www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 9 September 2018, Page No. 24214-24310
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i9.03

A New Approach to Automesh Generation of $all\ \beta$ graded triangular and quadrilateral Finite Elements over Analytical Surfaces by using the Parabolic Arcs passing through four points on the Boundary Curve

H.T. Rathod[a*] , Bharath Rathod[b]

[a] Department of Mathematics, Central College Campus, Bangalore University, Bangalore -560001, Karnataka state, India.
[b] Xavier Institute of Management and Entrepreneurship, Hosur Road, Electronic City Phase II, Bangalore-560034 , Karnataka state, India.

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. These simple regions are then triangulated to create 6-node triangles by inserting midside nodes to these triangles. Each isolated 6-node triangle is then split into four triangles according to the usual scheme, that is, by using straight lines to join the midside nodes. To preserve the mesh conformity a similar procedure is also applied to every triangle of the domain to fully discretize the given convex curved or cracked convex curved domains into all triangles, thus propagating$(0 < \beta < 1, \beta > 1)\beta\ graded$ refinements and $\beta = 1\ generates\ the\ classical\ uniform\ mesh$.The quadrangulation of β graded 3-node linear triangles is done by inserting three midside nodes and a centroidal node.Then each graded triangle is split into three quadrilaterals by using straight lines to join the centroid to the midside nodes. This simple method generates a high quality mesh whose elements confirm well to the requested shape by refining the problem domain.

We have approximated the curved boundary 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 triangles and quadrilaterals, thus propagating a uniform refinement. This simple method generates a high quality mesh whose elements confirm well to the requested shape by refining the problem domain. Examples of a **circular disk** and a **cracked circular disk** 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 triangular and quadrilateral mesh for application to finite element analysis.

Keywords: finite elements, triangular,quadrilateral mesh generation,analytical curved surfaces, curved triangular element, parabolic arcs,uniform refinement.

1. Introduction

The Finite Element Method (FEM) has been invented by engineers around 1950 for solving the partial differential equations arising in solid mechanics; the main idea was to use the principle of virtual work for designing discrete approximations of the boundary value problems. The most popular reference on FEM in solid mechanics is the book of Zienckienwicz [1]. Generalizations to other fields of physics or engineering have been done by applied mathematicians through the concept of variational formulations and weak forms of the partial differential equations .

The FEM discretizes the continuous domain of the problem by means of a series of simple geometric forms called finite elements, for which the governing relations on the entire continuous domain are valid on each element. Under this assumption, the approximate solution in the entire continuous domain of the problem can be obtained by means of trial functions also called the shape functions. The FEM transforms the differential equation into an algebraic system of equations which can then be solved easily by known numerical methods.

The term Finite Element Method (FEM) appeared first in 1960, but the ideas behind it are even older and can be traced back to the engineering sciences. Being a powerful numerical tool for a variety of engineering disciplines, the FEM quickly found its way into applied mathematics where stability, discretization errors, and convergence rates are thoroughly investigated. It has been a very active field of research ever since. The standard way to gain accuracy of the approximate solution is to refine the triangulation of the computational domain, which introduces more degrees of freedom. A vast amount of publications deals with this so called h-version. An alternative way of enlarging the number of unknowns is to increase the polynomial degree of finite elements. This, so called, p-version is less common and its convergence speed strongly depends on the regularity of the solution to the PDE.

The modelling and numerical simulation of complex systems play an important role in many industrial, medical and economical applications. Very often, such systems can mathematically be described by partial differential equations (PDEs). Here, one can think for example of heat flow in materials or human tissues, aerodynamic properties of airplanes or determination of option prices in finance. In the last decades the development of efficient numerical methods to solve PDEs gave people together with the rising computing power the opportunity to simulate complex systems. Today this is done very successfully in many areas.Finite Element Analysis (FEA) is widely used in many fields including structures and optimization. The FEA in engineering applications comprises three phases: domain discretization, equation solving and error analysis. The domain discretization or mesh generation is the preprocessing phase which plays an important role in the achievement of accurate solutions.

The use of an adequate mesh is one of the main ingredients for an accurate numerical simulation. In order to obtain such a mesh, a versatile mesh generator and a mesh adaptive procedure must be available. Automatic mesh generation has received much attention from researchers on computational simulation, to minimize manual intervention, to improve mesh quality and to obtain more efficient procedures. Unstructured methodologies are becoming predominant due to the ability of modeling geometrically complex designs and because they are the natural environment for adaptivity, which may be the only hope for resolving very small scale features (e.g. boundary layers). Most finite element and finite volume codes use unstructured triangulations due to their geometrical flexibility and the low cost of linear triangular elements.

In this work, an automatic triangular mesh generator based on the advancing front technique is used as the main building block of a computational system for mesh generation that can build triangular and quadrilateral meshes over arbitrary domains. Quadrilateral elements are generated from an original mesh of triangles through a process of splitting.

The rate of convergence of the finite element method is greatly influenced by the existence of corners on the boundary. Recent investigations shows that proper refinement of the elements around the corners leads to the rate of convergence which is the same as it would be on domain with smooth boundary.

The simplified domains of many engineering problems contain sharp edges and corners, and these often pose important challenges in numerical analyses. Typical examples are reentrant corners in fluid mechanics , and crack tip problems in fracture mechanics etc.
Numerical solutions of elliptic boundary value problems defined on domains with corners have singular behaviour near the corners. This occurs even when data of the underlying problem are very smooth. Such singular behaviour affects the accuracy of the finite element method throughout the whole domain. For example, for the Poisson equation with homogeneous Dirichlet boundary conditions defined on a polygonal domain with re-entrant corners.
The precise description of an object containing *rounded and reentrant corners* leads to consider meshes with a large number of nodes in the corner neighborhood when the finite element method (FEM) is straightforwardly applied. Dealing with such meshes makes the computational work a time and resource consuming task. Instead of using a uniform mesh, one would like to use a non-uniform mesh that is denser near the singularities and coarser elsewhere to minimize the number of unknowns. In this paper, we investigate the choices of nonuniform mesh that give fast converging solutions

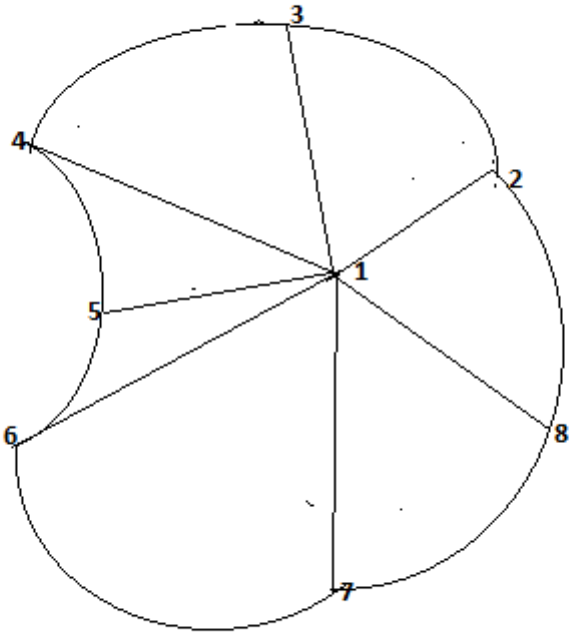
In section 2 of this paper, we present a novel mesh generation scheme of all graded triangles and quadrilaterals finite elements over analytical curved surfaces. This scheme converts the elements in background curved triangular mesh into triangles through the operation of graded subdivision. We first decompose the convex curved domain into simple subregions in the shape of curved triangles. These simple subregions are then triangulated to generate a fine mesh of 3-node graded triangles. We propose then an automatic 3-node graded triangular to 3-node triangular conversion scheme in which each isolated 3-node graded triangle is split into four triangles according to the usual scheme, that is by adding three vertices in the middle of edges and then joining these three vertices by straight lines. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this fully discretizes the given curved domain into all 3-node triangles, thus propogating uniform graded refinement. In section 3 of this paper, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of 6- node graded triangular elements. We can then split each of these 6-node graded triangles into three quadrilaterals by joining the centroid to midside nodes of these triangles. In section 4,we have presented a method of piecing together of all triangular subregions and eventually creating an all graded triangular and quadrilaqteral meshes for the given convex or cracked convex polygonal domain. In section 5,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for standard and arbitrary triangles,rectangles,squares, convex and cracked convex polygonal domains.

2.0 Mesh Generation

It is known that domain discretization plays an important role in finite element analysis as well as in numerous engineering analysis. As a result, various strategies and techniques for generating mesh automatically have been proposed [1-4]. The problem of meshing in two-dimensional case is defined as:
Given a 2D domain Ω together with grading functions $g= g(\xi)$, defined over the entire domain Ω, the task of meshing is to discretize the domain Ω into triangles T or quadrilaterals Q or combination of both in consistency with the given grading functions $g(\xi)$.
The grading functions $g(\xi)$, which specify the element size of the discretization can be defined based on the consideration of the current analysis interests e.g. loading concentrations, boundary conditions etc. The first step in mesh generation is to divide the region into several disjoint sub-regions by openings to holes or connector to holes. For simplicity, here we assume that the region is composed of only straight lines and it is simply connected.

2.1 Division of a Standard Triangle

With reference to following figure Fig.0 we present some necessary definitions and a numerical scheme to generate graded triangles.[25-29]over an arbitrary curved domain.



Fig, 0 AN ARBITRARY CURVED DOMAIN IN TWO DIMENSION

Definition(Algebraically graded meshes in 1D):

A graded mesh of (0,1) generated by a grading function g:[0,1]→[0.1] is graded with grading factor β>0,if $g(\xi)=\xi^{\beta}$

Example:We can generate the (n+1) graded mesh points $x_k^n=(k/n)^{\beta}$,

ie $x_k^n = g(k/n)$,k=0,1,2,...n,n∈ N

Let us assume that the axes of standard triangle u and v are to be referred as $\hat{x}_1$ and $\hat{x}_2$ and c refers to a corner of the standard triangle or unit triangle.

Triangular graded meshes: β - *meshes*

Let K_0 = conv{(1, 0), (0, 0), (0, 1)} be the unit triangle. On K_0, we construct a parametric family of meshes which are graded towards the vertex (0, 0) so as to ensure an optimal rate of convergence of Lagrange interpolating Finite Elements of order $p \geq 1$.. Given an integer $m \geq 2$ and a so-called grading parameter $\beta \geq 1$, let

$$x_l^n = \left(\frac{l}{n}\right)^{\beta} ,l = 0,1,2, \dots \dots n$$

The nodes of the mesh that lie on the rectangular edges of K_0 are $(x_l^n,\ 0)$ and $(0,\ x_l^n)$, $l = 0, 1, \ . \ . \ . \ , n$. Then,being d_l the the diagonal joining $(x_l^n, 0)$ and $(0, z)$, we divide d_l uniformly into $l + 1$ points. This defines all the nodes of a so-called β-graded mesh $T_{n,\beta}(K_0)$ on K_0.

If β = 1, *then* $T_{n,\beta}(K_0)$ *is quasi uniform with meshwidth* $\boldsymbol{h = 1/n}$.

Construction of $T_{n,\beta}(K_0)$,n∈ N on unit triangle K_0

(1) Generate 1D algebraically graded mesh on [0.1]

$0=x_0^n < x_1^n < \cdots\cdots\cdots < x_n^n = 1$, $x_l^n = (l/n)^{\beta}$,$l = 0,1,2, \dots \dots n$

(2) Define "layers": $L_j=\{\ \hat{x} \in K_0: x_{j-1}^n < (\hat{x}_1 + \hat{x}_2) < x_j^n\ \}$,$j = 1,2,3, \dots \dots n$

(3) Generate Triangular mesh $T_{n,\beta}(K_0)|_{L_j}$ on each layer $j = 1,2,3, \dots \dots n$

(4)Union of $T_{n,\beta}(K_0)|_{L_j}$ =triangular mesh of K_0

(5) $T_{n,\beta}(K_0)|_{L_1}$ consists of a single triangle K^* adjacent to (0,0).

When $0<\boldsymbol{\beta} < 1$,we generate an algebraically graded mesh with respect to the edge $\hat{x}_1 + \hat{x}_2 = 1$.

Mesh points over the unit triangle domain

The layer L_j is bounded by the diagonals

$(\hat{x}_1 + \hat{x}_2)=x_{J-1}^n$……..(1 a)

$(\hat{x}_1 + \hat{x}_2)=x_J^n$ ……(1b)

Where ,it is important to note that $x_j^n=(\frac{j}{n})^\beta$ and $0 < x_j^n < 1, j = 1,2,3,\ldots. n$; with x_0^n =0 and x_n^n =1.

Equation (b) above refers to the diagonal joining the points $(x_J^n,0)$ and $(0, x_J^n)$,the solution is $(\hat{x}_1,\hat{x}_2)$=((j-k) x_j^n/j, kx_j^n/j) ,k=0,1,2,……..j; and the solution to Equation(a) is similar

We solve the above equations for $(\hat{x}_1,\hat{x}_2)$ and obtain (j) and (j+1) points at equal distances along the diagonals.The corner (0,0) is a single point and the diagonal joining (1,0) and (0,1) is divided into (n+1) points ((n-k)/n, k/n), k=0,1,....,n. Now, we have all the mesh points necessary for the generation of a graded triangular mesh over a standard triangle.The Cartesian mesh points for an arbitrary triangle can be computed by using affine transformation

We have used the above concepts and written the MATLAB code

[u,v,w] =triangularmeshpoints4singularcorner(p,q)

where u,v,w are area coordinates over the standard triangle T={u,v|0≤ $u, v \leq 1, u + v \leq 1$},

$u + v + w = 1$.

We also have p=n=number of divisions along each side of the standard triangle and q=β is the grading factor as input to the above MATLAB code This code denser mesh points in the neighbourhood of a corner for β> 1 and denser points in the vicinity of the edge(hypotenuse)

Clearly,β = 1,generates a uniform triangular mesh. The following outputs of the above code demonstrate this point of view.

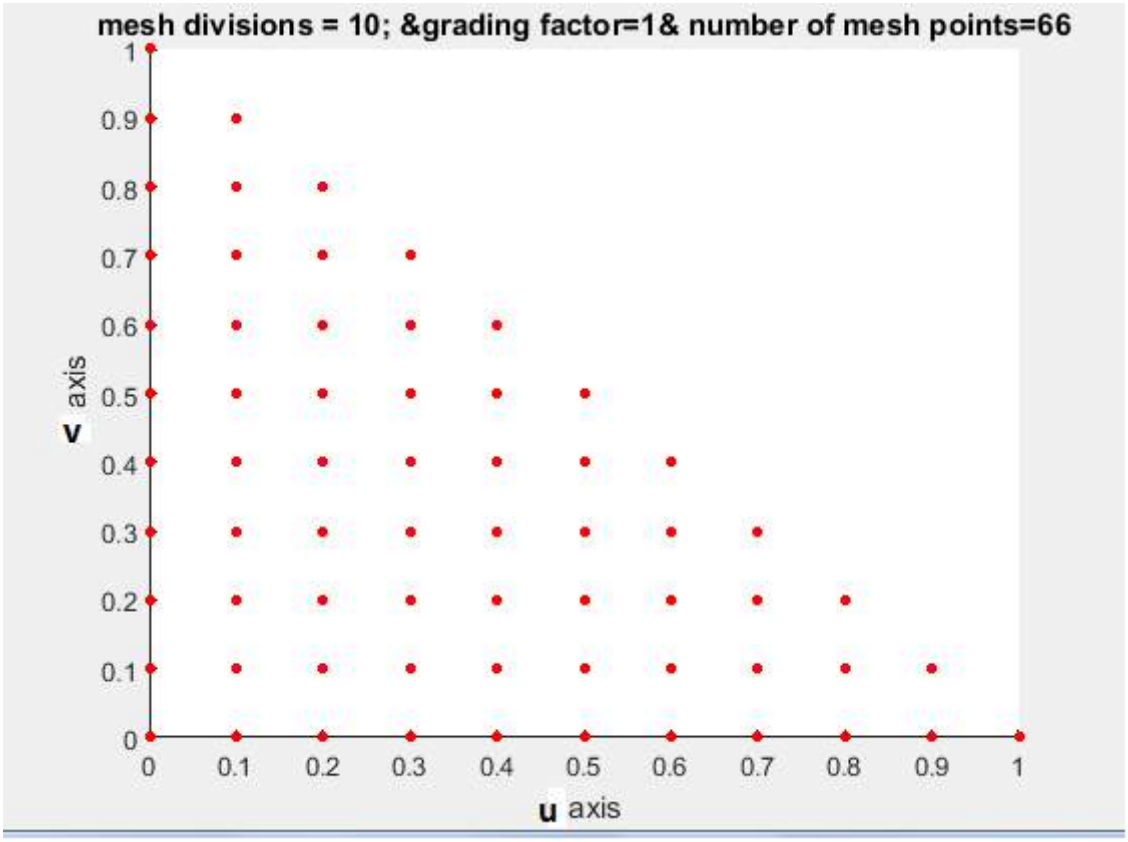


Fig.1a Distribution of mesh points for uniform mesh

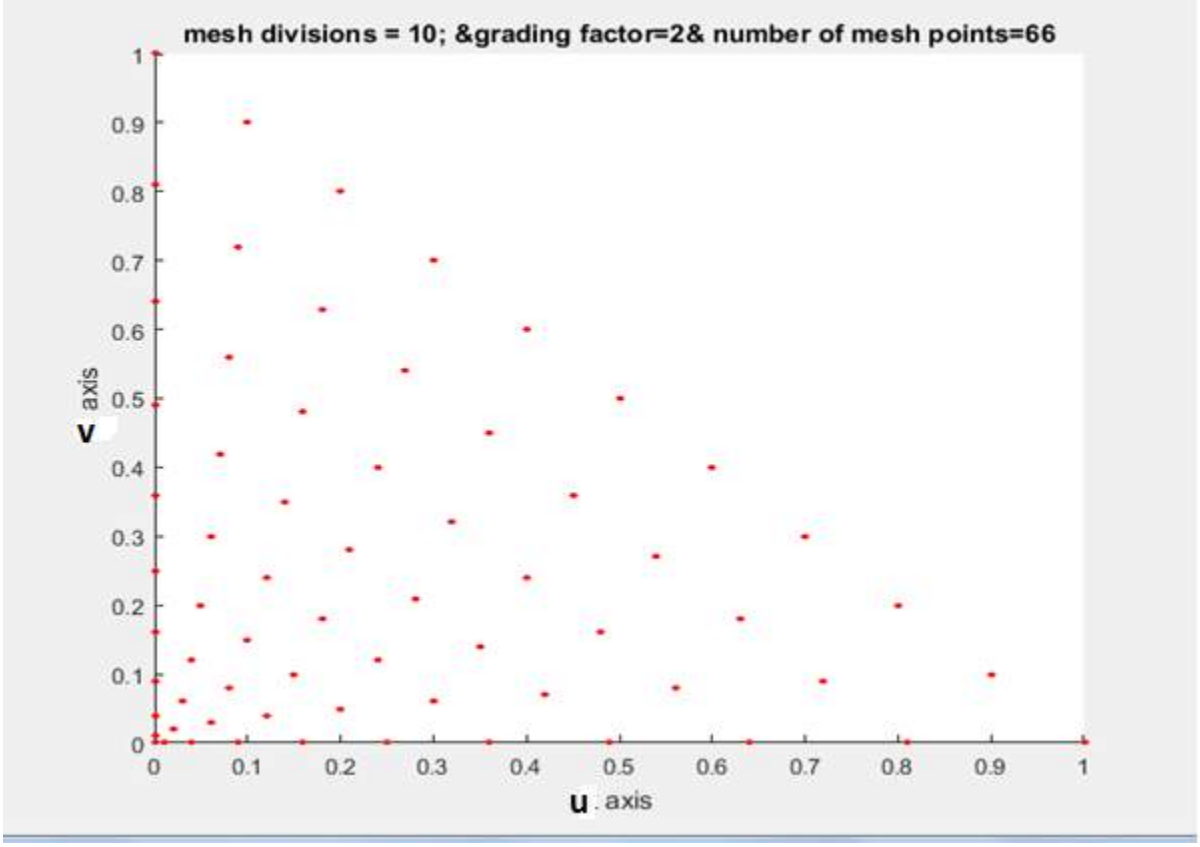


Fig.1b Distribution of mesh points for nonuniform mesh(graded mesh) denser near corner (0,0)

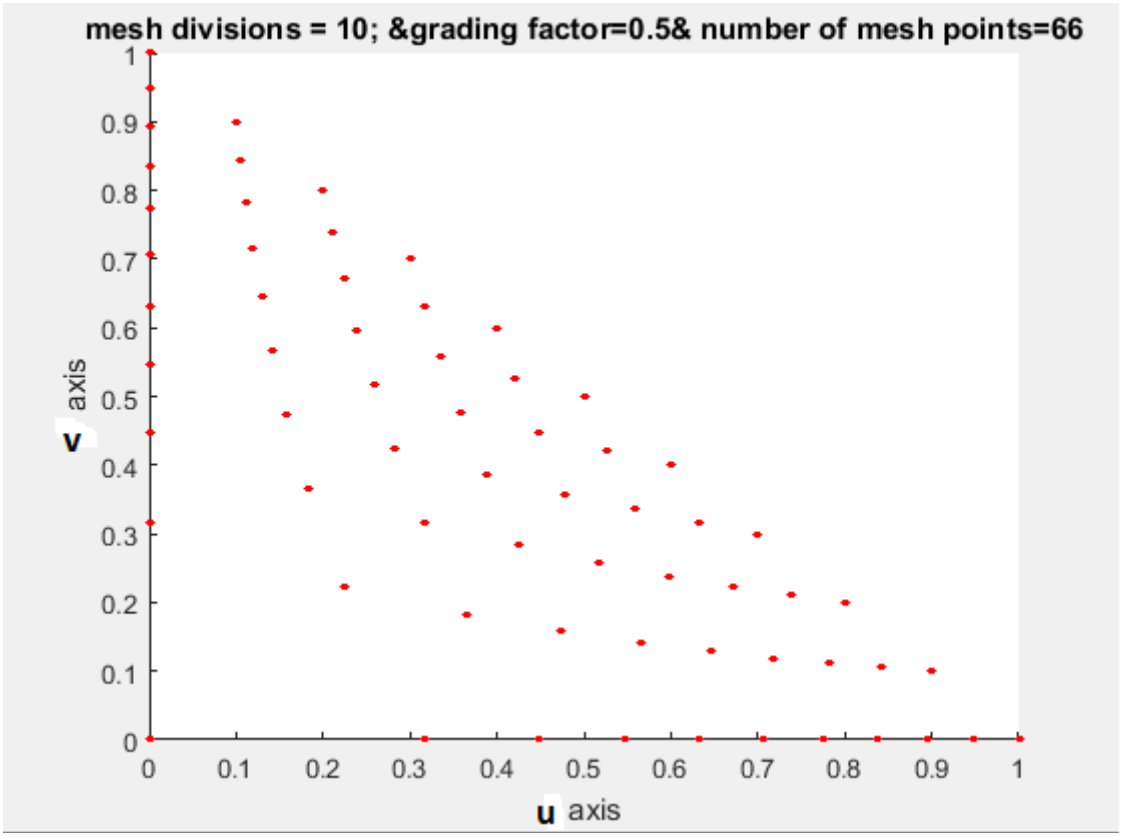


Fig.1c Distribution of mesh points for nonuniform mesh(graded mesh) denser near the edge(hypotenuse)

EXAMPLES OF GRADED MESHES OVER A STANDARD TRIANGLE

We present below a sample of graded meshes for β>1 which are denser in the vicinity of a corner.

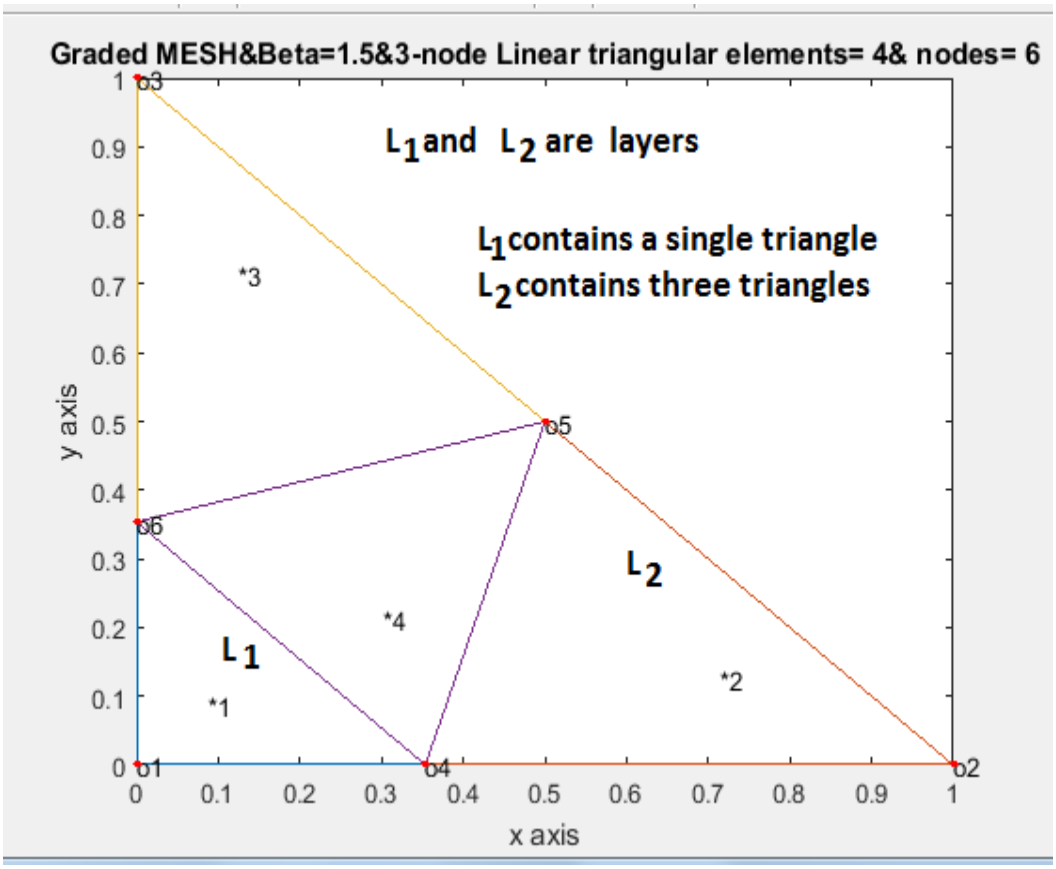


Fig.1d

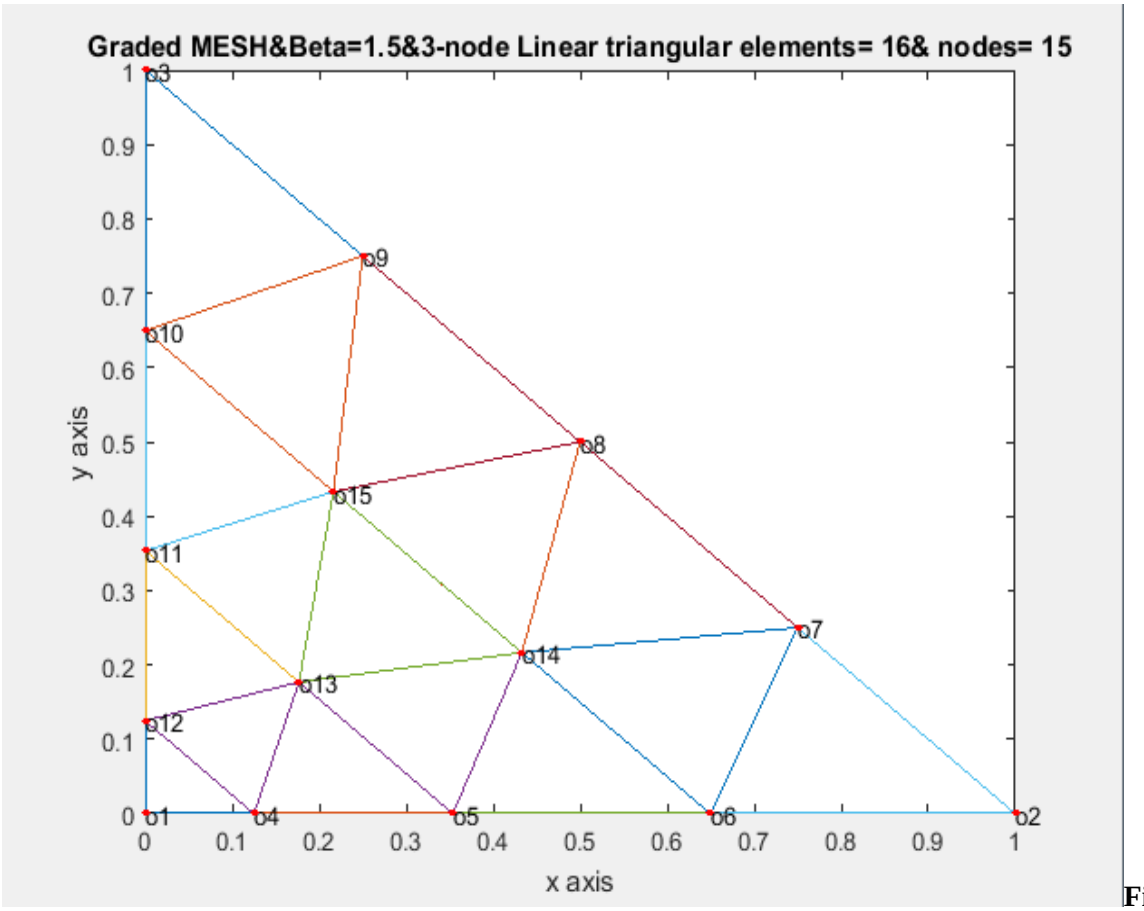


Fig.1e

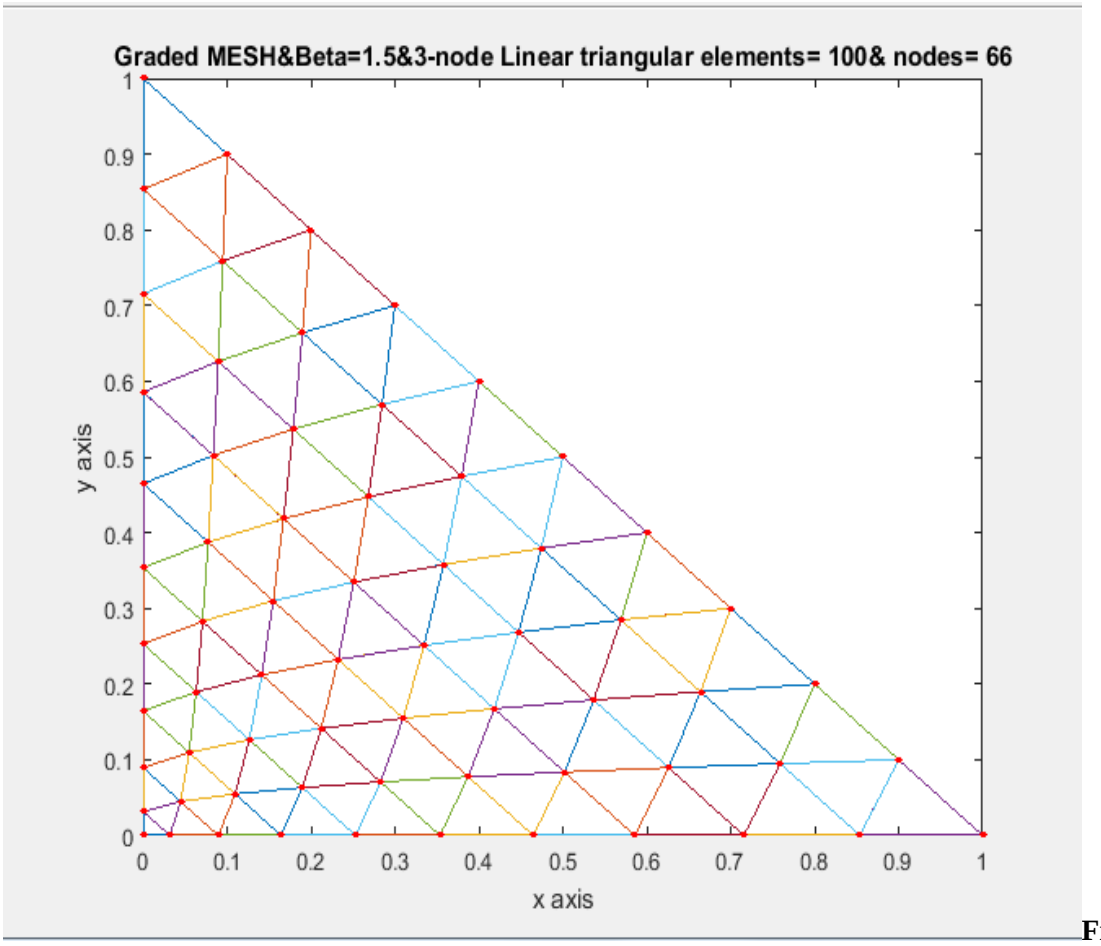


Fig.1f

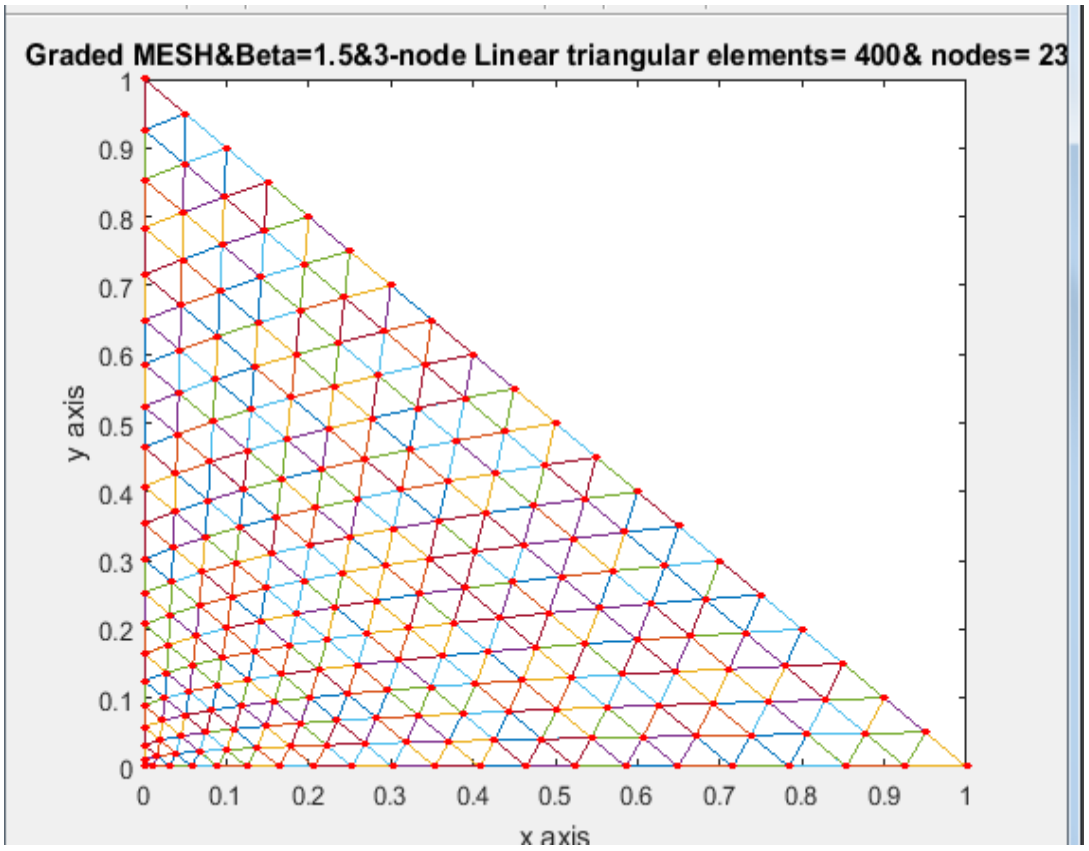


Fig.1g

The above examples indicate that for $\beta \neq 1$ the scheme can generate the 3-node graded triangles layer by layer only.If six node triangles are required,we can insert midside nodes to each of these.We can then insert an additional node at the centroid of these 3-node triangle and discritise this triangle into three

quadrilaterals,that is by joining the centroid to the midside nodes.This will be further explained in the succeeding sections. The mesh generation is explained for uniform mesh generation,that is for β=1 which is equally valid for graded triangles β≠1,β>0

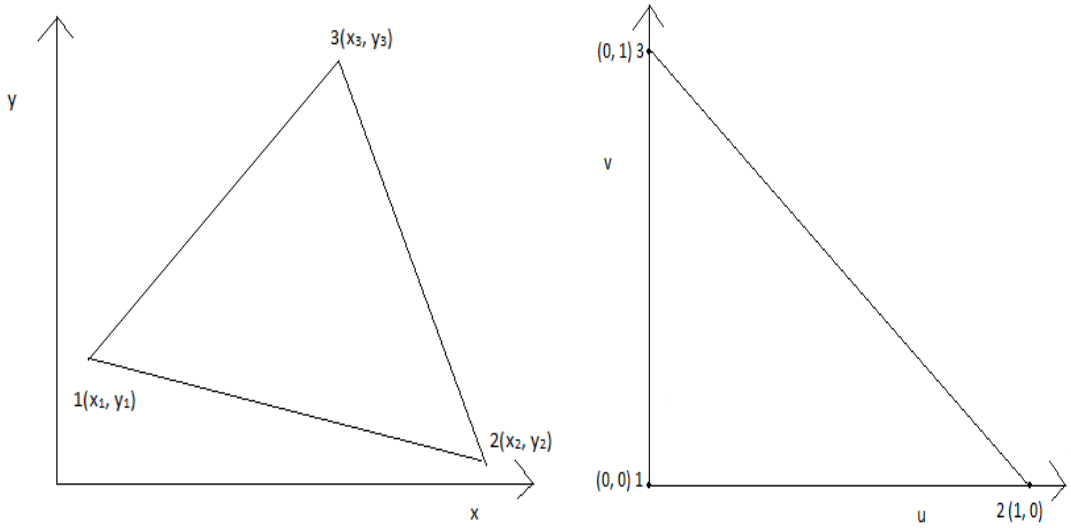
2.2 Division of an Arbitrary Triangle

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

$$\boldsymbol{x = x_1 + (x_2 - x_1)u + (x_3 - x_1)v}$$

$$\boldsymbol{y = y_1 + (y_2 - y_1)u + (y_3 - y_1)v} \qquad \textbf{(2a-b)}$$

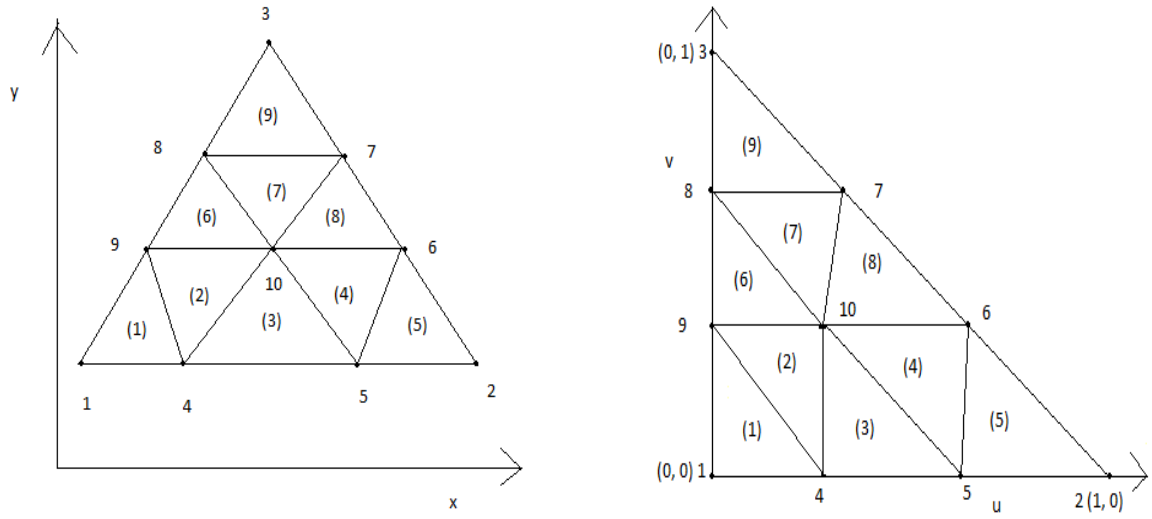
The mapping of eqn.(2a-b) describes a unique relation between the coordinate systems. This is illustrated by using the area coordinates and division of each side into three equal parts in Fig. 2a Fig. 2b. It is clear that all the coordinates of this division can be determined by knowing the coordinates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



1.h **1. i**

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

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



2a **2b**

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

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

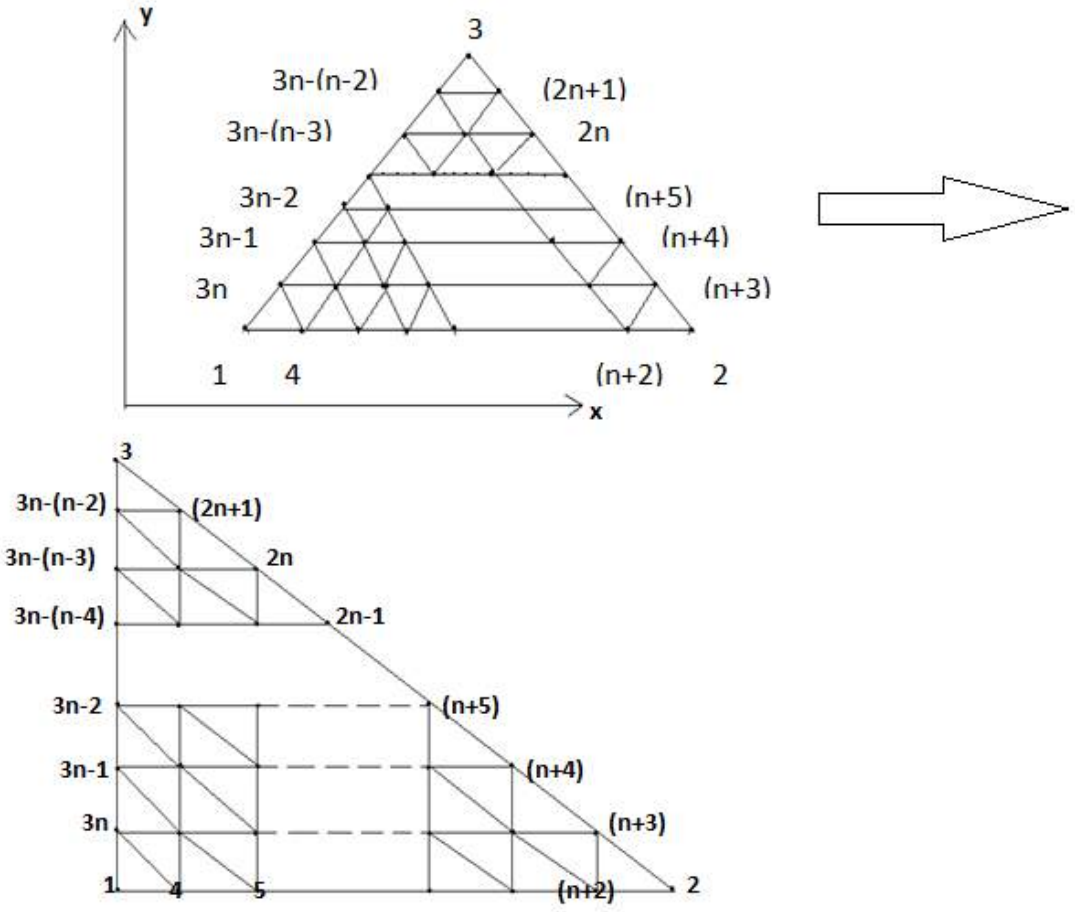


Fig. 3a Division of an arbitrary triangle with vertices$((x_i, y_i\ ), i = 1,2,3)$ into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisions of equal length

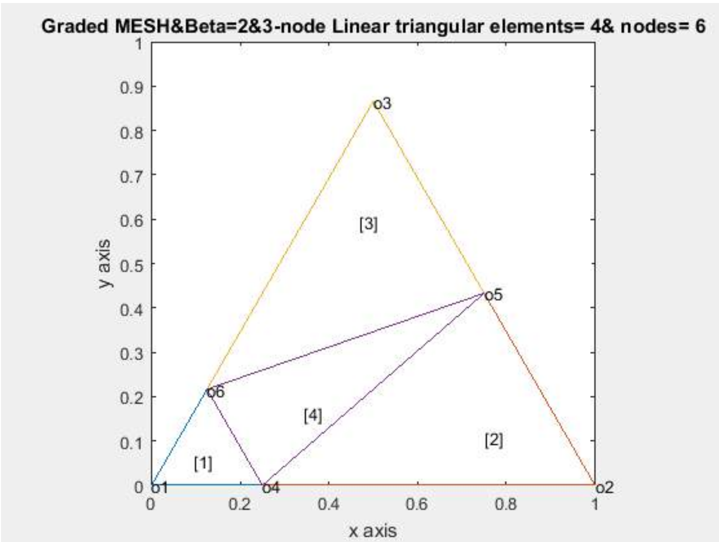
Fig. 3b Division of a right isosceles triangle with vertices{1(0,0),2(1,0),3(0,1)} into n^2 right isosceles triangle in (u, v) space, where each side is divided into n divisions of equal length

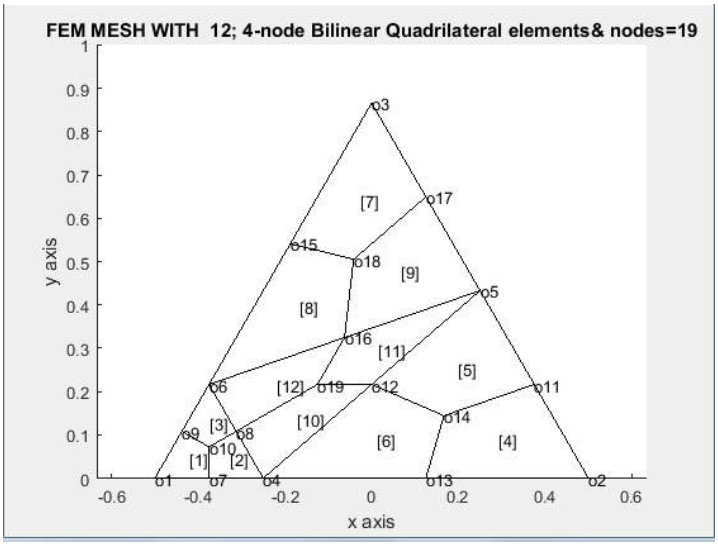
We have shown the division of an arbitrary triangle in Fig. 3a , Fig. 3b, We divided each side of the triangles (either in Cartesian space or natural space) into n equal parts and draw lines parallel to the sides of the triangles. This creates (n+1) (n+2)/2 nodes. These nodes are numbered from triangle 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

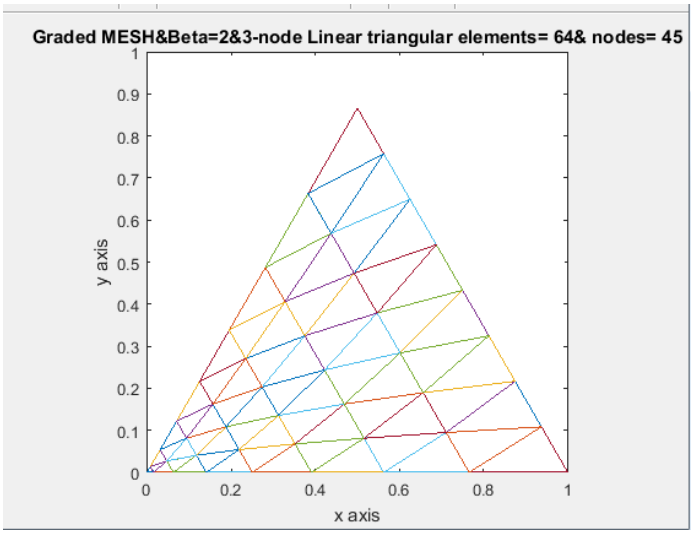
$$\underline{rr} = \begin{bmatrix} \mathbf{1}, & \mathbf{4}, & \mathbf{5}, \dots\dots\dots\dots, & (\boldsymbol{n}+\mathbf{2}), & \mathbf{2} \\ \mathbf{3n}, & (\mathbf{3n}+\mathbf{1}), \dots\dots\dots, & \mathbf{3n}+(\boldsymbol{n}-\mathbf{2}), & (\boldsymbol{n}+\mathbf{3}), & \mathbf{0} \\ \mathbf{3n}-\mathbf{1},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

-----------(2c)

EXAMPLES OF GRADED MESHES OVER AN ARBITRARY TRIANGLE







3.0 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

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

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 ξ *or* η 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 $t_{10} = \frac{1}{12}(t_1 + t_2 + 4t_3 + 3t_4 + 3t_5)$, $(t = x, y)$

…… (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 $(x_1, y_1), (x_4, y_4),$ $(x_5, y_5) \equiv \left(x_4 - \frac{1}{3}x_1 + \frac{1}{3}x_2, y_4 - \frac{1}{3}y_1 + \frac{1}{3}y_2\right)$ and (x_2, y_2) and hence the transformation equation in eq(3) reduces to

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

Equations (5a) can be written as

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

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

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

……(6c)

We display here the division of curved domain and its transformation to a right isosceles triangle.

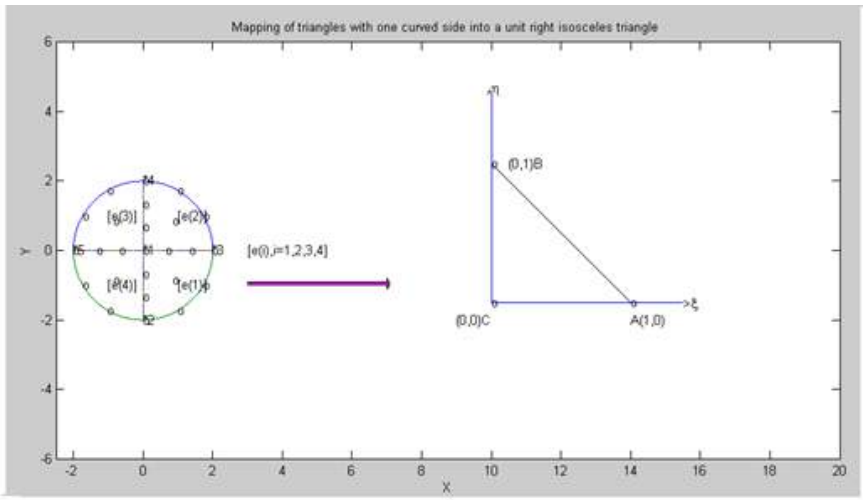
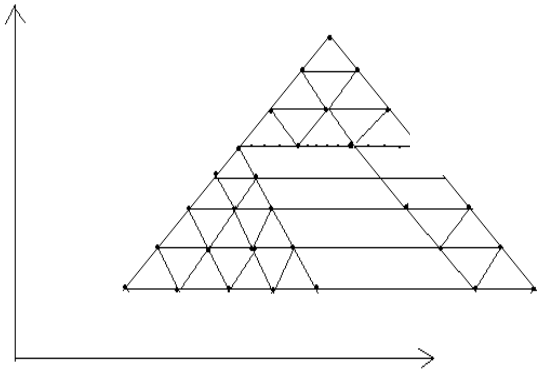
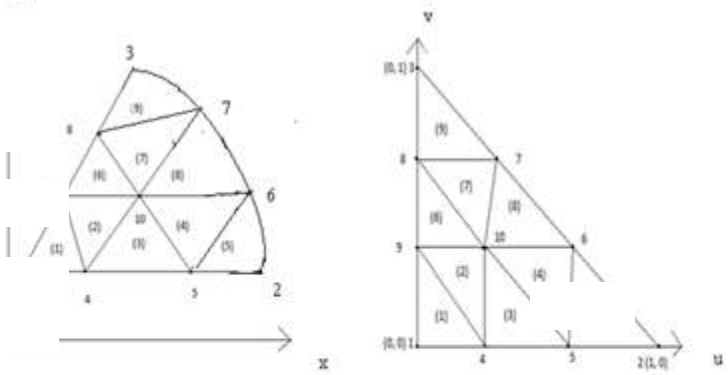


Fig.1c-a circle divided into four arbitrary curved triangles (e(i),i=1,2,3,4)

Fig.1d-a right isosceles triangle ABC

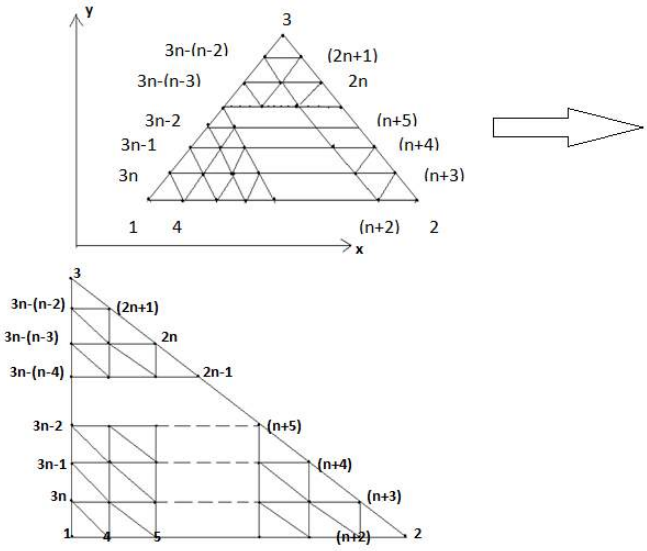




2(2(d)

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

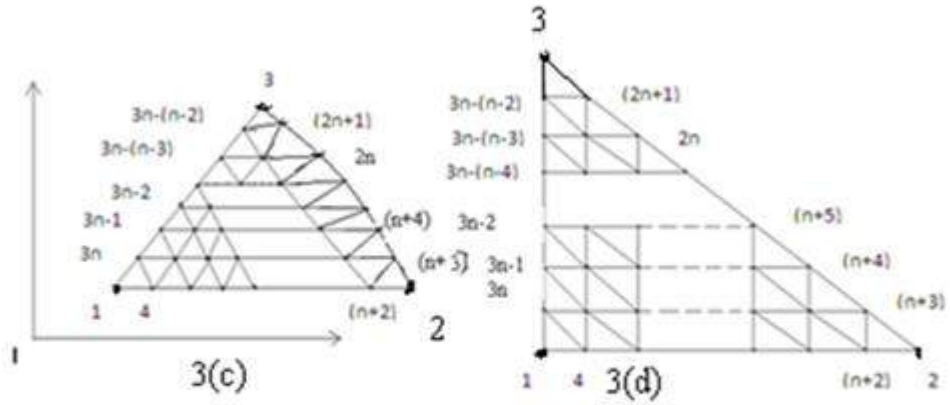


Fig 3c Division of a curved triangle into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisons of equal width

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,this matrix is already shown in eqn(2c)

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.

3.2 Triangulations and Quadrangulations

We now first consider the triangulations and quadrangulations of an arbitrary linear triangle. We divide the arbitrary linear 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

(i)Triangulation

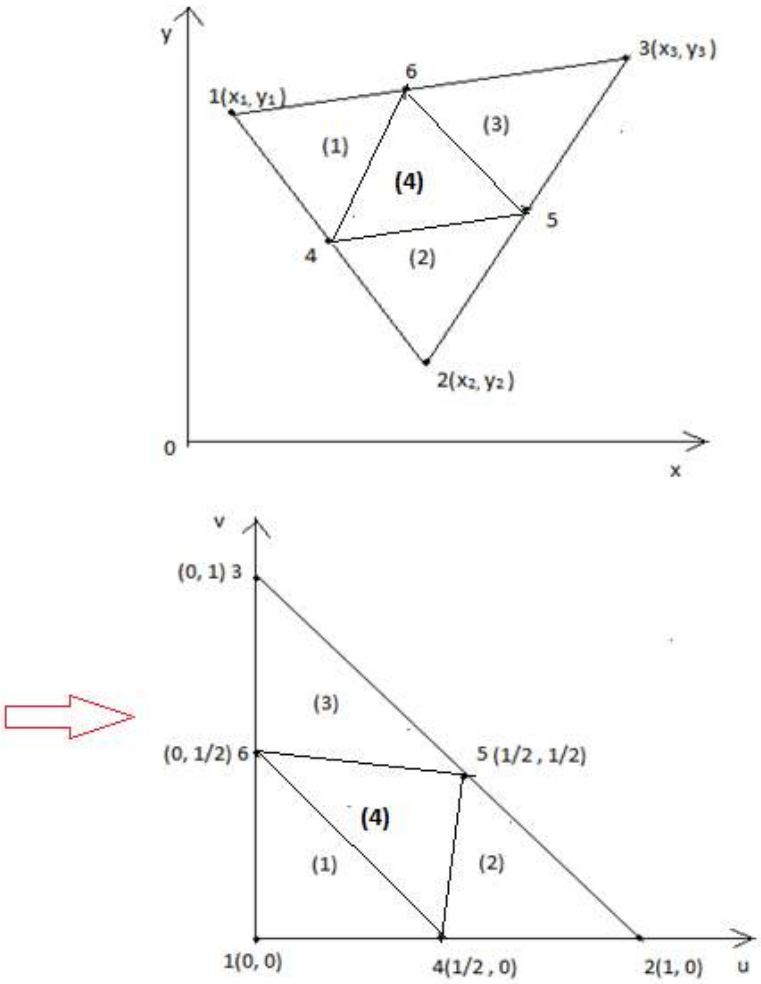


Fig 4(a). Division of an arbitrary triangle into four triangles

Fig 4(b). Division of a standard triangle into four triangles

(ii) Quadrangulation

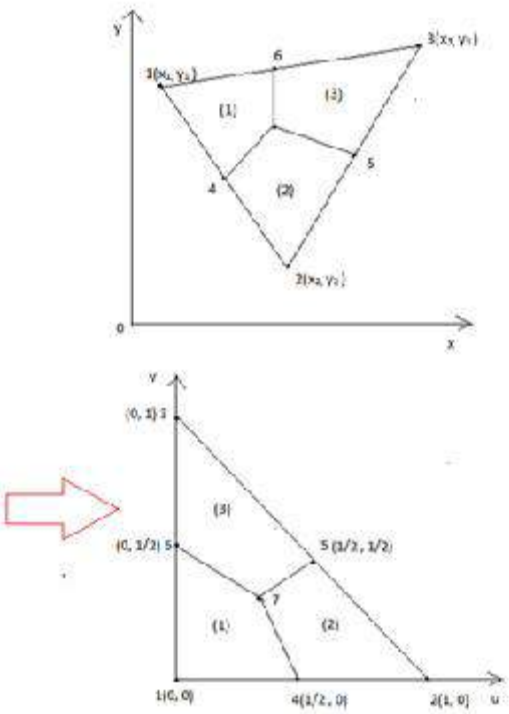


Fig 4(c). Division of an arbitrary triangle into three quadrilaterals

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

Four Divisions of Each side of an Arbitrary Linear Triangle

(i)Triangulation

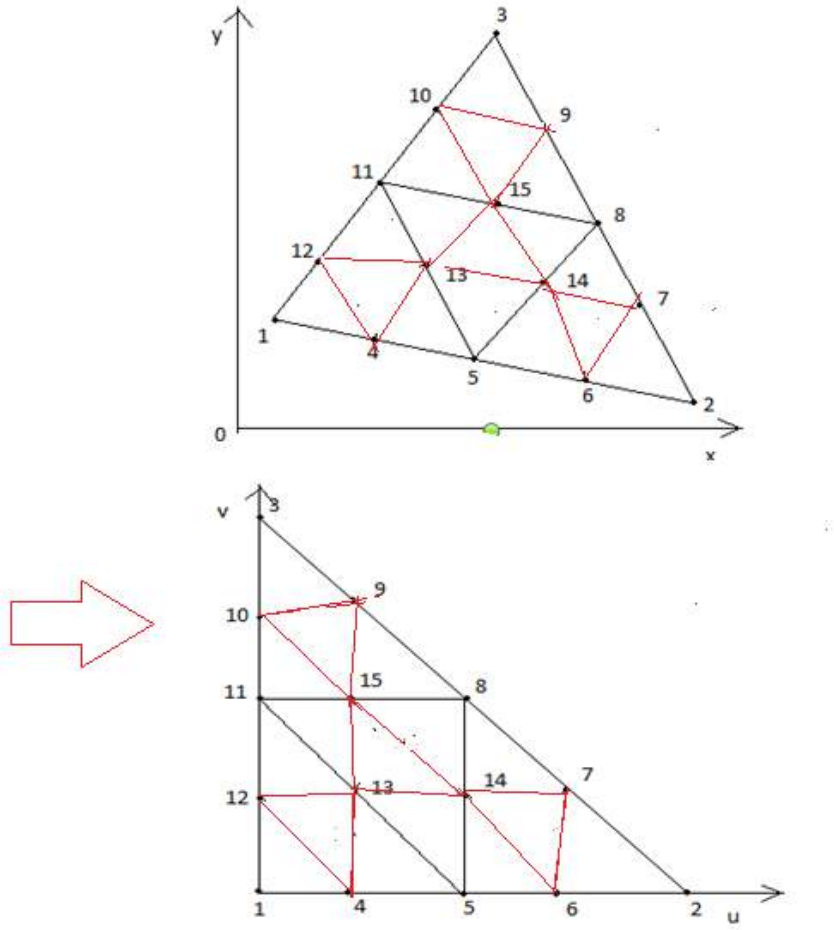
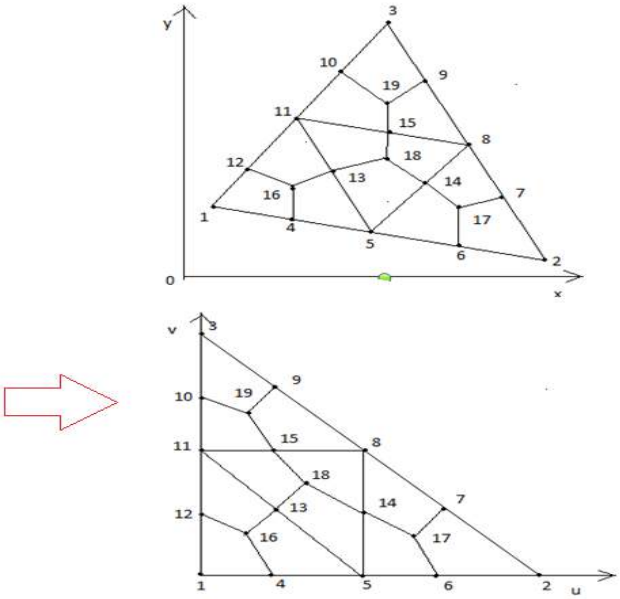


Fig 5a. Division of an arbitrary triangle into 4 six node triangles and then into sixteen triangles

Fig 5b. Division of a standard triangle into 4 six node triangles and then into sixteen triangles



(ii)Quadrangulation

Fig 5c. Division of an arbitrary triangle into 4 six node triangles and then into sixteen quadrilaterals Fig 5d. Division of a standard triangle into 4 six node triangles and then into sixteen quadrilaterals

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 $\underline{rr}$

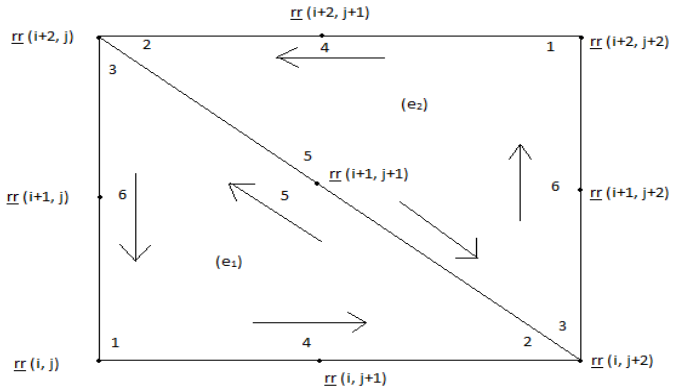


Fig. 6 Six node triangle formation for non zero sub matrix $\underline{rr}$

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

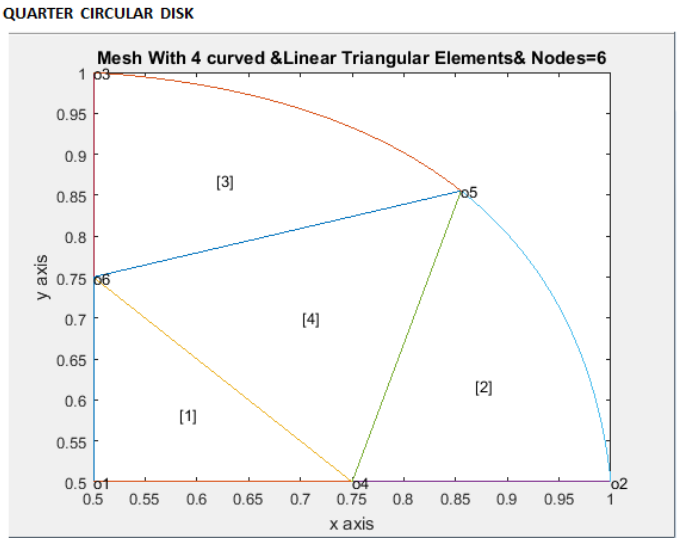
Q_{3n_2} { c_2 , $\underline{rr}$ $(i+1,\ j+1)$, $\underline{rr}$ $(i,\ j+2)$, $\underline{rr}$ $(i+1, j+2)$ } ---------- (5)

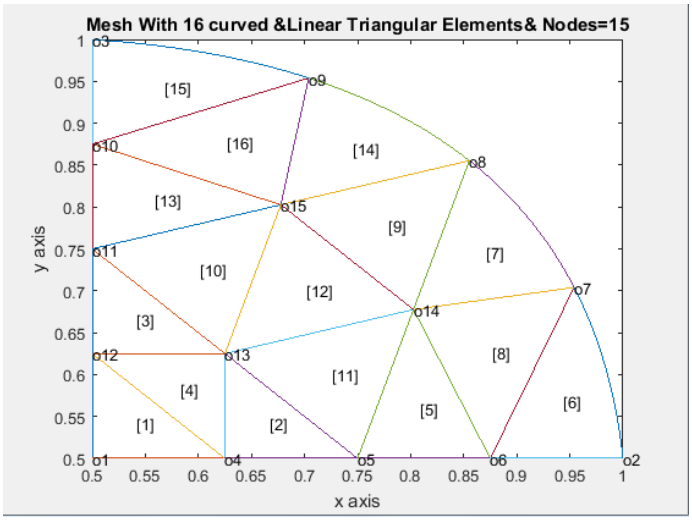
wherein again we read {… … …} as connecting the nodal addresses.

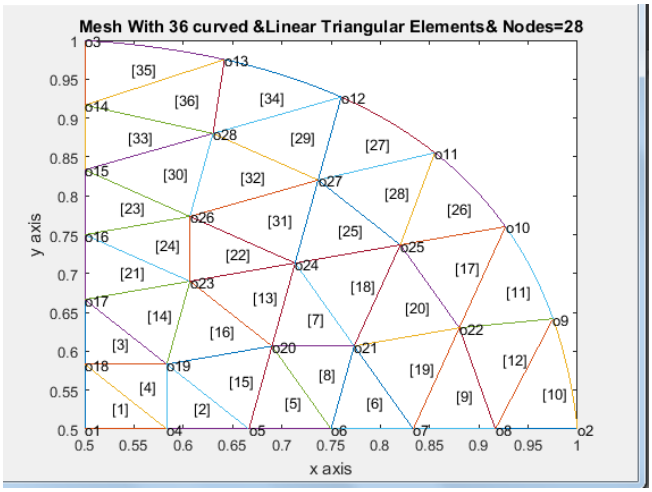
3.3 Some Illustrative Examples

Examples:UNIFORM TRIANGULAR MESHES OVER A QUARTER CIRCLE

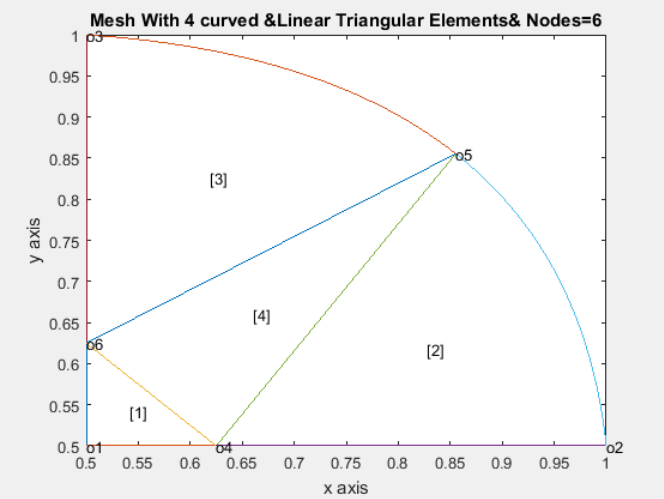
We now consider quarter circle as an example of a curved triangle.The mesh generation is displayed for selective examples in the following figures.

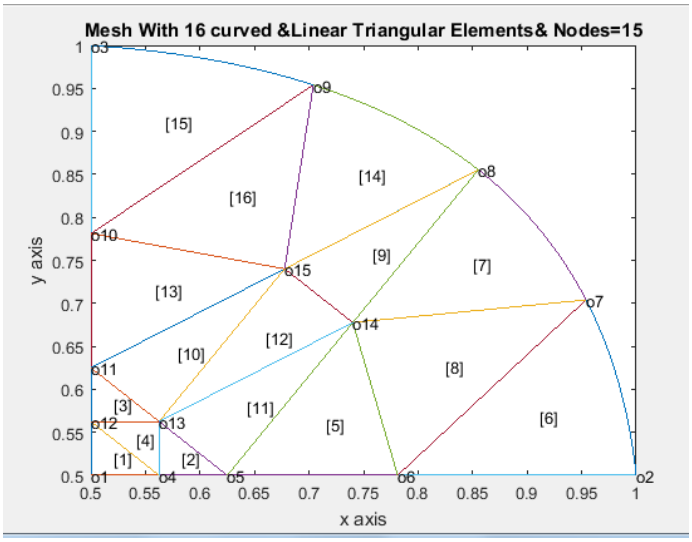


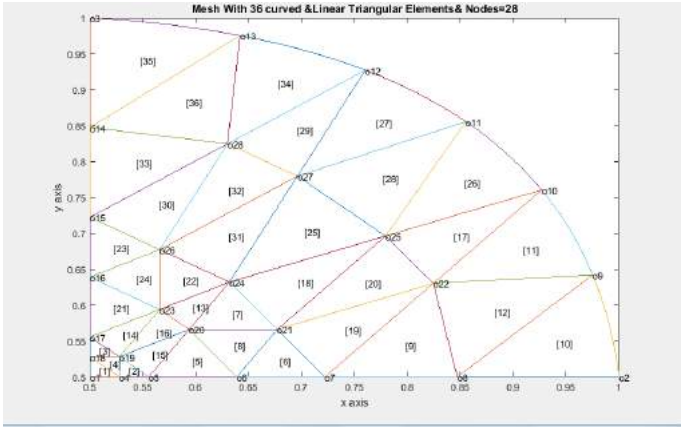


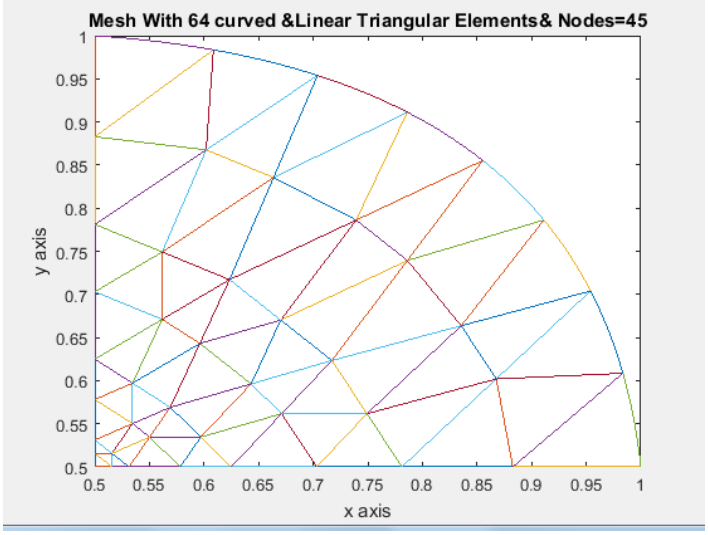


Examples:Graded Triangular Meshes over Quarter Circle

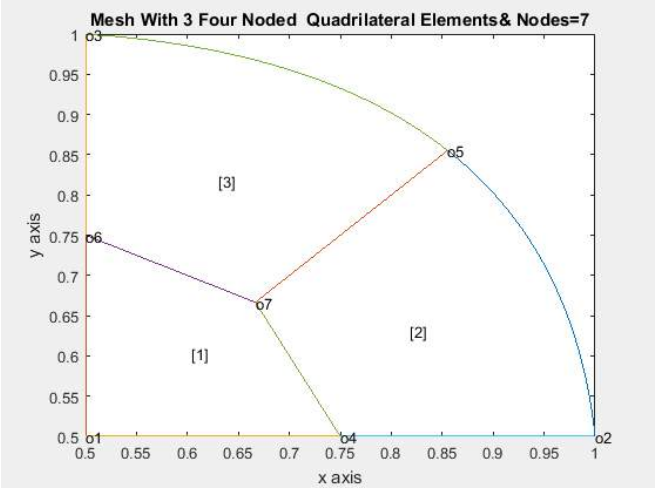


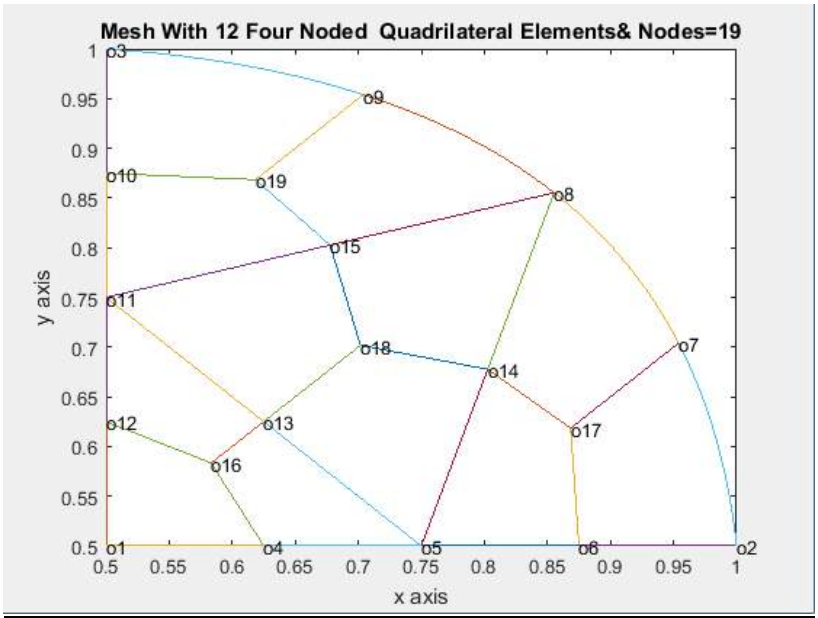


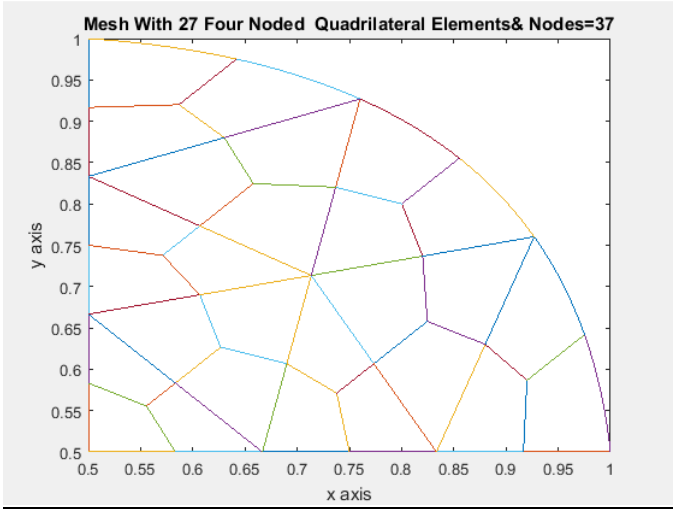




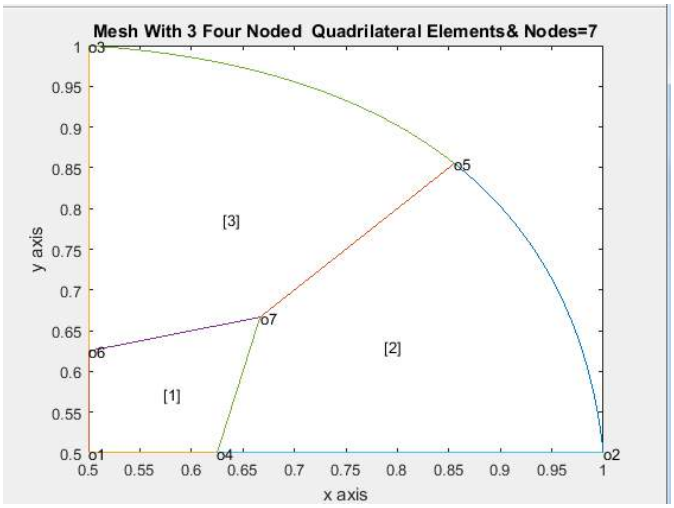
Examples:UNIFORM QUADRILATERAL MESHES OVER A QUARTER CIRCLE

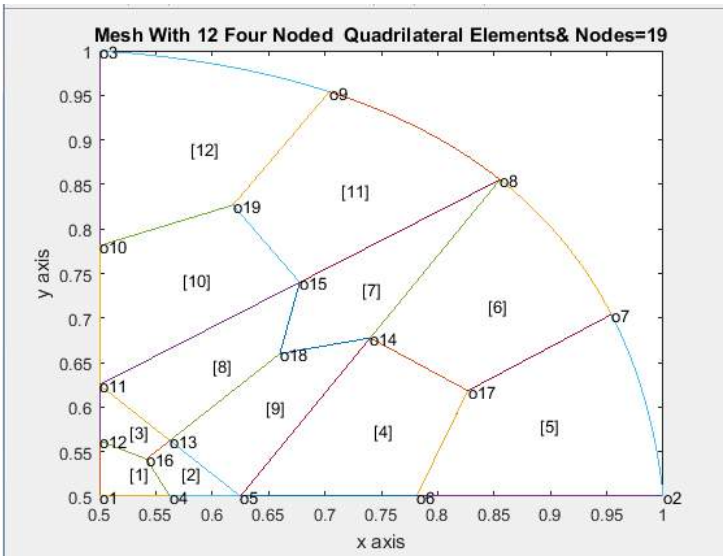


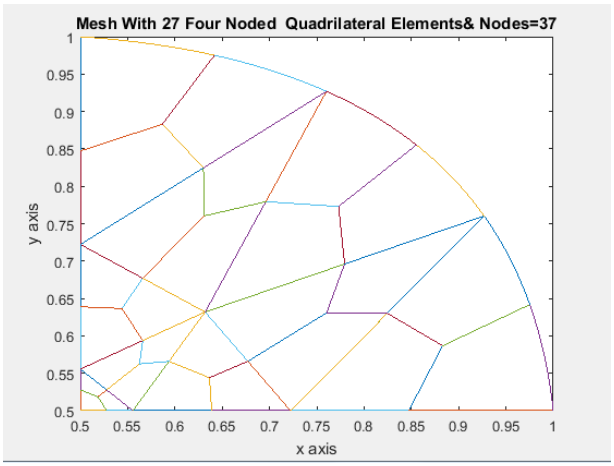




Examples:GRADED QUADRILATERAL MESHES OVER A QUARTER CIRCLE







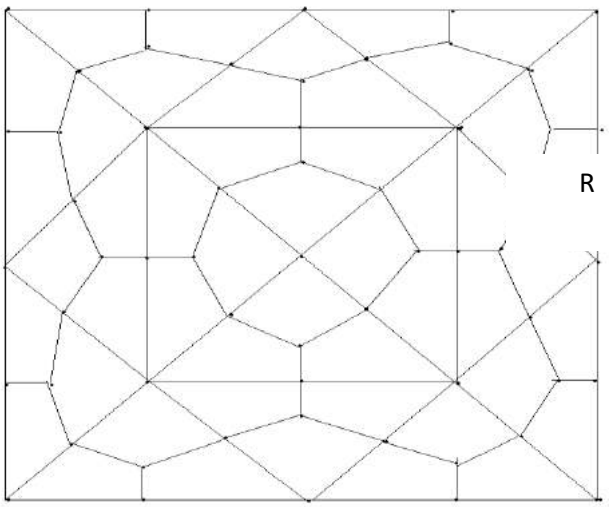
4. Triangulation and Quadrangulation of the Curved Surfaces by using Parabolic Arcs

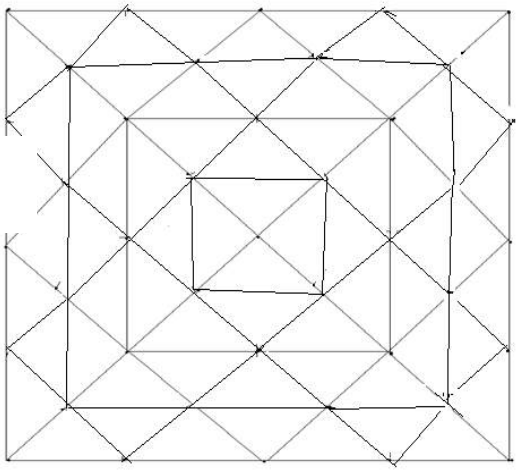
We can generate polygonal and analytical curved surface meshes by piecing together triangles with straight sides (linear triangle) and curved sides(curved triangle) respectively by using subsections (called LOOPs). The user specifies the shape of these LOOPs by designating six coordinates of each LOOP

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)

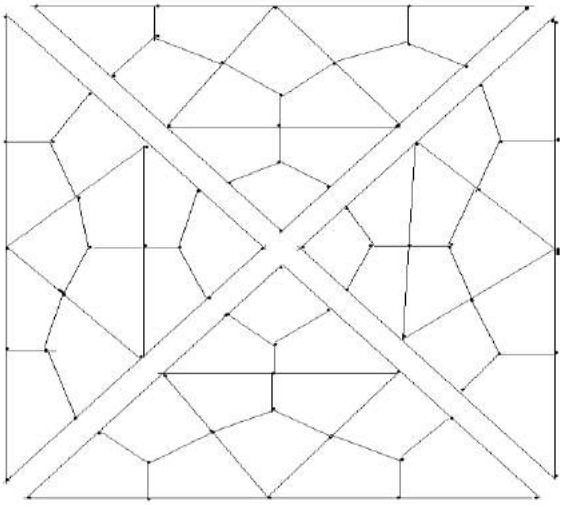
(i)Fig. 8(a) Region R to be analyzed

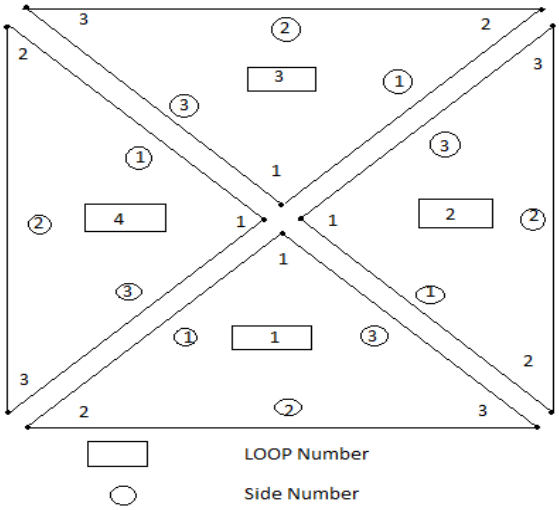




(ii) Fig. 8(b) Example of complete quadrangulated mesh and

(ii) Fig. 8(c) Example of complete triangulated mesh





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

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

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

A CURVED DOMAIN IN THE SHAPE OF AN ELLIPSE

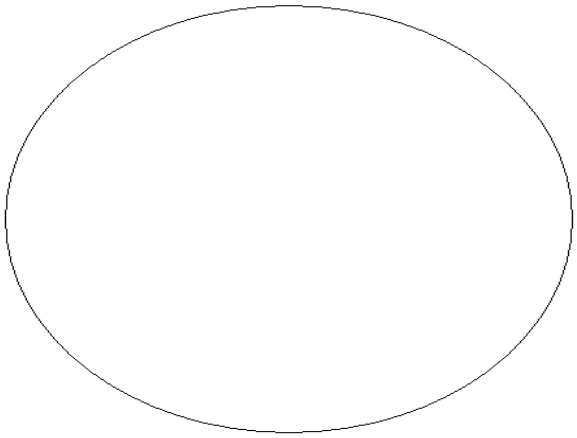


Fig.9a

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

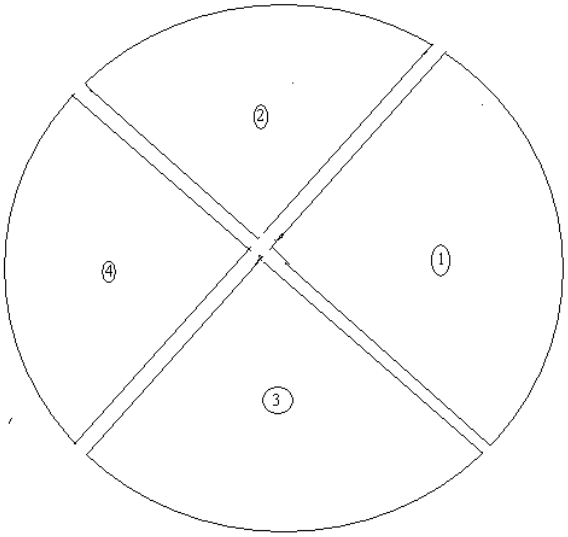


Fig.9b

ELLIPSE AS A CURVED DOMAIN MADE UP OF TWELVE QUADRILATERALS

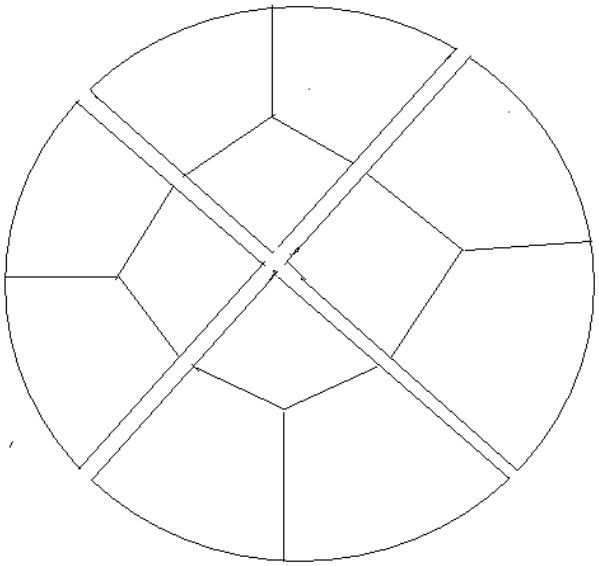


Fig.9c

AN ELLIPSE AS CURVED DOMAIN DISCRITISED INTO TWELVE QUADRILATERALS

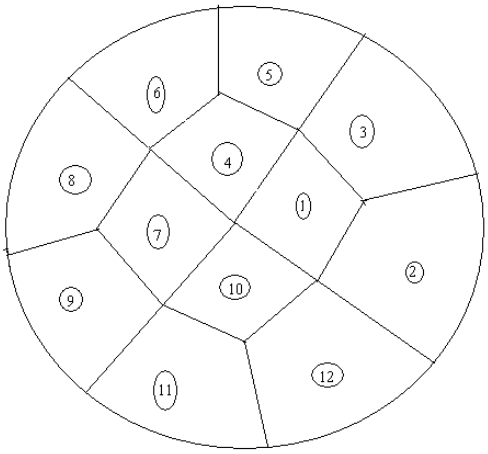


Fig.9d

AN ELLIPSE AS A CURVED DOMAIN DISCRETISED INTO SIXTEEN TRIANGLES

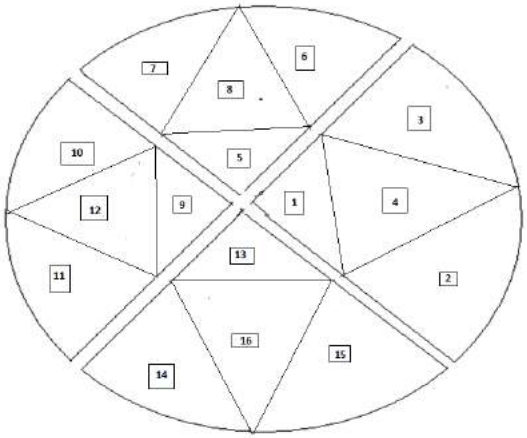


Fig 9e

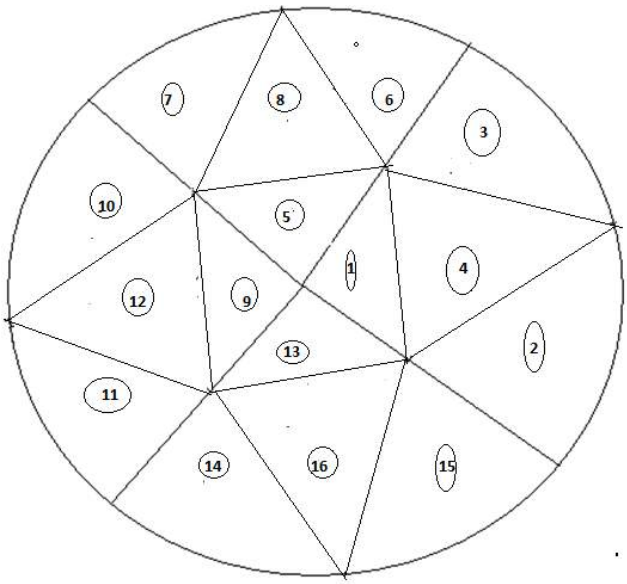


Fig.9f

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. Automesh Generation Scheme

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), $\eta_i = \frac{(i-1)}{n}$, i $= 1,2, \dots,$ n, (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.

(I)COMPUTER PROGRAMS FOR TRIANGULATION

Essential Programs
(1)curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3.m
(2)curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3.m
(3)curvedpolygonal_domain_QUADcoordinatesNtr3.m
(4) nodaladdresses_special_convex_quadrilateralsNtr3.m
(5) nodaladdresses4special_convex_quadrilateralsQUADtrialNtr3 .m
Common Program
(6) `triangularmeshpoints4singularcorner.m`
Application Specific Programs
A Quarter Circle,Semi Circle ,Three Quadrants of a Circle and A Complete Circle
(7) masterelementnodescoordinates_circulardisk.m
(8) globalnodalcoordinate_circulardisk.m

(II)COMPUTER PROGRAMS FOR QUADRANGULATION

Essential Programs
(9) curvedquadrilateralmesh4convexpolygoneightsidesq40N.m
(10) curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N.m
(11)curvedpolygonal_domain_QUADcoordinatesN.m
(12) nodaladdresses_special_convex_quadrilateralsN.m
(13) nodaladdresses4special_convex_quadrilateralsQUADtrialN.m

We have included some meshes generated by using the above MATLAB codes. We further illustrate the application of above codes by generating meshes for a quarter circle,a semicircle ,three quadrants of a circle and a complete circle. 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 which can be converted to form four triangles in each 6-noded triangle. 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 bilinear 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,three quadrants a semicircular region as illustrative 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 development 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] 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 , Bharath Rathod , K.V.Vijayakumar , K. Sugantha Devi, A new automatic finite element mesh generation scheme of all quadrilaterals over an analytical curved surface by using parabolic arcs,International Journal Of Engineering And Computer Science ISSN:2319-7242,Volume -3 Issue -8 August, 2014 Page No. 7437-7507
[30] 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)
[31] 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

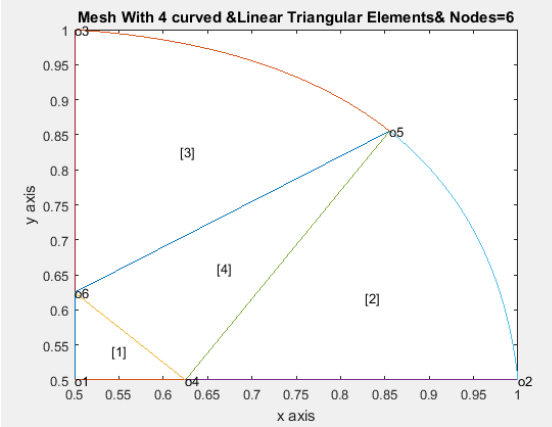
APPENDIX-A

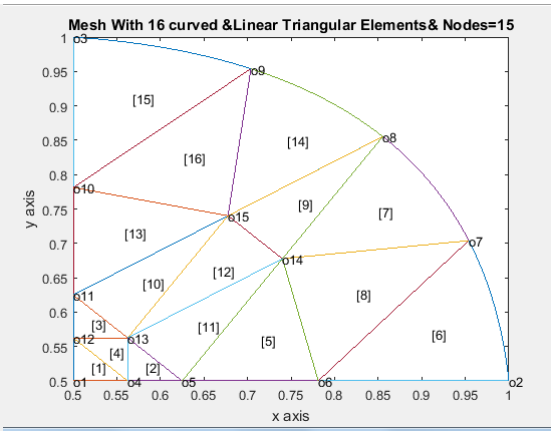
$\beta - GRADED\ AUTOMESH\ GENERATION\ OVER\ CURVED\ DOMAINS\ BY\ TRIANGLES\ , (\beta = 2)$

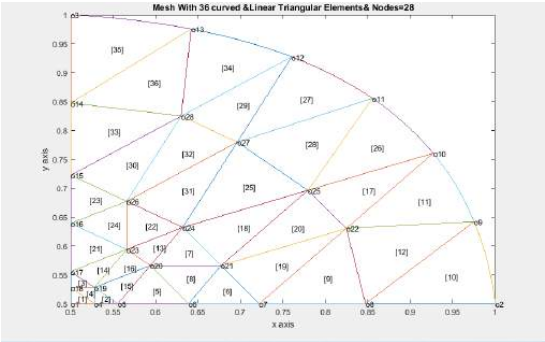
(1)QUARTER CIRCLE

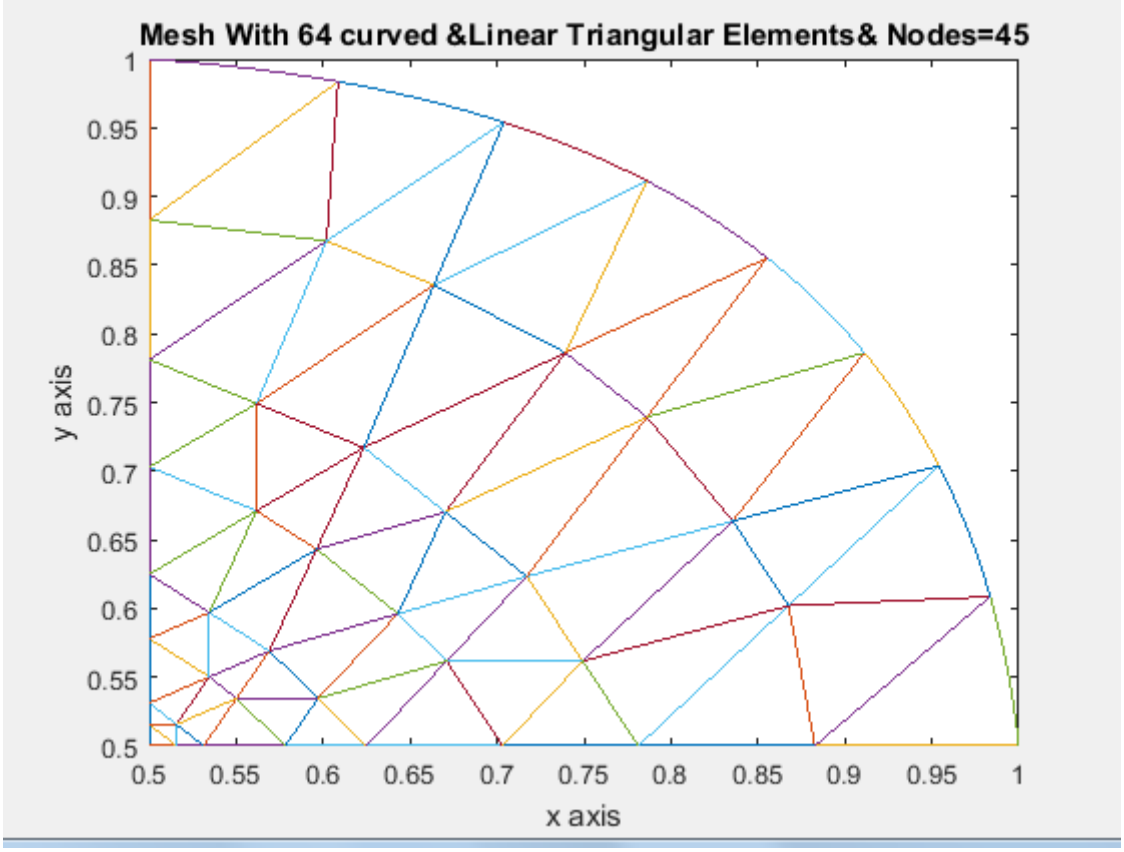
MATLAB COMMANDS

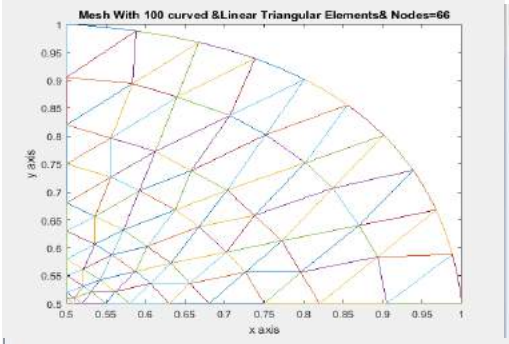
```
For ndiv=2:2:10
n=ndiv;
curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3([3],[1],[2],[4],[5],3,(n/2)^2,n,15,1,1)
end
curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3([3],[1],[2],[4],[5],3,100,20,15,1,1)
curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3([3],[1],[2],[4],[5],3,400,40,15,1,1)
```

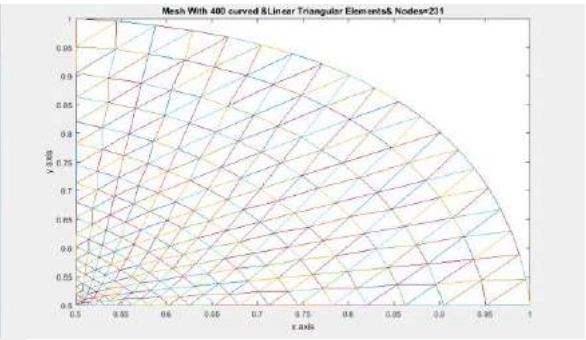

Mesh With 4 curved &Linear Triangular Elements& Nodes=6
y axis
x axis
[1]
[2]
[3]
[4]

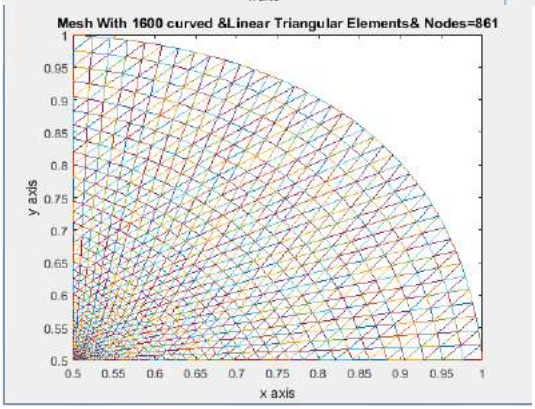

Mesh With 16 curved &Linear Triangular Elements& Nodes=15
y axis
x axis


Mesh With 36 curved &Linear Triangular Elements& Nodes=28


Mesh With 64 curved &Linear Triangular Elements& Nodes=45
y axis
x axis

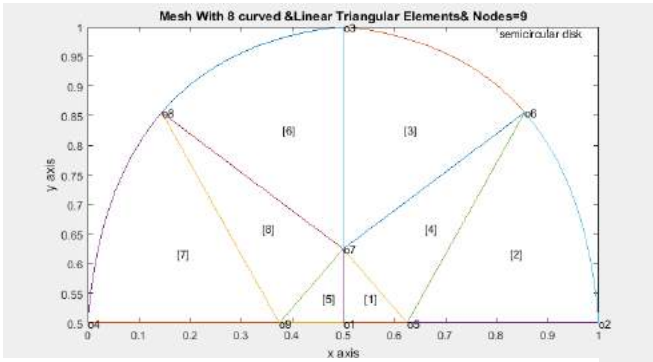


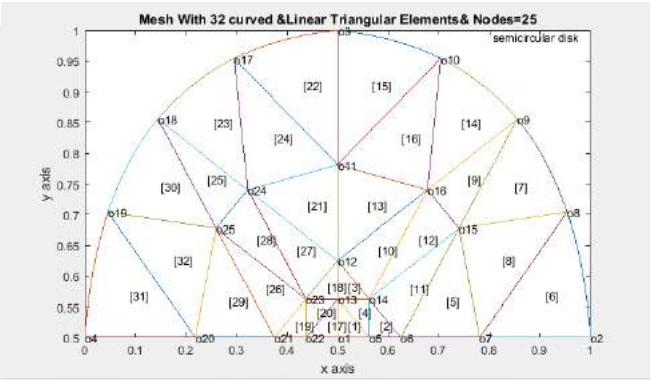


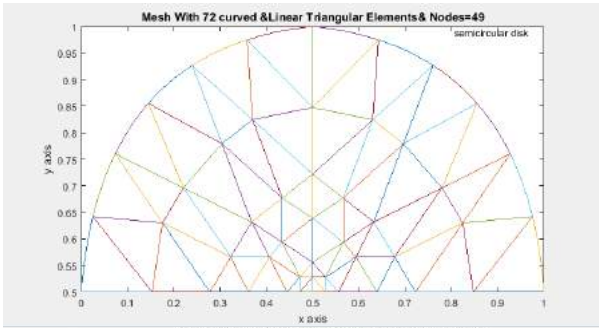


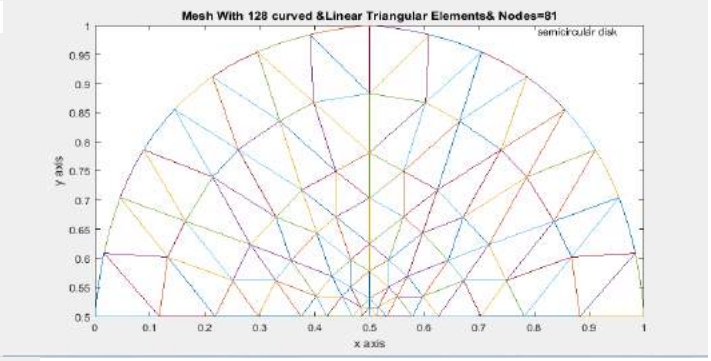
(2)SEMICIRCLE
MATLAB COMMANDS

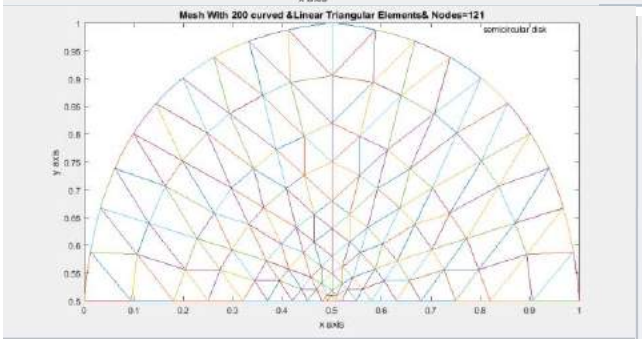
```
for ndiv=2:2:10
n=ndiv;
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,(n/2)^2,n,12,1,1)
end
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,100,20,12,1,1)
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,400,40,12,1,1)

keyboard input:1  and then  Y
```

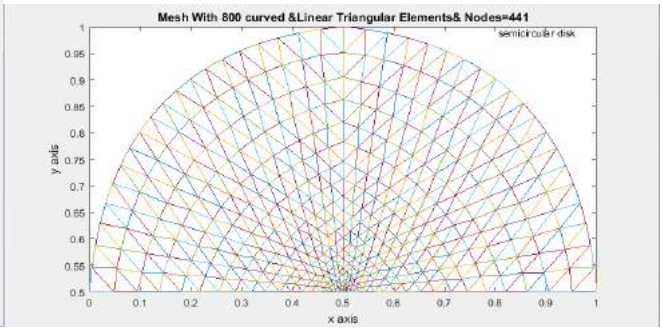


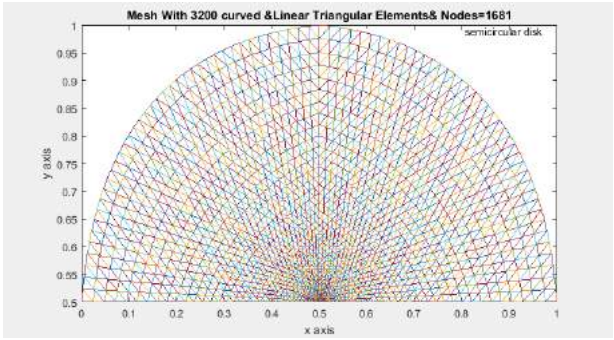








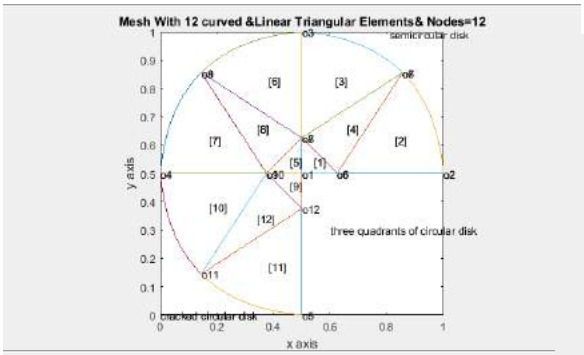


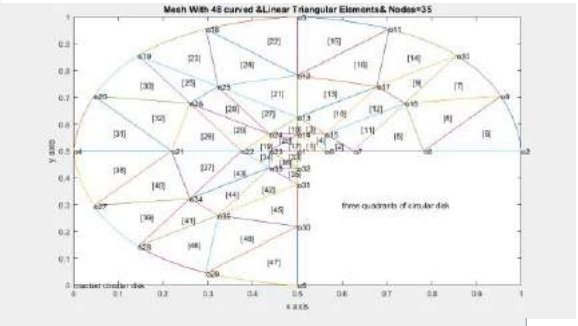


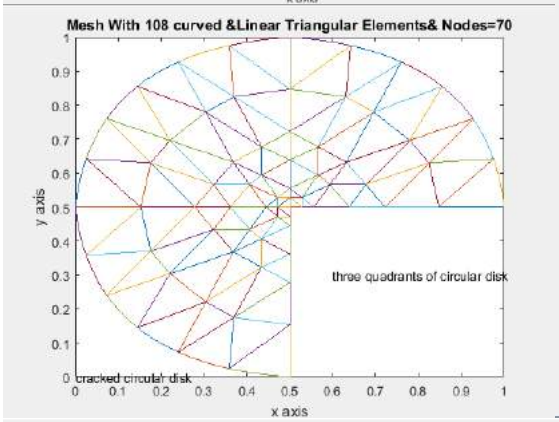
(3)THREE QUADRANTS OF A CIRCLE

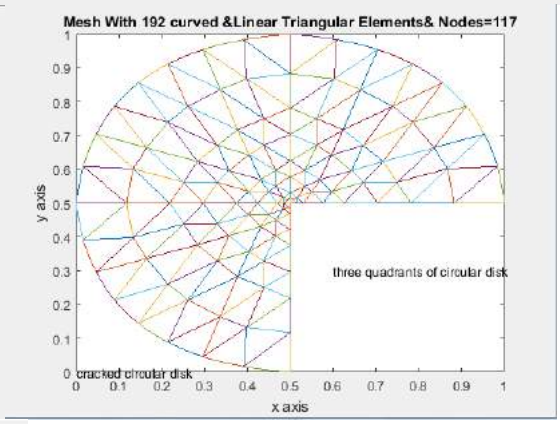
For ndiv=2:2:10
N=ndiv;
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,(n/2)^2,n,12,1,1)
end
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,100,20,12,1,1)
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,400,40,12,1,1)

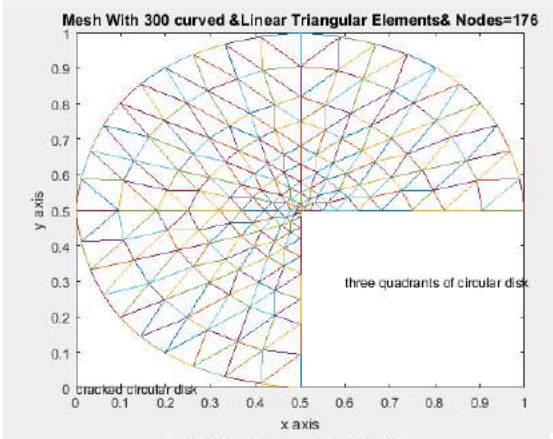
keyboard input:1 and then N

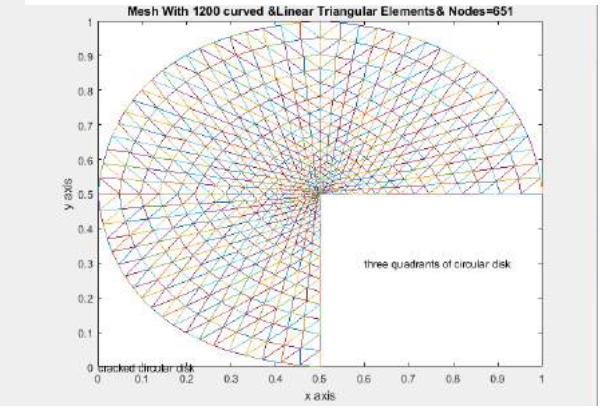


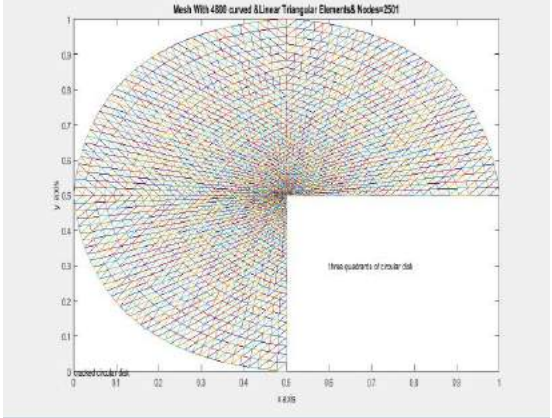












APPENDIX-B

β − GRADED AUTOMESH GENERATION OVER CURVED DOMAINS BY QUADRILATERALS ,($\beta = 2$)

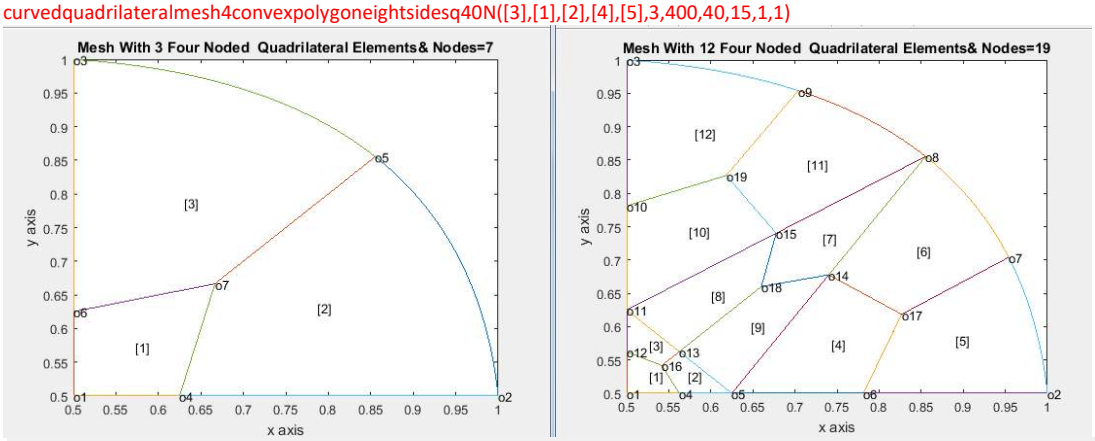
(1)QUARTER CIRCLE

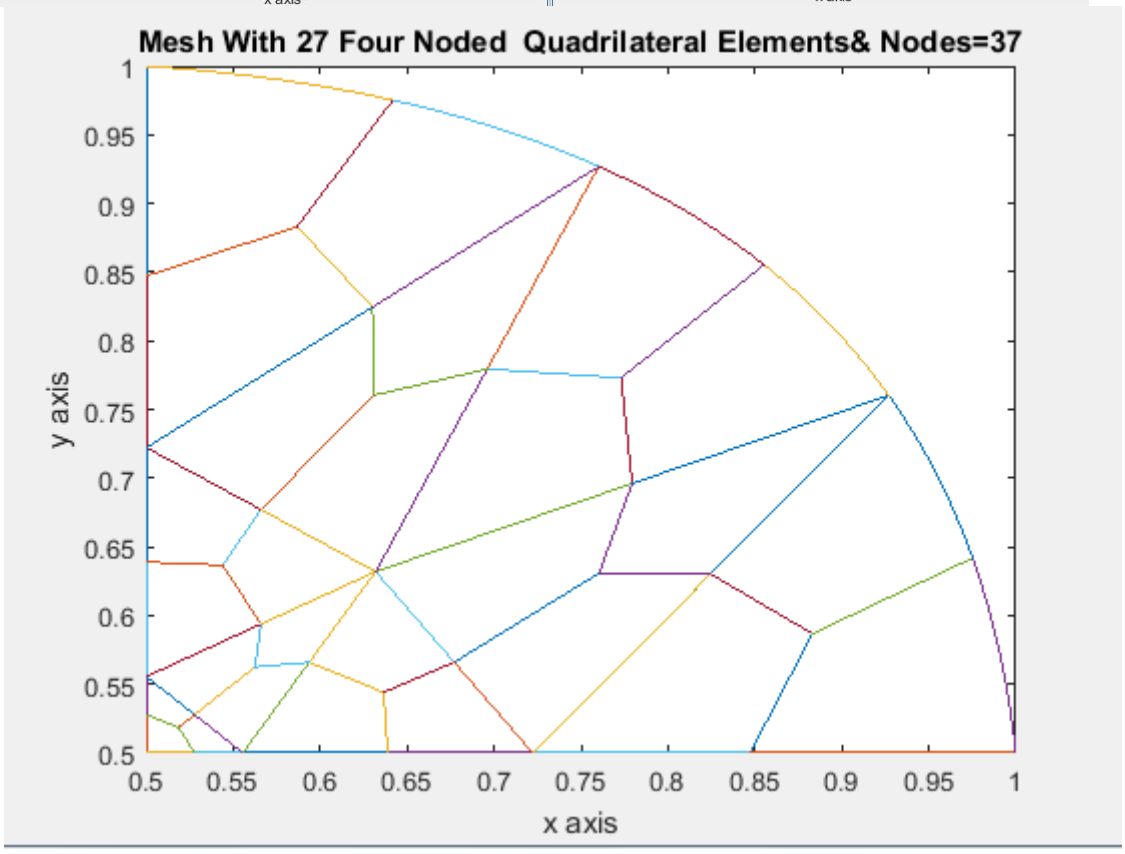
MATLAB COMMANDS

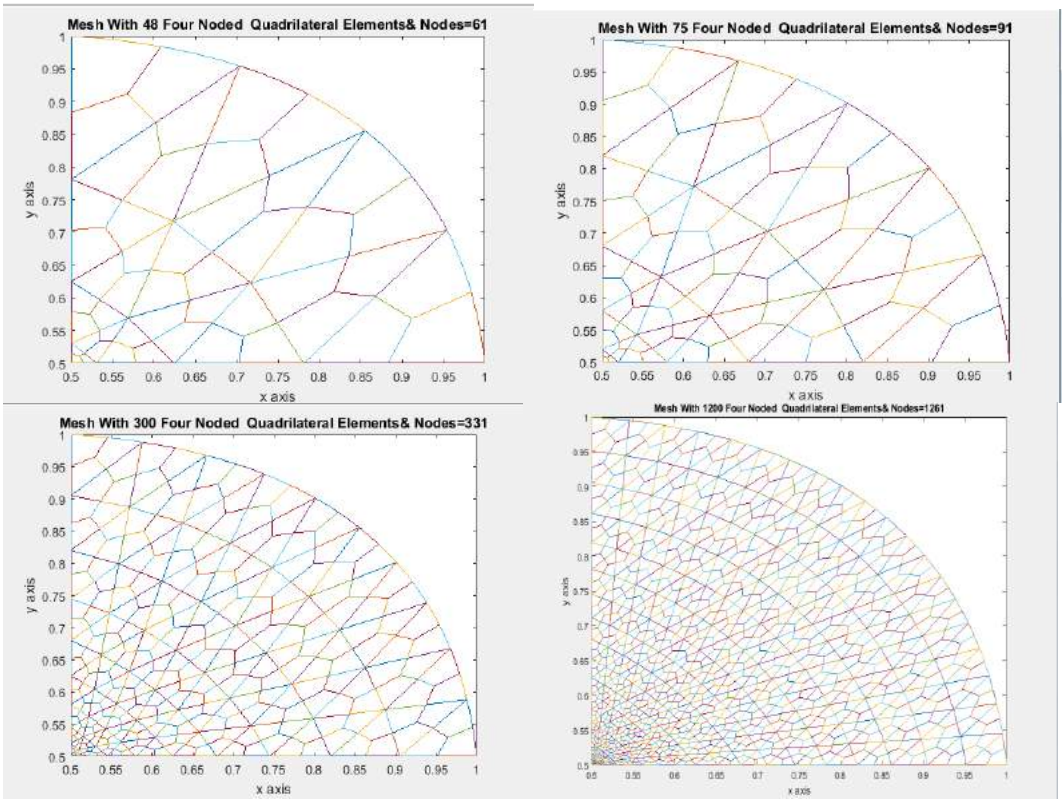
For ndiv=2:2:10

n=ndiv;

curvedquadrilateralmesh4convexpolygoneightsidesq40N([3],[1],[2],[4],[5],3,(n/2)^2,n,15,1,1)
end
curvedquadrilateralmesh4convexpolygoneightsidesq40N([3],[1],[2],[4],[5],3,100,20,15,1,1)
curvedquadrilateralmesh4convexpolygoneightsidesq40N([3],[1],[2],[4],[5],3,400,40,15,1,1)







(2) SEMICIRCLE

MATLAB COMMANDS

For ndiv=2:2:10

n=ndiv;

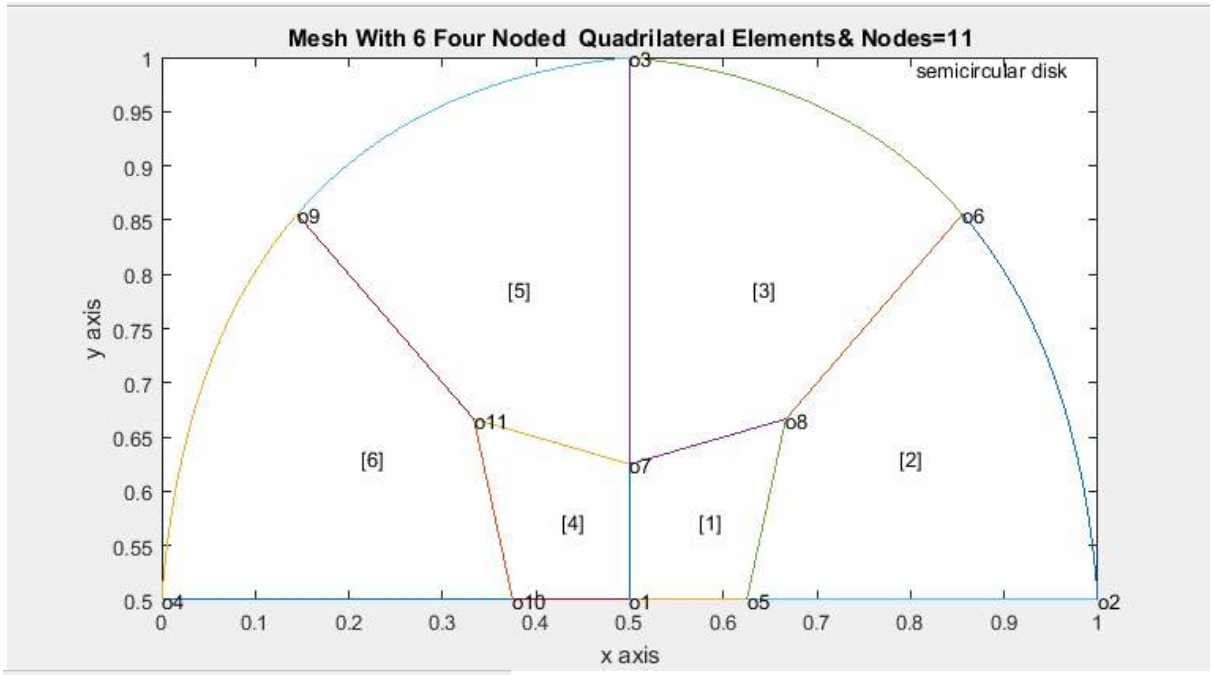
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,(n/2)^2,n,12,1,1)

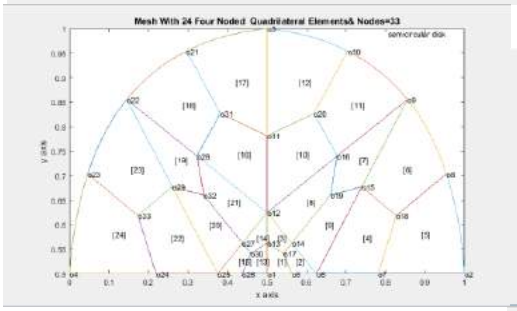
end

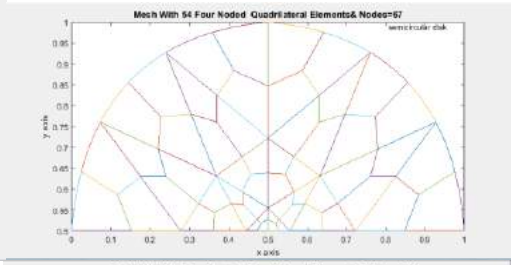
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,100,20,12,1,1)

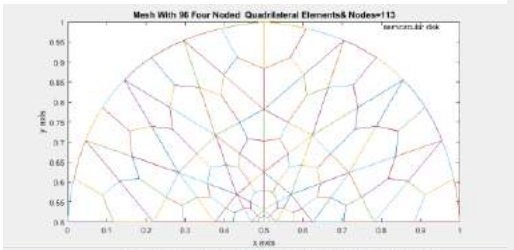
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,400,40,12,1,1)

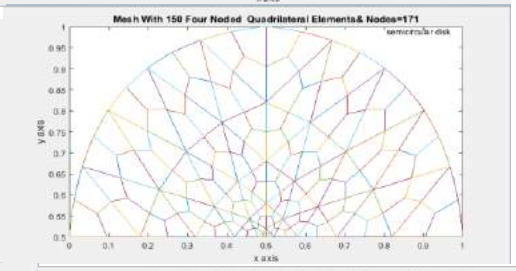
keyboard input:1 and then Y

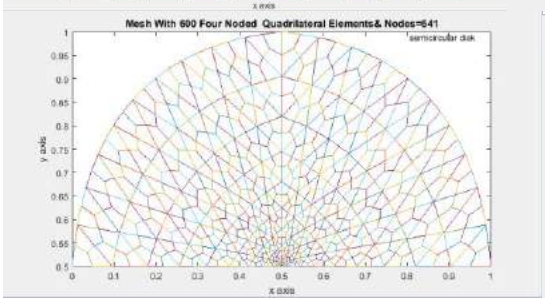


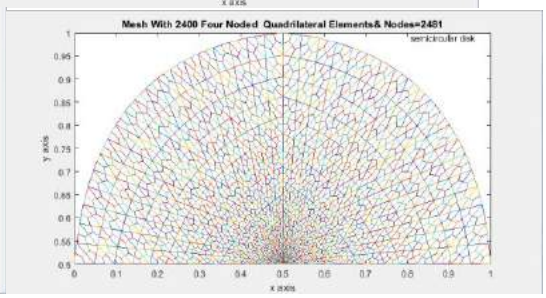






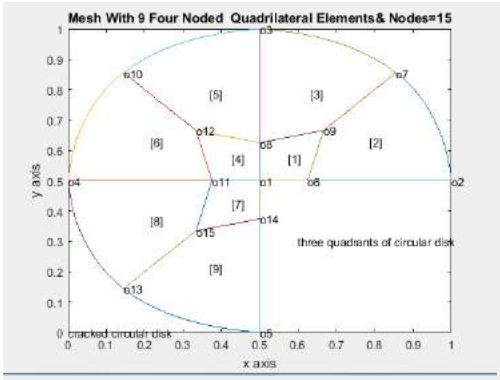


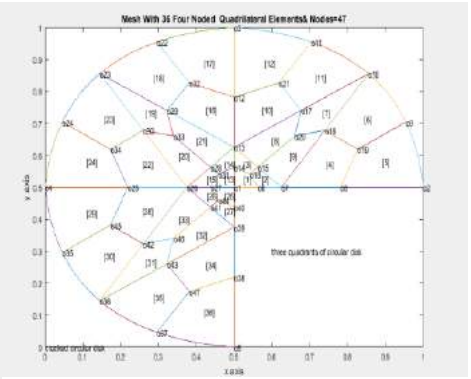


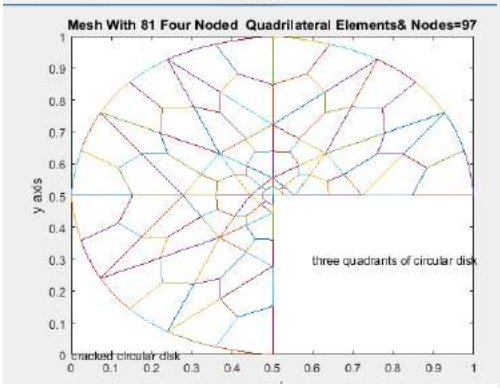


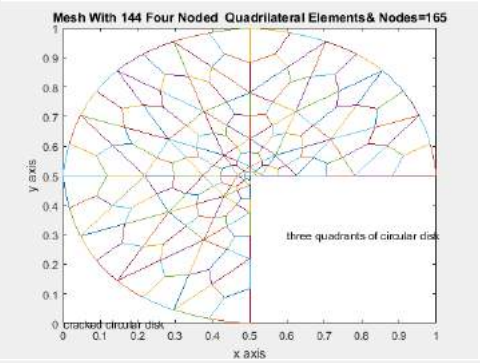
(3)THREE QUADRNTS OF CIRCULAR DISK

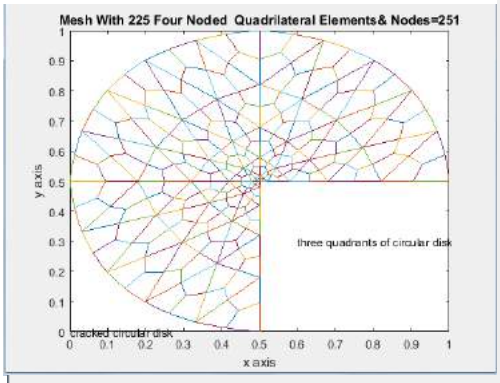
MATLAB COMMANDS
For ndiv=2:2:10
n=ndiv;
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,(n/2)^2,n,12,1,1)
end
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,100,20,12,1,1)
curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,400,40,12,1,1)
keyboard input:1 and then N

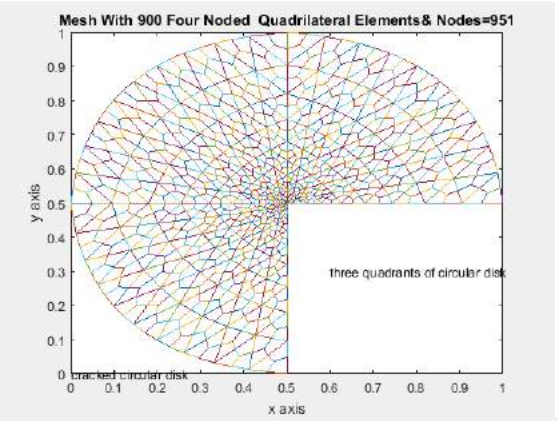

Mesh With 9 Four Noded Quadrilateral Elements& Nodes=15
three quadrants of circular disk
cracked circular disk
x axis
y axis

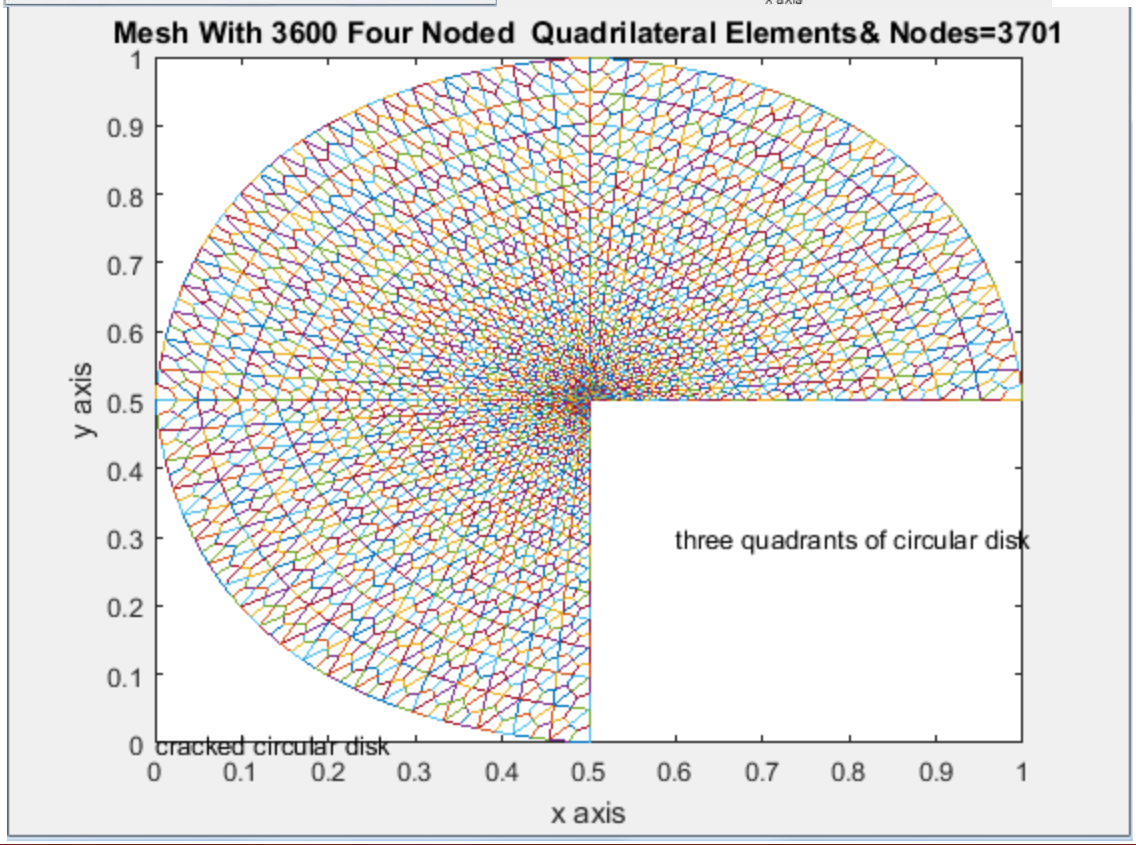

Mesh With 36 Four Noded Quadrilateral Elements& Nodes=47


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


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


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

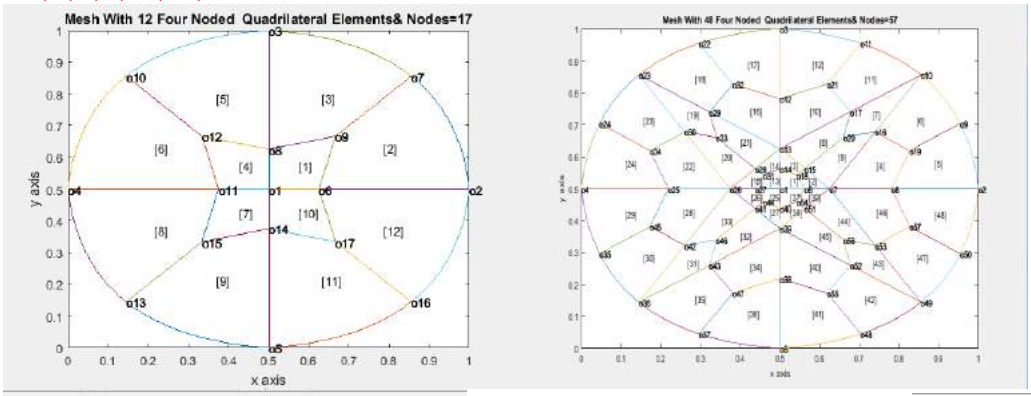

Mesh With 900 Four Noded Quadrilateral Elements& Nodes=951
three quadrants of circular disk
cracked circular disk
x axis
y axis

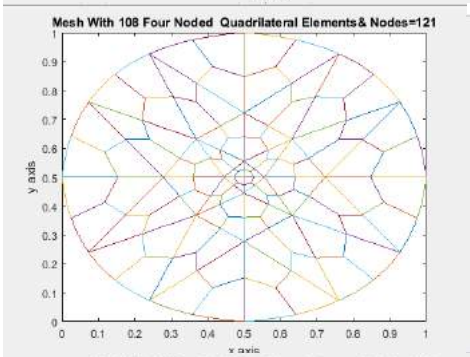

Mesh With 3600 Four Noded Quadrilateral Elements& Nodes=3701
three quadrants of circular disk
cracked circular disk
x axis
y axis

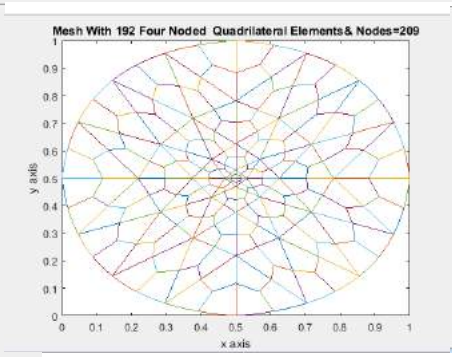
(4) CIRCULAR DISK

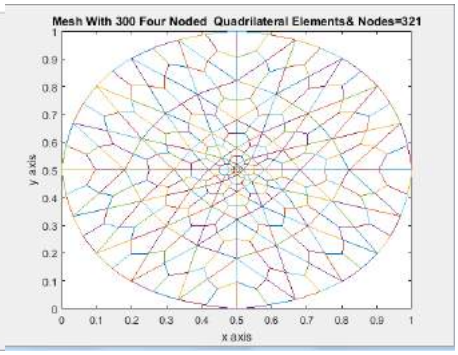
MATLAB COMMANDS

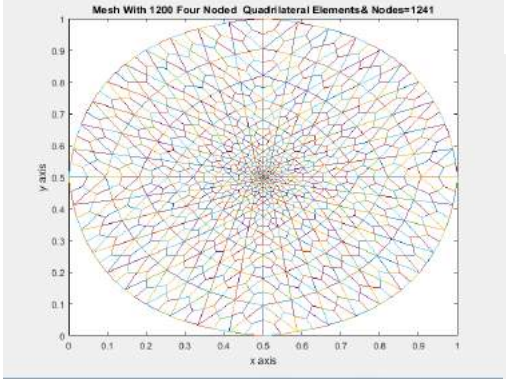
```
curvedquadrilateralmesh4convexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8;10;12],[7;9;11;13],5,(
n/2)^2,n,12,1,1)
n=2,4,6,8,10,20,40
```

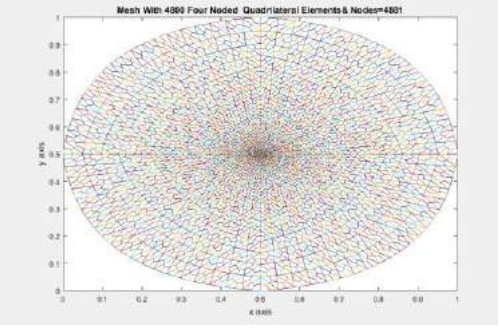












APPENDIX-C

PROGRAM-1

```
function[]=curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3(n1,n2,n3,n4,n5,nmax,n
umtri,ndiv,mesh,xlength,ylength)
%clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1
,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1
,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,2,1,1)
```

```
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4
,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,4,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;
2],7,1,2,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;
2],7,4,4,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;
6;7;8;2],8,1,2,8,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;
6;7;8;2],8,4,4,8,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;
4;5;6;7;8;1],9,1,2,9,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;
5;6;7;8;9;2],9,1,2,10,1,1)
%quadrilateralmesh4convexpolygoneightsidesq4([1],[2],[3],3,1,2,6,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([1],[2],[3],3,1,2,7,1,1)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrial([9;
9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,1,2)
%quadrilateral_mesh4convexpolygoneightsidesq4([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,11,1,
1)
%curvedquadrilateralmesh4convexpolygoneightsidesq4([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;1
0;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;25],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)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([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)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([3],[1],[2],[4],[5],3,1,2,15,1,1)
%**************************************************************************LATEST
AND UPDATED MATLAB COMMANDS****
%curvedquadrilateralmesh4convexpolygoneightsidesq40N([3],[1],[2],[4],[5],3,1,2,15,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3([3],[1],[2],[4],[5],3,1,2,15,1,
1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8
;10;12],[7;9;11;13],5,1,2,12,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40Ntr3([1;1;1;1],[2;3;4;5],[3;4;5;2],[
6;8;10;12],[7;9;11;13],5,1,2,12,1,1)
%**************************************************************************************
************************************
global edgen2n3
global TRINUM ncrack
global crnodes
ncrack=0
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 15

  axis([0 xlength 0 ylength])
 case 16

  axis([0 xlength 0 ylength])
   case 17

  axis([0 xlength 0 ylength])

end%mesh

[coord,gcoord,nodes,nodetel,tnodes,trielm,nnode,nel]=curvedpolygonal_domain_QUADcoordin
atesNtr3(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;
    nelm=nmax-2;
end
if (nmax==4)
    nitri=2;
    nelm=nmax-2;
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==15
  nelm=1
 %[mst_tri,theta,cord]=masterelementnodescoordinates_onecurvedtriangle(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [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==17
  nelm=2
  nitri=2
 %[mst_tri,theta,cord]=masterelementnodescoordinates_onecurvedtriangle(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [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
itri3=0;
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)
 %itri3=0;
for i=(itri-1)*numtri*3+1:itri*numtri*3
   % if rem(i,4)~=0
   itri3=itri3+1;
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)];
xt(itri3,1)=xvec(1,4);yt(itri3,1)=yvec(1,4);
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;
xlj=xl+(xj-xl)*s;
yij=yi+(yj-yi)*s;
yjk=yj+(yk-yj)*s;
ykl=yk+(yl-yk)*s;
yli=yl+(yi-yl)*s;
ylj=yl+(yj-yl)*s;
%plot(xij,yij)
%hold on
plot(xjk,yjk)
hold on
plot(xkl,ykl)
hold on
%plot(xli,yli)
%hold on
plot(xlj,ylj)
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
xlj=xl+(xj-xl)*s;ylj=yl+(yj-yl)*s;
plot(xlj,ylj)
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
xlj=xl+(xj-xl)*s;ylj=yl+(yj-yl)*s;
plot(xlj,ylj)
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 (rem(i,3)>=0)
if (ndiv<=6)
midx=mean(xvec(1,2:4))
midy=mean(yvec(1,2:4))
text(midx,midy,['[',num2str(itri3),']']);
end% if ndiv
end%if rem(i,3)>=0
 if rem(itri3+1,4)==0
 if (ndiv<=6)
     itri4=itri3+1
 midxt=(xt(itri3-2,1)+xt(itri3-1,1)+xt(itri3,1))/3
 midyt=(yt(itri3-2,1)+yt(itri3-1,1)+yt(itri3,1))/3
text(midxt,midyt,['[',num2str(itri4),']']);
end% if ndiv
itri3=itri3+1
end%if rem(itri3+1,4)==0

 end%i loop
```

```
% if rem(i,4)==0
%  if (ndiv<4)
% text(xi,yi,['[',num2str(i),']']);
% end% if ndiv
%end%if rem(i,4)==0
%itri3=itri4
end%itri loop

%%itri loop
xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(nitri*numtri*4);
st3=' curved &Linear ';
st4='Triangular';
st5=' Elements'
st6='& Nodes='
st7=num2str(nnode-numtri*nitri);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if (ndiv<=6)
    %number of six node triangles=numtri=number of centroids
for jj=1:nnode-numtri*nitri
text(xcoord(jj,1),ycoord(jj,1),['o',num2str(jj)]);
end
end%if ndiv
JJ=(1:nnode-numtri*nitri)';
%axis off
nitri
edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
[crnodes nodes]
[JJ xcoord(1:nnode-numtri*nitri) ycoord(1:nnode-numtri*nitri)]
JJJ=(1:nitri*numtri*4)';
[JJJ tnodes]
end%
%=========================================================================
PROGRAM-2
function[]=curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40Ntr3(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;1
0;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;25],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)
%curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40([5;5;5;5],[1;2;3;4],[2;3;4;1
],[6;8;10;12],[7;9;11;13],5,1,2,16,1,1)
global edgen2n3
global TRINUM ncrack nitri
global crnodes
figure(ndiv/2)
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])
%   case 17
%
%   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;
ncrack=input('ncrack=')
if ncrack==1%three quadrants
%[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(n1,n2,n
3,n4,n5,nmax,numtri,ndiv,mesh)
[coord,gcoord,nodes,nodetel,tnodes,trielm,nnode,nel]=curvedpolygonal_domain_QUADcoordin
atesNtr3(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh)

end
% if ncrack==2%two quadrants
% NN1(:,1)=n1(:,1);NN2(:,1)=n2(:,1);NN3(:,1)=n3(:,1);NN4(:,1)=n4(:,1);NN5(:,1)=n5(:,1);
% for iii=1:nmax-3
% N1(iii,1)=NN1(iii,1);
% N2(iii,1)=NN2(iii,1);
% N3(iii,1)=NN3(iii,1);
% N4(iii,1)=NN4(iii,1)+(-1);
```

```
% N5(iii,1)=NN5(iii,1)+(-1);
% end
% N1
% N2
% N3
% N4
% N5
% NMAX=nmax-1

%
[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(N1,N2,N3
,N4,N5,NMAX,numtri,ndiv,mesh)
% end
[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
%      if ncrack==2
%          nelm=nmax-3
%          nitri=2
%      end
[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
if (ncrack==1) & (mesh==12)
  reply=input('Do you have more cracks?Y/N [Y]:','s');
       if isempty(reply)
          reply = 'Y';
       end
    if reply=='Y'
    ncrack=2;
    nmax=4;
    end
   if reply=='N'
       ncrack=1;
    end
end

[nel,nnel]=size(nodes)
%number of special quadrilaterals in cracked circular disk
nel=nel-ncrack*(3*(ndiv/2)^2)
nitri=nitri-ncrack
%NITRI=nitri
spqd=nodes(1:nel,1:nnel)
nnode=max(max(spqd))
disp(['number of nodes and spqd elements in cracked circular disk=',num2str(nnode),' ,
', num2str(nel)])
%***********************************
% if (ncrack==2) & (mesh==12)
% NMAX=nmax;
% for iel=1:nel
%     for j=1:nnel
%     if nodes(iel,j)>(NMAX+1)
%         nodes(iel,j)=nodes(iel,j)-1;
%         COORD(NODES(iel,j),1)=coord(nodes(iel,j),1)
%          COORD(NODES(iel,j),2)=coord(nodes(iel,j),2)
%     end
%     end
% end
% spqd=NODES(1:nel,1:nnel)
% NNODE=max(max(spqd));
% if NNODE==(nnode-1)
% for iel=1:nel
%     for j=1:nnel
%          nodes(iel,j)=NODES(iel,j);
%         coord(NODES(iel,j),1)=COORD(NODES(iel,j),1);
%         coord(NODES(iel,j),2)=COORD(NODES(iel,j),2);
%
%     end
%  end
%
% else
```

```
%     [nnode NNODE]
% disp('error in reassigning nodal connectivity and coordinates')
% end%if NNODE
% nnode=NNODE
% end%if (ncrack==2) & (mesh==12)
%-------------------------------------------------------------------------
itri3=0;
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)
 %itri3=0;
for i=(itri-1)*numtri*3+1:itri*numtri*3
   % if rem(i,4)~=0
   itri3=itri3+1;
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)];
xt(itri3,1)=xvec(1,4);yt(itri3,1)=yvec(1,4);
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;
xlj=xl+(xj-xl)*s;
yij=yi+(yj-yi)*s;
yjk=yj+(yk-yj)*s;
ykl=yk+(yl-yk)*s;
yli=yl+(yi-yl)*s;
ylj=yl+(yj-yl)*s;
%plot(xij,yij)
%hold on
plot(xjk,yjk)
```

```
hold on
plot(xkl,ykl)
hold on
%plot(xli,yli)
%hold on
plot(xlj,ylj)
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
xlj=xl+(xj-xl)*s;ylj=yl+(yj-yl)*s;
plot(xlj,ylj)
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
xlj=xl+(xj-xl)*s;ylj=yl+(yj-yl)*s;
plot(xlj,ylj)
```

```
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 (rem(i,3)>=0)
if (ndiv<=4)
midx=mean(xvec(1,2:4))
midy=mean(yvec(1,2:4))
text(midx,midy,['[',num2str(itri3),']']);
end% if ndiv
end%if rem(i,3)>=0
 if rem(itri3+1,4)==0
 if (ndiv<=4)
     itri4=itri3+1
 midxt=(xt(itri3-2,1)+xt(itri3-1,1)+xt(itri3,1))/3
 midyt=(yt(itri3-2,1)+yt(itri3-1,1)+yt(itri3,1))/3
text(midxt,midyt,['[',num2str(itri4),']']);
end% if ndiv
itri3=itri3+1
end%if rem(itri3+1,4)==0


%-------------------------------------------------------------------------------
end%i loop
end%itri loop
if ncrack==1
TNODES=tnodes(1:nitri*numtri*4,1:3)
NNODE=max(max(TNODES))
%end%if crnodes
% if ncrack==2
%       nnode=nnode-1;
% end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(nitri*numtri*4);
st3=' curved &Linear ';
st4='Triangular';
st5=' Elements'
st6='& Nodes='
st7=num2str(NNODE);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if (ndiv<=4)
    %number of six node triangles=numtri=number of centroids
for jj=1:NNODE
text(xcoord(jj,1),ycoord(jj,1),['o',num2str(jj)]);
end
end%if ndiv
if( ncrack==1)
text(0.,0.,'cracked circular disk');

text(0.6,0.3,'three quadrants of circular disk');
end
JJ=(1:NNODE)'
%axis off

edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
```

```
[crnodes nodes]
[JJ xcoord(1:NNODE,1) ycoord(1:NNODE,1)]
JJJ=(1:nitri*numtri*4)';

numtri
nitri

[JJJ TNODES]

TRINUM

nitri
ncrack
nmax
end

TRINUM

nitri
ncrack
nmax

%*******************************************************
hold on
nnode=max(max(spqd(1:nel,2:nnel)))
%nnode=(n*(n+1)/2)*nitri+(n+1)
if( ncrack==2)&(nmax==4)&(mesh==12)

XCOORD(1:nmax,1)=xcoord(1:nmax,1);
YCOORD(1:nmax,1)=ycoord(1:nmax,1);
ii=nmax;
for jj=nmax+2:nnode
    ii=ii+1;
    XCOORD(ii,1)=xcoord(jj,1);
    YCOORD(ii,1)=ycoord(jj,1);
    [ii jj]
end
NNODE=ii
nnode
for pp=1:nel
  for qq=1:nnel
    if spqd(pp,qq)<(nmax+1)
       SPQD(pp,qq)= spqd(pp,qq);
    end

    if spqd(pp,qq)>(nmax+1)
       SPQD(pp,qq)= spqd(pp,qq)-1;
    end
  end
end
%***********************************
%put node numbers
NNODE=max(max(SPQD(1:nel,2:nnel)))
%if ncrack~=2
if (ndiv<=4)

for jj=1:NNODE
text(XCOORD(jj,1),YCOORD(jj,1),['o',num2str(jj)]);
end
end%if ndiv
%end%if ncrack~=2

JJ=(1:NNODE)';
%axis off
```

```
nitri
edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
[crnodes]
[SPQD]
[JJ XCOORD(1:NNODE,1) YCOORD(1:NNODE,1)]
%*********************************************************
xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(nitri*numtri*4);
st3=' curved &Linear ';
st4='Triangular';
st5=' Elements'
st6='& Nodes='
st7=num2str(NNODE);
title([st1,st2,st3,st4,st5,st6,st7])
TRINUM

nitri
ncrack
nmax


if( ncrack==2)&(mesh==12)
text(0.,0.,'cracked circular disk');
end

if( ncrack==2)&(mesh==12)
text(0.8,0.99,' semicircular disk');
end
end%if( ncrack==2)&(nmax==4)&(mesh==12)




%=========================================================================
PROGRAM-3
function[coord,gcoord,nodes,nodetel,tnodes,trielm,nnode,nel]=curvedpolygonal_domain_QUA
DcoordinatesNtr3(n1,n2,n3,n4,n5,nmax,numtri,n,mesh)
%note that %[coord,gcoord,nodes,nodetel,nnode,nel]=
%[coord,gcoord,nodes,nodetel,tnodes,trielm,nnode,nel]=curvedpolygonal_domain_QUADcoordi
natesNtr3(n1,n2,n3,n4,n5,nmax,numtri,ndiv,mesh)
%n3=node number at(0,0)for  a choosen triangle
%n1=node number at(1,0)for  a choosen triangle
%n2=node number at(0,1)for  a choosen triangle
%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
%joining the end poits of the segment to the interior point(e.g:the centroid) of the
polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=
2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,9,6)
```

```
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
global edgen2n3
global TRINUM ncrack nitri
global crnodes
%
syms U V W

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


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

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

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

x=sym([0;1;1/2])
y=sym([0;0;sqrt(3)/2])
case 8%for a convexpolygonsixside
  %    1    2   3  4   5  6  7  8
x=sym([0.5;0.1;0.7;1;.75;.5;.25;0])
y=sym([0.5;0;0.2;.5;.85;1;.625;.25])
case 9
%    1  2   3  4   5    6  7    8   9
x=([.2;0.5;.8;1.0;.75;0.5;0.25;0.0;0.5])
y=([.2;.05;.2;0.5;.85;1.0;0.90;0.6;0.5])
case 10
%    1  2   3  4   5    6  7    8   9
x=([0.5;.2;0.5;.8;1.0;.75;0.5;0.25;0.0])
y=([0.5;.2;.05;.2;0.5;.85;1.0;0.90;0.6])
case 11%lunar model
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
 if ncrack==2
     nelm=nmax-2
 end
[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 13
 nelm=nmax-1
 [mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeyway(nelm)
 [cord]=globalnodalcoordinate_circularshaft_keyway(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]
case 17
  nelm=2
 [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,tnodes,trielm]=nodaladdresses4special_convex_quadrilaterals
QUADtrialNtr3(n1,n2,n3,nmax,numtri,n)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrialN(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_quadrilateralsN(n);

%[eln,spqd,rrr,nodes,nodetel,tnodes,trielm]=nodaladdresses_special_convex_quadrilateral
sN(n)

[eln,spqd,rrr,nodes,nodetel,tnodes,trielm]=nodaladdresses_special_convex_quadrilaterals
Ntr3(n)
    rrr
   edgen2n3=edgen2n3(1:n+1,1);
end
%[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
[U,V,W] =triangularmeshpoints4singularcorner(n,2);
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;%for closed domain

if (nmax==3)
    nitri=1;
end
if (nmax==4)
    nitri=2;
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,tnodes,trielm]=nodaladdresses_special_convex_quadri
lateralsNtr3(n)
%function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_tria
l(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


mmm=0;
for iel=1:ne
    for jel=1:4
    %mm=mm+1;mmm=mmm+1;
       switch jel
       case 1
          % mm=mm+1;
           mmm=mmm+1;
          trielm(mmm,1:3)=[eln(iel,1) eln(iel,4) eln(iel,6)];
          tnodes(mmm,1:3)=trielm(mmm,1:3);
        % nodetel(mm,1:3)=[eln(iel,1) eln(iel,2) eln(iel,3)];
```

```
           case 2
           %mm=mm+1;
           mmm=mmm+1;
           trielm(mmm,1:3)=[eln(iel,2) eln(iel,5) eln(iel,4)];
           tnodes(mmm,1:3)=trielm(mmm,1:3);
           %nodetel(mm,1:3)=[eln(iel,2) eln(iel,3) eln(iel,1)];
           case 3
          % mm=mm+1;
           mmm=mmm+1;
           trielm(mmm,1:3)=[eln(iel,3) eln(iel,6) eln(iel,5)];
           tnodes(mmm,1:3)=trielm(mmm,1:3);
           %nodetel(mm,1:3)=[eln(iel,3) eln(iel,1) eln(iel,2)];
            case 4
            mmm=mmm+1;
            trielm(mmm,1:3)=[eln(iel,4) eln(iel,5) eln(iel,6)];
            tnodes(mmm,1:3)=trielm(mmm,1:3);


       end%switch
     end
   end

eln
trielm
tnodes


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

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



%to generate special quadrilaterals
mm=0;

for iel=1:ne
    for jel=1:3
    mm=mm+1;
        switch jel
         case 1
          spqd(mm,1:4)=[eln(iel,7) eln(iel,6) eln(iel,1) eln(iel,4)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,2) eln(iel,3) eln(iel,1)];
          case 2
          spqd(mm,1:4)=[eln(iel,7) eln(iel,4) eln(iel,2) eln(iel,5)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,3) eln(iel,1) eln(iel,2)];
          case 3
          spqd(mm,1:4)=[eln(iel,7) eln(iel,5) eln(iel,3) eln(iel,6)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,1) eln(iel,2) eln(iel,3)];
```

```
        end%switch
      end
    end
```

```
%=========================================================================
```

PROGRAM-5

```
function[eln,spqd,rrr,nodes,nodetel,tnodes,trielm]=nodaladdresses4special_convex_quadri
lateralsQUADtrialNtr3(n1,n2,n3,nmax,numtri,n)
%[eln,spqd,rrr,nodes,nodetel,tnodes,trielm]=nodaladdresses4special_convex_quadrilateral
sQUADtrialNtr3(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;
if (nmax==4)
    nitri=2;
end
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))
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd


end%itri

%*************************************
mmm=0;
for iel=1:ne
```

```
    for jel=1:4
    %mm=mm+1;mmm=mmm+1;
       switch jel
       case 1
          % mm=mm+1;
           mmm=mmm+1;
          trielm(mmm,1:3)=[eln(iel,1) eln(iel,4) eln(iel,6)];
          tnodes(mmm,1:3)=trielm(mmm,1:3);
         % nodetel(mm,1:3)=[eln(iel,1) eln(iel,2) eln(iel,3)];
          case 2
          %mm=mm+1;
          mmm=mmm+1;
          trielm(mmm,1:3)=[eln(iel,2) eln(iel,5) eln(iel,4)];
          tnodes(mmm,1:3)=trielm(mmm,1:3);
          %nodetel(mm,1:3)=[eln(iel,2) eln(iel,3) eln(iel,1)];
          case 3
         % mm=mm+1;
          mmm=mmm+1;
          trielm(mmm,1:3)=[eln(iel,3) eln(iel,6) eln(iel,5)];
          tnodes(mmm,1:3)=trielm(mmm,1:3);
          %nodetel(mm,1:3)=[eln(iel,3) eln(iel,1) eln(iel,2)];
           case 4
           mmm=mmm+1;
           trielm(mmm,1:3)=[eln(iel,4) eln(iel,5) eln(iel,6)];
           tnodes(mmm,1:3)=trielm(mmm,1:3);


       end%switch
     end
   end

eln
trielm
tnodes


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

%to generate special quadrilaterals
mm=0;

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

```
%for mmm=1:mm
    %spqd(:,1:4)
 %end
%
ss1='number of 6-node triangles with centroid=';
[p1,q1]=size(eln);
%=================================================================
```

PROGRAM-6

```
function [U,V,W] =triangularmeshpoints4singularcorner(p,q)
%draw outline for the triangle
%triangularmeshpoints4singularcorner(7,2)
%triangularmeshpoints4singularcorner(7,1/2)
clc
syms ui vi wi U V W
syms UI VI WI UU VV WW
axis square
% x=[1;0;0;1]
% y=[0;1;0;0]
axis([0 1 0 1])
kk=0;
for j=0:p
mn=j;
if j==0
jpq=0;
end
if j>0
jpq=(j/p)^q;
end
  for k=0:j
      kk=kk+1;
    %mn=m+n
    m=k;n=mn-m;
     if ((m+n)==0)
    ui(m+1,n+1)=0;
    vi(m+1,n+1)=0;
     wi(m+1,n+1)=1;
    end

    if ((m+n)>0)
    ui(m+1,n+1)=n*jpq/(m+n);
    vi(m+1,n+1)=m*jpq/(m+n);
    wi(m+1,n+1)=1-ui(m+1,n+1)-vi(m+1,n+1);
    end
%     U(kk,1)=ui(m+1,n+1);
%         V(kk,1)=vi(m+1,n+1);


   end%for k
end%for j

       % figure(1),scatter(U(:,1),V(:,1),10,'filled','r')
%   ui
%   vi
% [U(:,1) V(:,1)]
% [ui(1,2) vi(1,2)]
% [ui(2,1) vi(2,1)]
UI=ui';VI=vi';WI=wi';
kk=0;n=p;
for j=1:n+1
    for i=1:(n+1)-(j-1)
        kk=kk+1;
```

```
        U(kk,1)=UI(i,j);
        V(kk,1)=VI(i,j);
        W(kk,1)=WI(i,j);

    end

end
uii= vpa(ui,5);
 vii=vpa(vi,5);
 wii=vpa(wi,5);
% disp('-----------------------------------------------')
 UII=vpa(UI,5);
 VII=vpa(VI,5);
 WII=vpa(WI,5);
  kku=length(U)
  kkv=length(V)
   format short
%   disp('-----------------------------------------------')
 %figure(1),scatter(ui(:,:),vi(:,:),10,'filled','r')
 %display the mesh points,number of divisions and grading factor
 %to display the mesh points set display=1
 display=0
 if display==1
figure(p/2),scatter(U(:,1),V(:,1),10,'filled','r')
xlabel('x axis')
ylabel('y axis')
st1='mesh divisions = ';
st2=num2str(p);
st3='; &';
st4='grading factor=';
st5=num2str(q);
st6='& number of mesh points='
st7=num2str(kku);
title([st1,st2,st3,st4,st5,st6,st7])
 end
%================================================================================
```

PROGRAM-7

```
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)
if nelm>1
n=nelm;N=n;
for i=1:n
    N=N+2;
    if nelm==2
        N=N+3
    end
   % mst_tri(i,1)=n+1;
   mst_tri(i,1)=1;
   % mst_tri(i,2)=i;
   mst_tri(i,2)=i+1;
   % mst_tri(i,3)=i+1;
   mst_tri(i,3)=i+2;
   mst_tri(i,4)=N;
    mst_tri(i,5)=N+1;
end
%mst_tri(n,3)=1;
mst_tri(n,3)=2;
%mst_tri
```

```
syms pi theta
N=n/2;
if nelm==2
    N=2*n
end
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+1,1)=.5+.5*cos(t1);
     coord(i+1,2)=.5+.5*sin(t1);
     if (i+2)<(n+2)
     coord(i+2,1)=.5+.5*cos(t2);
     coord(i+2,2)=.5+.5*sin(t2);
     end
     if (i+2)==(n+2)
         coord(1,1)=1/2;coord(1,2)=1/2;
     end
end

% coord(n+1,1)=1/2;coord(n+1,2)=1/2;
%coord
coord=double(coord)
%end%nelm>3rem(nelm,4)==0
end% if nelm>1
 %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-8

```
 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

%=====================================================================
PROGRAM-9
function[]=curvedquadrilateralmesh4convexpolygoneightsidesq40N(n1,n2,n3,n4,n5,nmax,numt
ri,ndiv,mesh,xlength,ylength)
%clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1
,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1,1
,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4
,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,4,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;
2],7,1,2,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;
2],7,4,4,5,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;
6;7;8;2],8,1,2,8,1,1)
%quadrilateral_mesh4convexpolygonsixside_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;
6;7;8;2],8,4,4,8,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;
4;5;6;7;8;1],9,1,2,9,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;
5;6;7;8;9;2],9,1,2,10,1,1)
%quadrilateralmesh4convexpolygoneightsidesq4([1],[2],[3],3,1,2,6,1,1)
%quadrilateral_mesh4convexpolygoneightsidesq4([1],[2],[3],3,1,2,7,1,1)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrial([9;
9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,1,2)
```

```
%quadrilateral_mesh4convexpolygoneightsidesq4([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,11,1,
1)
%curvedquadrilateralmesh4convexpolygoneightsidesq4([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;1
0;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;25],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)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([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)
%curvedquadrilateralmesh4convexpolygoneightsidesq40([3],[1],[2],[4],[5],3,1,2,15,1,1)
%**************************************************************************LATEST
AND UPDATED MATLAB COMMANDS****
%curvedquadrilateralmesh4convexpolygoneightsidesq40N([3],[1],[2],[4],[5],3,1,2,15,1,1)
%curvedquadrilateralmesh4convexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],[6;8
;10;12],[7;9;11;13],5,1,2,12,1,1)
%n=2;curvedquadrilateralmesh4convexpolygoneightsidesq40N([1;1;1;1],[2;3;4;5],[3;4;5;2],
[6;8;10;12],[7;9;11;13],5,(n/2)^2,n,12,1,1)
%**************************************************************************************
************************************
global edgen2n3
global TRINUM ncrack
global crnodes
ncrack=0
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 15

  axis([0 xlength 0 ylength])
 case 16
```

```
  axis([0 xlength 0 ylength])
   case 17

  axis([0 xlength 0 ylength])

end%mesh

[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(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;
    nelm=nmax-2;
end
if (nmax==4)
    nitri=2;
    nelm=nmax-2;
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==15
  nelm=1
 %[mst_tri,theta,cord]=masterelementnodescoordinates_onecurvedtriangle(nelm)
```

```
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [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==17
  nelm=2
  nitri=2
 %[mst_tri,theta,cord]=masterelementnodescoordinates_onecurvedtriangle(nelm)
 %[cord]=globalnodalcoordinate_onecurvedtriangle(mst_tri,theta,cord)
 [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

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


xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(nel);
st3=' Four Noded  ';
st4='Quadrilateral';
st5=' Elements'
st6='& Nodes='
st7=num2str(nnode);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if (ndiv<=4)
for jj=1:nnode
text(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 ycoord]
end%itri loop

end%if crnodes
%=================================================================
```

PROGRAM-10

```
function[]=curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40N(n1,n2,n3,n4,n5,nm
ax,numtri,ndiv,mesh,xlength,ylength)

%curvedquadrilateralmesh4convexpolygoneightsidesq4([5;5;5;5],[1;2;3;4],[2;3;4;1],[6;8;1
0;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;25],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)
%curvedquadrilateralmesh4crackedconvexpolygoneightsidesq40([5;5;5;5],[1;2;3;4],[2;3;4;1
],[6;8;10;12],[7;9;11;13],5,1,2,16,1,1)
global edgen2n3
global TRINUM ncrack nitri
global crnodes
```

```
figure(ndiv/2)
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])
%   case 17
%
%   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;
% % ncrack=input('ncrack=')
if ncrack==1%three quadrants
[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(n1,n2,n3
,n4,n5,nmax,numtri,ndiv,mesh)
end
% % if (ncrack==2) & (mesh==12)
% %     ncrack=1
% %
[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(n1,n2,n3
,n4,n5,nmax,numtri,ndiv,mesh)
```

```
% % ncrack=2
% % nmax=4;
% % end
% if ncrack==2%two quadrants
% NN1(:,1)=n1(:,1);NN2(:,1)=n2(:,1);NN3(:,1)=n3(:,1);NN4(:,1)=n4(:,1);NN5(:,1)=n5(:,1);
% for iii=1:nmax-3
% N1(iii,1)=NN1(iii,1);
% N2(iii,1)=NN2(iii,1);
% N3(iii,1)=NN3(iii,1);
% N4(iii,1)=NN4(iii,1)+(-1);
% N5(iii,1)=NN5(iii,1)+(-1);
% end
% N1
% N2
% N3
% N4
% N5
% NMAX=nmax-1

%
[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(N1,N2,N3
,N4,N5,NMAX,numtri,ndiv,mesh)
% end
[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
%      if ncrack==2
%          nelm=nmax-3
%          nitri=2
%      end
[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
if (ncrack==1) & (mesh==12)
  reply=input('Do you have more cracks?Y/N [Y]:','s');
       if isempty(reply)
          reply = 'Y';
       end
    if reply=='Y'
    ncrack=2;
    nmax=4;
    end
   if reply=='N'
       ncrack=1;
    end
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))
%***********************************
% if (ncrack==2) & (mesh==12)
% NMAX=nmax;
% for iel=1:nel
%     for j=1:nnel
%     if nodes(iel,j)>(NMAX+1)
%         nodes(iel,j)=nodes(iel,j)-1;
%         COORD(NODES(iel,j),1)=coord(nodes(iel,j),1)
%          COORD(NODES(iel,j),2)=coord(nodes(iel,j),2)
%     end
%     end
% end
% spqd=NODES(1:nel,1:nnel)
% NNODE=max(max(spqd));
% if NNODE==(nnode-1)
% for iel=1:nel
%     for j=1:nnel
%          nodes(iel,j)=NODES(iel,j);
```

```
%         coord(NODES(iel,j),1)=COORD(NODES(iel,j),1);
%         coord(NODES(iel,j),2)=COORD(NODES(iel,j),2);
%
%     end
%  end
%
% else
%     [nnode NNODE]
% disp('error in reassigning nodal connectivity and coordinates')
% end%if NNODE
% nnode=NNODE
% end%if (ncrack==2) & (mesh==12)
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 ncrack~=2
if (ndiv<=4)
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
```

```
text(midx,midy,['[',num2str(i),']']);
end% if ndiv
%end%if ncrack~=2
end%i loop
end%itri loop
%end%if crnodes
% if ncrack==2
%      nnode=nnode-1;
% end

xlabel('x axis')
ylabel('y axis')
st1='Mesh With ';
st2=num2str(nel);
st3=' Four Noded  ';
st4='Quadrilateral';
st5=' Elements'
st6='& Nodes='
st7=num2str(nnode);
if ncrack==2
  st7=num2str(nnode-1);
end
title([st1,st2,st3,st4,st5,st6,st7])

if( ncrack==1)
text(0.,0.,'cracked circular disk');

text(0.6,0.3,'three quadrants of circular disk');
end
%put node numbers
if ncrack~=2
if (ndiv<=4)

for jj=1:nnode
text(xcoord(jj,1),ycoord(jj,1),['o',num2str(jj)]);
end
end%if ndiv
end%if ncrack~=2

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
nmax
%*******************************************************
hold on
if( ncrack==2)&(nmax==4)&(mesh==12)

XCOORD(1:nmax,1)=xcoord(1:nmax,1);
YCOORD(1:nmax,1)=ycoord(1:nmax,1);
ii=nmax;
for jj=nmax+2:nnode
    ii=ii+1;
    XCOORD(ii,1)=xcoord(jj,1);
    YCOORD(ii,1)=ycoord(jj,1);
    [ii jj]
end
```

```
NNODE=ii
nnode
for pp=1:nel
  for qq=1:nnel
    if spqd(pp,qq)<(nmax+1)
       SPQD(pp,qq)= spqd(pp,qq);
    end

    if spqd(pp,qq)>(nmax+1)
       SPQD(pp,qq)= spqd(pp,qq)-1;
    end
  end
end
%************************************
%put node numbers
%if ncrack~=2
if (ndiv<=4)

for jj=1:NNODE
text(XCOORD(jj,1),YCOORD(jj,1),['o',num2str(jj)]);
end
end%if ndiv
%end%if ncrack~=2

JJ=(1:NNODE)';
%axis off
nitri
edgen2n3=edgen2n3(1:ndiv+1,1:nitri)
[crnodes]
[SPQD]
[JJ XCOORD(1:NNODE,1) YCOORD(1:NNODE,1)]
%************************************************************
TRINUM

nitri
ncrack
nmax

% if ncrack==2
% 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])
% end
if( ncrack==2)&(mesh==12)
text(0.,0.,'cracked circular disk');
end

if( ncrack==2)&(mesh==12)
text(0.8,0.99,' semicircular disk');
end
end%if( ncrack==2)&(nmax==4)&(mesh==12)
%=================================================================
PROGRAM-11
function[coord,gcoord,nodes,nodetel,nnode,nel]=curvedpolygonal_domain_QUADcoordinatesN(
n1,n2,n3,n4,n5,nmax,numtri,n,mesh)
%note that %[coord,gcoord,nodes,nodetel,nnode,nel]=
%n3=node number at(0,0)for  a choosen triangle
%n1=node number at(1,0)for  a choosen triangle
```

```
%n2=node number at(0,1)for  a choosen triangle
%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
%joining the end poits of the segment to the interior point(e.g:the centroid) of the
polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=
2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,9,6)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
global edgen2n3
global TRINUM ncrack nitri
global crnodes
%
syms U V W

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


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

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

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

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

```
x=([.2;0.5;.8;1.0;.75;0.5;0.25;0.0;0.5])
y=([.2;.05;.2;0.5;.85;1.0;0.90;0.6;0.5])
case 10
%    1  2   3  4   5    6  7     8   9
x=([0.5;.2;0.5;.8;1.0;.75;0.5;0.25;0.0])
y=([0.5;.2;.05;.2;0.5;.85;1.0;0.90;0.6])
case 11%lunar model
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
 if ncrack==2
     nelm=nmax-2
 end
[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 13
 nelm=nmax-1
 [mst_tri,theta,cord]=masterelementnodescoordinates_circularshaftkeyway(nelm)
 [cord]=globalnodalcoordinate_circularshaft_keyway(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]
case 17
  nelm=2
 [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_quadrilateralsQUADtrialN(n1,
n2,n3,nmax,numtri,n)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtrialNCNT
RDS(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_quadrilateralsN(n);

%[eln,spqd,rrr,nodes,nodetel,tnodes,trielm]=nodaladdresses_special_convex_quadrilateral
sN(n)
    rrr
   edgen2n3=edgen2n3(1:n+1,1);
end
%[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
[U,V,W] =triangularmeshpoints4singularcorner(n,2);
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;%for closed domain
```

```
if (nmax==3)
    nitri=1;
end
if (nmax==4)
    nitri=2;
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-12

```
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilateralsN(n)
%function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_tria
l(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




eln
```

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

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


%to generate special quadrilaterals
mm=0;

for iel=1:ne
    for jel=1:3
    mm=mm+1;
        switch jel
         case 1
          spqd(mm,1:4)=[eln(iel,7) eln(iel,6) eln(iel,1) eln(iel,4)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,2) eln(iel,3) eln(iel,1)];
          case 2
          spqd(mm,1:4)=[eln(iel,7) eln(iel,4) eln(iel,2) eln(iel,5)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,3) eln(iel,1) eln(iel,2)];
          case 3
          spqd(mm,1:4)=[eln(iel,7) eln(iel,5) eln(iel,3) eln(iel,6)];
          nodes(mm,1:4)=spqd(mm,1:4);
          nodetel(mm,1:3)=[eln(iel,1) eln(iel,2) eln(iel,3)];




        end%switch
     end
   end

 %================================================================
Program-13
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilateralsQUADtr
ialN(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;
if (nmax==4)
    nitri=2;
```

```
end
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
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);
```