www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issues 7 July 2016, Page No. 17172-17238

A New Approach To Automatic Generation Of All Quadrilateral Meshes Over A Linear Convex Polygon With H-Refinements For Finite Element Analysis

H.T. Rathod[a*], K. Sugantha Devi[b]

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

[b] 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

Abstract

This paper presents an automatic mesh generation scheme for a linear convex polygonal domain P_N in $\Re^2$ with boundary composed of piecewise straight lines.We can express $P_N = \sum_{i=1}^{N} t_i = \sum_{i=1}^{N}\sum_{n=1}^{m^2} T_n{}^i = \sum_{i=1}^{N}(\sum_{n=1}^{m^2}\sum_{a=0}^{2} Q^i{}_{3n-a}) = \sum_{i=1}^{N}\sum_{n=1}^{m^2}\sum_{a=0}^{2}\sum_{b=1}^{4} Q^i{}_{3n-a,b}$

$=\sum_{i=1}^{N}\sum_{n=1}^{m^2}\sum_{a=0}^{2}\sum_{b=1}^{4}\sum_{c=1}^{4} Q^i{}_{3n-a,b,c}$

in which P_N is a polygonal domain of *N* oriented edges l_{ik} $(k = i+1,\ i = 1,2,3,\ldots,\ N)$, with end points (x_i, y_i), (x_k, y_k) and $(x_1, y_1) = (x_{N+1}, y_{N+1})$. Where,it is assumed that P_N can be discretised into a set of N triangles,$\{t_i\ , i = 1,2,3,\ldots..N\}$ and we can also divide each triangle t_i into m^2smaller triangles $T^i{}_n$ so that $t_i = \sum_{n=1}^{m^2} T^i{}_n$ which gives us a triangular mesh with h-refinement ,when this procedure is repeated for all triangles we obtain a triangular mesh with h-refinement for a linear convex polygon, further we can discretise each triangle $T^i{}_n$ into three special quadrilaterals $Q^i{}_{3n-a}$, a=0,1,2 which are obtained by joining the centroid to the midpoint of its sides and we have created an all quadrilateral mesh. We choose $T_n{}^i = T_{pqr}{}^{xy}$ an arbitrary triangle with vertices $((x_\alpha, y_\alpha),\ \alpha = p,q,r)$ in Cartesian space (x, y). We have furthered refined this mesh two times. We have taken this further and created 12-new quadrilaterals by joining the centroids of the 3-special quadrilaterals to the midpoints which can be called as first h-refinement.Then each of these quadrilateral is divided into four smaller quadrilaterals again by joining the centroids to the mid points of sides which we call as the second h- refinement.

The applications of all quadrilateral mesh and its first and second refinements to numerical integration is considered in our recent works [16,17,18].We know that integration can be accomplished by using linearity property of integrals.Hence finite element mesh generation for the complete linear convex polygon is not essential. However,for application in finite element analysis,element nodal connectivity and information on nodal coordinates is required for the complete domain.This paper gives algorithms and codes for first and second h-refinements of a linear convex polygon into an all quadrilateral mesh.

DOI: 10.18535/ijecs/v5i7.10

We have appended MATLAB programs which incorporate the mesh generation scheme developed in this paper. These programs provide valuable output on the nodal coordinates ,element connectivity and graphic display of the all quadrilateral mesh using first and second h-refinements for application to finite element analysis.

Keywords: finite elements, triangulation ,quadrilateral mesh generation, convex polygonal domain, uniform refinement,h-refinements

1. Introduction

Engineers and scientists use finite element analysis (FEA) software to build predictive computational models of real-world scenarios. The use of FEA software begins with a computer-aided design (CAD) model that represents the physical parts being simulated as well as knowledge of the material properties and the applied loads and constraints. This information enables the prediction of real-world behaviour, often with very high levels of accuracy.
It is interesting to note that finite element analysis in engineering had its origins in the 1950s and 1960s. Aerospace engineering was the focal point of activity during this time. By the late 1960s the first commercial computer programs (ASKA, NASTRAN, Stardyne, etc.) appeared. Subsequently, the finite element method spread to other engineering and scientific disciplines, and now its use is widespread and many commercial programs are available.
Each element in the finite element model is addressed by its number. Also each node is addressed by its number. The inter-connectivity of the elements is determined by the common nodes shared by the elements. In a model with few elements and nodes, the user can manually divide the domain, number each element and node, and keep track of the element connectivity. However, in models with many nodes and elements, the effort required to divide the domain into elements and attend to connectivity is great. It then becomes difficult to accomplish this task without committing errors. However, there are several finite element preprocessors which do this job automatically once the geometry is defined. Users can then devote more time to interpreting results.This paper aims to present such a pre-processor for applications in Finite Element Analysis.

The accuracy that can be obtained from any FEA model is directly related to the finite element mesh that is used. The finite element mesh is used to subdivide the CAD model into smaller domains called *elements*, over which a set of equations are solved. These equations approximately represent the governing equation of interest via a set of polynomial functions defined over each element. As these elements are made smaller and smaller, as the mesh is refined, the computed solution will approach the true solution.This process of mesh refinement is a key step in validating any finite element model and gaining confidence in the software, the model, and the results.

Reducing the element size is the easiest mesh refinement strategy, with element sizes reduced throughout the modeling domains. This approach is attractive due to its simplicity, but the drawback is that there is no preferential mesh refinement in regions where a locally finer mesh may be needed.
Adaptive finite element computations rely on adjustments of the spatial resolution of the domain discretization to deliver higher accuracy where it is needed. When the domain is discretized into a finite element mesh, a possible option, albeit somewhat expensive and in some cases complex, is to create a new mesh with the desired resolution, i.e., remeshing. Another alternative is to adjust the density of the mesh by performing local refinement (resp. unrefinement) of the existing mesh so that in some regions finite elements are split to decrease their "size", in other regions they are merged to reduce the resolution. Both choices, remeshing and refinement, have their advantages and disadvantages. We are not going to argue for one or the other option. Rather, we assume that refinement had been adopted as the method of choice.
What are the desirable properties of a mesh refinement algorithm? It should certainly be efficient in that it should not become a bottleneck of the adaptive computations; it should generate good quality geometric meshes, i.e., elements must remain well-shaped on refinement and unrefinement; and, finally, it must be robust, which is usually expressed as the requirement to terminate with a valid result in finite time.
In finite element analysis, solution accuracy is judged in terms of convergence as the element "mesh" is refined. There are two major methods of mesh refinement. In the first, known as h-refinement, mesh refinement refers to the process of increasing the number of elements used to model a given domain, consequently, reducing individual element size. In the second method, p-refinement, element size is unchanged but the order of the polynomials used as interpolation functions is increased. The objective of mesh refinement in either method is to obtain sequential solutions that exhibit asymptotic convergence to values representing the exact solution.
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]. 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.

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 of 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 [11-15].

In our recent paper[16], we presented a novel mesh generation scheme of all quadrilateral elements for convex polygonal domains. This scheme converts the elements in background triangular mesh into quadrilaterals through the operation of splitting. We first decompose the convex polygon into simple subregions in the shape of triangles. These simple subregions are then triangulated to generate a fine mesh of triangles. We propose then 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 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 a all quadrilateral mesh for the given linear convex polygonal domain. In section 5.1, we present a first refinement of the above mesh generation scheme[17], which takes this concept further and creates in all 12-new quadrilaterals by joining the centroids of the 3-special quadrilaterals to the midpoints of the sides.These 12-new quadrilaterals have a unform angular variation in the four corners. In section 5.2,we present the second refinement of the original all quadrilateral mesh explained in section 4 which takes this concept further and creates in all 48-new quadrilaterals by joining the centroids of the 12-special quadrilaterals to the midpoints of the sides [18].These new 48-quadrilaterals also have a unform angular variation in the four corners.In our recent paper[16],mesh generation over a convex polygon is fully explained and composite integration over a convex polygon for complicated integrals is solved.We have also solved problems with complicated integrals using the schemes proposed as first and second h- refinements. This was made possible by using linearity property of integrals over each triangular domain .But for the solution of partial differential equations over a convex polygonal domain,we require a finite element mesh which defines the element connectivity and the nodal coordinates for the entire domain.This was fully explained in our recent work[16] but this is not done for the first and second refinement schemes proposed in [17,18]. In this paper, we have modified the mesh generation scheme in [16] to create first and second h-refinements for application to finite element analysis over linear convex polygons. We have constructed computer codes for this purpose.

DOI: 10.18535/ijecs/v5i7.10

It is necessary to emphasie that we have proposed the finite element mesh refinements for the triangular and linear convex polygonal domains which follow the concept of h-refinements. This can be further explained as:

- **(i) h-refinement of of a triangle or convex polygon by triangles**
- **(ii) h-refinement of triangles or convex polygon by splitting a triangle into three special quadrilaterals(by joining the centroid to the mid points of triangles)**
- **(iii) h-refinement of triangles or convex polygon by splitting a triangle into twelve special quadrilaterals(by joining the centroid to the mid points of three special quadrilaterals)**
- (iv) **h-refinement of triangles or convex polygon by splitting a triangle into forty eight special quadrilaterals(by joining the centroid to the mid points of twelve special quadrilaterals**)

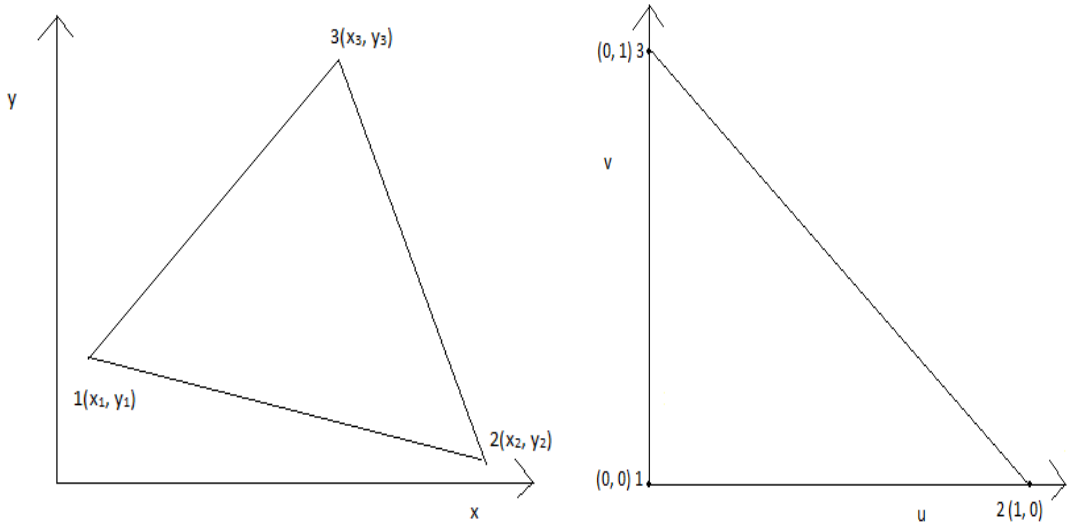
2. Division of an Arbitrary Triangle

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.

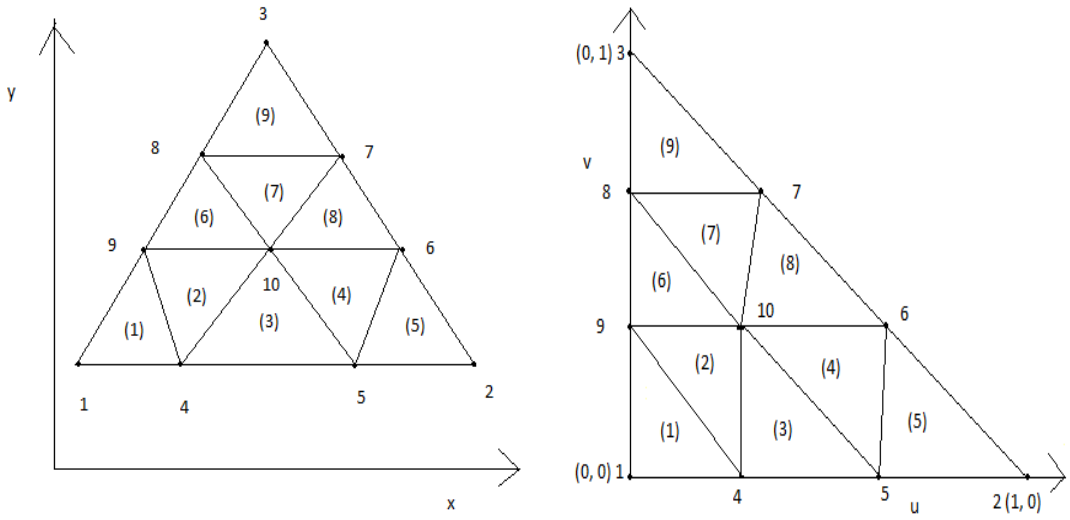


1.a 1. b

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

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

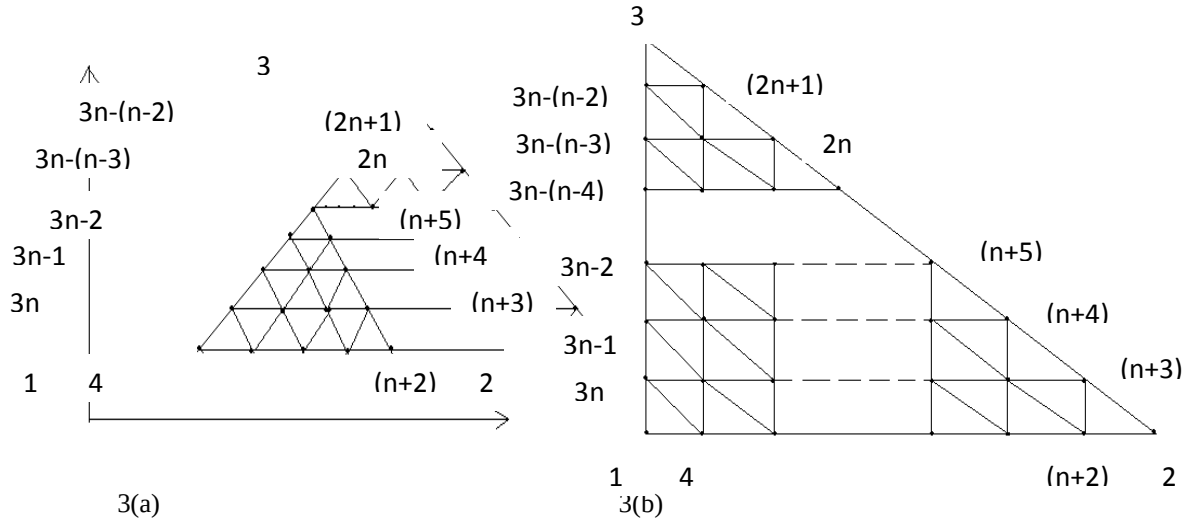
DOI: 10.18535/ijecs/v5i7.10



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



3(a) 3(b)

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

DOI: 10.18535/ijecs/v5i7.10

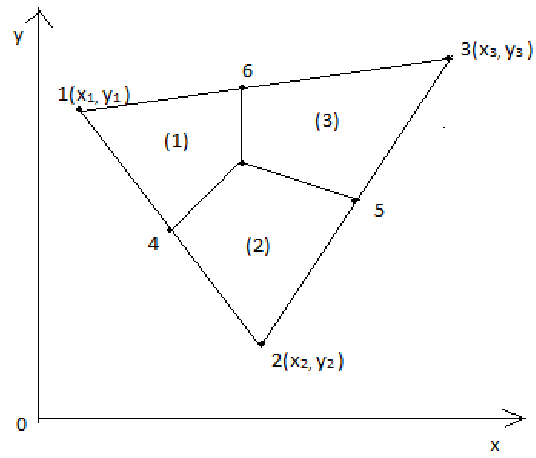
(n+1) (n+2)/2 nodes. This is shown in the $\underline{rr}$ matrix of size_$(n+1) \times (n+1)$, only nonzero entries of this matrix refer to the nodes of the triangles

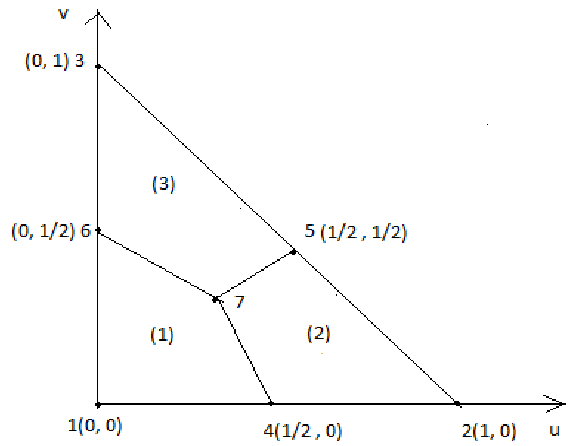
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \ldots\ldots\ldots\ldots\ldots\ldots, & (n+2) & 2 \\ 3n, & (3n+1), \ldots\ldots\ldots\ldots\ldots, & 3n+(n-2), & (n+3) & 0 \\ 3n-1, & 3n+(n-1) \ldots\ldots\ldots & ,3n+(n-2)+(n-3), & (n+4) & 0 \quad 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots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

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





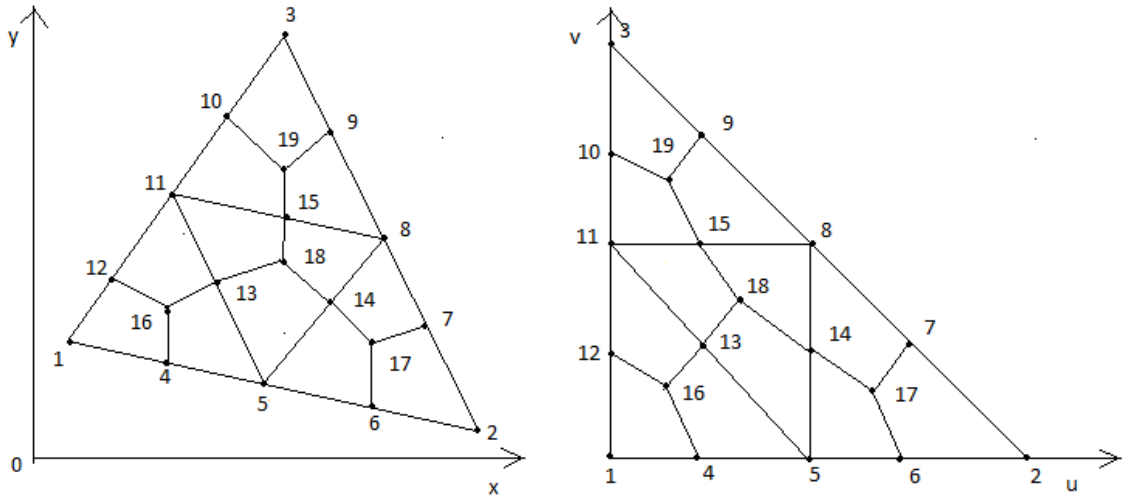
4(a) 4(b)

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

DOI: 10.18535/ijecs/v5i7.10



5(a) 5(b)

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

Fig 5b. 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 $\underline{rr}$

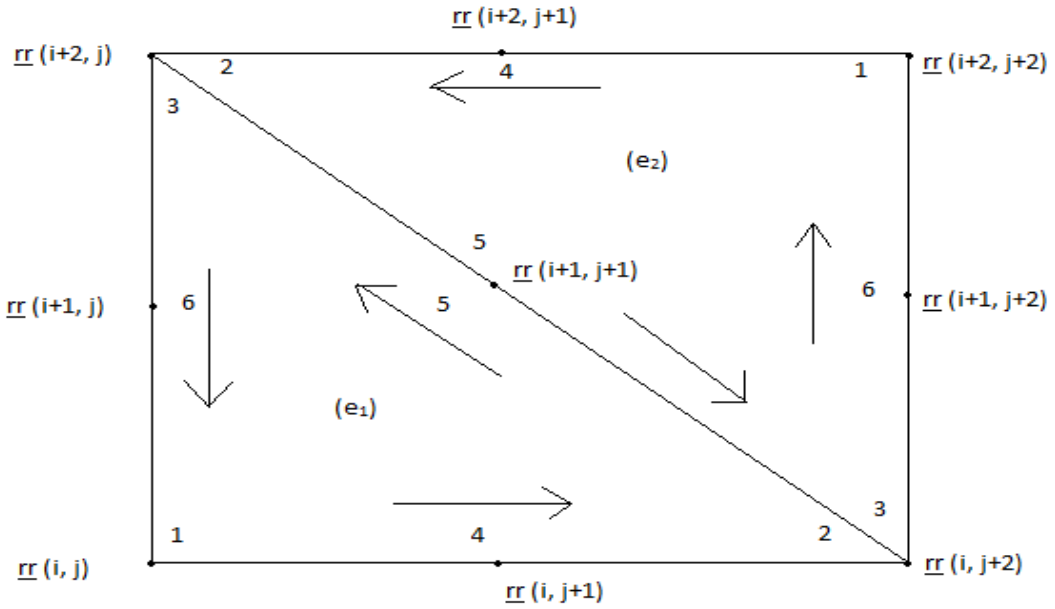


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

DOI: 10.18535/ijecs/v5i7.10

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

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

(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 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 (e_1) 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 (e_2) are given as

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

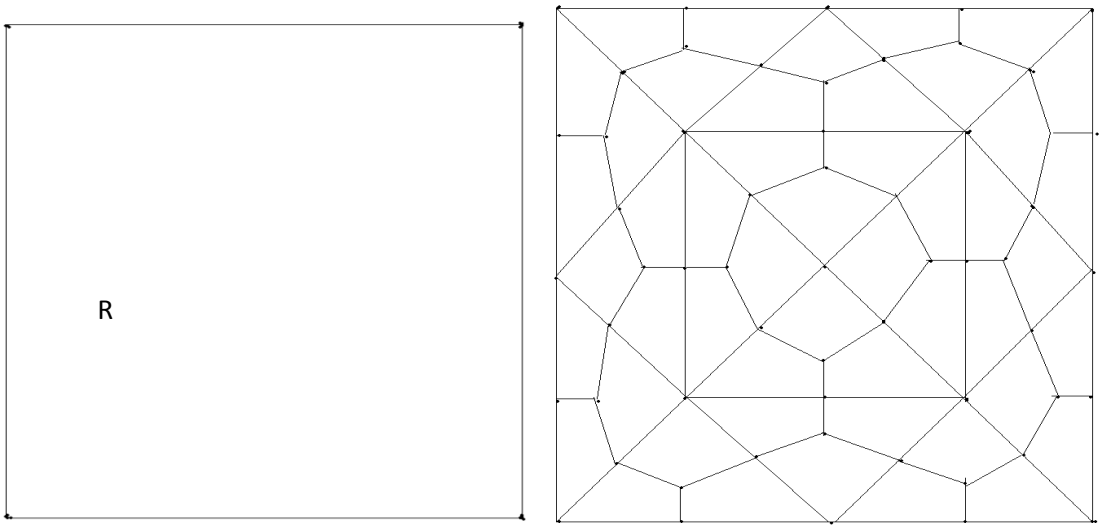
Q_{3n_2-1} < c_2 , $\underline{rr}\ (i+2,\ j+1),\ \underline{rr}\ (i+2,\ j)$, $\underline{rr}\ (i+1,\ j+1)$ >

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

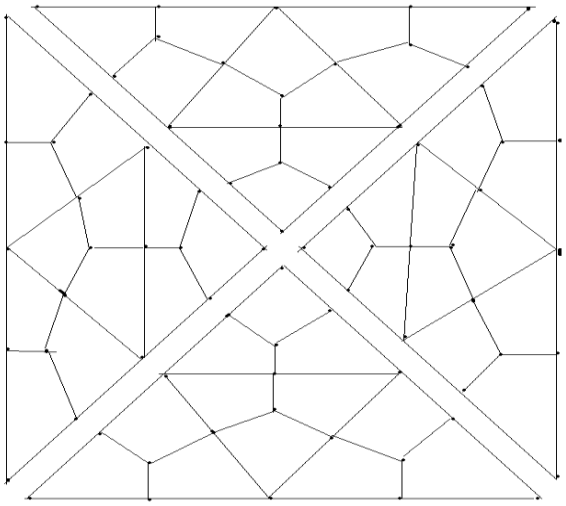
As an example, consider the geometry shown in Fig. 8(a). This is a a square region which is simply chosen for illustration. We divide this region into four LOOPs as shown in Fig.8(d). These LOOPs 1,2,3 and 4 are triangles each with three sides. After the LOOPs are defined, the number of elements for each LOOP is selected to produce the mesh shown in Fig. 8(c).The complete mesh is shown in Fig.8(b)

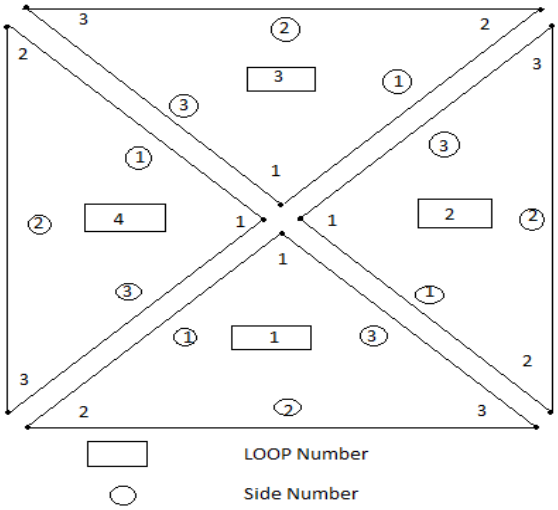


8 (a) 8 (b)

DOI: 10.18535/ijecs/v5i7.10

(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

5.1 First h-refinement of a triangle and linear convex polygon

The above finite element discritisation of the original triangles into 3- special quadrilaterals are created by joining the centroid of the triangle to the midpoints of its sides. The first refinement of the above scheme takes this concept further and creates in all 12-new quadrilaterals by joining the centroids of the 3-special quadrilaterals to the midpoints of the sides as shown in Fig.9a and Fig.9b. The first refinement of finite element mesh for a linear convex polygon depends on the algorithm stated in section 4 .

DOI: 10.18535/ijecs/v5i7.10

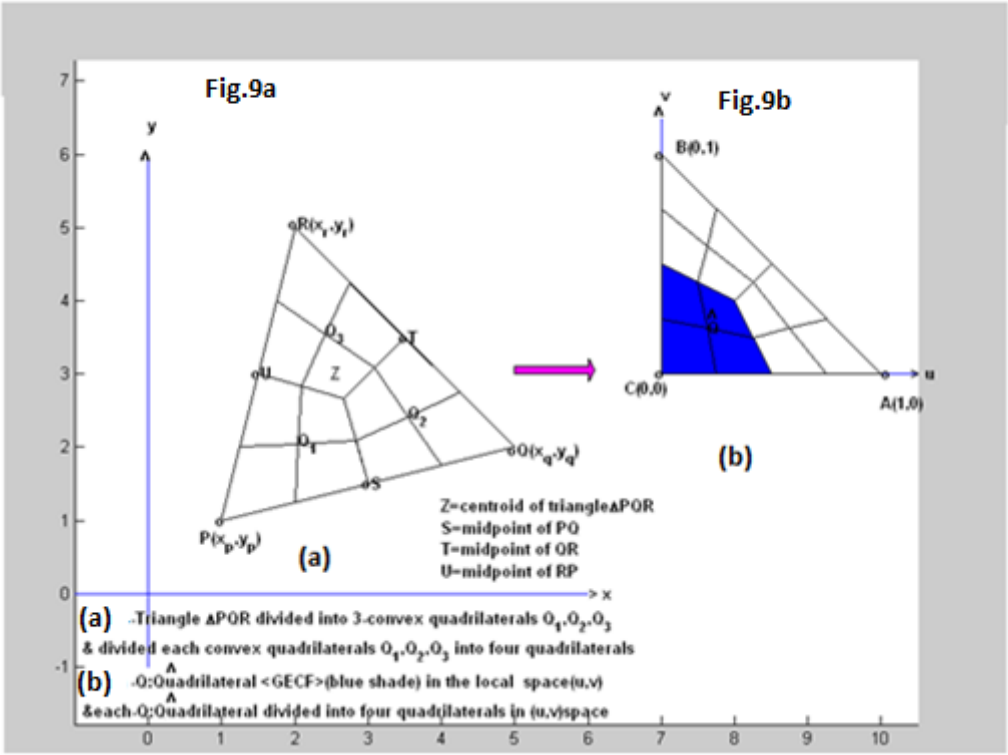


Fig.9a:Arbitrary triangle Δ_{PQR} in Cartesian Space(x,y)
Fig.9b Standard triangle in parametric space (u,v)

5.2 Second h-refinement of a triangle and linear convex polygon

The finite element discritisation of the original triangles into 3- special quadrilaterals are created by joining the centroid of the triangle to the midpoints of its sides. The first refinement of this scheme takes this concept further and creates in all 12-new small quadrilaterals obtained by joining the centroids of the original 3-special quadrilaterals to the midpoints of the sides as shown Fig.9a and Fig.9b. We take this concept further and creates in all 48-new small quadrilaterals by joining the centroids of the 12-special quadrilaterals of first refinement to the midpoints of the sides as shown in Fig.10a and Fig.10b.

DOI: 10.18535/ijecs/v5i7.10

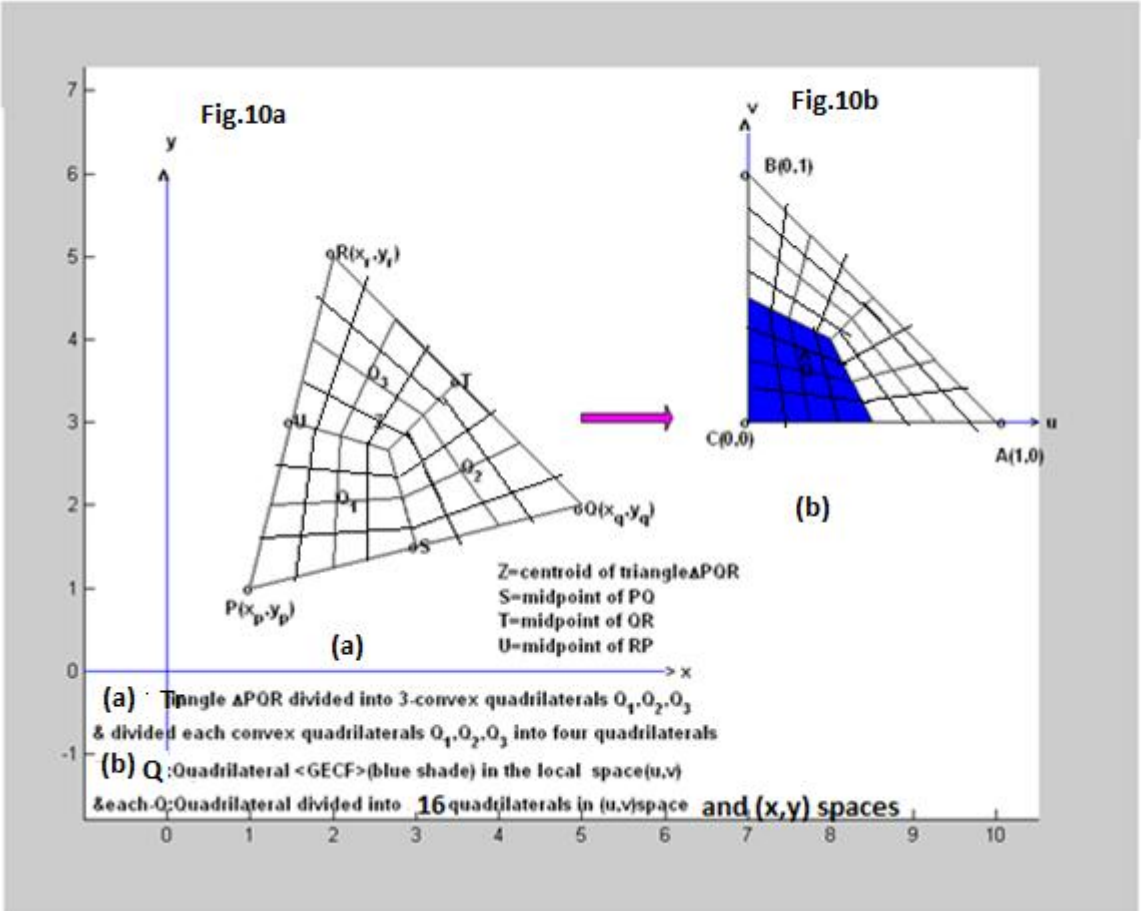


Fig.10a:Arbitrary triangle Δ_{PQR} in Cartesian Space(x,y)
Fig.10b Standard triangle in parametric space (u,v)\

5.3 Algorithm for first h-refinement and second h-refinement for a Triangle and linear

Convex Polygon

By application of the procedure developed in sections 2 and 3,we can obtain h-refinements of any triangle by triangular divisions.We take this further and apply an automatic triangular to quadrilateral conversion scheme in which each isolated triangle is split into three quadrilaterals by adding three vertices in the middle of edges and a vertex at the centroid of the triangular element. This creates an all quadrilateral mesh[16]. In this scheme each triangle is divided into three quadrilaterals.We now take up the refinement of this mesh by dividing each quadrilateral into four new quadrilaterals.This is done by locating the centroid and joining this centroid to the mid points of the sides of the quadrilateral.We do this by creating a midpoint array for all the nodes of the all quadrilateral mesh in [16]. and initialize this to zero.We then consider each element one by one and replace the zeros by actual node number.This is incorporated in the MATLAB codes.We follow a similar procedure for the second h-refinement for a triangle

By application of the procedure developed in section 4,we can obtain an all quadrilateral mesh for the linear convex polygon.We can now repeat the above procedure to obtain the first and second h-refinements for the convex polygon.

6. Application Examples

6.1 Mesh Generation Over an Arbitrary Triangle

In applications to boundary value problems due to symmetry considerations, we may have to discretize an arbitrary triangle. Our purpose is to have a code which automatically generates convex quadrangulations of the domain by assuming the input as coordinates of the vertices. We use the theory and procedure developed in section 2 and section 3 of this paper for this purpose.

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

6.2 Mesh Generation for a Linear Convex Polygonal Domain

In several physical applications in science and engineering, the boundary value problem require meshes generated over convex polygons. Again our aim is to have a code which automatically generates a mesh of convex quadrilaterals for the complex domains. We use the theory and procedure developed in sections 2, 3 and 4 for this purpose.

We have included some meshes generated by using the above MATLAB codes. We further illustrate the application of above codes by generating meshes for a square, five sided and six sided convex polygons. Some sample commands to generate the polygons stated above appear in the comment lines of programs. In all the cases the sample data is a part of the codes.

Conclusions

This paper presents an automatic indirect quadrilateral mesh generator for the two dimensional linear convex polygonal domains. This mesh generation is made fully automatic and allows the user to define the problem domain with minimum amount of input such as coordinates of boundary. Once this input is created, by selecting an appropriate interior point of the convex 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 centroid of the triangular element which creates three special quadrilaterals. 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 convex polygonal domain into all quadrilaterals, thus propogating a uniform refinement.This is the original h-refinement by special quadrilaterals. This simple method generates high quality mesh whose elements confirm well to the requested shape by refining the problem domain. Then each of these special quadrilateral is further divided into four quadrilaterals by joining the centroid of special quadrilateral to the midpoints of four sides.This procedure creates 12-special quadrilaterals.This discretises the entire polygonal domain into a finite number of special quadrilaterals.This can be referred as firs h-refinement by special quadrilaterals. **Then each of these quadrilateral is divided into four smaller quadrilaterals again by joining the centroids to the mid points of sides which we call as the second h-refinement by 48 special quadrilaterals**

These h-refinement methods 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 standard triangle, an arbitrary triangle, a square and an arbitrary convex polygonal domain. We believe that this work will be useful for various applications in science and engineering. The following is the **The list of MATLAB PROGRAMS**.

(I)FOR FIRST H-REFINEMENTS

(1a) quadrilateral_mesh4twocentroidsMOINEX_q4.m
(1b) polygonal_domain_twocentroidscoordinates.m
(1c) nodaladdresses4twocentroidspecial_convex_quadrilaterals.m
(1d) nodaladdresses4twocentroidspecial_convex_quadrilaterals_trial.m
(1e) generate_area_coordinate_over_the_standard_triangle.m

(II)FOR SECOND H-REFINEMENTS

(2a) quadrilateral_mesh4fivecentroidsMOINEX_q4.m
(2b) polygonal_domain_fivecentroidscoordinates.m
(2c) nodaladdresses4fivecentroidspecial_convex_quadrilaterals.m
(2d) nodaladdresses4fivecentroidspecial_convex_quadrilaterals_trial.m

(2e) `generate_area_coordinate_over_the_standard_triangle.m`

We may note that programs (1e) and (2e) are the same

REFERENCES

[1] O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, The Finite Element Method, its basis and fundamentals, Sixth edition, Butterworth-Heinemann, An Imprint of Elsevier,(2000).

[2] K. J. Bathe, Finite Element Procedures, Prentice Hall, Inc. Englewood Cliffs, N.J (1996).

[3] R. D. Cook, Concepts and Applications of Finite Element Analysis, 2nd Ed. John Wiley & Sons, Inc. New York (1981).

[4] G. R. Cowper, Gaussian Quadrature formulas for triangles, Int. J. Numer. Methods. Eng 7 (1973) 405 - 408.

[5] D. A. Dunavant, High degree efficient symmetrical Gaussian Quadrature rules for triangle, Int. J. Numer. Methods Eng 21 (1985) 1129-1148.

[6] A. H. Stroud, Approximate Calculation of Multiple integrals, in: Prentice Hall series in Automatic Computation, Prentice Hall, Inc. Englewood Cliffs. N. J (1971).

[7] G. Lague, R. Baldur, Extended numerical integration method for triangular surfaces, Int. J. Numer. Methods Eng 11 (1977) 388-392.

[8] H. T. Rathod, K. V. Nagaraja, B. Venkatesudu, N. L. Ramesh, Gauss Legendre Quadrature over a triangle, J. Indian Inst. Sci 84 (2004) 183-188.

[9] H. T. Rathod, K. V. Nagaraja, B. Venkatesudu, Symmetric Gauss Legendre Quadrature formulas for composite numerical integration over a triangular surface, Applied Mathematics and Computation 188 (2007) 865-876.

[10] H. T. Rathod, K.V. Nagaraja, B. Venkatesudu, On the application of two symmetric Gauss Legendre quadrature rules for composite numerical integration over s triangular surface, Applied Mathematics and Computation 190 (2007) 21-39.

[11] A. Sommariva, M. Vianello, Product Gauss cubature over polygons based on Green's Integration formula, BIT Numerical Mathematics, 47 (2007) 441-453.

[12] C. J. Li, P. Lamberti, C. Dagnino, Numerical integration over polygons using an eight-node quadrilateral finite element, Journal of Computational and Applied Mathematics, 233 (2009) 279-292.

[13] C. T. Reddy, D. J. Shippy, Alternative integration formulae for triangular finite elements, Int. J. Numer. Methods Eng 17 (1981) 1890-1896.

[14] Md. Shafiqual Islam and M. Alamgir Hossain, Numerical integration over an arbitrary quadrilateral region, Appl. Math. Computation 210 (2009) 515-524.

[15] Siraj-ul-Islam, Imran Aziz and Fazal Haq, A comparative study of numerical integration based on haar wavelets and hybrid functions, Computers and Mathematics with Applications, 59 (2010) No.6, 2026-2036.

[16] *H.T.Rathod,B.Venkatesh,K.T.Shivaram,andT.M.Mamtha, Numerical Integration Over Polygonal Domains using Convex Quadrangulation and Gauss Legendre Quadrature Rules, International Journal of* ***Engineering and Computer*** *Science , Vol.2, no.8(2013),pp2576-2610*

[17] ***H.T.Rathod ,K.V.Vijayakumar,A.S.Hariprasad,Gauss Legendre Formulas for Polygons Using an All Quadrilateral Finite Element Mesh of Triangular Surfaces, , International Journal of Engineering and Computer Science,ISSN 2319-7242, Vol.3,issue6(2014),pp6636-6680***

[18] **H. T. Rathod, K.V.Vijayakumar, A. S. Hariprasad , K. Sugantha Devi, Gauss Legendre Quadrature Formulas for Polygons By Using a Second Refinement of an All Quadrilateral Finite Element Mesh of Triangular Surfaces, *International Journal of* Engineering and computer Science, *ISSN 2319-7242, Vol.5,issue 06(2016)pp 16891-16976***

Matlab Programs

(I)FOR FIRST H-REFINEMENTS

(1a) quadrilateral_mesh4twocentroidsMOINEX_q4.m

```
function[]=quadrilateral_mesh4twocentroidsMOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
%clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
%quadrilateral_mesh4twocentroidsMOINEX_q4(1,2,3,3,1,2,6,1,1)
%quadrilateral_mesh4twocentroidsMOINEX_q4(1,2,3,3,4,4,6,1,1)
%quadrilateral_mesh4twocentroidsMOINEX_q4([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4twocentroidsMOINEX_q4([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1,1)
%quadrilateral_mesh4twocentroidsMOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,8,1,1)
[coord,gcoord,smspqd,nodetel,nnode,nel]=polygonal_domain_twocentroidscoordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(smspqd);

disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where i->node no. and j->x or y
%_____
```

```
%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
%clf
figure(ndiv/2)
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(smspqd(i,j),1);
y(1,j)=ycoord(smspqd(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])
end
 %
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=2
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='mesh with ';
st2=num2str(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<=2
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end
%axis off
```

(1b) `polygonal_domain_twocentroidscoordinates.m`

```
function[coord,gcoord,smspqd,nodetel,nnode,nel]=polygonal_domain_twocentroidscoordinates(n1,n2,n3,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
syms U V W xi yi

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

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

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

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

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

end
if (nmax>3)

[eln,smspqd,rrr,nodes,nodetel]=nodaladdresses4twocentroidspecial_convex_quadrilaterals_trial(n1,n2,n3,nmax,nu
mtri,n)
end

if (nmax==3)
   [eln,rrr,nodes,nodetel,smspqd]=nodaladdresses4twocentroidspecial_convex_quadrilaterals(n)
end

[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(smspqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
nodes
smspqd
%
nnode=max(max(smspqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])


nnode
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)]
x1=x(n1(itri,1),1)
```

```
x2=x(n2(itri,1),1)
x3=x(n3(itri,1),1)
%
y1=y(n1(itri,1),1)
y2=y(n2(itri,1),1)
y3=y(n3(itri,1),1)
rrr(:,:,itri)
U'
V'
W'
kk=0;
for ii=1:n+1
    for jj=1:(n+1)-(ii-1)
        kk=kk+1;
        mm=rrr(ii,jj,itri);
        uu=U(kk,1);vv=V(kk,1);ww=W(kk,1);
        xi(mm,1)=x1*ww+x2*uu+x3*vv;
        yi(mm,1)=y1*ww+y2*uu+y3*vv;
    end
end
[xi yi]
%add coordinates of centroid
 ne=(n/2)^2;
% stdnode=kk;
 for iii=1+(itri-1)*ne:ne*itri
     %kk=kk+1;
     mm1=eln(iii,1);
     mm2=eln(iii,2);
     mm3=eln(iii,3);
     mm=eln(iii,7);
     xi(mm,1)=(xi(mm1,1)+xi(mm2,1)+xi(mm3,1))/3;
     yi(mm,1)=(yi(mm1,1)+yi(mm2,1)+yi(mm3,1))/3;
     for tridiv=1:3
         iv=tridiv-3;
    %1=========================================
     nn1=nodes(3*iii+iv,1);nn2=nodes(3*iii+iv,2);nn3=nodes(3*iii+iv,3);
     nn4=nodes(3*iii+iv,4);nn5=nodes(3*iii+iv,5);nn6=nodes(3*iii+iv,6);
     nn7=nodes(3*iii+iv,7);nn8=nodes(3*iii+iv,8);nn9=nodes(3*iii+iv,9);
    %1=========================================================
    xi(nn5,1)=(xi(nn1,1)+xi(nn2,1))/2;yi(nn5,1)=(yi(nn1,1)+yi(nn2,1))/2;
    xi(nn6,1)=(xi(nn2,1)+xi(nn3,1))/2;yi(nn6,1)=(yi(nn2,1)+yi(nn3,1))/2;
    xi(nn7,1)=(xi(nn3,1)+xi(nn4,1))/2;yi(nn7,1)=(yi(nn3,1)+yi(nn4,1))/2;
    xi(nn8,1)=(xi(nn1,1)+xi(nn4,1))/2;yi(nn8,1)=(yi(nn1,1)+yi(nn4,1))/2;

xi(nn9,1)=(xi(nn1,1)+xi(nn2,1)+xi(nn3,1)+xi(nn4,1))/4;yi(nn9,1)=(yi(nn1,1)+yi(nn2,1)+yi(nn3,1)+yi(nn4,1))/4;
     end



 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));
```

(1c) nodaladdresses4twocentroidspecial_convex_quadrilaterals.m

```
function[eln,rrr,nodes,nodetel,smspqd]=nodaladdresses4twocentroidspecial_convex_quadrilaterals(n)
%[eln,rrr,nodes,nodetel,smspqd]=nodaladdresses4twocentroidspecial_convex_quadrilaterals(n)
%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 with centroid
%spqd=4-node special convex quadrilateral
%n=number of divisions of a side and n must be even,i.e.n=2,4,6,.....
%syms mst_tri x
%disp('vertex nodes of triangle')
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2,1)=3;
```

```
%disp('vertex nodes of triangle')
%base edge
kk=3;
for k=2:n
   kk=kk+1;
   elm(k,1)=kk;
end
%disp('left edge nodes')
nni=1;
for i=0:(n-2)
   nni=nni+(n-i)+1;
   elm(nni,1)=3*n-i;
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-2)
   nni=nni+(n-i);
   elm(nni,1)=(n+3)+i;
end

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

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



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

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

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

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

%for mmm=1:mm
   %spqd(:,1:4)
 %end
%
%ss1='number of 6-node triangles with centroid=';
%[p1,q1]=size(eln);
%disp([ss1 num2str(p1)])
%
%eln
%
%ss2='number of 4-node special convex quadrilaterals =';
%[p2,q2]=size(spqd);
%disp([ss2 num2str(p2)])
%
%spqd
%**************************************88
for inum=1:nnd
   for jnum=1:nnd
      mdpt(inum,jnum)=0;
   end
end
nd=nnd;
for mmm=1:mm
   mmm1=nodes(mmm,1);
   mmm2=nodes(mmm,2);
   mmm3=nodes(mmm,3);
   mmm4=nodes(mmm,4);
```

```
%midpoint side-1 of 4-node special quadrilateral
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 4-node special quadrilateral
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 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
   nd=nd+1;
  mdpt(mmm3,mmm4)=nd;
  mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
   nd=nd+1;
  mdpt(mmm4,mmm1)=nd;
  mdpt(mmm1,mmm4)=nd;
end
nd=nd+1;
  nodes(mmm,5)=mdpt(mmm1,mmm2);
  nodes(mmm,6)=mdpt(mmm2,mmm3);
  nodes(mmm,7)=mdpt(mmm3,mmm4);
  nodes(mmm,8)=mdpt(mmm4,mmm1);
  nodes(mmm,9)=nd;
end
nnode=nd;
nel=mm;
%

mmmm=0;
for iel=1:nel
   for jel=1:4
   mmmm=mmmm+1;
      switch jel
       case 1
         smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,8) nodes(iel,1) nodes(iel,5)];
       case 2
         smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,5) nodes(iel,2) nodes(iel,6)];
       case 3
         smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,6) nodes(iel,3) nodes(iel,7)];
       case 4
         smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,7) nodes(iel,4) nodes(iel,8)];

      end
    end
end

disp([smspqd])
```

(1d) nodaladdresses4twocentroidspecial_convex_quadrilaterals_trial.m

```
function[eln,smspqd,rrr,nodes,nodetel]=nodaladdresses4twocentroidspecial_convex_quadrilaterals_trial(n1,n2,n3
,nmax,numtri,n)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
```

```
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4)
%syms mst_tri x
ne=0;
nitri=nmax-1;
for itri=1:nitri
   elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
if itri==1
kk=nmax;
for k=2:n
   kk=kk+1
   elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
   elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
   lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
   mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
   hh=hh+1;
   nni=nni+(n-i);
   elm(nni,1)=(mmax+1)+i;
   qq(hh,1)=(mmax+1)+i;

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

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


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

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

```
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri



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

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

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

end% if rem(n,2)==0
ne
%for kk=1:ne
```

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

nnd=max(max(eln))
if (n>3)
for kkk=1+(itri-1)*numtri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
end
end
if n==2
 for kkk=itri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
 end
end
%for kk=1:ne
%[eln(kk,1:7)]
%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
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd


end%itri

%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
```

```
%for mmm=1:mm
   %spqd(:,1:4)
 %end
%
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)])
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])

%**********************************************************************
for inum=1:nnd
   for jnum=1:nnd
      mdpt(inum,jnum)=0;
   end
end
nd=nnd;
for mmm=1:mm
   mmm1=nodes(mmm,1);
   mmm2=nodes(mmm,2);
   mmm3=nodes(mmm,3);
   mmm4=nodes(mmm,4);
%midpoint side-1 of 4-node special quadrilateral
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 4-node special quadrilateral
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 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
   nd=nd+1;
  mdpt(mmm3,mmm4)=nd;
  mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
   nd=nd+1;
  mdpt(mmm4,mmm1)=nd;
  mdpt(mmm1,mmm4)=nd;
end
nd=nd+1;
  nodes(mmm,5)=mdpt(mmm1,mmm2);
  nodes(mmm,6)=mdpt(mmm2,mmm3);
  nodes(mmm,7)=mdpt(mmm3,mmm4);
  nodes(mmm,8)=mdpt(mmm4,mmm1);
  nodes(mmm,9)=nd;
end
nnode=nd;
nel=mm;
%

mmmm=0;
for iel=1:nel
   for jel=1:4
   mmmm=mmmm+1;
      switch jel
       case 1
```

```
        smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,8) nodes(iel,1) nodes(iel,5)];
      case 2
        smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,5) nodes(iel,2) nodes(iel,6)];
      case 3
        smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,6) nodes(iel,3) nodes(iel,7)];
      case 4
        smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,7) nodes(iel,4) nodes(iel,8)];

    end
  end
end

disp([smspqd])
```

(1e) generate_area_coordinate_over_the_standard_triangle.m

```
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
```

(II)FOR SECOND H-REFINEMENTS

(2a) quadrilateral_mesh4fivecentroidsMOINEX_q4.m

```
function[]=quadrilateral_mesh4fivecentroidsMOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
%clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
%quadrilateral_mesh4fivecentroidsMOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,8,1,1)
%quadrilateral_mesh4fivecentroidsMOINEX_q4([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4fivecentroidsMOINEX_q4(1,2,3,3,1,2,6,1,1)
%quadrilateral_mesh4fivecentroidsMOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
%quadrilateral_mesh4fivecentroidsMOINEX_q4([5;5;5;5],[1;2;3;4],[2;3;4;1],5,1,2,11,1,1)
%[coord,gcoord,smspqd,nodetel,nnode,nel]=polygonal_domain_fivecentroidscoordinates(n1,n2,n3,nmax,numtri,ndiv,
mesh)
%[nel,nnel]=size(smspqd);
[coord,gcoord,ssmspqd,nodetel,nnode,nel]=polygonal_domain_fivecentroidscoordinates(n1,n2,n3,nmax,numtri,ndiv,
mesh)
[nel,nnel]=size(ssmspqd);
disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where i->node no. and j->x or y
%_____________________________________________________________

%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
%clf
figure(ndiv/2)
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(ssmspqd(i,j),1);
y(1,j)=ycoord(ssmspqd(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])
end
 %
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=2

middx=mean(xvec(1,1:4));
middy=mean(yvec(1,1:4));
midx(i,1)=middx;
midy(i,1)=middy;
text(middx,middy,['*',num2str(i)]);

hold on

end

end;%i loop

xlabel('x axis')
ylabel('y axis')
st1='mesh with ';
st2=num2str(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<=2
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['*',num2str(jj)]);
%text(gcoord(jj,1),gcoord(jj,2),['o']);
end
hold on

end
```

(2b) polygonal_domain_fivecentroidscoordinates.m

```
function[coord,gcoord,ssmspqd,nodetel,nnode,nel]=polygonal_domain_fivecentroidscoordinates(n1,n2,n3,nmax,numt
ri,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
syms U V W xi yi

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


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

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

case 4%for a unit square: -0.5<=x,y<=0.5
  %  1    2    3   4   5   6    7   8
```

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

x=sym([0;1;1/2])
y=sym([0;0;sqrt(3)/2])
case 8%for a convexpolygonsixside
  %    1    2   3  4   5  6  7  8
x=sym([0.5;0.1;0.7;1;.75;.5;.25;0])
y=sym([0.5;0;0.2;.5;.85;1;.625;.25])
case 9%for a convexpolygeightside
%    1  2   3  4   5    6  7     8   9
x=([.2;0.5;.8;1.0;.75;0.5;0.25;0.0;0.5])
y=([.2;.05;.2;0.5;.85;1.0;0.90;0.6;0.5])
case 10
%    1  2   3  4   5    6  7     8   9
x=([0.5;.2;0.5;.8;1.0;.75;0.5;0.25;0.0])
y=([0.5;.2;.05;.2;0.5;.85;1.0;0.90;0.6])
 case 11 %a square
 %  1 2 3 4  5
x=([0;1;1;0;0.5])
y=([0;0;1;1;0.5])
end
if (nmax>3)

[eln,rrr,nodes,nodetel,nnodes,ssmspqd]=nodaladdresses4fivecentroidspecial_convex_quadrilaterals_trial(n1,n2,n
3,nmax,numtri,n)
end

if (nmax==3)
   [eln,rrr,nodes,nodetel,nnodes,ssmspqd]=nodaladdresses4fivecentroidspecial_convex_quadrilaterals(n)
end

[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(ssmspqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
%nodes
%smspqd
%
nnode=max(max(ssmspqd));
max(max(nnodes))
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])


nnode
nitri=nmax-1;
if (nmax==3)nitri=1
end
%initialise to zero the nine node quadrilaterals: qn
qn=0;
for itri=1:nitri
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
x1=x(n1(itri,1),1)
x2=x(n2(itri,1),1)
x3=x(n3(itri,1),1)
%
y1=y(n1(itri,1),1)
y2=y(n2(itri,1),1)
y3=y(n3(itri,1),1)
rrr(:,:,itri)
U'
V'
```

```
W'
kk=0;
for ii=1:n+1
    for jj=1:(n+1)-(ii-1)
        kk=kk+1;
        mm=rrr(ii,jj,itri);
        uu=U(kk,1);vv=V(kk,1);ww=W(kk,1);
        xi(mm,1)=x1*ww+x2*uu+x3*vv;
        yi(mm,1)=y1*ww+y2*uu+y3*vv;
    end
end
[xi yi]
%add coordinates of centroid
 ne=(n/2)^2;
% stdnode=kk;
%
 for iii=1+(itri-1)*ne:ne*itri
     %kk=kk+1;
     mm1=eln(iii,1);
     mm2=eln(iii,2);
     mm3=eln(iii,3);
     mm=eln(iii,7);
     xi(mm,1)=(xi(mm1,1)+xi(mm2,1)+xi(mm3,1))/3;
     yi(mm,1)=(yi(mm1,1)+yi(mm2,1)+yi(mm3,1))/3;

   for tridiv=1:3
         iv=tridiv-3;
    %1=======================================
     nn1=nodes(3*iii+iv,1);nn2=nodes(3*iii+iv,2);nn3=nodes(3*iii+iv,3);
     nn4=nodes(3*iii+iv,4);nn5=nodes(3*iii+iv,5);nn6=nodes(3*iii+iv,6);
     nn7=nodes(3*iii+iv,7);nn8=nodes(3*iii+iv,8);nn9=nodes(3*iii+iv,9);
    %1=======================================================
    xi(nn5,1)=(xi(nn1,1)+xi(nn2,1))/2;yi(nn5,1)=(yi(nn1,1)+yi(nn2,1))/2;
    xi(nn6,1)=(xi(nn2,1)+xi(nn3,1))/2;yi(nn6,1)=(yi(nn2,1)+yi(nn3,1))/2;
    xi(nn7,1)=(xi(nn3,1)+xi(nn4,1))/2;yi(nn7,1)=(yi(nn3,1)+yi(nn4,1))/2;
    xi(nn8,1)=(xi(nn1,1)+xi(nn4,1))/2;yi(nn8,1)=(yi(nn1,1)+yi(nn4,1))/2;

xi(nn9,1)=(xi(nn1,1)+xi(nn2,1)+xi(nn3,1)+xi(nn4,1))/4;yi(nn9,1)=(yi(nn1,1)+yi(nn2,1)+yi(nn3,1)+yi(nn4,1))/4;
    [3*iii+iv]
    [nn1 nn2 nn3 nn4 nn5 nn6 nn7 nn8 nn9]

    for quadiv=1:4
        qn=qn+1;
    %2=======================================
     nnn1=nnodes(qn,1);nnn2=nnodes(qn,2);nnn3=nnodes(qn,3);
     nnn4=nnodes(qn,4);nnn5=nnodes(qn,5);nnn6=nnodes(qn,6);
     nnn7=nnodes(qn,7);nnn8=nnodes(qn,8);nnn9=nnodes(qn,9);
    %2=====================================================
    xi(nnn5,1)=(xi(nnn1,1)+xi(nnn2,1))/2;yi(nnn5,1)=(yi(nnn1,1)+yi(nnn2,1))/2;
    xi(nnn6,1)=(xi(nnn2,1)+xi(nnn3,1))/2;yi(nnn6,1)=(yi(nnn2,1)+yi(nnn3,1))/2;
    xi(nnn7,1)=(xi(nnn3,1)+xi(nnn4,1))/2;yi(nnn7,1)=(yi(nnn3,1)+yi(nnn4,1))/2;
    xi(nnn8,1)=(xi(nnn1,1)+xi(nnn4,1))/2;yi(nnn8,1)=(yi(nnn1,1)+yi(nnn4,1))/2;

xi(nnn9,1)=(xi(nnn1,1)+xi(nnn2,1)+xi(nnn3,1)+xi(nnn4,1))/4;yi(nnn9,1)=(yi(nnn1,1)+yi(nnn2,1)+yi(nnn3,1)+yi(nnn4,1))/4;
    disp('=====================================================')

    qn
    nnodes(qn,1:9)
   [nnn1 nnn2 nnn3 nnn4 nnn5 nnn6 nnn7 nnn8 nnn9]
   disp('=====================================================')
    end%for quadiv
  end% for tridiv

end

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

(2c) nodaladdresses4fivecentroidspecial_convex_quadrilaterals.m

```
function[eln,rrr,nodes,nodetel,nnodes,ssmspqd]=nodaladdresses4fivecentroidspecial_convex_quadrilaterals(n)
%[eln,rrr,nodes,nodetel,smspqd]=nodaladdresses4twocentroidspecial_convex_quadrilaterals(n)
```

```
%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 with centroid
%spqd=4-node special convex quadrilateral
%n=number of divisions of a side and n must be even,i.e.n=2,4,6,.......
%syms mst_tri x
%disp('vertex nodes of triangle')
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2,1)=3;
%disp('vertex nodes of triangle')
%base edge
kk=3;
for k=2:n
    kk=kk+1;
    elm(k,1)=kk;
end
%disp('left edge nodes')
nni=1;
for i=0:(n-2)
    nni=nni+(n-i)+1;
    elm(nni,1)=3*n-i;
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-2)
    nni=nni+(n-i);
    elm(nni,1)=(n+3)+i;
end

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

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



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

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

for k=1:2:n-1
N=N-2;
i=k;
for j=1:2:N
    ne=ne+1;
```

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

%for mmm=1:mm
    %spqd(:,1:4)
 %end
%
%ss1='number of 6-node triangles with centroid=';
%[p1,q1]=size(eln);
%disp([ss1 num2str(p1)])
%
%eln
%
%ss2='number of 4-node special convex quadrilaterals =';
%[p2,q2]=size(spqd);
%disp([ss2 num2str(p2)])
%
%spqd
%************************************88

%*************************************************************************************************************
***

for inum=1:nnd
    for jnum=1:nnd
```

```
        mdpt(inum,jnum)=0;
    end
end
nd=nnd;
for mmm=1:mm
    mmm1=nodes(mmm,1);
    mmm2=nodes(mmm,2);
    mmm3=nodes(mmm,3);
    mmm4=nodes(mmm,4);
%midpoint side-1 of 4-node special quadrilateral
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 4-node special quadrilateral
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 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
    nd=nd+1;
   mdpt(mmm3,mmm4)=nd;
   mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
    nd=nd+1;
   mdpt(mmm4,mmm1)=nd;
   mdpt(mmm1,mmm4)=nd;
end
nd=nd+1;
   nodes(mmm,5)=mdpt(mmm1,mmm2);
   nodes(mmm,6)=mdpt(mmm2,mmm3);
   nodes(mmm,7)=mdpt(mmm3,mmm4);
   nodes(mmm,8)=mdpt(mmm4,mmm1);
   nodes(mmm,9)=nd;
end
nnode=nd;
nel=mm
%

mmmm=0;
for iel=1:nel
    for jel=1:4
    mmmm=mmmm+1;
        switch jel
          case 1
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,8) nodes(iel,1) nodes(iel,5)];
          case 2
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,5) nodes(iel,2) nodes(iel,6)];
          case 3
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,6) nodes(iel,3) nodes(iel,7)];
          case 4
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,7) nodes(iel,4) nodes(iel,8)];

        end
     end
end

disp([smspqd])
%%generate smaller 4-special quadrilaterals q4 out of one nine node convex
%%quadrilaterals smspqd
for nelmt=1:16*3*(n/2)^2
    ssmspqd(nelmt,1:4)=0;
end
%mesh generation of eight node special quadrilaterals
%
for inum=1:nnode
    for jnum=1:nnode
        mdpt(inum,jnum)=0;
    end
end
nd=nnode
nnodes=smspqd
mmmm
%return
```

```
%***************
%.....................................................................
for mmmmm=1:mmmm
    mmmm1=nnodes(mmmmm,1);
    mmmm2=nnodes(mmmmm,2);
    mmmm3=nnodes(mmmmm,3);
    mmmm4=nnodes(mmmmm,4);
%midpoint side-1 of 4-node special quadrilateral
if((mdpt(mmmm1,mmmm2)==0)&(mdpt(mmmm2,mmmm1)==0))
    nd=nd+1;
   mdpt(mmmm1,mmmm2)=nd;
   mdpt(mmmm2,mmmm1)=nd;
end
%midpoint side-2 of 4-node special quadrilateral
if((mdpt(mmmm2,mmmm3)==0)&(mdpt(mmmm3,mmmm2)==0))
    nd=nd+1;
   mdpt(mmmm2,mmmm3)=nd;
   mdpt(mmmm3,mmmm2)=nd;
end
%midpoint side-3 of 4-node special quadrilateral
if((mdpt(mmmm3,mmmm4)==0)&(mdpt(mmmm4,mmmm3)==0))
    nd=nd+1;
   mdpt(mmmm3,mmmm4)=nd;
   mdpt(mmmm4,mmmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmmm4,mmmm1)==0)&(mdpt(mmmm1,mmmm4)==0))
    nd=nd+1;
   mdpt(mmmm4,mmmm1)=nd;
   mdpt(mmmm1,mmmm4)=nd;
end
nd=nd+1;
   nnodes(mmmmm,5)=mdpt(mmmm1,mmmm2);
   nnodes(mmmmm,6)=mdpt(mmmm2,mmmm3);
   nnodes(mmmmm,7)=mdpt(mmmm3,mmmm4);
   nnodes(mmmmm,8)=mdpt(mmmm4,mmmm1);
   nnodes(mmmmm,9)=nd;
end
nnnode=nd
nel=mmmm
%smspqd=nnodes;
%return
%
nnodes
%smspqd
nnnn=0;
for iel=1:nel
    for jel=1:4
    nnnn=nnnn+1;
        switch jel
          case 1
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,8) nnodes(iel,1) nnodes(iel,5)];
          case 2
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,5) nnodes(iel,2) nnodes(iel,6)];
          case 3
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,6) nnodes(iel,3) nnodes(iel,7)];
          case 4
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,7) nnodes(iel,4) nnodes(iel,8)];

        end
     end
end

disp([ssmspqd])
nnnn
```

(2d) nodaladdresses4fivecentroidspecial_convex_quadrilaterals_trial.m

```
function[eln,rrr,nodes,nodetel,nnodes,ssmspqd]=nodaladdresses4fivecentroidspecial_convex_quadrilaterals_trial
(n1,n2,n3,nmax,numtri,n)
%[eln,rrr,nodes,nodetel,nnodes,ssmspqd]=nodaladdresses4fivecentroidspecial_convex_quadrilaterals_trial(n1,n2,
n3,nmax,numtri,n)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.......i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
```

```
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
ne=0;
nitri=nmax-1;
for itri=1:nitri
    elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
if itri==1
kk=nmax;
for k=2:n
    kk=kk+1
    elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
    elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
    lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
    lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
    mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
    hh=hh+1;
    nni=nni+(n-i);
    elm(nni,1)=(mmax+1)+i;
    qq(hh,1)=(mmax+1)+i;

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

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


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

if itri==nitri
disp('left edge nodes')
nni=1;gg=1;
for i=0:(n-2)
    gg=gg+1;
    nni=nni+(n-i)+1;
    elm(nni,1)=edgen1n2(gg,1);
end
%pp(n+1,1)=n3(itri,1);
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri
```

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

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

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

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

nnd=max(max(eln))
if (n>3)
for kkk=1+(itri-1)*numtri:ne
    nnd=nnd+1;
    eln(kkk,7)=nnd;
end
end
if n==2
```

```
 for kkk=itri:ne
    nnd=nnd+1;
    eln(kkk,7)=nnd;
 end
end
%for kk=1:ne
%[eln(kk,1:7)]
%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
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd


end%itri

%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




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

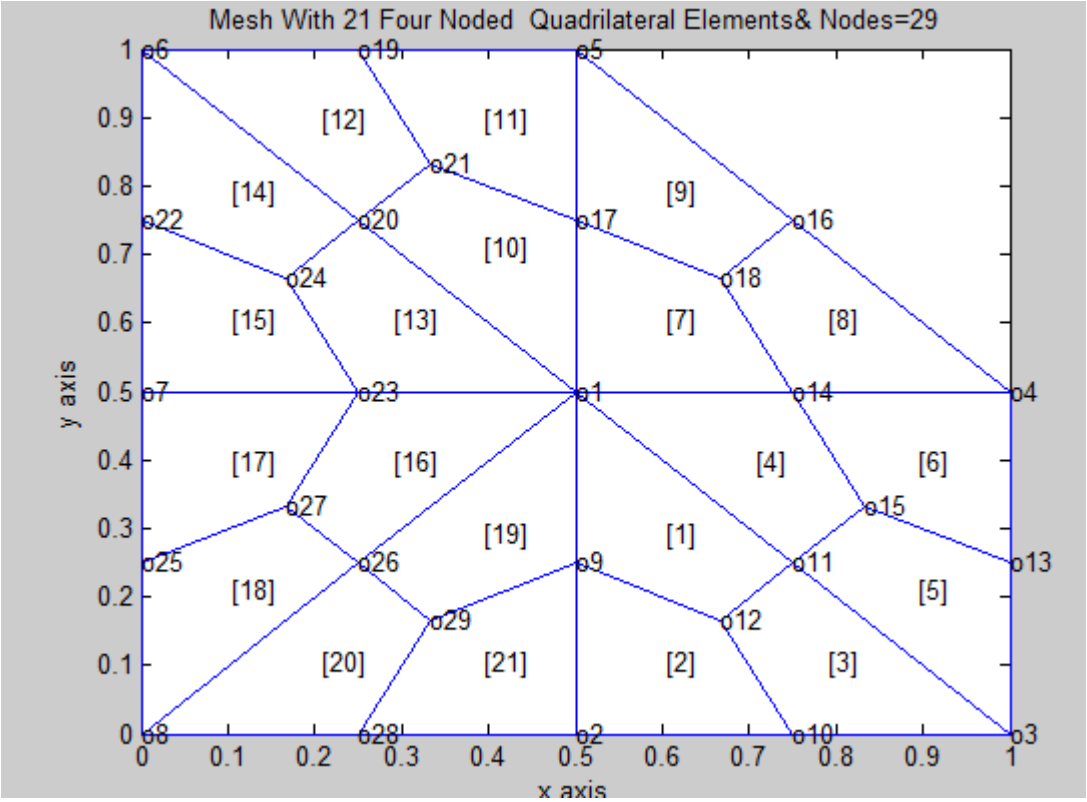
```
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])

%*********************************************************************
for inum=1:nnd
    for jnum=1:nnd
        mdpt(inum,jnum)=0;
    end
end
nd=nnd;
for mmm=1:mm
    mmm1=nodes(mmm,1);
    mmm2=nodes(mmm,2);
    mmm3=nodes(mmm,3);
    mmm4=nodes(mmm,4);
%midpoint side-1 of 4-node special quadrilateral
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 4-node special quadrilateral
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 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
    nd=nd+1;
   mdpt(mmm3,mmm4)=nd;
   mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
    nd=nd+1;
   mdpt(mmm4,mmm1)=nd;
   mdpt(mmm1,mmm4)=nd;
end
nd=nd+1;
   nodes(mmm,5)=mdpt(mmm1,mmm2);
   nodes(mmm,6)=mdpt(mmm2,mmm3);
   nodes(mmm,7)=mdpt(mmm3,mmm4);
   nodes(mmm,8)=mdpt(mmm4,mmm1);
   nodes(mmm,9)=nd;
end
nnode=nd;
nel=mm;
%

mmmm=0;
for iel=1:nel
    for jel=1:4
    mmmm=mmmm+1;
        switch jel
          case 1
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,8) nodes(iel,1) nodes(iel,5)];
          case 2
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,5) nodes(iel,2) nodes(iel,6)];
          case 3
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,6) nodes(iel,3) nodes(iel,7)];
          case 4
            smspqd(mmmm,1:4)=[nodes(iel,9) nodes(iel,7) nodes(iel,4) nodes(iel,8)];

        end
     end
end

disp([smspqd])
%*************************************************************************************************

%%generate smaller 4-special quadrilaterals q4 out of one nine node convex
%%quadrilaterals smspqd

for inum=1:nnode
    for jnum=1:nnode
```

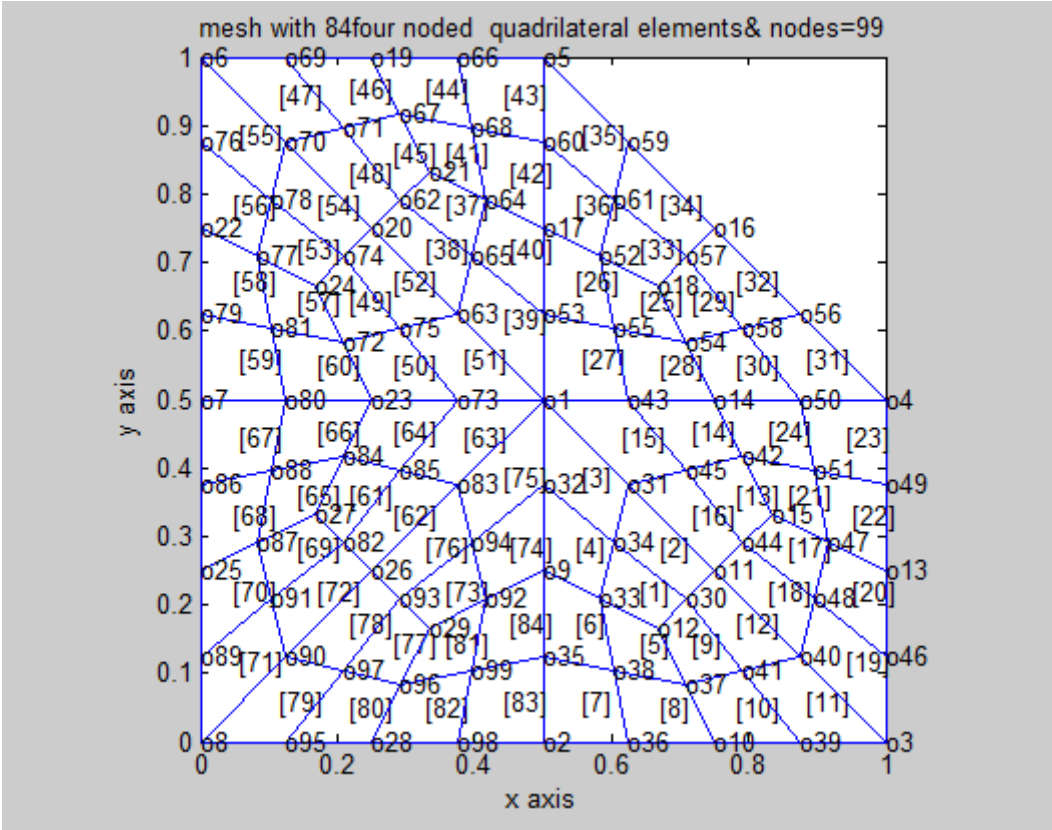
```
        mdpt(inum,jnum)=0;
    end
end
nd=nnode
nnodes=smspqd
mmmm
%return
%***************
%.....................................................................
for mmmmm=1:mmmm
    mmmm1=nnodes(mmmmm,1);
    mmmm2=nnodes(mmmmm,2);
    mmmm3=nnodes(mmmmm,3);
    mmmm4=nnodes(mmmmm,4);
%midpoint side-1 of 4-node special quadrilateral
if((mdpt(mmmm1,mmmm2)==0)&(mdpt(mmmm2,mmmm1)==0))
    nd=nd+1;
   mdpt(mmmm1,mmmm2)=nd;
   mdpt(mmmm2,mmmm1)=nd;
end
%midpoint side-2 of 4-node special quadrilateral
if((mdpt(mmmm2,mmmm3)==0)&(mdpt(mmmm3,mmmm2)==0))
    nd=nd+1;
   mdpt(mmmm2,mmmm3)=nd;
   mdpt(mmmm3,mmmm2)=nd;
end
%midpoint side-3 of 4-node special quadrilateral
if((mdpt(mmmm3,mmmm4)==0)&(mdpt(mmmm4,mmmm3)==0))
    nd=nd+1;
   mdpt(mmmm3,mmmm4)=nd;
   mdpt(mmmm4,mmmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmmm4,mmmm1)==0)&(mdpt(mmmm1,mmmm4)==0))
    nd=nd+1;
   mdpt(mmmm4,mmmm1)=nd;
   mdpt(mmmm1,mmmm4)=nd;
end
nd=nd+1;
   nnodes(mmmmm,5)=mdpt(mmmm1,mmmm2);
   nnodes(mmmmm,6)=mdpt(mmmm2,mmmm3);
   nnodes(mmmmm,7)=mdpt(mmmm3,mmmm4);
   nnodes(mmmmm,8)=mdpt(mmmm4,mmmm1);
   nnodes(mmmmm,9)=nd;
end
nnnode=nd
nel=mmmm
%smspqd=nnodes;
%return
%
nnodes
%smspqd
nnnn=0;
for iel=1:nel
    for jel=1:4
    nnnn=nnnn+1;
        switch jel
          case 1
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,8) nnodes(iel,1) nnodes(iel,5)];
          case 2
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,5) nnodes(iel,2) nnodes(iel,6)];
          case 3
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,6) nnodes(iel,3) nnodes(iel,7)];
          case 4
            ssmspqd(nnnn,1:4)=[nnodes(iel,9) nnodes(iel,7) nnodes(iel,4) nnodes(iel,8)];

        end
     end
end

disp([ssmspqd])

nnnn
```

SOME FINITE ELEMENT MESHES FOR FIRST H-REFINEMENTS

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

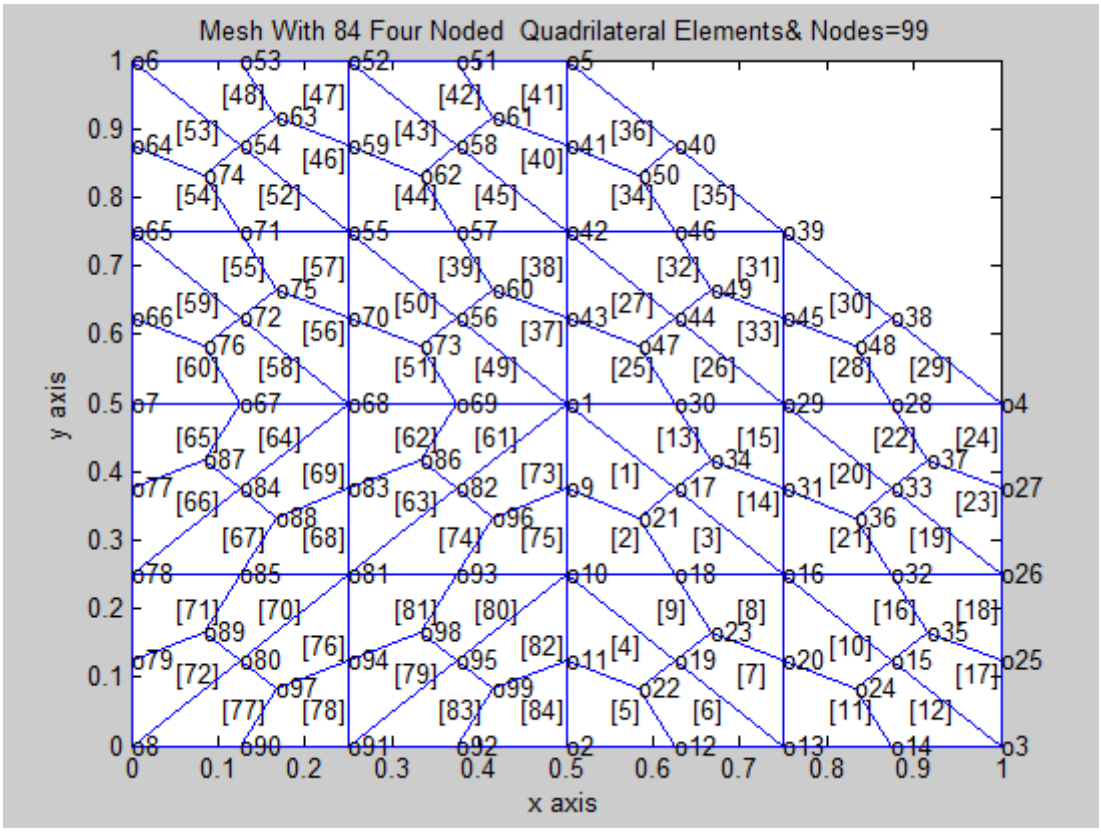


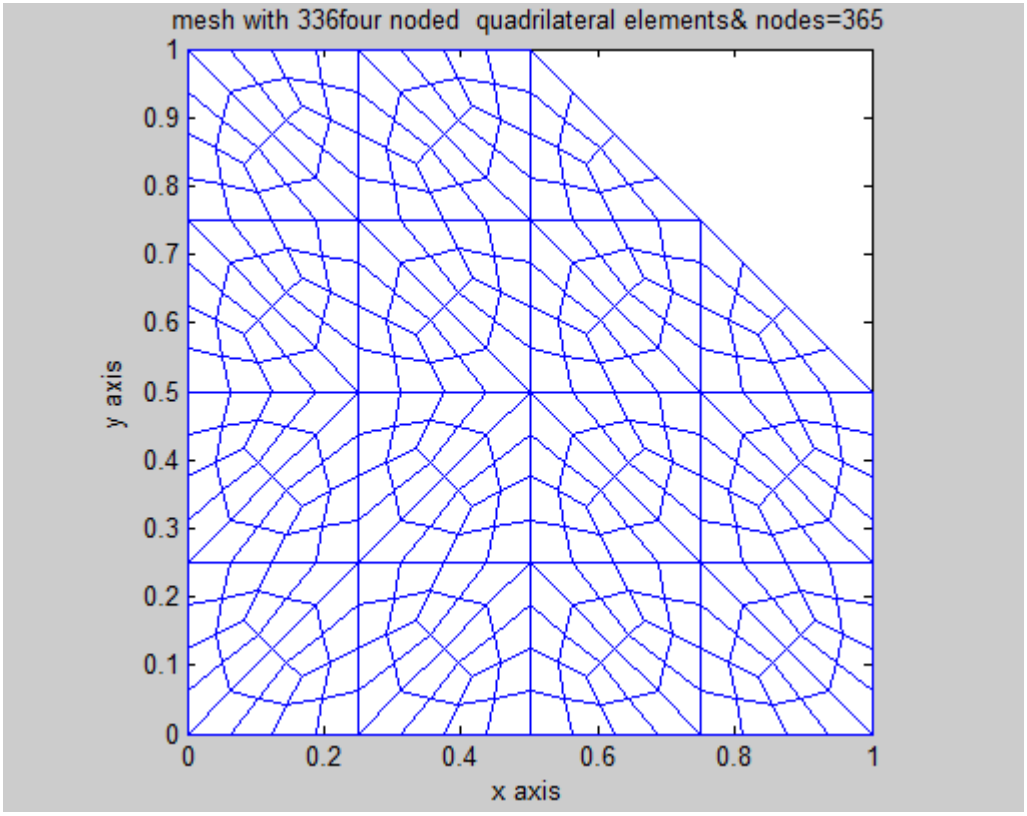


FIRST REFINEMENT

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

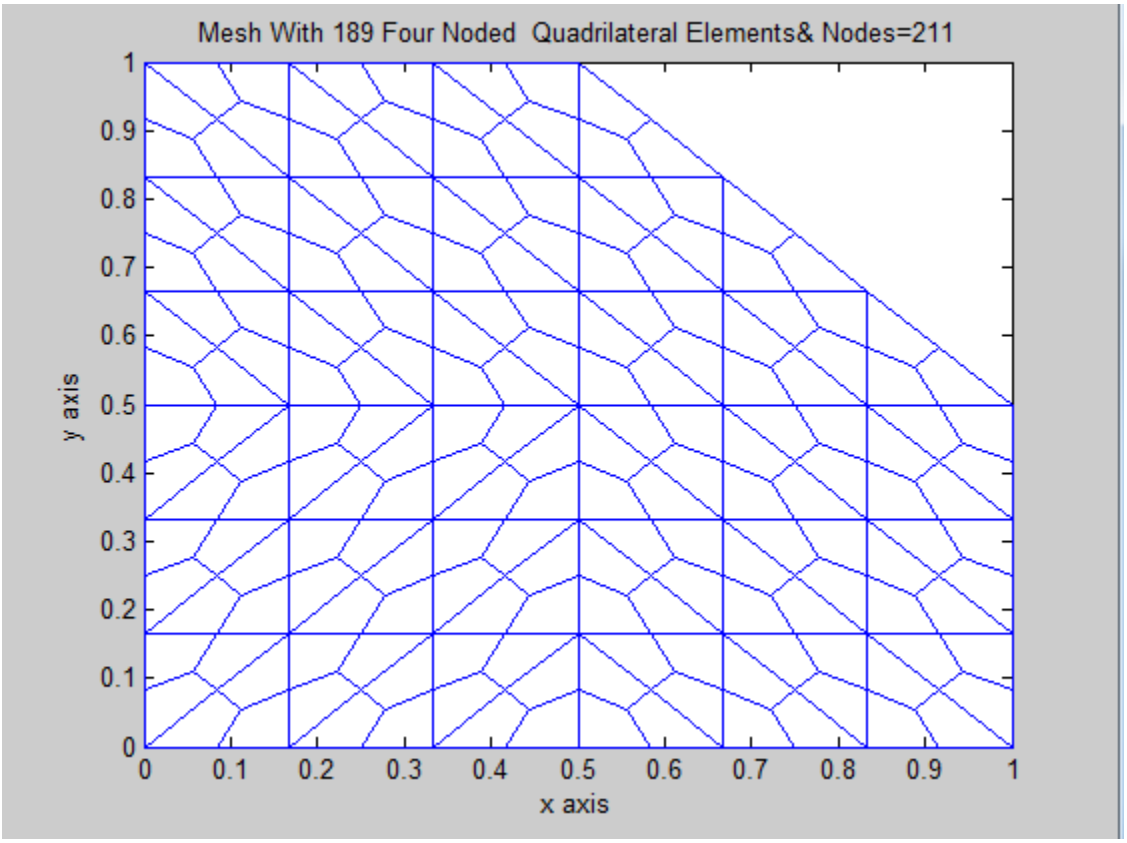


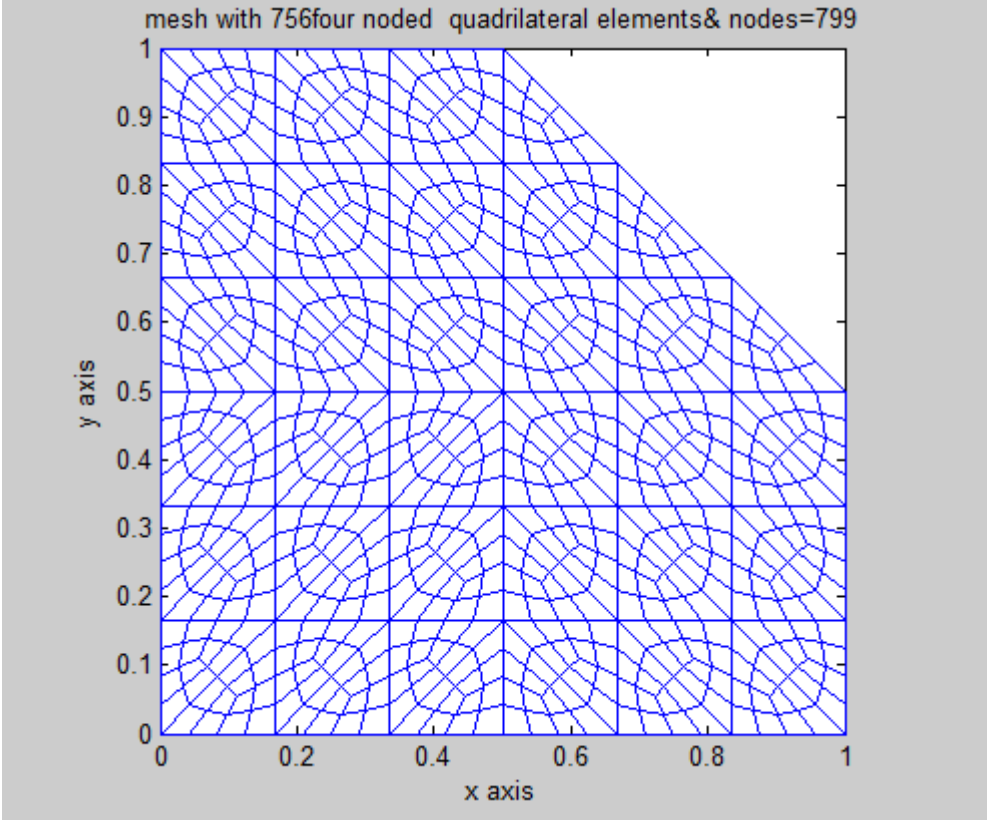


FIRST REFINEMENT

***ORIGINAL**==========

DOI: 10.18535/ijecs/v5i7.10

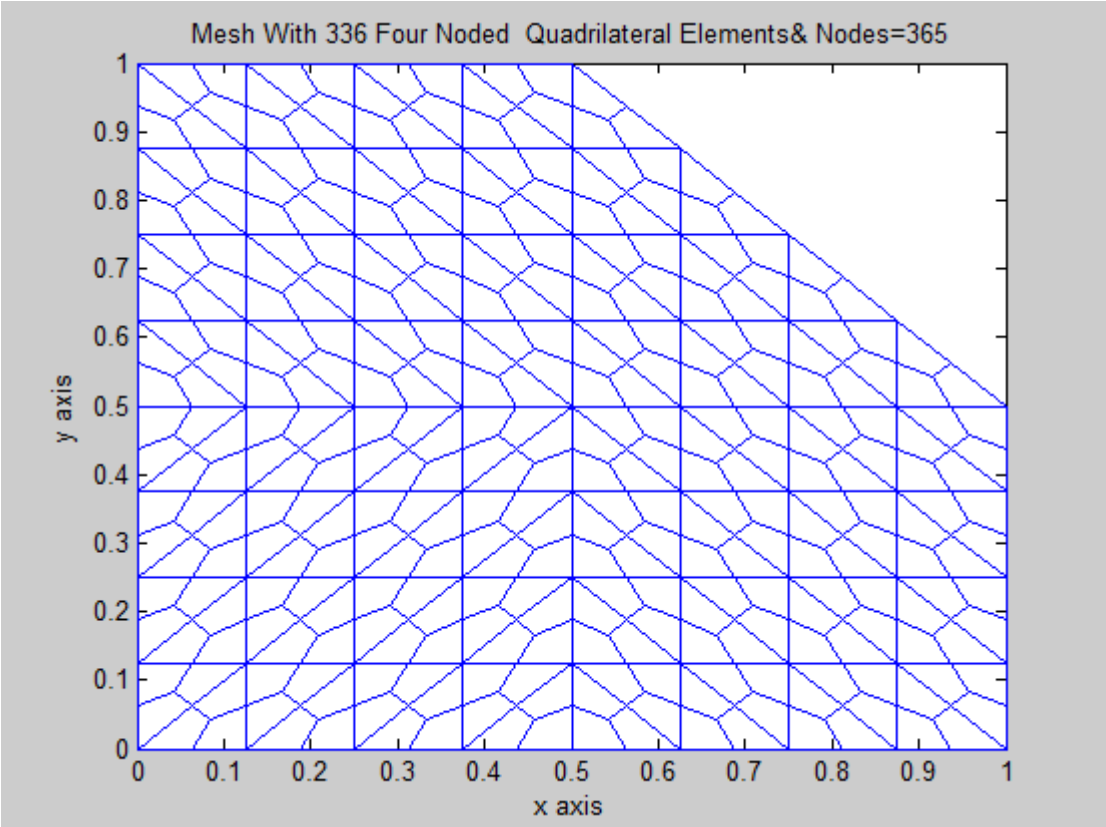


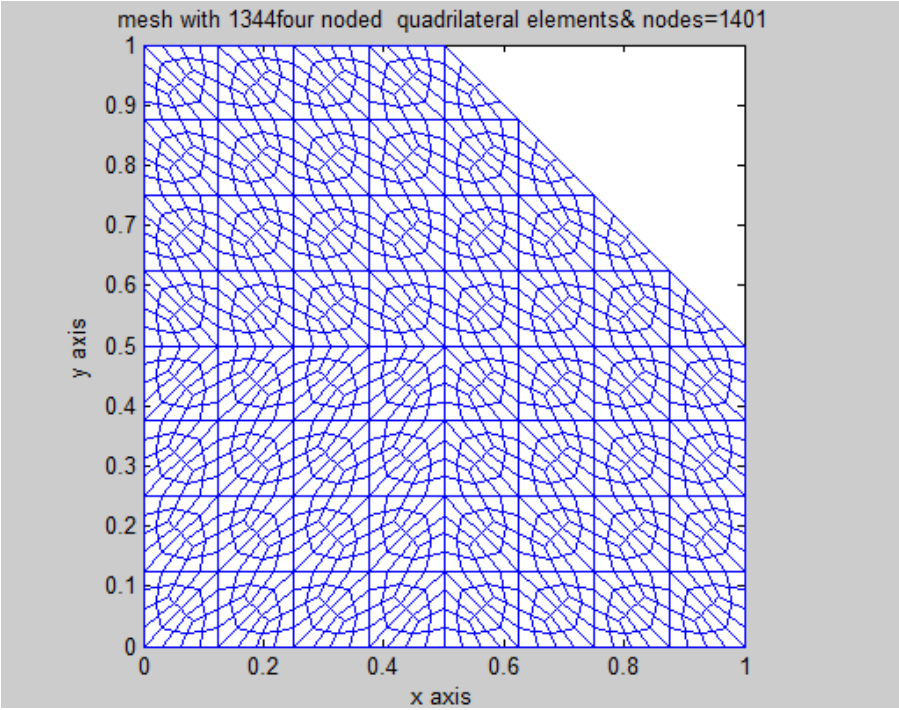


FIRST REFINEMENT

*ORIGINAL ==========

DOI: 10.18535/ijecs/v5i7.10

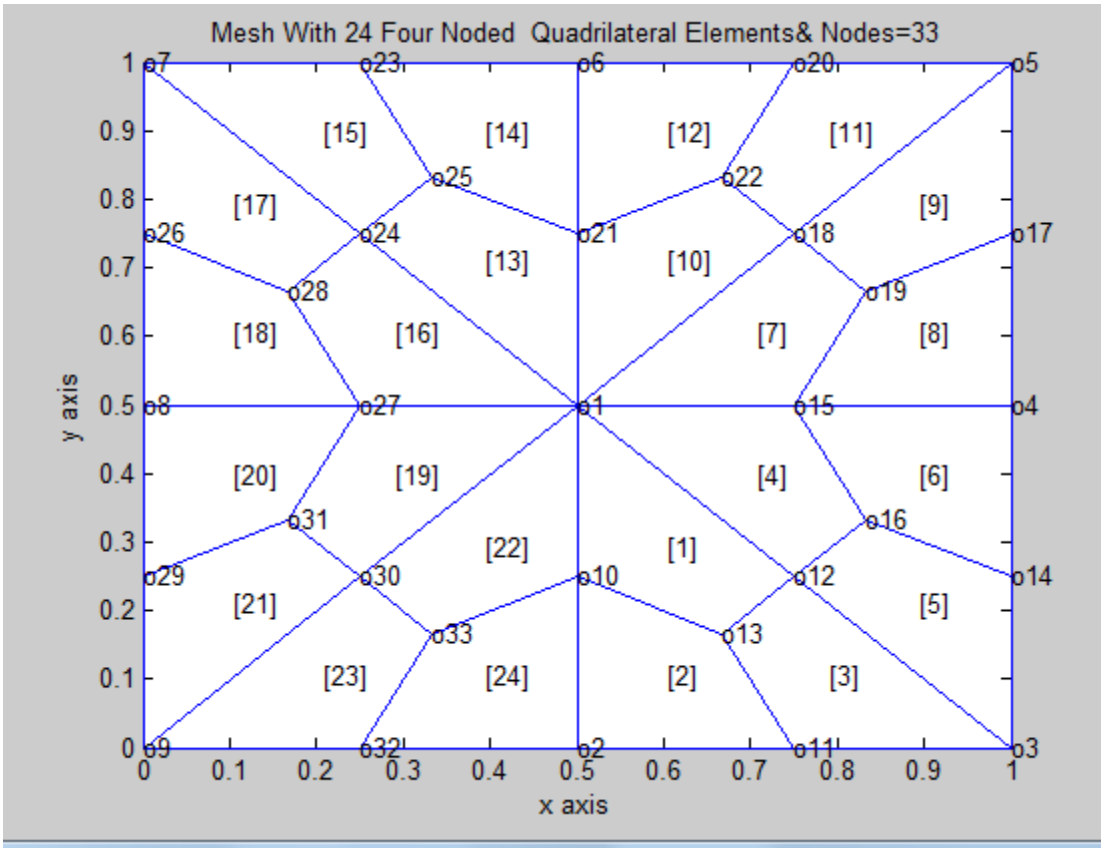


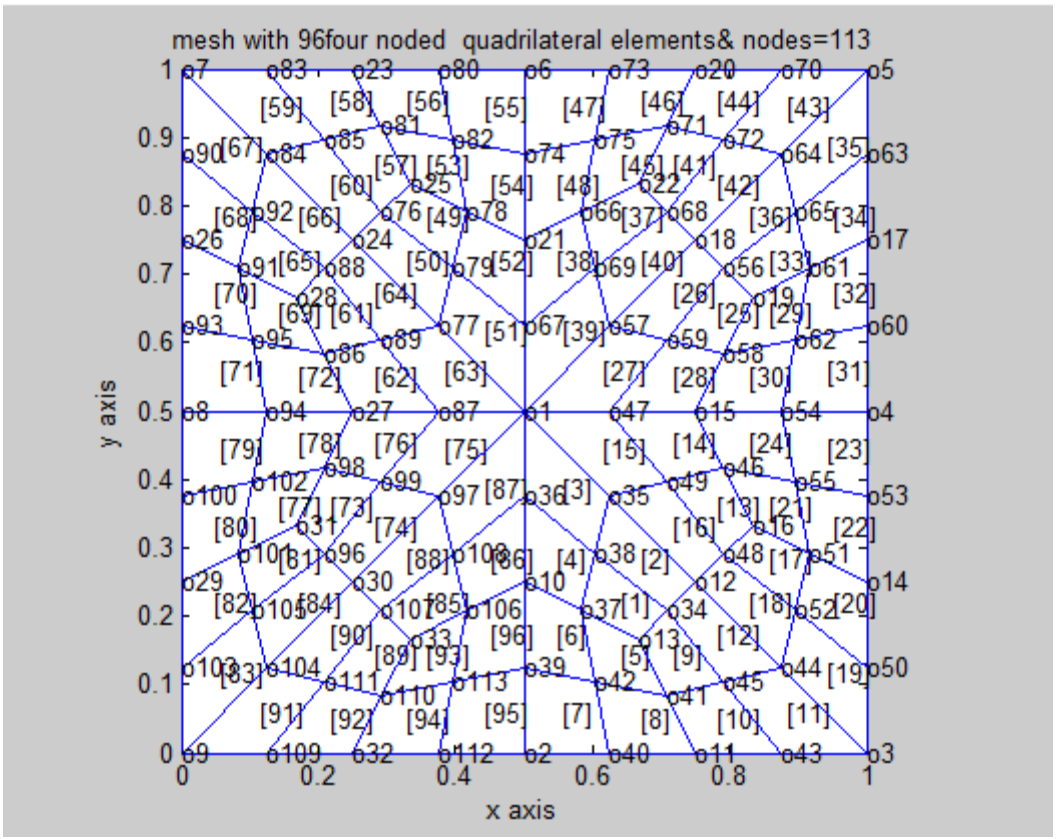


FIRST REFINEMENT

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

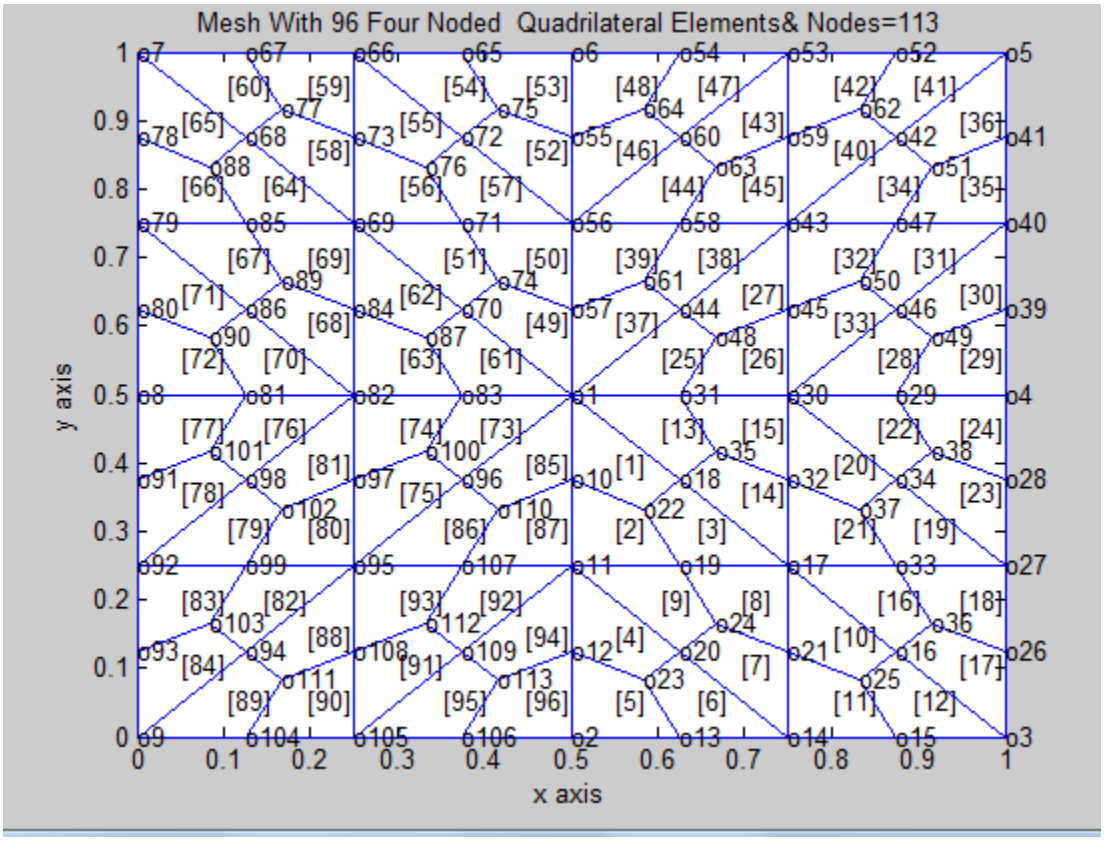


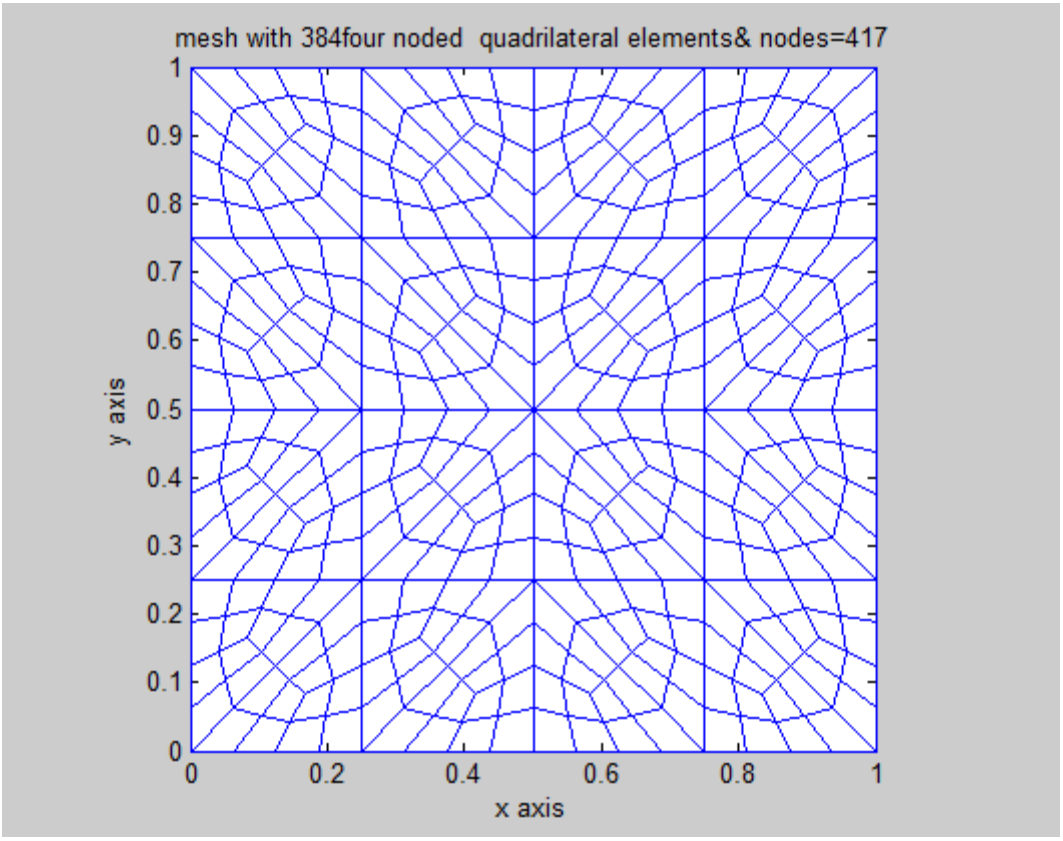


FIRST REFINEMENT

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

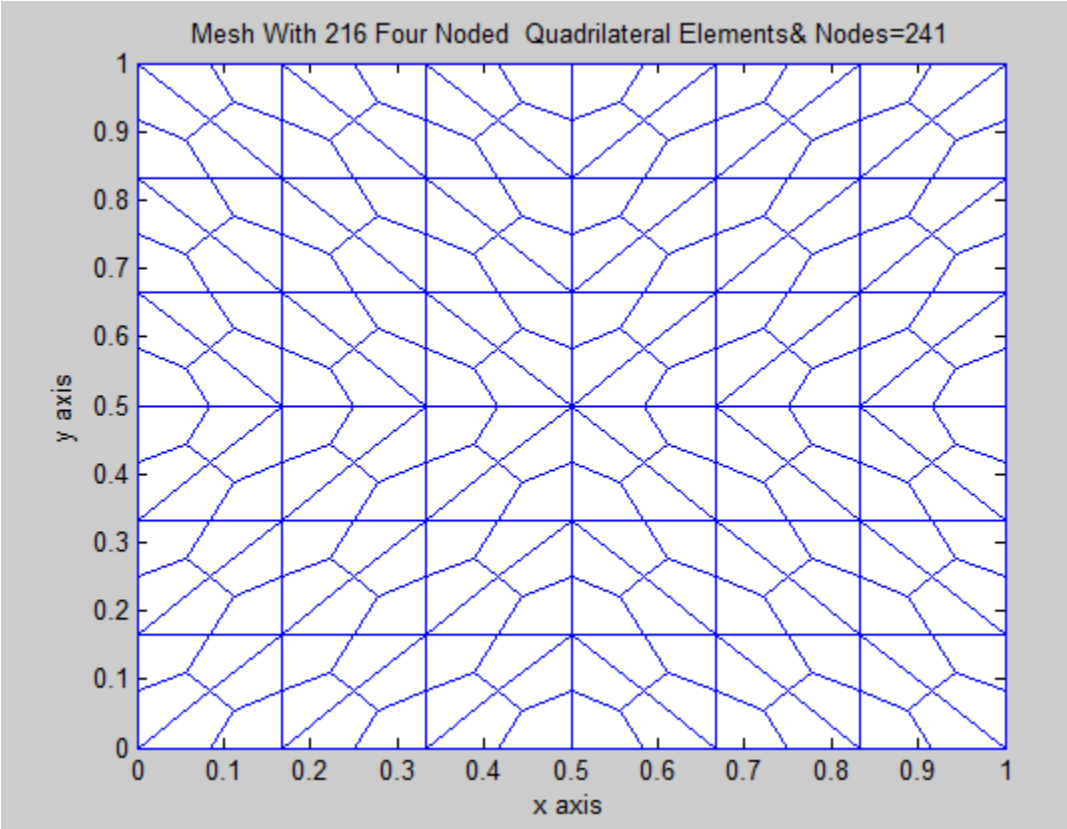


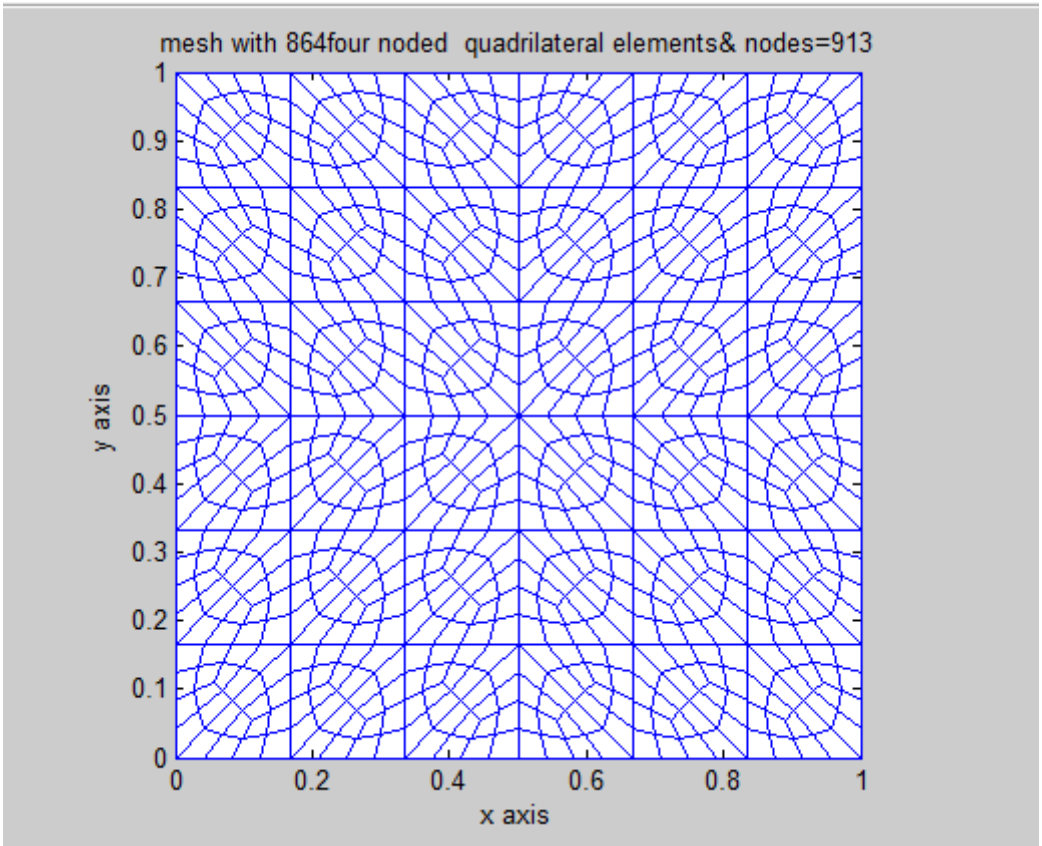


FIRST REFINEMENT

***ORIGINAL**==========

DOI: 10.18535/ijecs/v5i7.10

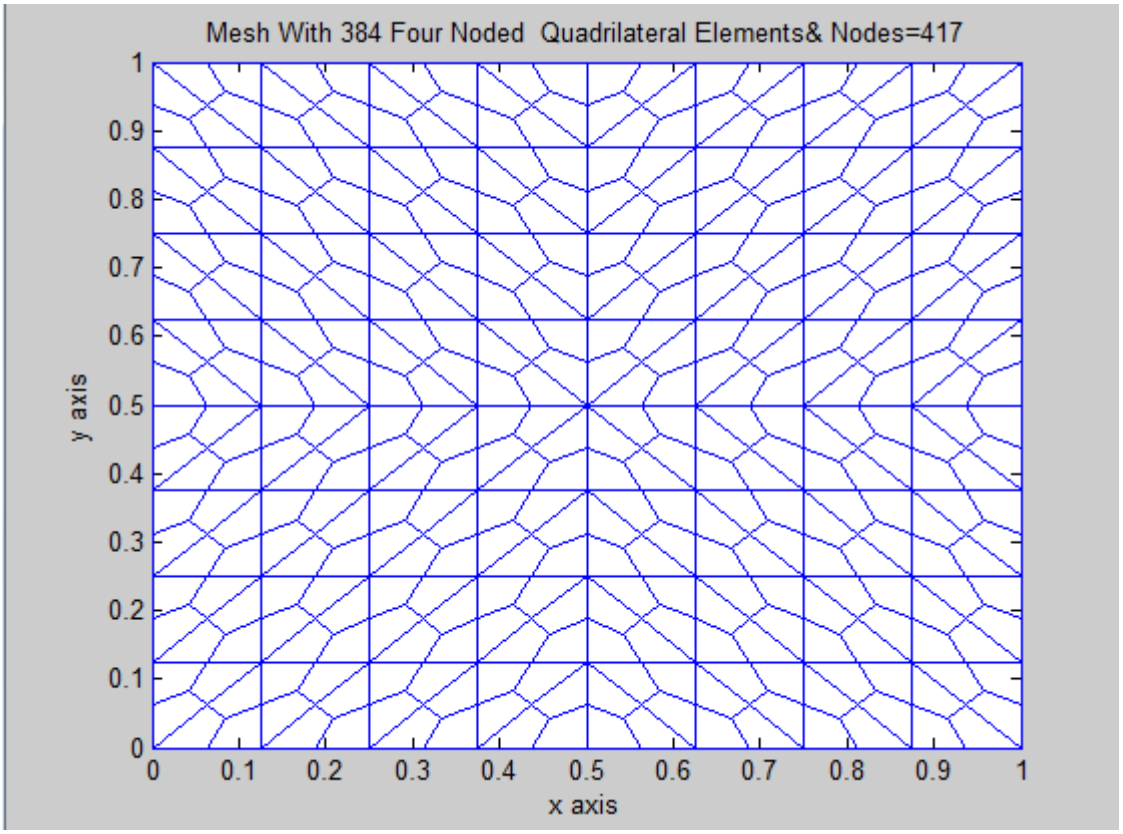


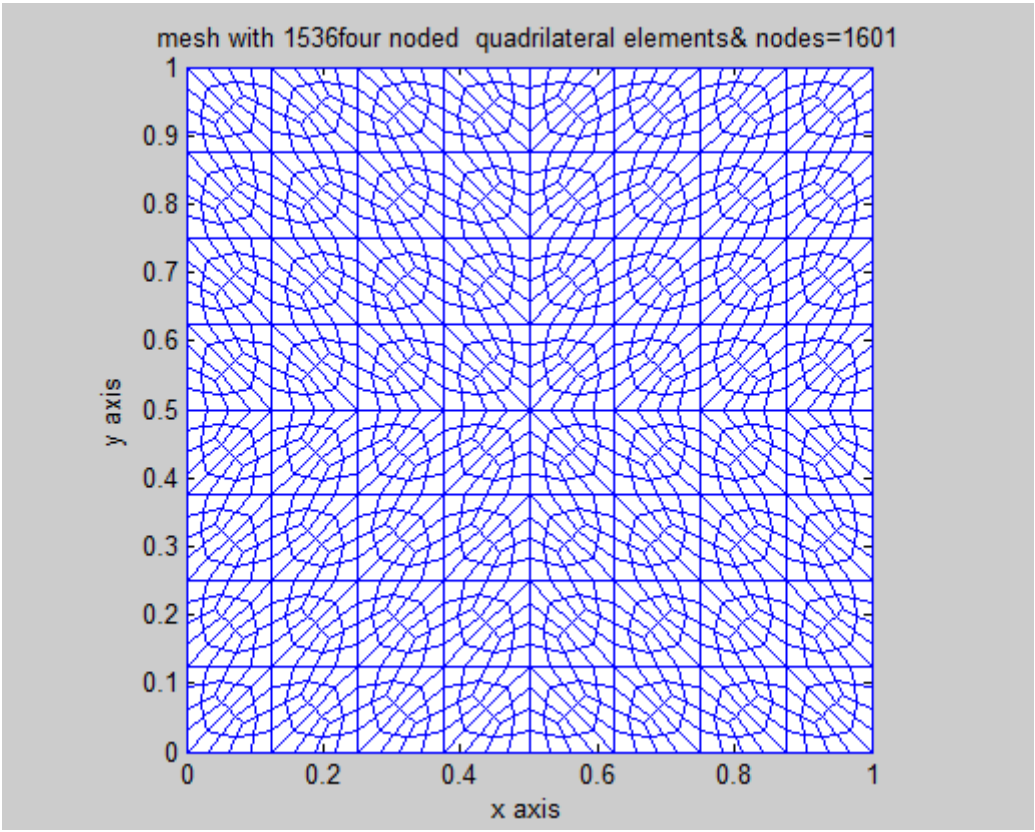


FIRST REFINEMENT

***ORIGINAL** ==========

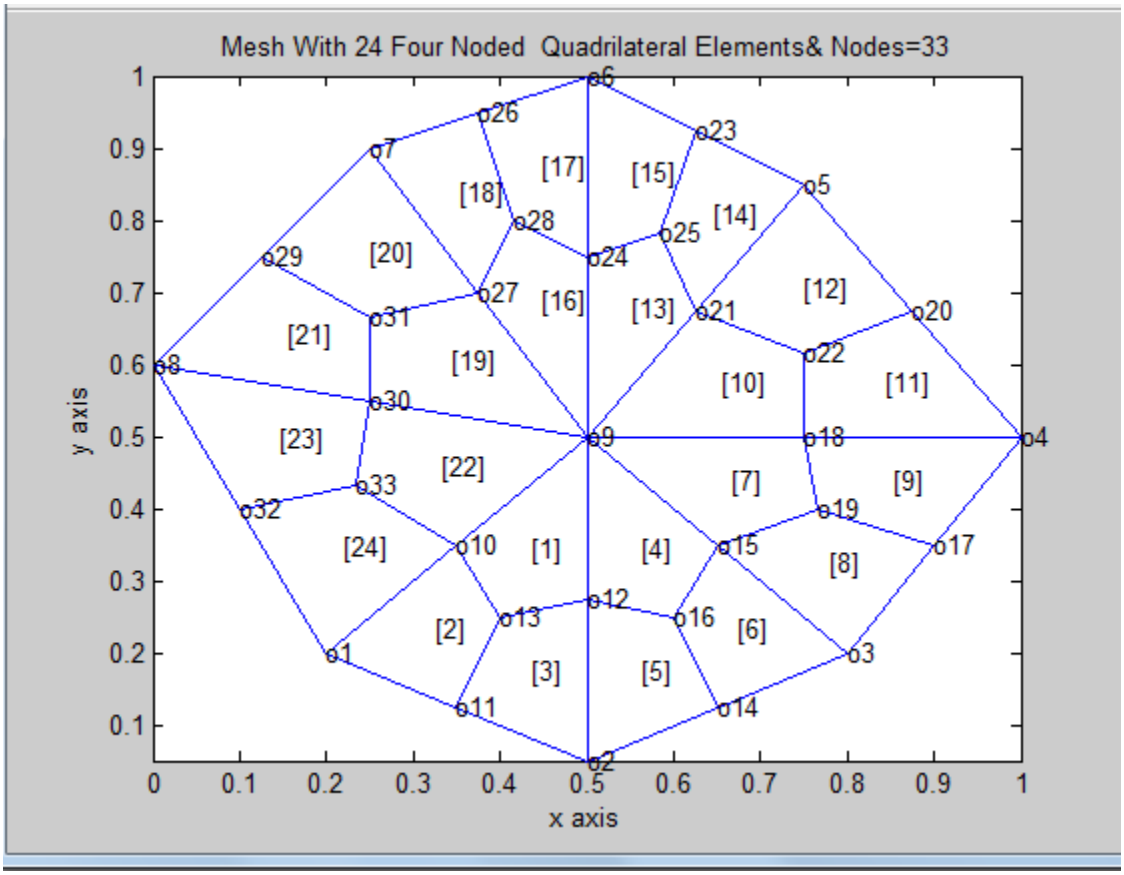
DOI: 10.18535/ijecs/v5i7.10

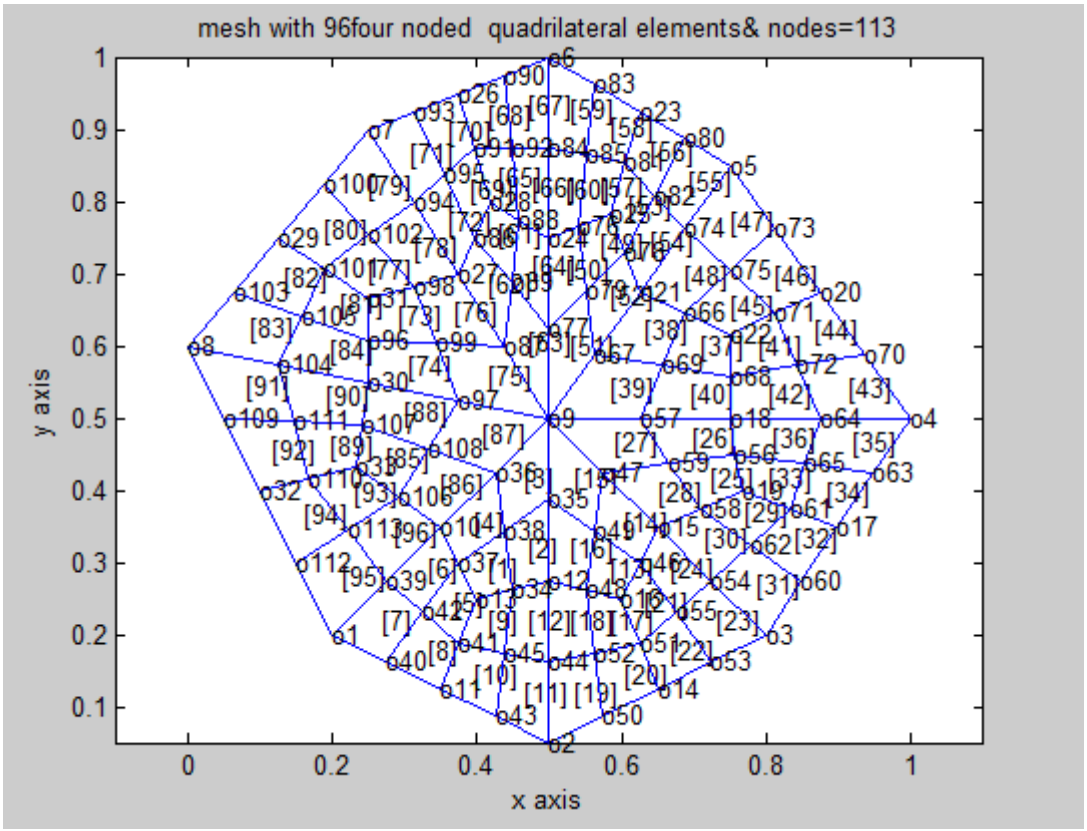




*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

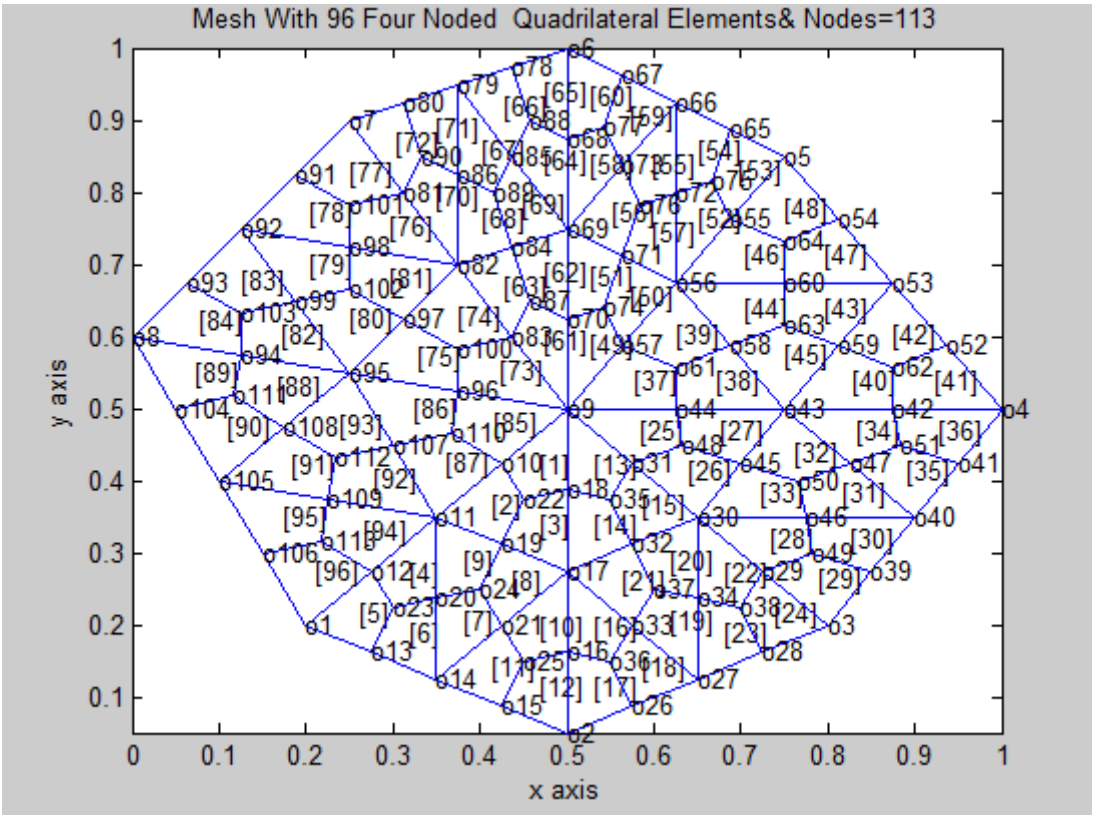


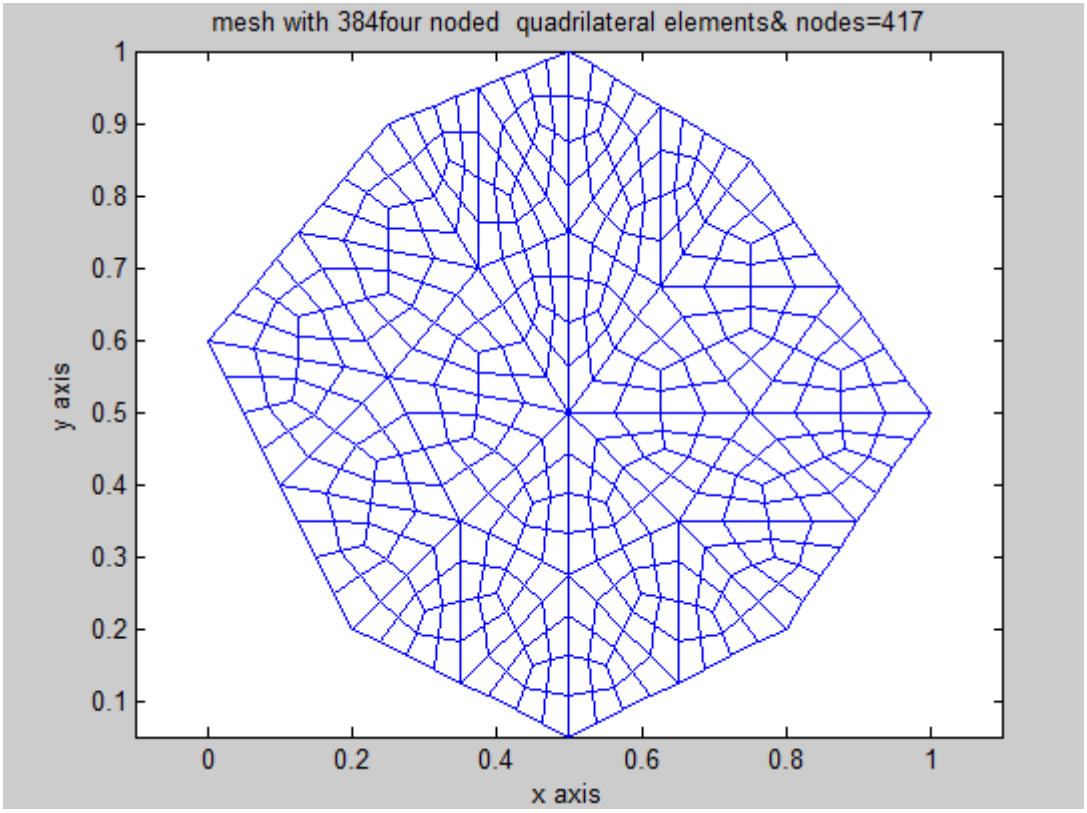


FIRST REFINEMENT

***ORIGINAL**==========

DOI: 10.18535/ijecs/v5i7.10

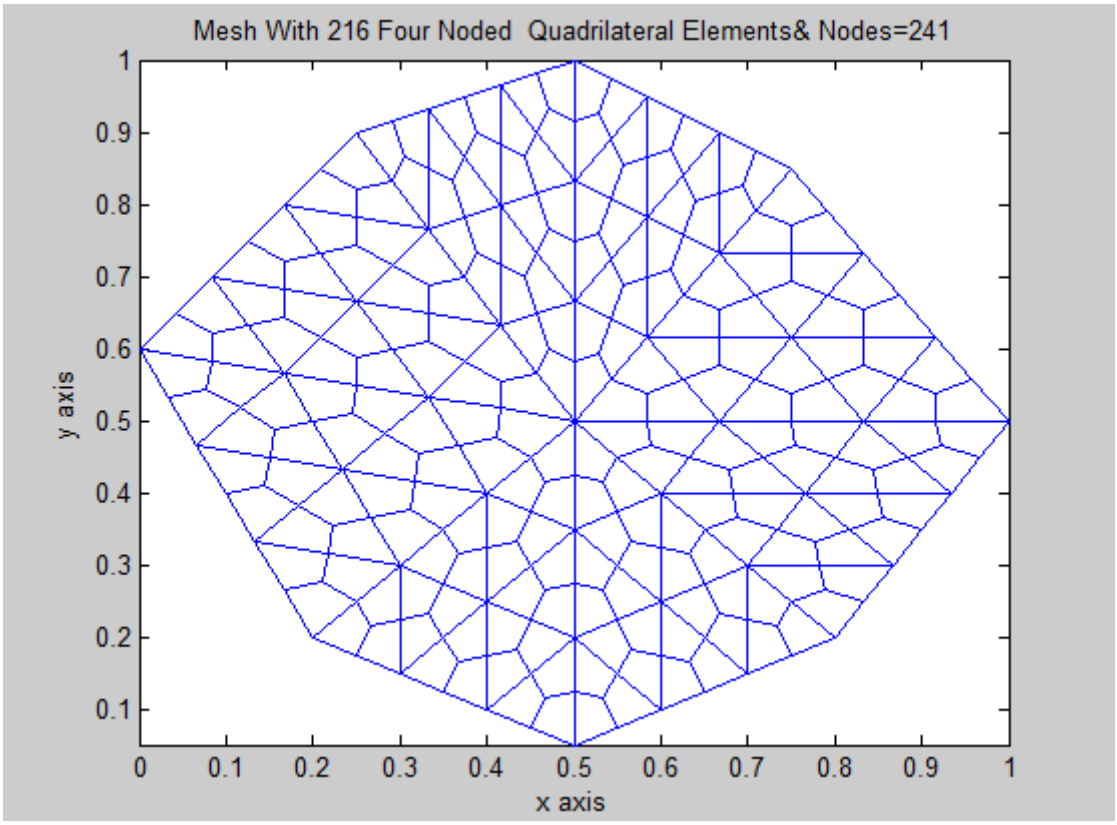


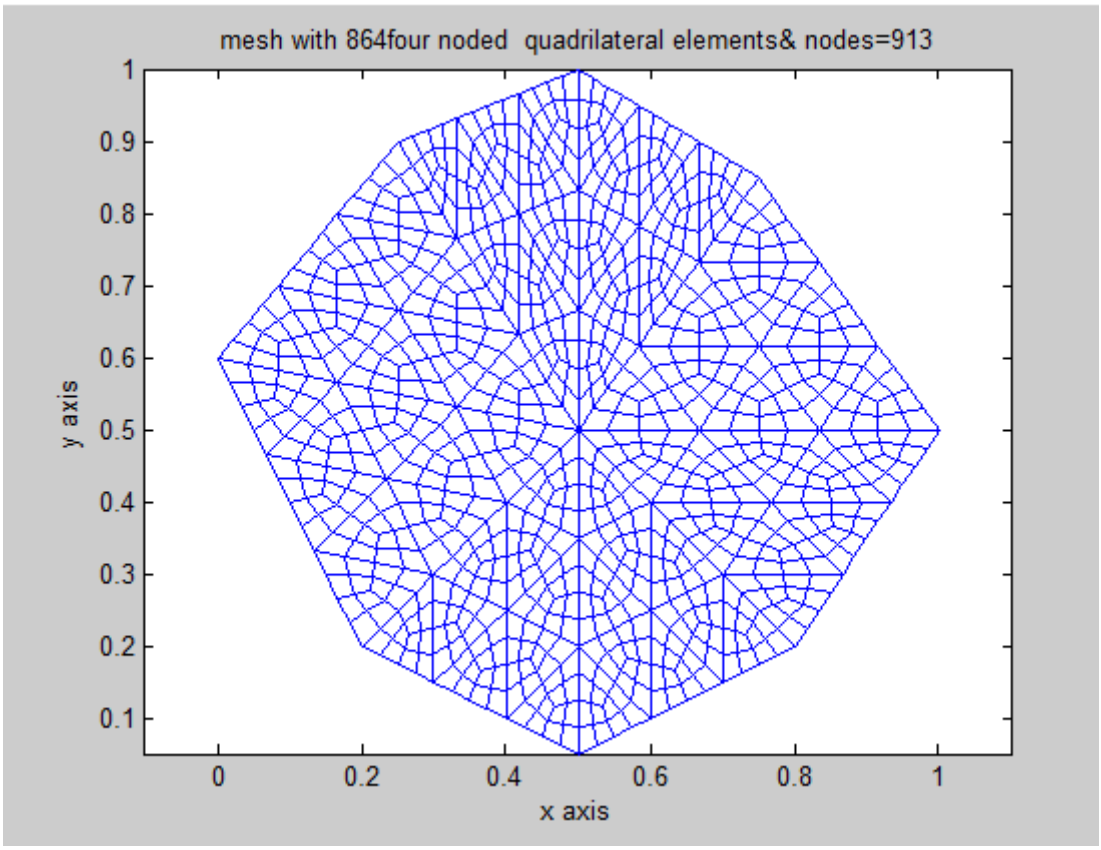


FIRST REFINEMENT

***ORIGINAL==========**

DOI: 10.18535/ijecs/v5i7.10

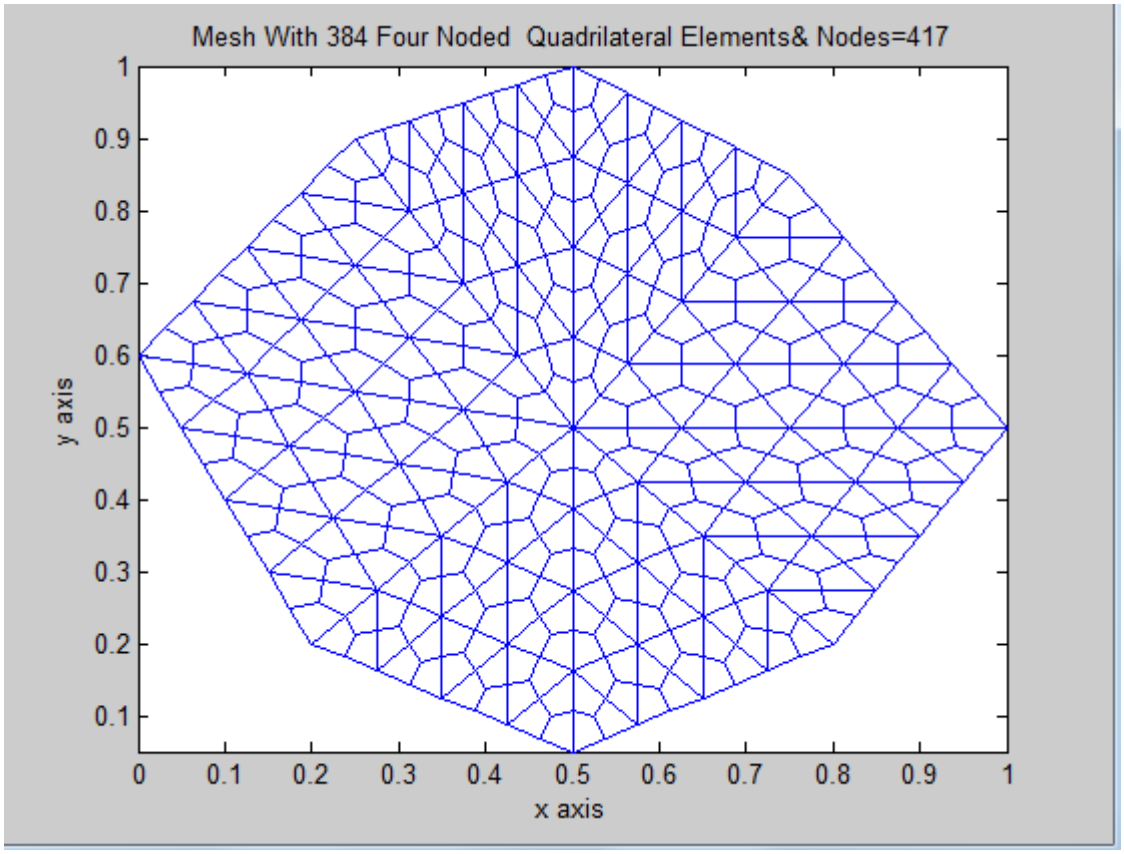


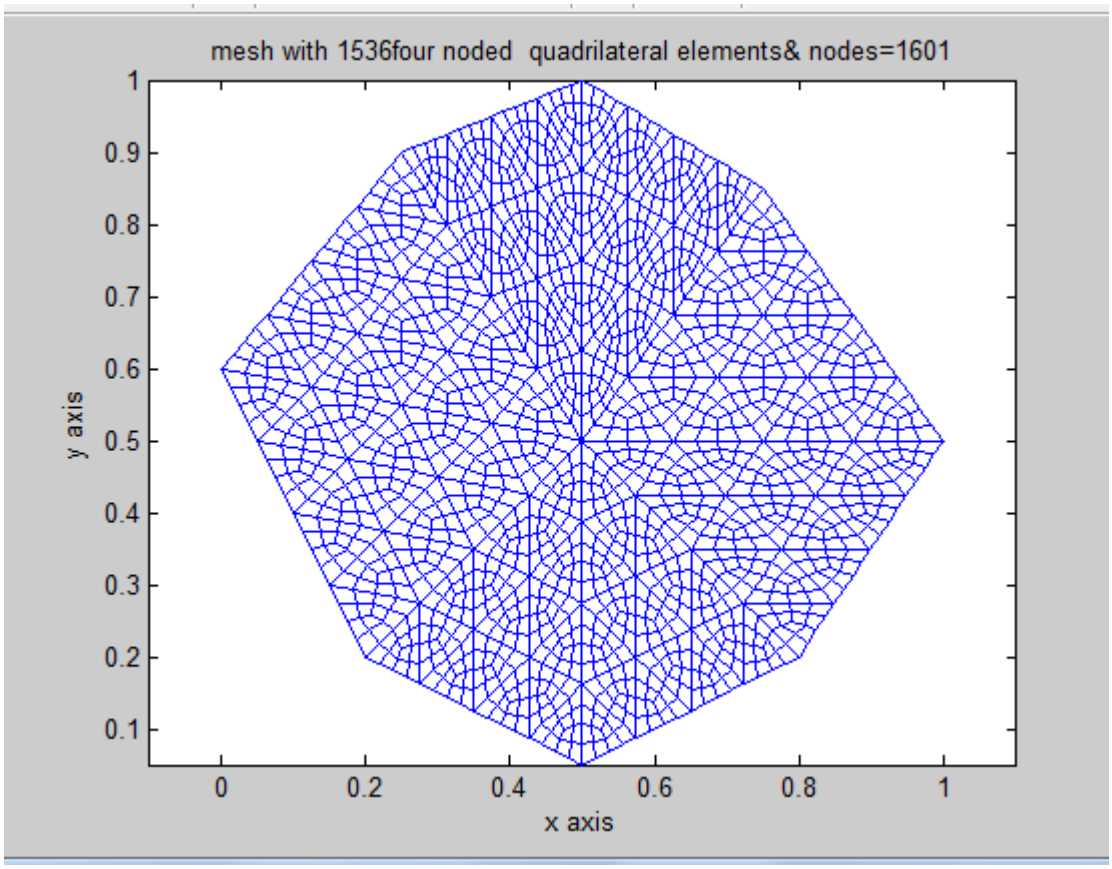


FIRST REFINEMENT

***ORIGINAL**==========

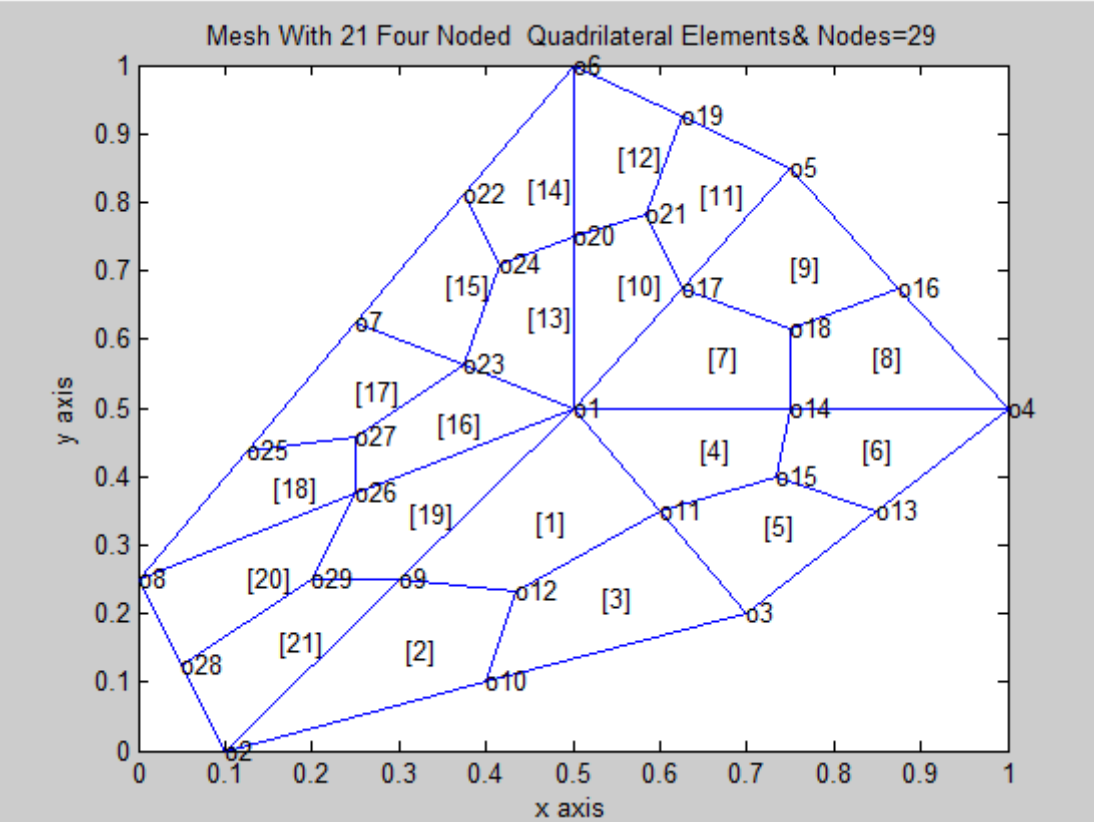
DOI: 10.18535/ijecs/v5i7.10

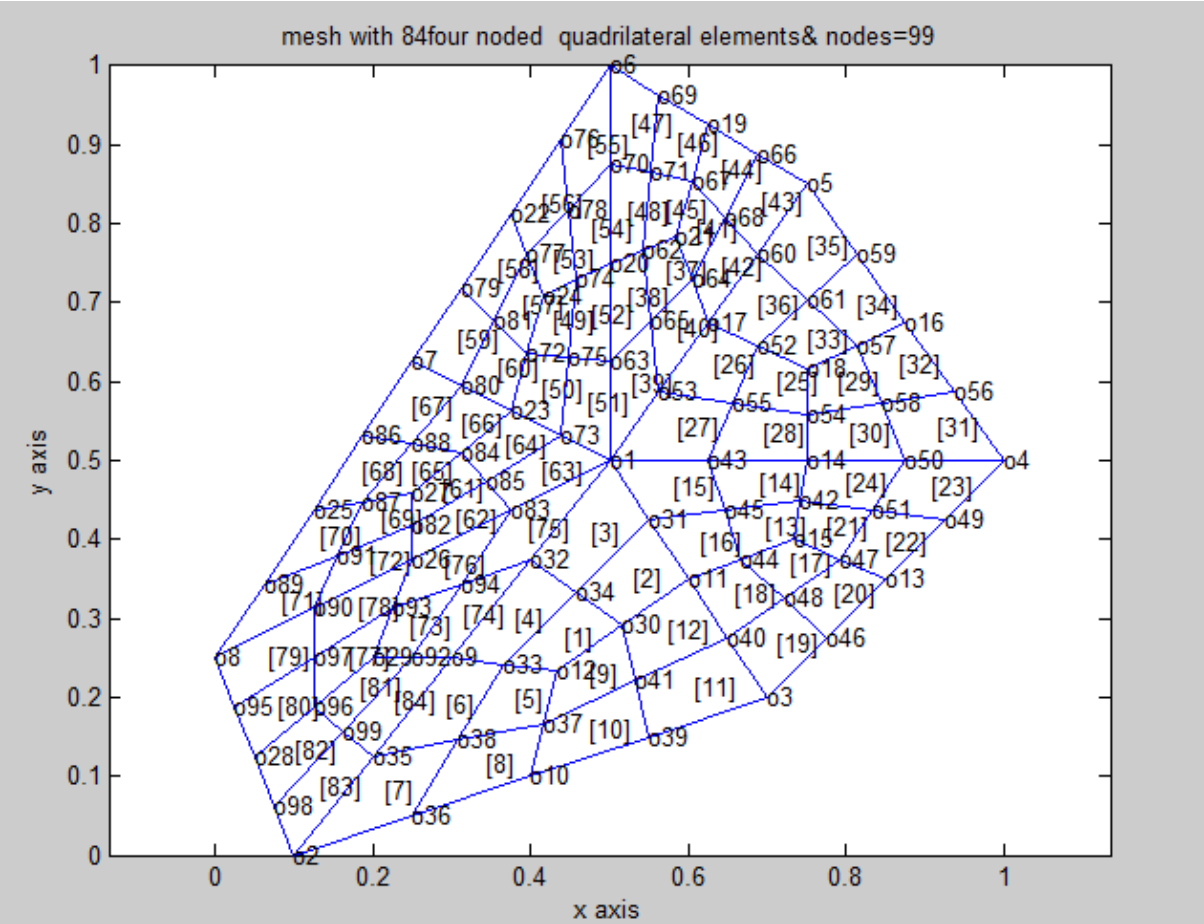




*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

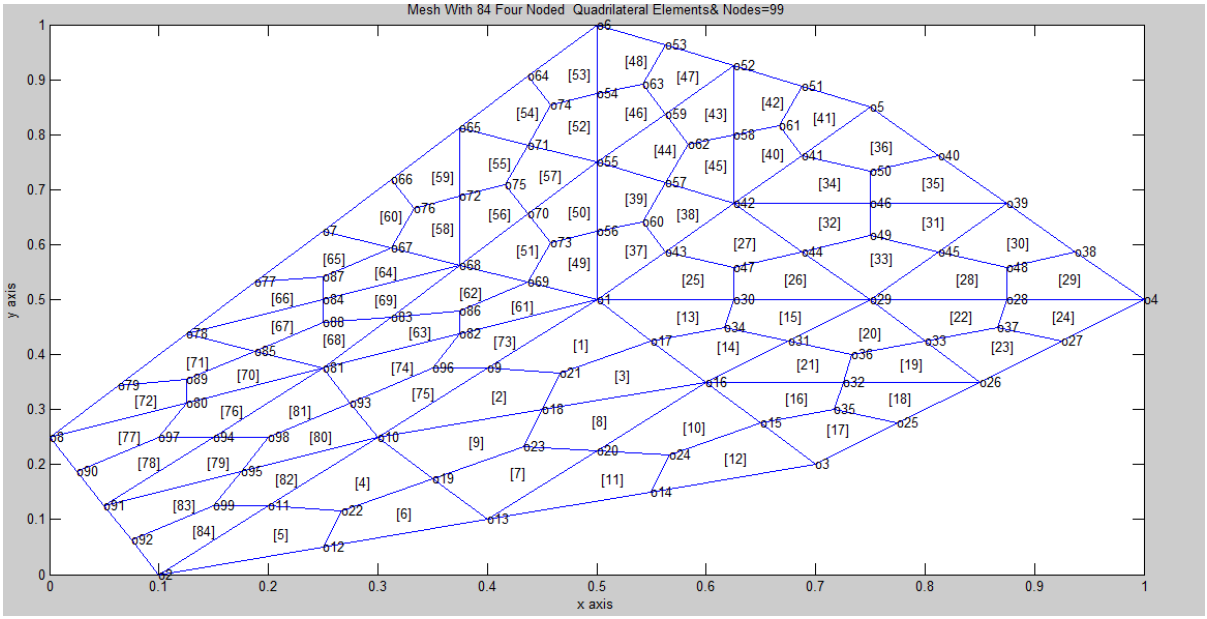


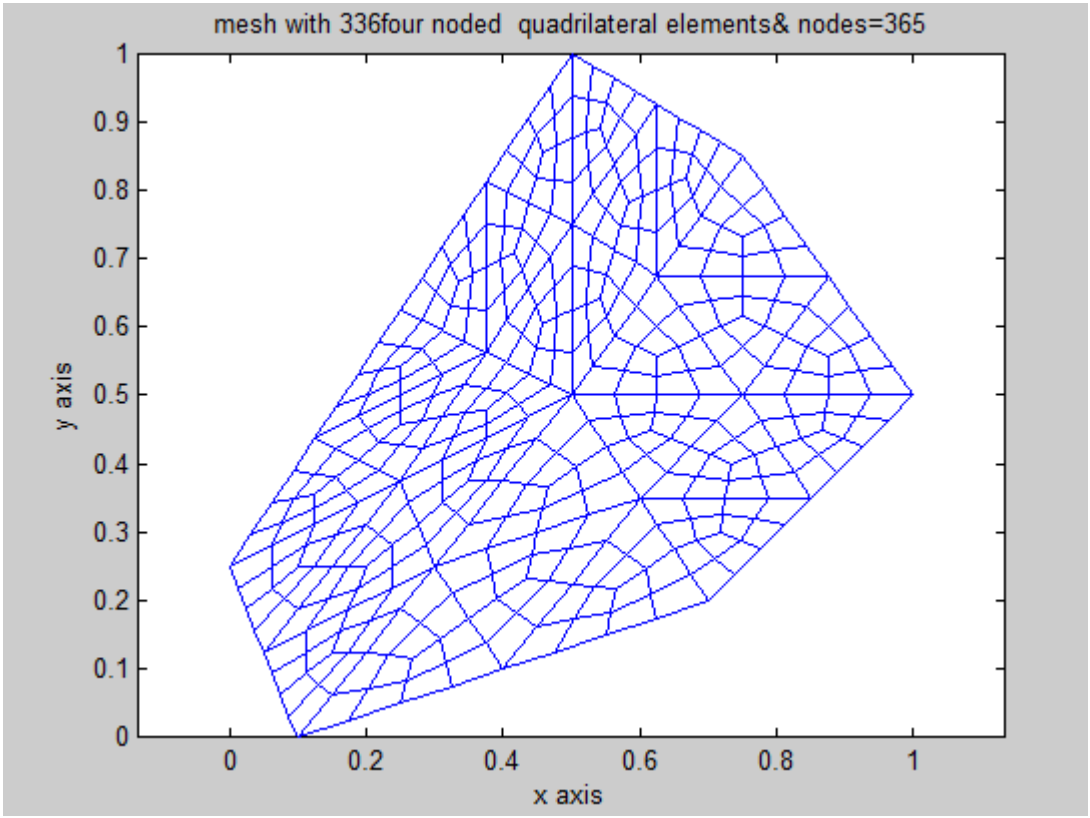


FIRST REINEMENT

*ORIGNAL==========

DOI: 10.18535/ijecs/v5i7.10

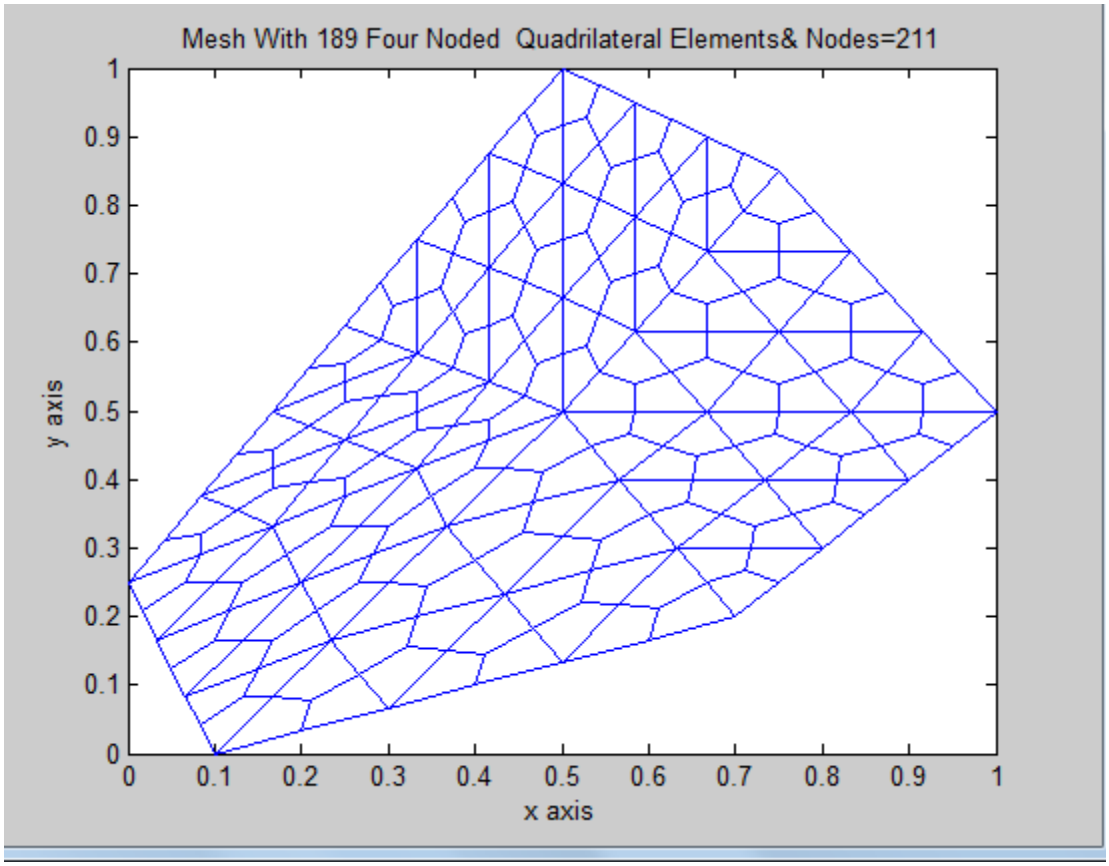


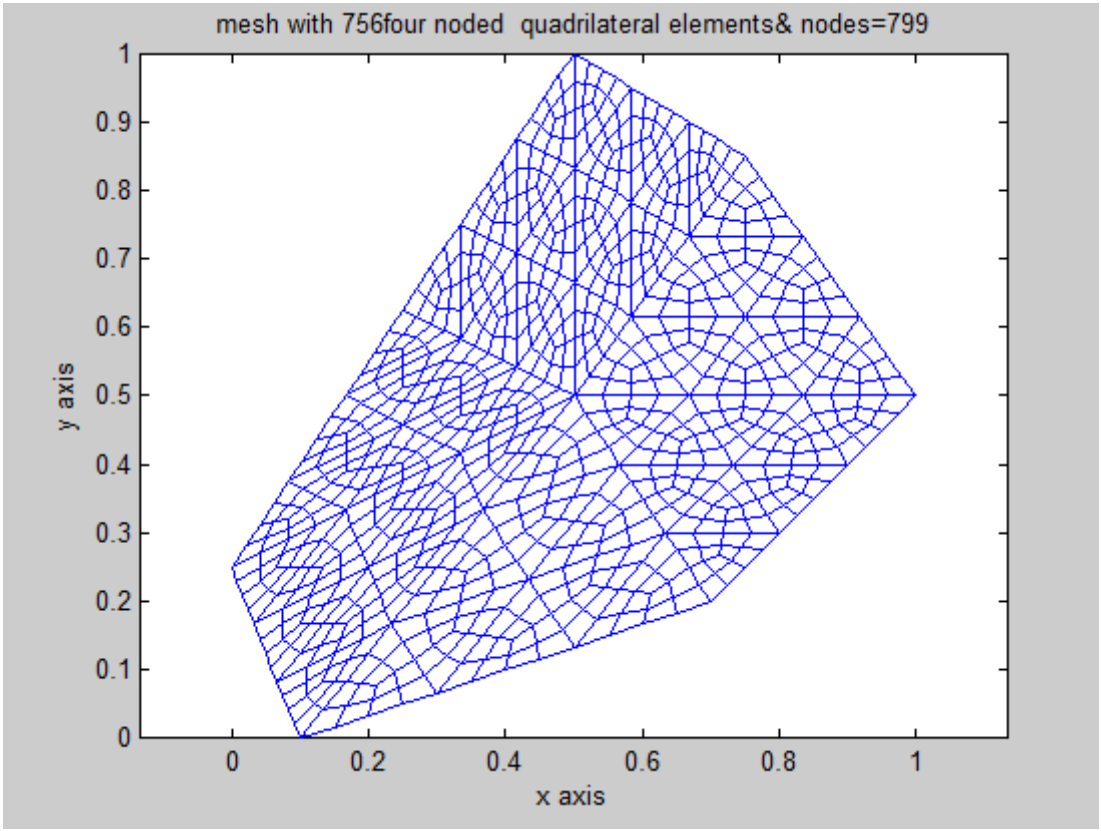


FIRST REFINEMENT

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10

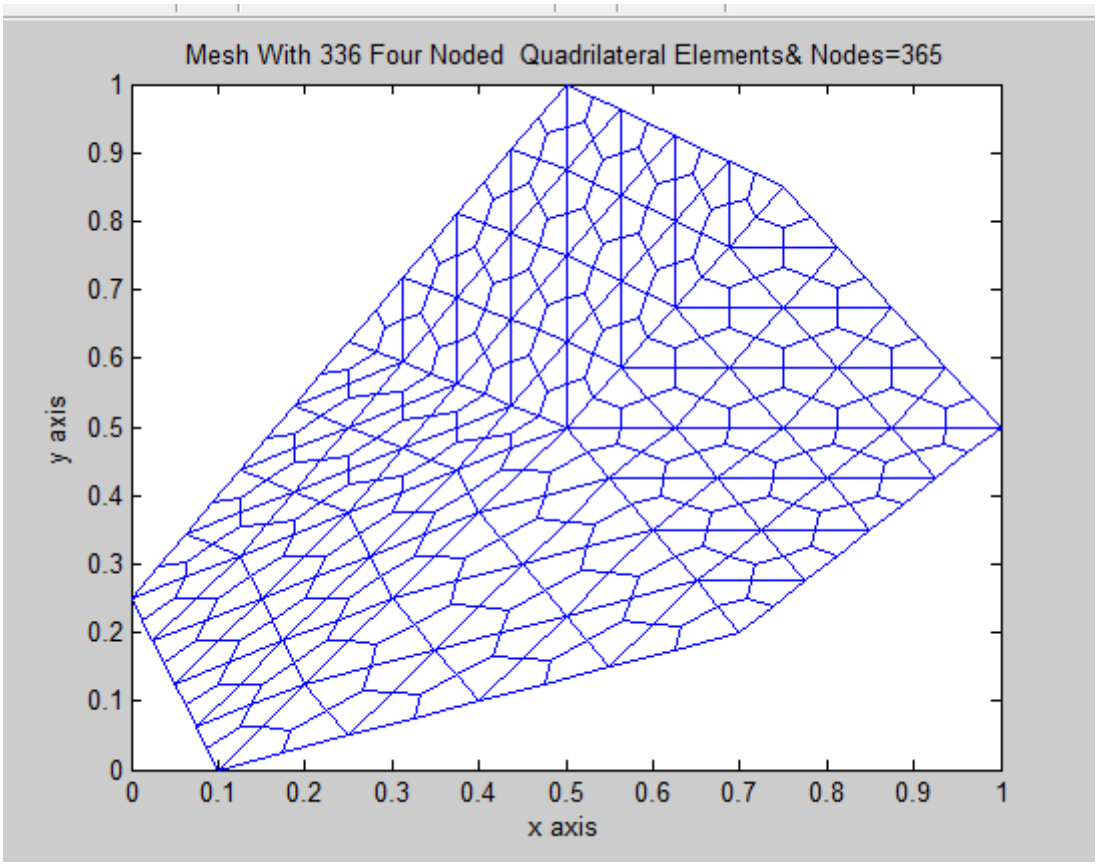


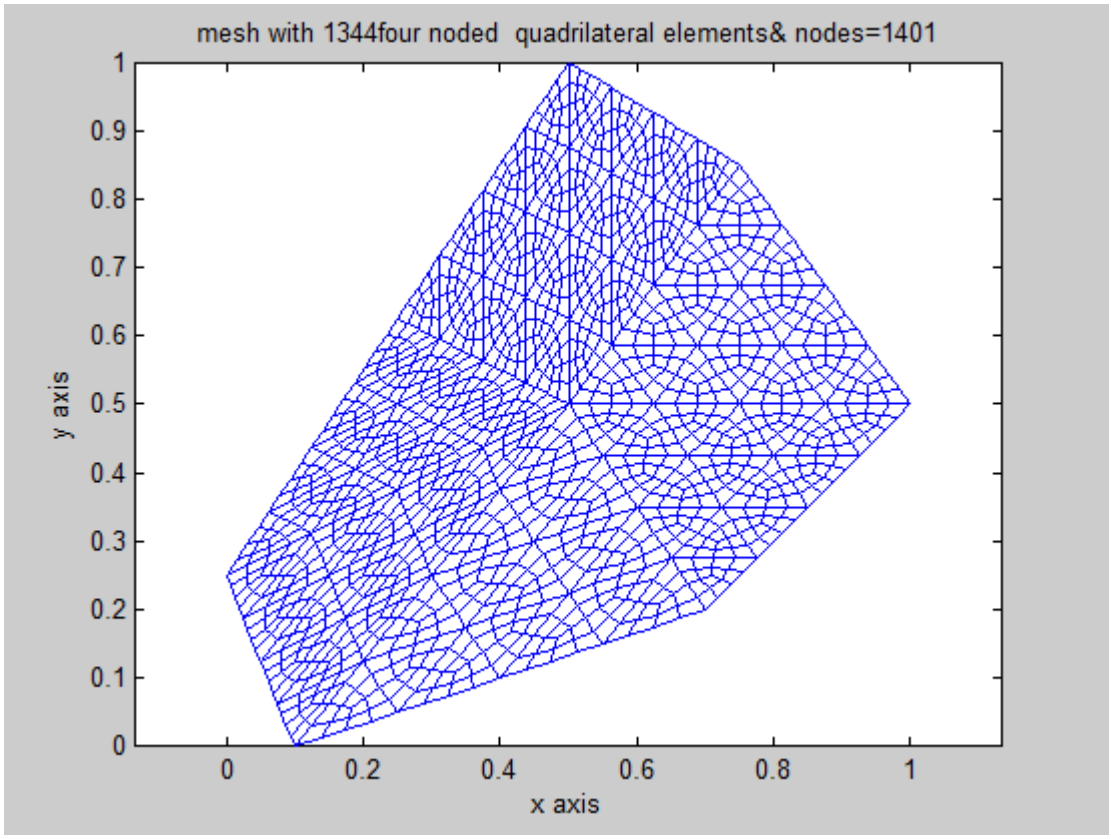


FIRST REFINEMENT

*ORIGINAL==========

DOI: 10.18535/ijecs/v5i7.10



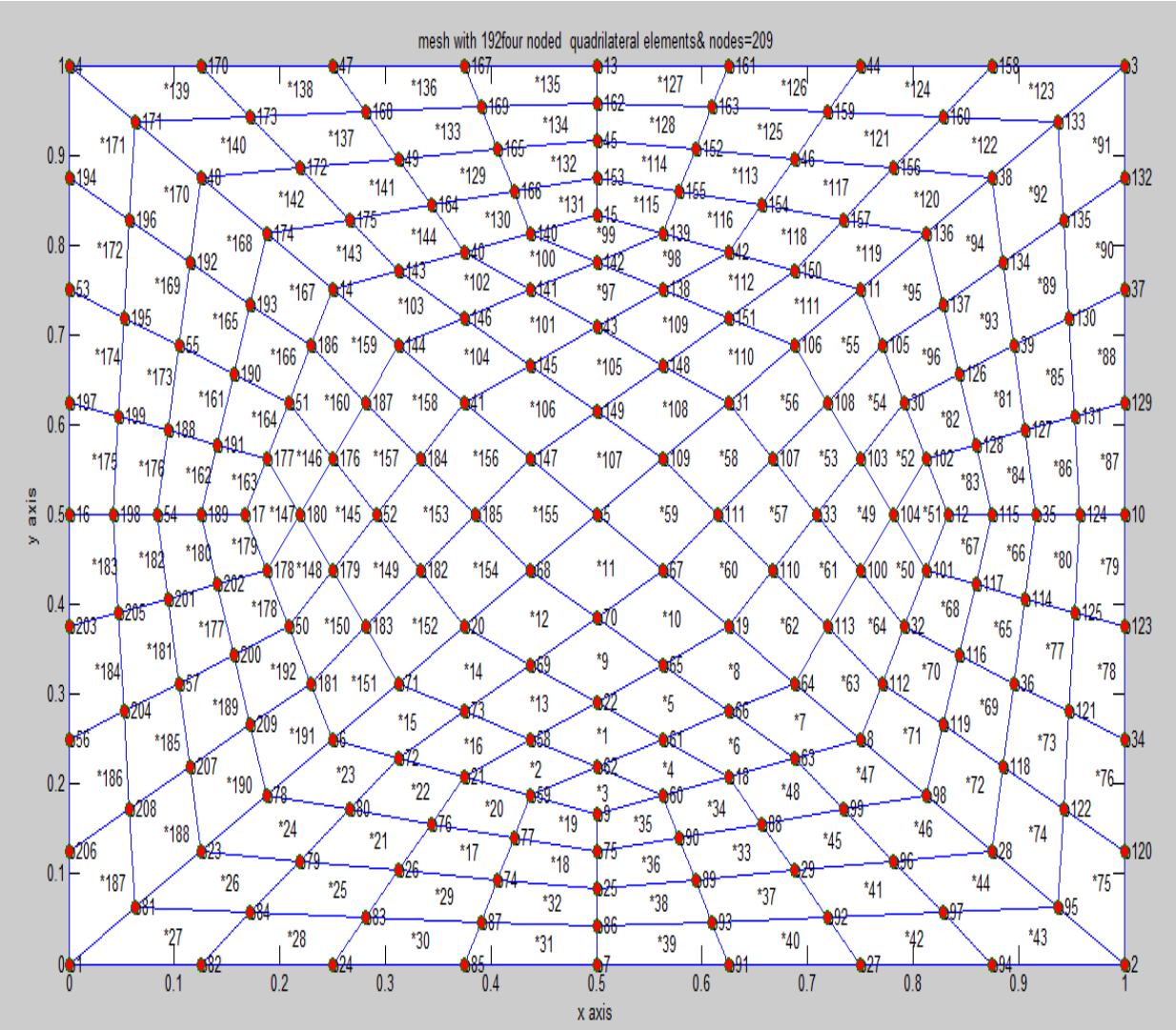


FIRST REFINEMENT

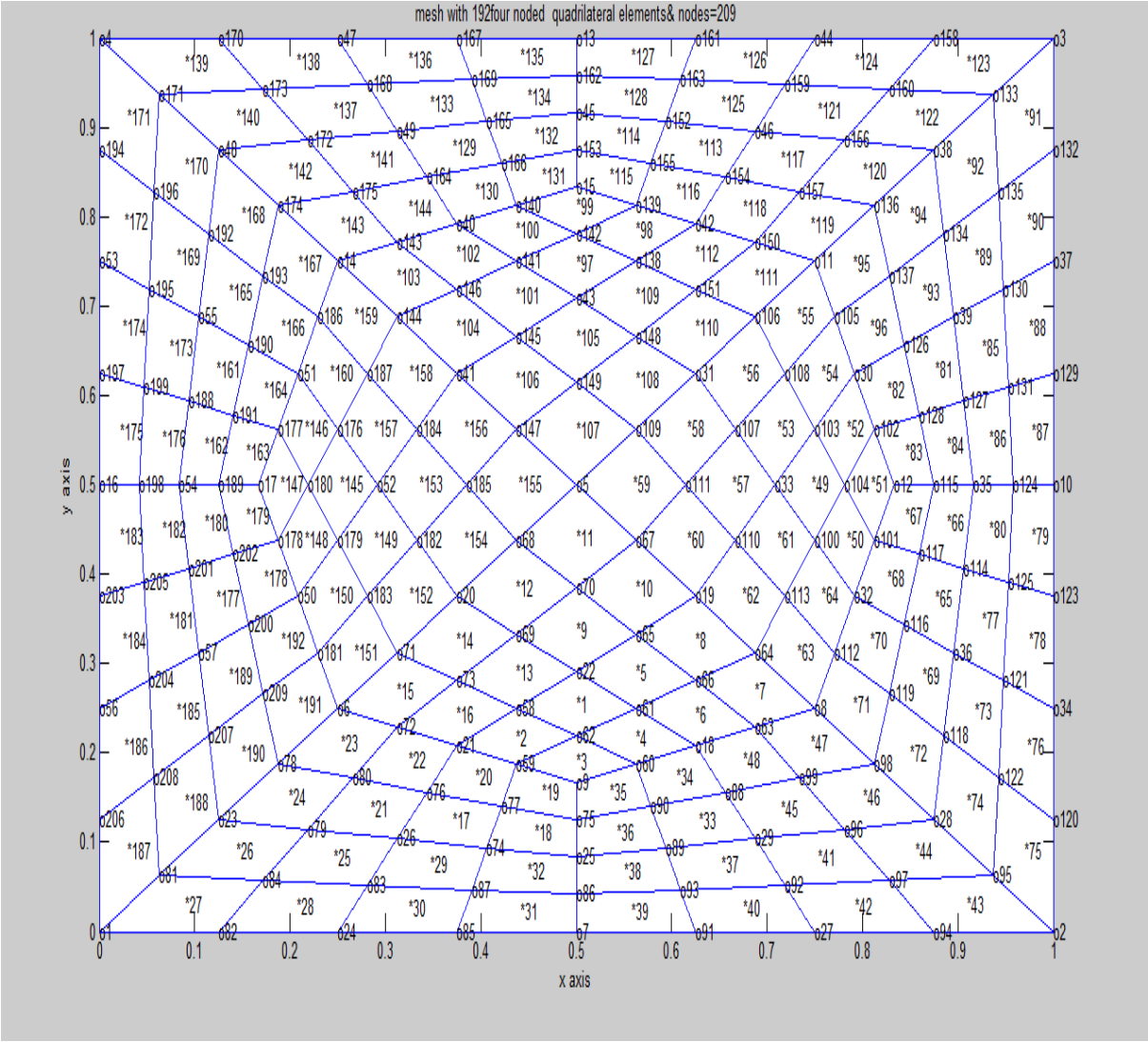
*==========

DOI: 10.18535/ijecs/v5i7.10

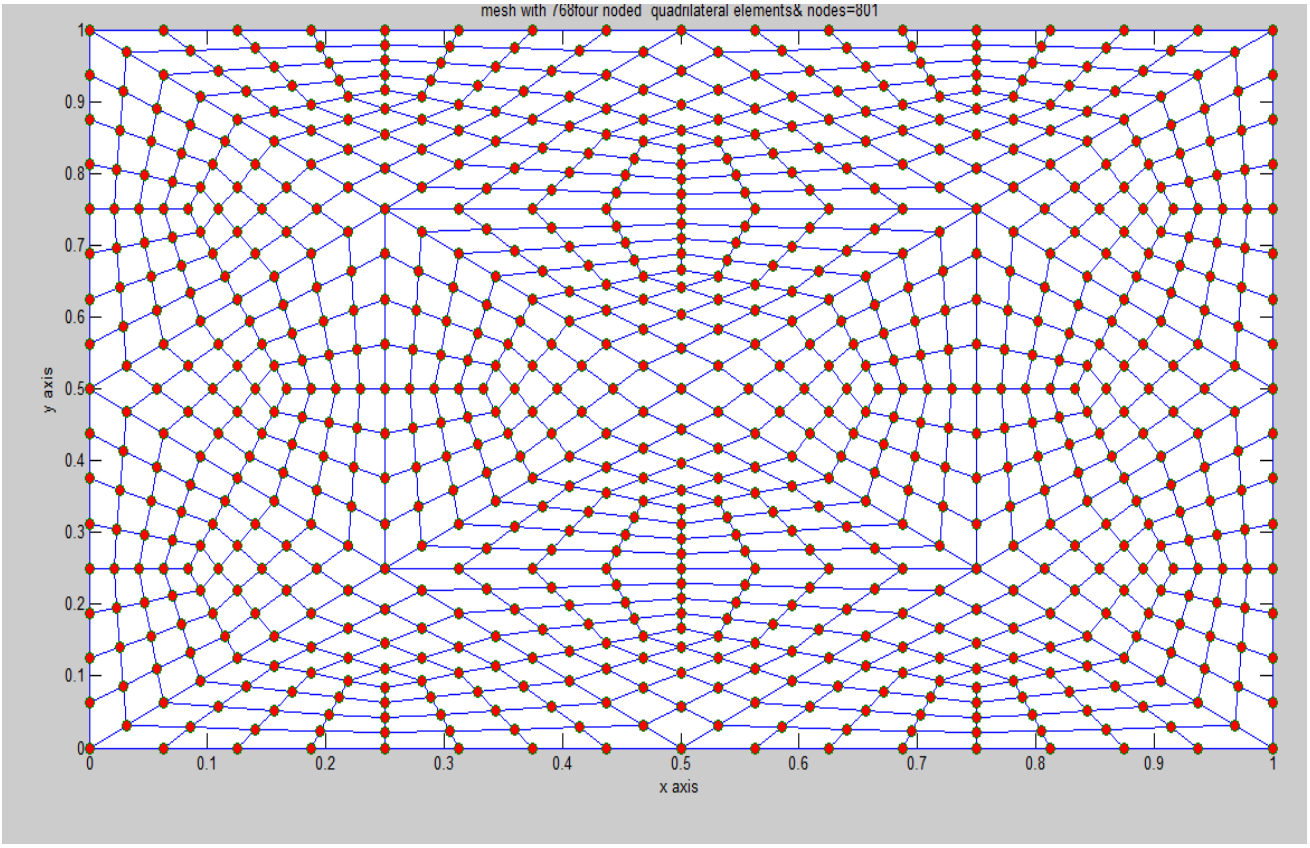
SOME FINITE ELEMENT MESHES FOR SECOND REFINEMENTS

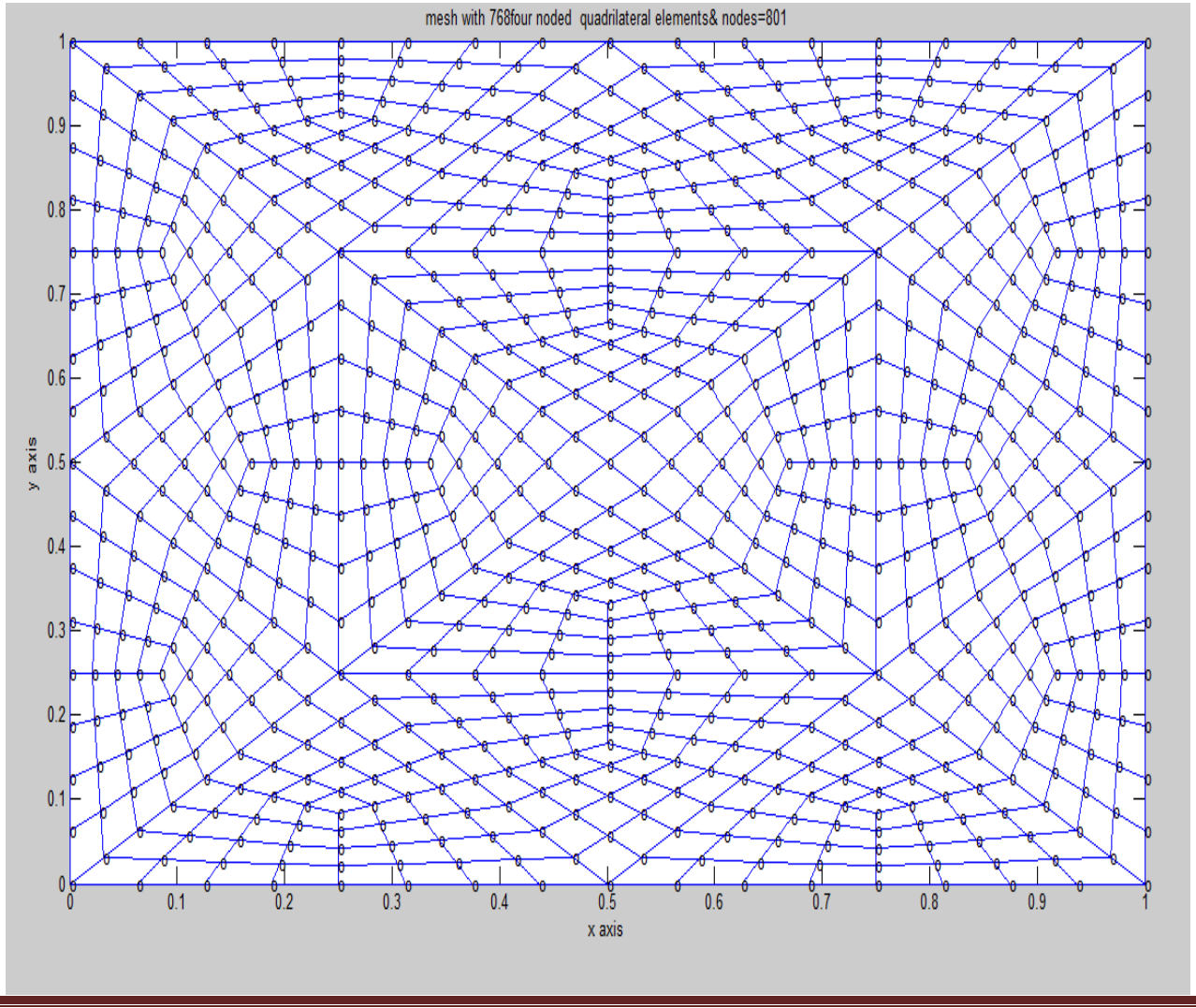


DOI: 10.18535/ijecs/v5i7.10

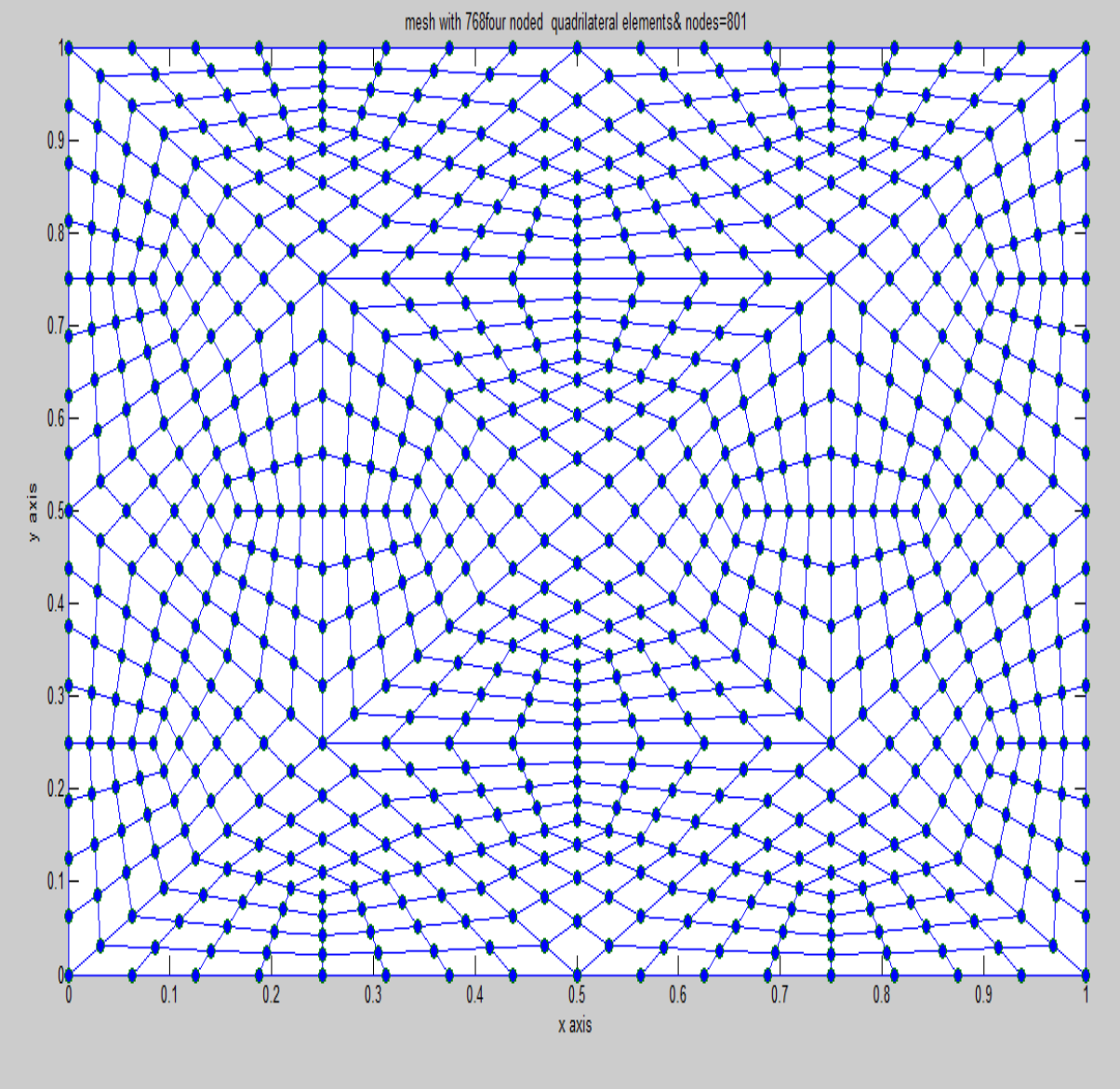


DOI: 10.18535/ijecs/v5i7.10

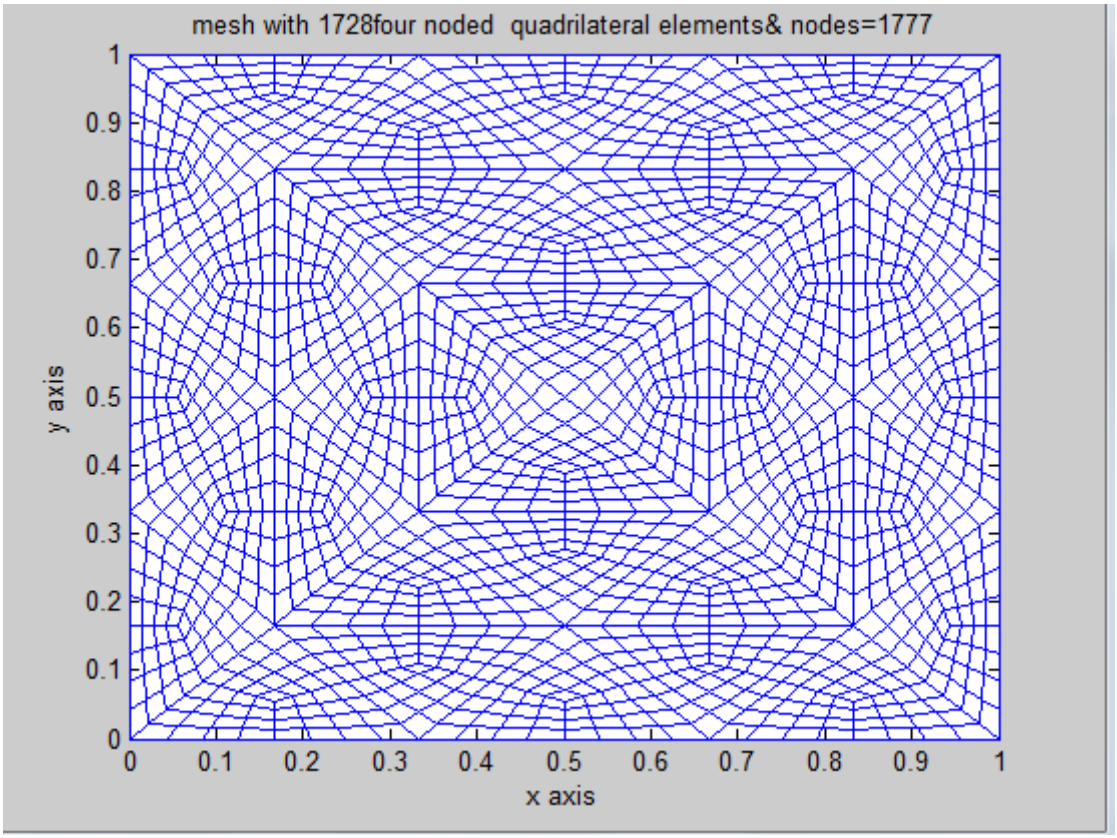


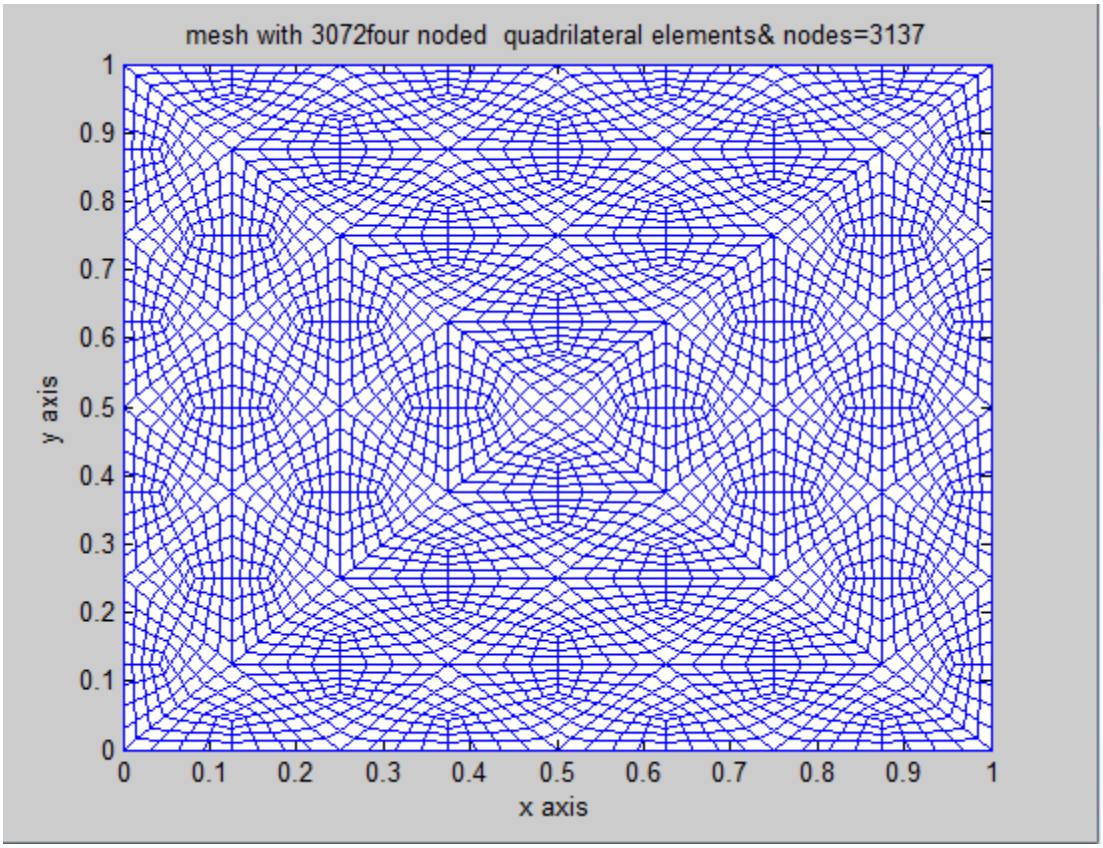


DOI: 10.18535/ijecs/v5i7.10

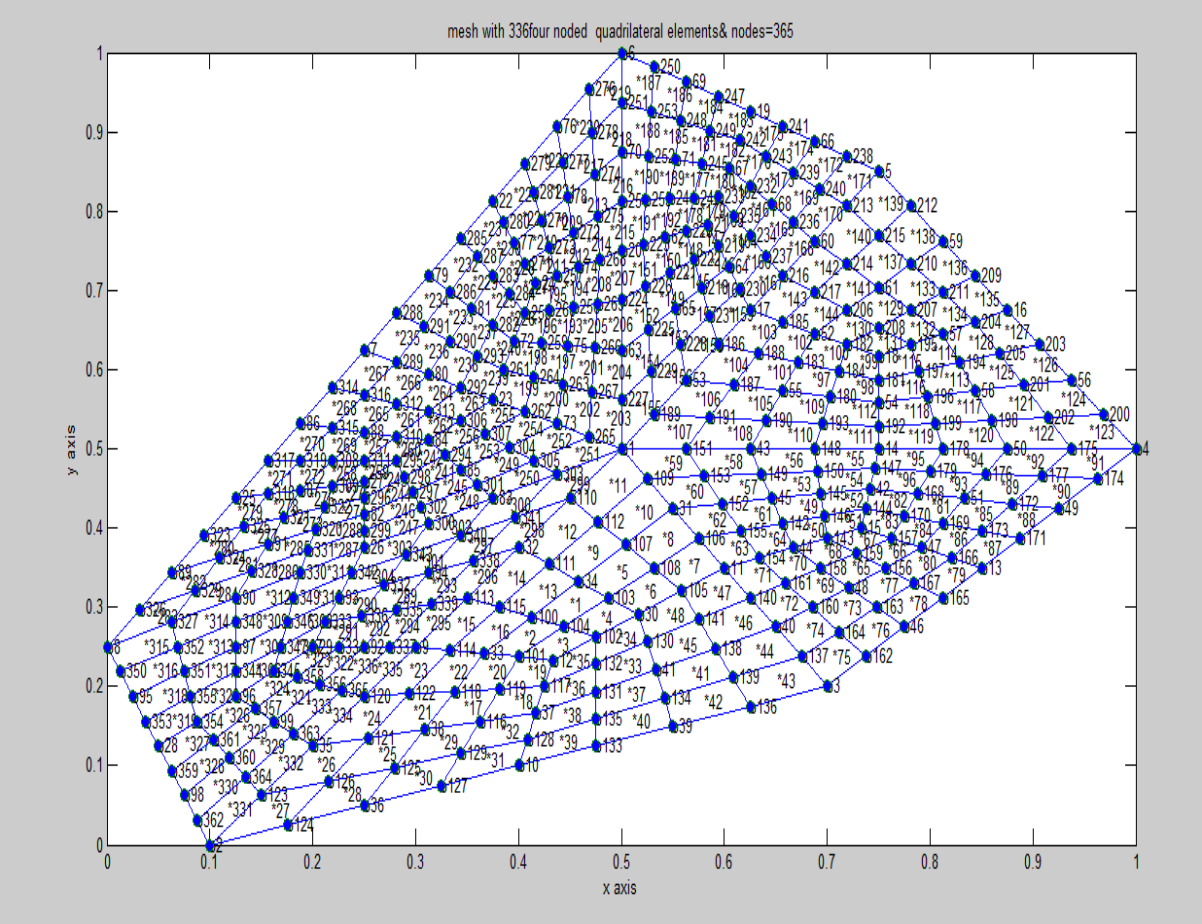


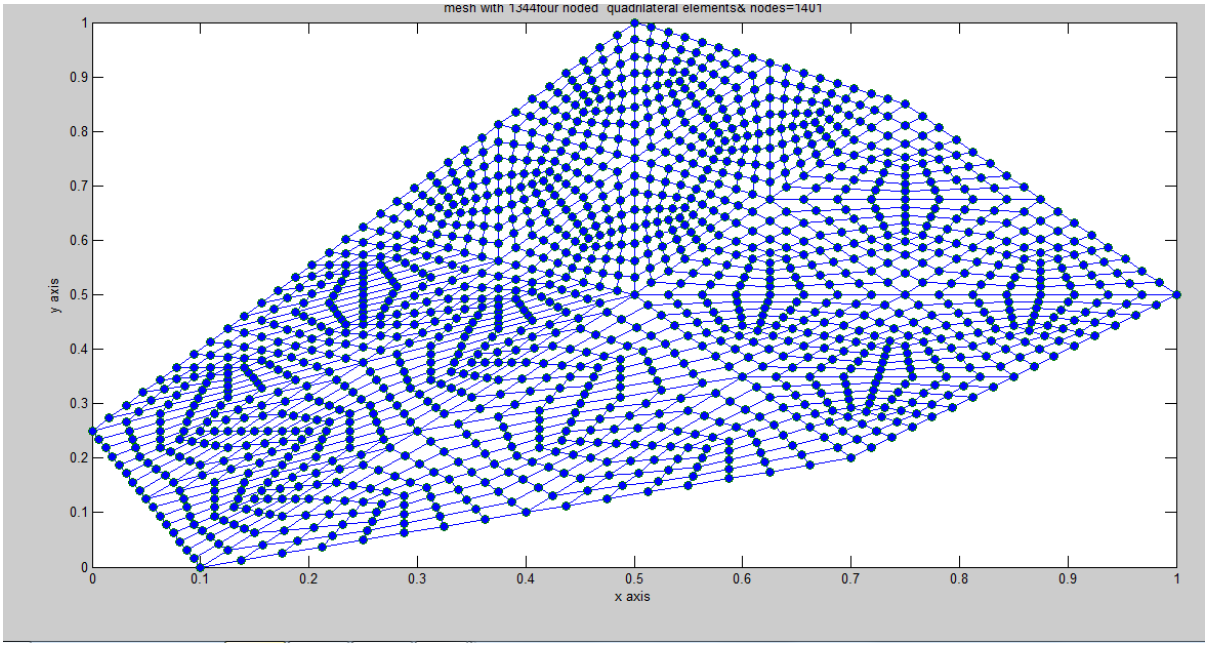
DOI: 10.18535/ijecs/v5i7.10



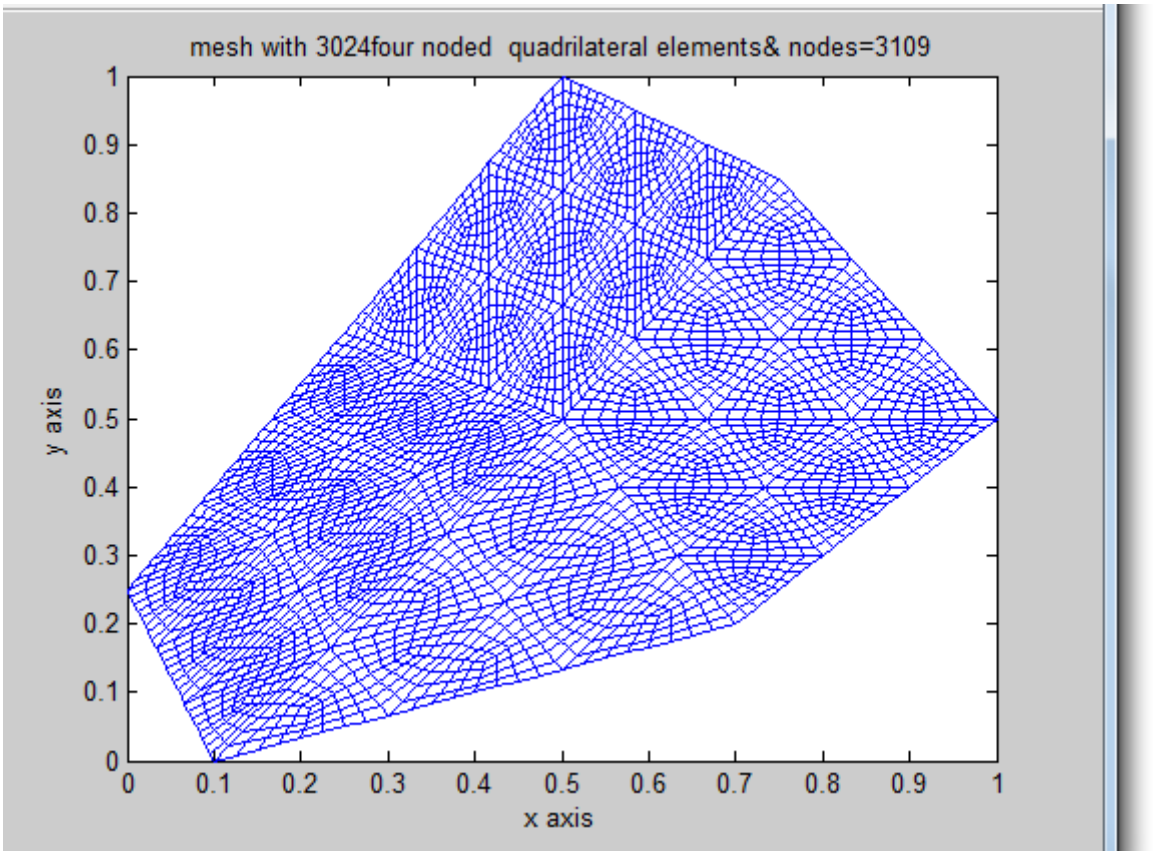


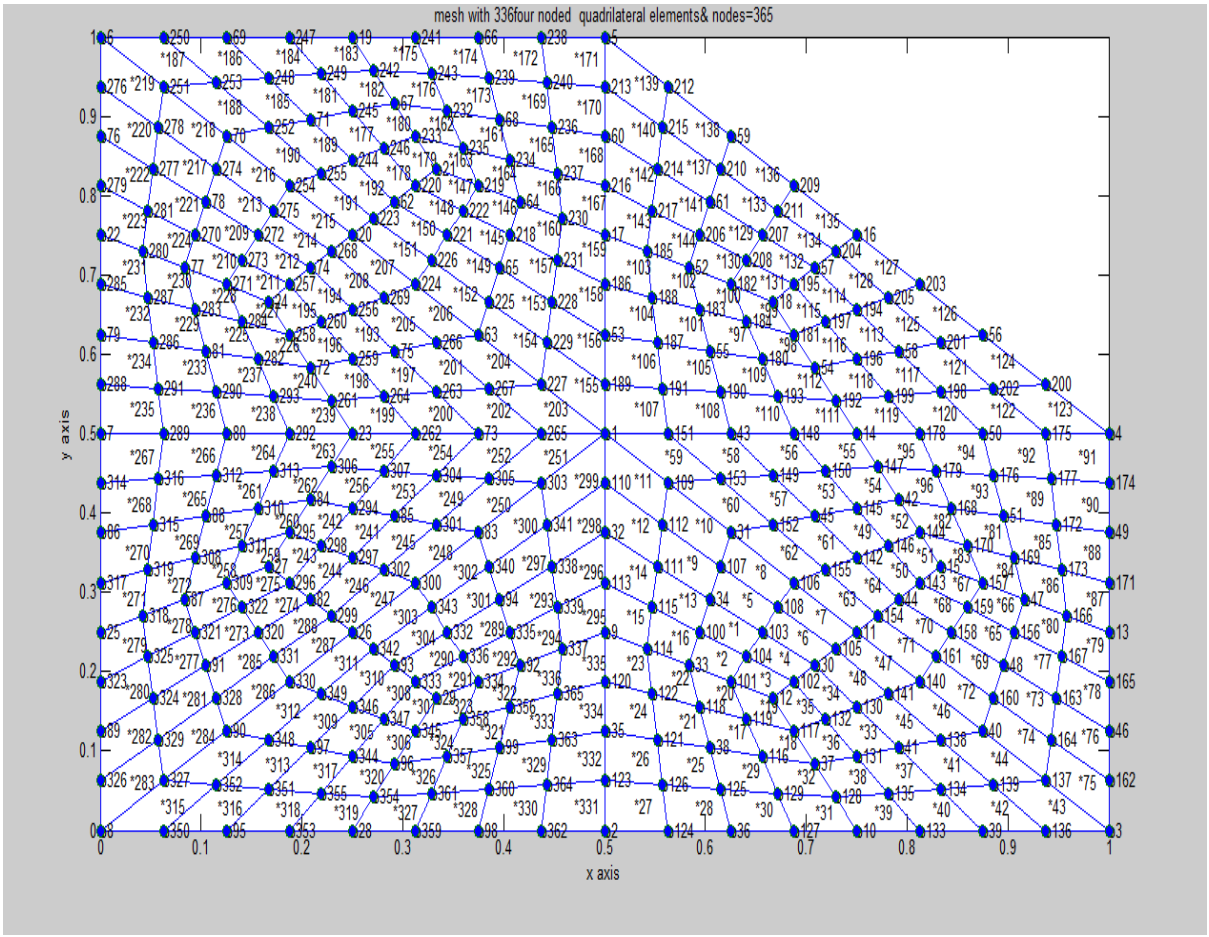
DOI: 10.18535/ijecs/v5i7.10



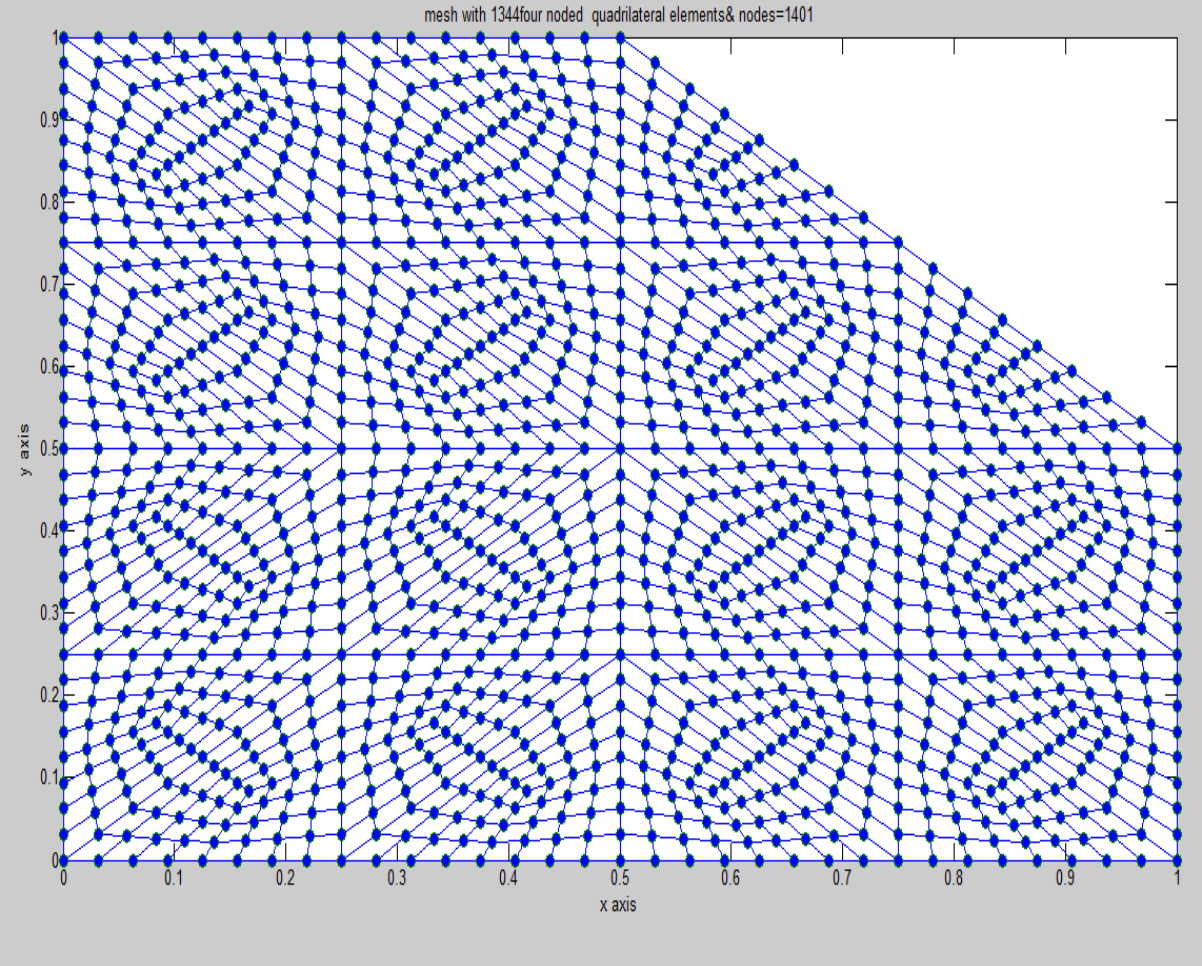


DOI: 10.18535/ijecs/v5i7.10

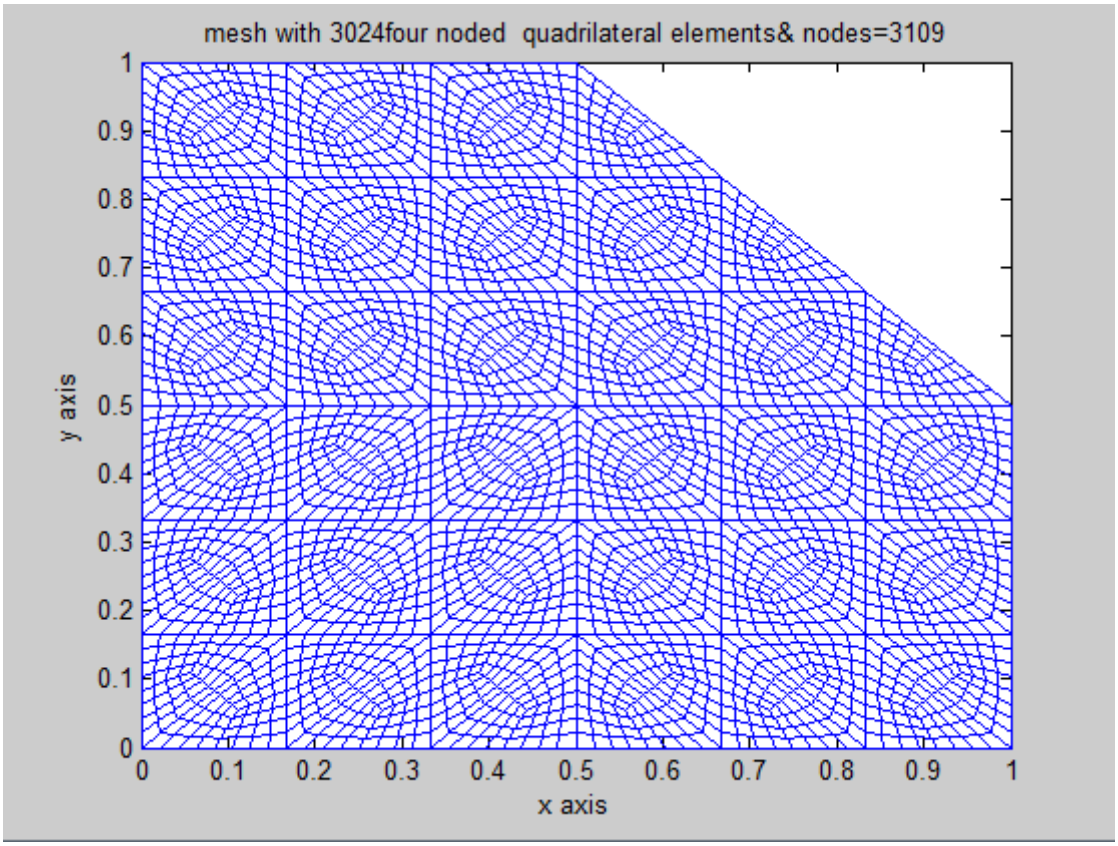


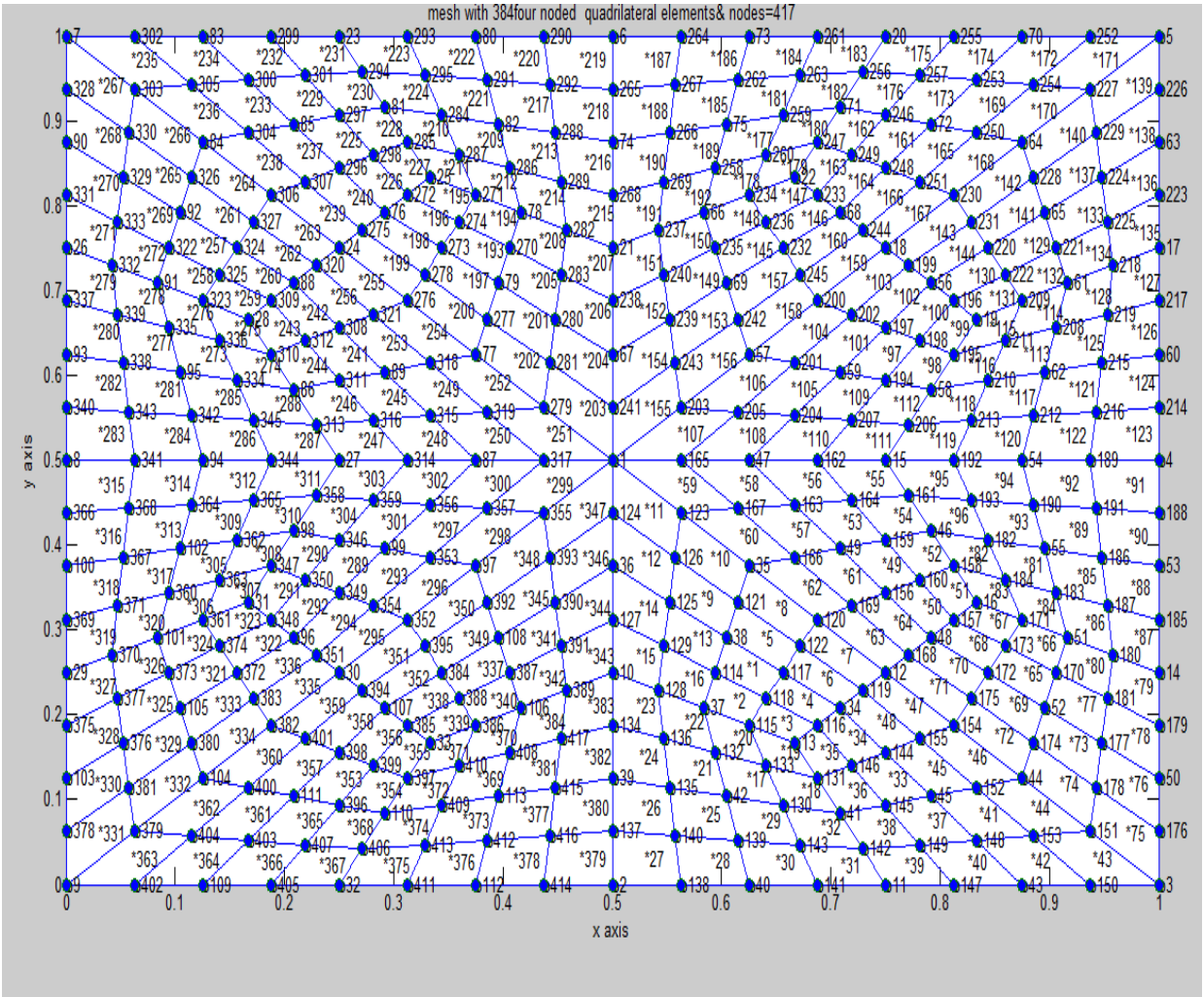


DOI: 10.18535/ijecs/v5i7.10

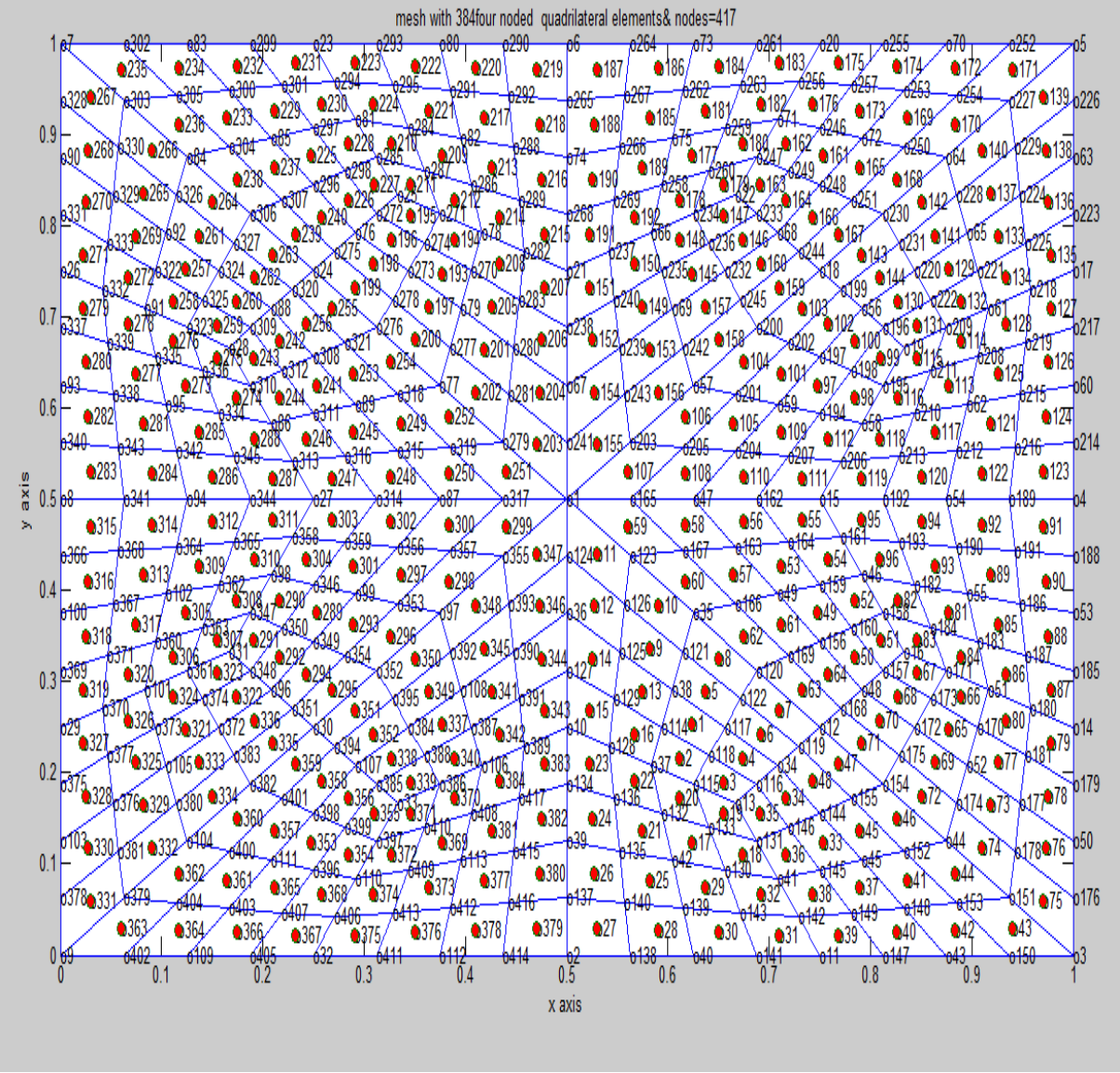


DOI: 10.18535/ijecs/v5i7.10

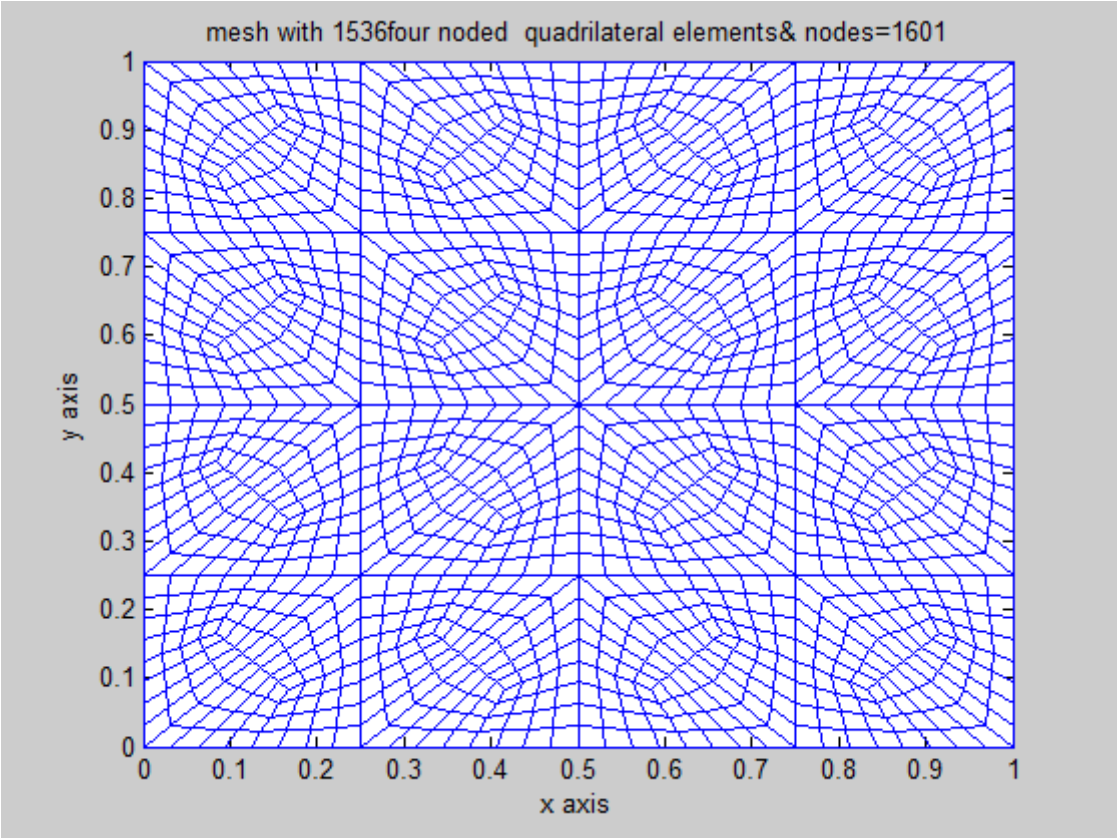


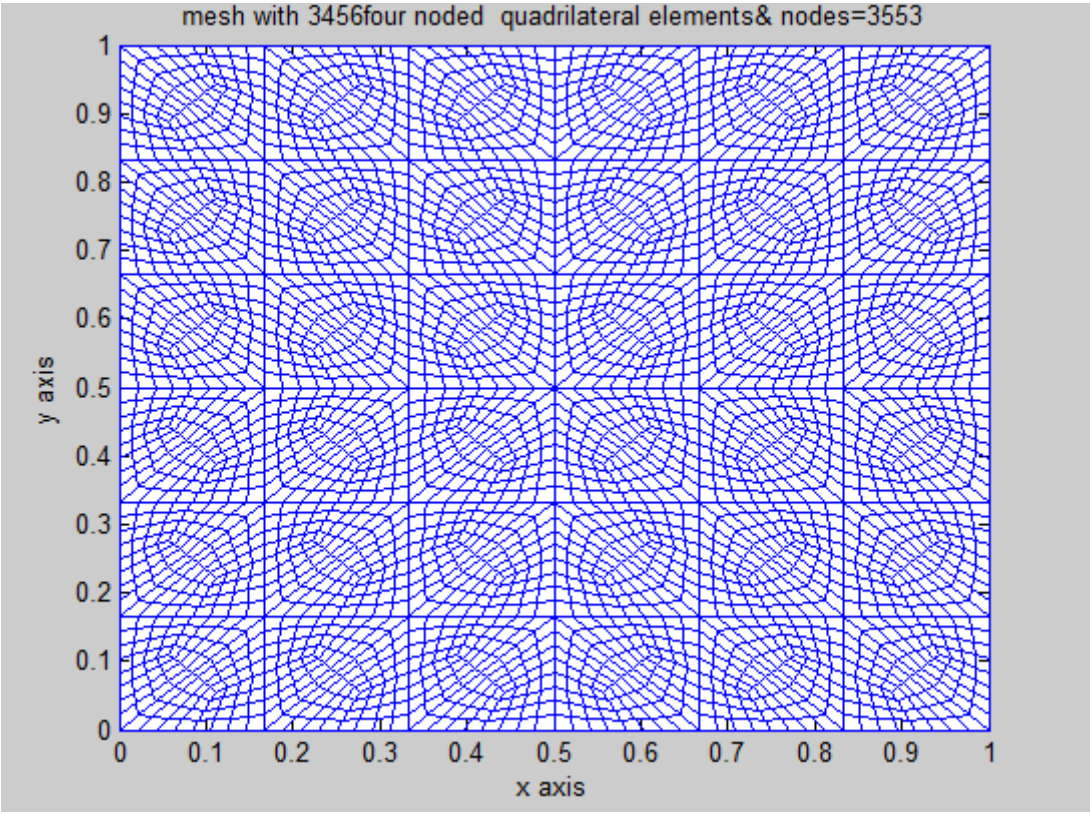


DOI: 10.18535/ijecs/v5i7.10

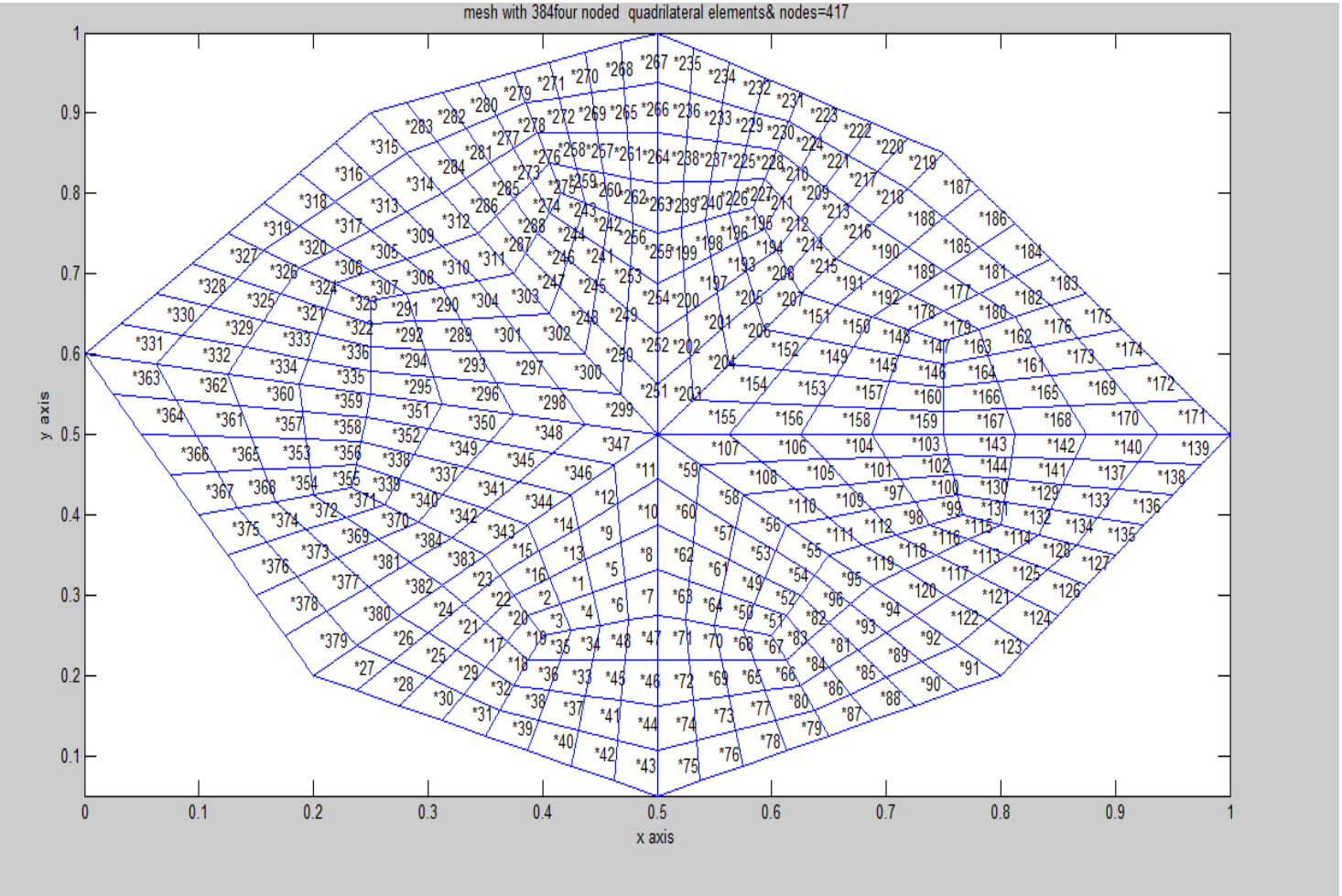


DOI: 10.18535/ijecs/v5i7.10

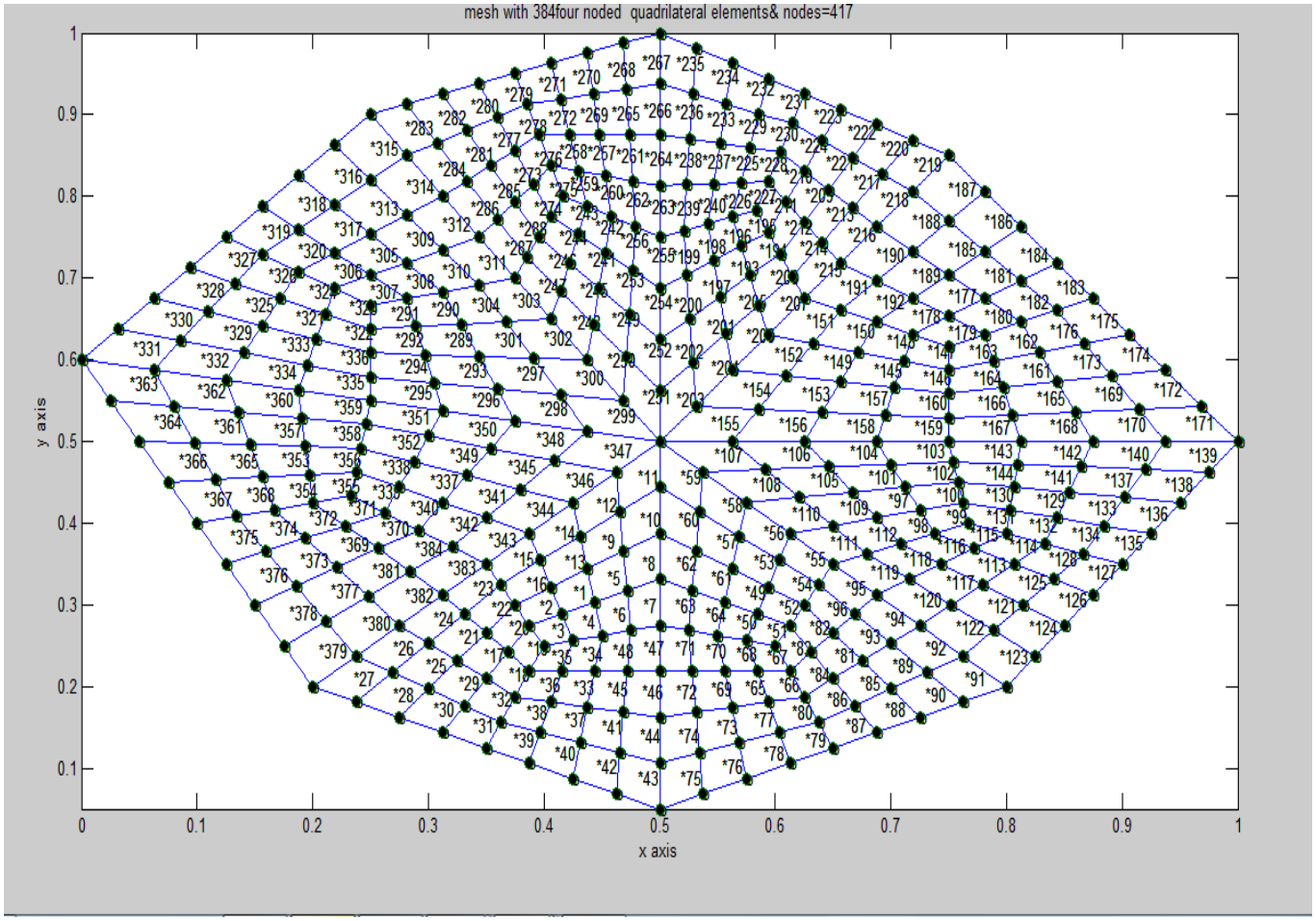




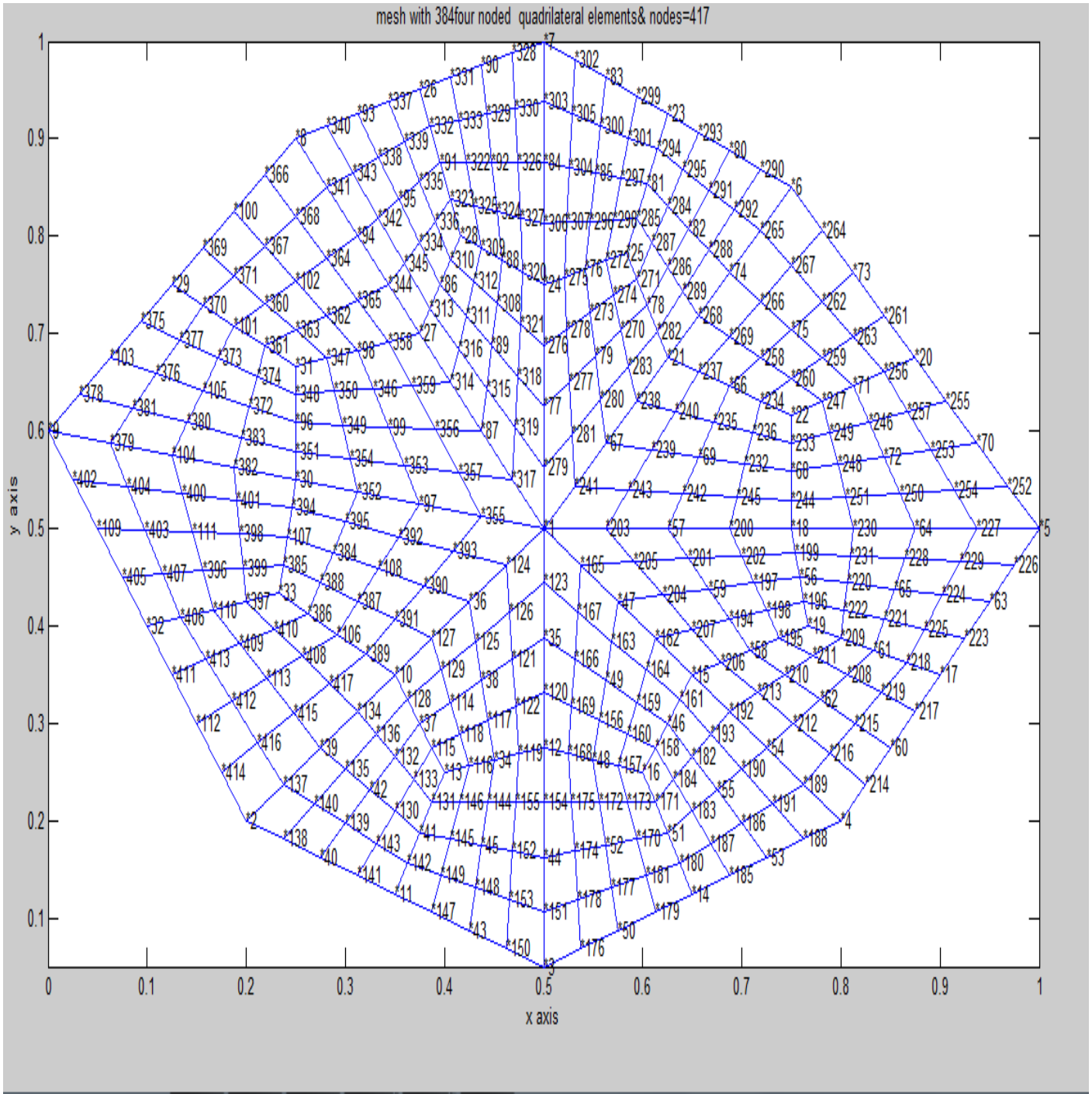
DOI: 10.18535/ijecs/v5i7.10



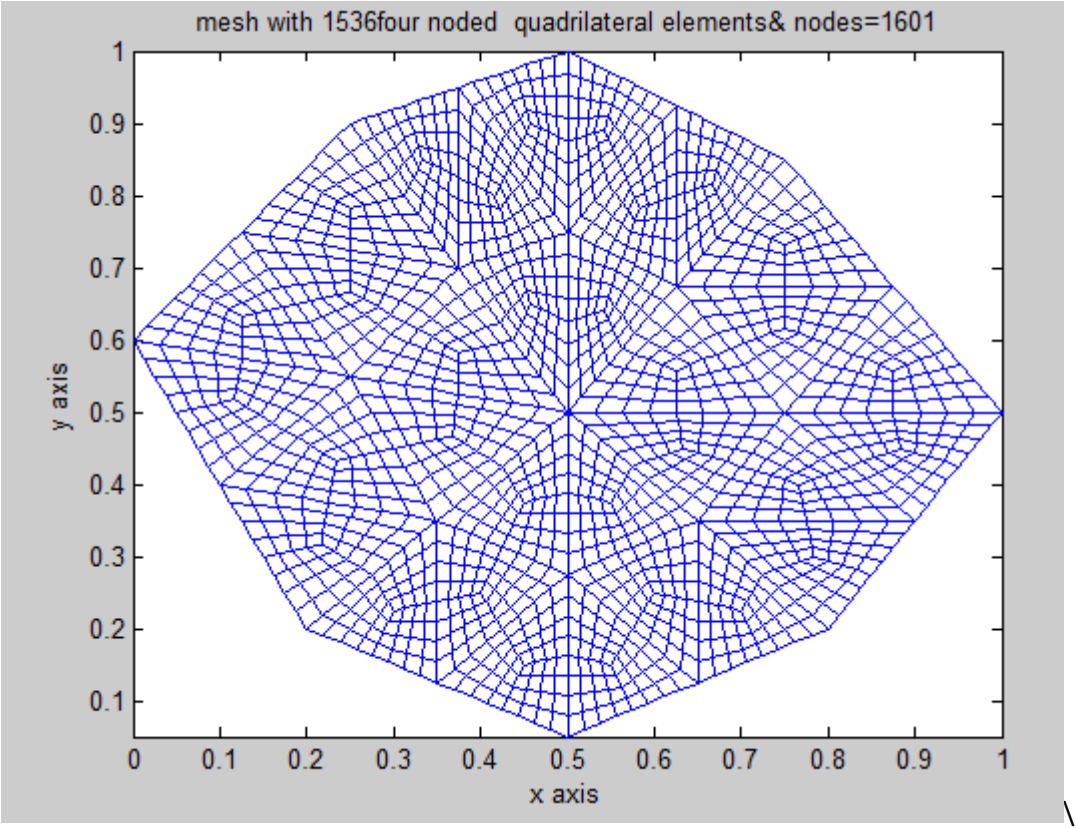
DOI: 10.18535/ijecs/v5i7.10



DOI: 10.18535/ijecs/v5i7.10



DOI: 10.18535/ijecs/v5i7.10



\