www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 12 Dec,2013 Page No. 3488-3530

A New Approach to Automatic Generation of All Quadrilateral Mesh For Finite Element Analysis

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

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

[b] Xavier Institute of Management and Entrepreneurship, Hosur Road,
Electronic City Phase II, Bangalore, Karnataka 560034.
Email: rathodbharath@gmail.com

[c] Department of Mathematics, Dayananda Sagar College of Engineering,
Bangalore- 560078, Karnataka state, India.
Email: shivaramktshiv@gmail.com

[d] Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post,
Kolar Gold Field, Kolar District, Karnataka state, Pin- 563120, India.
Email: suganthadevik@yahoo.co.in

Abstract

This paper presents a new mesh generation method for a convex polygonal domain. We first decompose the convex polygon into simple sub regions in the shape of triangles. These simple regions are then triangulated to generate a fine mesh of triangular elements. We propose then an automatic triangular to quadrilateral conversion scheme. Each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of the edges and a vertex at the barrycentre of the element. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given convex polygonal domain into all quadrilaterals, thus propagating uniform refinement. This simple method generates a high quality mesh whose elements confirm well to the requested shape by refining the problem domain. Examples are presented to illustrate the simplicity and efficiency of the new mesh generation method for standard and arbitrary shaped domains. We have appended MATLAB programs which incorporate the mesh generation scheme developed in this paper. These programs provide valuable output on the nodal coordinates ,element connectivity and graphic display of the all quadrilateral mesh for application to finite element analysis.

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

1. Introduction

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

FEM requires dividing the analysis region into many sub regions. These small regions are the elements which are connected with adjacent elements at their nodes. Mesh generation is a procedure of generating the geometric data of the elements and their nodes, and involves computing the coordinates of nodes, defining their connectivity and thus constructing the elements. Hence mesh designates aggregates of elements nodes and lines representing their connectivity. Though the FEM is a powerful and versatile tool, its usefulness is often hampered by the need to generate a mesh. Creating a mesh is the first step in a wide range of applications, including scientific and engineering computing and computer graphics. But generating a mesh can be very time consuming and prone to error if done manually. In recognition of this problem a large number of methods have been devised to automate the mesh generation task. An attempt to create a fully automatic mesh generator that is capable of generating a valid finite element meshes over arbitrary complex domains and needs only the information of the specified geometric boundary of the domain and the element size, started from the pioneering work [1] in the early 1970's. Since then many methodologies have been proposed and different algorithms have been devised in the development of automatic mesh generators [2-4]. In order to perform a reliable finite element simulation a number of researchers [5-7] have made efforts to develop adaptive FEA method which integrates with error estimation and automatic mesh modification. Traditionally adaptive mesh generation process is started from coarse mesh which gives large discretization error levels and takes a lot of iterations to get a desired final mesh. The research literature on the subject is vast and different techniques have been proposed [8]. 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 [11-20].

In this paper, we present 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 convex polygonal domain. In section 5,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for standard and arbitrary triangles,rectangles and convex polygonal domains.

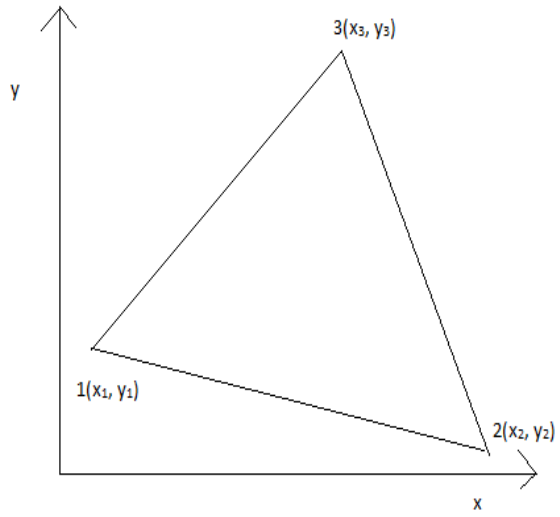
2. Division of an Arbitrary Triangle

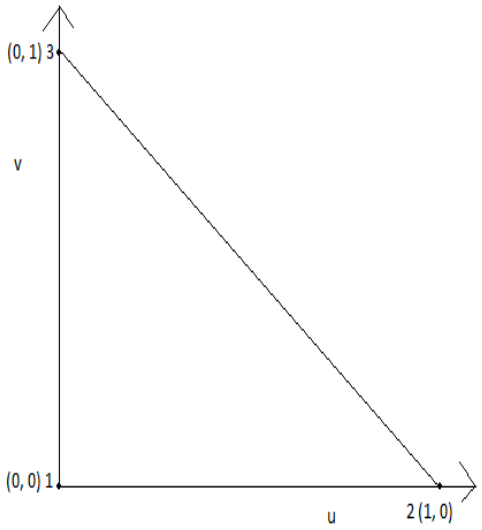
We can map an arbitrary triangle with vertices ($(x_i, y_i),\ i = 1, 2, 3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 1a, 1b. The necessary transformation is given by the equations.

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

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

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.

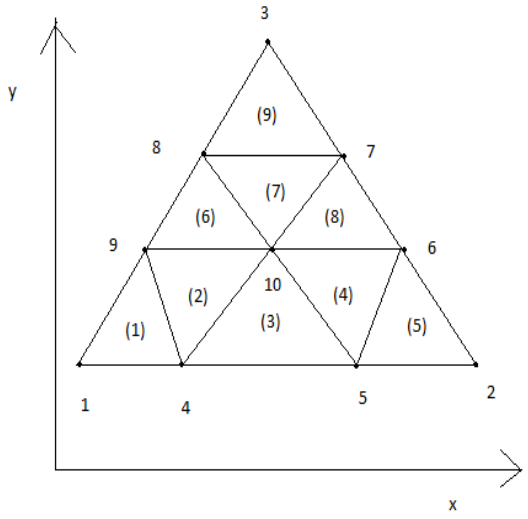


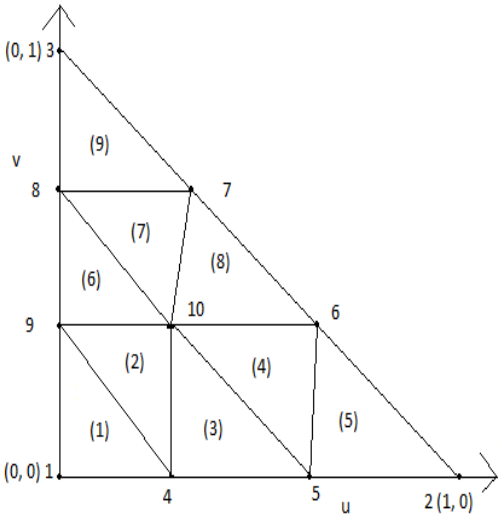


1.a 1. b

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

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

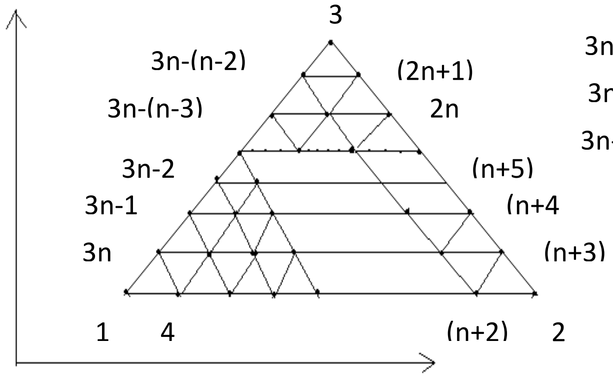




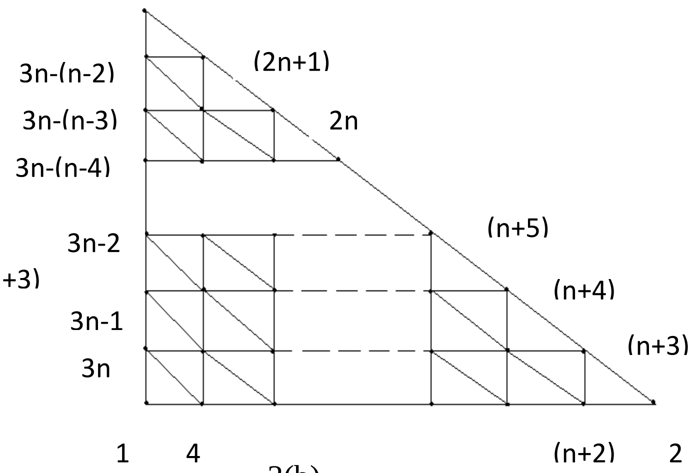
2(a) 2(b)

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 (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

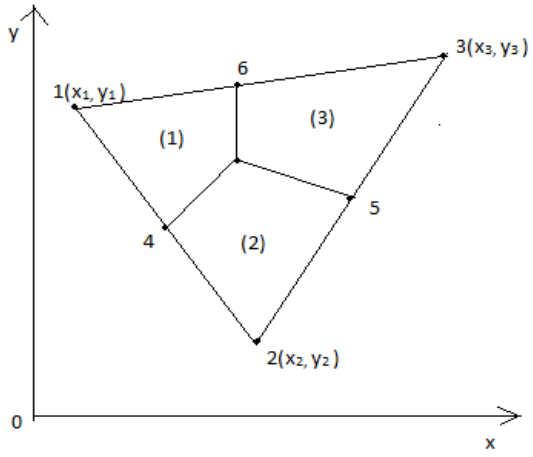
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, & (n+2) & 2 \\ 3n, & (3n+1), \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,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

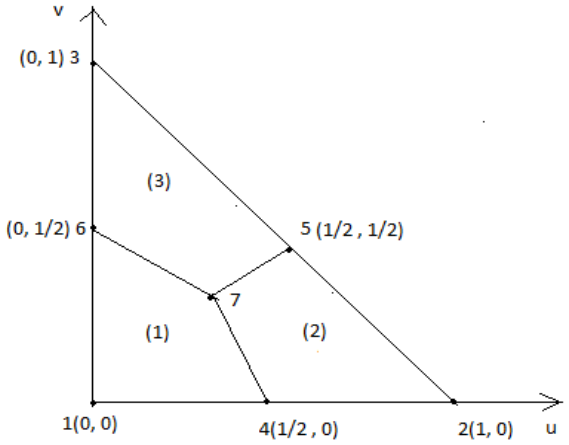
3. Quadrangulation of an Arbitrary Triangle

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



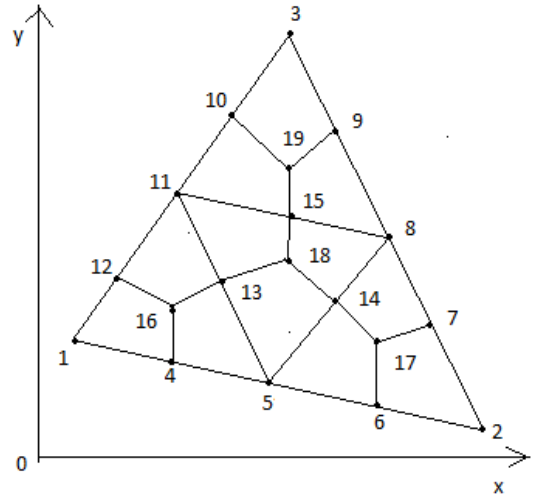


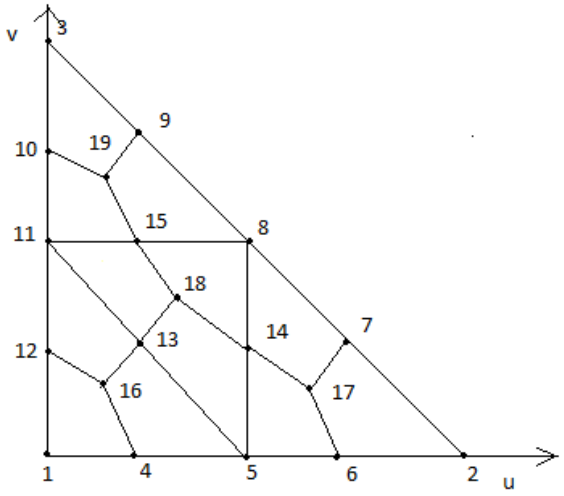
4(a) 4(b)

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





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 $\underline{rr}$

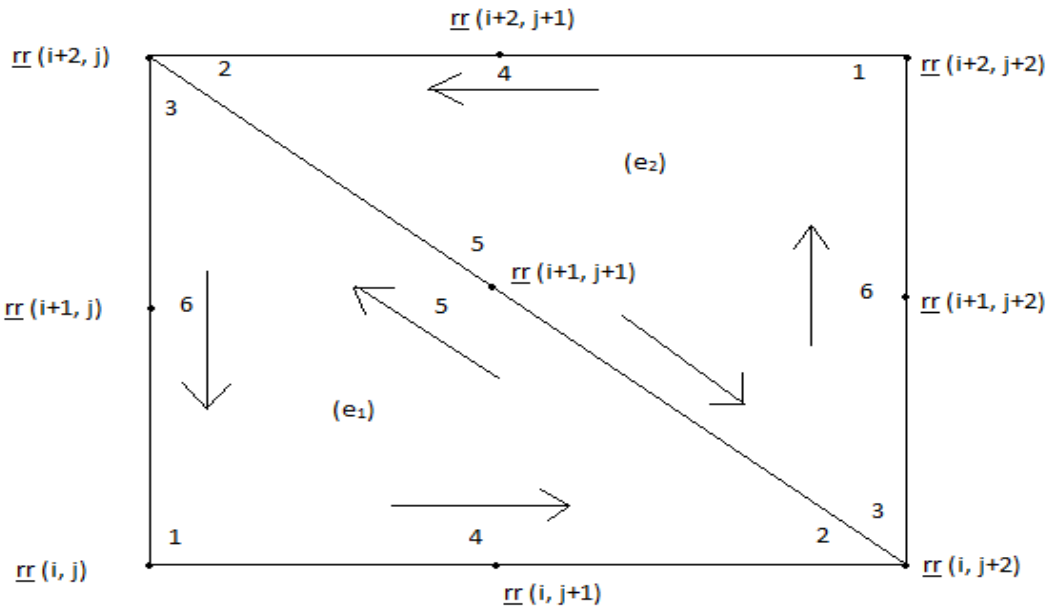


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

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

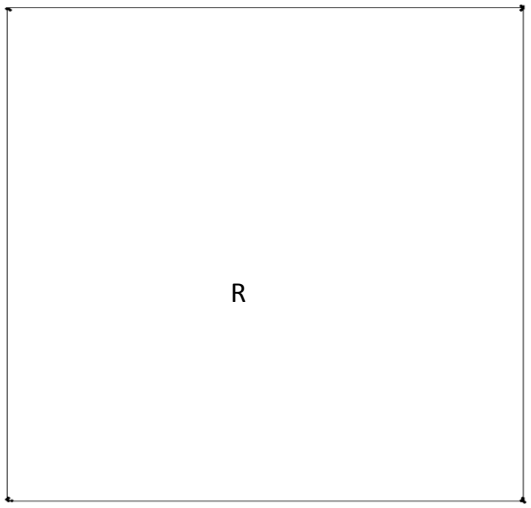
Q_{3n_2-1} < c_2 , $\underline{rr}\ (i+2,\ j+1),\ \underline{rr}\ (i+2,\ j)$, $\underline{rr}\ (i+1,\ j+1)$ >

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

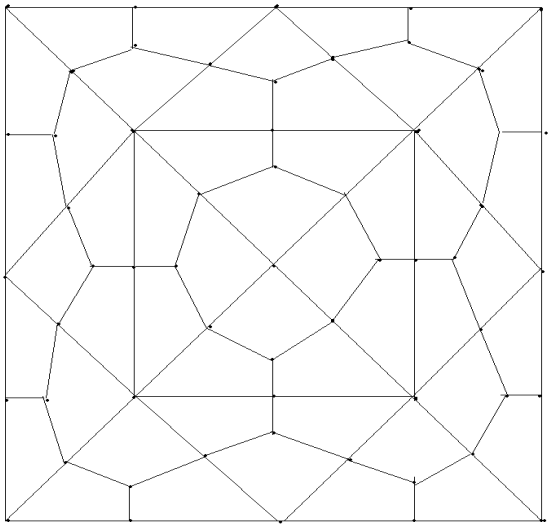
4. Quadrangulation of the Polygonal Domain

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)

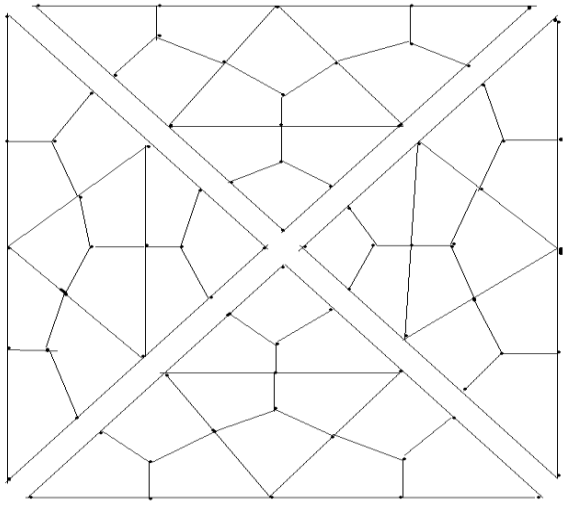


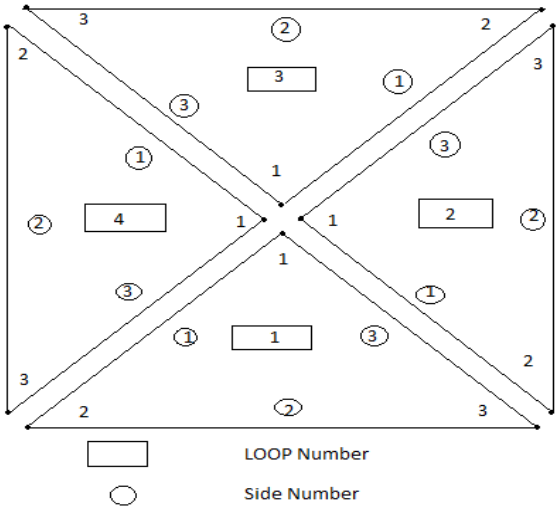
8 (a)



8 (b)

(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. Application Examples

5.1 Mesh Generation Over an Arbitrary Triangle

In applications to boundary value problems due to symmetry considerations, we may have to discretize an arbitrary triangle. Our purpose is to have a code which automatically generates 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. The following MATLAB codes are written for this purpose.

(1) polygonal_domain_coordinates. m
(2) nodaladdresses_special_convex_quadrilaterals. m
(3) generate_area_coordinate_over_standard_triangle. m
(4) quadrilateral_mesh4arbitrarytriangle_q4. m

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

5.2 Mesh Generation for a 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 such as those in [21,22]. We use the theory and procedure developed in sections 2, 3 and 4 for this purpose. The following MATLAB codes are written for this purpose.

(1) quadrilateral_mesh4MOINEX_q4. m
(2) nodaladdresses_special_convex_quadrilaterals_trial. m
(3) polygonal_domain_coordinates. m
(4) generate_area_coordinate_over_standard_triangle. m
(5) quadrilateral_mesh4convexpolygonsixside_q4. m

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. The development of the MATLAB code is partly inspired by a procedure given in [23] and the literature available on the internet[24]

Conclusions

An automatic indirect quadrilateral mesh generator which uses the splitting technique is presented for the two dimensional 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 barrycentre of the triangular element. This task is made a bit simple since a fine mesh of six node triangles is first generated. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this discretizes the given convex polygonal domain into all quadrilaterals, thus propogating a uniform refinement. This simple method generates high quality mesh whose elements confirm well to the requested

shape by refining the problem domain. We have also appended MATLAB programs which provide the nodal coordinates, element nodal connectivity and graphic display of the generated all quadrilateral 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.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[15] Masud. A, Khurram. R. A, A multiscale/stabilized finite element method for the advection-

diffusion equation, Comput. Methods. Appl. Mech. Eng 193, 1997-2018(2004)
[16] Johnston. B.P, Sullivan. J. M and Kwasnik. A, Automatic conversion of triangular finite element meshes to quadrilateral elements, Int. J. Numer. Meth. Eng. 31, 67-84(1991)

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

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

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

[20] Park. C, Noh. J. S, Jang. I. S and Kang. J. M, A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints, Computer Aided Design 39, 258-267(2007)
[21]Moin.P, Fundamentals of Engineering Numerical Analysis,second edition,Cambridge University Press(2010)
[22]Fausett.L.V, Applied Numerical Analysis Using MATLAB, second edition, Pearson Education.Inc(2008)
[23]Thompson.E.G, Introduction to the finite element method,John Wiley & Sons Inc.(2005)

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

PROGRAMS

```
%PROGRAM-(1) polygonal_domain_coordinates.m
function[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(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
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
syms U V W xi yi

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


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

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

case 4%for a unit square: -0.5<=x,y<=0.5
 % 1   2   3  4  5  6   7  8
 x=sym([0;  0; 1/2;1/2;1/2;  0;-1/2;-1/2;-1/2])
 y=sym([0;-1/2;-1/2;  0;1/2;1/2; 1/2;  0;-1/2])
case 5%for a convexpolygonsixside
 %   1   2  3 4  5 6 7
```

```
x=sym([0.5;0.1;0.7;1;.75;.5;0])
y=sym([0.5;0;0.2;.5;.85;1;.25])
case 6%standard triangle
x=sym([0;1;0])
y=sym([0;0;1])
case 7%equilateral triangle

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

end
if (nmax>3)
[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,n);
end
if (nmax==3)
   [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_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(spqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
spqd
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])



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;
    node1=eln(iii,1)
    node2=eln(iii,2)
    node3=eln(iii,3)
```

```
    mm=eln(iii,7)
    xi(mm,1)=(xi(node1,1)+xi(node2,1)+xi(node3,1))/3;
    yi(mm,1)=(yi(node1,1)+yi(node2,1)+yi(node3,1))/3;
  end

end
N=(1:nnode)'
[N xi yi]
%
coord(:,1)=(xi(:,1));
coord(:,2)=(yi(:,1));
gcoord(:,1)=double(xi(:,1));
gcoord(:,2)=double(yi(:,1));
%disp(gcoord)
%PROGRAM-(2) nodaladdresses_special_convex_quadrilaterals.m
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_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

%PROGRAM-(3) 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
%  ui
 % vi
  %wi
  U'
  V'
  W'

%PROGRAM-(4) quadrilateral_mesh4arbitrarytriangle_q4.m
function[]=quadrilateral_mesh4arbitrarytriangle_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_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,1,2,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,4,4,5,1,1)
%quadrilateral_mesh4arbitrarytriangle_q4(n1=1,n2=2,n3=3,nmax=3,numtri=1,ndiv=2,mesh=6,xlength=1,ylength=1)
%quadrilateral_mesh4arbitrarytriangle_q4(n1=1,n2=2,n3=3,nmax=3,numtri=4,ndiv=4,mesh=6,xlength=1,ylength=1)
%quadrilateral_mesh4arbitrarytriangle_q4(1,2,3,3,4,4,6,1,1)
%quadrilateral_mesh4arbitrarytriangle_q4(1,2,3,3,4,4,7,1,1)

[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);

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

%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
clf
for i=1:nel
```

```
for j=1:nnel
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
axis equal
%axis tight
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 5
axis([0 xlength 0 ylength])
case 6
axis([0 xlength 0 ylength])
case 7
axis([0 xlength 0 ylength])


end
 %
plot(xvec,yvec);%plot element
hold on;
if (ndiv<=14)
%place element number
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['[',num2str(i),']']);
end%if ndiv
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<=16)
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end%ndiv
%axis off
%PROGRAM(5)- quadrilateral_mesh4MOINEX_q4.m
function[]=quadrilateral_mesh4MOINEX_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_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,9,6,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);
```

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

%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
clf
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
axis equal
%axis tight
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
end
 %
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=4
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['[',num2str(i),']']);
end
end;%i loop
xlabel('x axis')
ylabel('y axis')
st1='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<=4
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end
%axis off
%PROGRAM(6)- nodaladdresses_special_convex_quadrilaterals_trial.m
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,n)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
```

```
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1],[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],[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)])

%PROGRAM(7)- quadrilateral_mesh4convexpolygonsixside_q4.m
function[]=quadrilateral_mesh4convexpolygonsixside_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_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,1,2,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,4,4,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_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_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,8,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);

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

%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
clf
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
%axis equal
axis tight
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 5
axis([0 xlength 0 ylength])
case 8
   axis([0 xlength 0 ylength])
```

```
end
 %
plot(xvec,yvec);%plot element
hold on;
%place element number
if (ndiv<=4)
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['[',num2str(i),']']);
end% if ndiv
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<=4)
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end%if ndiv
%axis off
```

FIGURES

FIGURE-1

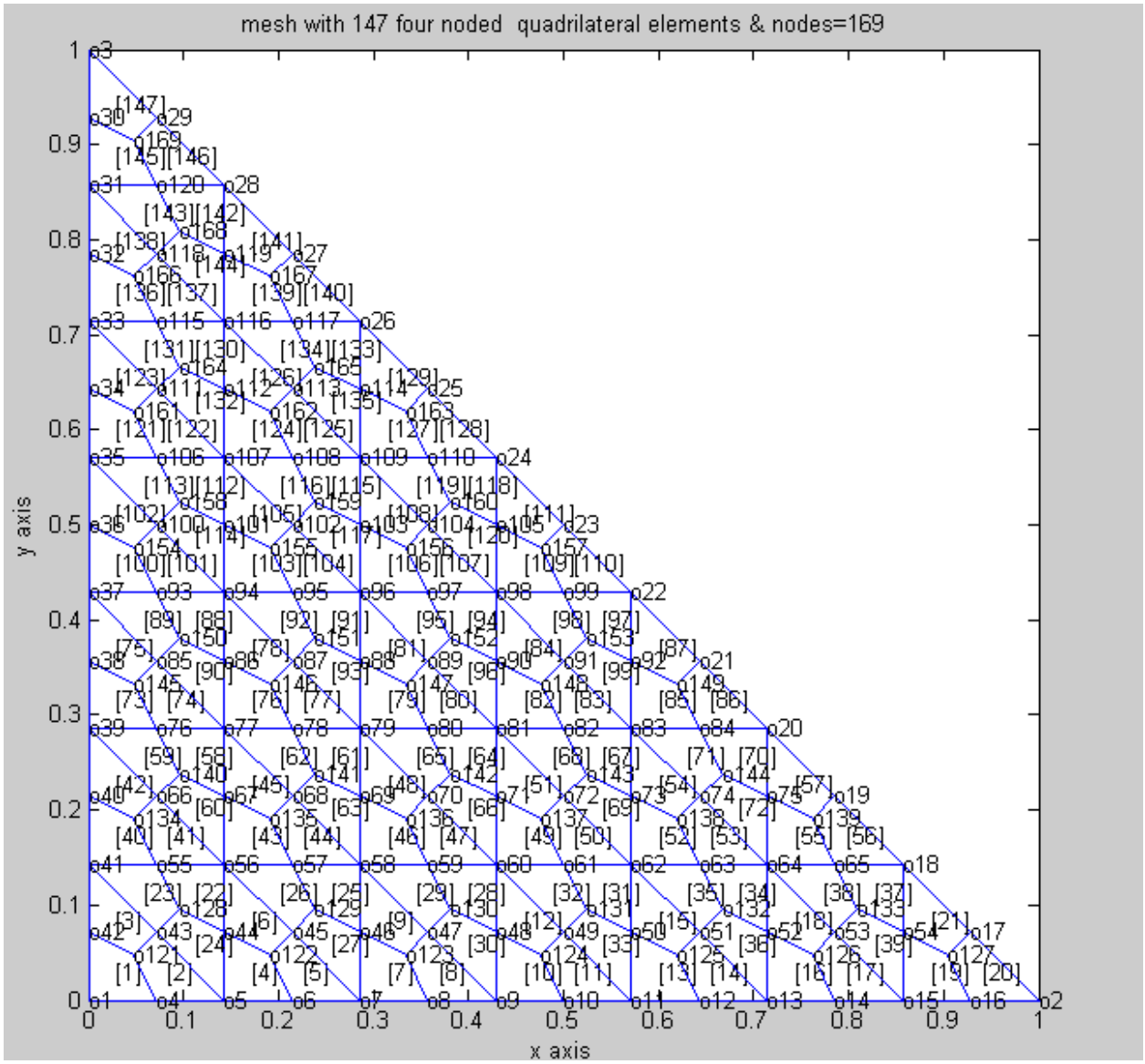

mesh with 147 four noded quadrilateral elements & nodes=169
y axis
x axis

FIGURE-2

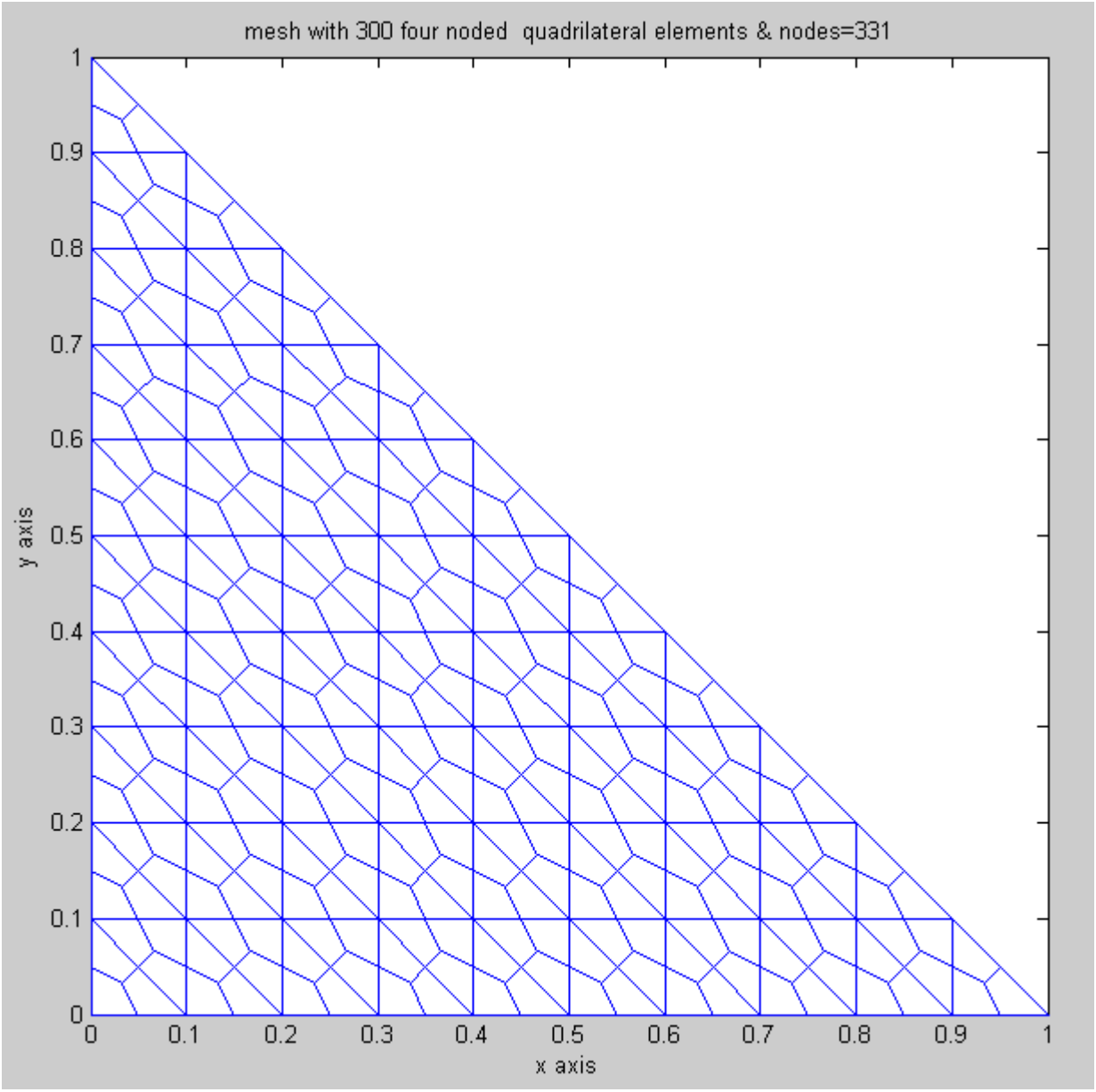

mesh with 300 four noded quadrilateral elements & nodes=331
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-3

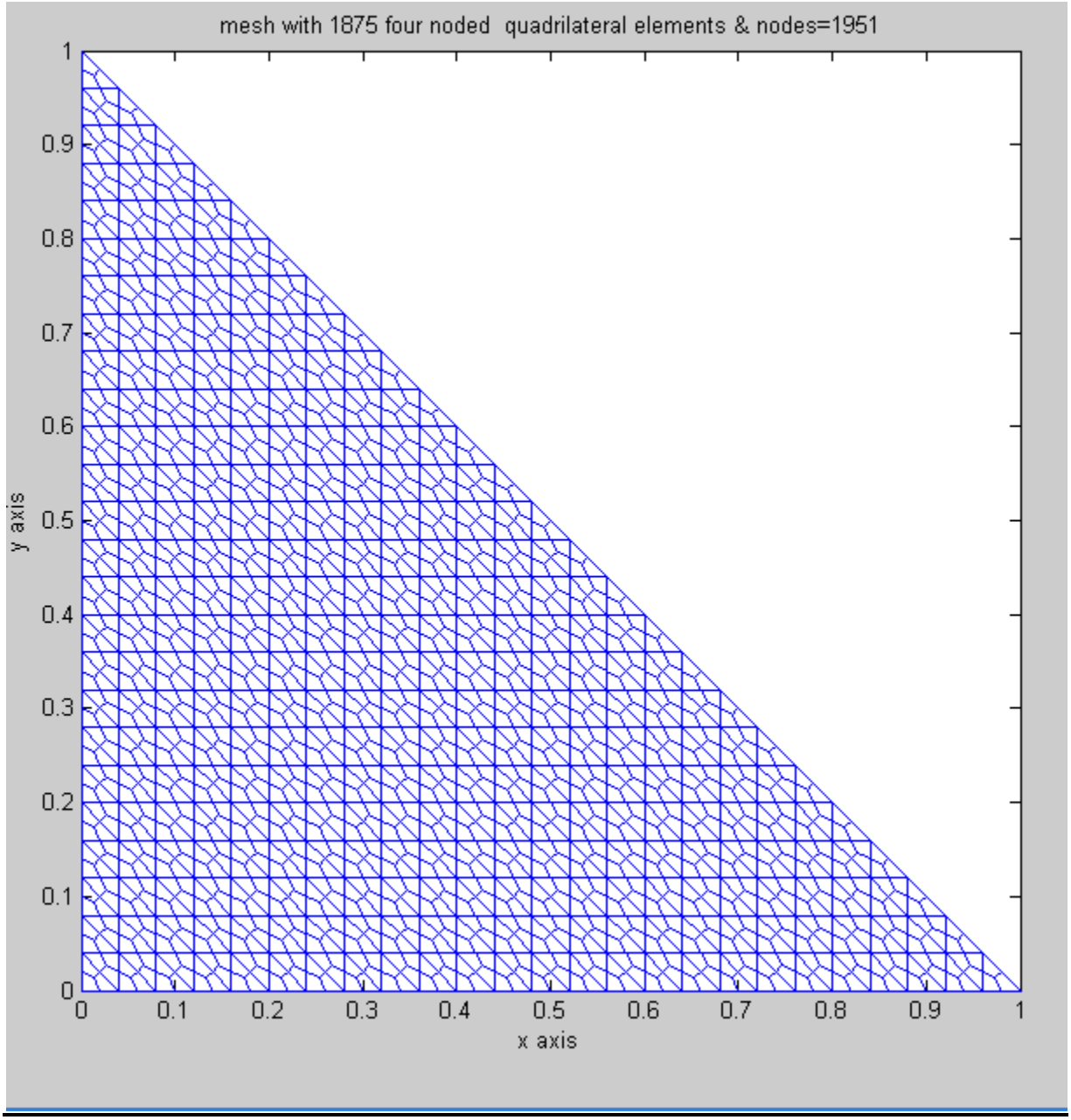

mesh with 1875 four noded quadrilateral elements & nodes=1951
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-4

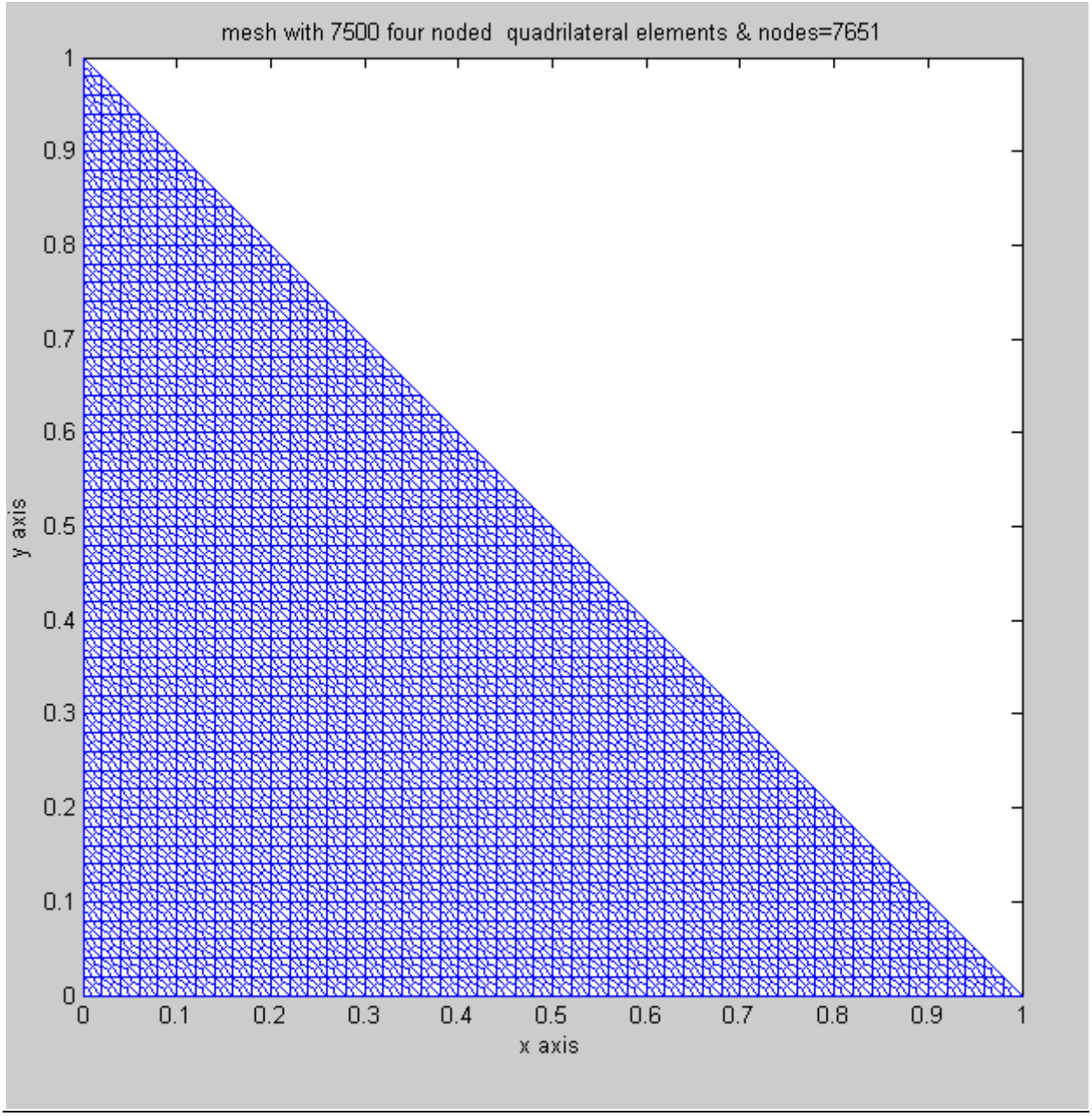

mesh with 7500 four noded quadrilateral elements & nodes=7651
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-5

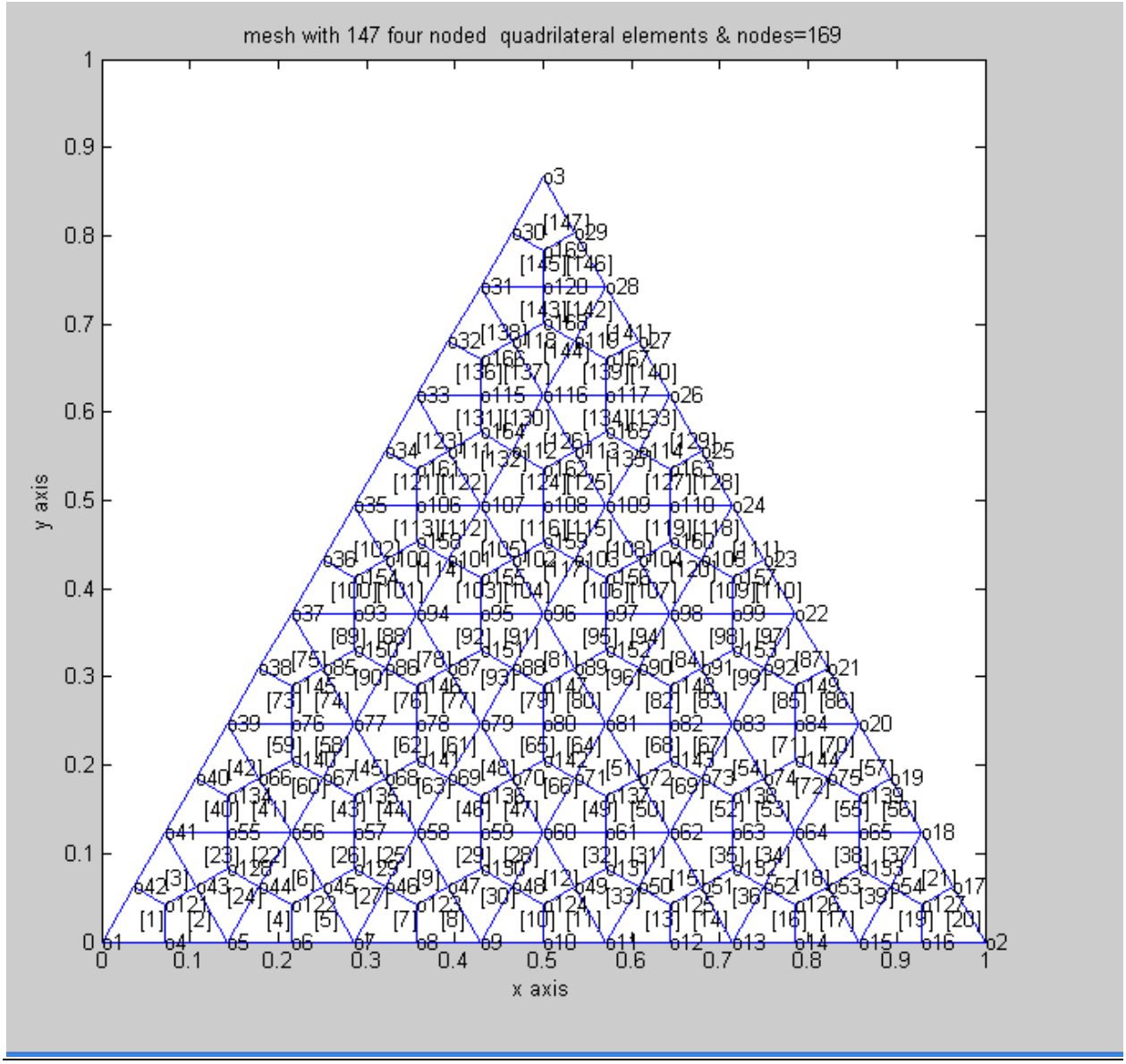

mesh with 147 four noded quadrilateral elements & nodes=169
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-6

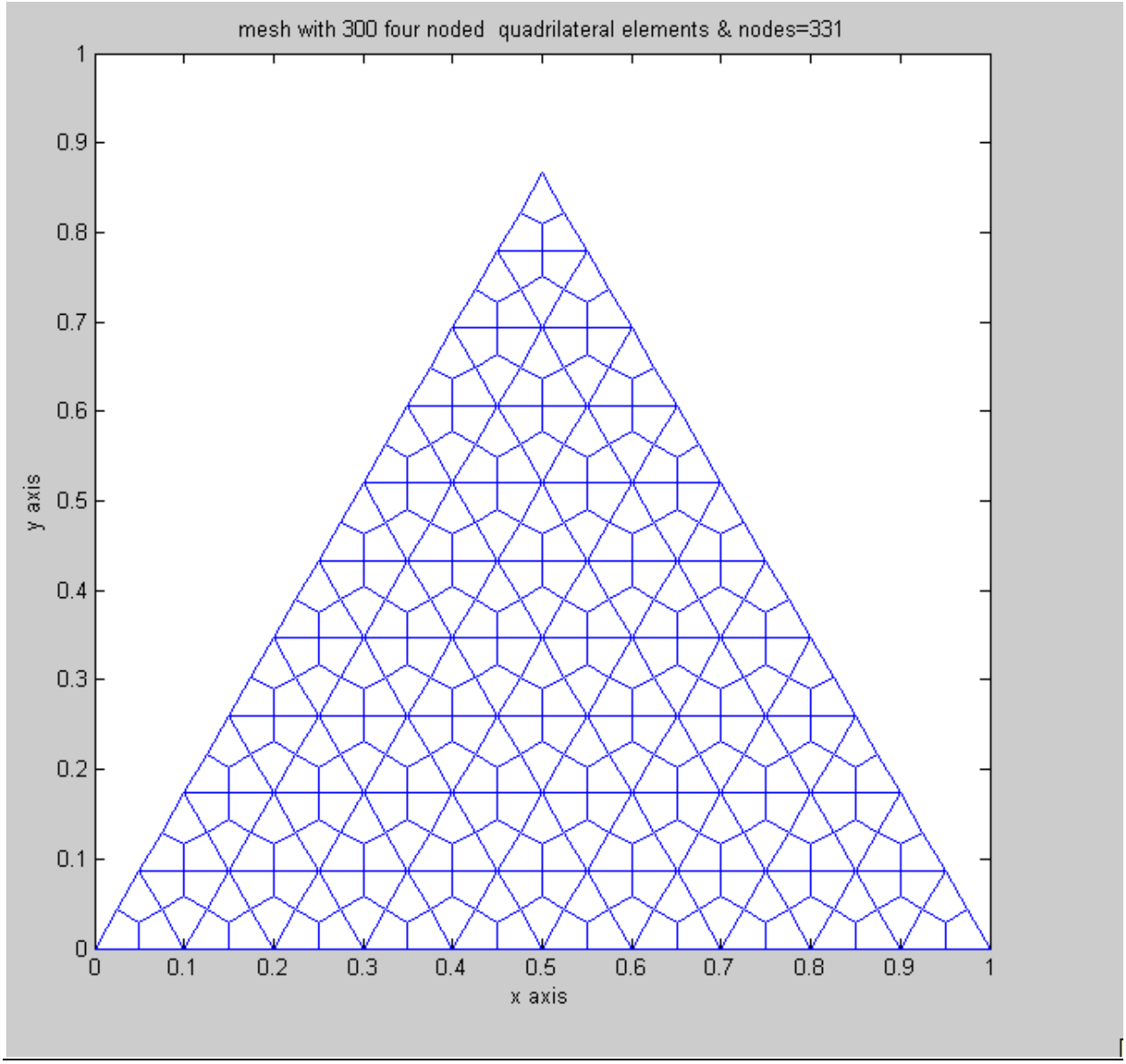

mesh with 300 four noded quadrilateral elements & nodes=331
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-7

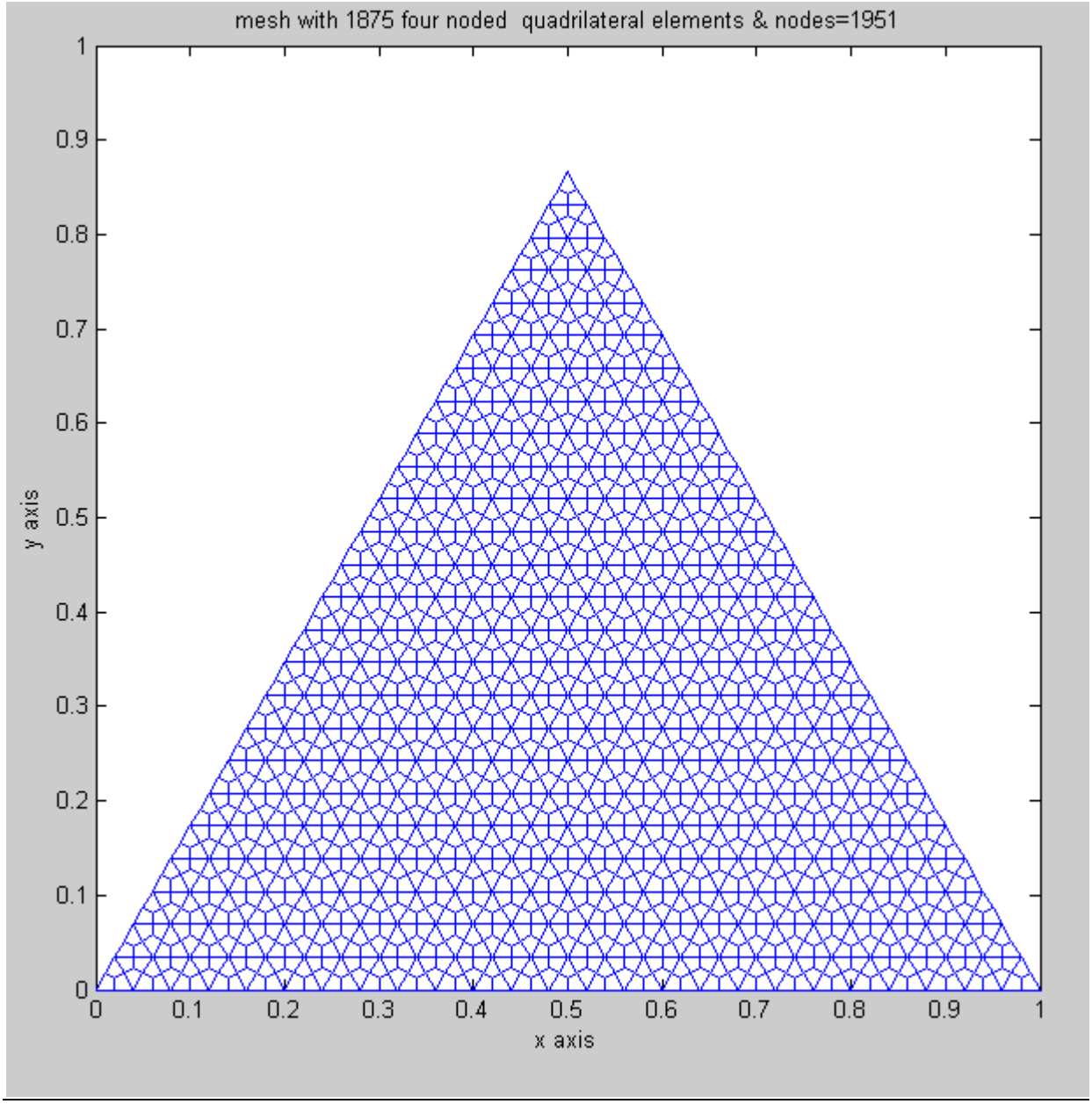

mesh with 1875 four noded quadrilateral elements & nodes=1951
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-8

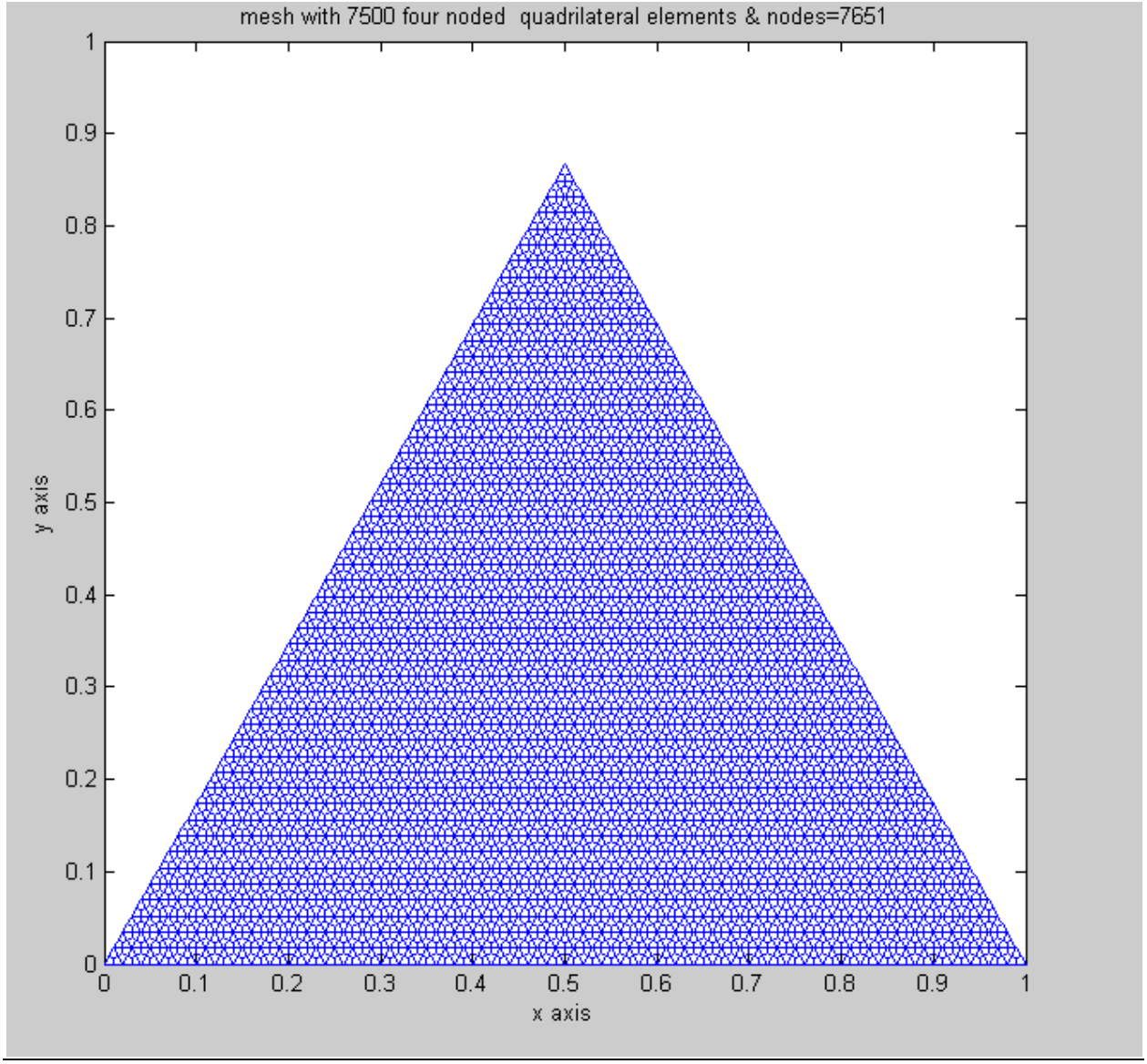

mesh with 7500 four noded quadrilateral elements & nodes=7651
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-9

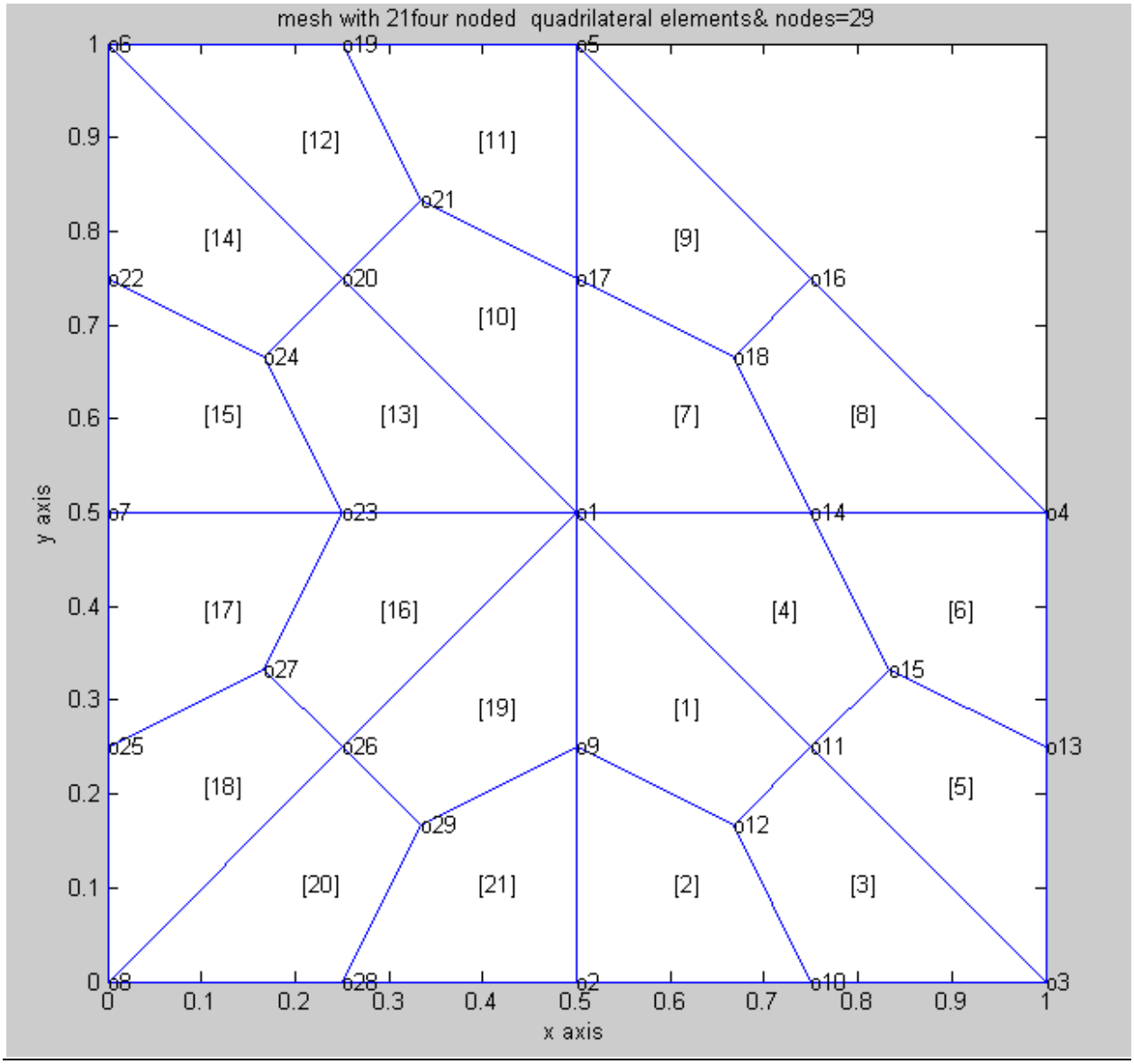

mesh with 21four noded quadrilateral elements& nodes=29
y axis
x axis
o6
o19
o5
o21
o22
o20
o17
o16
o24
o18
o7
o23
o1
o14
o4
o27
o15
o25
o26
o9
o11
o13
o29
o12
o8
o28
o2
o10
o3
[12]
[11]
[14]
[9]
[10]
[15]
[13]
[7]
[8]
[17]
[16]
[4]
[6]
[19]
[1]
[18]
[5]
[20]
[21]
[2]
[3]

FIGURE-10

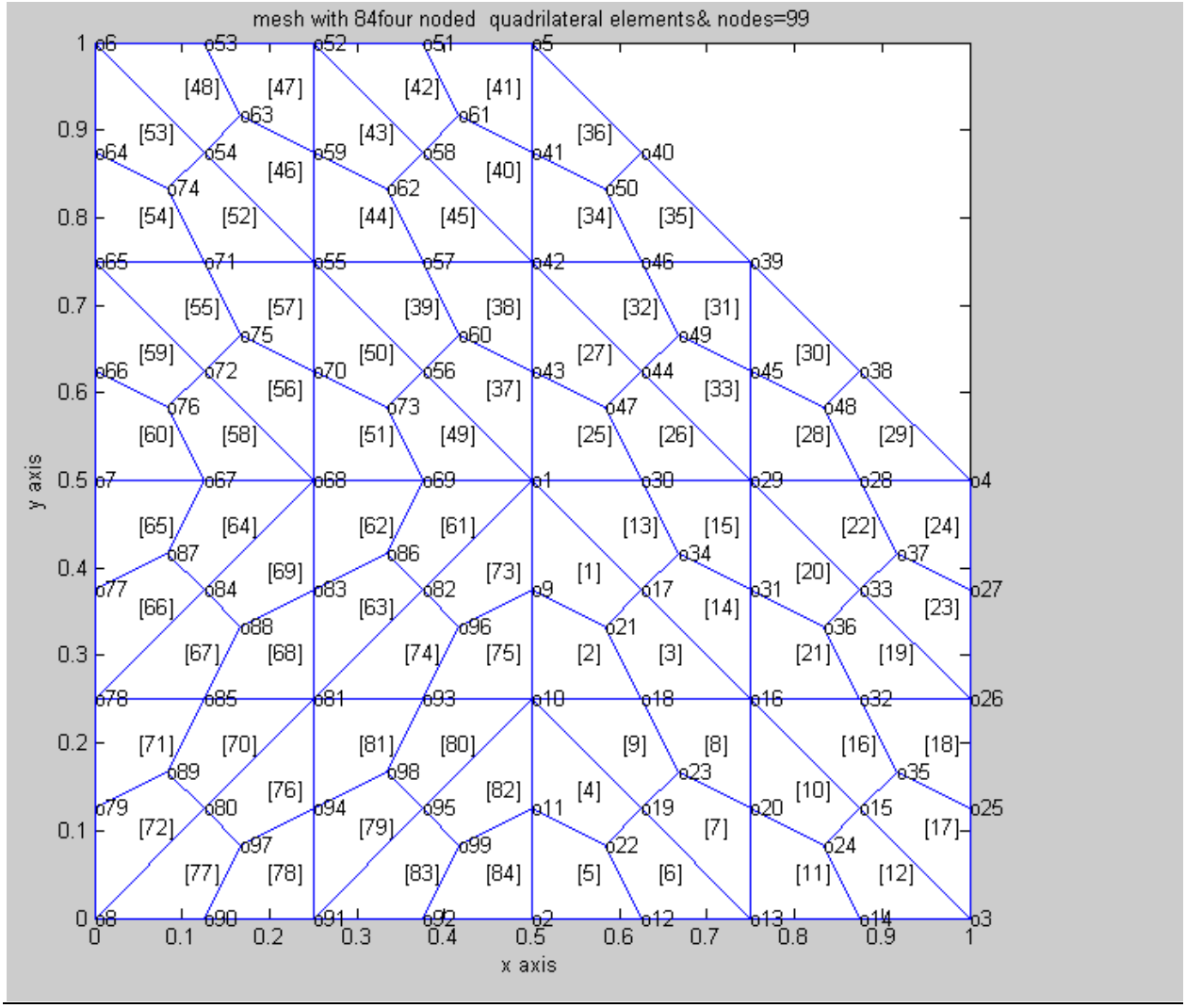

mesh with 84four noded quadrilateral elements& nodes=99
y axis
x axis

FIGURE-11

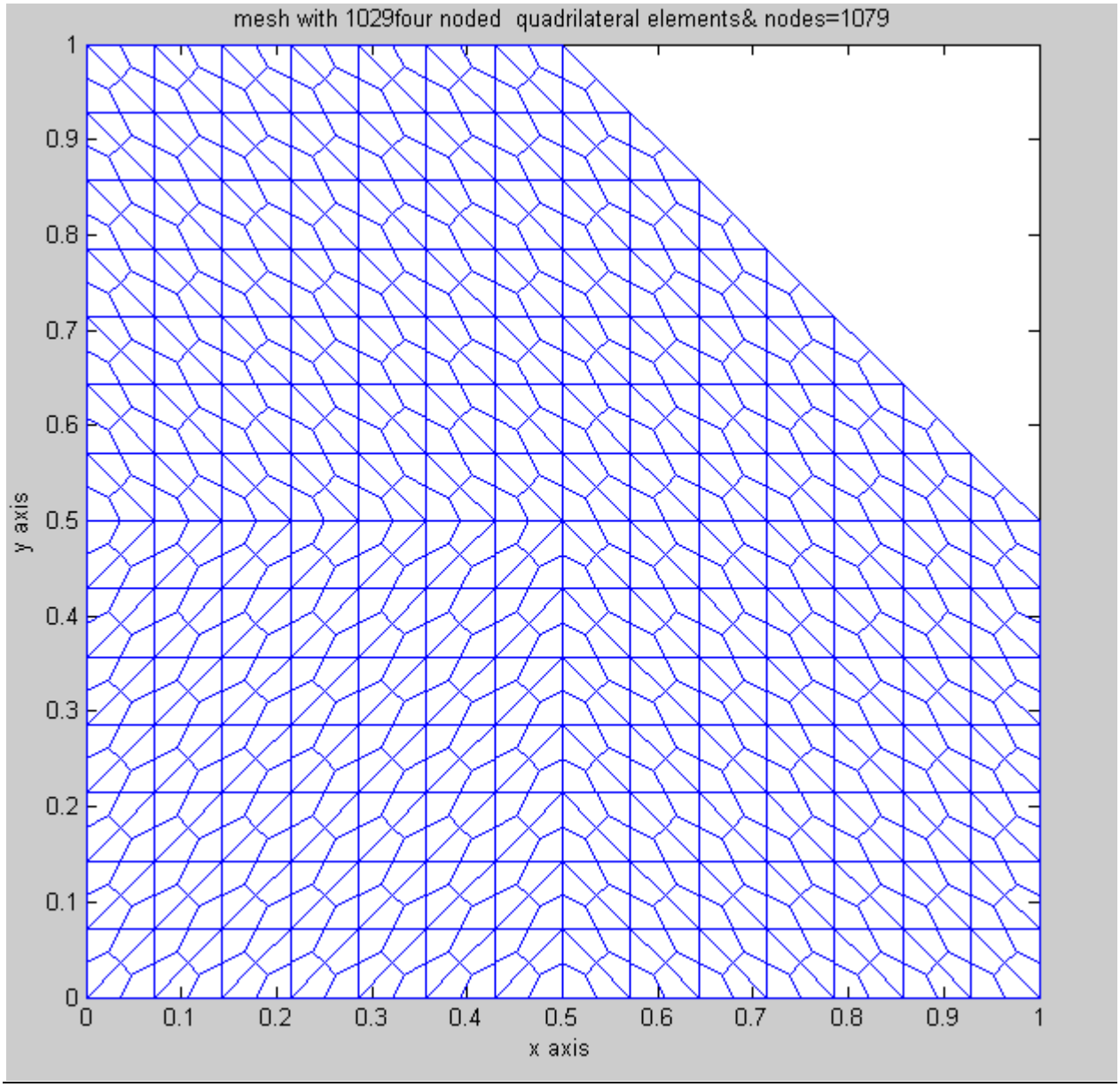

mesh with 1029four noded quadrilateral elements& nodes=1079
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-12

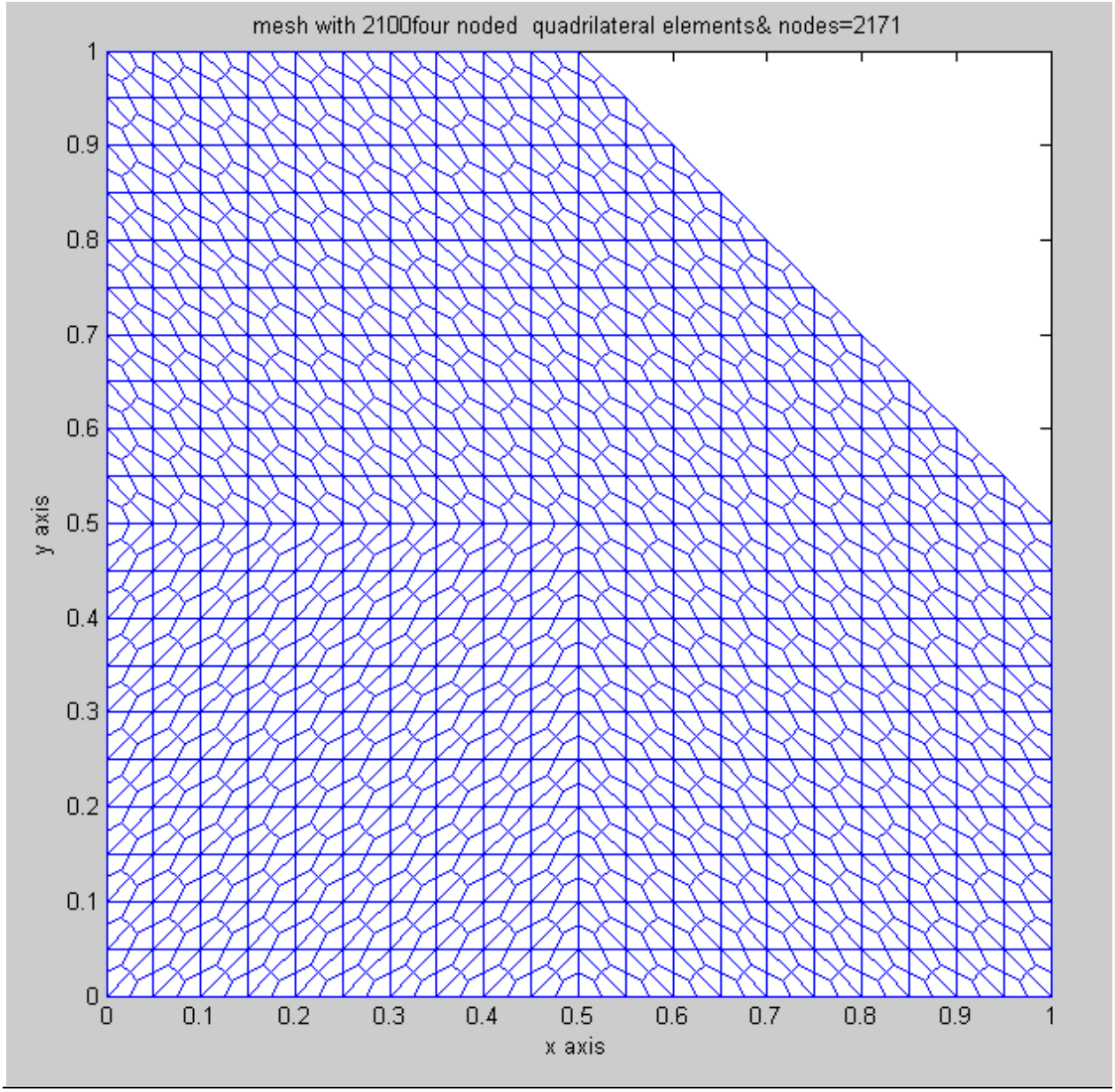

mesh with 2100four noded quadrilateral elements& nodes=2171
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-13

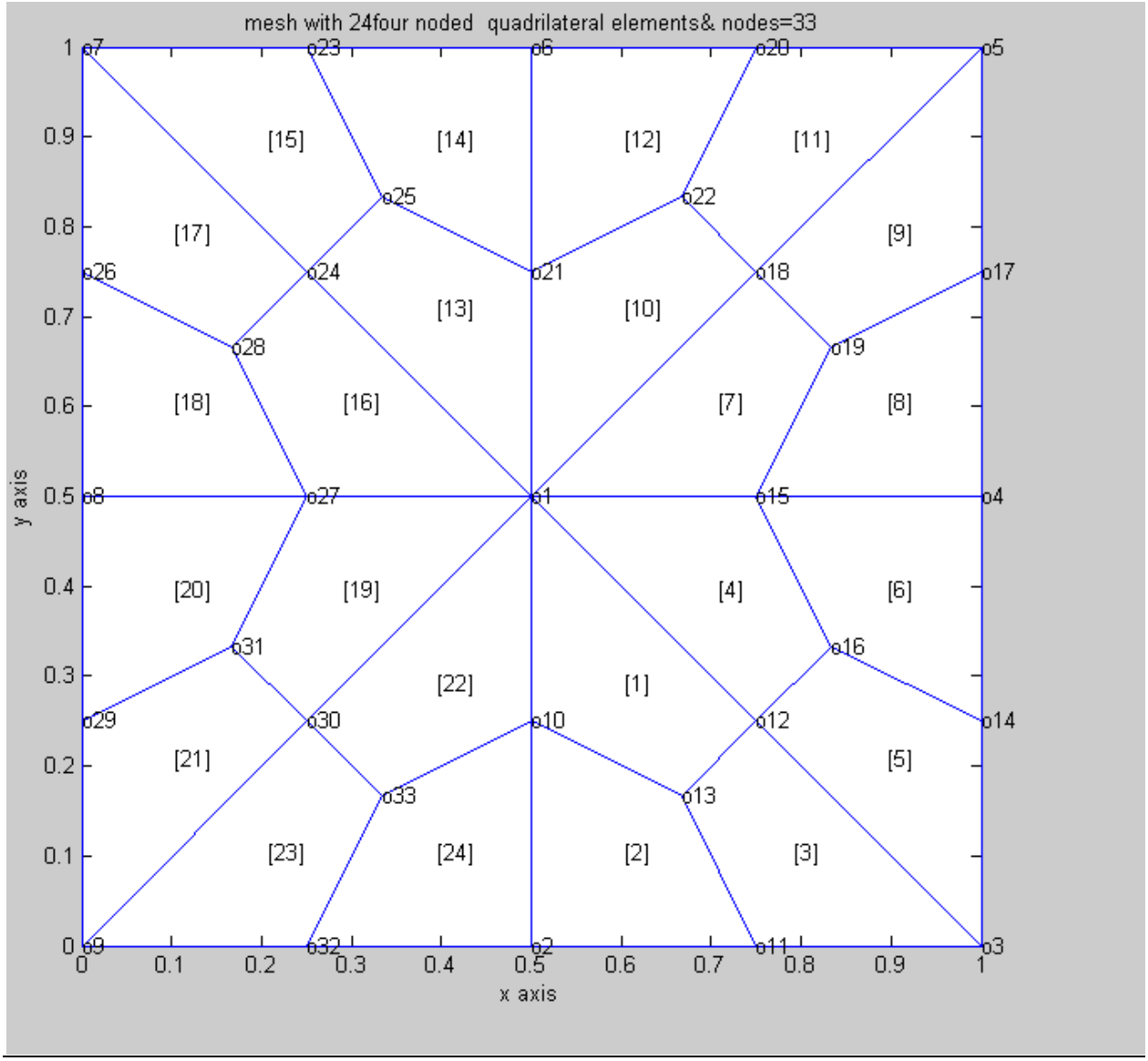

mesh with 24four noded quadrilateral elements& nodes=33
y axis
x axis
[15]
[14]
[12]
[11]
[17]
[9]
[13]
[10]
[18]
[16]
[7]
[8]
[20]
[19]
[4]
[6]
[22]
[1]
[21]
[5]
[23]
[24]
[2]
[3]

FIGURE-14

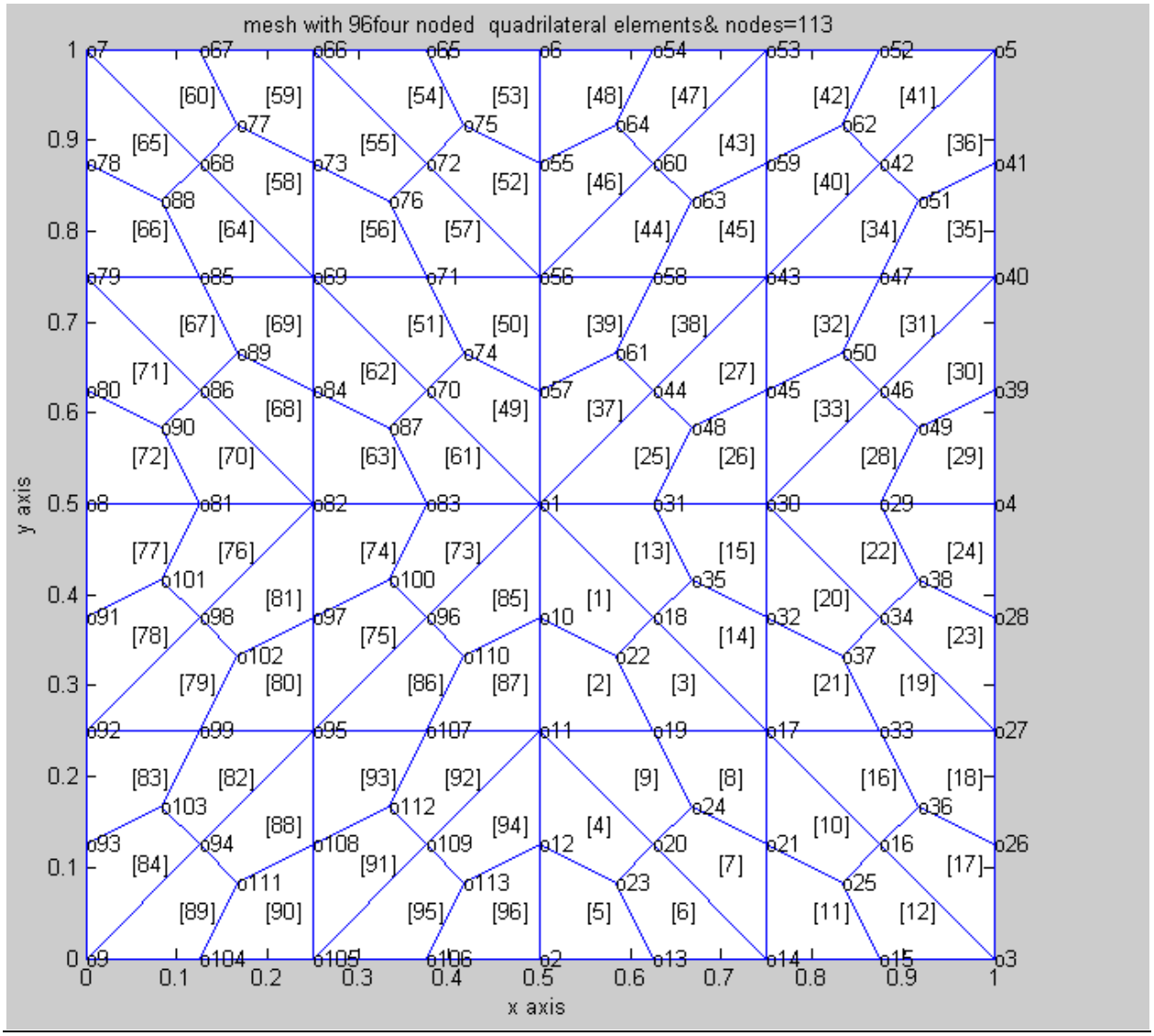

mesh with 96four noded quadrilateral elements& nodes=113
y axis
x axis

FIGURE-15

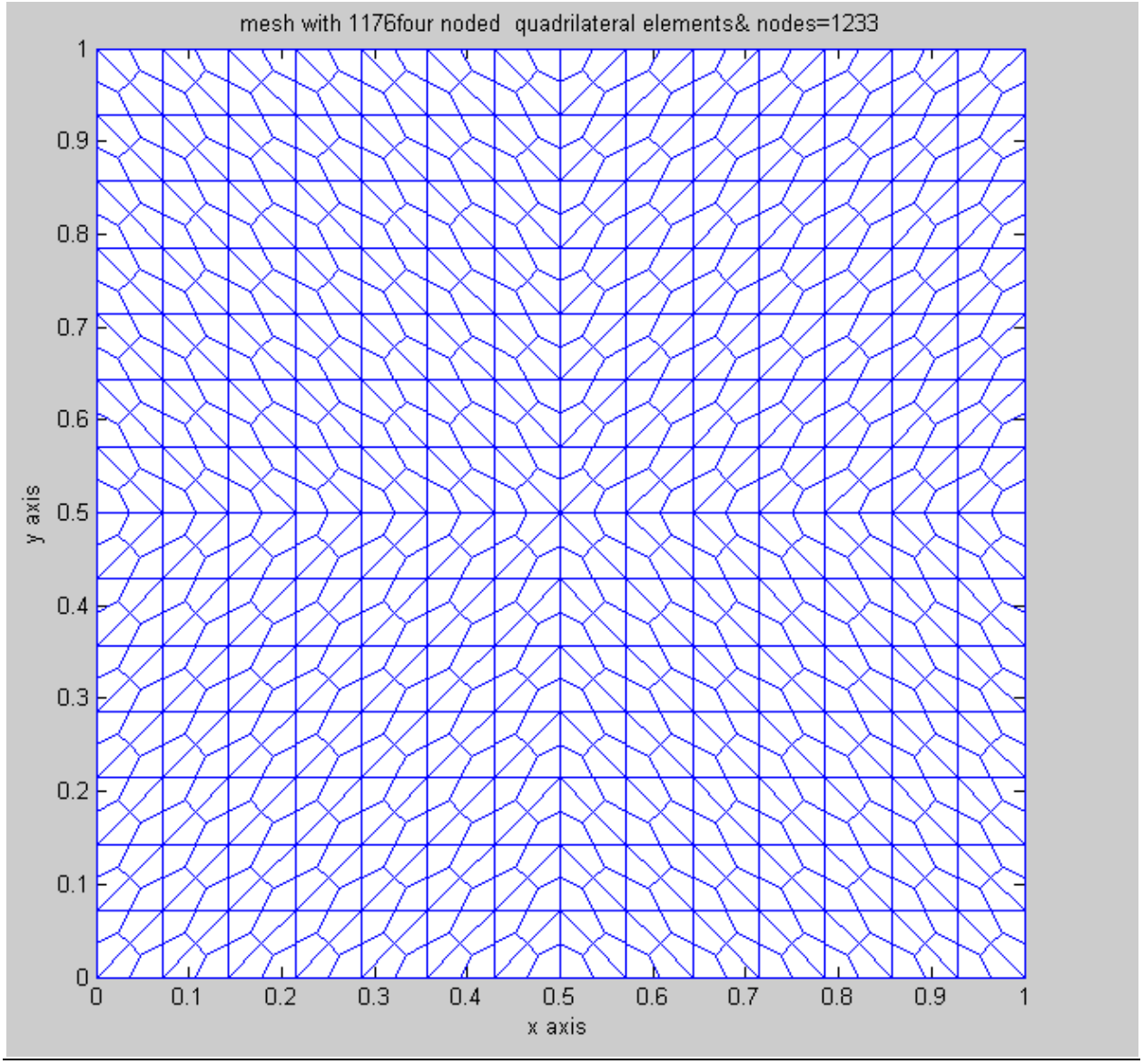

mesh with 1176four noded quadrilateral elements& nodes=1233
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-16

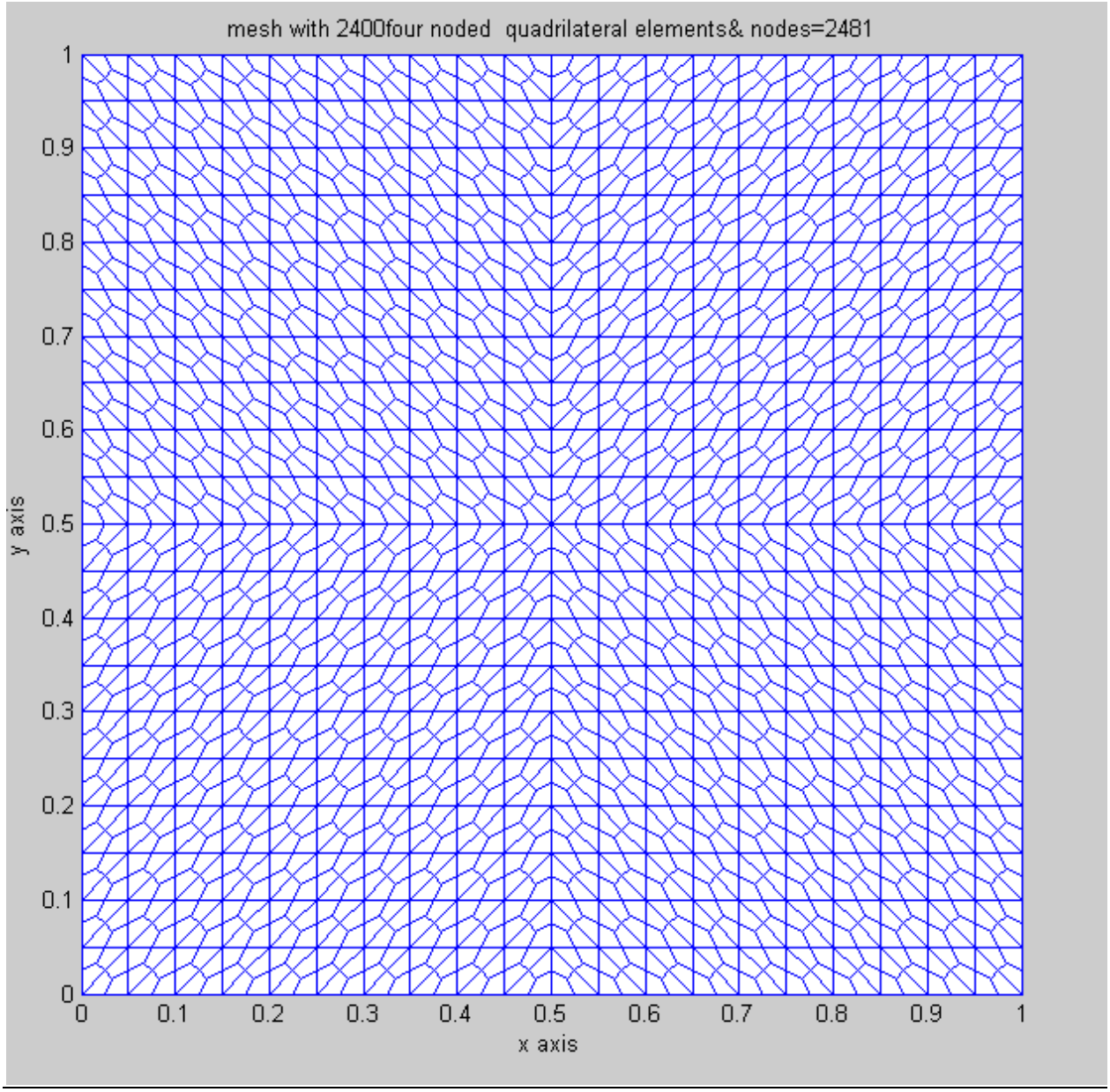

mesh with 2400four noded quadrilateral elements& nodes=2481
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-17

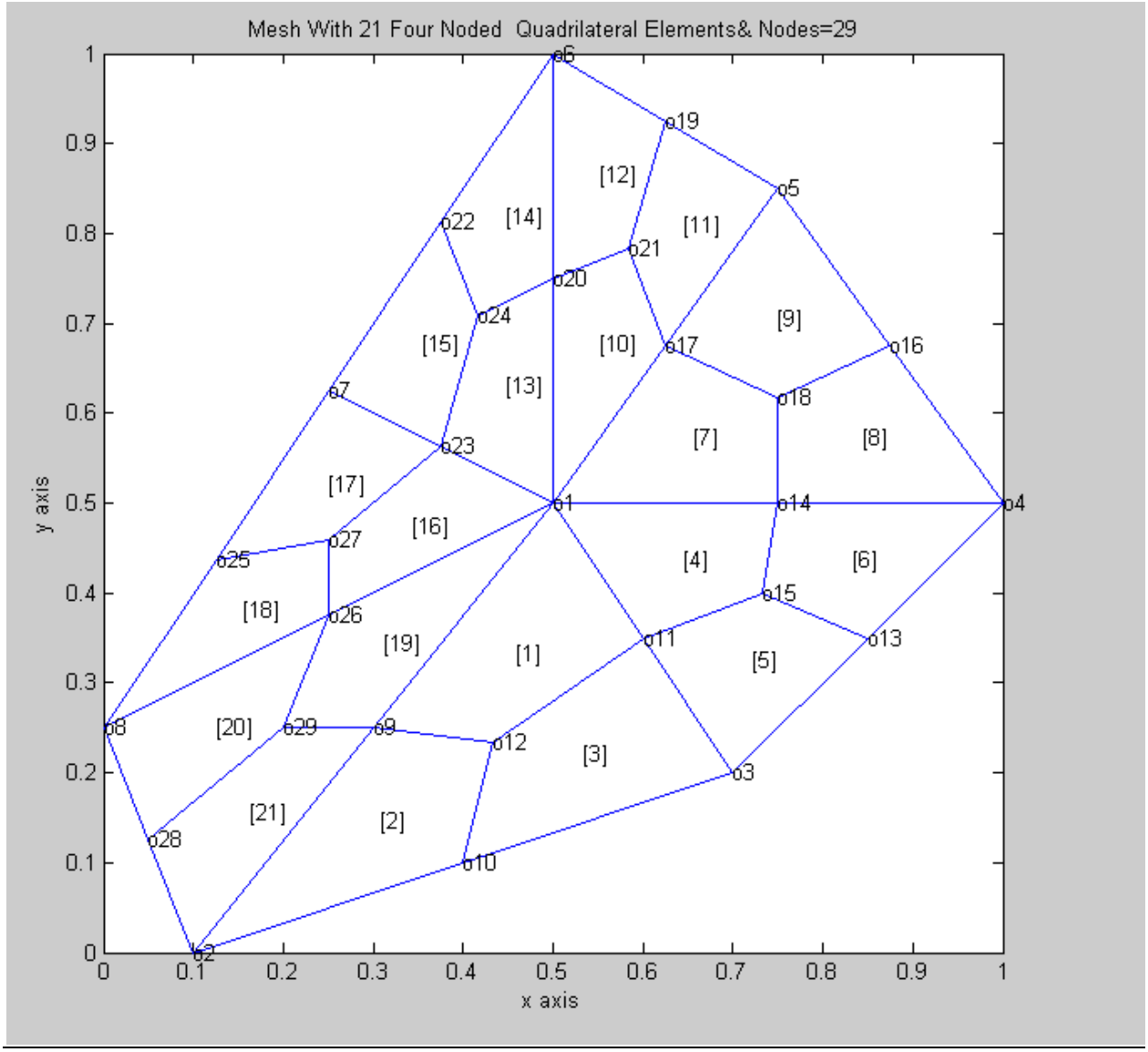

Mesh With 21 Four Noded Quadrilateral Elements& Nodes=29
y axis
x axis
o6
o19
o5
o22
o21
o20
o24
o17
o16
o7
o18
o23
o1
o14
o4
o27
o25
o26
o15
o11
o13
o8
o29
o9
o12
o3
o28
o10
o2
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[16]
[17]
[18]
[19]
[20]
[21]

FIGURE-18

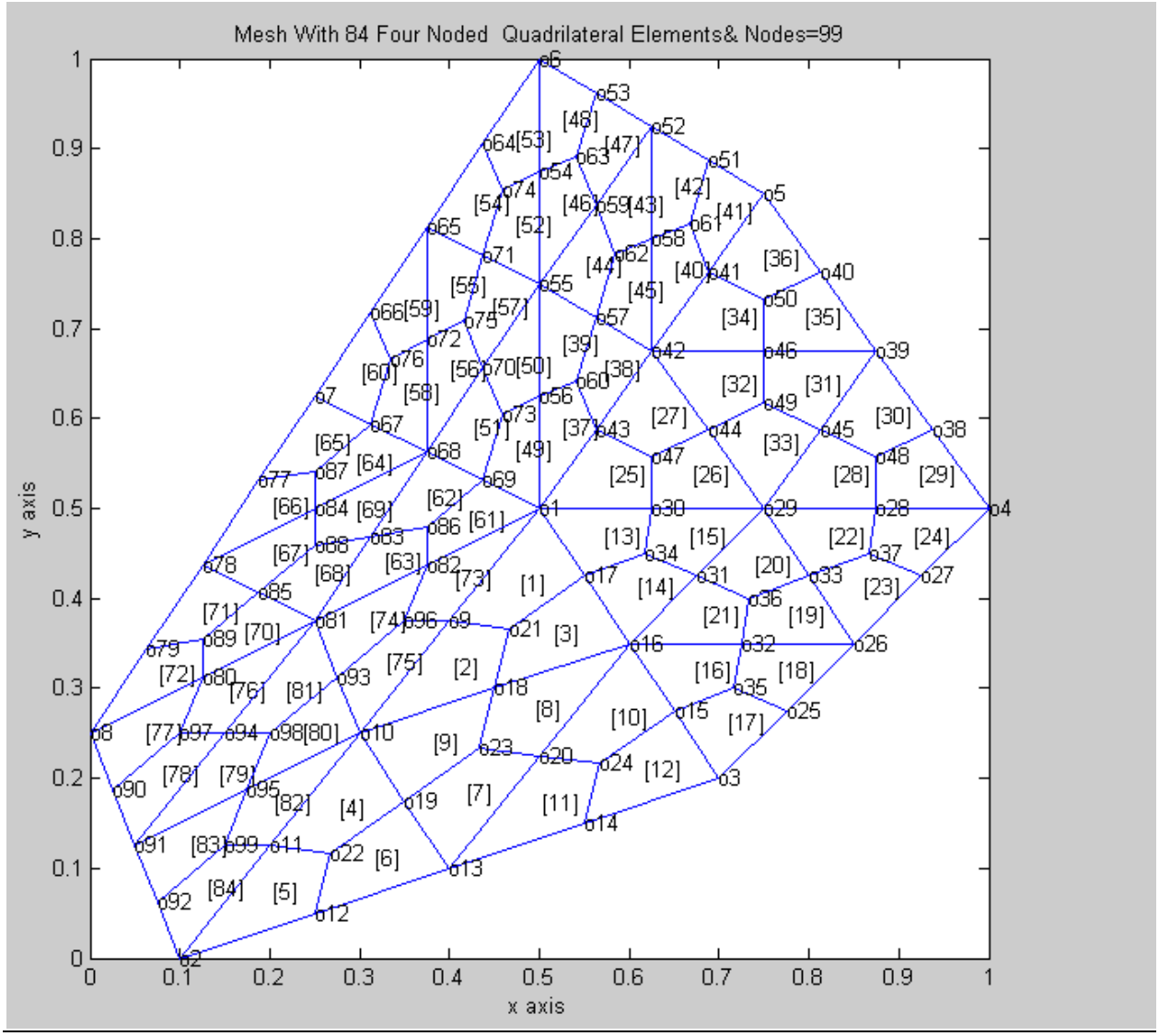

Mesh With 84 Four Noded Quadrilateral Elements& Nodes=99
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-19

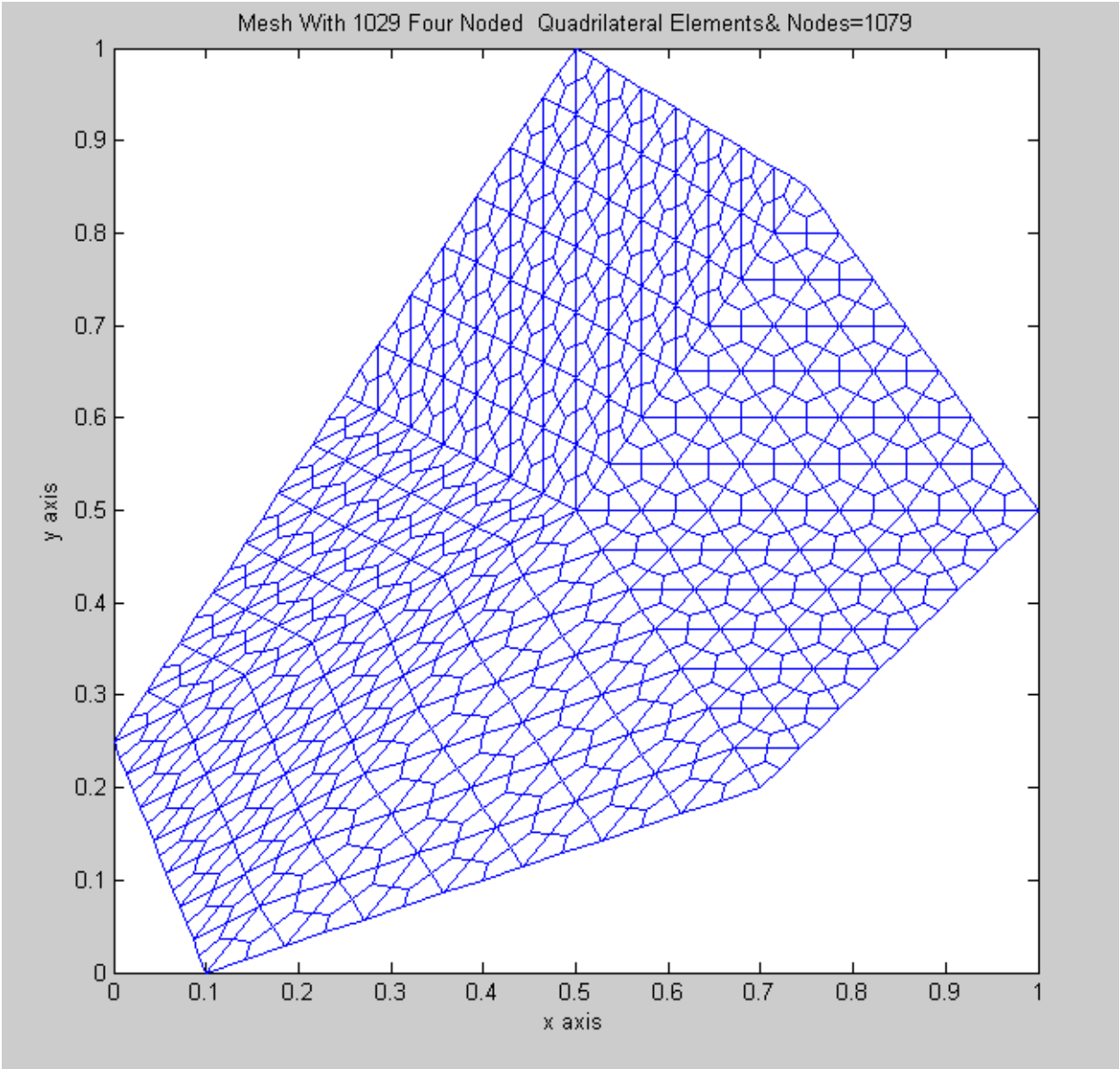

Mesh With 1029 Four Noded Quadrilateral Elements& Nodes=1079
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-20

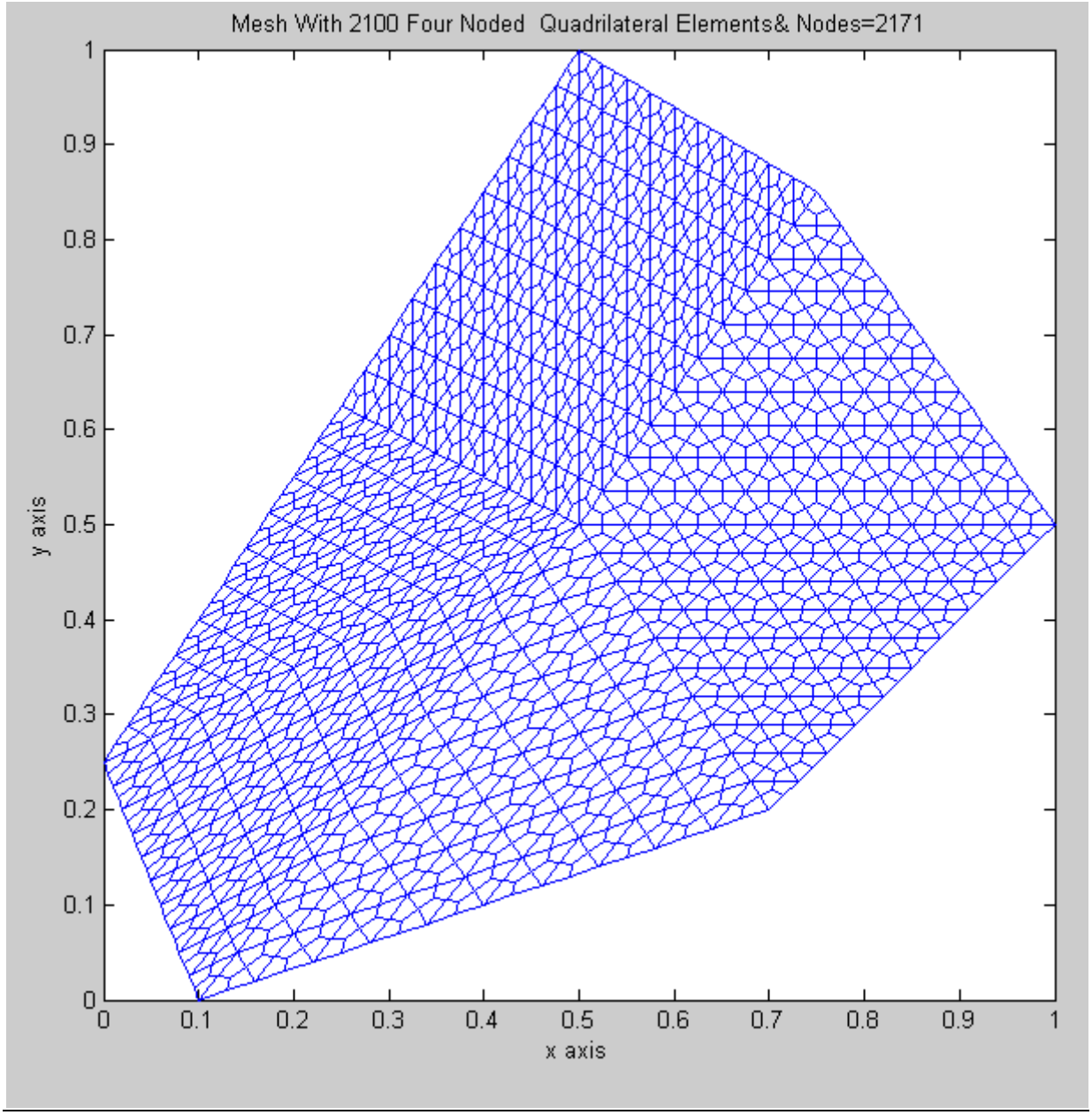
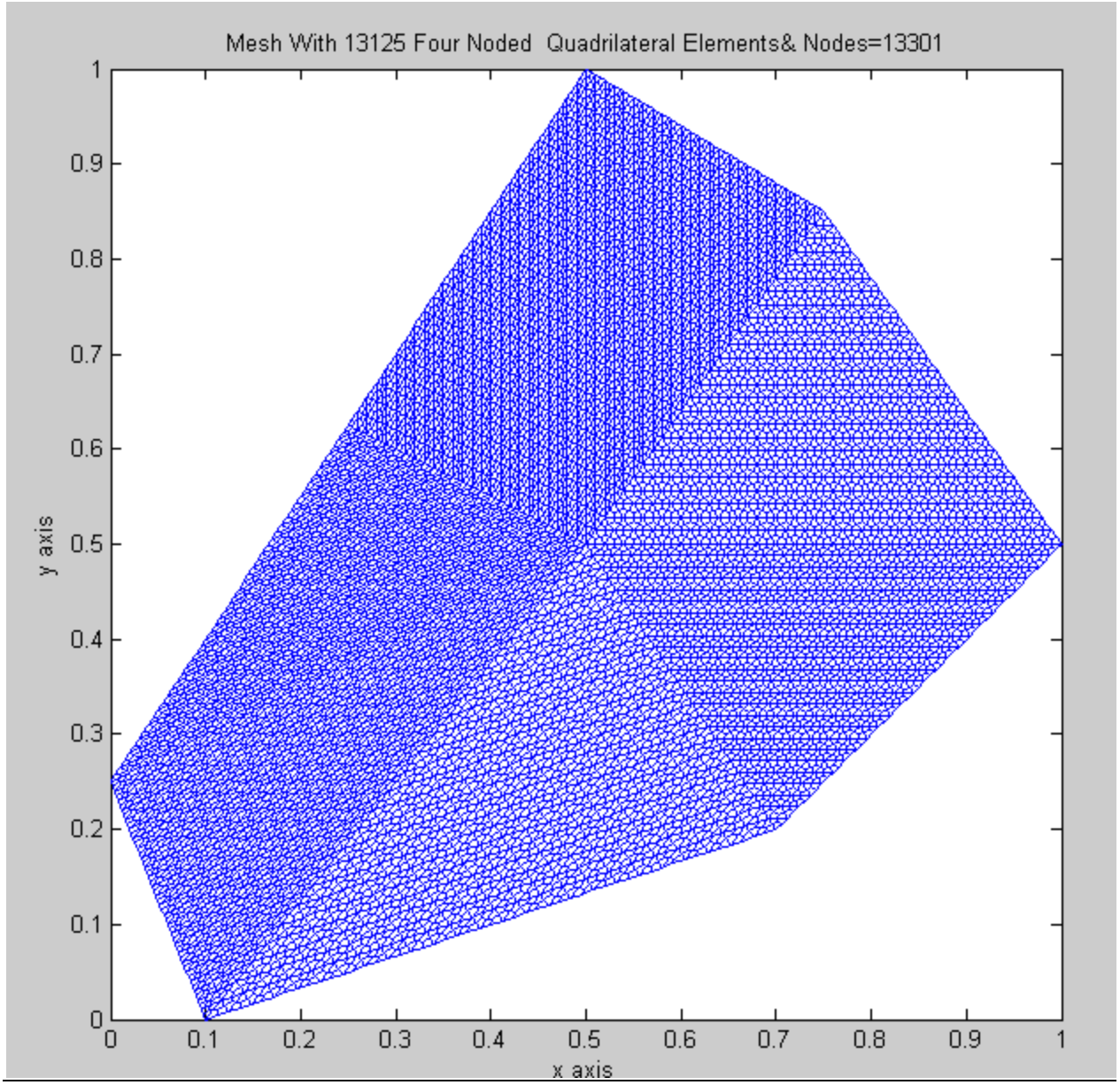

Mesh With 2100 Four Noded Quadrilateral Elements& Nodes=2171
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

FIGURE-21


Mesh With 13125 Four Noded Quadrilateral Elements& Nodes=13301
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1