www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue -8 August, 2014 Page No. 7437-7507

A new automatic finite element mesh generation scheme of all quadrilaterals over an analytical curved surface by using parabolic arcs

H.T. Rathod[a*] , *Bharath Rathod*[b] , *K.V.Vijayakumar*[c] , *K. Suganth 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-560034 , Karnataka state, India.
Email: rathodbharath@gmail.com

[c] Department of Mathematics,B.M.S.Institute of Technology,Avalahalli,
Bangalore-560064, Karnataka state, India.
Email: kallurvijayakumar@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 simply connected curved domain of a planar region which has curved boundary described by one or more analytical equations. We first decompose this curved domain into simple sub regions in the shape of curved triangles. These simple regions are then triangulated to generate a fine mesh of linear triangles in the interior and curved triangles near to the boundary of curved domain. We then propose, an automatic triangular to quadrilateral conversion scheme. Each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of the edges which are either a straight segment or a curved arc and a vertex at the barrycentre(a point located at the average of three vertices) of the element. We have approximated the curved arcs by equivalent **parabolic arcs.**To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given curved 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 on a **circular disk**, on a **cracked circular disk** and on a **lunar model** are presented to illustrate the simplicity and efficiency of the new mesh generation method. We have appended the 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,analytical curved surfaces, curved triangular element, parabolic arcs,uniform refinement.

1. Introduction

The finite element method (FEM) is 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 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].

The domain of real problems often contains curved boundaries. In classical finite element applications curved boundaries are discretised by extremely refined meshes because simplifying the curved domains by polygonal domains may cause global changes in the physical solution of the problem. Curved boundaries are often more accurately modeled by curved finite elements than by straight edged elements, as straight sides are perfectly satisfactory if the domain has a polygonal boundary. The use of curved elements can model the complex geometry by fewer elements and this result in faster convergence to the desired solution. Curved triangular element with one curved side and two straight sides are found very useful in the solution of two dimensional boundary value problems[28]

In recent papers[26-27], authors presented a novel mesh generation scheme of all quadrilateral elements for polygonal domains. In this paper, we present a novel mesh generation scheme of all quadrilateral elements for the analytical surfaces. This scheme converts the elements in background triangular mesh into quadrilaterals through the operation of splitting. We first decompose the analytical curved surface into simple subregions in the shape of curved 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 analytical curved surface into all quadrilaterals, thus propogating uniform refinement.We may note that for an analytical curved surface the domain is discritised into straight edged quadrilaterals in the interior of the domain and curved quadrilaterals near to the boundary of the domain. In section 2.1 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 2.2, we consider the derivation of a curved triangular element with two straight sides and one curved side. The curved side is modeled by a quadratic curve passing through four points of the original curved side. 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 analytical curved surface. In section 5,we present the mesh generation for a quarter circle,a circular disk,a lunar model and also the typical examples on cracked analytical surfaces of a circular disk to illustrate the simplicity and efficiency of the proposed mesh generation method.

2. Division of an Arbitrary Triangle

2.1 Arbitrary Linear 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.

2.2 Cubic Curved Triangular Element

We first consider the ten node triangular element in which all the three sides are curved. The transformation which maps such a general curved triangular element in Cartesian space (x, y) into a right isosceles triangle with sides of 1 unit in local parametric (ξ, η) is shown in **Fig 1c, 1d.**

The necessary transformation for this purpose is given by

$$x = \sum_{i=1}^{10} x_i\ N_i(\xi, \eta), \qquad y = \sum_{i=1}^{10} y_i\ N_i(\xi, \eta), \qquad 1 = \xi + \eta + \zeta \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (2)$$

where (x_i, y_i) are the cartesian coordinates of ith node and

$N_1(\xi,\eta,\zeta) = \frac{1}{2}(3\xi - 1)(3\xi - 2)\xi$,$N_2(\xi,\eta,\zeta) = \frac{1}{2}(3\eta - 1)(3\eta - 2)\eta$

$N_3(\xi,\eta,\zeta) = \frac{1}{2}(3\zeta - 1)(3\zeta - 2)\zeta$,$N_4(\xi,\eta,\zeta) = \frac{9}{2}\xi\eta(3\eta - 1)$

$N_5(\xi,\eta,\zeta) = \frac{9}{2}\xi\eta(3\xi - 1)$,$N_6(\xi,\eta,\zeta) = \frac{9}{2}\zeta\eta(3\zeta - 1)$

$N_7(\xi,\eta,\zeta) = \frac{9}{2}\zeta\eta(3\eta - 1)$,$N_8(\xi,\eta,\zeta) = \frac{9}{2}\xi\zeta(3\xi - 1)$

$N_9(\xi,\eta,\zeta) = \frac{9}{2}\xi\zeta(3\zeta - 1)$,$N_{10}(\xi,\eta,\zeta) = 27\xi\zeta\eta$

………… ……………………………(3)

If nodes 6, 7, 8 and 9 are at trisection point of two straight sides as shown in **Fig 1e-1f**, then eqn (2) reduces to:

$t = t_3 + (t_1 - t_3)\xi + (t_2 - t_3)\eta + \frac{9}{2}\xi\eta(-t_1 - t_2 - 2t_3 - t_4 - t_5 + 6t_{10}) + \frac{9}{2}\xi^2\eta(t_2 + 2t_3 + 3t_4 - 6t_{10})\ +$
$\frac{9}{2}\xi\eta^2(t_1 + 2t_3 + 3t_5 - 6t_{10})$, (t=x, y)……………………………………………….(4)

which shows that the curve of eqn (3) passing through the points $(x_1, y_1), (x_2, y_2), (x_4, y_4), (x_5, y_5)$ is a cubic curve. This can be shown by substituting from $\xi + \eta - 1 = 0$ and eliminating one of the variables $\xi\ or\ \eta$ in eqn (3). In general, it is shown that cubic curve is not desirable as an approximation to a simple smooth curve [1-2,1-3,1-4]. However if we choose,

$x_5 = x_4 - \frac{1}{3}(x_1 - x_2),\ \ y_5 = y_4 - \frac{1}{3}(y_1 - y_2)$ and $t_{10} = \frac{1}{12}(t_1 + t_2 + 4t_3 + 3t_4 + 3t_5),\ \ (t = x, y)$

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.\ (5)$

The equation (3) for $\xi + \eta - 1 = 0$, is a cubic curve through $(x_1, y_1), (x_2, y_2), (x_4, y_4), (x_5, y_5)$ degenerates in a unique parabola through the four points $(x_1, y_1), (x_4, y_4)$, $(x_5, y_5) \equiv \left(x_4 - \frac{1}{3}x_1 + \frac{1}{3}x_2, y_4 - \right.$ *13y1+13y2* and *x2, y2* and hence the transformation equation in eq(3) reduces to

$t = t_3 + (t_1 - t_3)\xi + (t_2 - t_3)\eta + \frac{9}{4}\xi\eta(t_4 + t_5 - t_1 - t_2) \quad (t = x, y)$ ……………(6a)

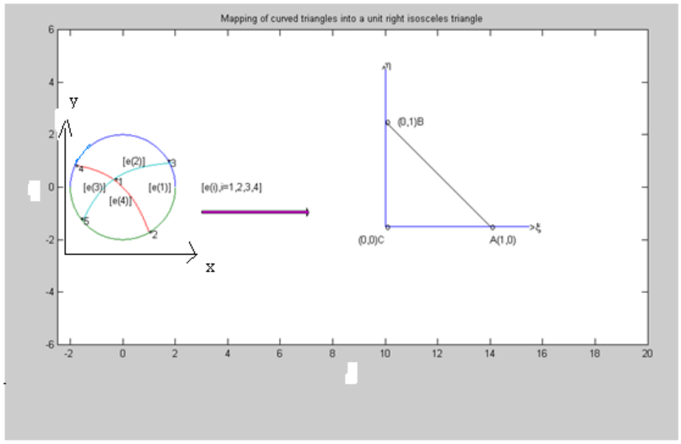
Equations (5a) can be written as

$x = a_{00} + a_{10}\xi + a_{01}\eta + a_{11}\xi\eta, \quad y = b_{00} + b_{10}\xi + b_{01}\eta + b_{11}\xi\eta$ …………………(6b)

where $a_{00} = x_3, \; a_{10} = x_1 - x_3, \; a_{01} = x_2 - x_3, \; a_{11} = \frac{9}{4}(x_4 + x_5 - x_1 - x_2),$

$b_{00} = y_3, \; b_{10} = y_1 - y_3, \; b_{01} = y_2 - y_3, \; b_{11} = \frac{9}{4}(y_4 + y_5 - y_1 - y)$

………………(6c)



.

Fig.1c-a circle divided into four arbitrary curved triangles (e(i),i=1,2,3,4)

Fig.1d-a right isosceles triangle ABC

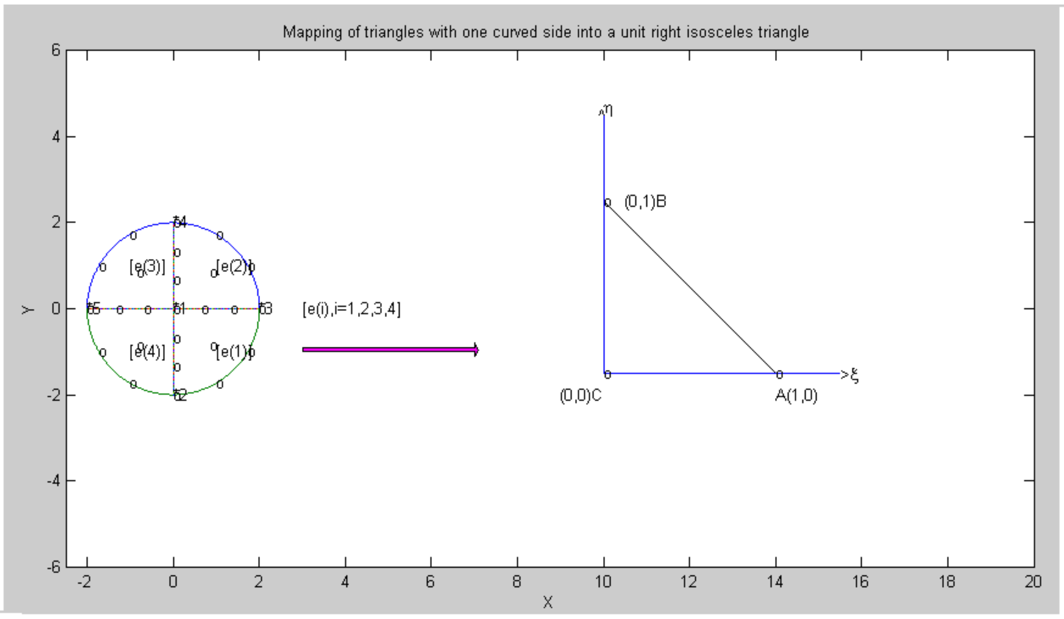
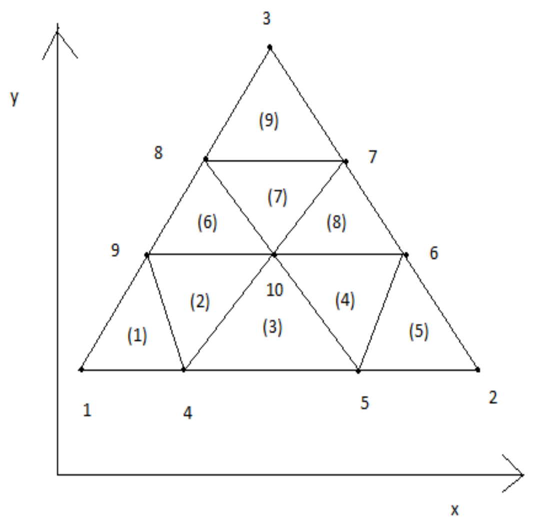

Mapping of triangles with one curved side into a unit right isosceles triangle
[e(3)]
[e(2)]
[e(4)]
[e(1)]
[e(i),i=1,2,3,4]
η
(0,1)B
ξ
(0,0)C
A(1,0)
X
Y

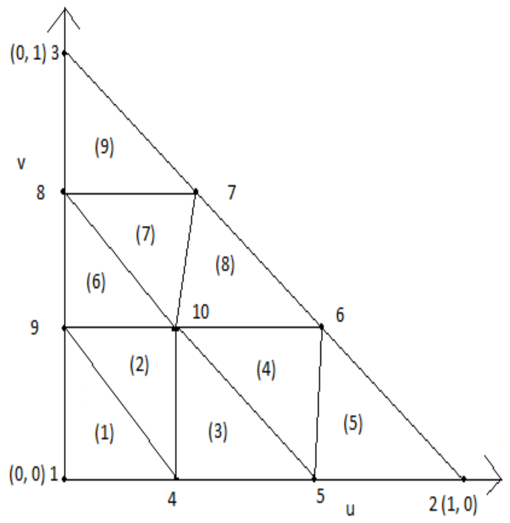
Fig.1e-a circle divided into four curved triangles (e(i),i=1,2,3,4)

(by a curved triangle, here we mean a triangle with two straight sides and one curved side)

Fig.1f-a right isosceles triangle ABC



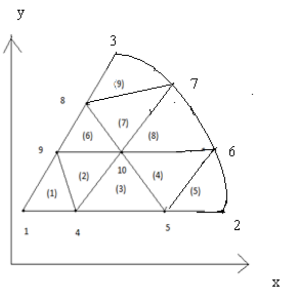
2(a)



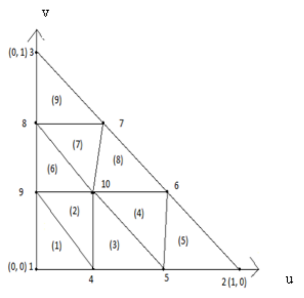
2(b)

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



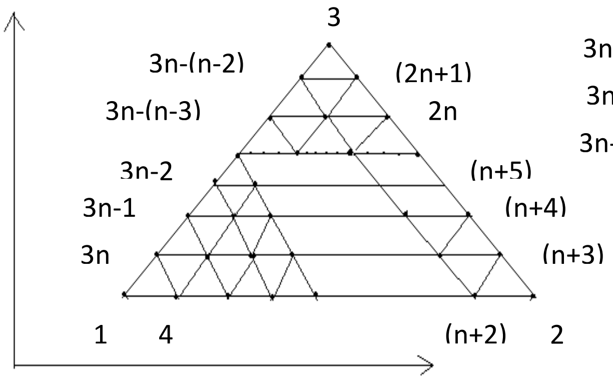
2(c)



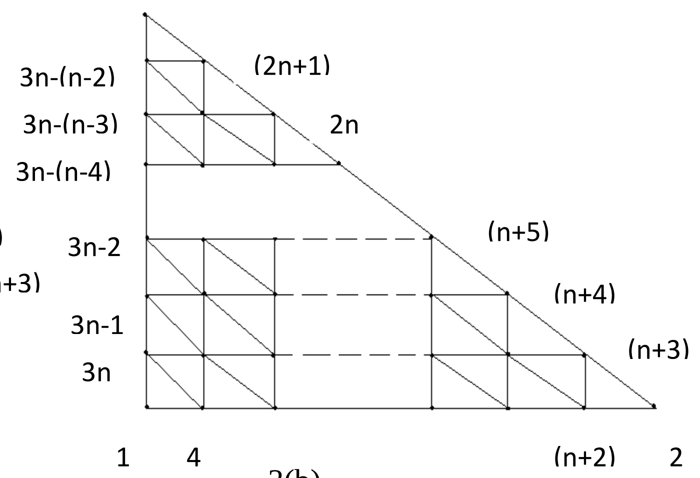
2(d)

Fig. 2-c Division of a curved triangle into nine triangles in Cartesian space (x,y)

Fig. 2-d Division of a right isosceles 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

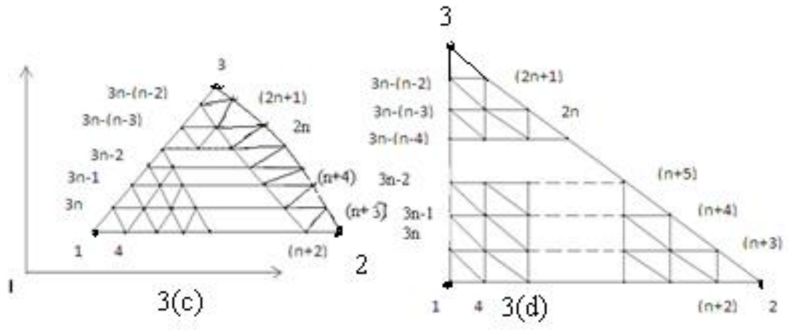


Fig.3c Division of a curved triangle into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisons of equal width

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

We have shown the division of arbitrary linear triangle and curved triangle in Fig. 3a-b.,and Fig.3c-d respectively. We divided each side of the triangles (either in Cartesian space or natural space) into n equal parts and drawn lines parallel to the sides of the triangles. This creates (n+1) (n+2)/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

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

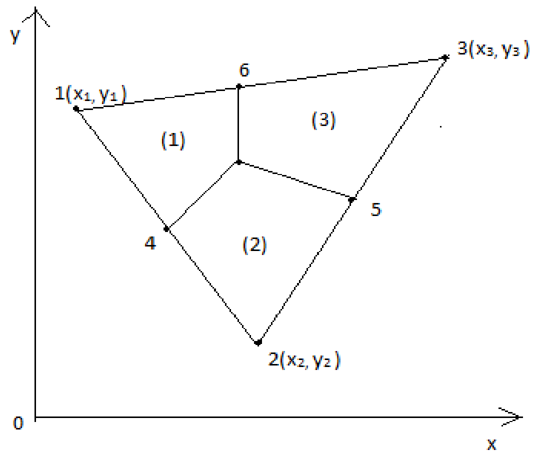
………………………………………..(7)

We note that in Fig.3c-d ,a curved triangle is mapped into a standard triangle.Since ,we are interested in linear triangles in the interior of curved triangle ,all the interior arcs in Cartesian space are approximated by straight sides and only boundary arcs of curved triangle in Cartesian space are approximated by parabolic arcs.We may also note that lines lines parallel to ξ=constant and η=constant in parametric space are mapped into straight lines in Cartesian space (x,y) under the transformations of eqns(6a,b,c),since in this case mapping from (x,y) to (ξ ,η) reduce to linear equations in either ξ or η only.

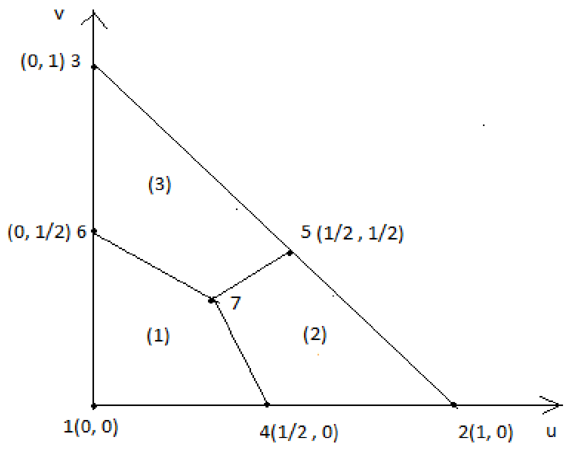
3. Quadrangulations of an Arbitrary Linear Triangle and a Curved Triangle

We now consider the quadrangulation of an arbitrary linear 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 Linear Triangle



4(a)

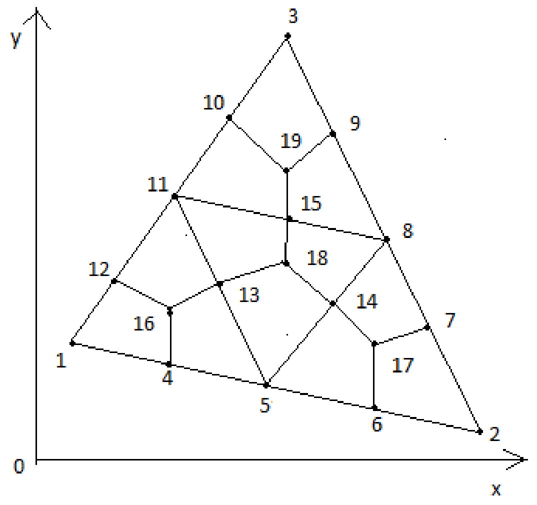


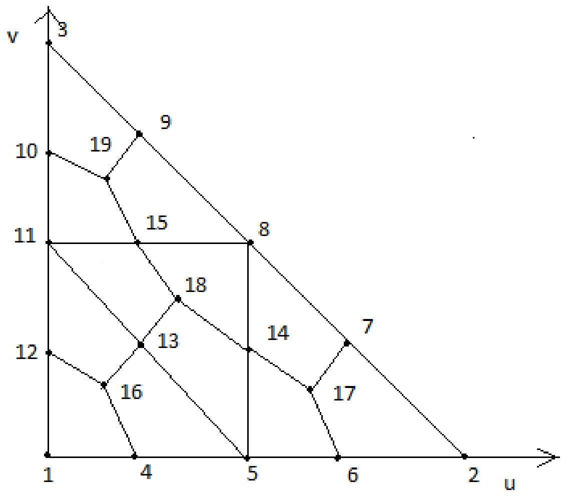
4(b)

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 Linear 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 linear triangle into equal size six node triangles, 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.**Similar divisions can be performed on a curved triangle.**This is possible because a curved triangle can also be mapped into standard triangle as explained in section 2.2 of this paper.We can locate unique points in Cartesian space for each point in the standard triangle of the parametric space. 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.(7). 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 triangles using sub matrix $\underline{rr}$

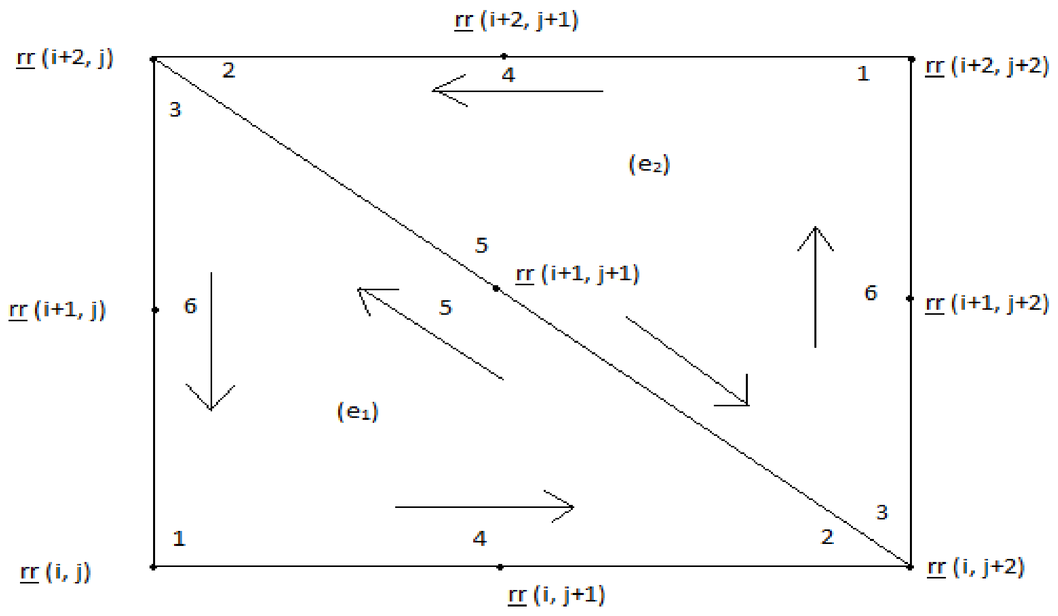


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

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 , wherein we read $\{\ldots\ldots\ldots\}$ as connecting the nodal addresses

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

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

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

If the elements of sub matrix ($(\underline{rr}\ (\ k,l), k = i, i+1, i+2),\ l = j,\ j+1,\ j+2)$ are nonzero, then as stated earlier, we can construct two six node 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} , Q_{3n_1-1} , Q_{3n_1}

wherein we read $\{\ldots\ldots\ldots\}$ as connecting the nodal addresses.

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$_2$, $\underline{rr}$ $(i+2,\ j+1)$, $\underline{rr}\,(i+2,\ j)$, $\underline{rr}\,(i+1,\ j+1)$ }

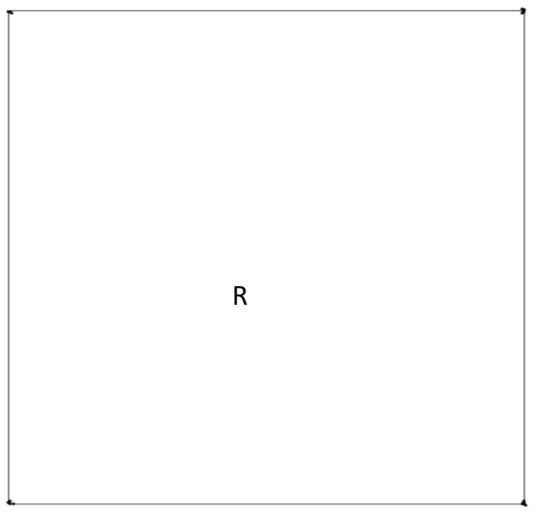
Q_{3n_1} { c$_2$, $\underline{rr}$ $(i+1,\ j+1)$, $\underline{rr}\,(i,\ j+2)$, $\underline{rr}\,(i+1, j+2)$ } -------------------- (5)

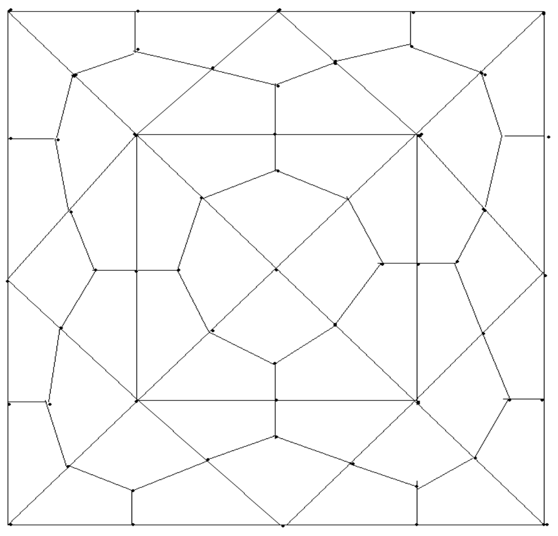
wherein again we read $\{\dots\dots\dots\}$ as connecting the nodal addresses.

4. Quadrangulation of the Analytical Curved Surfaces by using Parabolic Arcs

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

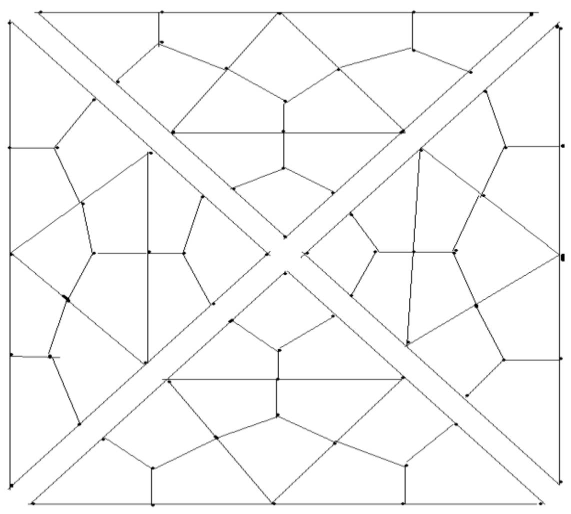
As an example, consider the geometry shown in Fig. 8(a). This is a rectangular region which is simply chosen for illustration. We divide this region into four LOOPs as shown in Fig.8(d). These LOOPs 1,2,3 and 4 are triangles each with three sides. After the LOOPs are defined, the number of elements for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

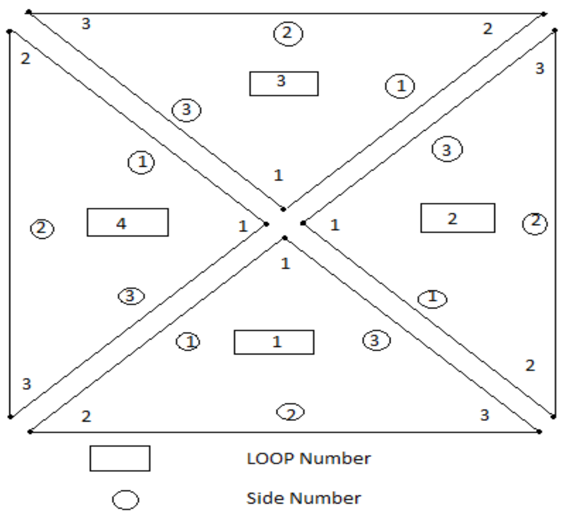




8 (a) 8 (b)

(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

We next illustrate the above procedure for an analytical curved surface ,this shown in Figs.9a-9d ,with reference to an elliptical region.

A CURVED DOMAIN IN THE SHAPE OF AN ELLIPSE

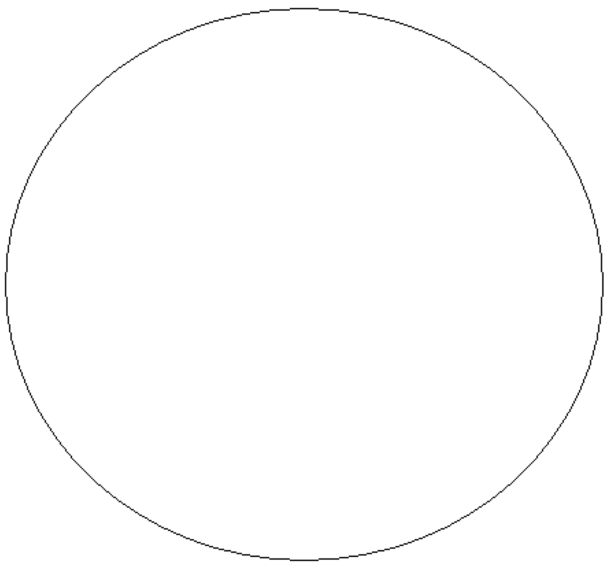


Fig.9a

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

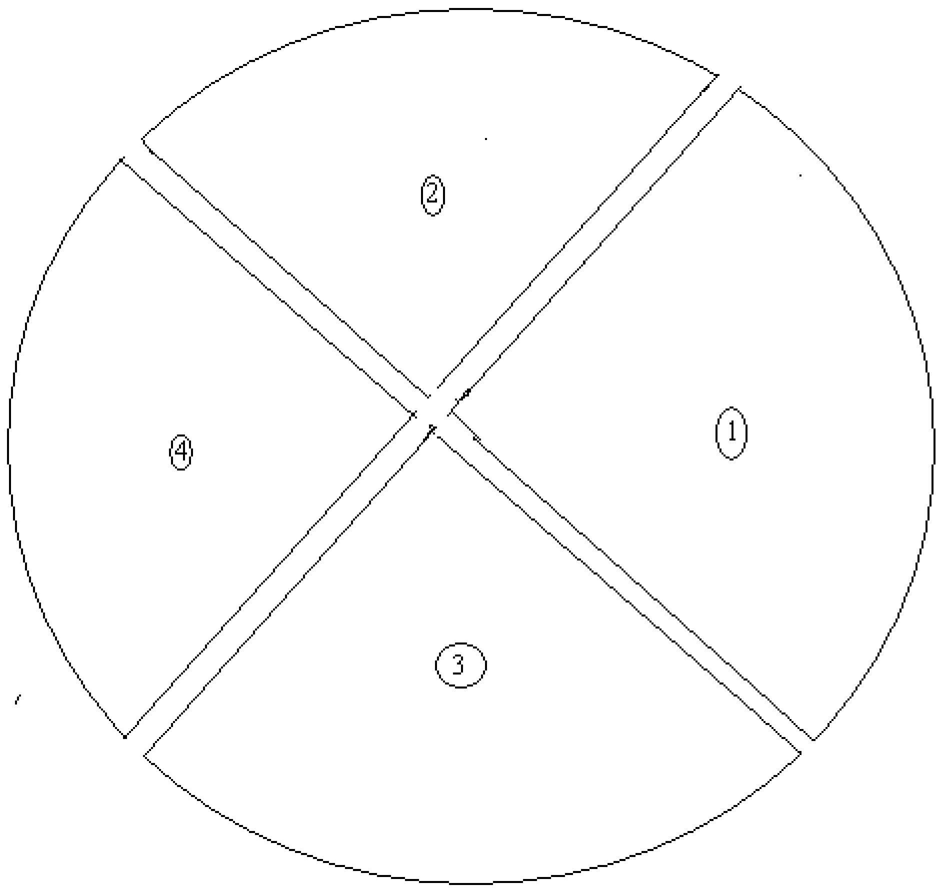


Fig.9b

ELLIPSE AS A CURVED DOMAIN MADE UP OF TWELVE QUADRILATERALS

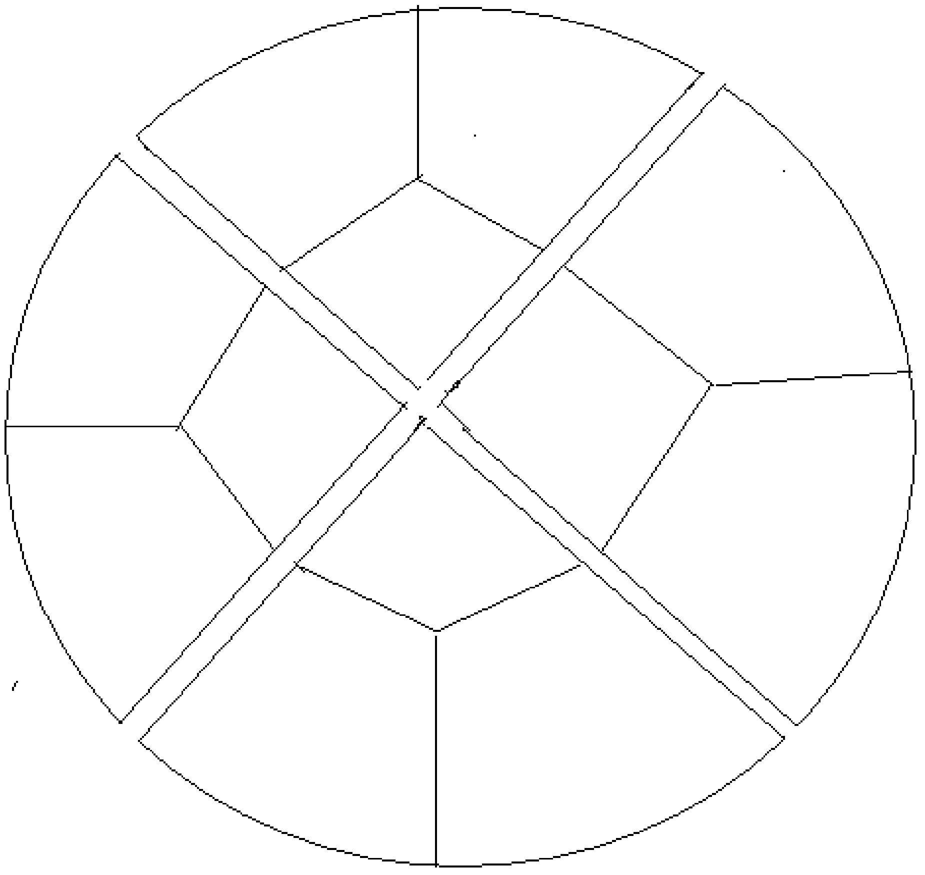


Fig.9c

AN ELLIPSE AS CURVED DOMAIN DISCRITISED INTO TWELVE QUADRILATERALS

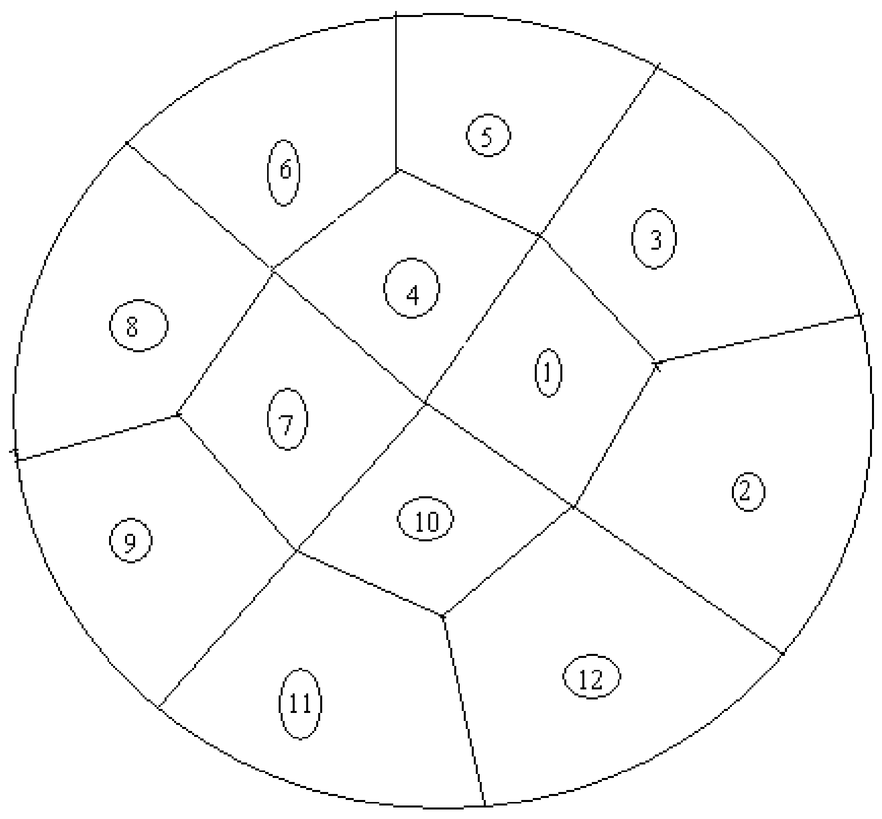


Fig.9d

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

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

In the present mesh generation code, we aim to create an all quadrilateral mesh for an analytical curved surface. This requires a simple procedure. We join side 3 0f LOOP 1 to side 1 of LOOP 2, side 3 of LOOP 2 will joined to side 1 of LOOP 3, side 3 of LOOP 3 will be joined to side 1 of LOOP 4. Finally side 3 of LOOP 4 will be joined to side 1 of LOOP 1.

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

5. Application Examples

In authors' recent work[25-28], an automatic indirect quadrilateral mesh generator which uses the splitting technique is presented for the two dimensional convex polygonal domains.It presents the mesh generation over an arbitrary linear triangle and also the mesh generation for a convex polygonal domain.In the present paper, our aim is to generate a finite element mesh of all quadrilaterals over an analytical curved surface.

5.1 Mesh Generation Over a Curved Triangle

In applications to boundary value problems due to symmetry considerations, we may have to discretize a curved triangle. Our purpose is to have a code which automatically generates con quadrangulations of the domain by assuming the input as coordinates of the vertices. We use the theory and procedure developed in sections 2,3 and 4 of this paper for this purpose. The mesh generation of this paper uses the parametric equations of straight lines for all the interior points of the domain.The boundary curve is approximated by parabolic arcs.We have adopted the the following procedure:

We determine the points in the standard triangle.We divide standard triangle into n^2 sub -triangles of equal area.This can be done by dividing each side into n equal parts and then joining these points appropriately by straight lines to generate smaller right isosceles triangles which requires (n+1)(n+2)/2 nodal points.We then find the corresponding points in the Cartesian space for the curved by the eqns(6b)

$x = a_{00} + a_{10}\xi + a_{01}\eta + a_{11}\xi\eta,\ \ y = b_{00} + b_{10}\xi + b_{01}\eta + b_{11}\xi\eta$ …………………(6b)

where $a_{00} = x_3,\ a_{10} = x_1 - x_3,\ a_{01} = x_2 - x_3,\ a_{11} = \frac{9}{4}(x_4 + x_5 - x_1 - x_2),$

$b_{00} = y_3,\ b_{10} = y_1 - y_3,\ b_{01} = y_2 - y_3,\ b_{11} = \frac{9}{4}(y_4 + y_5 - y_1 - y)$

and($(x_i; y_i)$,i=1,2,3) are the three vertices of the triangle,for a curved boundary the points ((x_i, y_i),i=4,5)

must be found by satisfying the relation

$x_5 = x_4 - \frac{1}{3}(x_1 - x_2),\ \ y_5 = y_4 - \frac{1}{3}(y_1 - y_2)$ and $t_{10} = \frac{1}{12}(t_1 + t_2 + 4t_3 + 3t_4 + 3t_5),\ \ (t = x, y)$

The curve corresponds to ξ+η=1 ,on substituting this in eqn(6c) ,we obtain the equation of parabolic arc which is a quadrtic equation in one of the variables, either in ξ or η. The most important thing is to decide on the boundary nodes passing through this parabolic arc .The boundary nodes are along the edge joining points ($(x_i\,, y_i)$,i=1,2,4,5).However ($(x_i\,, y_i)$,i=4,5) are not used in the global node numbering.The nodes on the edge2-3 wil be known to us as output from a program. The parametric equations of the curved boundary can be obtained from eqn( ) as function of one of the variables,say,η from eqn(6b) as

t(1-η,η)=$t_1 + \eta\left[(t_2 - t_1) + \frac{9}{4}(t_4 + t_5 - t_1 - t_2)\right] - \eta^2\ \left[\frac{9}{4}(t_4 + t_5 - t_1 - t_2)\right], (t = x, y), 0 \le \eta \le 1$

………………..(8)

The division of the boundary into n equal parts can be done by obtaining (n+1) points:

T_i =t(1-η_i ,η_i), η_i =$\frac{(i-1)}{n}$, $\mathrm{i} = 1,2, \dots, \mathrm{n}, (\mathrm{n}+1)$; clearly,$T_1$ =t_1 and $T_{(n+1)}$ =t_2 ………………………………..(9)

The eqn(8-9) is used to plot the curved boundary.However,since we want to generate straight edge quadrilaterals in the interior of the domain centroid (average of three vertices) of the respective interior triangle is used.

We illustrate the mesh generation for a curved triangle with reference to a quadrant of circular region.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 Over an Analytical Curved Surface

In several physical applications in science and engineering, the boundary value problem require meshes generated over curved surfaces whose boundary is defined by analytical equations of curves. Again our aim is to have a code which automatically generates a mesh of linear convex quadrilaterals in the interior of the domain and quadrilaterals with three straight edges and one curved edge near to the boundary of curved surface for the complex domains such as those in [19-23]. We use the theory and procedure developed in sections 2, 3 and 4. The following MATLAB codes are written for this purpose.

Essential Programs
(1)curvedquadrilateralmesh4convexpolygoneightsidesq40.m
(2)curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40.m
(3)curvedpolygonal_domain_QUADcoordinates.m
(4) nodaladdresses_special_convex_quadrilaterals.m
(5) nodaladdresses4special_convex_quadrilateralsQUADtrial.m
Application Specific Programs
(i)A Quarter Circle and A Circular Disk
(1a) masterelementnodescoordinates_circulardisk.m
(1b) globalnodalcoordinate_circulardisk.m
(ii)LUNAR MODEL
(2a) masterelementnodescoordinates.m
(2b) globalnodalcoordinate.m
(iii)Three Quadrants of a Circle
(3a) masterelementnodescoordinates_circulardisk.m
(3b) globalnodalcoordinate_circulardisk
(iv) A Cracked Circular Disk
(4a) masterelementnodescoordinates_circularshaftkeywayascrack.m
(4b) globalnodalcoordinate_circularshaft_keywayascrack.m
SOME MATLAB COMMANDS

```
>>curvedquadrilateralmesh4convexpolygoneightsidesq40([3],[1],[2],[4],[5],3,100,20,15,1,1)
>>curvedquadrilateralmesh4convexpolygoneightsidesq4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],[10;12;14
;16;18;20;22;24],[11;13;15;17;19;21;23;25],9,1,2,11,1,1)
>>n1=[5;5;5;5];n2=[1;2;3;4];n3=[2;3;4;1];n4=[6;8;10;12];n5=[7;9;11;13];
nmax=5;mesh=16;xlength=1;ylength=1;
for ndiv=2:2:20
numtri=(ndiv/2)^2;
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh,xlength,ylen
gth)
end
```

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 quarter circle,a circular disk,a lunar model,three quadrants of a circle and a cracked circular disk. Some sample commands to generate the curved domains stated above also appear in the comment lines of programs. In all the cases the sample data is a part of the codes.

Conclusions

An automatic indirect quadrilateral mesh generator which uses the splitting technique is presented for the two dimensional analytical curved surfaces. This mesh generation is made fully automatic and allows the user to define the problem domain with minimum amount of input such as coordinates of boundary. Once this input is created, by selecting an appropriate interior point of the curved domain, we form the subdomains in the shape of curved triangles. These subdomains are then triangulated to generate a fine mesh of six node 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 curved domain into all convex linear quadrilaterals in the interior of the curved domain and quadrilaterals with three straight sides and one curved side which forms part of the curved boundary, thus propogating a uniform refinement. This simple method generates high quality mesh whose elements confirm well to the requested shape by refining the problem domain. We have also appended MATLAB programs which provide the nodal coordinates, element nodal connectivity and graphic display of the generated all quadrilateral mesh for the curved triangle, a circular disk,alunar model,three quadrants of a circular region and a cracked circular disk as few application examples. We believe that this work will be useful for various applications in science and engineering. The quality of the quadrilateral mesh can be subsequently enhanced by a series of mesh modifications and element shape improvement procedures.One advantage of the mesh is for applications to two dimensional boundary value problems,because the jacobian of all the interior quadrilaterals is linear expression,as explained in our works[29-30]. The elements near to the boundary are a few quadrilaterals having one curved side and three straight sides.Thus an algorithm based on the proposed mesh generation scheme has computational convenience and it can be easily coded.

References

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

[23]Sadd.M.H,Elasticity,Theory,Applications,and Numerics,Academic Press(2005)

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

[25] Rathod H.T, Venkatesh.B, Shivaram. K.T,Mamatha.T.M, Numerical Integration over polygonal domains using convex quadrangulation and Gauss Legendre Quadrature Rules, International Journal of Engineering and Computer Science, Vol. 2,issue 8,pp2576-2610(2013)

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

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

[28] **H**.T.Rathod, A.S.Hariprasad, K.V.Vijayakumar, Bharath Rathod, C.S.Nagabhushana, Numerical Integration Over Curved Domains using Convex Quadrangulation and Gauss Legendre Quadrature Rules, International Journal of Engineering and Computer Science ,vol.2,no.11(2013),pp3290-3332

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

[30] H.T. Rathod, Bharath Rathod, Shivaram K.T , H. Y. Shrivalli , Tara Rathod ,K. Sugantha Devi ,An explicit finite element integration scheme using automatic mesh generation technique for linear convex quadrilaterals over plane regions, International Journal of Engineering and Computer Science, vol.3,issue4(2014),pp5400-5435

Computer Programs:essential programs

program(1)

```
function[]=curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh,xlength,ylength)

%curvedquadrilateralmesh4convexpolygoneightsidesq4([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;10;12],[7;9;11;13],5,1,2,11,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],[10;12;14;16;18;20;22;24],[11;13;15;17;19;21;23;2
5],9,1,2,11,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;10;12],[7;9;11;13],5,1,2,12,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;10;12],[7;9;11;13],5,4,4,12,1,1)
%curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40([7;7;7;7;7;7],[1;2;3;4;5;6],[2;3;4;5;6;1],[8;10;12;14;16;18],[9;11;13;15;17;19],7,1,2,13,1,1)
%curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40([4;4;4],[1;2;3],[2;3;1],[5;7;9],[6;8;10],4,1,2,16,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],[10;12;14;16;18;20;22;24],[11;13;15;17;19;21;23;
25],9,1,2,11,1,1)
global edgen2n3
global TRINUM ncrack nitri
global crnodes

axis equal
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 5
axis([0 xlength 0 ylength])

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


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

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

  axis([0 xlength 0 ylength])
  case 12

  axis([0 xlength 0 ylength])
  case 13
yl=ylength/2;
axis([0 xlength  -yl yl])
  case 14
```

```
yl=ylength/2;
axis([0 xlength  -yl yl])

   case 16

  axis([0 xlength 0 ylength])

end
TRINUM=0;%acounter for number of coarse triangles
%ncrack=1 if the cracked circular disk has to be a perfect mesh generation model
%ncrack=2,3,....will change the element node numbering
%ncrack=input('for cracked domain,enter the number of triangles to be deleted=')
ncrack=1;

[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinates(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);
figure(ndiv/2)
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
  edgen2n3

 nitri=nmax-1;
if (nmax==3)
    nitri=1;
end
 numtri=(ndiv/2)^2

if mesh==11
     nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates(nelm);
[cord]=globalnodalcoordinate(mst_tri,theta,cord);
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==12
     nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm);
[cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord);
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==13
nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeyway(nelm)
[cord]=globalnodalcoordinate_circularshaft_keyway(mst_tri,theta,cord)
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end


if mesh==14
nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeywayascrack(nelm)
[cord]=globalnodalcoordinate_circularshaft_keywayascrack(mst_tri,theta,cord)
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==16
nelm=nmax-1
 nelm=nmax-1;
 [mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord)
```

```
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end

[nel,nnel]=size(nodes)
nel=nel-ncrack*(3*(ndiv/2)^2)
nitri=nitri-ncrack
%NITRI=nitri
spqd=nodes(1:nel,1:nnel)
nnode=max(max(spqd))


for itri=1:nitri

disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1) ]
x3=xx(n1(itri,1),1)
x1=xx(n2(itri,1),1)
x2=xx(n3(itri,1),1)
x4=xx(n4(itri,1),1)
x5=xx(n5(itri,1),1)
%
y3=yy(n1(itri,1),1)
y1=yy(n2(itri,1),1)
y2=yy(n3(itri,1),1)
y4=yy(n4(itri,1),1)
y5=yy(n5(itri,1),1)

for i=(itri-1)*numtri*3+1:itri*numtri*3
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)];
if (crnodes(i,2)==2)&(crnodes(i,3)~=0)
   disp('error,first element node on the boundary curve')
   break
end
 %
%plot(xvec,yvec);%plot element

%if ((crnodes(i,2)~=0)|(crnodes(i,3)~=0))
if ((crnodes(i,1)==i)&(crnodes(i,2)<=1))%plot all elements except those having a ponit on the boundary
%if ((crnodes(i,1)==i)&(crnodes(i,2)==0))
%if ((crnodes(i,1)==i))%PLOT ALL THE ELEMENTS

disp('element no. i=')
disp(crnodes(i,1))

xi=x(1,1);xj=x(1,2);xk=x(1,3);xl=x(1,4);
yi=y(1,1);yj=y(1,2);yk=y(1,3);yl=y(1,4);
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;xjk=xj+(xk-xj)*s;xkl=xk+(xl-xk)*s;xli=xl+(xi-xl)*s;
yij=yi+(yj-yi)*s;yjk=yj+(yk-yj)*s;ykl=yk+(yl-yk)*s;yli=yl+(yi-yl)*s;
plot(xij,yij)
hold on
plot(xjk,yjk)
hold on
plot(xkl,ykl)
hold on
plot(xli,yli)
hold on
end
%place element number
%if (ndiv<=6)
%midx=mean(xvec(1,1:4))
%midy=mean(yvec(1,1:4))
%text(midx,midy,['[',num2str(i),']']);
%end% if ndiv
%plot the elements on the boundary
if crnodes(i,2)==2
%first check for element nodes 2&3
xi=x(1,1);xj=x(1,2);xk=x(1,3);xl=x(1,4);
yi=y(1,1);yj=y(1,2);yk=y(1,3);yl=y(1,4);
```

```
if ((crnodes(i,4)~=0)&(crnodes(i,5)~=0))
for k=1:ndiv
if (crnodes(i,4)==edgen2n3(k,itri))&(crnodes(i,5)==edgen2n3(k+1,itri))
   s0=(k-1)/ndiv;s1=k/ndiv;
end
end
% plot straight sides of boundary element
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;yij=yi+(yj-yi)*s;
plot(xij,yij)
hold on
xkl=xk+(xl-xk)*s;ykl=yk+(yl-yk)*s;
plot(xkl,ykl)
hold on
xli=xl+(xi-xl)*s;yli=yl+(yi-yl)*s;
plot(xli,yli)
hold on
%now plot the parabolic arc for the boundary  element
ss=linspace(s0,s1,101);
xxx=x1+((x2-x1)+2.25*(x4+x5-x1-x2))*ss-2.25*(x4+x5-x1-x2)*ss.*ss;
yyy=y1+((y2-y1)+2.25*(y4+y5-y1-y2))*ss-2.25*(y4+y5-y1-y2)*ss.*ss;
plot(xxx,yyy)
hold on
end
%next check for element nodes 3&4
if ((crnodes(i,5)~=0)&(crnodes(i,6)~=0))
for k=1:ndiv
if (crnodes(i,5)==edgen2n3(k,itri))&(crnodes(i,6)==edgen2n3(k+1,itri))
   s0=(k-1)/ndiv;s1=k/ndiv;
end
end
% plot straight sides of boundary element
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;yij=yi+(yj-yi)*s;
plot(xij,yij)
hold on
xjk=xj+(xk-xj)*s;yjk=yj+(yk-yj)*s;
plot(xjk,yjk)
hold on
xli=xl+(xi-xl)*s;yli=yl+(yi-yl)*s;
plot(xli,yli)
hold on
%now plot the parabolic arc for the boundary  element
ss=linspace(s0,s1,101);
xxx=x1+((x2-x1)+2.25*(x4+x5-x1-x2))*ss-2.25*(x4+x5-x1-x2)*ss.*ss;
yyy=y1+((y2-y1)+2.25*(y4+y5-y1-y2))*ss-2.25*(y4+y5-y1-y2)*ss.*ss;
plot(xxx,yyy)
hold on
end
end%crnodes(i,2)==2
%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
end%itri loop
%end%if crnodes
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])

if( ncrack==1)&(mesh==14)
text(0.,0.,'cracked circular disk');
end

if( ncrack==1)&(mesh==16)
```

```
text(0.6,0.3,'three quadrants of circular disk');
end
%put node numbers
if (ndiv<=4)
for jj=1:nnode
text(xcoord(jj,1),ycoord(jj,1),['o',num2str(jj)]);
end
end%if ndiv
JJ=(1:nnode)';
%axis off
nitri
edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
[crnodes nodes]
[JJ xcoord(1:nnode,1) ycoord(1:nnode,1)]

TRINUM

nitri
ncrack
```

program(2)

```
function[]=curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh,xlength,ylength)
%n1=[5;5;5;5];n2=[1;2;3;4];n3=[2;3;4;1];n4=[6;8;10;12];n5=[7;9;11;13];nmax=5;mesh=16;xlength=1;ylength=1;
%for ndiv=2:2:20
%numtri=(ndiv/2)^2;
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh,xlength,ylength)
end
%input number of triangles to be deleted=ncrack, give as online input data
global edgen2n3
global TRINUM ncrack nitri
global crnodes

axis equal
switch mesh
case 1
axis([0 xlength 0 ylength])
case 2
axis([0 xlength 0 ylength])
case 3
 xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 4
xl=xlength/2;yl=ylength/2;
axis([-xl xl -yl yl])
case 5
axis([0 xlength 0 ylength])

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


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

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

  axis([0 xlength 0 ylength])
  case 12

  axis([0 xlength 0 ylength])
  case 13
yl=ylength/2;
axis([0 xlength  -yl yl])
  case 14
yl=ylength/2;
axis([0 xlength  -yl yl])
```

```
  case 16

  axis([0 xlength 0 ylength])

end
TRINUM=0;%acounter for number of coarse triangles
%ncrack=1 if the cracked circular disk has to be a perfect mesh generation model
%ncrack=2,3,....will change the element node numbering
%ncrack=input('for cracked domain,enter the number of triangles to be deleted=')
ncrack=1;

[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinates(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);
figure(ndiv/2)
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
  edgen2n3

 nitri=nmax-1;
if (nmax==3)
   nitri=1;
end
 numtri=(ndiv/2)^2

if mesh==11
    nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates(nelm);
[cord]=globalnodalcoordinate(mst_tri,theta,cord);
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==12
    nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm);
[cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord);
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==13
nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeyway(nelm)
[cord]=globalnodalcoordinate_circularshaft_keyway(mst_tri,theta,cord)
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end


if mesh==14
nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeywayascrack(nelm)
[cord]=globalnodalcoordinate_circularshaft_keywayascrack(mst_tri,theta,cord)
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
end
if mesh==16
nelm=nmax-1
 nelm=nmax-1;
 [mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord)
xx=cord(:,1)
yy=cord(:,2)
[n1 n2 n3 n4 n5]
```

```
end

[nel,nnel]=size(nodes)
nel=nel-ncrack*(3*(ndiv/2)^2)
nitri=nitri-ncrack
%NITRI=nitri
spqd=nodes(1:nel,1:nnel)
nnode=max(max(spqd))


for itri=1:nitri

disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1) ]
x3=xx(n1(itri,1),1)
x1=xx(n2(itri,1),1)
x2=xx(n3(itri,1),1)
x4=xx(n4(itri,1),1)
x5=xx(n5(itri,1),1)
%
y3=yy(n1(itri,1),1)
y1=yy(n2(itri,1),1)
y2=yy(n3(itri,1),1)
y4=yy(n4(itri,1),1)
y5=yy(n5(itri,1),1)

for i=(itri-1)*numtri*3+1:itri*numtri*3
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)];
if (crnodes(i,2)==2)&(crnodes(i,3)~=0)
   disp('error,first element node on the boundary curve')
   break
end
 %
%plot(xvec,yvec);%plot element

%if ((crnodes(i,2)~=0)|(crnodes(i,3)~=0))
if ((crnodes(i,1)==i)&(crnodes(i,2)<=1))%plot all elements except those having a ponit on the boundary
%if ((crnodes(i,1)==i)&(crnodes(i,2)==0))
%if ((crnodes(i,1)==i))%PLOT ALL THE ELEMENTS

disp('element no. i=')
disp(crnodes(i,1))

xi=x(1,1);xj=x(1,2);xk=x(1,3);xl=x(1,4);
yi=y(1,1);yj=y(1,2);yk=y(1,3);yl=y(1,4);
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;xjk=xj+(xk-xj)*s;xkl=xk+(xl-xk)*s;xli=xl+(xi-xl)*s;
yij=yi+(yj-yi)*s;yjk=yj+(yk-yj)*s;ykl=yk+(yl-yk)*s;yli=yl+(yi-yl)*s;
plot(xij,yij)
hold on
plot(xjk,yjk)
hold on
plot(xkl,ykl)
hold on
plot(xli,yli)
hold on
end
%place element number
%if (ndiv<=6)
%midx=mean(xvec(1,1:4))
%midy=mean(yvec(1,1:4))
%text(midx,midy,['[',num2str(i),']']);
%end% if ndiv
%plot the elements on the boundary
if crnodes(i,2)==2
%first check for element nodes 2&3
xi=x(1,1);xj=x(1,2);xk=x(1,3);xl=x(1,4);
yi=y(1,1);yj=y(1,2);yk=y(1,3);yl=y(1,4);
if ((crnodes(i,4)~=0)&(crnodes(i,5)~=0))
for k=1:ndiv
if (crnodes(i,4)==edgen2n3(k,itri))&(crnodes(i,5)==edgen2n3(k+1,itri))
```

```
   s0=(k-1)/ndiv;s1=k/ndiv;
end
end
% plot straight sides of boundary element
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;yij=yi+(yj-yi)*s;
plot(xij,yij)
hold on
xkl=xk+(xl-xk)*s;ykl=yk+(yl-yk)*s;
plot(xkl,ykl)
hold on
xli=xl+(xi-xl)*s;yli=yl+(yi-yl)*s;
plot(xli,yli)
hold on
%now plot the parabolic arc for the boundary  element
ss=linspace(s0,s1,101);
xxx=x1+((x2-x1)+2.25*(x4+x5-x1-x2))*ss-2.25*(x4+x5-x1-x2)*ss.*ss;
yyy=y1+((y2-y1)+2.25*(y4+y5-y1-y2))*ss-2.25*(y4+y5-y1-y2)*ss.*ss;
plot(xxx,yyy)
hold on
end
%next check for element nodes 3&4
if ((crnodes(i,5)~=0)&(crnodes(i,6)~=0))
for k=1:ndiv
if (crnodes(i,5)==edgen2n3(k,itri))&(crnodes(i,6)==edgen2n3(k+1,itri))
   s0=(k-1)/ndiv;s1=k/ndiv;
end
end
% plot straight sides of boundary element
s=linspace(0,1,101);
xij=xi+(xj-xi)*s;yij=yi+(yj-yi)*s;
plot(xij,yij)
hold on
xjk=xj+(xk-xj)*s;yjk=yj+(yk-yj)*s;
plot(xjk,yjk)
hold on
xli=xl+(xi-xl)*s;yli=yl+(yi-yl)*s;
plot(xli,yli)
hold on
%now plot the parabolic arc for the boundary  element
ss=linspace(s0,s1,101);
xxx=x1+((x2-x1)+2.25*(x4+x5-x1-x2))*ss-2.25*(x4+x5-x1-x2)*ss.*ss;
yyy=y1+((y2-y1)+2.25*(y4+y5-y1-y2))*ss-2.25*(y4+y5-y1-y2)*ss.*ss;
plot(xxx,yyy)
hold on
end
end%crnodes(i,2)==2
%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
end%itri loop
%end%if crnodes
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])

if( ncrack==1)&(mesh==14)
text(0.,0.,'cracked circular disk');
end

if( ncrack==1)&(mesh==16)
text(0.6,0.3,'three quadrants of circular disk');
end
%put node numbers
```

```
if (ndiv<=4)
for jj=1:nnode
text(xcoord(jj,1),ycoord(jj,1),['o',num2str(jj)]);
end
end%if ndiv
JJ=(1:nnode)';
%axis off
nitri
edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
[crnodes nodes]
[JJ xcoord(1:nnode,1) ycoord(1:nnode,1)]

TRINUM

nitri
ncrack
```

program(3)

```
function[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinates(n1,n2,n3,n4,n5,nmax,numtri,n,mesh)
%nelm=numbr of elements(4,8,12,....etc)
%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
global edgen2n3
global TRINUM ncrack nitri
global crnodes
%
syms U V W
switch mesh
case 11%lunar model
nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates(nelm);
[cord]=globalnodalcoordinate(mst_tri,theta,cord);
x=cord(:,1)
y=cord(:,2)
[n1 n2 n3 n4 n5]
case 12
 nelm=nmax-1
[mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm);
[cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord);
x=cord(:,1)
y=cord(:,2)
[n1 n2 n3 n4 n5]
case 14
 nelm=nmax-1
 [mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeywayascrack(nelm)
 [cord]=globalnodalcoordinate_circularshaft_keywayascrack(mst_tri,theta,cord)
x=cord(:,1)
y=cord(:,2)
[n1 n2 n3 n4 n5]
case 15
  nelm=1
 [mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord)
x=cord(:,1)
y=cord(:,2)
[n1 n2 n3 n4 n5]
 case 16
  nelm=nmax-1;
 [mst_tri,theta,cord]=masterelementnodescoordinates_circulardisk(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [cord]=globalnodalcoordinate_circulardisk(mst_tri,theta,cord)
x=cord(:,1)
y=cord(:,2)
[n1 n2 n3 n4 n5]


end
```

```
if (nmax>3)
[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrial(n1,n2,n3,nmax,numtri,n)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilateralsQUADtrial(n1,n2,n3,nmax,numtri,n);
[nel,nnel]=size(nodes)
%nel=nel-ncrack*(3*(n/2)^2)
%nitri=nitri-ncrack
%NITRI=nitri
spqd=nodes(1:nel,1:nnel)
nnode=max(max(spqd))
eln

end
if (nmax==3)
   [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals(n);
   %rrr
  edgen2n3=edgen2n3(1:n+1,1);
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)])
%==========================================================

%================================================================
%to decide about a boundary element
%crnodes is a (nel X 6) MATRIX
%first column of crnodes stores element number
%second column of crnodes stores the number of nodes on curved boundary:it
%should be either of 0,1,2 only
%third,fourth,fifth and sixth column of crnodes stores first,second,third,fourth  node numbers of curved boundary
%element
%1- if third to sixth columns of crnodes are zeros then that element
%...is not a boundary element
%2- if   two of the third to sixth columns of crnodes are nonzeros then that element
%...is a boundary element
%=======================================================================
%
ndiv=n
  crnodes=zeros(nel,nnel+2);

nitri=nmax-1;
if (nmax==3)
   nitri=1;
end
numtri=(ndiv/2)^2
for itri=1:nitri
for i=(itri-1)*numtri*3+1:itri*numtri*3
   crnodes(i,1)=i;
      ncrnd=0;
     for k=1:(ndiv+1)

         if(edgen2n3(k,itri)==nodes(i,1))
            ncrnd=ncrnd+1;
            crnodes(i,2)=ncrnd;
            crnodes(i,3)=nodes(i,1);
         end%if
      end
     for k=1:(ndiv+1)
         if(edgen2n3(k,itri)==nodes(i,2))
            ncrnd=ncrnd+1;
            crnodes(i,2)=ncrnd;
            crnodes(i,4)=nodes(i,2);
        end%if
     end
    for k=1:(ndiv+1)
        if(edgen2n3(k,itri)==nodes(i,3))
            ncrnd=ncrnd+1;
            crnodes(i,2)=ncrnd;
            crnodes(i,5)=nodes(i,3);
        end%if
    end

    for k=1:(ndiv+1)
        if(edgen2n3(k,itri)==nodes(i,4))
            ncrnd=ncrnd+1;
            crnodes(i,2)=ncrnd;
            crnodes(i,6)=nodes(i,4);
        end%if
    end




end%for i
end%itri




%=================================================================

%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) ]
x3=x(n1(itri,1),1)
x1=x(n2(itri,1),1)
x2=x(n3(itri,1),1)
x4=x(n4(itri,1),1)
x5=x(n5(itri,1),1)
%
y3=y(n1(itri,1),1)
y1=y(n2(itri,1),1)
y2=y(n3(itri,1),1)
y4=y(n4(itri,1),1)
y5=y(n5(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);
      %UU(mm,1)=uu;VV(mm,1)=vv;
     % xi(mm,1)=x1*ww+x2*uu+x3*vv;
      %yi(mm,1)=y1*ww+y2*uu+y3*vv;
      a00=x3;a10=x1-x3;a01=x2-x3;a11=(9/4)*(x4+x5-x1-x2);
      b00=y3;b10=y1-y3;b01=y2-y3;b11=(9/4)*(y4+y5-y1-y2);

      xi(mm,1)=double(a00+a10*uu+a01*vv+a11*uu*vv);
      yi(mm,1)=double(b00+b10*uu+b01*vv+b11*uu*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)
    node4=eln(iii,4);
    node5=eln(iii,5);
    node6=eln(iii,6);
    node7=eln(iii,7)
    xinode1=xi(node1,1); xinode2=xi(node2,1); xinode3=xi(node3,1);
    yinode1=yi(node1,1); yinode2=yi(node2,1); yinode3=yi(node3,1);
    xi(node7,1)=(xinode1+xinode2+xinode3)/3;
    yi(node7,1)=(yinode1+yinode2+yinode3)/3;

   crnode1=crnodes(3*iii-2,2);crnode2=crnodes(3*iii-1,2);crnode3=crnodes(3*iii,2);
   if((crnode1+crnode2+crnode3)<=1)
      xi(node4,1)=(xinode1+xinode2)/2;
      yi(node4,1)=(yinode1+yinode2)/2;
      xi(node5,1)=(xinode2+xinode3)/2;
      yi(node5,1)=(yinode2+yinode3)/2;
      xi(node6,1)=(xinode3+xinode1)/2;
      yi(node6,1)=(yinode3+yinode1)/2;
   end
    %UU1=UU(node1,1);UU2=UU(node2,1);UU3=UU(node3,1);uu=(UU1+UU2+UU3)/3;
    %VV1=VV(node1,1);VV2=UU(node2,1);VV3=UU(node3,1);vv=(VV1+VV2+VV3)/3;
    %xi(mm,1)=double(a00+a10*uu+a01*vv+a11*uu*vv);
    %yi(mm,1)=double(b00+b10*uu+b01*vv+b11*uu*vv);

 end %for iii

end% for itri
[nnn,dim]=size(xi)
[mmm,dim]=size(yi)
N=(1:nnode)'
[N xi yi]
%
coord(:,1)=(xi(:,1));
coord(:,2)=(yi(:,1));
gcoord(:,1)=double(xi(:,1));
gcoord(:,2)=double(yi(:,1));
%disp(gcoord)
 rrr
   edgen2n3
```

program(4)

```
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals(n)
%function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n3,nmax,numtri,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')
global edgen2n3
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2,1)=3;
%disp('vertex nodes of triangle')
%base edge
kk=3;
for k=2:n
   kk=kk+1;
   elm(k,1)=kk;
end
%disp('left edge nodes')
nni=1;
for i=0:(n-2)
   nni=nni+(n-i)+1;
   elm(nni,1)=3*n-i;
```

```
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-2)
   nni=nni+(n-i);
   elm(nni,1)=(n+3)+i;
end

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

jj=0;kk=0;
for j=0:n-1
   jj=j+1;
for k=1:(n+1)-j
   kk=kk+1;
   row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=3;
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;
%rr=row_nodes;
%rr;
rrr(:,:,1)=rr;
%nodes along edge:n2-n3
edgen2n3(1:n+1,1)=zeros(n+1,1);
for i=1:(n+1)
      edgen2n3(i,1)=row_nodes(i,(n+1)-i+1);
   end


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

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

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

nnd=(n+1)*(n+2)/2;
for kkk=1:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
end
%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)];
   %      case 2
   %       spqd(mm,1:4)=[eln(iel,7) eln(iel,4) eln(iel,2) eln(iel,5)];
   %      case 3
   %       spqd(mm,1:4)=[eln(iel,7) eln(iel,5) eln(iel,3) eln(iel,6)];
   %    end
   %  end
   %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(5)

```
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrial(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
%syms mst_tri x
global edgen2n3
global TRINUM ncrack nitri
ne=0;
nitri=nmax-1;
for itri=1:nitri
   TRINUM=TRINUM+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);
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)])
```

APPLICATION SPECIFIC PROGRAMS

PROGRAM-1a

```
function[mst_tri,theta,coord]=masterelementnodescoordinates_circulardisk(nelm)
%COMPUTES NODAL COORDINATES FOR THE CURVED TRIANGLE
%IN THE LUNAR MODEL
%CURVED TRIANGLE IS A CRCULAR SECTOR OF  ANGLE (2*pi/nelm)
%circular arcs is described by (x-1/2)^2+(y-1/2)^2=1/4
if (rem(nelm,4)==0)
n=nelm;N=n;
for i=1:n
   N=N+2;
   mst_tri(i,1)=n+1;
   mst_tri(i,2)=i;
   mst_tri(i,3)=i+1;
   mst_tri(i,4)=N;
   mst_tri(i,5)=N+1;
end
mst_tri(n,3)=1;
%mst_tri
syms pi theta
N=n/2;
for i=1:n
theta(i,1)=(i-1)*pi/N;
theta(i,2)=i*pi/N;
end

Theta=theta;
theta=double(theta);
syms coord
N=n;
%coord(n+1,1)=1/2;coord(n+1,2)=1/2;
for i=1:n
 t1=theta(i,1);
 t2=theta(i,2);
   coord(i,1)=.5+.5*cos(t1);
   coord(i,2)=.5+.5*sin(t1);
   coord(i+1,1)=.5+.5*cos(t2);
   coord(i+1,2)=.5+.5*sin(t2);
end

coord(n+1,1)=1/2;coord(n+1,2)=1/2;
```

```
%coord
coord=double(coord)
end%nelm>3
 %syms pi theta

 if nelm==1
    mst_tri(1,1)=3;
    mst_tri(1,2)=1;
    mst_tri(1,3)=2;
    mst_tri(1,4)=4;
    mst_tri(1,5)=5;



 theta(1,1)=0;
theta(1,2)=double(pi/2);%for one quadrant
%theta(1,2)=double(pi);%for semicircle
i=1
 t1=theta(i,1);
 t2=theta(i,2);
    coord(i,1)=.5+.5*cos(t1);
    coord(i,2)=.5+.5*sin(t1);
    coord(i+1,1)=.5+.5*cos(t2);
    coord(i+1,2)=.5+.5*sin(t2);
coord(3,1)=1/2;coord(3,2)=1/2;

%coord
coord=double(coord)

end%nelm=1
```

Program-1b

```
function[coord]=globalnodalcoordinate_crculardisk(mst_tri,theta,coord)
%computes intermediate points (x4,y4) and (x5,y5)of the curved triangle
%uses the input :three nodal coordinates of the curved triangle
%(x1,y1),(x2,y2),(x3,y3)
%checks for the location of circular sector and decides value for arc
%arc=1,2,3,4 for first,second,third and fourth quadrants respectively
[nel,nnel]=size(mst_tri);
for iel=1:nel
    node3=mst_tri(iel,1);
    node1=mst_tri(iel,2);
    node2=mst_tri(iel,3);
    x1=coord(node1,1);
    x2=coord(node2,1);
    x3=coord(node3,1);
    y1=coord(node1,2);
    y2=coord(node2,2);
    y3=coord(node3,2);

    t1=theta(iel,1);
    t2=theta(iel,2);
    if ((t1>=0)&(t2<=(pi/2)))arc=1;end
    if ((t1>=(pi/2))&(t2<=(pi)))arc=2;end
    if ((t1>=pi)&(t2<=3*(pi/2)))arc=3;end
    if ((t1>=3*(pi/2))&(t2<=(2*pi)))arc=4;end
switch arc
case 1
a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
  if (x4(j,1)>0.5)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
 end%j
case 2

a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
    if (x4(j,1)>0)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

case 3
  %a=(x1-x2)/3;b=(y1-y2)/3;
 %Y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));
 %Y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));

%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
 %for j=1:2
 %if ((Y4(j,1)>0)&(Y4(j,1)-b)>0)
 %y4=Y4(j,1);y5=(Y4(j,1)-b);
 %if((y4<.5)&(y5<.5))
 %x4 =sqrt(1/4-y4^2);
  %x5=x4-a;
  %end
  %end
  %end
 %format long

  %if (x5>0)&(y5>0)

   % X1=x4;
   % X2=x5;
   % Y1=y4;
   % Y2=y5;
   %end

 a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);

 format long
  for j=1:nn
    if (x4(j,1)>0)&((y4(j,1)>0)&(y4(j,1)<0.5))
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j


case 4
 a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
```

```
 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);

 for j=1:nn
  if (x4(j,1)>0.5)&((y4(j,1)>0)&(y4(j,1)<0.5))

    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

end%switch

  node4=mst_tri(iel,4);
  node5=mst_tri(iel,5);
  coord(node4,1)=X1;
  coord(node5,1)=X2;
  coord(node4,2)=Y1;
  coord(node5,2)=Y2;

end%iel
%[coord]
```

Program-2a

```
function[mst_tri,theta,coord]=masterelementnodescoordinates(nelm)
%COMPUTES NODAL COORDINATES FOR THE CURVED TRIANGLE
%IN THE LUNAR MODEL
%CURVED TRIANGLE IS A CRCULAR SECTOR OF  ANGLE (2*pi/nelm)
%outer circular arcs are described by (x-1/2)^2+(y-1/2)^2=1/4
%inner circular arcs is described by x^2+y^2=1/4
n=nelm;N=n;
for i=1:n
    N=N+2;
    mst_tri(i,1)=n+1;
    mst_tri(i,2)=i;
    mst_tri(i,3)=i+1;
    mst_tri(i,4)=N;
    mst_tri(i,5)=N+1;
end
mst_tri(n,3)=1;
%mst_tri
syms pi theta
N=n/2;
for i=1:n
theta(i,1)=(i-1)*pi/N;
theta(i,2)=i*pi/N;
end
Theta=theta;
theta=double(theta);
syms coord
N=n;
%coord(n+1,1)=1/2;coord(n+1,2)=1/2;
for i=1:n
 t1=theta(i,1);
 t2=theta(i,2);
    if ((t1>=0)&(t2<=(pi/2)))arc=1;end
    if ((t1>=(pi/2))&(t2<=(pi)))arc=2;end
    if ((t1>=pi)&(t2<=3*(pi/2)))arc=3;end
    if ((t1>=3*(pi/2))&(t2<=(2*pi)))arc=4;end

 switch arc
 case {1,2,4}
      T1=Theta(i,1);
      T2=Theta(i,2);

    coord(i,1)=.5+.5*cos(T1);
    coord(i,2)=.5+.5*sin(T1);
    coord(i+1,1)=.5+.5*cos(T2);
    coord(i+1,2)=.5+.5*sin(T2);
```

```
  case 3
     T1=3*pi/2-Theta(i,1);
     T2=3*pi/2-Theta(i,2);
   coord(i,1)=.5*cos(T1);
   coord(i,2)=.5*sin(T1);
   coord(i+1,1)=.5*cos(T2);
   coord(i+1,2)=.5*sin(T2);
end

end
coord(n+1,1)=1/2;coord(n+1,2)=1/2;

%coord
coord=double(coord)
```

Program-2b

```
function[coord]=globalnodalcoordinate(mst_tri,theta,coord)
%computes intermediate points (x4,y4) and (x5,y5)of the curved triangle
%uses the input :three nodal coordinates of the curved triangle
%(x1,y1),(x2,y2),(x3,y3)
%checks for the location of circular sector and decides value for arc
%arc=1,2,3,4 for first,second,third and fourth quadrants respectively
[nel,nnel]=size(mst_tri);
for iel=1:nel
    node3=mst_tri(iel,1);
    node1=mst_tri(iel,2);
    node2=mst_tri(iel,3);
    x1=coord(node1,1);
    x2=coord(node2,1);
    x3=coord(node3,1);
    y1=coord(node1,2);
    y2=coord(node2,2);
    y3=coord(node3,2);

    t1=theta(iel,1);
    t2=theta(iel,2);
    if ((t1>=0)&(t2<=(pi/2)))arc=1;end
    if ((t1>=(pi/2))&(t2<=(pi)))arc=2;end
    if ((t1>=pi)&(t2<=3*(pi/2)))arc=3;end
    if ((t1>=3*(pi/2))&(t2<=(2*pi)))arc=4;end

switch arc
case 1
a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
  if (x4(j,1)>0.5)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
 end%j
case 2

a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
    if (x4(j,1)>0)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

case 3
  a=(x1-x2)/3;b=(y1-y2)/3;
 Y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));
 Y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));

%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
 for j=1:2
 if ((Y4(j,1)>0)&(Y4(j,1)-b)>0)
 y4=Y4(j,1);y5=(Y4(j,1)-b);
 if((y4<.5)&(y5<.5))
 x4 =sqrt(1/4-y4^2);
  x5=x4-a;
end
end
end
 format long

  if (x5>0)&(y5>0)

    X1=x4;
    X2=x5;
    Y1=y4;
    Y2=y5;
  end



case 4
 a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);

 for j=1:nn
  if (x4(j,1)>0.5)&((y4(j,1)>0)&(y4(j,1)<0.5))

    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j



end%switch

  node4=mst_tri(iel,4);
  node5=mst_tri(iel,5);
  coord(node4,1)=X1;
  coord(node5,1)=X2;
  coord(node4,2)=Y1;
  coord(node5,2)=Y2;

end%iel
%[coord]
```

PROGRAM-3a

```
function[mst_tri,theta,coord]=masterelementnodescoordinates_circulardisk(nelm)
%COMPUTES NODAL COORDINATES FOR THE CURVED TRIANGLE
%IN THE LUNAR MODEL
%CURVED TRIANGLE IS A CRCULAR SECTOR OF  ANGLE (2*pi/nelm)
%circular arcs is described by (x-1/2)^2+(y-1/2)^2=1/4
if (rem(nelm,4)==0)
n=nelm;N=n;
for i=1:n
   N=N+2;
   mst_tri(i,1)=n+1;
   mst_tri(i,2)=i;
   mst_tri(i,3)=i+1;
   mst_tri(i,4)=N;
   mst_tri(i,5)=N+1;
end
mst_tri(n,3)=1;
%mst_tri
syms pi theta
N=n/2;
for i=1:n
theta(i,1)=(i-1)*pi/N;
theta(i,2)=i*pi/N;
end


Theta=theta;
theta=double(theta);
syms coord
N=n;
%coord(n+1,1)=1/2;coord(n+1,2)=1/2;
for i=1:n
 t1=theta(i,1);
 t2=theta(i,2);
   coord(i,1)=.5+.5*cos(t1);
   coord(i,2)=.5+.5*sin(t1);
   coord(i+1,1)=.5+.5*cos(t2);
   coord(i+1,2)=.5+.5*sin(t2);
end

coord(n+1,1)=1/2;coord(n+1,2)=1/2;

%coord
coord=double(coord)
end%nelm>3
 %syms pi theta

 if nelm==1
   mst_tri(1,1)=3;
   mst_tri(1,2)=1;
   mst_tri(1,3)=2;
   mst_tri(1,4)=4;
   mst_tri(1,5)=5;



 theta(1,1)=0;
theta(1,2)=double(pi/2);%for one quadrant
%theta(1,2)=double(pi);%for semicircle
i=1
 t1=theta(i,1);
 t2=theta(i,2);
   coord(i,1)=.5+.5*cos(t1);
   coord(i,2)=.5+.5*sin(t1);
   coord(i+1,1)=.5+.5*cos(t2);
   coord(i+1,2)=.5+.5*sin(t2);
coord(3,1)=1/2;coord(3,2)=1/2;

%coord
coord=double(coord)

end%nelm=1
```

PROGRAM-3b

```
function[coord]=globalnodalcoordinate_crculardisk(mst_tri,theta,coord)
```

```
%computes intermediate points (x4,y4) and (x5,y5)of the curved triangle
%uses the input :three nodal coordinates of the curved triangle
%(x1,y1),(x2,y2),(x3,y3)
%checks for the location of circular sector and decides value for arc
%arc=1,2,3,4 for first,second,third and fourth quadrants respectively
[nel,nnel]=size(mst_tri);
for iel=1:nel
   node3=mst_tri(iel,1);
   node1=mst_tri(iel,2);
   node2=mst_tri(iel,3);
   x1=coord(node1,1);
   x2=coord(node2,1);
   x3=coord(node3,1);
   y1=coord(node1,2);
   y2=coord(node2,2);
   y3=coord(node3,2);

   t1=theta(iel,1);
   t2=theta(iel,2);
   if ((t1>=0)&(t2<=(pi/2)))arc=1;end
   if ((t1>=(pi/2))&(t2<=(pi)))arc=2;end
   if ((t1>=pi)&(t2<=3*(pi/2)))arc=3;end
   if ((t1>=3*(pi/2))&(t2<=(2*pi)))arc=4;end
switch arc
case 1
a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
  if (x4(j,1)>0.5)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
 end%j
case 2

a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
    if (x4(j,1)>0)&(y4(j,1)>0.5)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

case 3
  %a=(x1-x2)/3;b=(y1-y2)/3;
 %Y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));
 %Y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2)));

%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
%[ 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2-a^6+a^4+a^2*b^2)^(1/2))]
 %for j=1:2
 %if ((Y4(j,1)>0)&(Y4(j,1)-b)>0)
 %y4=Y4(j,1);y5=(Y4(j,1)-b);
 %if((y4<.5)&(y5<.5))
 %x4 =sqrt(1/4-y4^2);
  %x5=x4-a;
  %end
  %end
  %end
 %format long

  %if (x5>0)&(y5>0)

   % X1=x4;
   % X2=x5;
   % Y1=y4;
   % Y2=y5;
   %end

 a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);

 format long
  for j=1:nn
    if (x4(j,1)>0)&((y4(j,1)>0)&(y4(j,1)<0.5))
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j


case 4
 a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));

 x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);

 for j=1:nn
  if (x4(j,1)>0.5)&((y4(j,1)>0)&(y4(j,1)<0.5))

    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

end%switch

  node4=mst_tri(iel,4);
  node5=mst_tri(iel,5);
  coord(node4,1)=X1;
  coord(node5,1)=X2;
  coord(node4,2)=Y1;
  coord(node5,2)=Y2;

end%iel
%[coord]
```

PROGRAM-4a

```
function[mst_tri,theta,coord]=masterelementnodescoordinates_circularshaftkeywayascrack(nelm)
%COMPUTES NODAL COORDINATES FOR THE CURVED TRIANGLE
%IN THE LUNAR MODEL
%CURVED TRIANGLE IS A CRCULAR SECTOR OF  ANGLE (2*pi/nelm)
%outer circular arcs are described by (x-1/2)^2+(y-1/2)^2=1/4
%inner circular arcs is described by x^2+y^2=1/4
nelm=5
n=nelm;N=n;
for i=1:n
```

```
    N=N+2;
    mst_tri(i,1)=n+1;
    mst_tri(i,2)=i;
    mst_tri(i,3)=i+1;
    mst_tri(i,4)=N;
    mst_tri(i,5)=N+1;
end
mst_tri(n,3)=1;
%mst_tri
syms pi theta

t=atan(.196/.46);
theta=[pi+t 3*pi/2;...%1
       3*pi/2 2*pi;...%2
       0    pi/2;...%3
       pi/2 pi-t;...%4
       pi-t pi+t];



Theta=theta;
theta=double(theta);
syms coord
N=n;
%coord(n+1,1)=1/2;coord(n+1,2)=1/2;
for i=1:n
 t1=theta(i,1);
 t2=theta(i,2);
    if ((t1>=(pi+t))&(t2<=3*pi/2))arc=1;end
    if ((t1>=3*pi/2)&(t2<=(2*pi)))arc=2;end
    if ((t1>=0)&(t2<=(pi/2)))arc=3;end
    if ((t1>=(pi/2))&(t2<=(pi-t)))arc=4;end
    if ((t1>=(pi-t))&(t2<=(pi+t)))arc=5;end

 switch arc

case 1
     coord(i,1)=0.04;
    coord(i,2)=-0.196;
    coord(i+1,1)=.5;
    coord(i+1,2)=-.5;
case 2
    coord(i,1)=0.5;
    coord(i,2)=-0.5;
    coord(i+1,1)=1;
    coord(i+1,2)=0;

  case 3

    coord(i,1)=1;
    coord(i,2)=0;
    coord(i+1,1)=.5;
    coord(i+1,2)=.5;
case 4
 coord(i,1)=.5;
    coord(i,2)=.5;
    coord(i+1,1)=.04;
    coord(i+1,2)=.196;
  case 5
    coord(i,1)=0.04;
    coord(i,2)=0.196;
    coord(i+1,1)=0.2;
    coord(i+1,2)=0;
 case 6
     coord(i,1)=0.2;
    coord(i,2)=0;
    coord(i+1,1)=0.04;
    coord(i+1,2)=-0.196;

end

end
coord(n+1,1)=1/2;coord(n+1,2)=0;

%coord
mst_tri
```

```
coord=double(coord);
nn=(1:n+1)';
[nn coord]
```

PROGRAM-4b

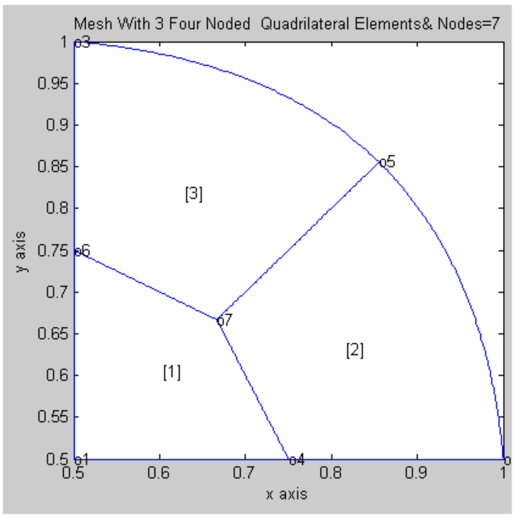
```
function[coord]=globalnodalcoordinate_circularshaft_keywayascrack(mst_tri,theta,coord)
%computes intermediate points (x4,y4) and (x5,y5)of the curved triangle
%uses the input :three nodal coordinates of the curved triangle
%(x1,y1),(x2,y2),(x3,y3)
%checks for the location of circular sector and decides value for arc
%arc=1,2,3,4 for first,second,third and fourth quadrants respectively
[nel,nnel]=size(mst_tri);
t=atan(.196/(.5-.04));
for iel=1:nel
   node3=mst_tri(iel,1);
   node1=mst_tri(iel,2);
   node2=mst_tri(iel,3);
   x1=coord(node1,1);
   x2=coord(node2,1);
   x3=coord(node3,1);
   y1=coord(node1,2);
   y2=coord(node2,2);
   y3=coord(node3,2);
   t1=theta(iel,1);
   t2=theta(iel,2);
    if((t1>=(pi+t))&(t2<=3*pi/2))arc=1;end
   if ((t1>=3*pi/2)&(t2<=2*pi))arc=2;end
   if ((t1>=0)&(t2<=(pi/2)))arc=3;end
   %if ((t1>=(pi/2))&(t2<=(pi)))arc=2;end
   if(t1>=pi/2)&(t2<=(pi-t))arc=4;end
   %if ((t1>=pi)&(t2<=3*(pi/2)))arc=3;end
   %if( (t1<=acos(.2))&(t2>=0)arc=3;end
   if ((t1>=(pi-t))&(t2<=(pi+t)))arc=5;end


switch arc
 case 1%third quadrant

a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
y4(2,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
x4=1/2*(a^2-2*y4*b+b^2+a)/a;
x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
    if ((x4(j,1)>0)&(x4(j,1)-a)>0)&(y4(j,1)<0)&((y4(j,1)-b)<0)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
end%j

 case 2 %fourth quadrant
a=(x1-x2)/3;b=(y1-y2)/3;
y4(1,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
y4(2,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
x4=1/2*(a^2-2*y4*b+b^2+a)/a;
x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);


  for j=1:nn
  if (x4(j,1)>0.5)&((x4(j,1)-a)>0.5)&(y4(j,1)<0)&((y4(j,1)-b)<0)
    X1=x4(j,1);
    X2=x4(j,1)-a;
    Y1=y4(j,1);
    Y2=y4(j,1)-b;
  end
 end%j
```

```
case 3
a=(x1-x2)/3;b=(y1-y2)/3;
%y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
%y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
%x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
y4(1,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
y4(2,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
x4=1/2*(a^2-2*y4*b+b^2+a)/a;
x5=x4-a;y5=y4-b;
format long
 nn=length(x4);
  for j=1:nn
  if (((x4(j,1)>=0.5)&(x4(j,1)<=1.0))&((y4(j,1)>=0)&(y4(j,1)<=0.5)))
    X1=x4(j,1);
    Y1=y4(j,1);
  end
   if (((x5(j,1)>=0.5)&(x5(j,1)<=1.0))&((y5(j,1)>=0)&(y5(j,1)<=0.5)))
    X2=x4(j,1)-a;
    Y2=y4(j,1)-b;
  end

 end%j
case 4

a=(x1-x2)/3;b=(y1-y2)/3;
%y4(1,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b+4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
%y4(2,1)=( 1/2/(4*a^2+4*b^2)*(4*b^3+4*a^2+4*b^2+4*a^2*b-4*(-b^4*a^2+a^4+a^2*b^2-2*a^4*b^2-a^6)^(1/2)));
%x4 =1/2*(b^2+a^2+a-2*y4*b+b)/a;
y4(1,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
y4(2,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
x4=1/2*(a^2-2*y4*b+b^2+a)/a;

  x5=x4-a;y5=y4-b;
 format long
 nn=length(x4);



  for j=1:nn
    if ((x4(j,1)>=0.04)&(x4(j,1)<=0.5))&((y4(j,1)>=0)&(y4(j,1)<=0.5))
    X1=x4(j,1);
    Y1=y4(j,1);
  end
  if ((x5(j,1)>=0.04)&(x5(j,1)<=0.5))&((y5(j,1)>=0)&(y5(j,1)<=0.5))
    X2=x4(j,1)-a;
    Y2=y4(j,1)-b;
  end


end%j

case 5 % KEYWAY AS CRACK
  a=(x1-x2)/3;b=(y1-y2)/3;

y4(1,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b+4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
y4(2,1)=( 1/2/(4*b^2+4*a^2)*(4*b^3+4*a^2*b-4*(-b^4*a^2-2*a^4*b^2+a^2*b^2-a^6+a^4)^(1/2)))
x4=1/2*(a^2-2*y4*b+b^2+a)/a;
 format long
 nn=length(x4);
  for j=1:nn
       YY=y4(j,1);YYY=YY-b;XX=x4(j,1);XXX=XX-a;
    if ((XX>=0)&(XX<=0.04))&((YY>=-0.196)&(YY<=0.196))
    X1=XX;
    Y1=YY;
  end
  if ((XXX>=0)&(XXX<=0.04))&((YYY>=-0.196)&(YYY<=0.196))
    X2=XXX;
    Y2=YYY;
  end


end%j
```

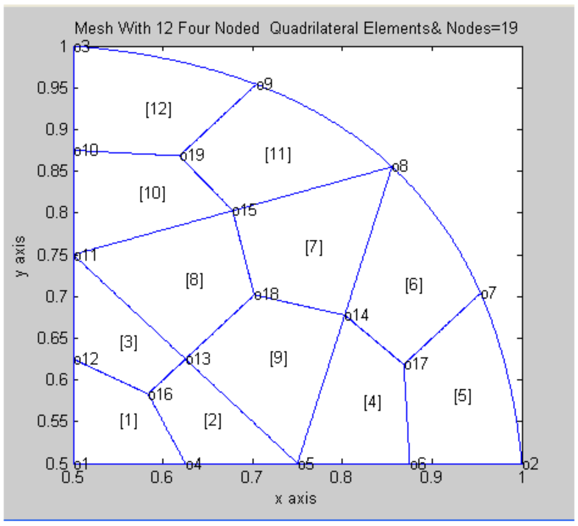
```
end%switch

  node4=mst_tri(iel,4);
  node5=mst_tri(iel,5);
  coord(node4,1)=X1;
  coord(node5,1)=X2;
  coord(node4,2)=Y1;
  coord(node5,2)=Y2;

end%iel
%[coord]

[num,dim]=size(coord);

NN=(1:num)';
[NN coord]
```

FIGURES

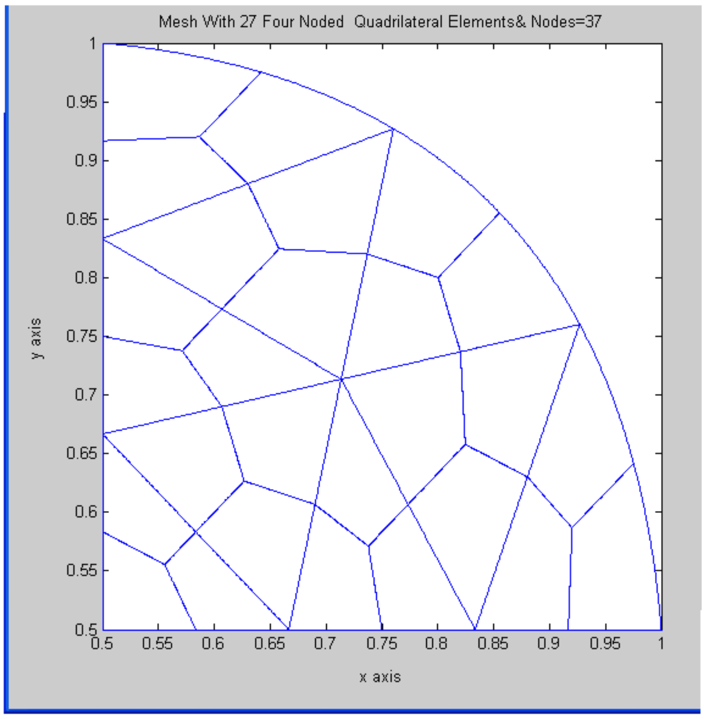
QUARTER CIRCLE AS CURVED TRIANGLE -01



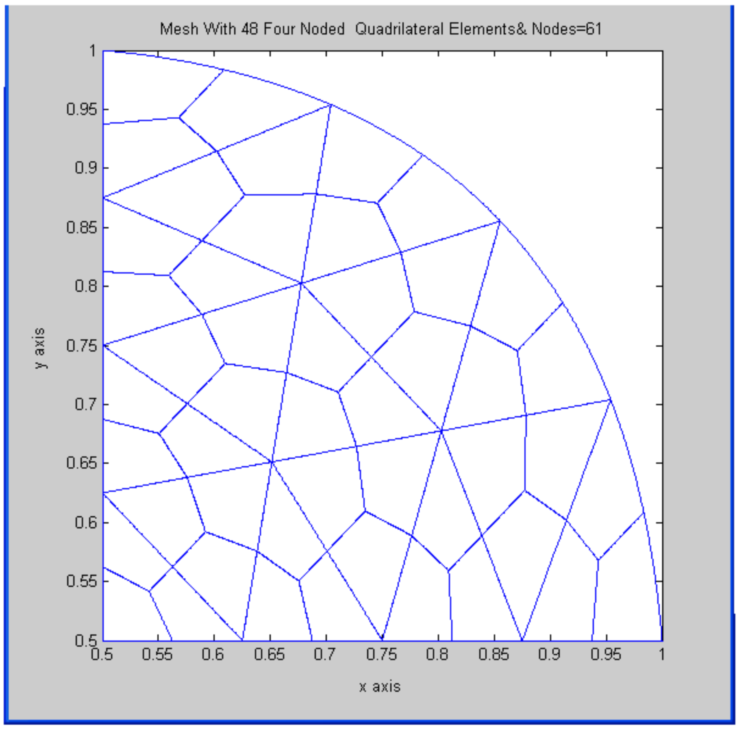
QUARTER CIRCLE AS ONE CURVED TRIANGLE-2



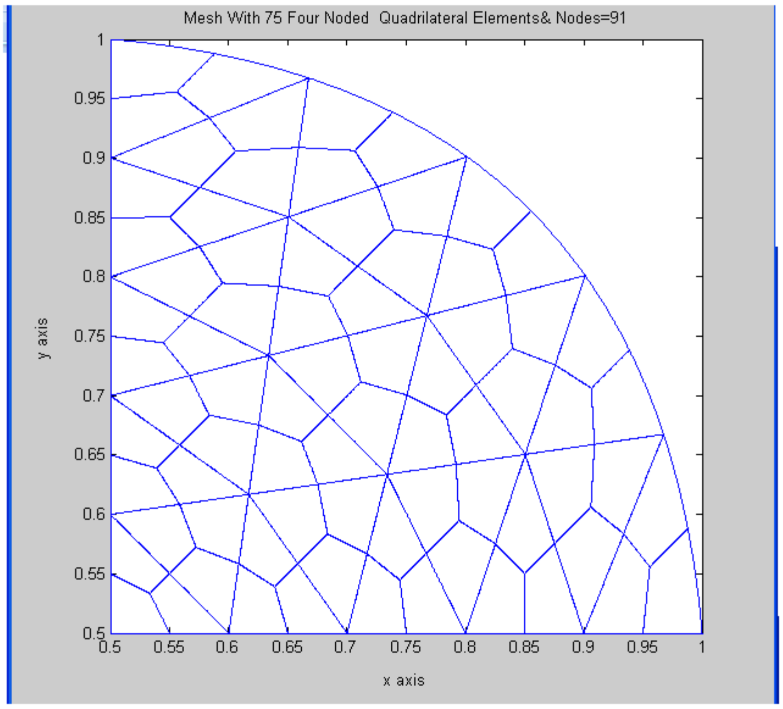
QUARTER CIRCLE AS A CURVED TRIANGLE-03



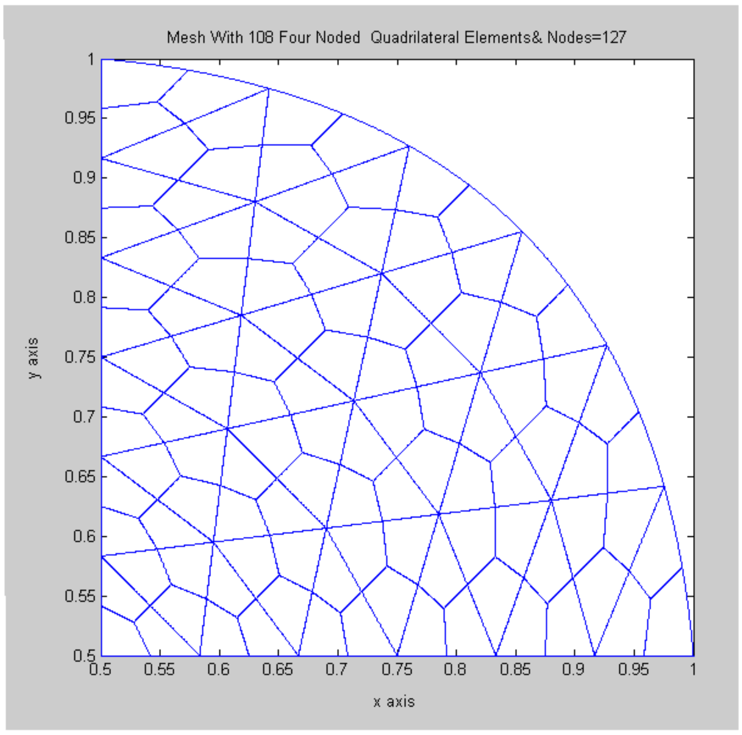
QUARTER CIRCLE AS CURVED TRIANGLE-04



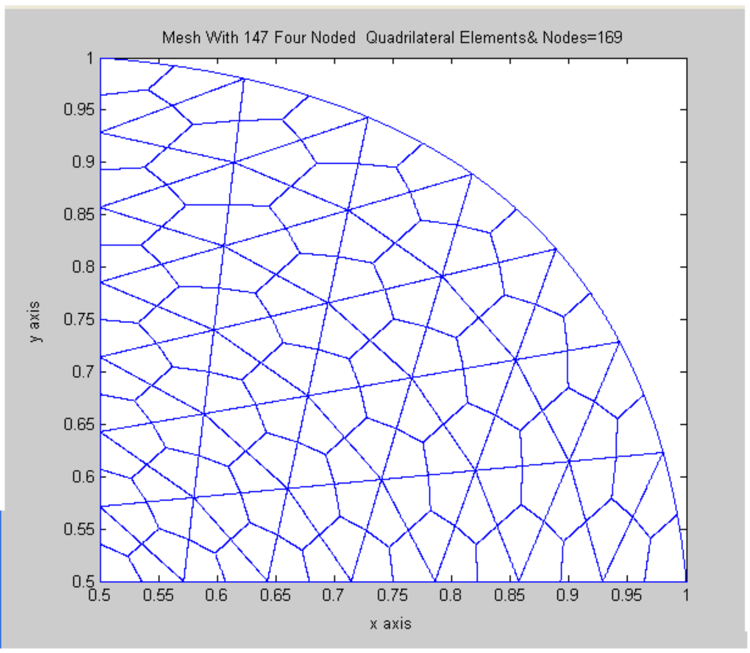
QUARTER CIRCLE AS CURVED TRIANGLE- 05



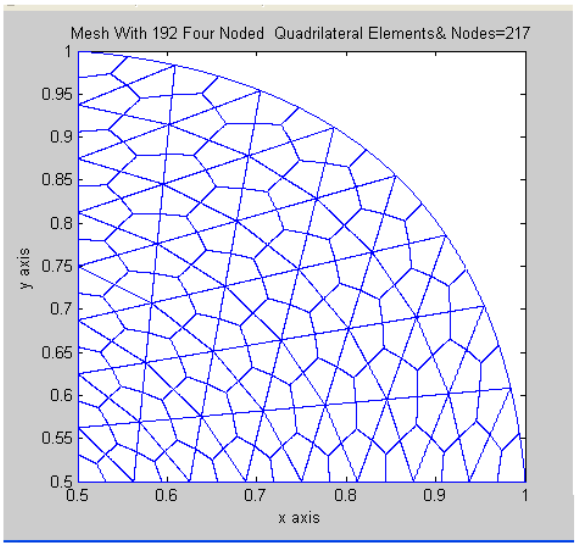
QUARTER CIRCLE AS ACURVED TRIANGLE-06



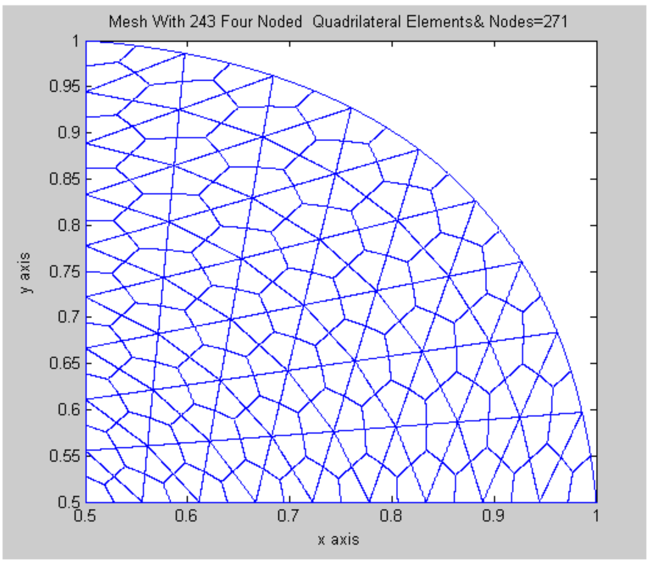
QUARTER CIRCLE AS A CURVED TRIANGLE- 07



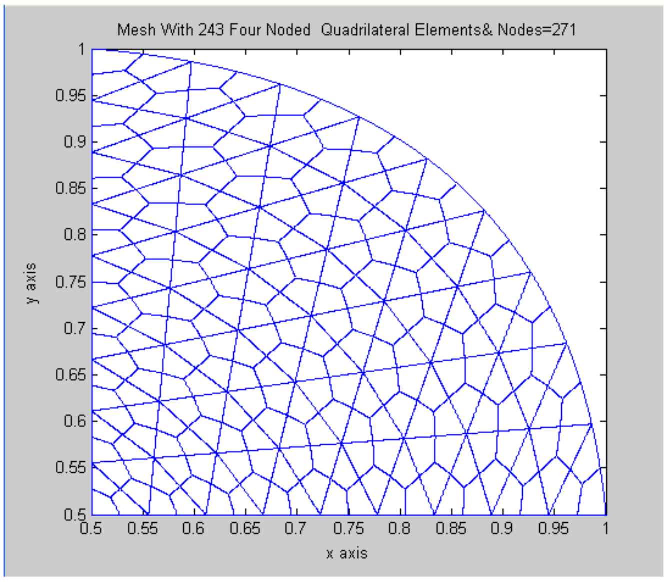
QUARTER CIRCLE AS A CURVED TRIANGLE-08



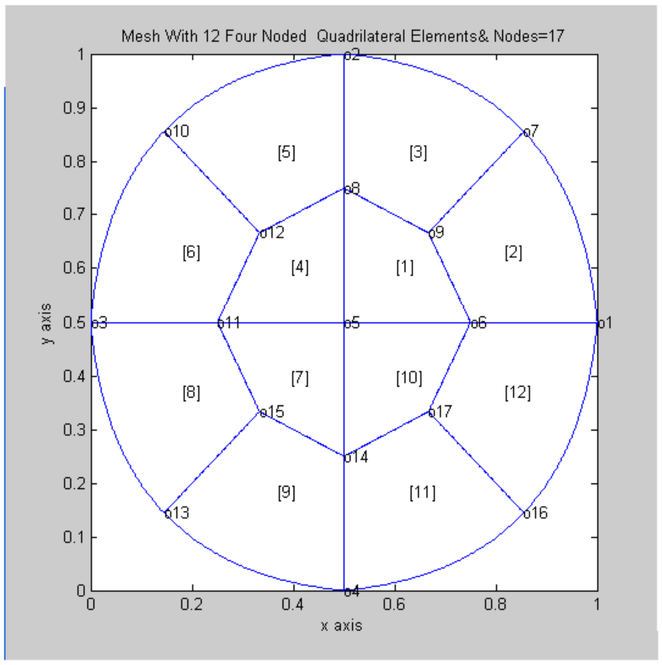
QUARTER CIRCLE AS A CURVED TRIANGLE - 09



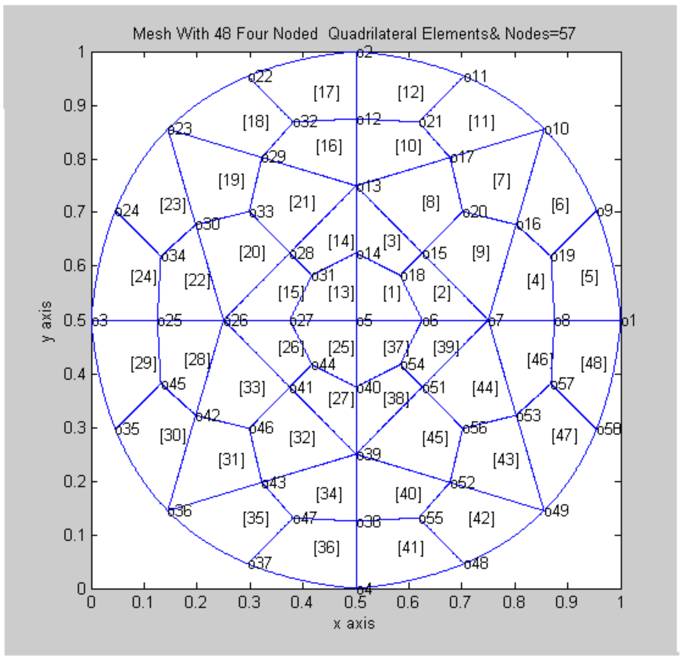
QUARTER CIRCLE AS A CURVED TRIANGLRE -10



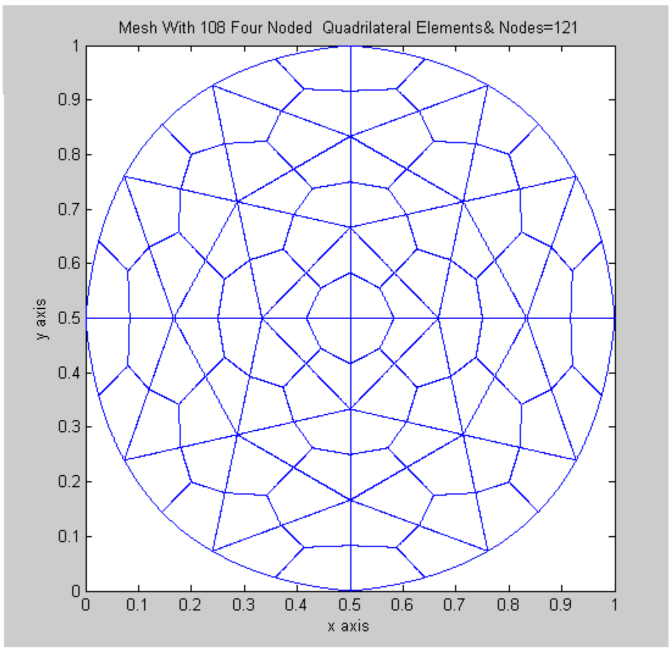
CIRCULAR DISK-NDIV=02



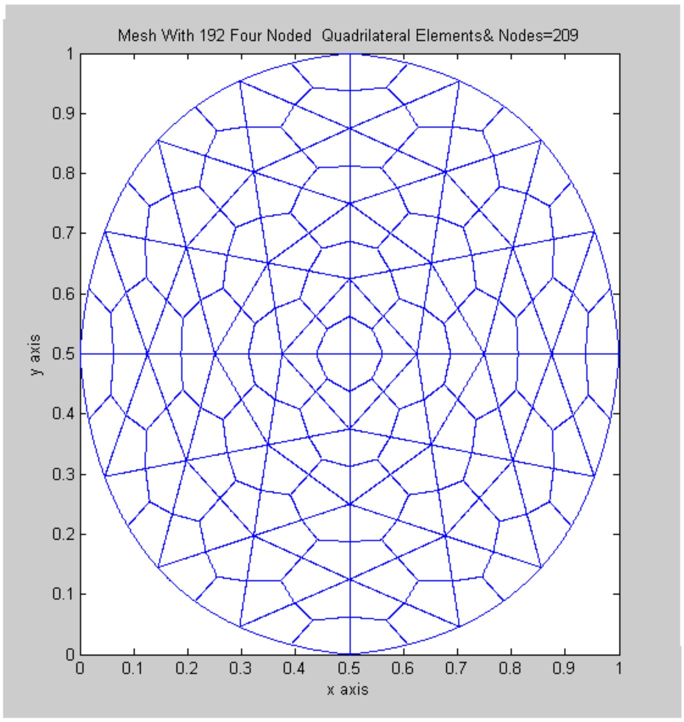
CIRCULAR DISK NDIV=04



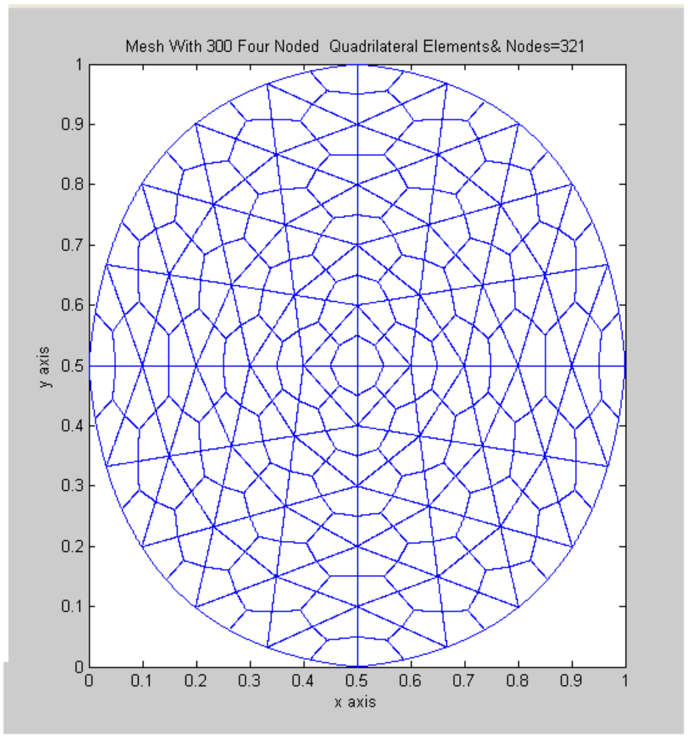
CIRCULAR DISK NDIV=06



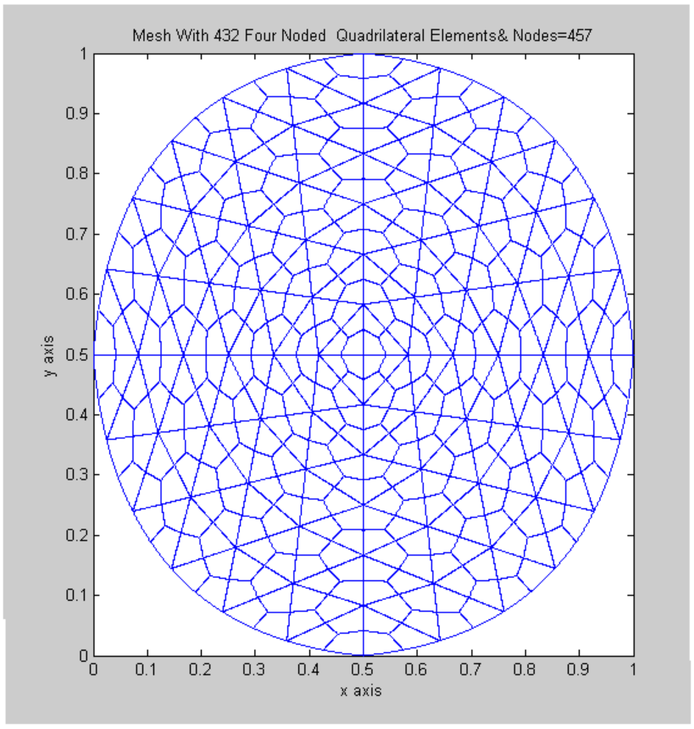
CIRCULAR DISK NDIV=08



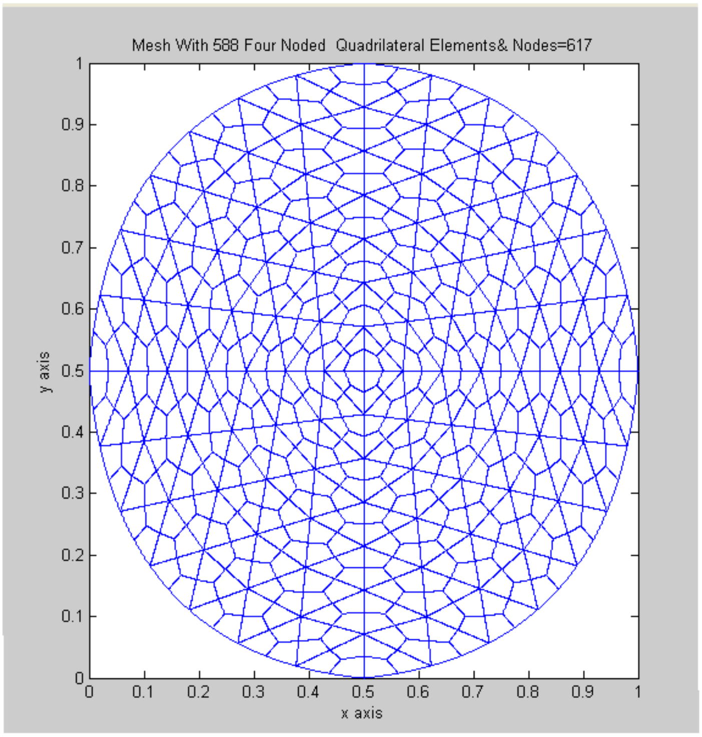
CIRCULAR DISK NDIV=10



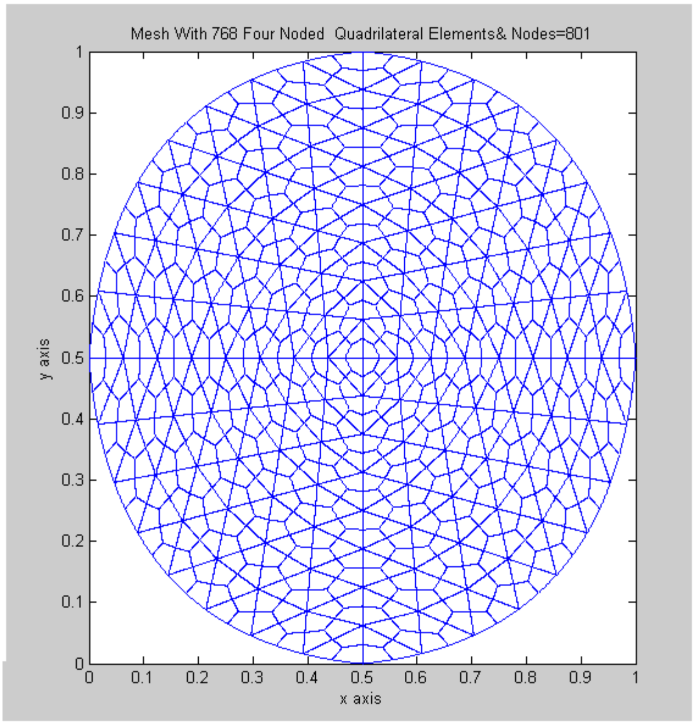
CIRCULAR DISK NDIV=12



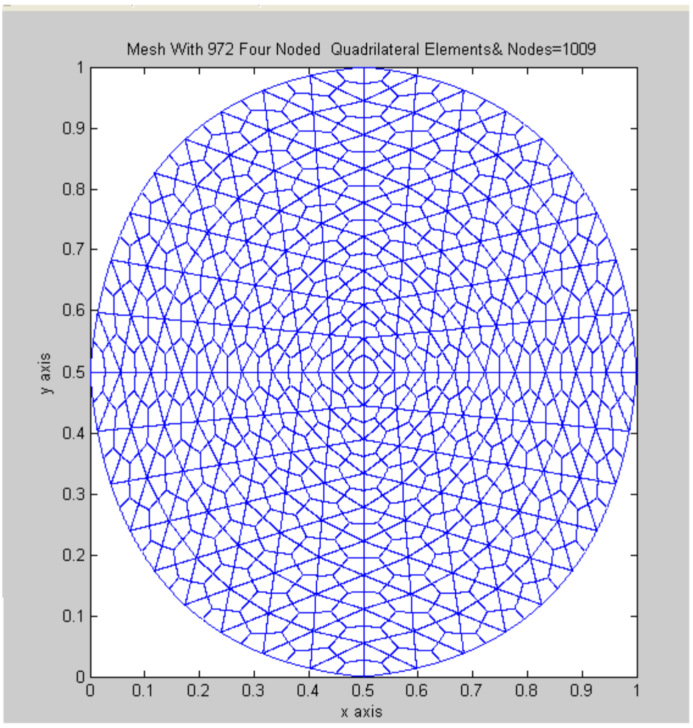
CIRCULAR DISK NDIV=14



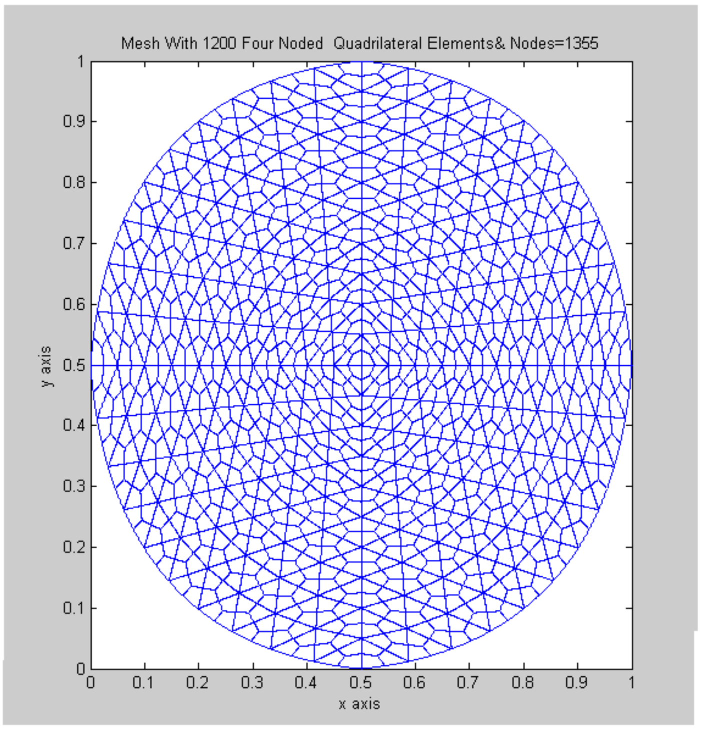
CIRCULAR DISK NDIV=16



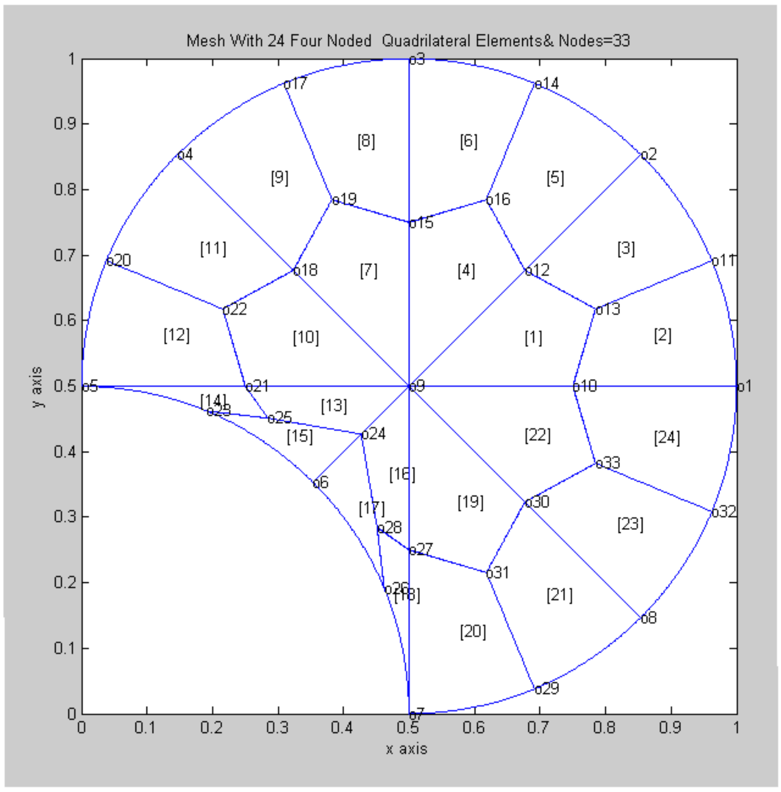
CIRCULAR DISK NDIV=18



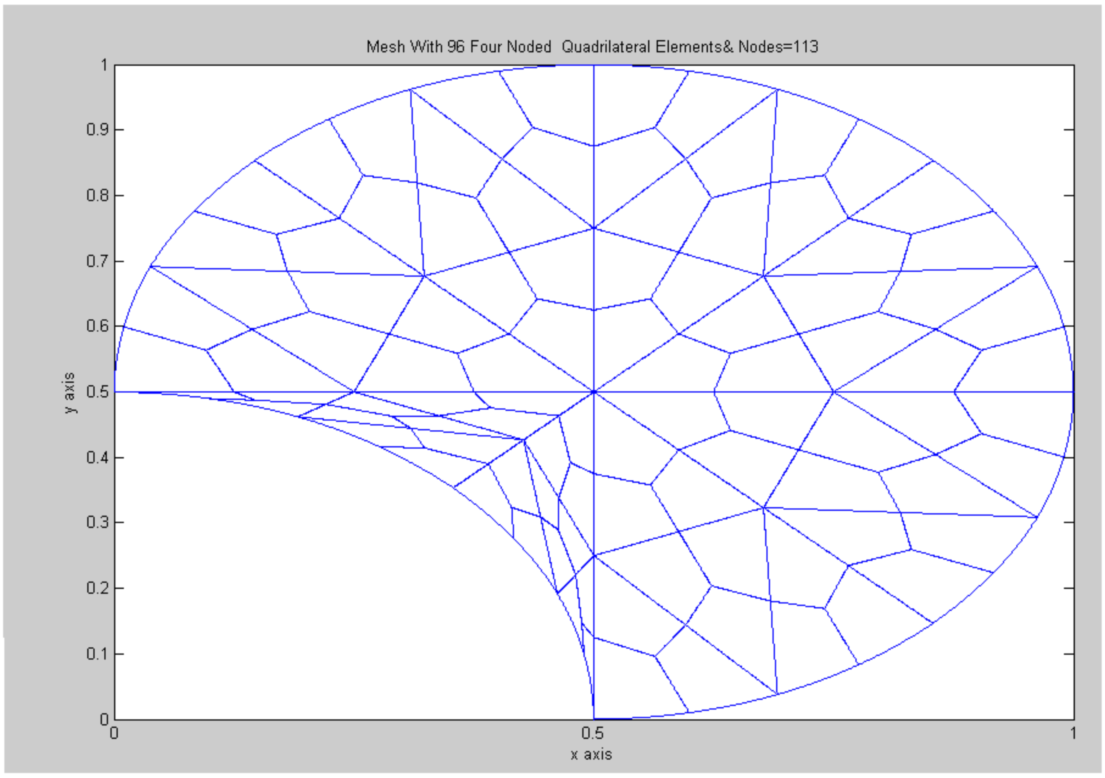
CIRCULAR DISK NDIV=20


Mesh With 1200 Four Noded Quadrilateral Elements& Nodes=1355
y axis
x axis

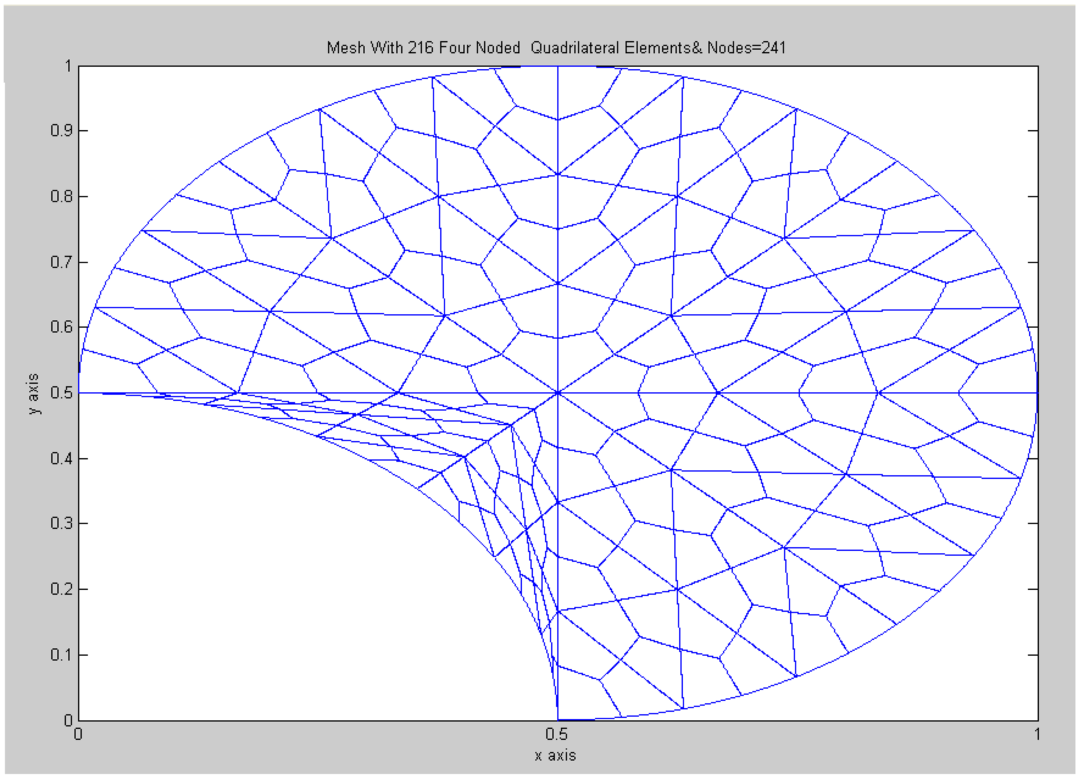
LUNAR FEM MODEL NDIV=02



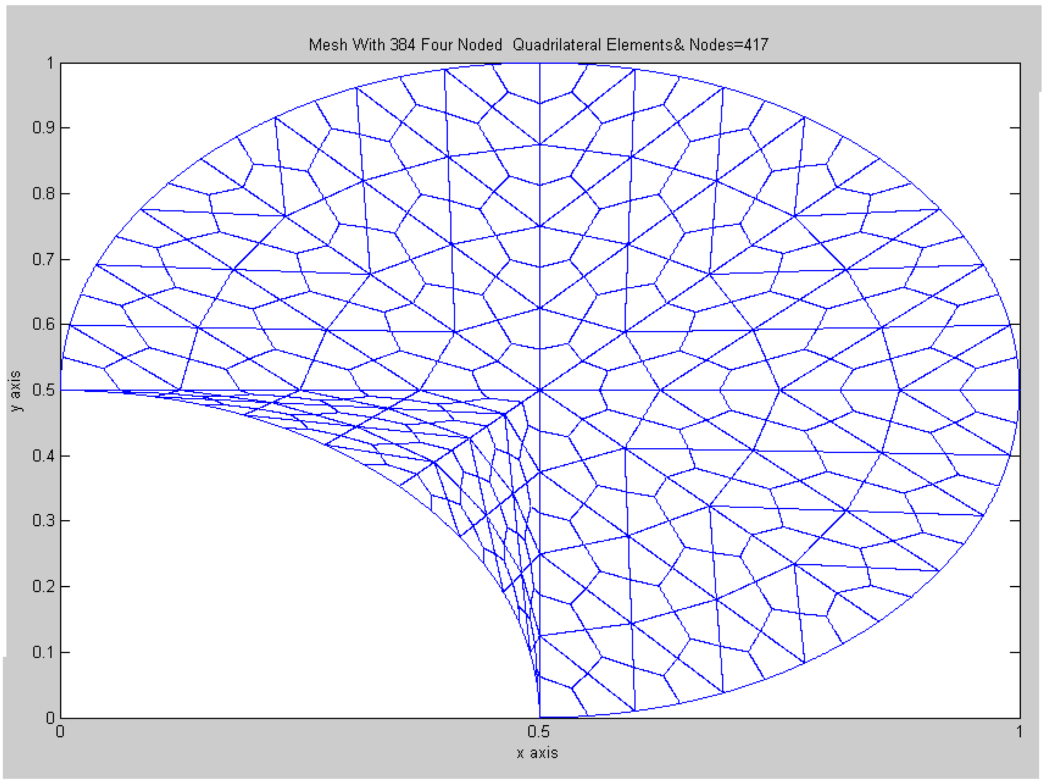
LUNAR MODEL FEM MESH NDIV=04



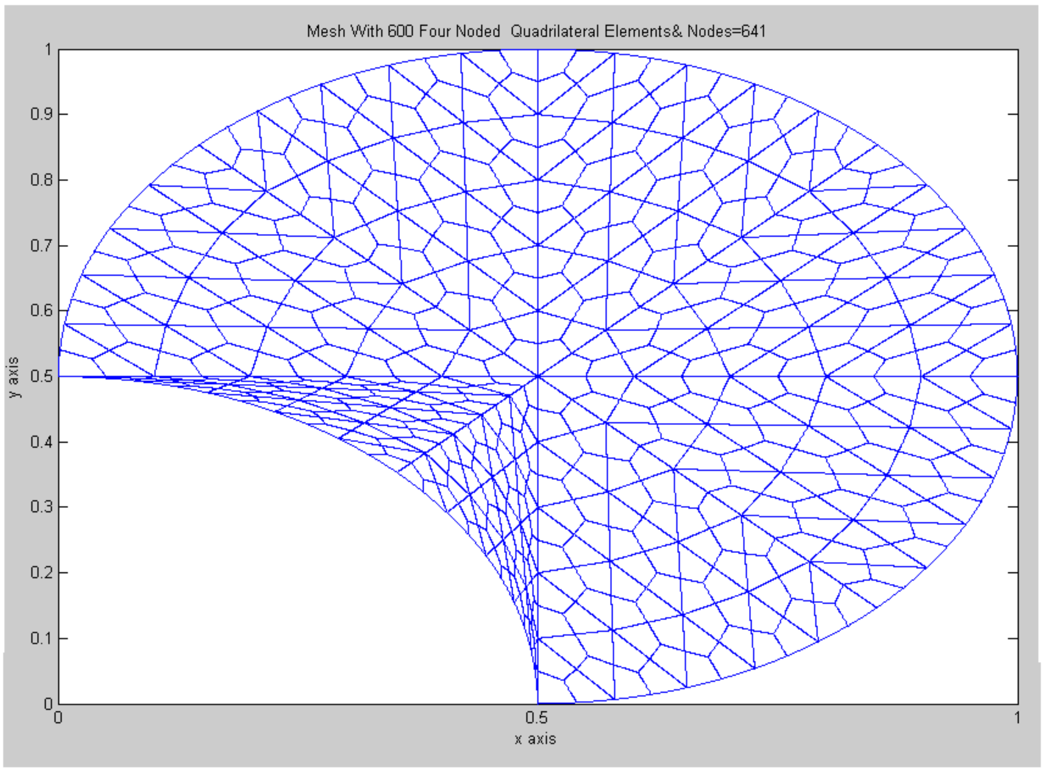
LUNAR MODEL FEM MESH NDIV=06



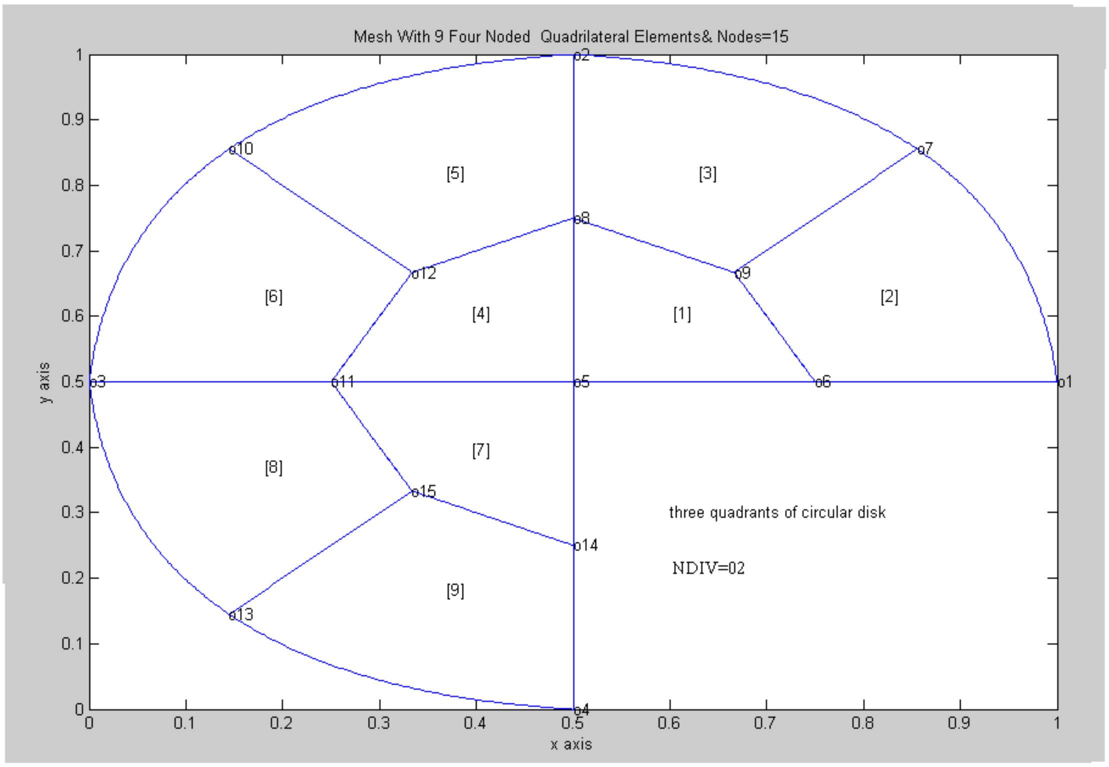
LUNAR MODEL FEM MESH NDIV= 08

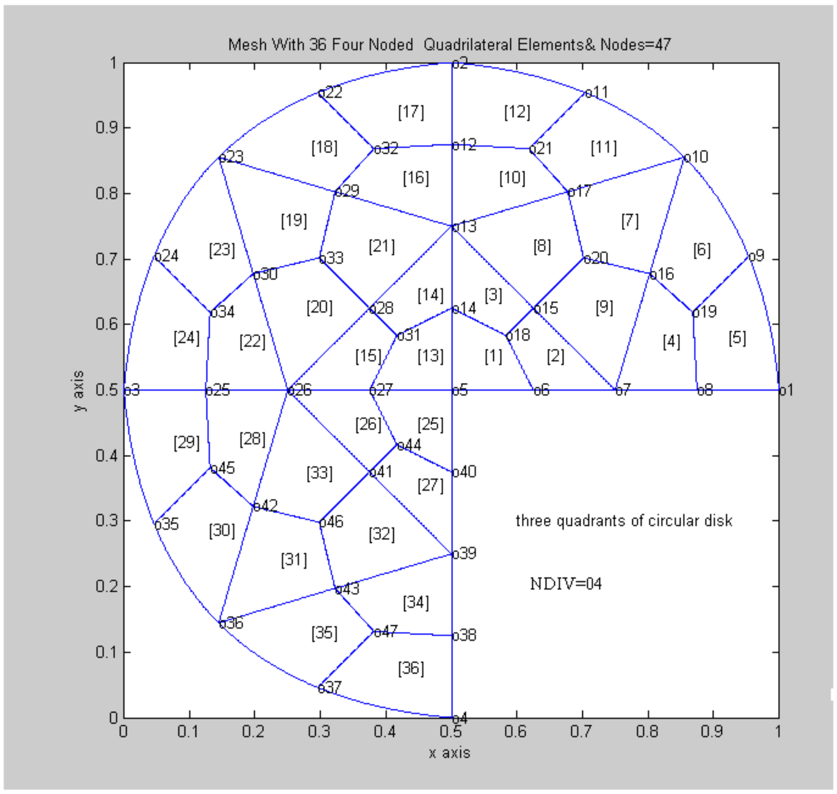


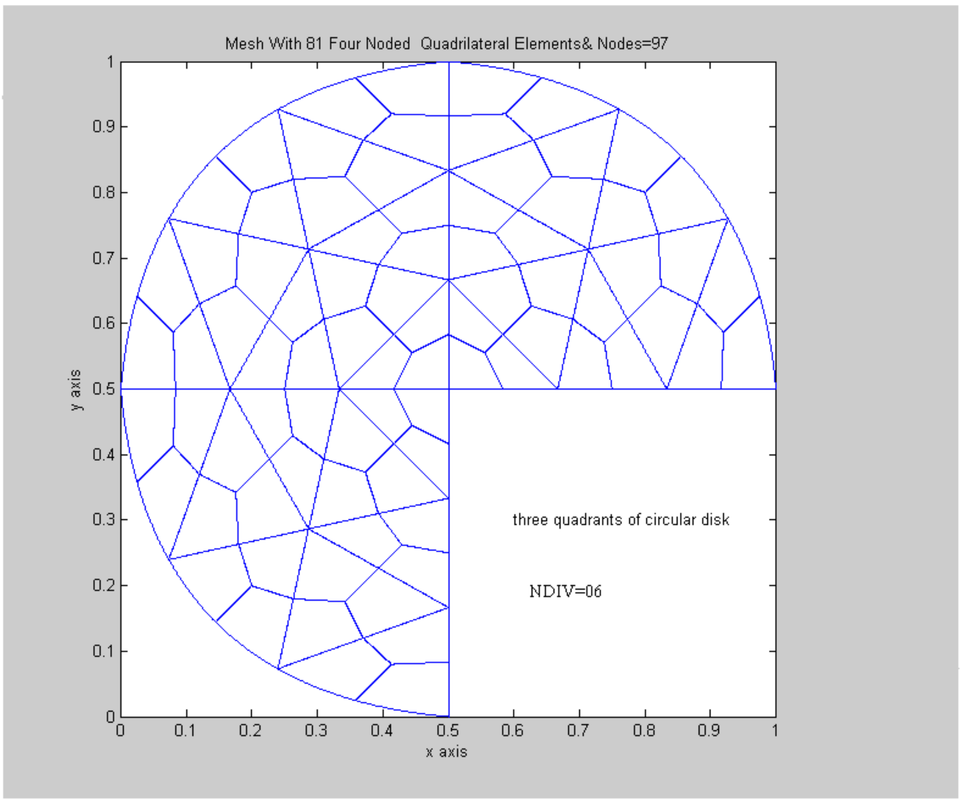
LUNAR MODEL FEM MESH NDIV=10

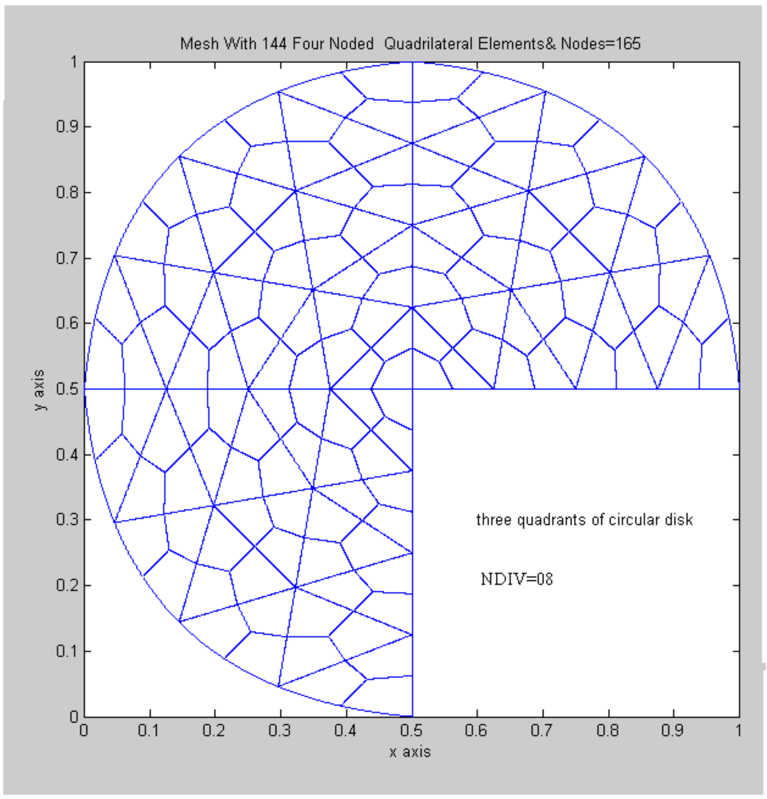


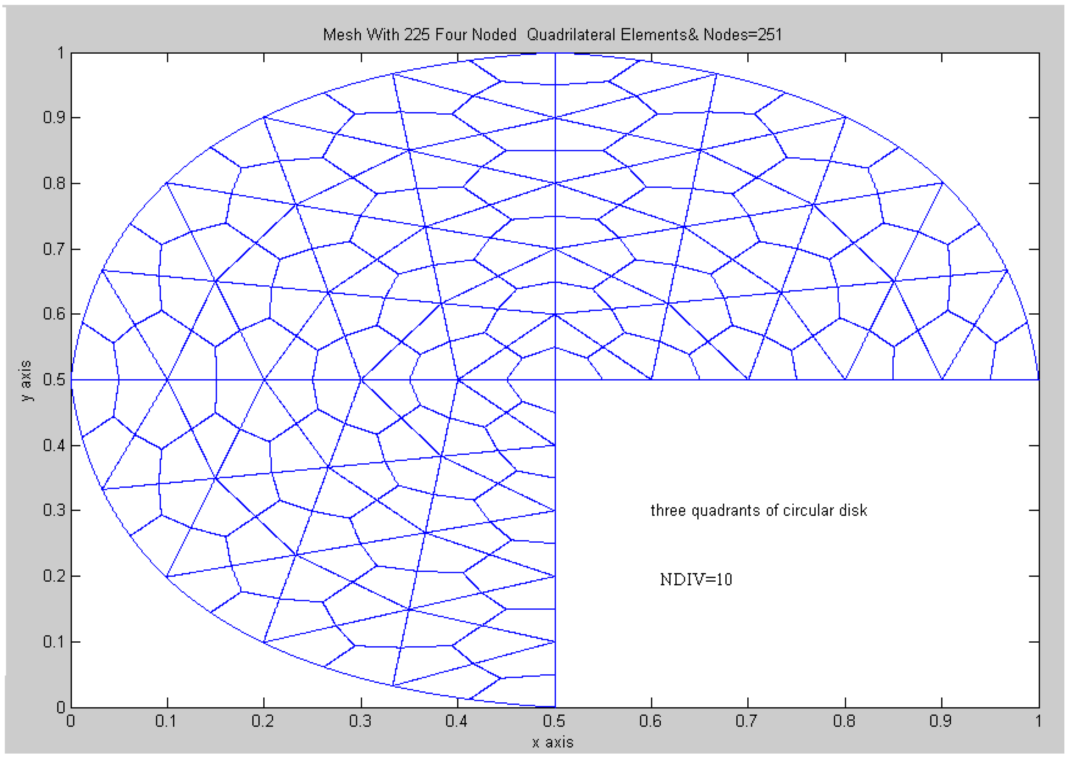
FIGURES ON CRACKED DISKS.....THREE QUADRANTS OF A CIRCULAR DISK


Mesh With 9 Four Noded Quadrilateral Elements& Nodes=15
three quadrants of circular disk
NDIV=02
x axis
y axis


Mesh With 36 Four Noded Quadrilateral Elements& Nodes=47
three quadrants of circular disk
NDIV=04
x axis
y axis

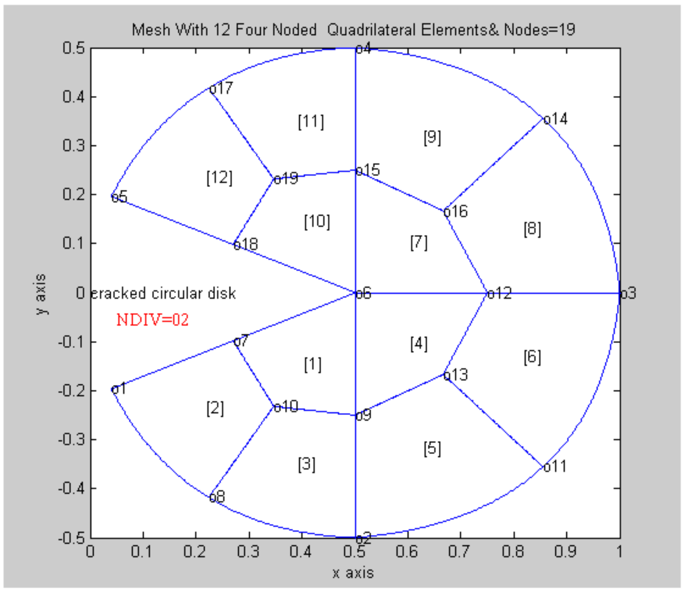

Mesh With 81 Four Noded Quadrilateral Elements& Nodes=97
three quadrants of circular disk
NDIV=06
x axis
y axis

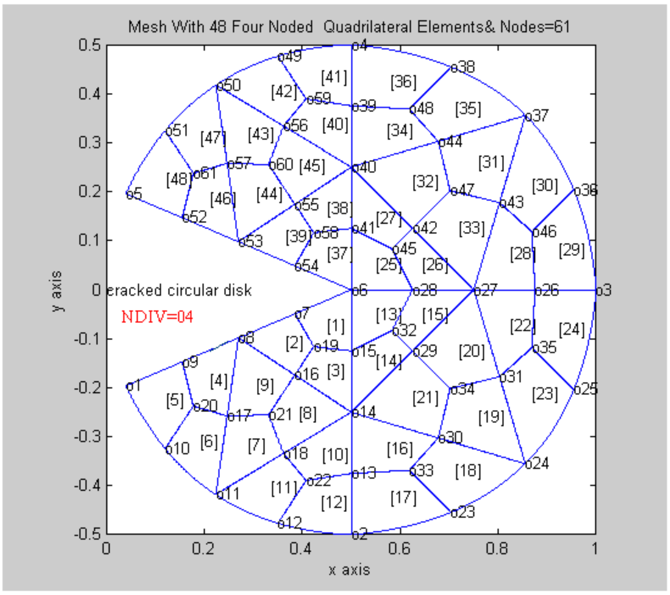

Mesh With 144 Four Noded Quadrilateral Elements& Nodes=165
three quadrants of circular disk
NDIV=08
x axis
y axis

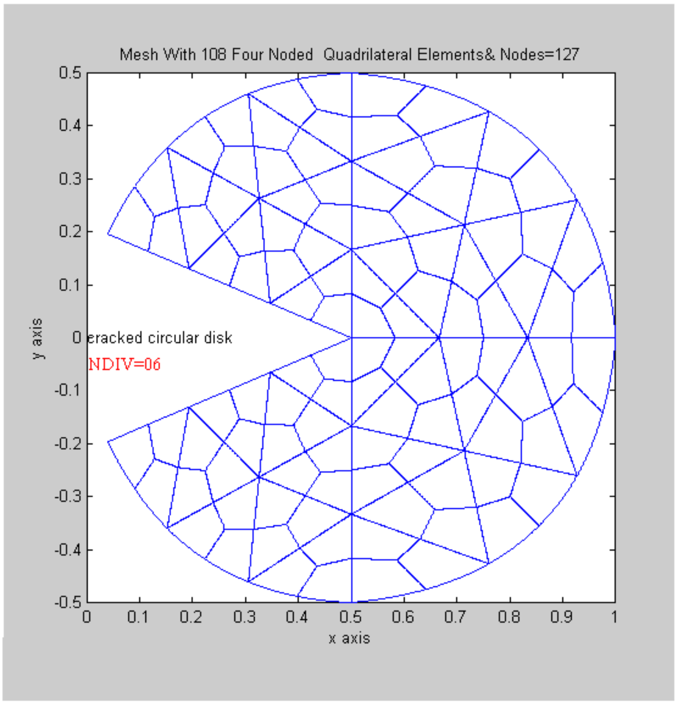


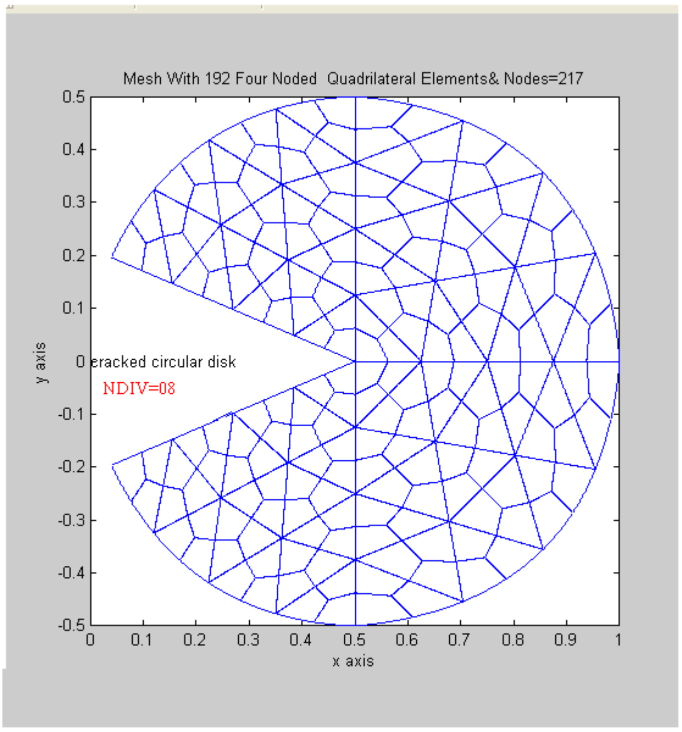
Mesh With 225 Four Noded Quadrilateral Elements& Nodes=251
three quadrants of circular disk
NDIV=10
x axis
y axis

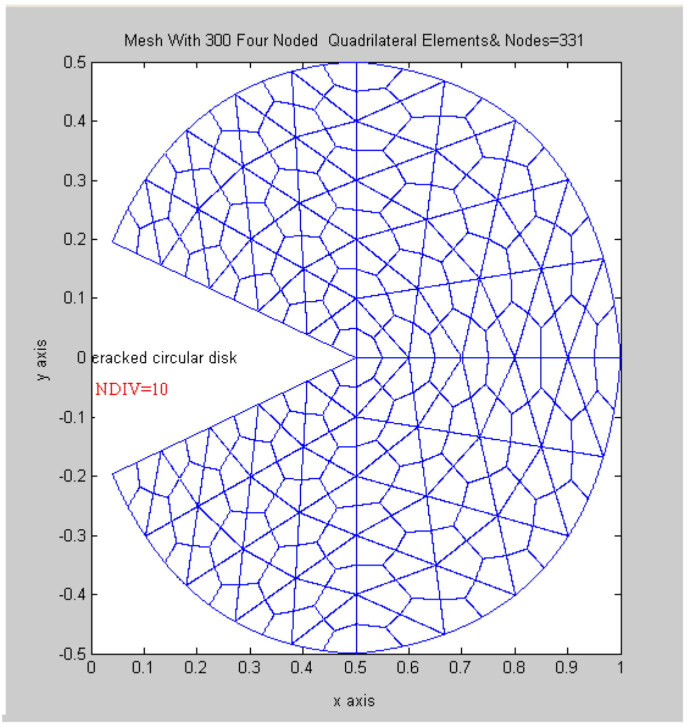
FIGURES ON CRACKED CIRCULAR DISK

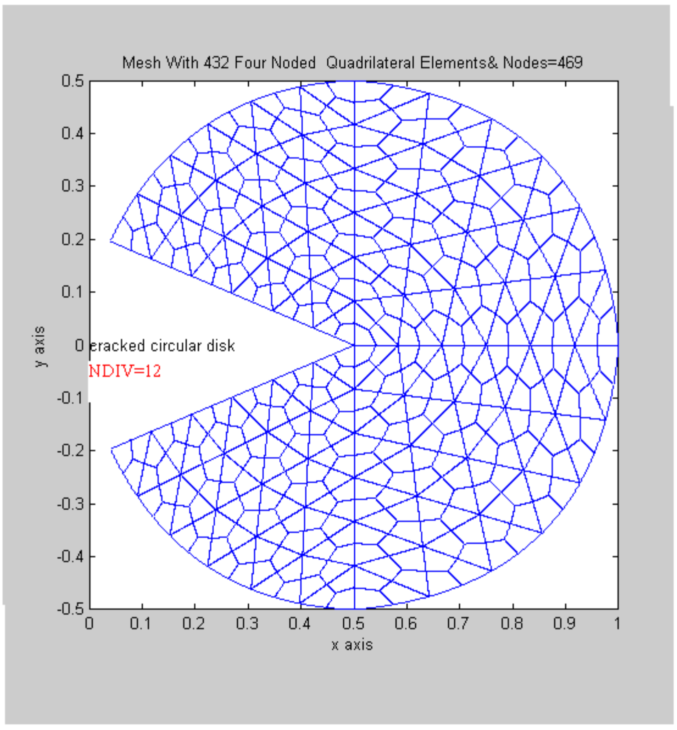


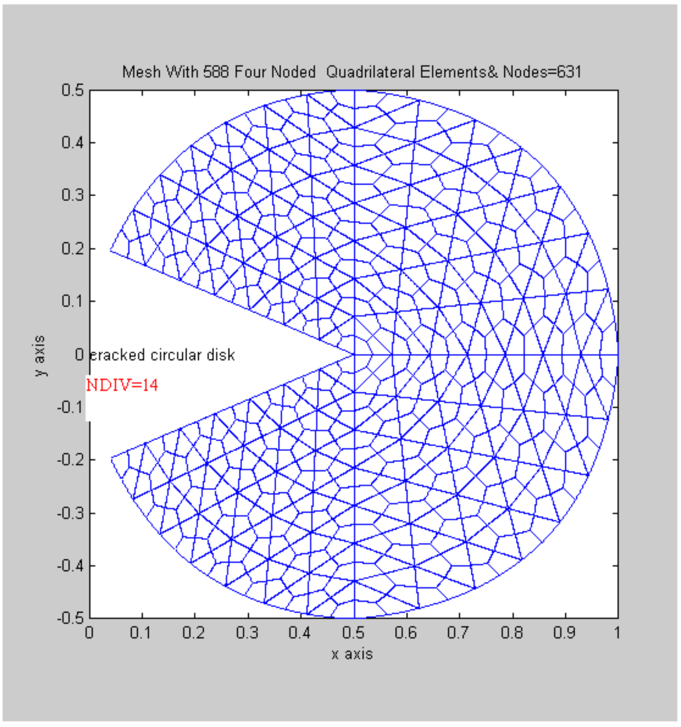

Mesh With 48 Four Noded Quadrilateral Elements& Nodes=61
cracked circular disk
NDIV=04
x axis
y axis

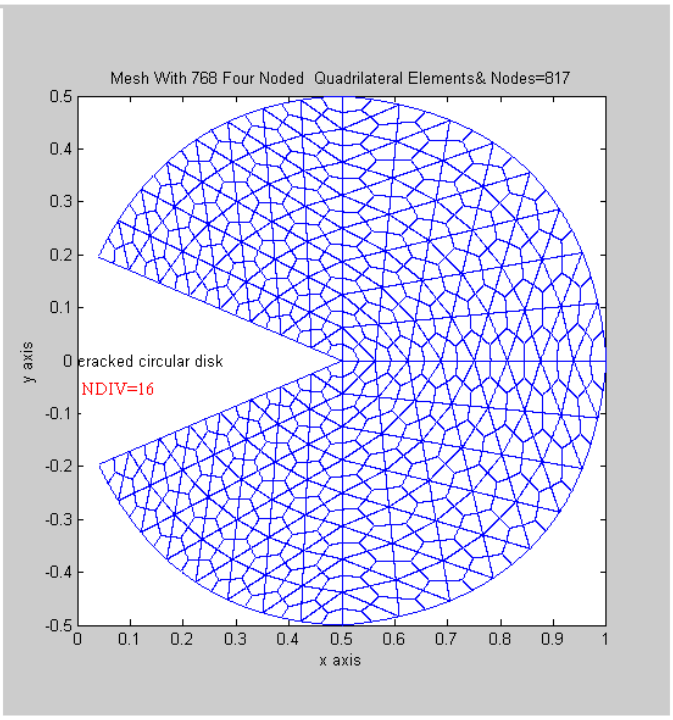

Mesh With 108 Four Noded Quadrilateral Elements& Nodes=127
cracked circular disk
NDIV=06
x axis
y axis

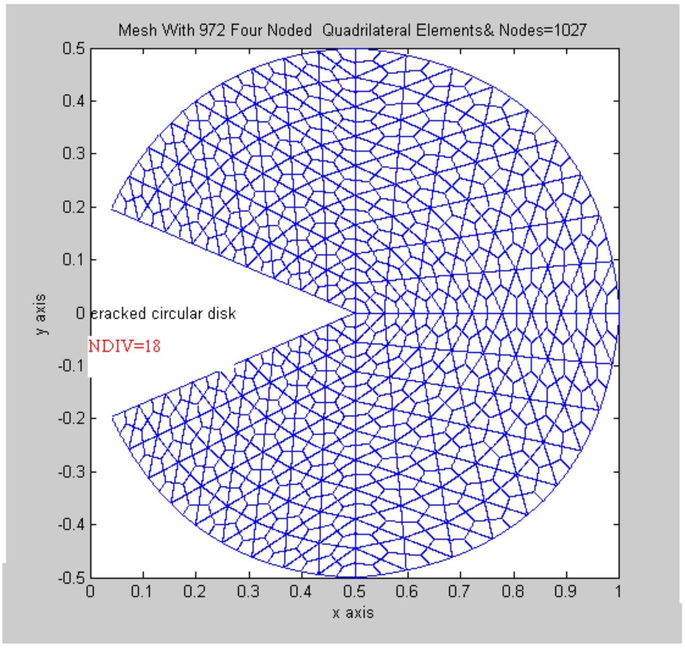

Mesh With 192 Four Noded Quadrilateral Elements& Nodes=217
cracked circular disk
NDIV=08
x axis
y axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

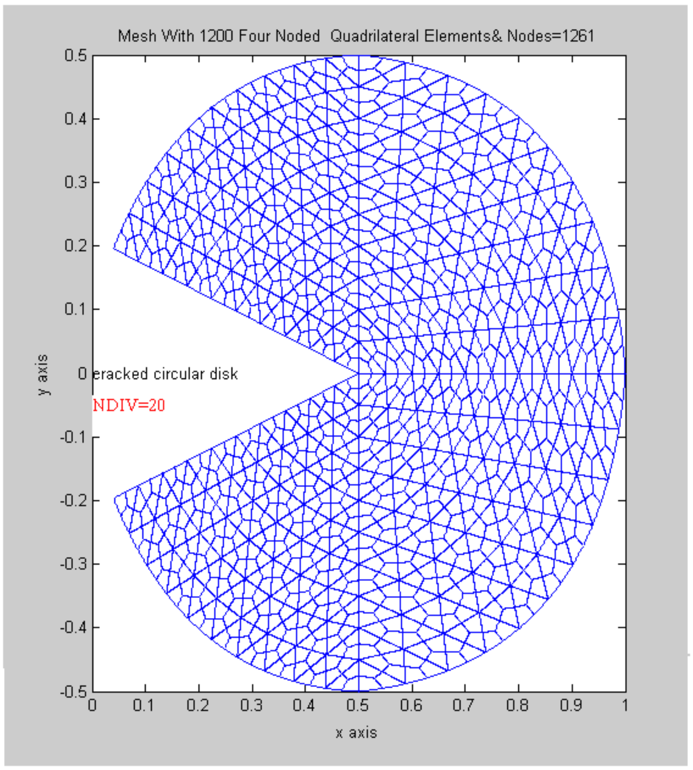

Mesh With 300 Four Noded Quadrilateral Elements& Nodes=331
cracked circular disk
NDIV=10
x axis
y axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Mesh With 432 Four Noded Quadrilateral Elements& Nodes=469
cracked circular disk
NDIV=12
y axis
x axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Mesh With 588 Four Noded Quadrilateral Elements& Nodes=631
cracked circular disk
NDIV=14
y axis
x axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Mesh With 768 Four Noded Quadrilateral Elements& Nodes=817
cracked circular disk
NDIV=16
y axis
x axis


Mesh With 972 Four Noded Quadrilateral Elements& Nodes=1027
cracked circular disk
NDIV=18
y axis
x axis


Mesh With 1200 Four Noded Quadrilateral Elements& Nodes=1261
cracked circular disk
NDIV=20
y axis
x axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1