=2:n
kk=kk+1;
elm(k,1)=kk;
end

%disp('right edge nodes')
nni=n+1;
for i=0:(n-m)

nni=nni+(n-i);
%elm(nni,1)=(n+3)+i;
if i<(n-m)
elm(nni,1)=(kmax+n)+i;
end
if i==(n-m)
elm(nni,1)=N4(itri,1);
end
end
%disp('left edge nodes')
nni=1;
for i=0:(n-m-1)
nni=nni+(n-i)+1;
%elm(nni,1)=3*n-m-i;
elm(nni,1)=kmax+n+n-m+i;
end
%disp('interior nodes')
if (itri==1)
nni=1;jj=0;
for i=0:(n-m)
nni=nni+(n-i)+1;
for j=1:(n-2-i)
jj=jj+1;
nnj=nni+j;
%elm(nnj,1)=3*n-m+jj;
elm(nnj,1)=kmax+n+n-m+n-m-1+jj;
end
end
end%if

%disp('changed interior nodes')
%-----
if ((itri>1))
nni=1;jj=0;
for i=0:(n-m)
nni=nni+(n-i)+1;
for j=1:(n-2-i)
jj=jj+1;
nnj=nni+j;
%elm(nnj,1)=3*n-m+jj;
if (itri<nitri)
    elm(nnj,1)=kmax+n+n-m-1+jj;
end
if (itri==nitri)
 elm(nnj,1)=kmax+n-1+jj;
end
end
end
end%if
%----------

%if ((itri==nitri))
%nni=1;jj=0;
%for i=0:(n-m)
%nni=nni+(n-i)+1;
%for j=1:(n-2-i)
```

```
%jj=jj+1;
%nnj=nni+j;
%elm(nnj,1)=3*n-m+jj;
%elm(nnj,1)=kmax+n-1+jj;
%end
%end
%end%if








end%kmax>4
disp(elm)
disp(length(elm))

jj=0;kk=0;
for j=0:n-m+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)=0;

%change vertices of trapezium

[row_nodes]
rr=row_nodes;


rr
rrr(:,:,itri)=rr;


%nodes along edge:n2-n4
edgen2n4(1:n+2-m,itri)=zeros(n+2-m,1);
for i=1:(n+2-m)
        edgen2n4(i,itri)=row_nodes(i,(n+1)-i+1);
end



edgen2n4(1:n+2-m,itri)
n+2-m
%rr
%disp('element computations')
%nodes along edge:n1-n3
edgen1n3(1:n+2-m,itri)=zeros(n+2-m,1);
%edgen1n3(1,1)=1;
for i=1:(n+2-m)
    edgen1n3(i,itri)=row_nodes(i,1);
end
disp('itri=')
disp(itri)
disp('edgen1n3=')
disp(edgen1n3(1:n+2-m,itri))
disp('edgen2n4=')
disp(edgen2n4(1:n+2-m,itri))
disp('length of edge vectors=')
(n+2-m)
%now assign the common edge to the adjacent elements
```

```
%---------------------------------------------------------------------------

if (itri>1)

%for ii=1:n+1
 % for jj=2:n+1
  %  if (rr(ii,jj)~=0)
   %  rr(ii,jj)=kmax-4+rr(ii,jj);
   % end
  %end%for
%end%for
for i=1:(n+2-m)
    rr(i,1)=edgen2n4(i,itri-1);
end
%rr(1,1)=N1(itri,1);rr(1,n+1)=N2(itri,1);
%rr(n-m+2,1)=N3(itri,1);rr(n-m+2,m)=N4(itri,1);
end%if
if itri==nitri
    for i=1:(n+2-m)
    rr(i,(n+1)-i+1)=edgen1n3(i,1);
    end
end

 rr(1,1)=N1(itri,1);rr(1,n+1)=N2(itri,1);
rr(n-m+2,1)=N3(itri,1);rr(n-m+2,m)=N4(itri,1);

%

rrr(:,:,itri)=rr;
%---------------------------------------------------------------------------
if rem(n,2)==0
ne=0;N=n+1;
for k=1:2:n-m+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
ne
for kk=1:ne
[eln(kk,1:6)]
end

%--------------------------------------------
```

```
%add node numbers for element centroids
if (itri==1)
nnd=nnd;
end
if (itri>1)
    if (itri<nitri)
    nnd=nnd-(n-m);
    end
    if (itri==nitri)
    nnd=nnd-(n-m)-(n-m);
    end

end
%-------------------------------------------------
NND=nnd
for kkk=1:ne
NND=NND+1;
eln(kkk,7)=NND;
end

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

ne
end%type==2

%------------------------------------------------------
```

PROGRAM-4

```
function[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n)
syms ui vi wi U V W
kk=0;
for j=1:n+1
    for i=1:(n+1)-(j-1)
        kk=kk+1;
        ui(i,j)=(i-1)/n;
        vi(i,j)=(j-1)/n;
        wi(i,j)=1-ui(i,j)-vi(i,j);
        U(kk,1)=ui(i,j);
        V(kk,1)=vi(i,j);
        W(kk,1)=wi(i,j);
    end
end
```

```
%  ui
 % vi
  %wi
  U'
  V'
  W'
 kk
```

PROGRAM-5

```
function[]=trapezium_mesh(N)
global XCOORD YCOORD NODES
%FEMmeshExample4quadrilateralpolarSpquadrilateral(N)%N=1,2,3.........8,9,10
n1=[1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16]
n2=[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;1]
n3=[17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17]
N1=[1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16]
N2=[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;1]
N3=[17;18;19;20;21;22;23;24;25;26;27;28;29;30;31;32]
N4=[18;19;20;21;22;23;24;25;26;27;28;29;30;31;32;17]
%trapezium_mesh(1)
 clf
    figure(N)
switch N

case 1
for mesh= 31:46
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,32,17,4,4,3,
mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 2
for mesh= 31:46
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,32,17,9,6,3,
mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 3
for mesh= 31:46
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,32,17,16,8,3
,mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause

end
case 4
for mesh= 31:46
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,32,17,25,10,
3,mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end
case 5
for mesh= 31:46
```

```
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,32,17,49,14,
3,mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end

end
```

PROGRAM-6

```
function[]=trapezium_mesh0(N)
global XCOORD YCOORD NODES
n1=[1;2;3;4;5;6;7;8;9;10;11;12]
n2=[2;3;4;5;6;7;8;9;10;11;12;1]
n3=[13;13;13;13;13;13;13;13;13;13;13;13]
N1=[1;2;3;4;5;6;7;8;9;10;11;12]
N2=[2;3;4;5;6;7;8;9;10;11;12;1]
N3=[13;14;15;16;17;18;19;20;21;22;23;24]
N4=[14;15;16;17;18;19;20;21;22;23;24;13]
 clf
 figure(N)
switch N

case 1
for mesh= 19:30
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,24,13,4,4,3,
mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end
case 2
for mesh= 19:30
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,24,13,9,6,3,
mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end
case 3
for mesh= 19:30
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,24,13,16,8,3
,mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end
case 4
for mesh= 19:30
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,24,13,25,10,
3,mesh,5,5)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
end

end
```

PROGRAM-7

```
function[]=square_mesh(N)
disp('FEM MESH FOR A SQUARE DOMAIN WITH A RECTANGULAR HOLE INSIDE')
global XCOORD YCOORD NODES
```

```
%square_mesh(1)
n1=[1;2;3;4]
n2=[2;3;4;1]
n3=[5;5;5;5]
N1=[1;2;3;4]
N2=[2;3;4;1]
N3=[5;6;7;8]
N4=[6;7;8;5]
 clf
 figure(N)

switch N

case 1
for mesh= 15:18

quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,8,5,4,4,3,me
sh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 2
for mesh= 15:18
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,8,5,9,6,3,me
sh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 3
for mesh= 15:18
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,8,5,16,8,3,m
esh,1,1)

NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 4
for mesh= 15:18
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,8,5,25,10,3,
mesh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end

end
```

PROGRAM-8

```
 function[]=square_mesh1(N)
disp('FEM MESH FOR A SQUARE DOMAIN WITH A RECTANGULAR HOLE INSIDE')
global XCOORD YCOORD NODES
%square_mesh(1)
n1=[1;2;3;4;5;6;7;8]
n2=[2;3;4;5;6;7;8;1]
```

```
n3=[9;9;9;9;9;9;9;9]
N1=[1;2;3;4;5;6;7;8]
N2=[2;3;4;5;6;7;8;1]
N3=[9;10;11;12;13;14;15;16]
N4=[10;11;12;13;14;15;16;9]
 clf
    figure(N)
switch N

case 1
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,4,4,3,m
esh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 2
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,9,6,3,m
esh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end

case 3
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,16,8,3,
mesh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end

case 4
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,25,10,3
,mesh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 5
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,9,6,5,m
esh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
case 6
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,16,8,5,
mesh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
pause
end
```

```
case 7
for mesh= 51:58
quadrilateral_mesh4convexpolygoneightsidesq4trapezium(n1,n2,n3,N1,N2,N3,N4,16,9,25,10,5
,mesh,1,1)
NODES
NNNNNN=(1:length(XCOORD))';
[NNNNNN XCOORD YCOORD]
Pause
 end
end
```