www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 4 April, 2014 Page No. 5224-5272

A New Approach to an All Quadrilateral Mesh Generation over Arbitrary Linear Polygonal Domains for Finite Element Analysis

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

[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, Sai Vidya Institute of Technology, Rajanukunte,
Bangalore – 560064, India.
Email: 1) ashariprasad@yahoo.co.in
2) kallurvijayakumar@gmail.com

[e] 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 describes a scheme for finite element mesh generation of a convex, non-convex polygon and multiply connected linear polygon. We first decompose the arbitrary linear polygon into simple sub regions in the shape of polygons.These subregions may be simple convex polygons or cracked polygons.We can divide a nonconvex polygon into convex polygons and cracked polygons We then decompose these polygons 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. The proposed scheme has been realized as computer programs and a number of examples have been included to demonstrate the technique. Although the paper describes the scheme as applied to planar domains, it could be extended to three dimensions as well. 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, nonconvex, and multiply connected polygonal domains, uniform refinement

1. Introduction

The finite element method (FEM) is a numerical procedure that can be used to obtain solutions to a large class of engineering problems in stress analysis, heat transfer, electromagnetism and fluid flow etc. FEM has now become a staple for predicting and simulating the physical behaviour of scientific, engineering, medical and business applications.

The use of symbolic computation in support of computational methods in engineering and sciences is steadily growing. This is due to technical improvements in general purpose computer algebra systems (CAS) such as Mathematica and Maple, as well as the availability of inexpensive personal computers and laptops. Furthermore, Maple's symbolic maths tools box is available in widely used Matlab system. In finite element work, CAS tools can be used for a spectrum of tasks, formulation, prototyping, implementation, performance evaluation and automatic code generation.

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

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

vast and different techniques have been proposed [8], 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 triangles. The question arises as to what is the 'best' triangulation on a given set of points. One particular scheme, namely Delaunay triangulation [8], is considered by many researchers to be most suitable for finite element analysis. If the problem domain is a subset of the Cartesian plane, triangular or quadrilateral meshes are typically employed.

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

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

The uniform refinement in the above work is different for each triangle of the convex polygon ,if the size of the triangles is different In order to achieve uniform refinement for the whole convex polygon we may have to divide the given convex polygon into several convex polygons and the triangulate each of them and then assemble to achieve a uniform refinement for the convex polygon

In the present paper,we propose an advancement of the recent work[ ]. In the present scheme,we first propose to divide the given planar region into a number of convex polygons to achieve a uniform refinement for the convex polygon.We then propose a scheme to generate an all quadrilateral mesh for the cracked polygon.We have further shown that a nonconvex polygon can be formed by joining a cracked polygon and a convex polygon.We have also proposed a scheme to generate an all quadrilateral mesh for the multiply connected domains.In the formation of the above domain shapes,we have to assemble several convx polygons or a combination of cracked polygons and convex polygons

In section 2 of this paper, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of six node triangular elements. In section 3,we explain the procedure to split these triangles into quadrilaterals. In section 4,we have presented a method of piecing together of all triangular subregions and eventually creating an all quadrilateral mesh for the given convex polygonal domain. The matter of sections 2-4 is already explained in our recent work[ ].This is just necessary to generate an all quadrilateral mesh over a single convex polygon.In section 5,we have proposed the scheme to assemble the convex polygons to form a multiply connected domain and we have also shown that a nonconvex polygon can be formed by assembling a convx polygon and a cracked polygon

In section 6,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for the refinement of convex polygonal domain,a square duct, a cracked polygon and a nonconvex polygon.

In the following sections 2-4,we now present the scheme to generate an all quadrilateral mesh for a convex polygon. This is necessary to understand the mesh generation for the multiply connected domain,the cracked polygon and the nonconvex polygon.

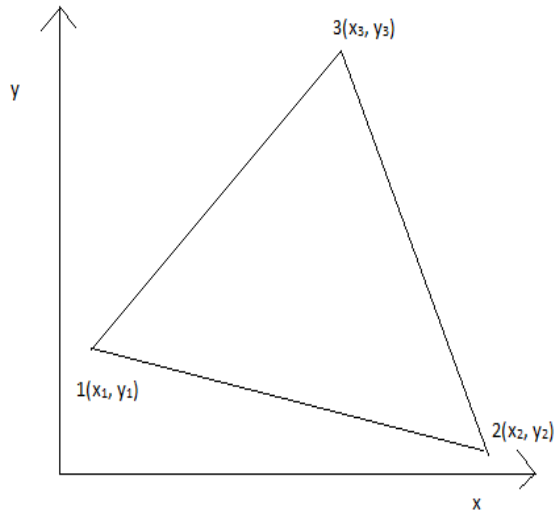
2. Division of an Arbitrary Triangle

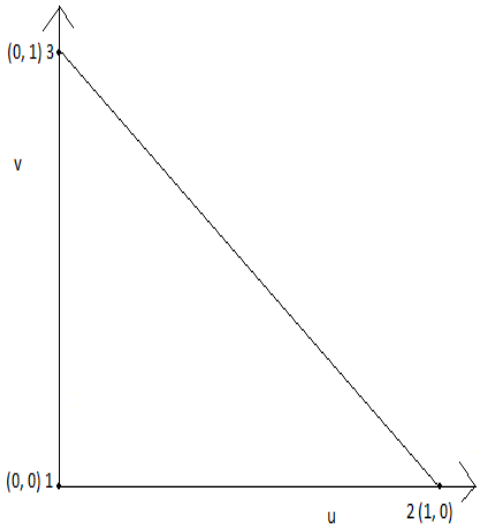
We can map an arbitrary triangle with vertices ($(x_i, y_i),\ i = 1, 2, 3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 1a, 1b. The necessary transformation is given by the equations.

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

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

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.

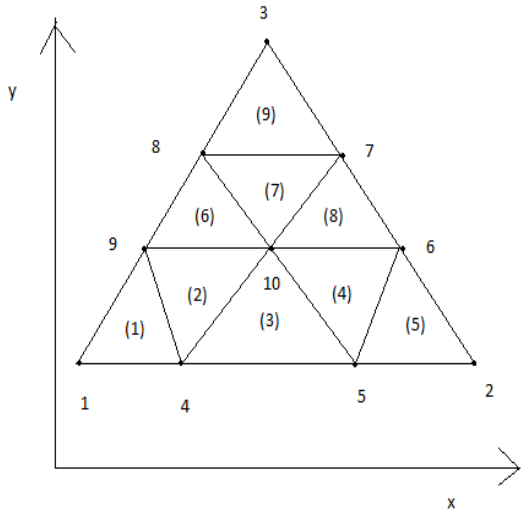


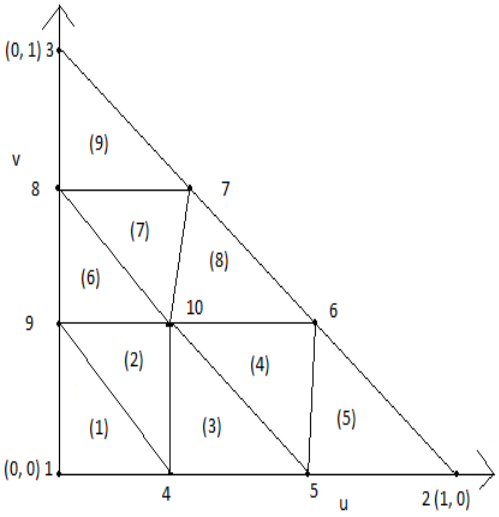


1.a 1. b

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

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

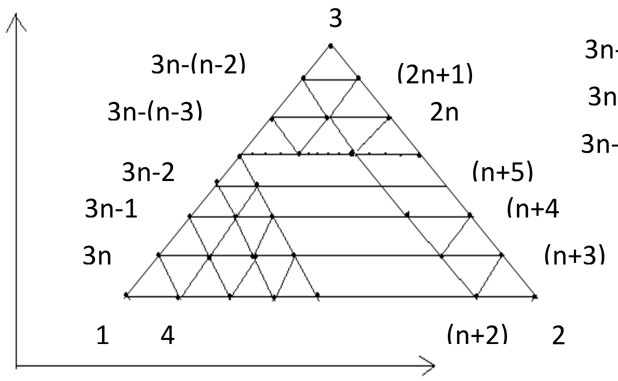




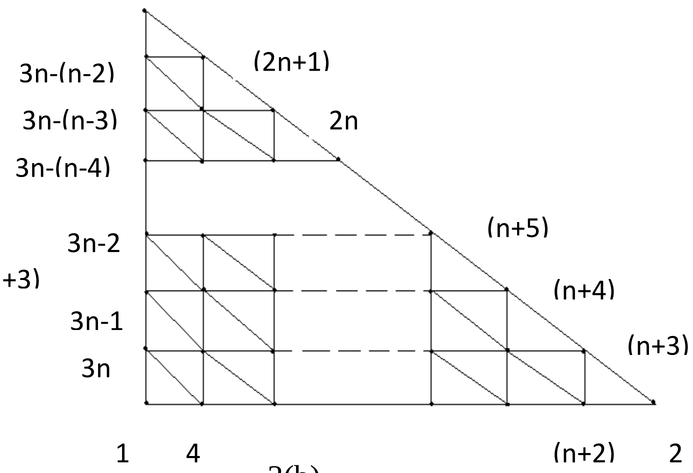
2(a) 2(b)

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

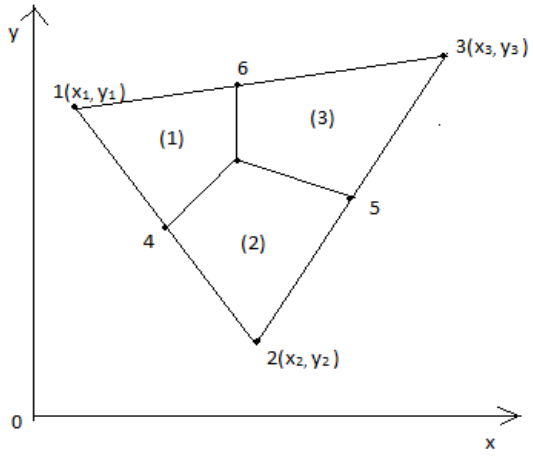
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \ldots\ldots\ldots\ldots\ldots\ldots\ldots, & (n+2), & 2 \\ 3n, & (3n+1), \ldots\ldots\ldots\ldots\ldots\ldots, & 3n+(n-2), & (n+3), & 0 \\ 3n-1, & 3n+(n-1), \ldots\ldots\ldots, & 3n+(n-2)+(n-3), & (n+4), & 0, \quad 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldot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

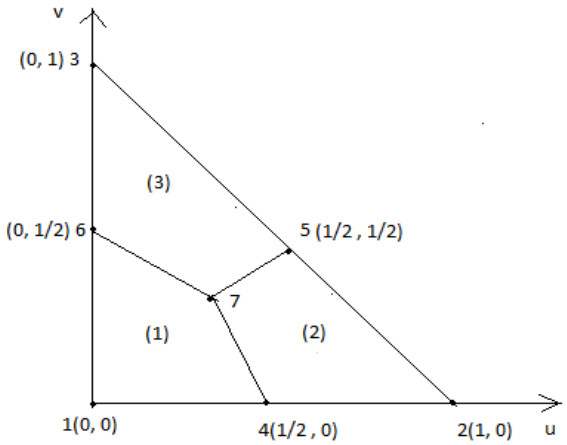
3. Quadrangulation of an Arbitrary Triangle

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



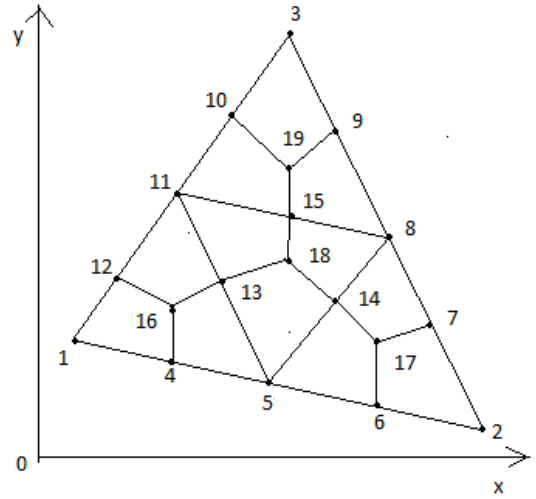


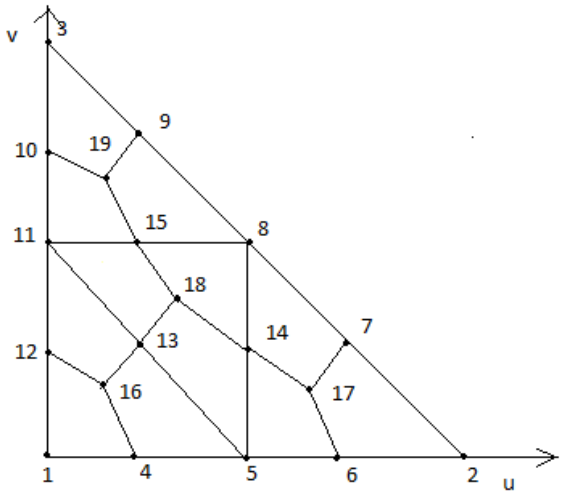
4(a) 4(b)

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

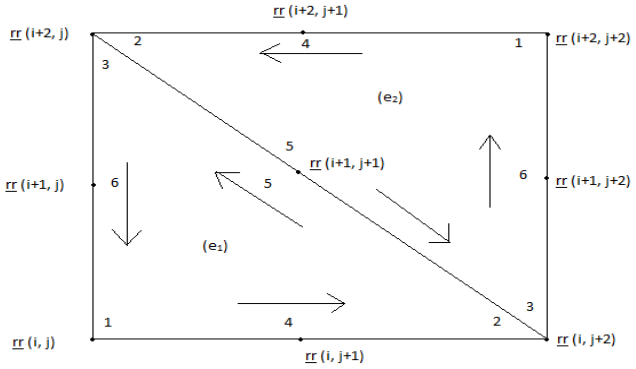


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

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$_1$, $\underline{rr}\ (i+1,\ j),\ \underline{\square\ \square}(\square,\ \square)\ ,\ \underline{\square\ \square}(\square,\ \square+1)$ >

$\square_{3\square_1-1}$ < c$_1$, $\underline{\square\ \square}\ (\square,\ \square+1),\ \underline{\square\ \square}(\square,\ \square+2)\ ,\ \underline{\square\ \square}(\square+1,\ \square+1)$ >

$\square_{3\square_1}$ < c$_1$, $\underline{\square\ \square}\ (\square+1,\ \square+1),\ \underline{\square\ \square}(\square+2,\ \square)\ ,\ \underline{\square\ \square}(\square+1,\ \square)$ >

and the nodal connectivity for the 3 quadrilaterals created in (e$_2$) are given as

$\square_{3\square_2-2}$ < c$_2$, $\underline{\square\ \square}\ (\square+1,\ \square+2),\ \underline{\square\ \square}(\square+2,\ \square+2)\ ,\ \underline{\square\ \square}(\square+2,\ \square+1)$ >

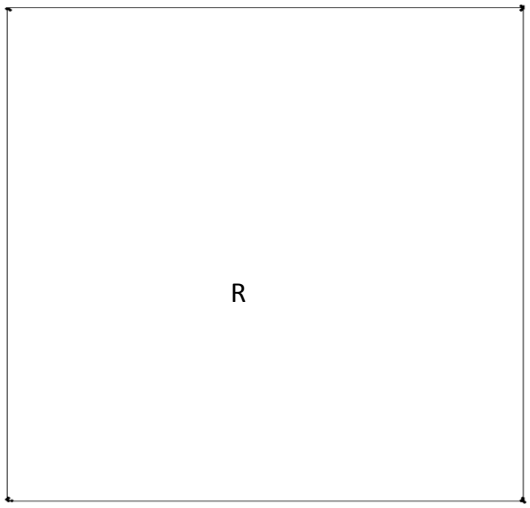
$\square_{3\square_2-1}$ < c$_2$, $\underline{\square\ \square}\ (\square+2,\ \square+1),\ \underline{\square\ \square}(\square+2,\ \square)\ ,\ \underline{\square\ \square}(\square+1,\ \square+1)$ >

$\square_{3\square_1}$ < c_2 , $\underline{\square\ \square}$ ($\square$ + *1*, $\square$ + *1*), $\underline{\square\ \square}$($\square$, $\square$ + *2*) , $\underline{\square\ \square}$($\square$ + *1*, $\square$ + *2*) > ---------- (5)

4. Quadrangulation of the Polygonal Domain

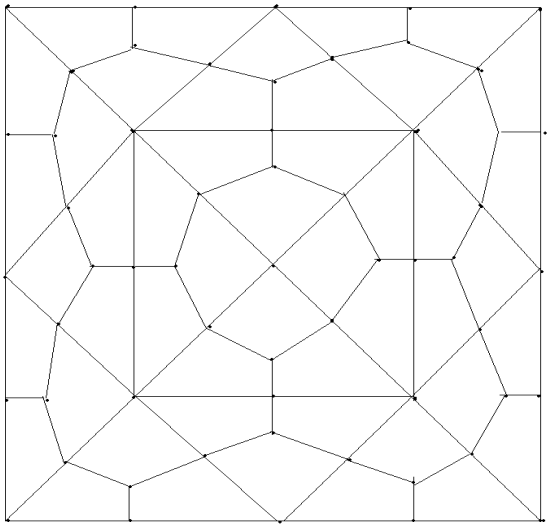
We can generate polygonal meshes by piecing together triangular with straight sides. Subsection (called LOOPs). The user specifies the shape of these LooPs by designating six coordinates of each LOOP

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



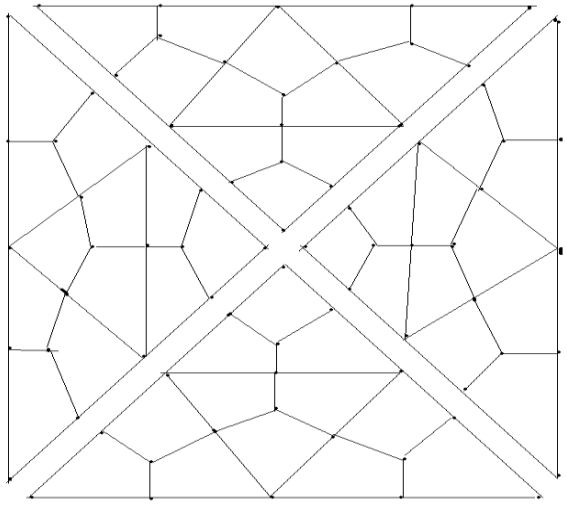
8 (a)

(i)Fig. 8(a) Region R to be analyzed

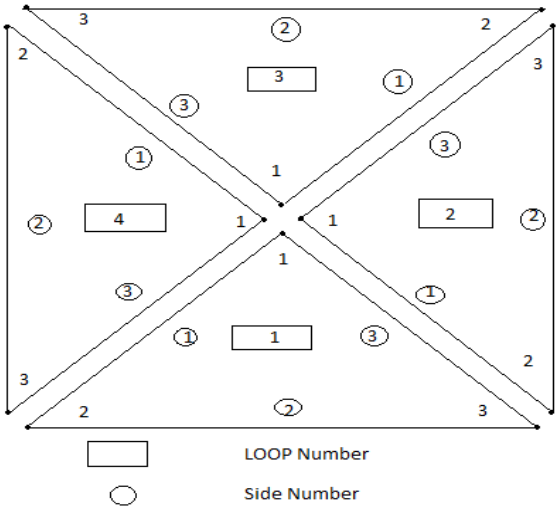


8 (b)

(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 ($\square$) and side of LOOP ($\square + 1$) 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 ($\square + 1$). This will be required for allowing the anticlockwise numbering for element connectivity

5. Mesh Generation for a Linear Polygonal Domain

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

5.1 Mesh Generation for a multiply connected domain

As an example,let us consider a square duct which is made-up of 12 squares. We divide each square into four triangles as shown in Fig. . Every triangle can be transformed into a right isosceles triangle which has two adjacent sides and a hypotenuse.In conformity with this notion,every triangle has two adjacent sides, say side 1-2 and side 1-3 ;and a hypotenuse,say,side 2-3. The square duct shown in Fig. ,has 48-triangles.We can express this as a sum, $\sum_{\square=1}^{\square=12} \square_{\square}^{\square} {}_{4} = \sum_{\square=1}^{\square=12} \sum_{\square=1}^{\square=4} \square_{\square}^{\square}$. We can set up a counter on triangles,thus for a fixed m,(m=4,for the present example) : $\square_{\square}^{\square} = \square_{(\square-1)\square+\square}$,$i=1,2,\ldots m$

Mesh Generation Over a Square Duct

(0, 2.0) (2.0, 2.0)

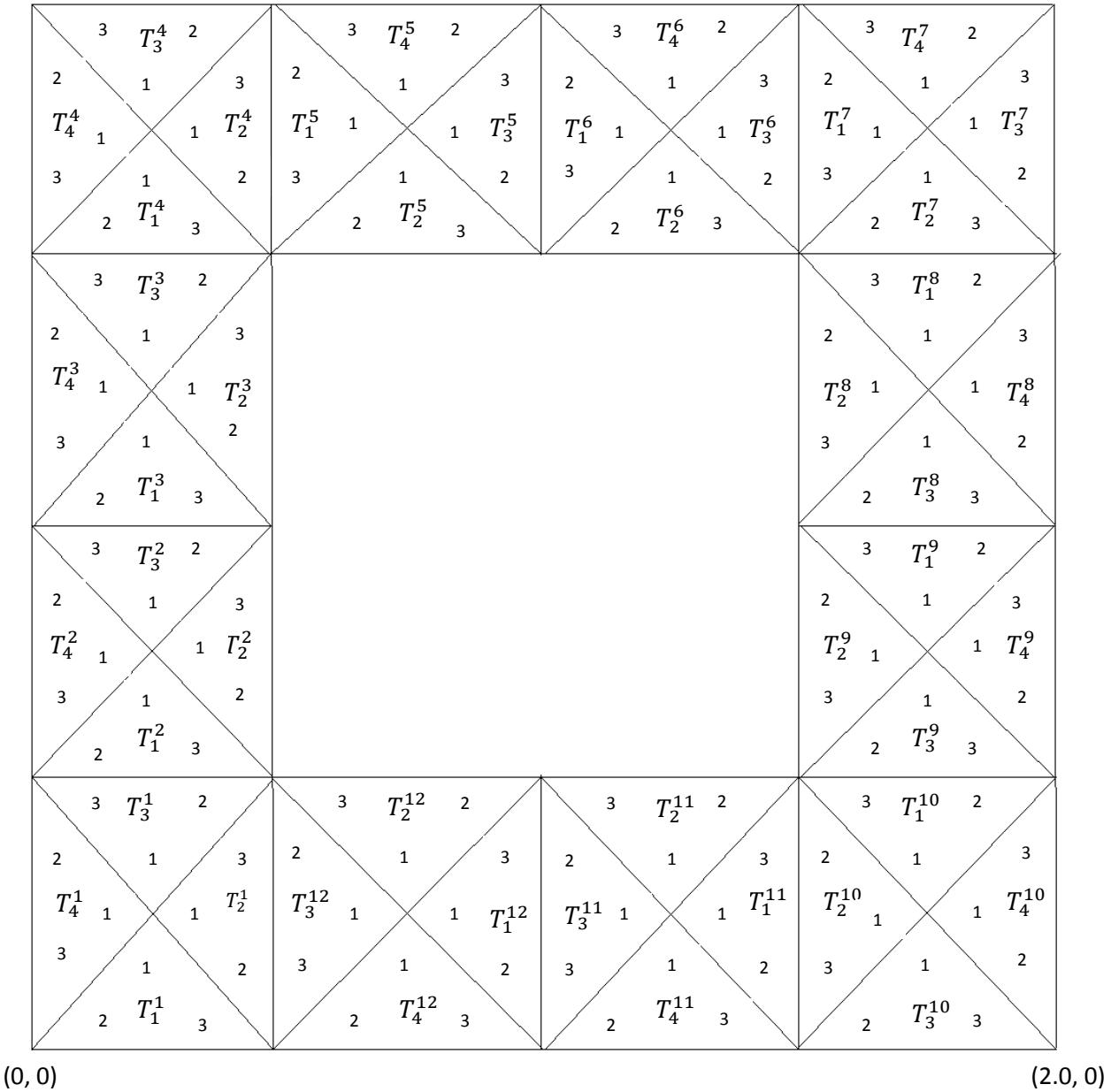


Fig. 9 A Square duct made-up of 12 squares $(\square_{\square}^{\square}, \square = \square, \square, - - --, \square\ \square)$ $\square\ \square\ \square\ \square\ \square_{\square}^{\square} = \sum_{\square=\square}^{\square} \square_{\square}^{\square}$ **,** $(\square = \square, \square, - - --, \square\ \square)$

5.1.1 FIRST PROGRAM

We now refer to our earlier work[25],wherein we have presented a program:

nodaladdresses_special_convex_quadrilaterals_trial.m

This program is very important as it is already tried and tested in the generation of an all quadrilateral finite element mesh for any convex polygon.This is based on the concept that by joining an interior point to the vertices of the boundary points,one can form as many triangles as there are sides out of a convex polygon.This program creates the element nodal connectivity for all the special convex quadrilaterals thus formed. We take this concept further and suitably modify this program to generate an all quadrilateral mesh for the linear polygonal domain.This program has to be called as many times as there are convex polygons made-up in the linear polygonal domain. The important features of the program which is modified to generate an all quadrilateral mesh for the linear polygonal domain are further explained in the following lines.

We have retained the basic scheme of our earlier programs[25] by the name mentioned therein..The names of the earlier set of programs are not changed but the contents are suitably modified.
We have preserved the capacity of earlier programs and they are made more potent and now they can be used to generate FEM meshes for a variety of applications. The basic idea is to extract all the information regarding the each triangle of the convex polygon.As a firt step in this direction,it is necessary to know node vectors along all the three edges of triangle,because this information will be shared by the neighbouring triangles.We have created the following vectors for insertion in **nodaladdresses_special_convex_quadrilaterals_trial.m .** We have defined the vectors.
Letting n1,n2,n3 to represent the node numbers of the three vertices of triangles of the current convex polygon,
edgen1n2(1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices n1 and n2(base edge)
edgen1n3(1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices n1 and n3(left edge)
edgen2n3(1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices n1 and n3(right edge)
(itri=1:nitri),nitri=number of triangles in the current convex polygon
We have made use of **global statement** to share information of these vectors with any triangle of a succeeding convex polygon. A assembly process is initiated by instructing the right edge of the first triangle in the current polygon to join with the right edge of any triangle in the immediate previous polygon.The information of all the three edges(base edge,left edge,right edge)of the immediately previous is made available to the current convex polygon by another set of vectors.
Letting N1,N2,N3 to represent the node numbers of the three vertices of triangles of the immediately previous convex polygon,
EDGEN1N2(1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices N1 and N2(base edge)
EDGEN1N3 (1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices N1 and N3(left edge)
EDGEN2N3 (1:n+1,itri)→ vector for triangle 'itri' joining nodal vertices N2 and N3(right edge)
(itri=1:nitri),nitri=number of triangles in the immediately previous convex polygon

5.1.2 SECOND PROGRAM

This program is called only once.It inputs coordinates of the coarse discretisation of the given problem domain as triangles. This data is input as ordered sequence of x-coordinate and y-coordinate of the nodal vertices.
We now refer to our earlier work[25],wherein we have presented a program: **polygonal_domain_coordinates.m**
This program calls the program **nodaladdresses_special_convex_quadrilaterals_trial.m .** as many times as ther are convex polygons made-up in the planar domain.This program shares the necessary data information with the first program through global statements.

5.1.3 THIRD PROGRAM

This program is also called only once. It inputs nodal connectivity data for the coarse discretisation of the given problem domain as triangles. This data is input as ordered sequence of element node numbers for the triangles made in the planar domain.One particular feature of the element nodal connectivity node numbers is that if the first node number is same for a group of elements then it is clear that these elements belong to the same convex polygon.This will be checked in this program and based on this information references to the different convex polygons will be taken up in the program **polygonal_domain_coordinates.m** A PROGRAM to check this same number concept is implemented in the program: **checksamenumbers.m**
All the above programs are implemented in the MATLAB programming language

5.2 Refinement of a convex polygon

In our earlier paper[ ], the generation of a convex polygon is made possible by joining a finite number of triangles which originated from a common node point,for example a convex polygon with linear boundary node numbers 2,3,4,5,6,7,8 and originating from node number 1,is considered. This convex polygon contains seven different triangles Letting $\square_{\square,\square,\square}$,to represent a triangle with nodal vertices: I,j,k in anticlockwise orientation, we see that the above convex polygon is madeup of the seven triangles $\square_{1,2,3}$,$\square_{1,3,4}$,$\square_{1,4,5}$,$\square_{1,5,6}$,$\square_{1,6,7}$,$\square_{1,7,8}$ and $\square_{1,8,2}$ .The propsed mesh generation scheme refines each of these

triangles by further division of their sides.If these triangles are madeup of same area ,then refinement of the convex polygon is uniform. Otherwise, we have to employ different strategies.One simple method is to create more than one convex polygon and join them to form an original polygon. In this paper we have used the above modified programs to assemble the three convex polygons to form the original convex polygon.This is shown in Fig.10.

A Uniform Refinement Scheme For a Convex Polygon.

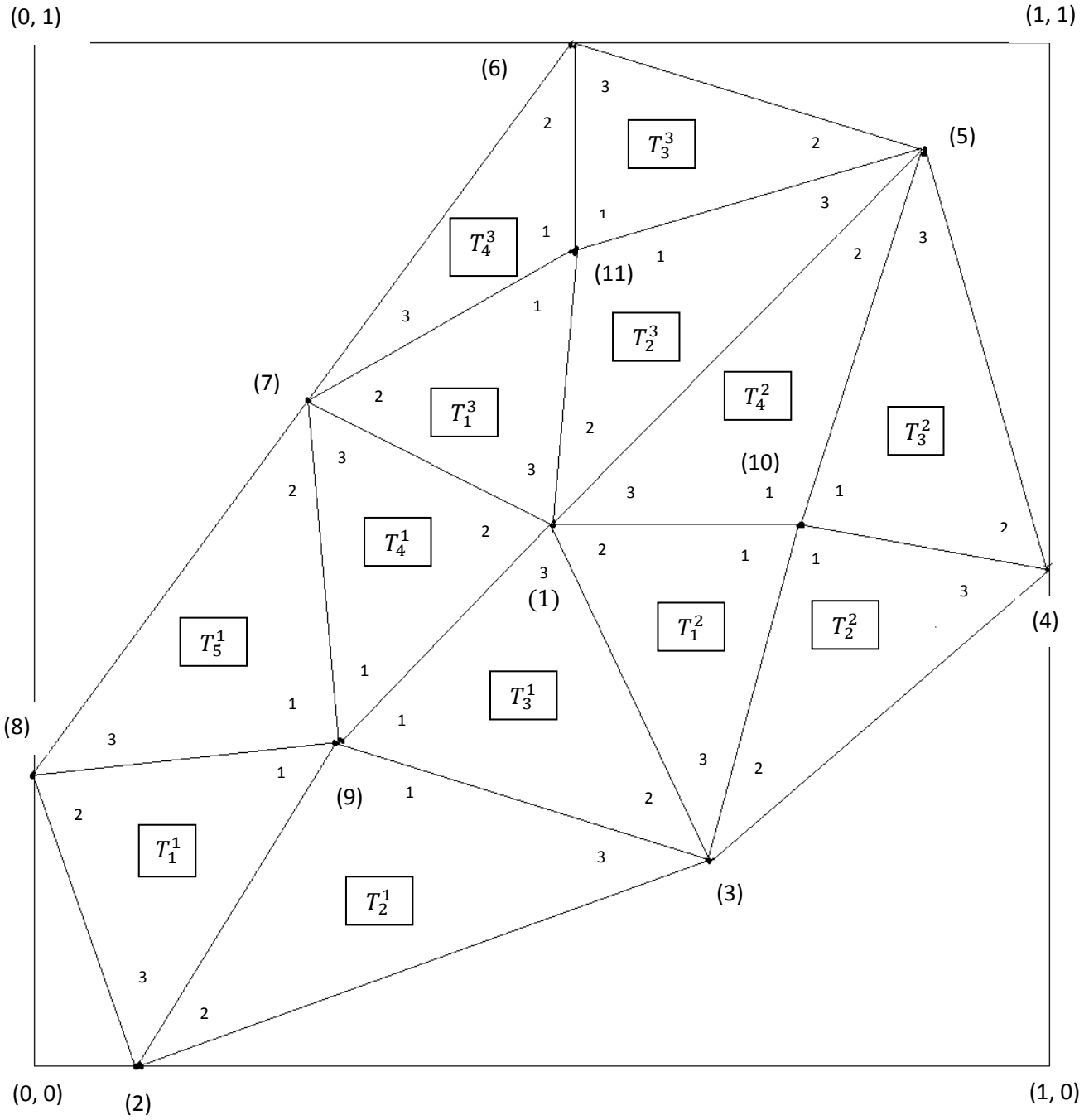


Fig. 10. An arbitrary convex polygon discretised into three convex polygons

$\square_{\square}^{\square} = \sum_{\square=\square}^{\square} \square_{\square}^{\square}$, $\square_{\square}^{\square} = \sum_{\square=\square}^{\square} \square_{\square}^{\square}$ **and** $\square_{\square}^{\square} = \sum_{\square=\square}^{\square} \square_{\square}^{\square}$

5.3 Mesh generation for a non-convex polygon

We can create a nonconvex polygon by joining a cracked polygon and a convex polygon. A Cracked polygon can be created from a convex polygon by deleting one or more triangles out of a convex polygon. In this paper we have used the above modified programs to join a convex polygon and a cracked polygon to form a non-convex polygon.

Conclusions

An automatic indirect quadrilateral mesh generator which uses the splitting technique is presented. This paper describes a scheme for finite element mesh generation of a convex,cracked polygon, non-convex polygon and multiply connected linear polygonal domain. This mesh generation is made fully automatic and allows the user to define the problem domain with minimum amount of input such as coordinates of boundary and element nodal connectivity in counter clockwise orientation for the coarse discretisation. Once this input is created, by selecting appropriate number of interior points of the linear polygonal domain, we form the triangular subdomains. These subdomains are then triangulated to generate a fine mesh of six node triangular elements. We have then proposed an automatic triangular to quadrilateral conversion scheme in which each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of the edges and a vertex at the barrycentre of the triangular element. This task is made a bit simple since a fine mesh of six node triangles is first generated. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this discretizes the given linear polygonal domain into all quadrilaterals, thus propogating a uniform refinement. This simple method generates high quality mesh whose elements confirm well to the requested shape by refining the problem domain. We have also appended MATLAB programs which provide the nodal coordinates, element nodal connectivity and graphic display of the generated all quadrilateral finite element mesh for a square duct, a convex, a non-convex and cracked convex polygons. We believe that this work will be useful for various applications in science and engineering.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

FIGURES ON MESH GENERATION

Figure. 11:SQUARE DUCT MESH-1

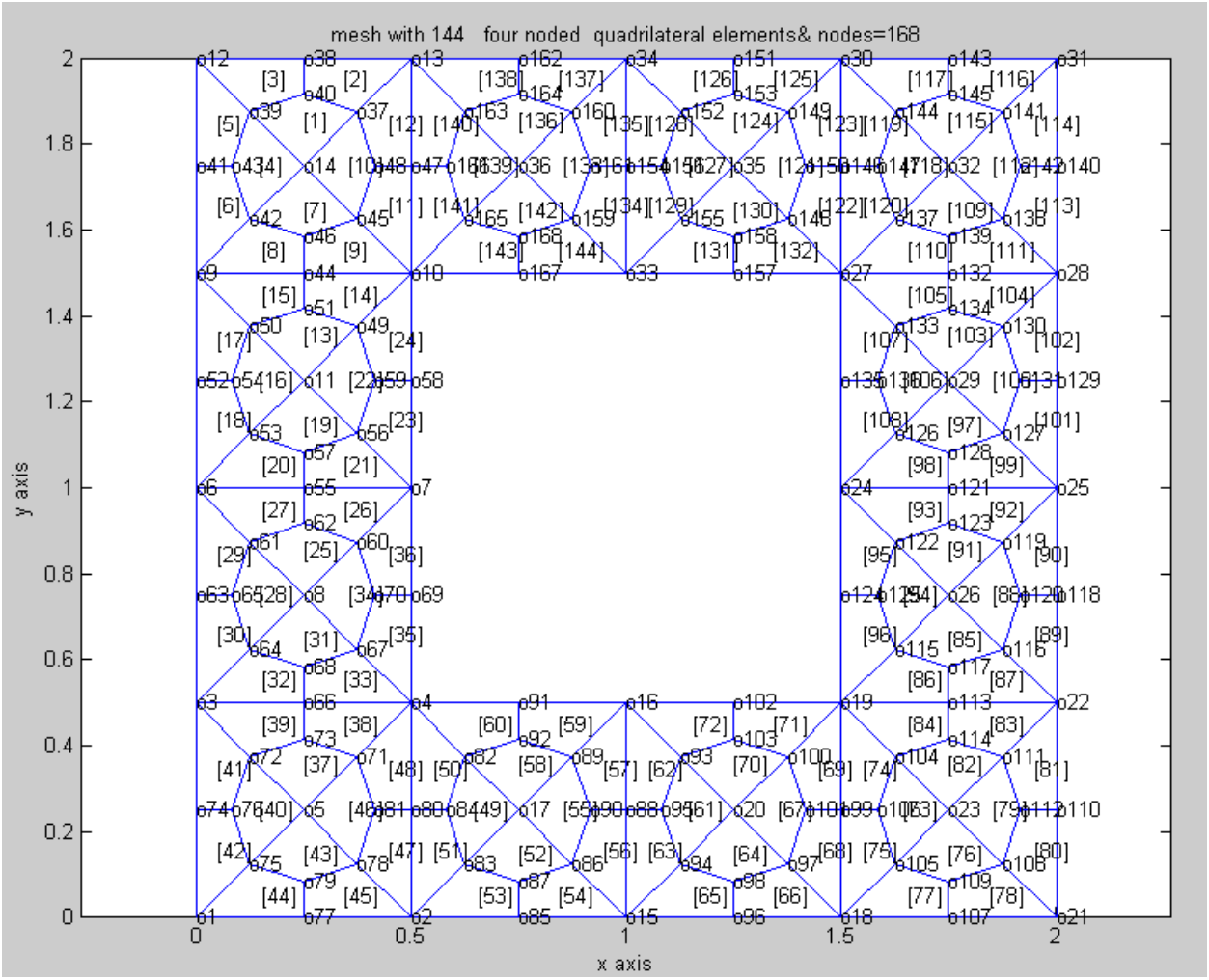


Figure 12:SQUARE DUCT MESH-2

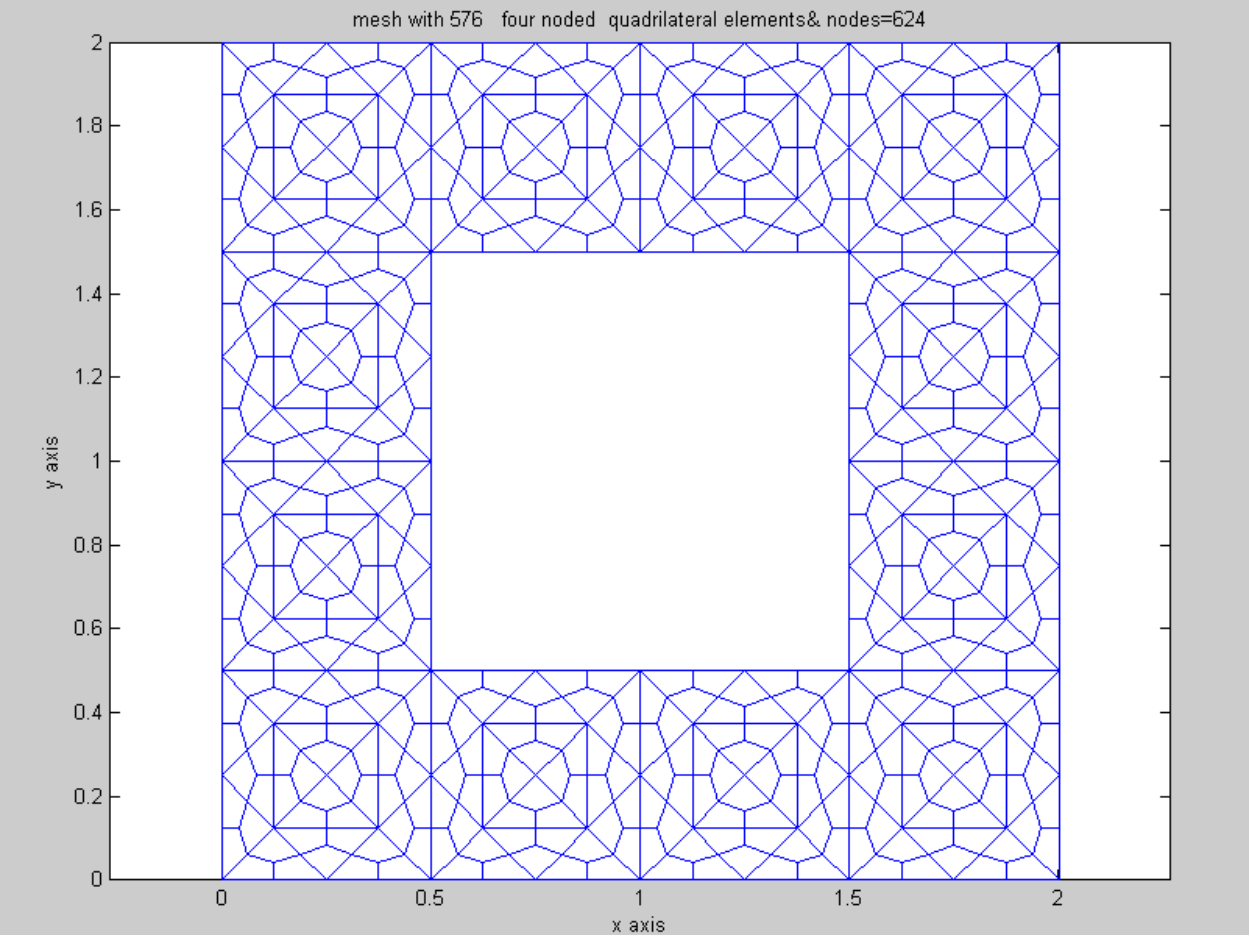


Figure 13:SQUARE DUCT MESH-3

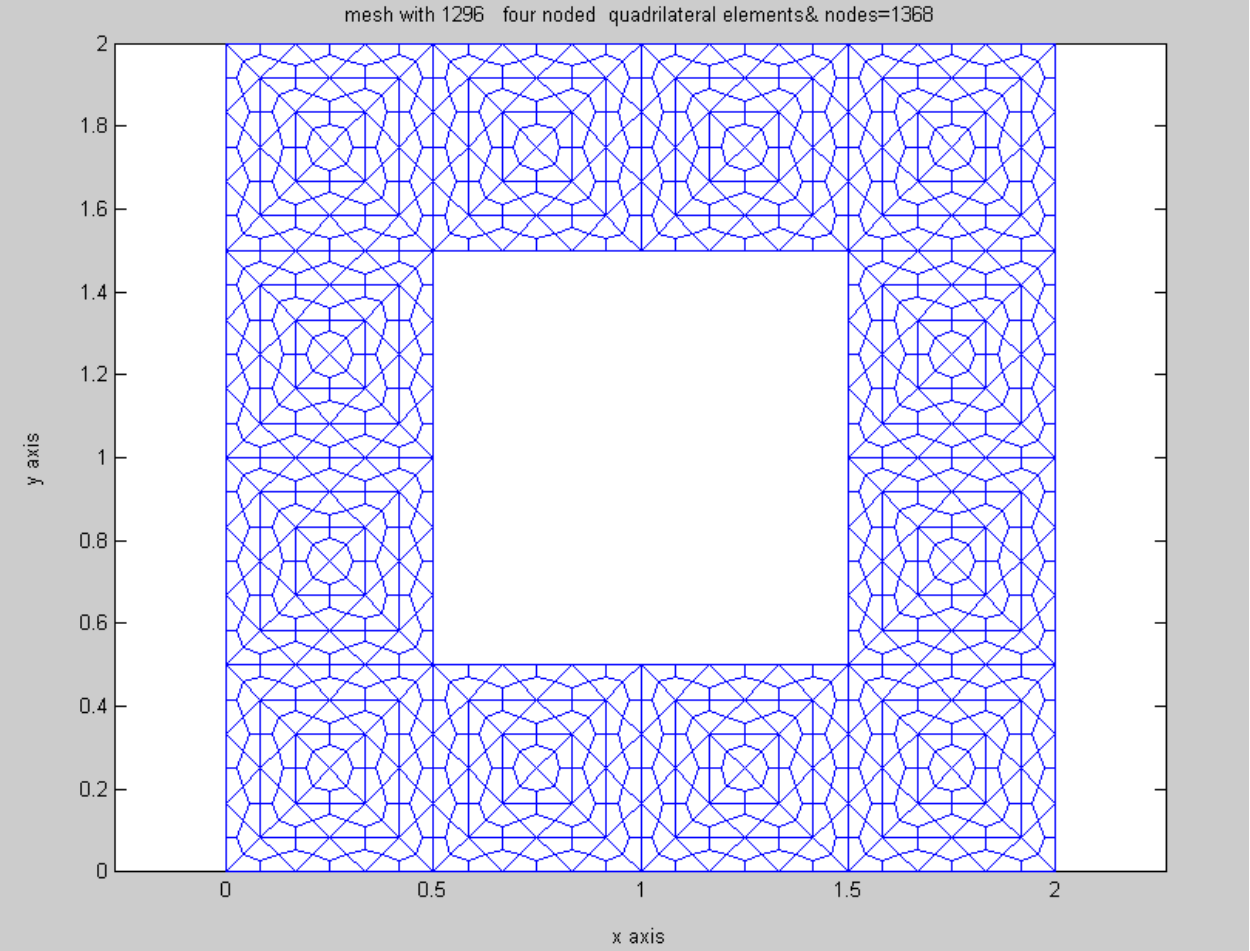


Figure 14:SQUARE DUCT MESH-4

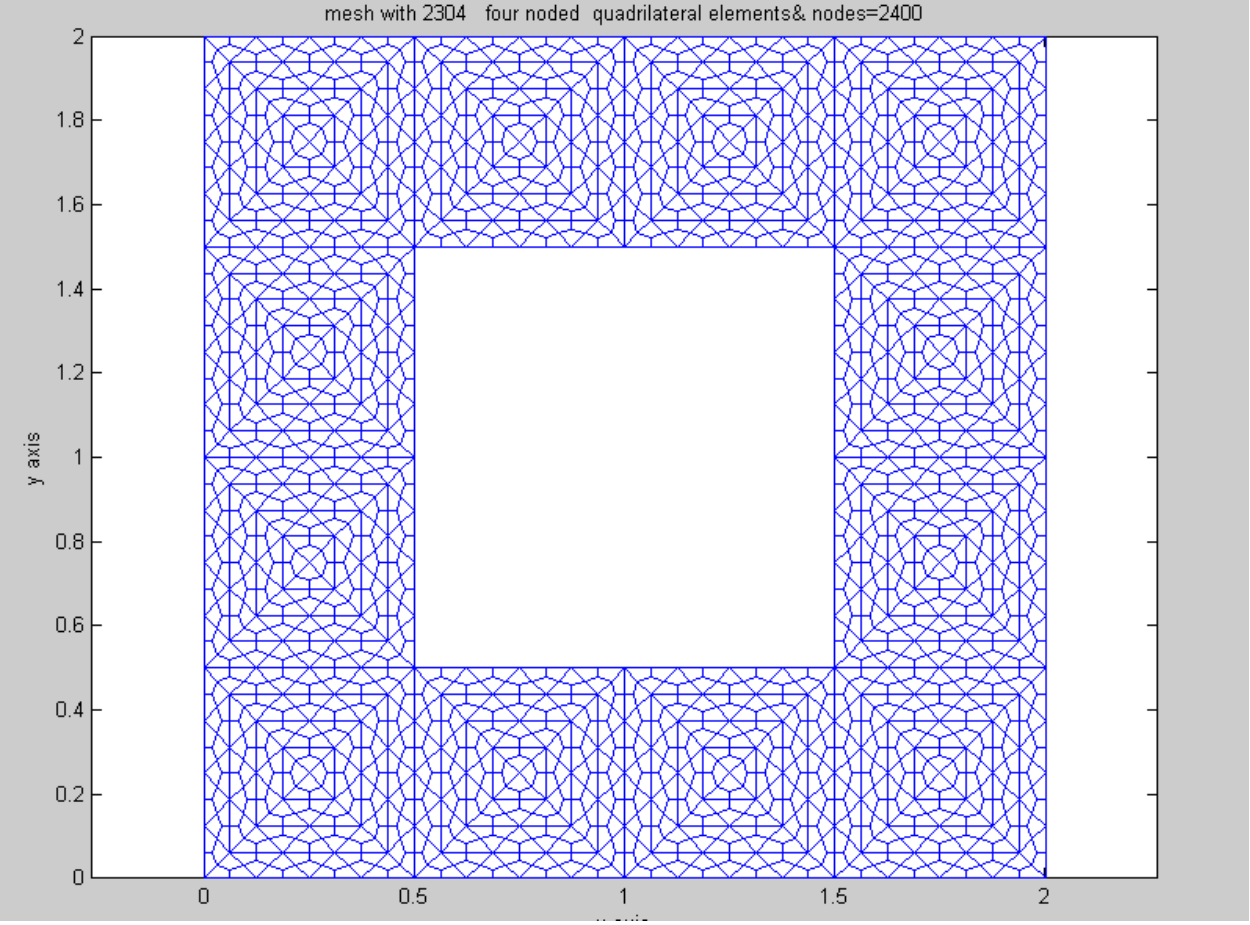


Figure 15:SQUARE DUCT MESH-5

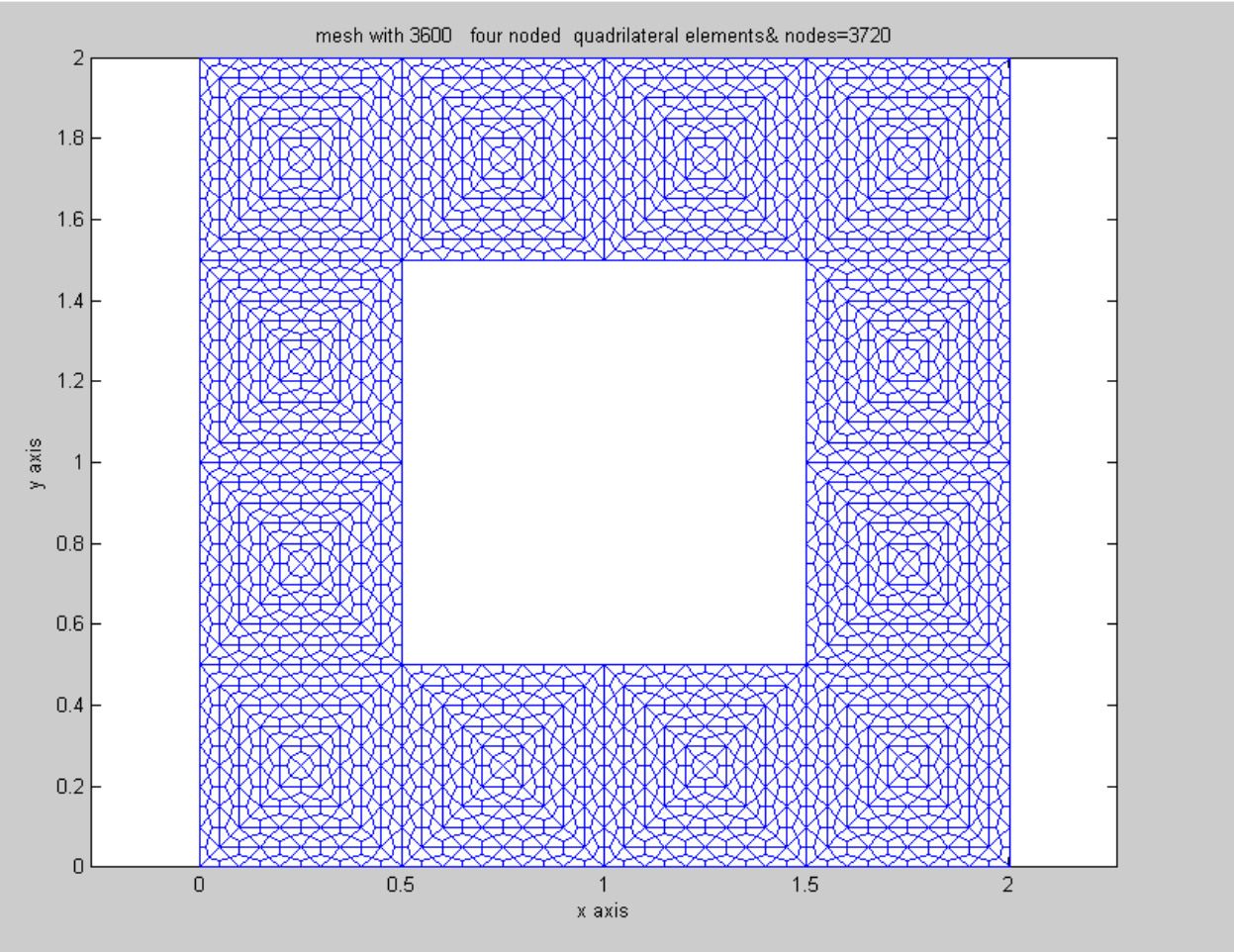


Figure 16: Cracked Convex Polygon Mesh-1

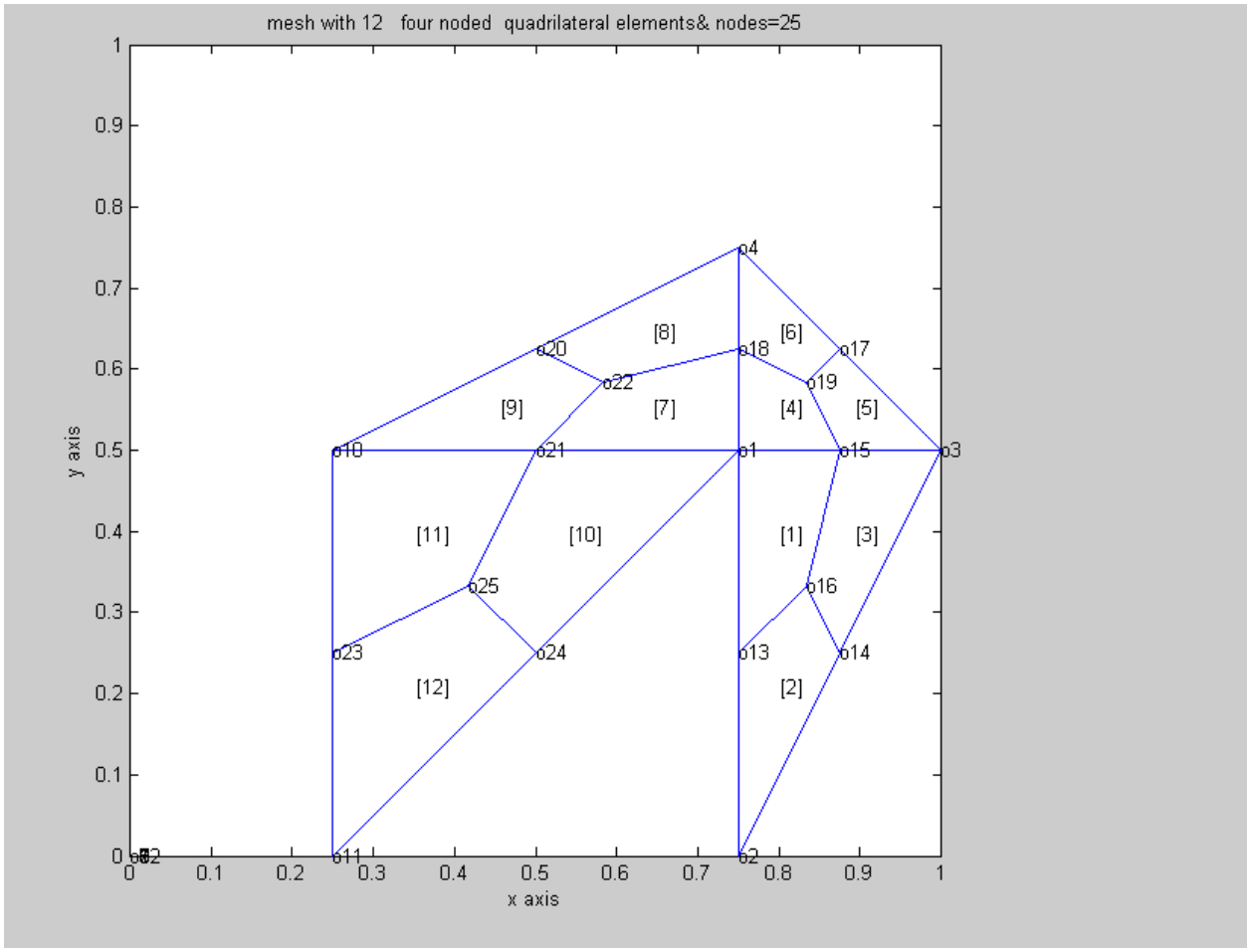


Figure 17: Cracked Convex Polygon Mesh-2

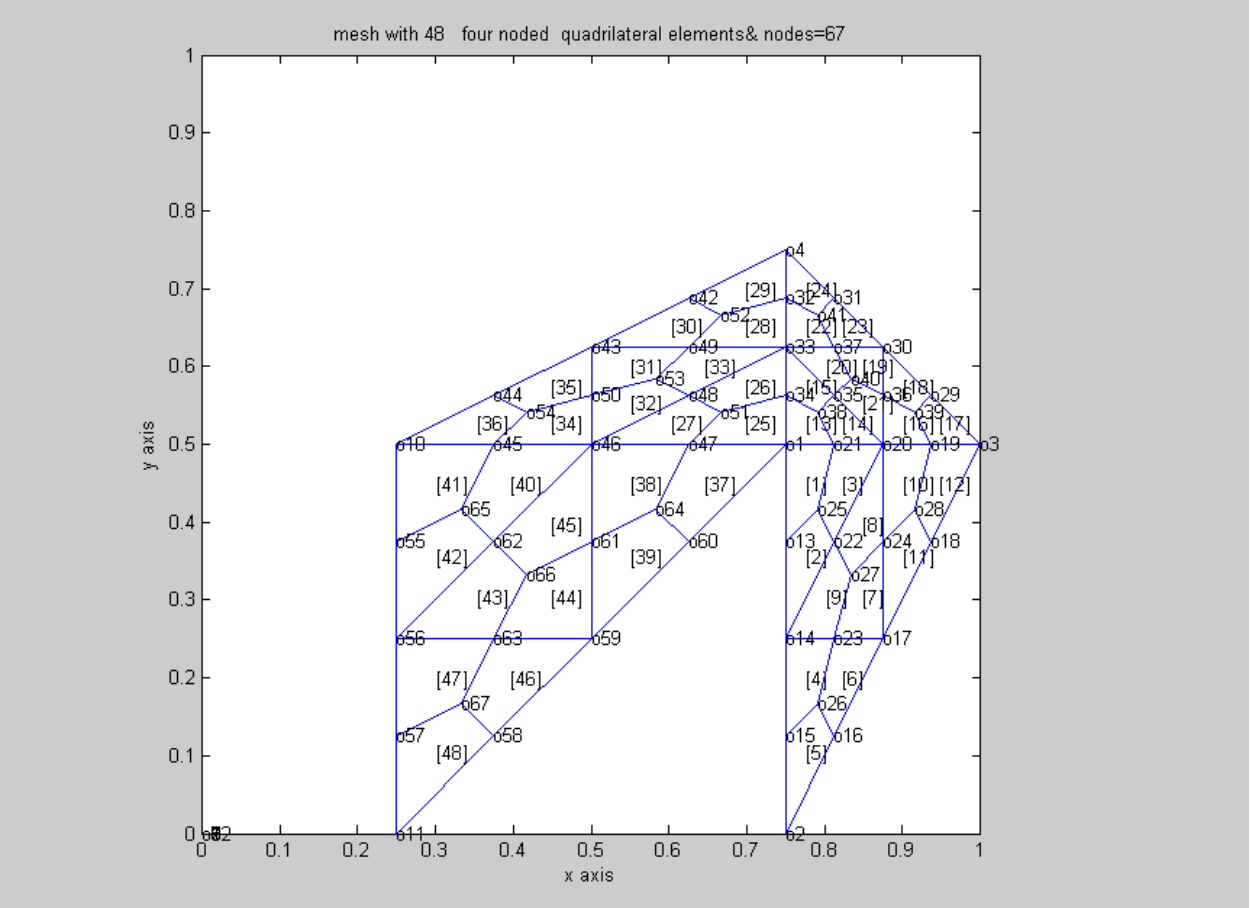


Figure 18: Cracked Convex Polygon Mesh-3

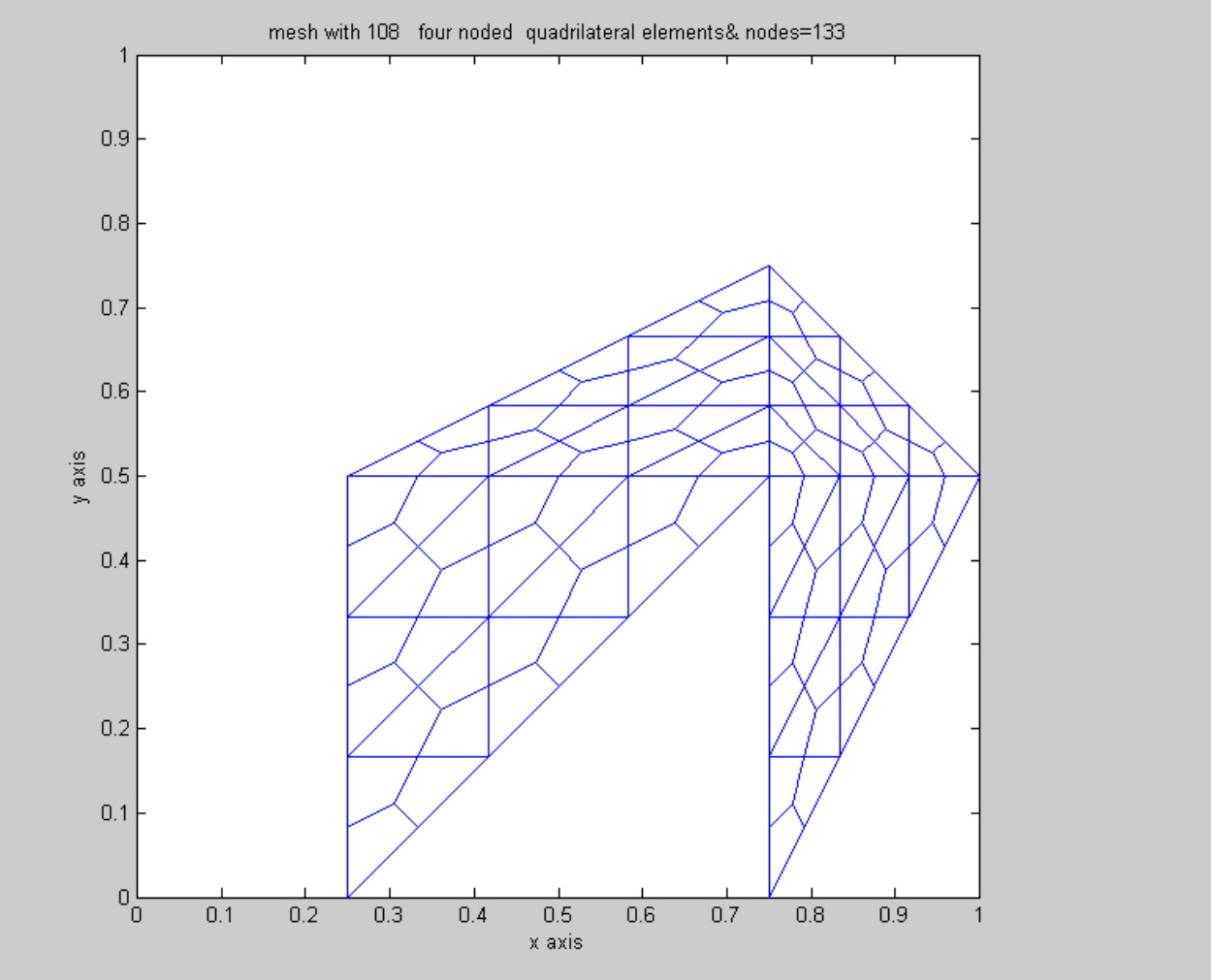


Figure 19: Cracked Convex Polygon Mesh-4

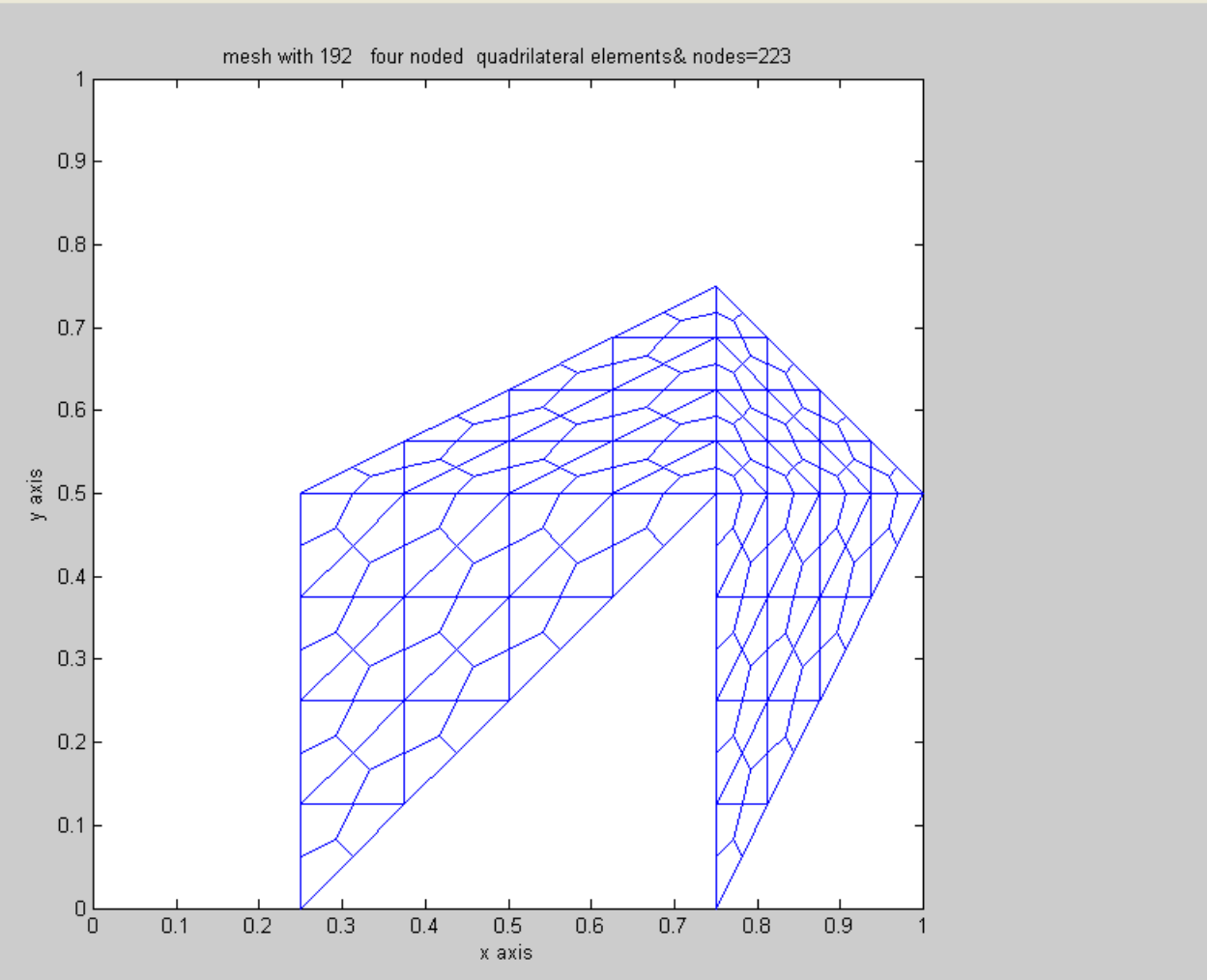


Figure 20: Cracked Convex Polygon Mesh-5

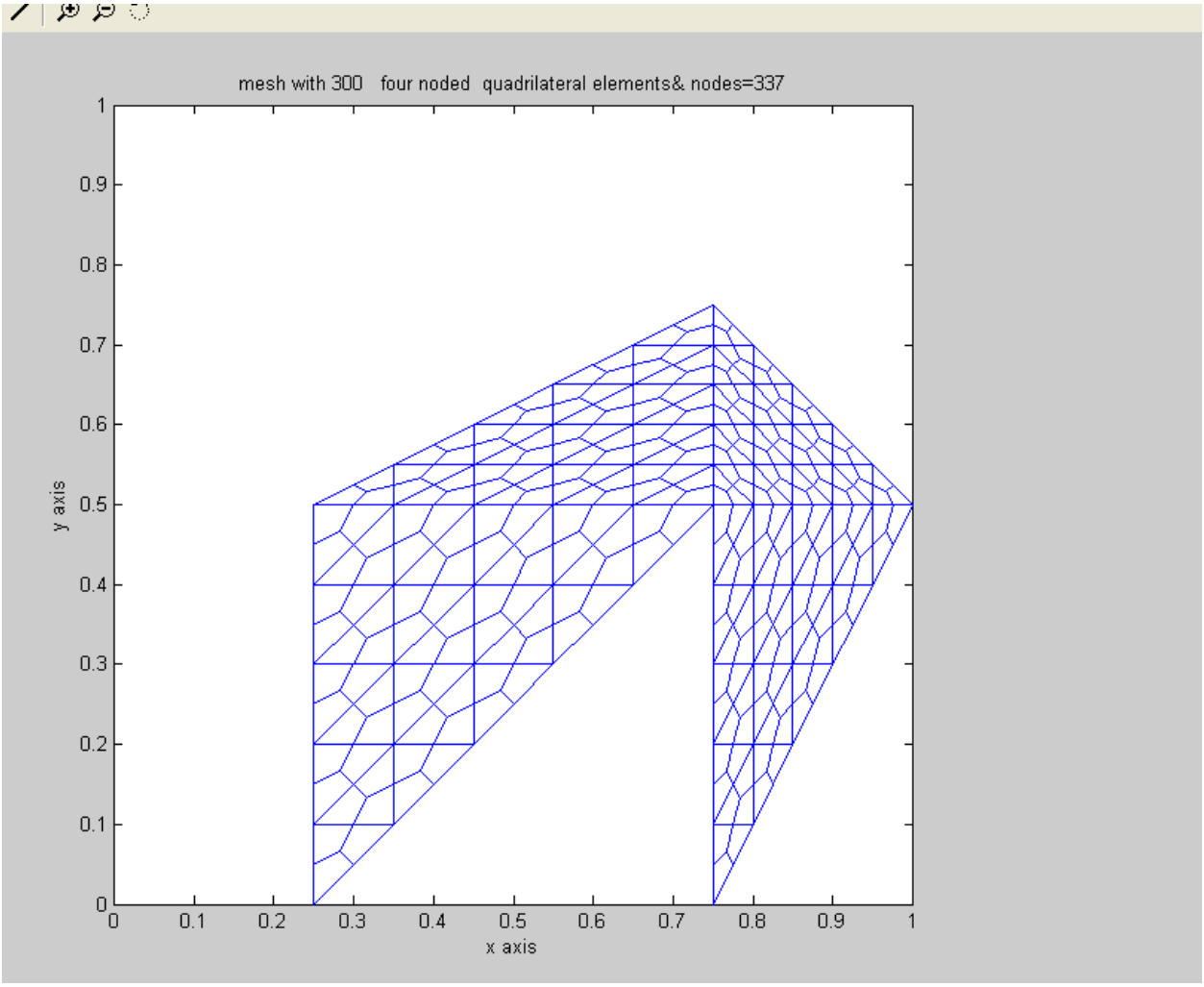


Figure 21:NONCONVEX POLYGON MESH-1

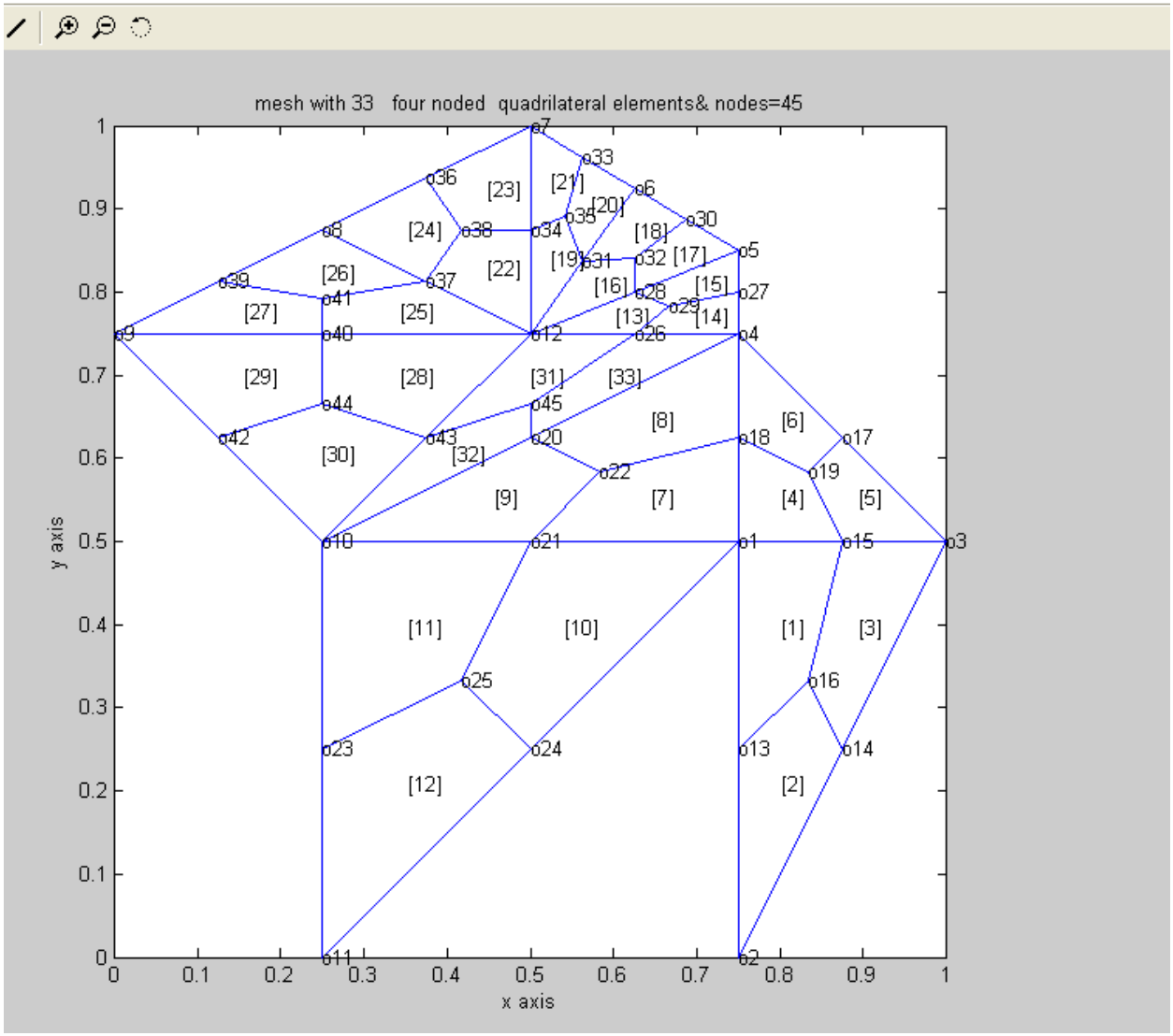


Figure 22:NONCONVEX POLYGON MESH-2

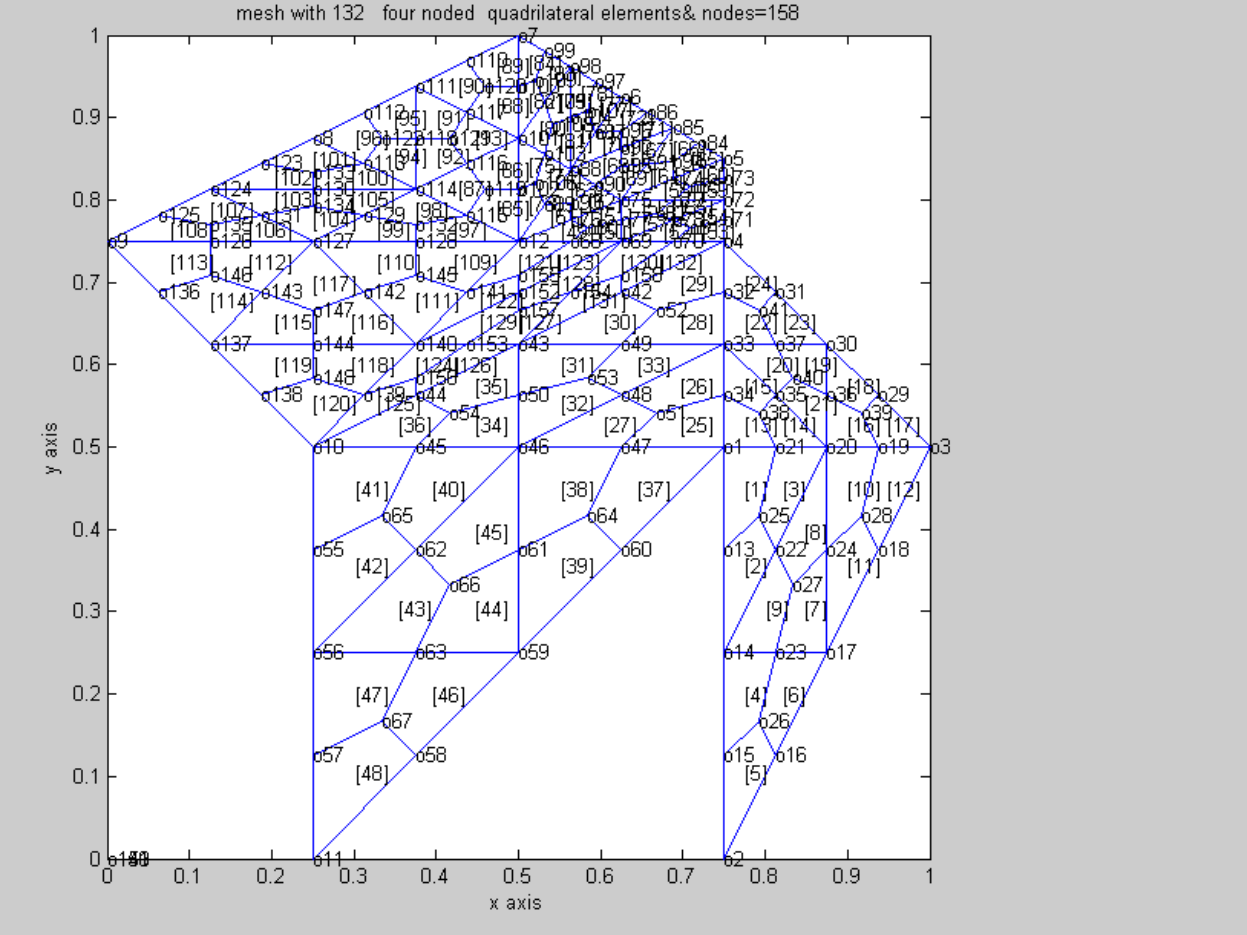


Figure 23:NONCONVEX POLYGON MESH-3

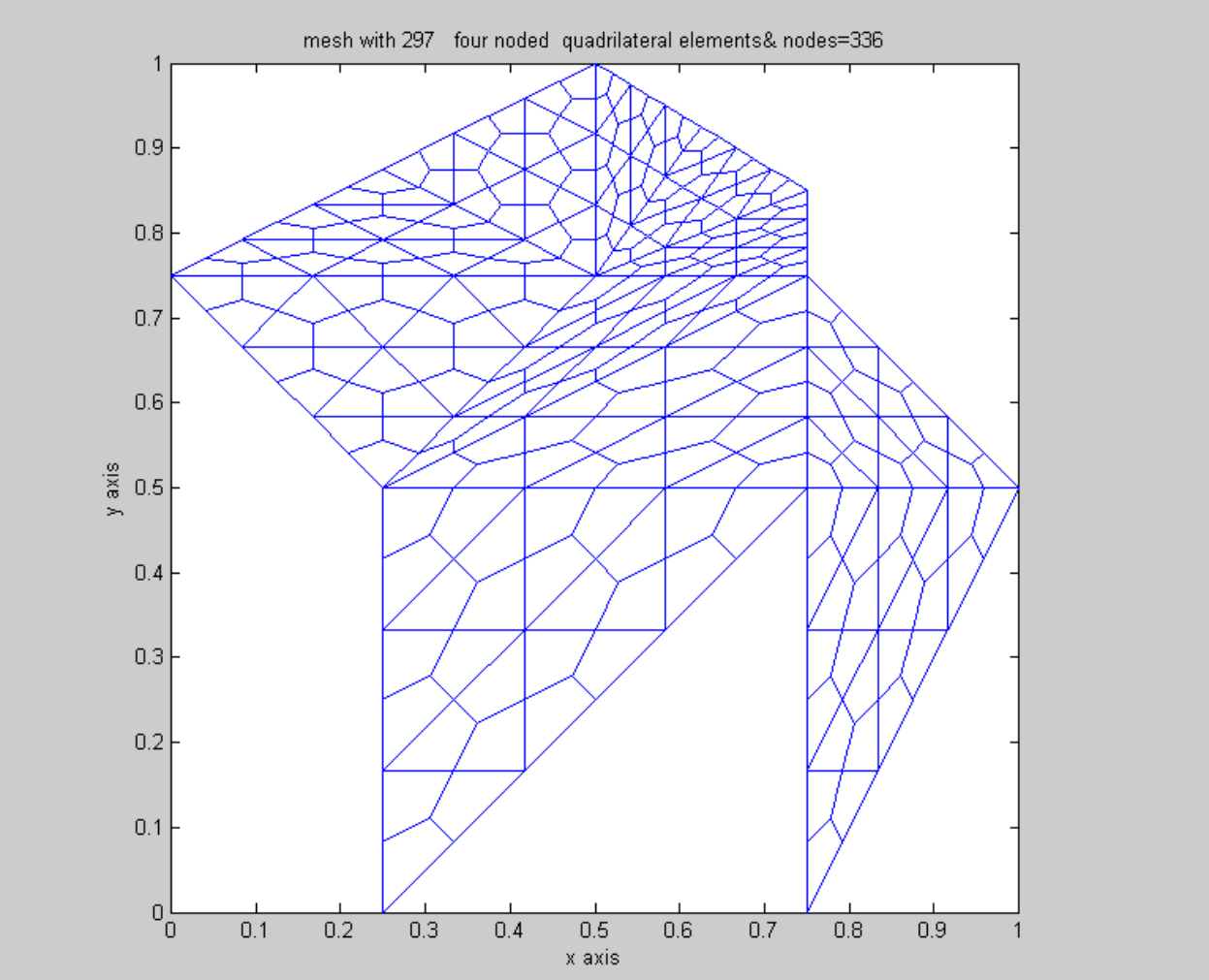


Figure 24 :NONCONVEX POLYGON MESH-4

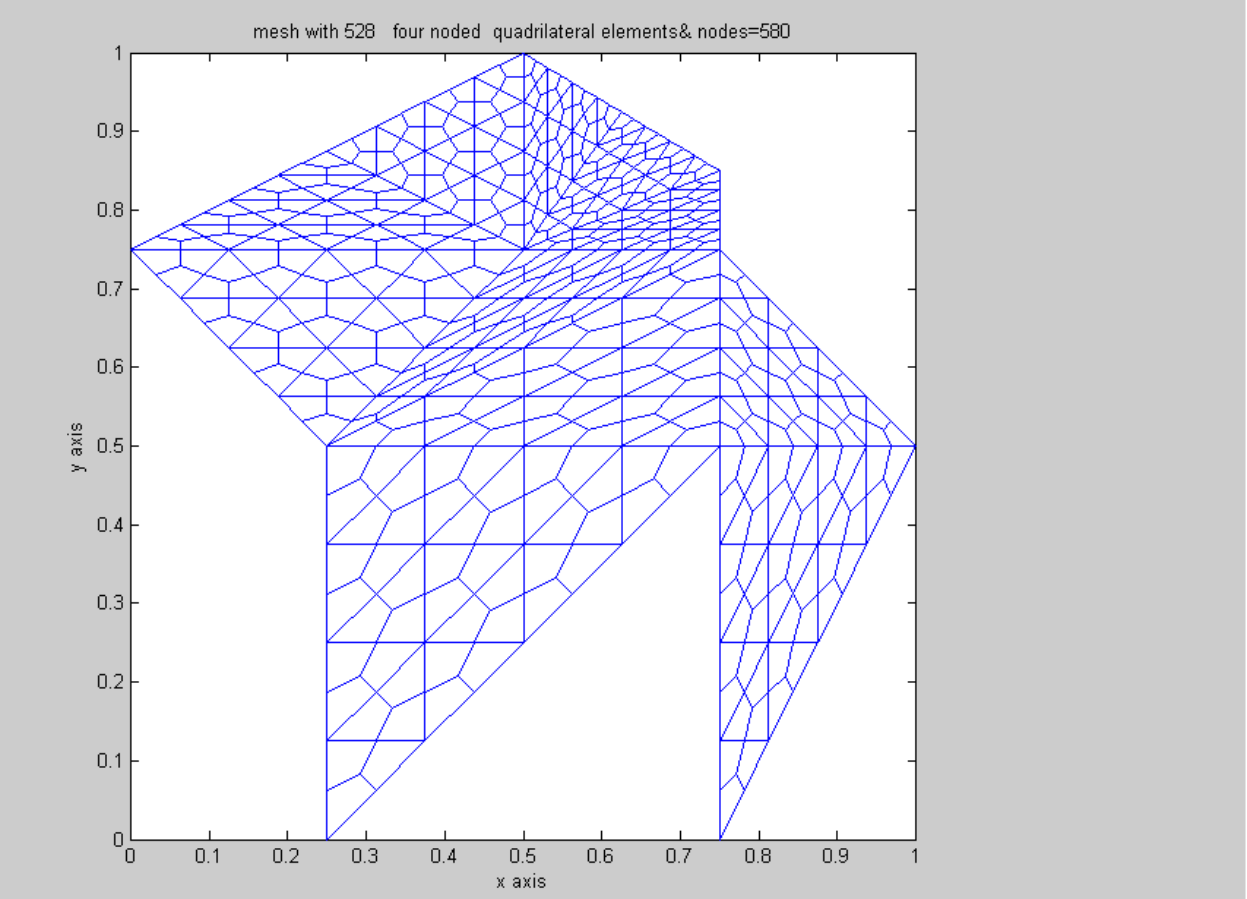


Figure 25:NONCONVEX POLYGON MESH-5

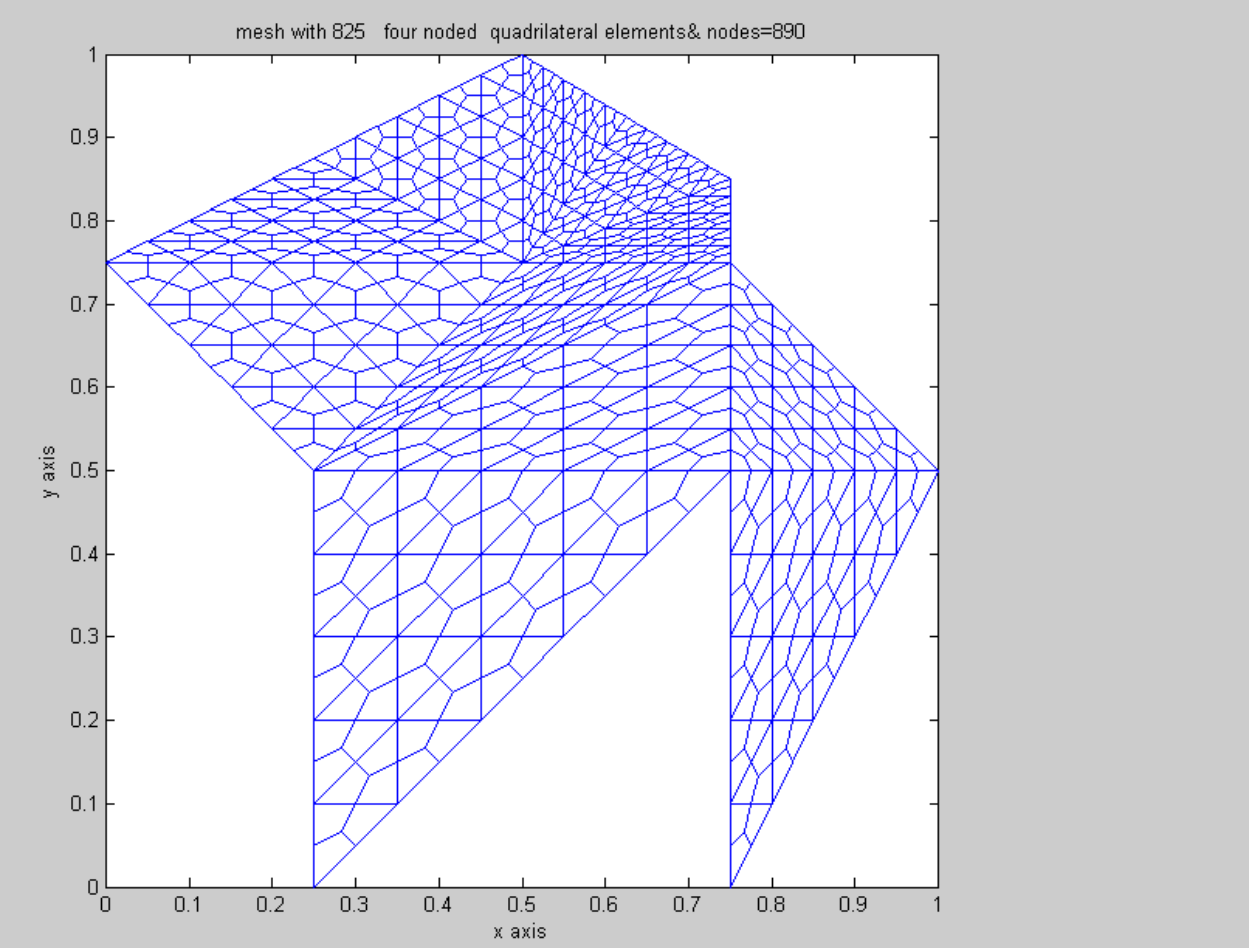


Figure 26: A Mesh on Uniform Refinement For a Convex Polygon-1

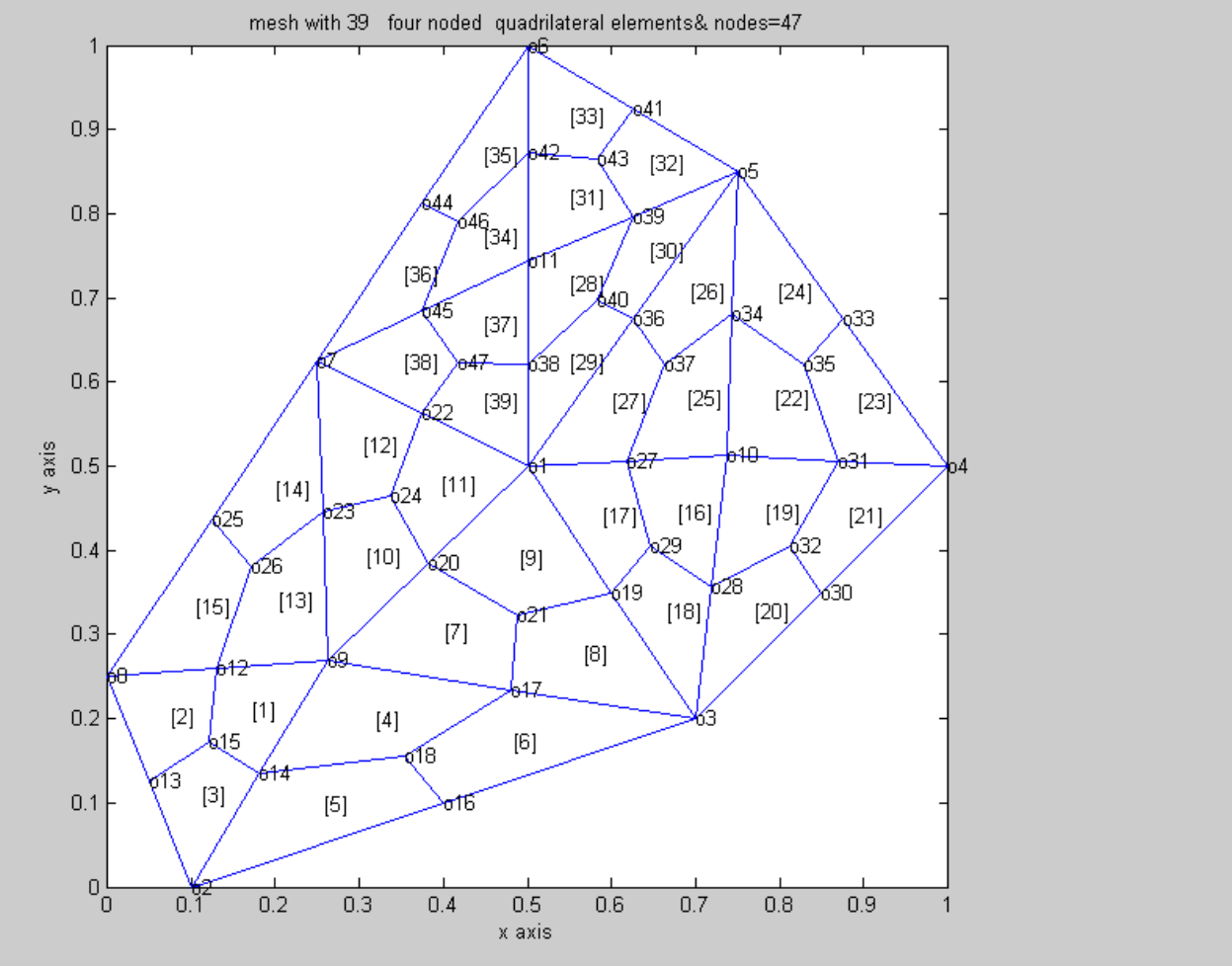


Figure 27: A Mesh on Uniform Refinement For a Convex Polygon-2

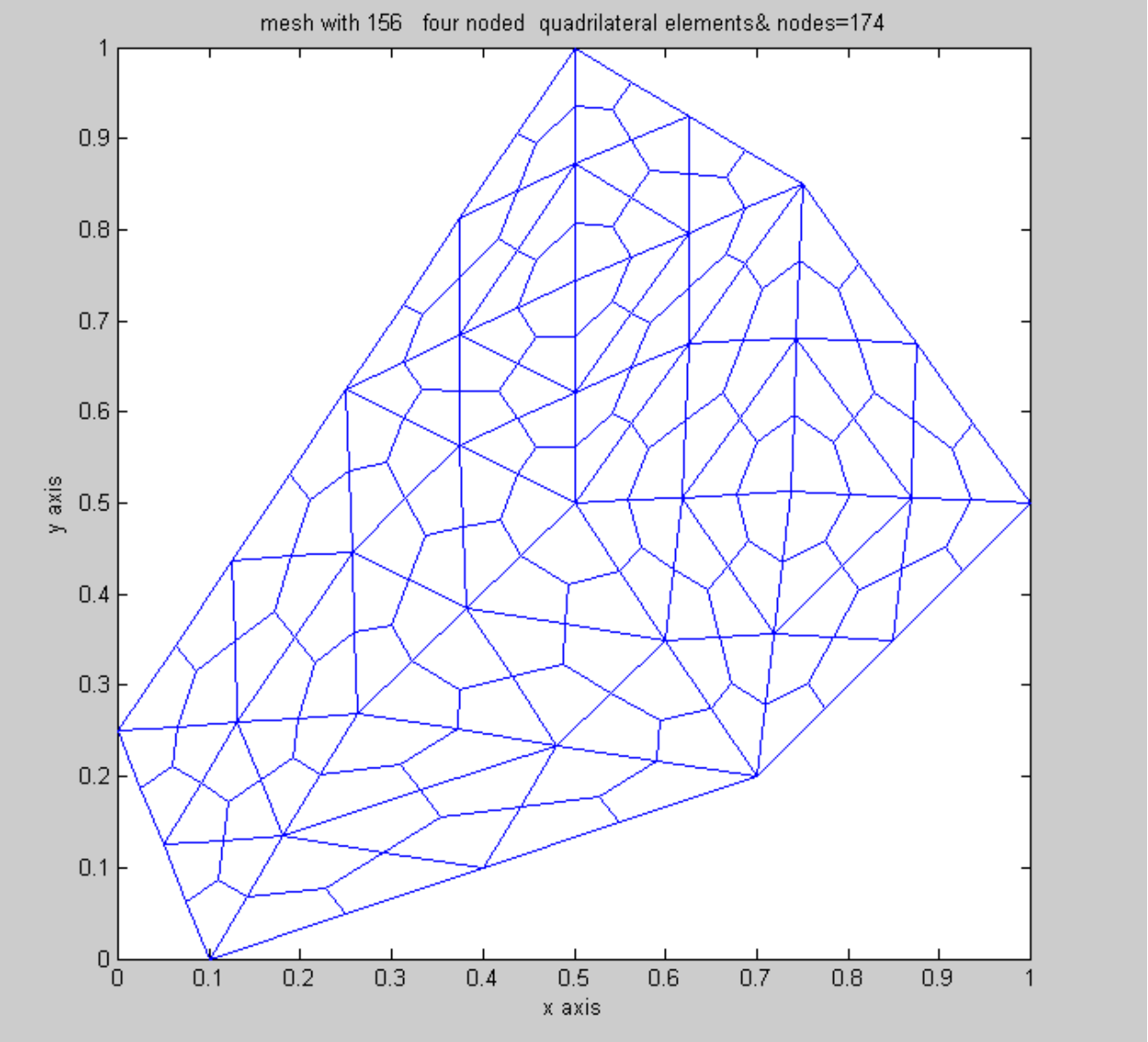


Figure 28: A Mesh on Uniform Refinement For a Convex Polygon-3

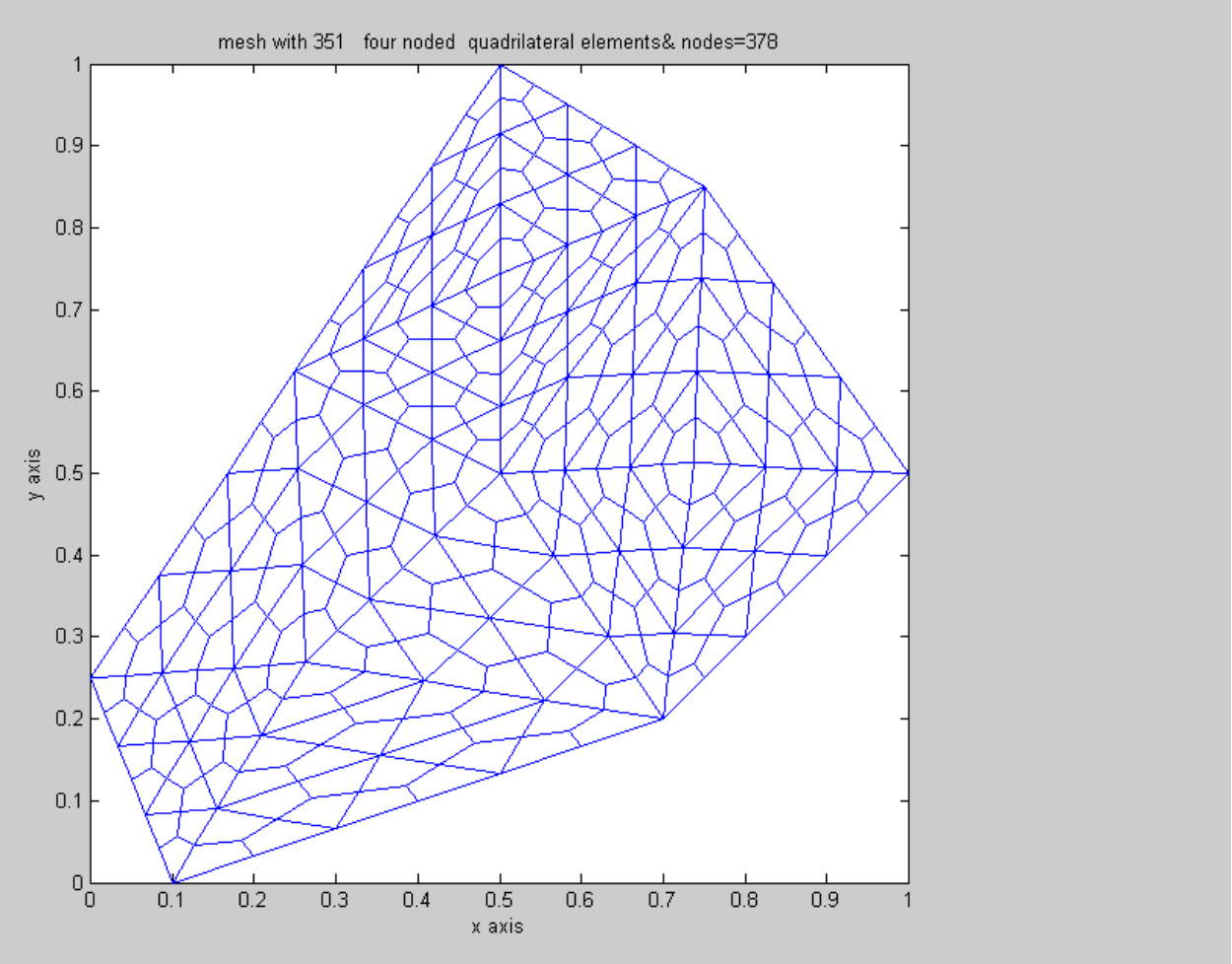


Figure 29: A Mesh on Uniform Refinement For a Convex Polygon-4

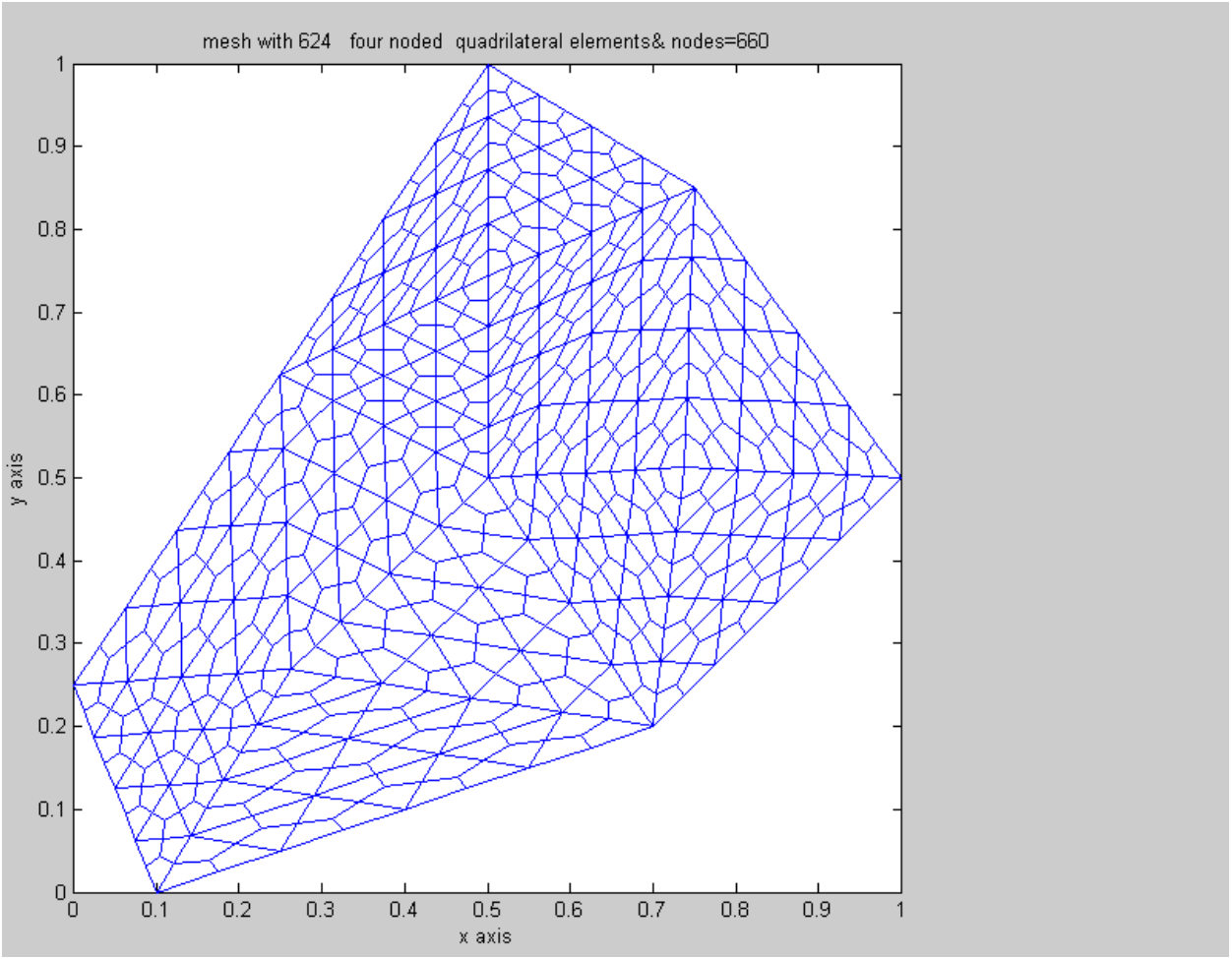
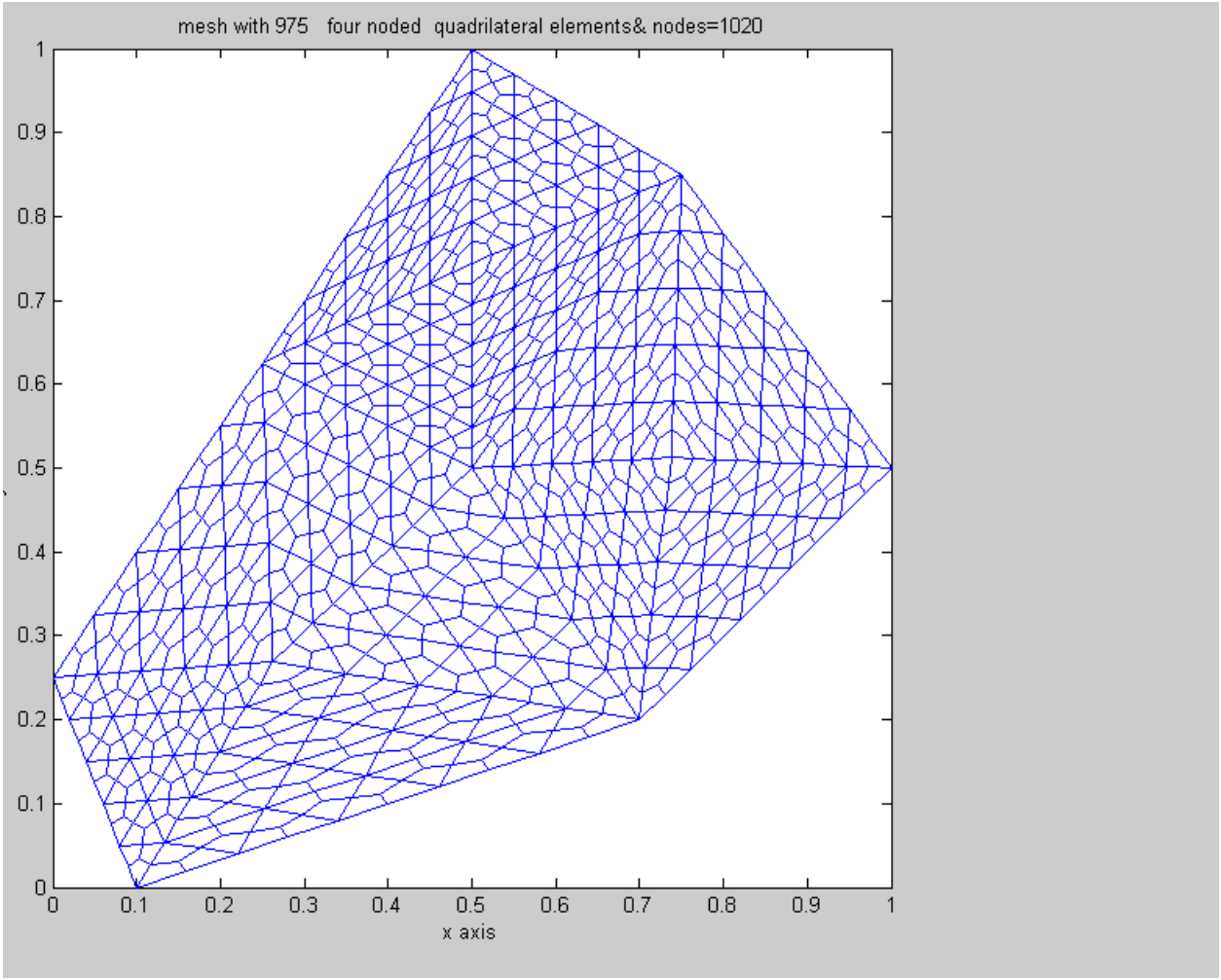


Figure 30: A Mesh on Uniform Refinement For a Convex Polygon-5



MATLAB_PROGRAMS_LINEAR_POLYGONAL_DOMAIN

```
%=========================================================
%PROGRAM_1
%=========================================================
function[]=quadrilateral_mesh4crackedconvexpolygon_q4(trvs,nmax,numtri,ndiv,mesh,xlength,ylength,crack)
%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
%trvs=vertices of the triangles
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 12;1 12 10;1 10 11;1 11 2],12,1,2,10,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 12;1 12 10;1 10 11;1 11 2],12,1,2,10,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 1;12 1 4],12,1,2,10,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 1;12 1 4],12,1,2,10,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 1;12 1 4],12,1,2,10,1,1,2)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2],12,1,2,10,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2],12,1,2,10,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 4],12,1,2,10,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 4],12,1,2,10,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 6 7;1 7 8;1 8 9;1 9 2;1 2 3;1 3 4;1 4 5;1 5 6],9,1,2,11,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 6 7;1 7 8;1 8 9;1 9 2;1 2 3;1 3 4;1 4 5;1 5 6],9,1,2,11,1,1,2)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2],12,1,2,10,1,1,[1;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 4],12,1,2,10,1,1,0)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2;12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 4],12,1,2,10,1,1,[1;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2],12,25,10,10,1,1,[1])%cracked convex polygon
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 9;1 9 2],9,1,2,11,1,1,2)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 4 5;1 5 6;1 6 7;1 7 8;1 8 9;1 9 2;1 2 3;1 3 4;],9,1,2,11,1,1,2)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2],8,1,2,13,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2],8,1,2,14,1,1,1)
```

%quadrilateral_mesh4crackedconvexpolygon_q4([14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,1)
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,[1,1])
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,[1;1])
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,[1;1])
%clf&then
%for ndiv=2:2:10&then
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,(ndiv/2)^2,ndiv,14,1,1,[1;1])
%end
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;111 2;12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 104],12,1,2,10,1,1,[1;0])%nonconvexpolygon
% 1 2 3 4 5 6 7 8
%quadrilateral_mesh4crackedconvexpolygon_q4([7 1 6;7 2 1;7 3 2;7 8 3;9 3 8;9 4 3;9 5 4;9 10 5],10 1 2,15,1,1,[0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3],8,1,2,15,1,1,[0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6],11,1,2,16,1,1,[0;0;0])%three convex polygons
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,[1;1])%heat transfer
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6],11,1,2,16,1,1,[0;0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9],14,1,2,17,1,1,[0;0;0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13],17,1,2,18,1,1,[0;0;0;0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4],17,1,2,19,1,1,[0;0;0;0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16],20,1,2,20,1,1,[0;0;0;0;0;0])
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 5;1 5 6;1 6 7;1 7 8;1 8 2;14 8 7;14 7 9;14 9 10;14 10 11;14 11 12;14 12 13;14 13 8],14,1,2,14,1,1,[1;1])%heat transfer
%
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19]
%trvs=[5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,26,1,2,24,1,1,zeros(8,1))
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 19 22;26 22 25;26 25 24;26 24 19]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,26,1,2,25,1,1,zeros(8,1))
%trvs=[5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21;29 25 24;29 24 27;29 27 28;29 28 25]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,29,1,2,26,1,1,zeros(9,1))
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 19 22;26 22 25;26 25 24;26 24 19;29 24 25;29 25 28;29 28 27;29 27 24]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,29,1,2,27,1,1,zeros(9,1))
%trvs=[5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21;29 25 24;29 24 27;29 27 28;29 28 25;32 28 27;32 27 30;32 30 31;32 31 28]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,32,1,2,28,1,1,zeros(10,1))
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 19 22;26 22 25;26 25 24;26 24 19;29 24 25;29 25 28;29 28 27;29 27 24;32 27 28;32 28 31;32 31 30;32 30 27]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,32,1,2,29,1,1,zeros(10,1))
%trvs=[5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21;29 25 24;29 24 27;29 27 28;29 28 25;32 28 27;32 27 30;32 30 31;32 31 28;35 30 27;35 27 34;35 34 33;35 33 30;36 33 34;36 34 4;36 4 2;36 2 33]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,36,1,2,32,1,1,zeros(12,1))
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21;29 25 24;29 24 27;29 27 28;29 28 25;32 28 27;32 27 30;32 30 31;32 31 28;35 30 27;35 27 34;35 34 33;35 33 30;36 33 34;36 34 13;36 13 10;36 10 33]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,36,1,2,33,1,1,zeros(12,1))
%1====domain=1(closed)
%1=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%2=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%3=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%4=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%5=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%6=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%7=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%8=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%9=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%10=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%11=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%12=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
%===
%===
%commands for 12-polygons forming a square duct, trifirst=4;trilast=47;%down
%trvs=[14 13 12;14 12 9;14 9 10;14 10 13;11 10 9;11 9 6;11 6 7;11 7 10;8 7 6;8 6 3;8 3 4;8 4 7;5 4 3;5 3 1;5 1 2;5 2 4;17 4 2;17 2 15;17 15 16;17 16 4;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 19 22;26 22 25;26 25 24;26 24 19;29 24 25;29 25 28;29 28 27;29 27 24;32 27 28;32 28 31;32 31 30;32 30 27;35 27 30;35 30 34;35 34 33;35 33 27;36 33 34 ;36 34 13;36 13 10;36 10 33]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,36,1,2,33,1,1,zeros(12,1))
%1====domain=1(closed)

```
%4====trifirst(triangle number in the first convexpolygon)
%47===trilast(triangle number in the last convexpolygon)
%.....

%.....
%commands for 12-polygons forming a square duct,trifirst=2;trilast=47;%up
%trvs=[5 1 2;5 2 4;5 4 3;5 3 1;8 3 4;8 4 7;8 7 6;8 6 3;11 6 7;11 7 10;11 10 9;11 9 6;14 9 10;14 10 13;14 13 12;14 12 9;17 13 10;17 10 15;17 15 16;17 16 13;20 16 15;20 15 18;20 18 19;20 19 16;23 19 18;23 18 21;23 21 22;23 22 19;26 21 18;26 18 24;26 24 25;26 25 21;29 25 24;29 24 27;29 27 28;29 28 25;32 28 27;32 27 30;32 30 31;32 31 28;35 30 27;35 27 34;35 34 33;35 33 30;36 33 34;36 34 4;36 4 2;36 2 33]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,36,1,2,32,1,1,zeros(12,1))
%1====domain=1(closed)
%2
%47
%.....
%three in one convexpolygon
%trvs=[9 8 2;9 2 3;9 3 1;9 1 7;9 7 8;10 1 3;10 3 4;10 4 5;10 5 1;11 1 5;11 5 6;11 6 7;11 7 1]
%quadrilateral_mesh4crackedconvexpolygon_q4(trvs,11,4,4,35,1,1,[0;0;0])%cracked convex polygon
%1====domain=1(closed)
%4====trifirst(triangle number in the first convexpolygon)
%13===trilast(triangle number in the last convexpolygon)
%================================================
%for a nonconvexpolygon-->cracked polygon joining the first triangle of convex polygon
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2;12 10 4;12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10],12,1,2,10,1,1,[1;0])%nonconvexpolygon
%0--->domain=0
%=============================================================================================================
%for a nonconvexpolygon-->cracked polygon joining the last triangle of convex polygon
%quadrilateral_mesh4crackedconvexpolygon_q4([1 2 3;1 3 4;1 4 10;1 10 11;1 11 2;12 4 5;12 5 6;12 6 7;12 7 8;12 8 9;12 9 10;12 10 4],12,1,2,10,1,1,[1;0])%nonconvexpolygon
%1--->domain=1
%10000====trifirst(triangle number in the first convexpolygon---->a large positive integer)
%20000===trilast(triangle number in the last convexpolygon---->a different large positive integer)

%=========================================================================
global EDGEN1N2 EDGEN2N3 EDGEN1N3  ncrack npart  NMAX NITRI part nitri NE N1 N2 N3 rrr TRINUM trifirst trilast qqfirst
%global firstpoly lastpoly nitri1 EDGEN1N2FIRST EDGEN2N3FIRST EDGEN1N3FIRST
%crack=number of cracks in the polygon
%crack=0,for a convex polygon
%crack=1,for a cracked convex polygon
%let crack be a column vector of zeros and ones
m1=trvs(:,1);m2=trvs(:,2);m3=trvs(:,3);
[npart,part,numsame,maxpart]=checksamenumbers(m1)
ncrack(1:npart,1)=crack(1:npart,1)
%if npart==1
 %   n1=m1;n2=m2;n3=m3;
 %end
%if npart==2
 %   ii=part(1,1)+1;
  %  jj=part(1,1)+part(2,1)
   % n1=trvs(ii:jj,1)
   % n2=trvs(ii:jj,2)
    %n3=trvs(ii:jj,3)
    %end
TRINUM=0;%acounter for number of coarse triangles
trifirst=-1;trilast=-2;%arbtrary data useful in the  mesh generation of cracked polygon
    [coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(m1,m2,m3,nmax,numtri,ndiv,mesh)
ncrack
ndiv
fgnum=ndiv/2;
[nel,nnel]=size(nodes);
%if ncrack==1
%nel=nel-3*(ndiv/2)^2
%nnode=nnode-(ndiv-1)*(ndiv)/2-(ndiv/2)^2
%nitri=nitri-1
%end
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:nnode,1)=gcoord(1:nnode,1);
ycoord(1:nnode,1)=gcoord(1:nnode,2);
%extract coordinates for each element
%clf
figure(fgnum)
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 9
    axis([0 xlength 0 ylength])
case 10
 axis([0 xlength 0 ylength])
case 11
 axis([0 xlength 0 ylength])
case 12
 axis([0 xlength 0 ylength])
case 13
 axis([0 xlength 0 ylength])
 case 14
 axis([0 2*xlength 0 ylength])
case 15
  axis([0 2*xlength 0 2*ylength])
case 16
  axis([0 2*xlength 0 2*ylength])

case 17
  axis([0 2*xlength 0 2*ylength])
 case 18
  axis([0 2*xlength 0 2*ylength])
 case 19
  axis([0 2*xlength 0 2*ylength])
 case 20
  axis([0 2*xlength 0 2*ylength])
 case 21
  axis([0 2*xlength 0 2*ylength])
 case 22
  axis([0 2*xlength 0 2*ylength])

 case 23
  axis([0 2*xlength 0 2*ylength])
 case 24
  axis([0 2*xlength 0 2*ylength])
  case 25
  axis([0 2*xlength 0 2*ylength])
  case 26
  axis([0 2*xlength 0 2*ylength])
  case 27
  axis([0 2*xlength 0 2*ylength])
  case 28
  axis([0 2*xlength 0 2*ylength])
  case 29
  axis([0 2*xlength 0 2*ylength])
   case 30
  axis([0 2*xlength 0 2*ylength])
   case 31
  axis([0 2*xlength 0 2*ylength])
  case 32
  axis([0 2*xlength 0 2*ylength])
  case 33
  axis([0 2*xlength 0 2*ylength])
  case 34
  axis([0 2*xlength 0 ylength])
  case 35
    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
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
disp([xlength,ylength,nnode,nel,nnel])
disp(nodes(1:nel,:))
disp([xcoord(1:nnode,1) ycoord(1:nnode,1)])
%SIDE(:,itri,1)---->side joining vertices n1&n2 of triangle itri
%SIDE(:,itri,2)---->side joining vertices n2&n3 of triangle itri
%SIDE(:,itri,3)---->side joining vertices n1&n3 of triangle itri
for itri=1:nitri
itri
disp('rrr(:,:,itri)=')
disp(rrr(:,:,itri))
end
n=ndiv;
for itri=1:nitri
   %increasing rows=side-3
   side(1:n+1,itri,3)=rrr(1:n+1,1,itri);
    %increasing columns=side-1
   side(1:n+1,itri,1)=rrr(1,1:n+1,itri);
   %side-2(diagonal increasing rows)
   for i=1:n+1
      side(i,itri,2)=rrr(i,(n+1)-i+1,itri);
   end
   itri
   disp([side(1:n+1,itri,1) side(1:n+1,itri,2) side(1:n+1,itri,3)])
end

disp('[ncrack]=')
disp([ncrack])
disp('[ NMAX NITRI]=')
disp([NMAX NITRI])

%disp([side(1:n+1,trifirst,1) side(1:n+1,trifirst,2) side(1:n+1,trifirst,3)])
%disp([side(1:n+1,trilast,1) side(1:n+1,trilast,2) side(1:n+1,trilast,3)])

%=========================================================
%PROGRAM_2
%=========================================================
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,n)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4)
global EDGEN1N2 EDGEN2N3 EDGEN1N3  ncrack npart  NMAX NITRI part nitri NE N1 N2 N3 rrr TRINUM trifirst trilast qqfirst
ne=0
 [n1 n2 n3]
rrr(1:n+1,1:n+1,1:nitri)=zeros(n+1,n+1,nitri)
for itri=1:nitri
   TRINUM=TRINUM+1
```

```
   rr(1:n+1,1:n+1)=zeros(n+1,n+1)
   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);
   edgen1n2(1:n+1,itri)=elm(1:n+1,1);
end%if itri>1
if itri==1
   lmax=nmax+3*(n-1);
end%if itri==1

if (npart>1)&(NE~=0)&((trifirst<0)&(trilast<0))
if itri==1%old statement
lmax=nmax+2*(n-1);
end
end

if (itri>1)&(itri<nitri)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
if (TRINUM==trilast)
   lmax=lmax-(n-1);
end

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
edgen1n3(1:n+1,nitri)=edgen1n2(1:n+1,1);
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]
%search for common edge from previous polygon for the last triangle
% of the present convex polygon,if a common edge exists then replace it by current edge
if (npart>1)&(NE~=0)
if itri==nitri%old statement
   for jtri=1:NITRI
      qqo=n2(nitri,1);
      qqe=n3(nitri,1);
      if((qqo==N3(jtri,1))&(qqe==N2(jtri,1)))
         for i=1:n+1
            qqi(i,1)=EDGEN2N3((n+1)-i+1,jtri)
            row_nodes(i,(n+1)-i+1)=qqi(i,1);
         end
         edgen2n3(1:n+1,nitri)=qqi(1:n+1,1);%old statement
      end
   end%old statement
end
end
%first element of (npart=2,3,4 etc.... ) second convex polygon joining
%last element  of first  convex polygon
if (npart>1)&(NE~=0)
if itri==1%old statement

   for jtri=1:NITRI
      qqo=n2(1,1);
      qqe=n3(1,1);
      if((qqo==N3(jtri,1))&(qqe==N2(jtri,1)))
         for i=1:n+1
            qqi(i,1)=EDGEN2N3((n+1)-i+1,jtri)
            row_nodes(i,(n+1)-i+1)=qqi(i,1);
         end
         edgen2n3(1:n+1,1)=qqi(1:n+1,1);%old statement
      end
   end%old statement
end
end

if (TRINUM==trifirst)
  for i=1:(n+1)
   qqfirst(i,1)=edgen2n3((n+1)-i+1,trifirst);
  end
```

```
end
if (TRINUM==trilast)
  for i=1:(n+1)
   row_nodes(i,(n+1)-i+1)=qqfirst(i,1);
  end
  edgen2n3(1:n+1,trilast)=qqfirst(1:n+1,1);
end



rr=row_nodes;
rr
%[rri rrj]=size(rr)
%for i=1:rri
% for j=1:rrj
 %    rrr(rri,rrj,itri)=rr(rri,rrj);
 %end
 %end
rrr(1:n+1,1:n+1,itri)=rr(1:n+1,1:n+1);
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
nmax=max(max(eln));
%nel=mm;
%
%ne


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
%
EDGEN1N2=edgen1n2;
EDGEN2N3=edgen2n3;
EDGEN1N3=edgen1n3;

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

EDGEN1N2
EDGEN2N3
EDGEN1N3
for itri=1:nitri
itri
side(:,itri,1)=EDGEN1N2(:,itri);
side(:,itri,2)=EDGEN2N3(:,itri);
side(:,itri,3)=EDGEN1N3(:,itri);
[side(:,itri,1) side(:,itri,2) side(:,itri,3)]
end

disp('this is trial')
%NE=ne
%NITRI=nitri
NE=ne
disp(NE)
%N1=n1;N2=n2;N3=n3;
%[N1 N2 N3]
spqd
nodetel
%==========================

 TRINUM

 eln


%============================================================
%PROGRAM_3
%============================================================
function[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(m1,m2,m3,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
%choice=0
global EDGEN1N2 EDGEN2N3 EDGEN1N3  ncrack npart  NMAX NITRI part nitri NE N1 N2 N3 rrr TRINUM trifirst trilast qqfirst
syms U V W xi yi
%ncrack(1:npart,1)=crack(1:npart,1)
domain=input('multipolygonal domain,if closed :domain=1;if open: domain=0')
if domain==1
trifirst=input('domain=1,enter the value for: trifirst ')
trilast=input('domain=1,enter the value for: trilast ')
end
n+1
switch mesh
case 1%for MOIN POLYGON
x=sym([1/2;1/2;1;  1;1/2;0;  0;0])%for MOIN EXAMPLE
y=sym([1/2;  0;0;1/2;  1;1;1/2;0])%for MOIN EXAMPLE


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

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

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

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

case 10
   %   1   2   3   4   5   6    7   8     9   10  11  12
   x=[.75;.75;1.0;.75;.75;1.25/2;0.5;0.2500;0.0;.25;.25;0.5]
   y=[0.5;0.0;0.5;.75;.85;1.85/2;1.0;1.75/2;.75;0.5;0.0;.75]
case 11
   %   1  2 3  4 5  6 7 8  9
   x=[.5;.5;0;0.;0.;.5;1.;1.;1]
   y=[.5;1.;1;.5;0.;0.;0.;.5;1]
case 12
   %   1  2  3  4  5  6  7  8  9
   x=[.5;0.;0.;.5;1.;1.;1.;.5;0.]
   y=[.5;.5;0.;0.;0.;.5;1.;1.;1.]
case 13
  %   1  2  3  4  5  6  7 8  9
   x=[.5;.5;0;0.;0.;.5;1.;1.]
   y=[.5;1.;1;.5;0.;0.;0.;.5]
case 14%heat transfer
  %    1  2  3  4  5  6 7 8  9 10 11  12 13  14
   x=[.5;.5;0;0.;0.;.5;1.;1.;1.5;2;2.0;2.;1.5;1.5]
   y=[.5;1.;1;.5;0.;0.;0.;.5;0.0;0;0.5;1.;1.0;0.5]
case 15% rectangle=2
   % 1   2   3    4   5   6   7   8   9   10
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75]
case 16% rectangle=3
   % 1   2   3    4   5   6   7   8   9   10  11
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25]

 case 17% rectangle=4
```

```
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75]
 case 18% rectangle=5,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16 17
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75]
    case 19% rectangle=5,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16 17
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25]
    case 20% rectangle=6,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75]
   case 21% rectangle=6,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17  18  19 20
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25]
 case 22% rectangle=7,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22  23
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75]
 case 23% rectangle=7,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25]
  case 24% rectangle=8,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22 23 24 25 26
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75;1.5;2;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75;1.0;1;1.25]
  case 25% rectangle=8,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23 24 25 26
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75;1.5;2;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25;1.0;1;0.75]
    case 26% rectangle=9,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22 23 24 25 26 27 28 29
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75;1.5;2;1.75;1.5;2.;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75;1.0;1;1.25;0.5;.5;0.75]
   case 27% rectangle=9,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23 24 25 26 27 28 29
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75;1.5;2;1.75;1.5;2.0;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25;1.0;1;0.75;1.5;1.5;1.25]
case 28% rectangle=10,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22 23 24 25 26 27 28 29 30 31 32
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75;1.5;2;1.75;1.5;2.;1.75;1.5;2;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75;1.0;1;1.25;0.5;.5;0.75;0.0;0;0.25]
 case 29% rectangle=10,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23 24 25 26 27 28 29  30 31 32
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75;1.5;2;1.75;1.5;2.0;1.75;1.5;2.0;1.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25;1.0;1;0.75;1.5;1.5;1.25;2.0;2.0;1.75]
 case 30% rectangle=11,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75;1.5;2;1.75;1.5;2.;1.75;1.5;2;1.75;1;1.0;1.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75;1.0;1;1.25;0.5;.5;0.75;0.0;0;0.25;0;0.5;0.25]
  case 31% rectangle=11,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23 24 25 26 27 28 29  30 31 32  33 34 35
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75;1.5;2;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.0;1;1.25]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25;1.0;1;0.75;1.5;1.5;1.25;2.0;2.0;1.75;1.5;2;1.75]
    case 32% rectangle=12,up
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15  16 17 18 19  20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  36
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2.0;2;1.75;1.5;2;1.75;1.5;2.;1.75;1.5;2;1.75;1;1.0;1.25;.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.5;2;1.75;1.0;1;1.25;0.5;.5;0.75;0.0;0;0.25;0;0.5;0.25;.25]
    case 33% rectangle=12,down
   % 1  2  3   4  5  6  7  8  9  10 11 12 13 14 15 16  17 18  19 20 21 22 23 24 25 26 27 28 29  30 31 32  33 34 35 36
   x=[0.0;0.5;0.0;0.5;0.25;0.;.5;.25;0.0;0.5;0.25;0;0.5;0.25;1.0;1.0;0.75;1.5;1.5;1.25;2;2.;1.75;1.5;2;1.75;1.5;2.0;1.75;1.5;2.0;1.75;1.0;1;1.25;0.75]
   y=[0.0;0.0;0.5;0.5;0.25;1.;1.;.75;1.5;1.5;1.25;2;2.0;1.75;0.0;0.5;0.25;0.0;0.5;0.25;0;.5;0.25;1.0;1;0.75;1.5;1.5;1.25;2.0;2.0;1.75;1.5;2;1.75;1.75]
case 34%heat transfer
   %   1 2 3  4  5  6 7 8  9 10 11  12 13  14   15
    x=[.5;.5;0;0.;0.;.5;1.;1.;1;1.5;2;2.0;2.;1.5;1.5]
    y=[.5;1.;1;.5;0.;0.;0.;.5;1;0.0;0;0.5;1.;1.0;0.5]
case 35%mesh for convex polygon(=union of three convex polygons)%made out of a convex polygon of six  sides
  %   1   2  3 4  5 6 7    8     9    10    11
x=([0.5;0.1;0.7;1.;.75;.5;0.25;0.0;1.050/4;2.95/4;0.50000])
y=([0.5;0.0;0.2;.5;.85;1.;.625;.25;1.075/4;2.05/4;2.975/4])
end
 npart
if (nmax>3)
k0=0;k1=0;nelx0=0;nelx1=0;nitrix0=0;nitrix1=0;NE=0;  ne=(n/2)^2
for sss=1:npart
   pp=part(sss,1)
```

```
   nitri=pp
   k0=k1+1
   k1=k1+pp
   n1(1:pp,1)=m1(k0:k1,1)
   n2(1:pp,1)=m2(k0:k1,1)
   n3(1:pp,1)=m3(k0:k1,1)
   TRINUM
   [elnx,spqdx,rrrx,nodesx,nodetelx]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,n);
   [nelx,nnel]=size(nodesx);
   nelx=nelx-ncrack(sss,1)*(3*(n/2)^2)
   nelx0=nelx1+1;nelx1=nelx1+nelx
   nitrix=pp-ncrack(sss,1)
   NITRI=nitrix
   nitrixx(sss,1)=nitrix
   nn0(sss,1)=nitrix1+1;nne(sss,1)=(nitrix1+nitrix)
   nitrix0=ne*nitrix1+1;nitrix1=(nitrix1+nitrix);
   spqdx=nodesx(1:nelx,1:nnel)
   nnodex=max(max(spqdx))
   N1(1:NITRI,1)=n1(1:nitrix,1);
   N2(1:NITRI,1)=n2(1:nitrix,1);
   N3(1:NITRI,1)=n3(1:nitrix,1);
[N1 N2 N3]
 eln(nitrix0:ne*nitrix1,1:7)=elnx(1:ne*nitrix,1:7)
   nodes(nelx0:nelx1,1:nnel)=nodesx(1:nelx,1:nnel)
   spqd(nelx0:nelx1,1:nnel)=nodesx(1:nelx,1:nnel)
   nodetel(nelx0:nelx1,1:3)=nodetelx(1:nelx,1:3)
   RRR(1:n+1,1:n+1,nn0(sss,1):nne(sss,1))=rrrx(1:n+1,1:n+1,1:nitrixx(sss,1));
   nmax=nnodex
end

nnode=max(max(spqd))
for iiii=1:length(m1)
   n1(iiii,1)=0;
   n2(iiii,1)=0;
   n3(iiii,1)=0;
end
nitri=0;
for ssss=1:npart
nnnx=nitrixx(ssss,1)
nitri=nitri+nnnx
nnn0=nn0(ssss,1)
nnne=nne(ssss,1)
if (ssss==1)
   ppp0=0
   pppe=nnnx
end
if (ssss>1)
   ppp0=0;
   for ii=1:ssss-1
      ppp0=ppp0+part(ii,1)
   end
    pppe=ppp0+nnnx
end

n1(nnn0:nnne,1)=m1(ppp0+1:pppe,1);
n2(nnn0:nnne,1)=m2(ppp0+1:pppe,1);
n3(nnn0:nnne,1)=m3(ppp0+1:pppe,1);
end
rrr=RRR;

end%nmax>3
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)])
```

```
xi(1:nnode,1)=zeros(nnode,1);
yi(1:nnode,1)=zeros(nnode,1);

%nitri=nmax-1;
if (nmax==3)nitri=1
end
eln
[n1 n2 n3]
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(1:n+1,1:n+1,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
itri
[xi yi]
end
%add coordinates of centroid

 ne=(n/2)^2

% stdnode=kk;
eln
nitri
for itri=1:nitri
 for iii=1+(itri-1)*ne:ne*itri
    %kk=kk+1;
    %eln(iii,1:7)
    node1=eln(iii,1)
    node2=eln(iii,2)
    node3=eln(iii,3)
    [node1 xi(node1,1) yi(node1,1);node2 xi(node2,1) yi(node2,1);node3 xi(node3,1) yi(node3,1)]
    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)
nodes=spqd
NMAX=nnode
```

```
%=========================================================
%PROGRAM_4
%=========================================================
function[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n)
syms ui vi wi U V W
kk=0;
for j=1:n+1
   for i=1:(n+1)-(j-1)
      kk=kk+1;
      ui(i,j)=(i-1)/n;
      vi(i,j)=(j-1)/n;
      wi(i,j)=1-ui(i,j)-vi(i,j);
```

```
      U(kk,1)=ui(i,j);
      V(kk,1)=vi(i,j);
      W(kk,1)=wi(i,j);
   end
end
 U'
 V'
 W'
%=====================================================================
%PROGRAM_5
%=====================================================================
function[npart,part,numsame,maxpart]=checksamenumbers(n1)
%npart=number of parts with same numbers
%part=column vector for number of repetitions in each part(how many
%repetitions?)
%numsame=column vector storing the number which is repeated(which number is
%repeated?)
%checksamenumbers([1;1;1;1;1;5;5;5;5;5;5;5;5])
%[npart,part,numsame,maxpart]=checksamenumbers([2;2;2;2;2;2;2;2;2;1;1;1;1;1;5;5;5;5;5;5;5;5;9;9;9;9;9;9;9])
L=length(n1);
%how many different numbers
k=1;
same=1;
for i=1:L-1
   samenum0=n1(i,1);
   samenum1=n1(i+1,1);
   if (samenum0==samenum1),same=same+1;end

   if (samenum0~=samenum1),break,end

   samenum=samenum0;
end
disp('n1=')
samenum*ones(same,1);
n1(1:same,1);
part(k,1)=same;
sum=part(k,1);
numsame(k,1)=samenum;

while sum<=L
   if (sum==L),break,end
   same=1
for i=sum+1:L-1
   samenum0=n1(i,1);
   samenum1=n1(i+1,1);
   if (samenum0==samenum1),same=same+1;end

   if (samenum0~=samenum1),break,end

   samenum=samenum0;
end
k=k+1;
samenum*ones(same,1);
n1(sum+1:sum+same,1);
part(k,1)=same;
sum=sum+same;
numsame(k,1)=samenum;
end
part;
npart=length(part);
disp(['number of parts= ',num2str(npart)])
%
%disp(['part=',num2str(1),' ,number of same numbers=',num2str(part(1,1)),' &same num=',num2str(numsame(1,1))])
%disp([n1(1:part(1,1),1)])
%[n1(part(1,1)+1:part(1,1)+part(2,1),1)]
%ss=0;
%for kk=1:k-1
 %   disp(['part=',num2str((kk+1)),' ,number of same numbers=',num2str(part(kk+1,1)),' &same num=',num2str(numsame(kk+1,1))])
  %  ss=ss+part(kk,1);
   % disp([n1(ss+1:ss+part(kk+1,1),1)])
   %end

ss=0;
for kk=1:npart
   disp(['part=',num2str((kk)),' ,number of same numbers=',num2str(part(kk,1)),' &same num=',num2str(numsame(kk,1))])
   disp([n1(ss+1:ss+part(kk,1),1)])
   N1(1:part(kk,1),kk)=numsame(kk,1)*ones(part(kk,1),1);
  ss=ss+part(kk,1);
end
N1
maxpart=max(part)
```

```
L
%sum(part)
sumpart=0
for ii=1:npart
   sumpart=sumpart+part(ii,1);
end
sumpart
```