www.ijecs.in

International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 10 Oct. 2016, Page No. 18413-18427

Finite Element Analysis Of Linear Elastic Torsion For Regular Polygons

H.T. Rathod[a*], *K. Sugantha Devi*[b], *C.S.Nagabhushana*[c], *H.M.Chudamani*[d]

[a] Department of Mathematics, Central College Campus,
Bangalore University, Bangalore- 560001
E-mail: htrathod2010@gmail.com
[b] Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post,
Kolar Gold Field, Kolar District, Karnataka state, Pin- 563120, India.
Email: suganthadevik@yahoo.co.in
c [c] Department of Mathematics, HKBK College of Engineering, Nagavara,
Bangalore – 560045, India.
E-mail: csnagabhushana@gmail.com
[d] Department of Mathematics, A.S.C College ,Industrial Estate,1st Main,
Rajaji Nagar Bangalore – 560010, India.
E-mail: savitha.hmc@gmail.com

Abstract :

This paper presents an explicit finite element integration scheme to compute the stiffness matrices for linear convex quadrilaterals. Finite element formulationals express stiffness matrices as double integrals of the products of global derivatives. These integrals can be shown to depend on triple products of the geometric properties matrix and the matrix of integrals containing the rational functions with polynomial numerators and linear denominator in bivariates as integrands over a 2-square. These integrals are computed explicitely by using symbolic mathematics capabilities of MATLAB. The proposed explicit finite element integration scheme can be applied to solve boundary value problems in continuum mechanics over convex polygonal domains.We have also developed an automatic all quadrilateral mesh generation technique for convex polygonal domain which provides the nodal coordinates and element connectivity.We have demonstrated the proposed explicit integration scheme to solve the Poisson Boundary Value Problem for a linear elastic torsion of a non-circular bar with cross sections having profiles of equilateral triangle, a square and regular polygons (pentagon(5-gon)to icosagon(20-gon)) which are inscribed in a circle of unit radius. Monotonic convergence from below is observed with known analytical solutions for the Prandtl stress function and torsional constant .We have shown the solutions in Tables which list both the FEM and exact solutions. The graphical solutions of contour level curves and the corresponding finite element meshes are also displayed.

Key words: Explicit Integration, Finite Element Method, Matlab Symbolic Mathematics, All Quadrilateral Mesh Generation Technique,Poisson Equation,Dirichlet Boundary Conditions ,Polygonal Domain ,Torsion of Noncircular Cross Sections,Regular Polygons(5-gon to 20-gon)

1. INTRODUCTION

The general theory of torsion is very well established in literature and remains a classic in the field of solid mechanics and elasticity. Most often, problems concerning torsion, particularly that of prismatic bars, have great variety of engineering applications which range from analyzing stresses to designing of machine members and structures. Mostly, established solutions to torsion problems are for most cases given analytically. Only in some applications where difficulties in obtaining exact solutions may arise, numerical methods of solutions are sought. These difficulties are often a result of complexities in geometry or perhaps, the boundary conditions of the governing partial differential equation of the machine member or structure being analyzed. The most common of torsion problems encountered in engineering is that of circular-section bars. Only less familiar are those of triangular, elliptical, and rectangular sections.

Analysis of properties, states and behavior of technical objects is an important task of Engineering Mechanics. Strain-stress analysis of solid bodies ranks to these problems Continuum mechanics and especially elasticity theory provides tools for this analysis. The strain-stress analysis passed through its development, analytic approaches predominating in the past are at present replaced by numerical tools as Finite Element Method, Finite Volume Method, Boundary Element Method etc. In comparison to the classical methods the numerical methods are universal in sense that their applicability is independent of geometry of the body, material characteristics, etc. This may indicate that the period of analytical elasticity ended. But it is not the case. There are reasons to keep on using both the analytical methods and the numerical methods in the strain-stress analysis. The numerical methods (e.g. FEM) enable to compute numerical values of strain-stress state of the particular material in particular points under particular loads. On the other hand they provide no formulas which can be used for predicting the change of the values under changing the load, size, stiffness etc. The analytical methods yielding formulas enable this prediction. Unfortunately they can be applied only to special shapes of bodies and loads. Also interpreting results obtained by numerical computations requires basic knowledge of mathematical elasticity, which causes troubles to many engineers in practice and leads to doubts whether the results obtained by numerical methods are correct. Analytical theory of elasticity can be a tool for verification of the numerical results. This knowledge, hidden in formulas obtained by analytical methods, is a suitable tool for finding critical areas of loaded bodies.

The torsional theory of circular sections cannot be applied to the torsion of noncircular sections, as the shear stresses for non-circular sections are no longer circumferential. Furthermore, plane cross-sections do not remain plane and undistorted on the application of torque, and in fact, warping of the cross-section takes place. Torsion of the elastic bars is studied in several textbooks, but the results are mostly introduced without proofs or circular cross-section only is considered. In this circular case the cross-sections remains planar, but in case of non-circular bar, the real cross-sections are deflected from the planar shape.

Structural elements with very different cross sectional shapes are widely used in various engineering structures.The exact solutions for torsion have been found for some simple cross-sectional shapes such as circles, ellipses, and triangles. However, in the theory of elasticity, it is difficult to obtain analytical solutions for complicated cross-sections. To solve general cross-sectional problems, numerical methods are usually necessary. For more complicated shapes, numerical methods are usually employed. This study will then perform a numerical calculation to determine the stress distribution of cylindrical torsion beams of arbitrary non-circular section. Indeed, the complexity of this section induces that the exact solution does not exist, and of course our interest is directed toward the search for approximate numerical solutions. The finite element method was introduced and applied successfully since it adapts well to any selected section

Generally, in torsion problems, two methods of analyses or approaches are known: (i) the approach first introduced by Saint-Venant which uses displacement components and associated warping function; and (ii) the approach first introduced by Prandtl which uses the concept of membrane analogy and associated stress function. These approaches lead to solving torsion problems in the form of partial differential equations of either Laplace or Poisson type. In particular, Saint-Venant's approach yields Laplace's equation while Prandtl's approach yields Poisson's equation. In other words, torsion problems are generally boundary value problems (BVPs) which can be solved either analytically or numerically. Analytical methods of solutions are only practical when the cross-section of the prismatic bar being analyzed is regular, otherwise numerical solutions must be sought. In the case of rectangular sections, the method of Prandtl is common such as the one used in [22-23], because of its simplicity in boundary condition, i.e. of Dirichlet type.

2.0 Formulation of the Saint-Venant Torsion Theory:

There are two ways in which the torsion problem can be formulated. The first is by way of a warping function and the second is by way of a stress function. Both formulations are clearly explained and illustrated in references[22-23]. We focus on the stress function formulation since it easier to implement. We give a brief outline of the formulation, leaving the details to be looked up in references [22-23]. We assume that the prismatic bar is parallel to the z axis with x,y coordinates lying in the nominal cross section of the undeformed bar. We also assume that there are equal and opposite torques, directed along the z axis, applied to the two ends of the bar. The Saint-Venant Principle allows us to assume that we need not be concerned with how these torques are applied as long as they are statically equivalent to the torque produced by the stresses in cross section planes of the bar. The approach is that there exists a stress function $\phi(x, y)$ from which the *nonzero* stresses are computed by the formulas Type equation here.

$$\sigma_{xz} = \frac{\partial\phi}{\partial y}, \sigma_{yz} = -\frac{\partial\phi}{\partial x} \quad ....(1)$$

By imposing compatibility on these stresses one can show that the stress function satisfies the equation

$$\nabla^2 \phi = -2G\theta \text{ , in cross section domain } \Omega \quad ....(2)$$

Where,

$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}, G = \frac{E}{2(1+\nu)}, \nu\ is\ Poisson'sratio, G\ is\ the\ torisonal\ rigidity\ of\ the\ bar. E\ is\ the\ Young's$ modulus of the material and θ is the twist angle per unit length of the bar.

The boundary conditions are derived from the fact that there are no tractions on the periphery of the cross section which results in the requirement that

$$\frac{d\,\phi}{ds} = 0, \text{ over the boundary of the cross section } domain\ \partial\Omega \quad ....(3)$$

where s is arc-length along the boundary, measured in the counter clockwise direction. This condition is equivalent to $\phi = 0$, over the boundary of the cross section domain $\partial\Omega$

Once the stress function $\phi(x, y)$ is determined one can compute the stresses by formulas (1) and then the imposed torque T is balanced by the torque of the shear stresses over the cross section Ω to determine the twist angle θ. The integration of the shear stress torque over the cross section domain Ω is

$$T = \iint_{\Omega} \left[x\sigma_{yz} - y\sigma_{xz}\right] d\,\mathrm{xdy} \qquad \text{...(4)}$$

where Ω is the area of the cross section. By substituting the stress equations (1) into equation (4) one obtains

$$T = \iint_{\Omega} \left[-x\frac{\partial\phi}{\partial x} - y\frac{\partial\phi}{\partial y}\right] d\,\mathrm{xdy} \qquad \text{...(5)}$$

which can be written as

$$T = 2\iint_{\Omega} \phi\, d\,\mathrm{xdy} - \iint_{\Omega} \left[\frac{\partial(x\phi)}{\partial x} + \frac{\partial(y\phi)}{\partial y}\right] d\,\mathrm{xdy} \qquad \text{...(6)}$$

and applying Gauss' theorem to the second integration on the right of (6) we obtain

$$T = 2\iint_{\Omega} \phi\, d\,\mathrm{xdy} - \oint_{\partial\Omega} \vec{\mathrm{n}}.\left[x\phi\vec{\mathrm{i}} + y\phi\vec{j}\,\right]\mathrm{d}\ell \qquad \text{..(7)}$$

The line integral around the boundary $\partial\Omega$ is zero when $\phi = 0$ on $\partial\Omega$. Thus the integration of the stresses (4) reduces to

$$T = 2\iint_{\Omega} \phi\, d\,\mathrm{xdy} \qquad \text{..(8)}$$

This states that we can solve the problem of torsion by calculating the value of a single unknown, called the Prandtl stress function $\phi(\mathrm{x}, \mathrm{y})$ This function is differentiable and is associated with the constraints according to relation (1)

If it is assumed that ϕ =2Gθu in eqn(2),we have the following form

∇^2 u=f in Ω ; u=0 on $\partial\Omega$

with f=-1

....(9)

and $\frac{\sigma_{xz}}{G\Theta} = 2\frac{\partial u}{\partial y}$; $\frac{\sigma_{yz}}{G\Theta} = -2\frac{\partial u}{\partial x}$; $\theta = \frac{T}{GC}$;C=$4\iint_{\Omega} \mathrm{u}(\mathrm{x}, \mathrm{y})\, d\,\mathrm{xdy}$ ....(10)

The finite element solution of Poisson boundary value problem given above in eqn(9) can be found by the method proposed in our recent work[ ].We solve the bvp of eqn(9) over regular polygonal domains.

2.1 Statement of the Poisson Boundary Value Problem

The Poisson equation[33]

$$-\nabla^2 \mathrm{u} = f \qquad \text{...(11)}$$

is the simplest and most famous elliptic partial differential equations.The source (or load) function is given on some two or three dimensional domain$\Omega \subset \mathcal{R}^2$ or $\mathcal{R}^3$. A solution u satisfying (1.1) will also satisfy boundary conditions on the boundary $\partial\Omega$ of Ω ;for example

$$\alpha u + \beta\frac{\partial \mathrm{u}}{\partial \mathrm{n}} = \mathrm{g} \quad \text{on} \quad \partial\Omega \qquad \text{..(12)}$$

where $\partial\mathrm{u}\,/\partial\mathrm{n}$ denotes directional derivative in the direction normal to the boundary $\partial\Omega$ (conveniently pointing outwards) and α and β are constants, although variable coefficients are also possible.The combination of (1.1) and (1.2) together is referred to as boundary value problem. If the constant β in (1.2) is zero,then the boundary condition is known as the Dirichlet type, and the boundary value problem is referred as the Dirichlet problem for the Poisson equation. Alternatively, if the constant α in (1.2) is zero,then we correspondingly have a Neumann boundary value problem. A third possibility is that Dirichlet conditions hold on part of the boundary $\partial\Omega_{\mathrm{D}}$ and Neumann conditions(or indeed mixed conditions where α and β are both nonzero) hold on remainder $\partial\Omega\backslash\,\partial\Omega_{\mathrm{D}}$. The case $\alpha = 0, \beta = 1$ in (1.2) demands special attention.First, since u=constant satisfies the homogeneous problem with $f = 0, g = 0$,it is clear that a solution to a Neumann problem can only be unique up to an additive constant.Second,integrating (1.1) over Ω using Gauss's theorem gives

$$-\int_{\partial\Omega}\frac{\partial \mathrm{u}}{\partial \mathrm{n}} = --\int_{\Omega} \nabla^2 \mathrm{u} = \int_{\Omega} f \qquad \text{...(13)}$$

thus a necessary condition for the existence of a solution to the Neumann problem is that the source and boundary data satisfy the compatibility condition:

$$\int_{\partial\Omega} g + \int_{\Omega} f = 0 \qquad \text{.. (14)}$$

2.2 Weak Formulation of the Poisson Boundary Value Problem

A sufficiently smooth function u satisfying both eqns(1) and (2) is known as classical solution to the Poisson boundary value problem. For a Dirichlet problem, u is a classical solution only if it has continuousecond derivatives in Ω (i.e. u is $C^2(\,\Omega\,)$ and is continuous up to the boundary i.e.u is in $C^0(\overline{\Omega})$). In case of nonsmooth domains or discontinuous source functions,the function u satisfying eqns(1) and (2) may not be smooth (or regular) enough to be regarded as classical solution. For problems which arise from , perfectly reasonable mathematical models an alternative description of the boundary value problem is required. Since this alternative description is less restrictive in terms of admissible data it is called weak formulation.

To derive a weak formulation of a Poisson problem,we require that for an appropriate set of test functions v,

$$\int_{\Omega}(\nabla^2 u + f)\, v = 0 \qquad \text{...(15)}$$

This formulation exists provided that the integrals are well defined. If u is a classical solution then it must also satisfy eqn (5). If v is sufficiently smooth however,then the smoothness required of u can be reduced by using the derivative of a product rule and the divergence theorem

$$-\int_\Omega v\nabla^2 u = \int_\Omega \nabla u \,.\nabla v \text{ -}\int_\Omega \nabla.(v\nabla u)$$

$$=\int_\Omega \nabla u \,.\nabla v \text{ -}\int_{\partial\Omega} v\frac{\partial \mathrm{u}}{\partial \mathrm{n}},$$

so that

$$\int_\Omega \nabla u \,.\nabla v = \int_\Omega vf + \int_{\partial\Omega} v\frac{\partial \mathrm{u}}{\partial \mathrm{n}} \quad \text{...} (16a)$$

The point here is that the problem posed byeqn(6) may have a solution u called a weak solution,that is not smooth enough to be a classical solution. If a classical solution does exist then eqn(16)is equivalent to eqns (11) and (12) and the weak solution is classical.

The case of Neumann problem $(\alpha = 0 \ ,\beta = 1)$ in eqn(2) is particularly straight forward. Substituting from eqn(2) into eqn(6)gives us the following formulation: find u defined on Ω such that

$$\int_\Omega \nabla u \,.\nabla v = \int_\Omega vf + \int_{\partial\Omega} vg \quad \text{...} (16b)$$

for all suitable test functions v .

2.3 Finite Elements for Poisson's Equation with Dirichlet conditions: Implementation and Review Of Theory

2.3.1 Weak Form

Given Poisson Equation:

$-\Delta u(\mathbf{x})$=$f(\mathbf{x})$ **for all** $\mathbf{x}\in \boldsymbol{\Omega}$ **...**(17a)

$u = g(\boldsymbol{x})$ on $\partial\Omega$ ....(17b)

We have already obtained in eqn(6) with $(\alpha = 1 \ ,\beta = 0)$ the weak form of the equation by multiplying both sides by a test function v (i.e a function which is infinitely differentiable and has compact support,integrating over the domain Ω and performing integration by parts or by application of Divergence(GREEN) theorem. The result is

$\int_\Omega \nabla u \,.\nabla v \, d\,\boldsymbol{x} = \int_\Omega vf \, d\boldsymbol{x}$ ....(17c)

$u = g(\boldsymbol{x})$ on $\partial\Omega$ ....(17d)

For all test functions v .

2.3.2 Finite Elements

To find an approximation to the solution u , we choose a finite dimensional space V_h and ask that eqn(7a-b) is satisfied only for $v \; in \; V_h$ rather than for all test functions v. Then we look for a function $u_h \in V_h$ which satisfies

$\int_\Omega \nabla u_h \,.\nabla v \, d\,\boldsymbol{x} = \int_\Omega vf \, d\boldsymbol{x}, \forall\, \boldsymbol{v} \in V_h$ ....(18)

u_h is called the finite element solution and functions in V_h are called finite elements.

Note that it is also common for the triangles or quadrilaterals in the mesh to be called elements.

If a basis for V_h is $\{\varphi_j\}_{j=1}^{j=N}$ then we can write $u_h = \sum_{\mathbf{j=1}}^{\mathbf{j=N}} \boldsymbol{\alpha}_{\mathbf{j}}\boldsymbol{\varphi}_{\mathbf{j}}$ **.** Substituting this in eqn(8) and choosing v to be a basis function φ_i gives the following set of equations

$\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{\alpha_j} \int_\Omega \nabla\varphi_{\,i} \,.\nabla\varphi_j \, d\,\boldsymbol{x} = \int_\Omega f\varphi_i \, d\boldsymbol{x}$**,(**,i=1,2,3,....,N) ....(19)

This is really a linear system of the form

Ku=f **...(20)**

Where, $\boldsymbol{u}$ =$(\alpha_1\,,\alpha_2\,,\alpha_3\,,\ldots\ldots\ldots\alpha_N)^T$ and

$K_{i,j} = \int_\Omega \nabla\varphi_{\,i} \,.\nabla\varphi_j \, d\,\boldsymbol{x}$ **,** ....(21a)

$\boldsymbol{f_i} = \int_\Omega f\varphi_i \, d\boldsymbol{x}$ ....(21b)

and $\boldsymbol{K}$ is called stiffness matrix because the linear system looks like Hookes law if $\boldsymbol{f}$ represents forces and $\boldsymbol{u}$ represents displacements.

In general,$\Omega = \sum_{e=1}^{N_e} \Omega^e$, where N_e is the number of elements discritised in the domain Ω. In two dimensions the mesh elements are triangles or quadrilaterals.The choice of finite element spaces are usually piecewise polynomials.

2.3.3 Overview on the implementation of Finite Element Method

Once we have choosen the finite element space (and the element type),then we can implement the finite element method.The implementation is divided into three steps:

1. **Mesh Generation:**how does one perform a triangulation or quadrangulation of the domain Ω ?
2. **Assembling the Stiffness Matrix**:how does one compute the entries in the stiffness matrix in an efficient way?
3. **Solving the linear System:**What kind of methodse suited for solving the linear system?

In this paper,we present new approach to mesh generation [ ] and explicit computations for the entries in the stiffness matrix [ ] which is vital in Assembling the Stiffness Matrix,since we believe that the methods of solving linear system are well researched and standardised.

We shall first take up the derivations regarding the topic on **Assembling the Stiffness Matrix.** The **Mesh Generation** topic will be discussed immediately there after.

2.3.4 Assembling the Stiffness Matrix

In order to assemble the stiffness matrix,we need to compute integrals of the form(see eqn(21) in section 2.3.2)

$$K_{i,j} = \int_{\Omega} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\boldsymbol{x} \qquad \text{..(21a)}$$

The most obvious way to assemble the stiffness matrix is to compute the integrals $K_{i,j}$ for the nodal pairs i and j ; this is a node oriented computation and we need to know the common support of basis functions $\varphi_{\,i}$ *and* φ_j.This means we need to know which elements contain both i and j.The mesh generator provides us with the information regarding the nodes on a particular element so we would need to do some extra processing to find the elements that contain a particular node.This is an issue which is very complicated.Hence,in practice assembling is focussed on elements rather than on nodes.We note that on a particular element,the basis functions have a simple expression and the elements themselves are very simple domains like triangles and quadrilaterals. It is very easy to make a change of variables for integrals over triangles and quadrilaterals to standard triangles and squares. In the element oriented computation,we rewrite or interpret the integral in eqn(21) as

$$K_{i,j} = \sum_{\Omega^e\ \varepsilon\Omega_h^e} K_{i,j}^e \qquad \text{...(22a)}$$

where

$$K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\boldsymbol{x} \qquad \text{..................................(22b)}$$

and Ω_h^e is the set of (mesh) elements in Ω contributing to $K_{i,j}$ and $\Omega = \sum_{e=1}^{N_e} \Omega^e$, $\Omega^e$is an element contained in the set Ω_h^e .This says us that we can compute $K_{i,j}$ by computing the integrals over each element Ω^e and then summing up over all elements Ω_h^e .

Notice that the integrals

$K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\mathbf{x}$ look like the entries $K_{i,j} = \int_{\Omega} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\boldsymbol{x} = \sum_{\Omega^e\ \varepsilon\Omega_h^e} \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\boldsymbol{x}$ except the domain of integration is an element Ω^e. So,we only need to save all entries of K^e =$[K_{i,j}^e]$ which corresponds to nodes on Ω^e. Then if Ω^e has d nodes , we can think of K^e as a dxd matrix. In view of the above,the procedure for computing the stiffness maerix is done on an element by element basis.

We must also compute the integrals

$$\boldsymbol{f_i} = \int_{\Omega} f\varphi_i\, d\boldsymbol{x} = \sum_{e=1}^{N_e} \boldsymbol{f_i^e} \qquad \text{..(22c)}$$

where

$$\boldsymbol{f_i^e}\text{=} \int_{\Omega^e} f\varphi_i d\,\mathbf{x} \qquad \text{..(22d)}$$

Now further assume that on an element Ω^e , $u_h = \mathbf{u^e} = \sum_{\mathbf{j=1}}^{\mathbf{j=d}} \mathbf{u_j^e}\boldsymbol{\varphi}_\mathbf{j}$

From eqn(9) and eqns(12a-d) it follows that ***Ku=f*** **is equivalent to**

$$\sum_{e=1}^{N_e} K^e\, \mathbf{u^e} = \sum_{e=1}^{N_e} \boldsymbol{f^e} \qquad \textbf{...(22e)}$$

Where

$\mathbf{u^e}$ **= (**$\mathbf{u_1^e}$ **,** $\mathbf{u_2^e}$**,** $\mathbf{u_3^e}$**, ...** $\mathbf{u_d^e}$ **)**$^{\boldsymbol{T}}$ **,**

$\boldsymbol{f}^\mathbf{e}$ **= (**$\boldsymbol{f}_\mathbf{1}^\mathbf{e}$ **,** $\boldsymbol{f}_\mathbf{2}^\mathbf{e}$**,** $\boldsymbol{f}_\mathbf{3}^\mathbf{e}$**, ...** $\boldsymbol{f}_\mathbf{d}^\mathbf{e}$ **)**$^{\boldsymbol{T}}$ **...(22f)**

d referes to number of nodes per element, N_e referes to the total number of elements in the domain Ω

2.3.5 Computing the Integrals $\boldsymbol{K_{i,j}^e}$ and $\boldsymbol{f_i^e}$

In order to compute the local/element stiffness matrices,we need to compute the integrals $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\mathbf{x}$ .These integrals are computed by making a change of variables to a reference element. We now outline a brief procedure for element oriented computation

(1)For each element Ω^e , compute it's local stiffness matrix K^e. This requires computing the integrals $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\mathbf{x}$ which we compute by transforming to a reference element. In two dimensions Ω^e is an arbitrary linear triangle and each triangle will be further discritised three convex quadrilaterals Q_{3e-2} , Q_{3e-1} *and* Q_{3e} Each triangle will be transformed to the corresponding reference elements:the standard triangle(a right isosceles triangle) and further Each triangle will be transformed to the corresponding reference elements:the standard triangle(a right isosceles triangle) and further Each triangle will be transformed to the corresponding reference elements:the standard triangle(a right isosceles triangle) and further each quadrilateral will be transformed into a standard square(1-square or a 2-square).Since in two dimensional space **x**= (x,y) the explicit form of $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_{\,i}\,.\nabla\varphi_j\, d\,\mathbf{x}$ is given by

$K_{i,j}^{e} = \int_{\Omega^e} \nabla\varphi_{\,i} \,.\nabla\varphi_j \, d\mathbf{x}$ $=\int_{\Omega^e}\{\frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y}\} \, dxdy = \sum_{e=1}^{N_e}\sum_{n=0}^{2}\int_{Q_E}\{\frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y}\}dxdy$ $=\sum_{e=1}^{N_e}\sum_{n=0}^{2} S_{i,j}^{E}$ ....(12g)

Where $S_{i,j}^{E} = \int_{Q_E}\{\frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y}\}dxdy$ and E=3e+n-2,e=1,2,... N_e andn=0,1,2

and hence we must be careful about the derivatives when we perform the change of variables. These bring extra factors involving the affine transformations (when Ω^e is an arbitrary linear triangle) and bilinear transformations(when Ω^e is an arbitrary linear convex quadrilateral)

$\boldsymbol{f_i^e}$ = $\int_{\Omega^e} f\varphi_i$ dxdy can be computed in a straight forward manner if f is a simple function otherwise we have to apply numerical integration

(2) For each element Ω^e, first compute the local stiffness matrices $S^E = [S_{i,j}^{E}\,]$ and then add contribution of $K^e = S^{3e-2}$ $+S^{3e-1} + S^{3e}$, to the global stiffness matrix K. We repeat this procedure for all elements i.e for e=1,2,......, N_e; where N_e is the number of elements Ω^e *which* are discritised in the domain Ω ,in fact we have $\Omega = \sum_{e=1}^{N_e}\Omega^e$ $=\sum_{e=1}^{N_e}\sum_{n=0}^{2} Q_E$, E=3e+n-2

2.3.6 Linear Convex Quadrilateral Elements :

Let us first consider an arbitrary four noded linear convex quadrilateral in the global (Cartesian) coordinate system (u, v) as in Fig 1a, which mapped into a 2-square in the local(natural) parametric coordinate (ξ, η) as in Fig 1b.

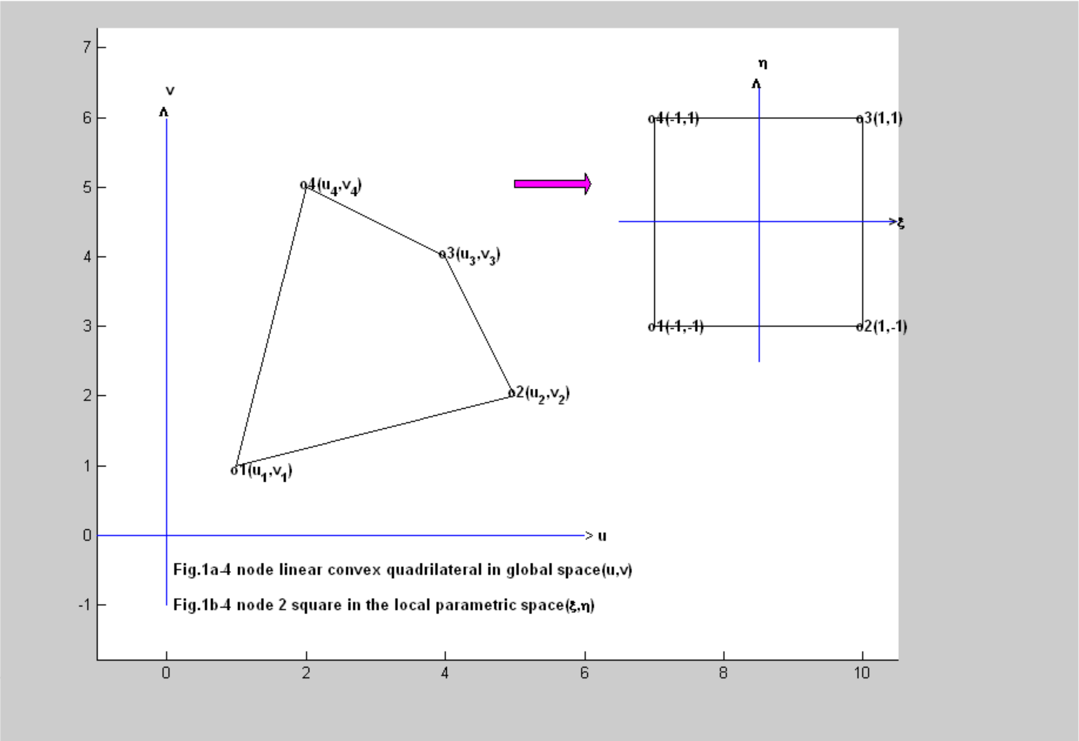


Fig.1a-4 node linear convex quadrilateral in global space(u,v)

Fig.1b-4 node 2 square in the local parametric space(ξ,η)

$\binom{u}{v} = \sum_{k=1}^{4}\binom{u_k}{v_k}$ $M_k(\xi,\eta)$ .... (23)

Where (u_k, v_k) , (k=1,2,3,4) are the vertices of the original arbitrary linear convex quadrilateral in (u, v) plane and $M_k(\xi,\eta)$ denote the well known bilinear basis functions [1-3] in the local parametric space (ξ, η) and they are given by

$M_k(\xi,\eta) = \frac{1}{4}\,(1+\xi\xi_k)(1+\eta\eta_k)$, $k = 1, 2,3,4$ ---------------------- (24a)

Where { $(\xi_k, \eta_k), k = 1,2,3,4$} = {(-1,-1),(1,-1),(1,1),(-1,1)} ---------------------- (24b) describe two transformations over a linear convex quadrilateral element from the original global space into the local parametric space.

2.3.7 Isoparametric Transformation :

For the isoparametric coordinate transformation over the linear convex quadrilateral element as shown in Fig 1, we select the field variables, say ϕ, ψ , etc governing the physical problem as

$$\begin{pmatrix} \phi \\ \psi \end{pmatrix} = \sum_{k=1}^{4} \begin{pmatrix} \phi_k \\ \psi_k \end{pmatrix} N_k{}^e(\xi, \eta) \qquad \text{------------------------ (25)}$$

Where ϕ_k , ψ_k refer to unknowns at node k and the shape functions $N_k{}^e = M_k$, and M_k are defined as in Eqn.(2a-b)

2.3.8 Subparmetric Transformation :

For the subparametric transformation over the nde – noded element we define the field variables ϕ, ψ (say) governing the physical problem as

$$\begin{pmatrix} \phi \\ \psi \end{pmatrix} = \sum_{k=1}^{nde} \begin{pmatrix} \phi_k{}^e \\ \psi_k{}^e \end{pmatrix} N_k{}^e(\xi, \eta) \qquad \text{------------------------ (26)}$$

Where ϕ_k , ψ_k refer to unknowns at node k and nde $>$4

In our recent paper[ ], the explicit finite element integration scheme is presented by using the isoparametric transformation over the 4 node linear convex quadrilateral element which is applied to torison of square shaft,on considering symmetry mesh generation for 1/8 of the cross section which is a triangle was discritised into an all quadrilateral mesh.In this paper we consider applications to polygonal domains.

2.3.9 Explicit Form of the Jacobian and Global Derivatives :

Jacobian

Let us consider an arbitrary linear convex quadrilateral in the global Cartesian space (u, v) as in Fig 1a , c which is mapped into a 8- node 2-square in the local parametric space (ξ, η) as in Fig 1b, d

From the Eq.(13) and Eq.(14), we have

$$\frac{\partial u}{\partial \xi} = \sum_{k=1}^{4} u_k \frac{\partial M_k}{\partial \xi} = \frac{1}{4}[\,(-u_1 + u_2 + u_3 - u_4) + (u_1 - u_2 + u_3 - u_4)\,\eta\,] \qquad \text{------------ (27a)}$$

$$\frac{\partial u}{\partial \eta} = \sum_{k=1}^{4} u_k \frac{\partial M_k}{\partial \eta} = \frac{1}{4}[\,(-u_1 - u_2 + u_3 + u_4) + (u_1 - u_2 + u_3 - u_4)\,\xi\,] \qquad \text{------------ (27b)}$$

$$\frac{\partial v}{\partial \xi} = \frac{1}{4}[\,(-v_1 + v_2 + v_3 - v_4) + (v_1 - v_2 + v_3 - v_4)\,\eta\,] \qquad \text{------------ (27c)}$$

$$\frac{\partial v}{\partial \eta} = \frac{1}{4}[\,(-v_1 - v_2 + v_3 + v_4) + (v_1 - v_2 + v_3 - v_4)\,\xi\,] \qquad \text{------------ (27d)}$$

Hence the Jacobian, J can be expressed as [1, 2, 3]

$$J = \frac{\partial(u,v)}{\partial(\xi,\eta)} = \frac{\partial u}{\partial \xi}\frac{\partial v}{\partial \eta} - \frac{\partial u}{\partial \eta}\frac{\partial v}{\partial \xi} = \alpha + \beta\xi + \gamma\,\eta \qquad \text{-------------- (28a)}$$

Where

$$\alpha = \frac{1}{8}[\,(u_4 - u_2)(v_1 - v_3) + (u_3 - u_1)(v_4 - v_2)\,]$$

$$\beta = \frac{1}{8}[\,(u_4 - u_3)(v_2 - v_1) + (u_1 - u_2)(v_4 - v_3)\,]$$

$$\gamma = \frac{1}{8}[\,(u_4 - u_1)(v_2 - v_3) + (u_3 - u_2)(v_4 - v_1)\,] \qquad \text{---------------- (28b)}$$

Global Derivatives:

If N_i^e denotes the basis functions of node i of any order of the element e, then the chain rule of differentiation from Eq.(1) we can write the global derivative as in [1, 2, 3]

$$\begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix} = \frac{1}{J} \begin{bmatrix} \frac{\partial v}{\partial \eta} & -\frac{\partial v}{\partial \xi} \\ -\frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} \\ \frac{\partial N_i^e}{\partial \eta} \end{bmatrix} \quad \text{--------------------------- (29)}$$

Where $\frac{\partial u}{\partial \xi}, \frac{\partial u}{\partial \eta}, \frac{\partial v}{\partial \xi}$ and $\frac{\partial v}{\partial \eta}$ are defined as in Eqs.(17a)–(17d) while J is defined in Eq.(18a-b) , ($i, j = 1,2,3, - - - - -$, nde) , nde = the number of nodes per element.

2.3.10 Discretisation of an Arbitrary Triangle :

A linear convex polygon in the physical plane (x, y) can be always discretised into a finite number of linear triangles. However, we would like to study a particular discretization of these triangles further into linear convex quadrilaterals. This is stated in the following Lemma [6].

Lemma 1. Let Δ PQR be an arbitrary triangle with the vertices P(x_p, y_p) , Q(x_q, y_q) and R(x_r, y_r) and S, T, U be the midpoints of sides PQ, QR and RP respectively and let Z be its centroid. We can obtain three linear convex quadrilaterals ZTRU, ZUPS and ZSQT from triangle Δ PQR as shown in Fig2. If we map each of these quadrilaterals into 2-squares in which the nodes are oriented in counter clockwise from Z then Jacobian J for each element e is given by

$$J = J^e = \frac{1}{48} \Delta\, pqr\, (4 + \xi + \eta), \qquad e = 1,2,3 \quad \text{--------------- (30)}$$

Where Δ pqr is the area of the triangle Δ PQR

$$2\Delta\, pqr = \begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix} = [\, (x_p - x_r)(y_q - y_r) - (x_q - x_r)(\, y_p - y_r)\,] \quad \text{------------ (31)}$$

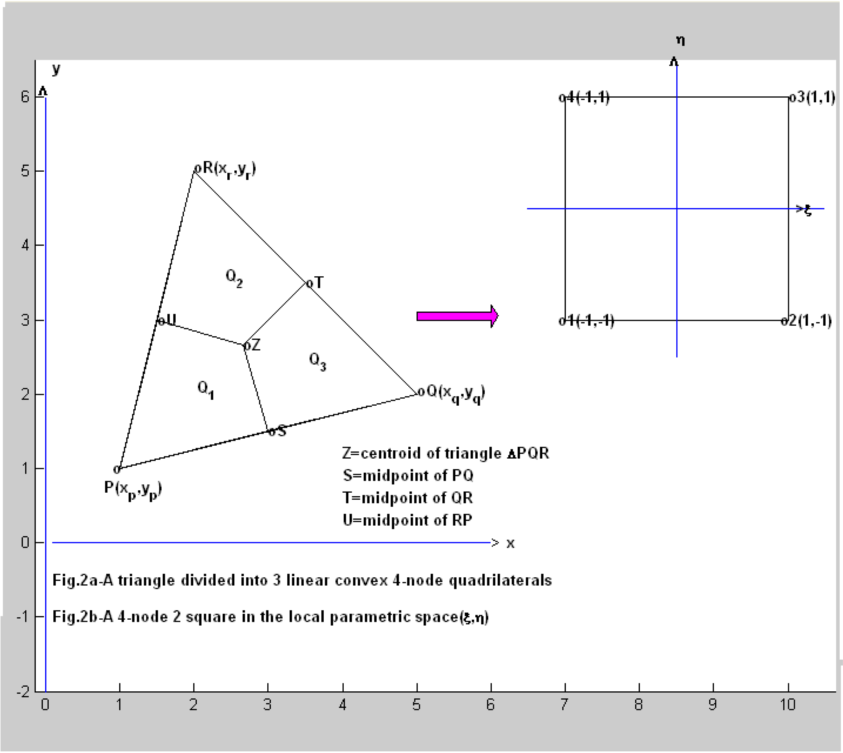
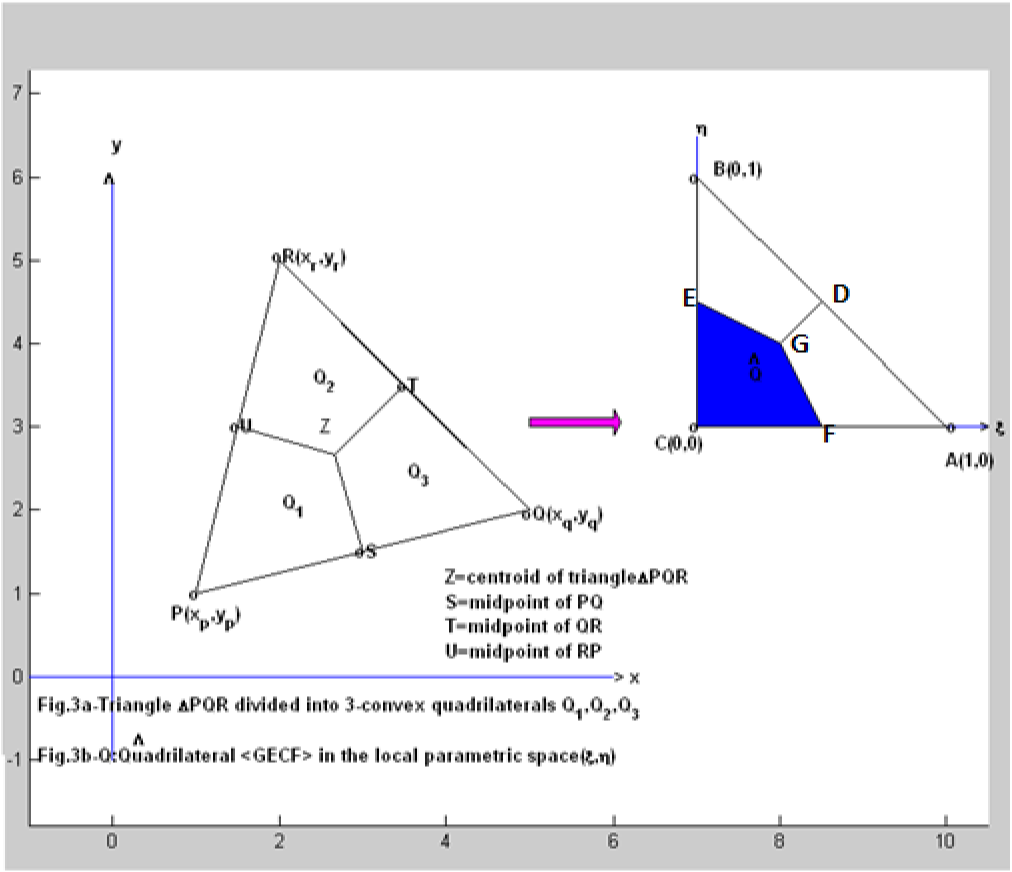


Fig.2a-A triangle divided into 3 linear convex 4-node quadrilaterals

Fig.2b-A 4-node 2 square in the local parametric space(ξ,η)

Proof : Proof is straight forward and it can be elaborated on the lines of proof given in [17].

Lemma 2. Let Δ PQR be an arbitrary triangle with the vertices P(x_p, y_p) , Q(x_q, y_q) and R(x_r, y_r) , let S, T, U be the midpoints of sides PQ, QR, and RP and let Z be the centroid of Δ PQR, Then we obtain three quadrilaterals Q_1, Q_2, Q_3 spanning the vertices <ZUPS> , <ZSQT> and <ZTRU> . these quadrilaterals can be mapped into the quadrilateral spanning vertices GECF with G(1/3, 1/3), E(0, ½) , C(0, 0) and F(1/2, 0) of the right isosceles triangle Δ ABC with spanning vertices A(1, 0), B(0, 1) and C(0, 0) in the (u, v) space as shown in Fig 3a and Fig 3b



Z=centroid of triangle ΔPQR
S=midpoint of PQ
T=midpoint of QR
U=midpoint of RP

Fig.3a-Triangle ΔPQR divided into 3-convex quadrilaterals Q_1,Q_2,Q_3

Fig.3b-$\hat{Q}$:Quadrilateral <GECF> in the local parametric space(ξ,η)

Proof : The sum of the quadrilaterals Q₁+ Q₂+ Q₃ = Δ PQR as shown in Fig 2a & Fig 3a. The linear transformations

$$\begin{pmatrix} x^{(1)} \\ y^{(1)} \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} w + \begin{pmatrix} x_q \\ y_q \end{pmatrix} u + \begin{pmatrix} x_r \\ y_r \end{pmatrix} v \qquad \text{---------------------------- (32a)}$$

$$\begin{pmatrix} x^{(2)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} x_q \\ y_q \end{pmatrix} w + \begin{pmatrix} x_r \\ y_r \end{pmatrix} u + \begin{pmatrix} x_p \\ y_p \end{pmatrix} v \qquad \text{-------------------------- (32b)}$$

$$\begin{pmatrix} x^{(3)} \\ y^{(3)} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} w + \begin{pmatrix} x_p \\ y_p \end{pmatrix} u + \begin{pmatrix} x_q \\ y_q \end{pmatrix} v \qquad \text{------------------------ (32c)}$$

with $w = 1 - u - v$ ------------------------ (32d)

map the arbitrary triangle Δ PQR into a right isosceles triangle A(1, 0) , B(0, 1) and C(0, 0) in the uv–plane We can now verify that quadrilateral Q₁ spanned by vertices Z($\frac{x_p+x_q+x_r}{3}$, $\frac{y_p+y_q+y_r}{3}$) , U($\frac{x_r+x_p}{2}$, $\frac{y_r+y_q}{2}$) , P(x_p, y_p) ,S($\frac{x_p+x_q}{2}$, $\frac{y_p+y_q}{2}$) in xy- plane is mapped into the quadrilateral spanning the vertices G(1/3, 1/3) , E(0, ½), C(0, 0) and F(1/2, 0) by use of the transformation given in Eq.(32a),

Similarly, we see that the quadrilateral Q₂ spanned by vertices Z,S,Q,T is mapped into the quadrilateral spanned by vertices G(1/3, 1/3), E(0, ½), C(0, 0) and F(½, 0) by use of the transformation of Eq.(32b), Finally the quadrilateral Q₃ in the xy- plane is mapped into the quadrilateral GECF in uv- plane by use of the linear transformation of Eq.(32c),

This completes the proof

We have shown in the present section that an arbitrary triangle can be discretised into three linear convex quadrilaterals. Further, each of these quadrilaterals can be mapped into a unique quadrilateral in uv-plane spanned by vertices (1/3, 1/3), (0, ½), (0, 0) and (½, 0) .

3.0 Computing the Integrals : $K_{i,j}^{e}$

3.1 Global Derivative Integrals: $\int_{\Omega} \nabla\varphi_{i} . \nabla\varphi_{j} \, d\boldsymbol{x}$

If $N_i^{(e)}$ denotes the basis function for node i of element e , then by chain rule of partial differentiation

$$\begin{pmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{pmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial u}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{bmatrix} \qquad \text{-------------------------- (33)}$$

We note that to transform $Q_e(e = 1, 2, 3)$ of ΔPQR in Cartesian space (x,y) into $\widehat{Q}$, the quadrilateral spanned by vertices (1/3,1/3), (0,½), (0,0) and (1/2,0) in uv-plane we must use the earlier transformations.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} + \begin{pmatrix} x_q - x_p \\ y_q - y_p \end{pmatrix} u + \begin{pmatrix} x_r - x_p \\ y_r - y_p \end{pmatrix} v \quad \text{for } Q_1 \text{ in } \Delta PQR \qquad \text{--------------------------- (34a)}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} x_r - x_q \\ y_r - y_q \end{pmatrix} u + \begin{pmatrix} x_p - x_q \\ y_p - y_q \end{pmatrix} v \quad \text{for } Q_2 \text{ in } \Delta PQR \qquad \text{-------------------------- (34b)}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \begin{pmatrix} x_p - x_r \\ y_p - y_r \end{pmatrix} u + \begin{pmatrix} x_q - x_r \\ y_q - y_r \end{pmatrix} v \quad \text{for } Q_3 \text{ in } \Delta PQR \qquad \text{------------------------ (34c)}$$

and the above transformations viz Eqs.(24a)-(24c) are of the form

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + \begin{pmatrix} x_a - x_c \\ y_a - y_c \end{pmatrix} u + \begin{pmatrix} x_b - x_c \\ y_b - y_c \end{pmatrix} v \qquad \text{------------------------ (34d)}$$

which can map an arbitrary triangle ΔABC , A(x_a ,y_a) ,B(x_b ,y_b) , C(x_c , y_c) in xy – plane into a right isosceles triangle in the uv – plane

Hence, we have

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} (x_a - x_c) & (x_b - x_c) \\ (y_a - y_c) & (y_b - y_c) \end{pmatrix}^{-1} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} \qquad \text{------------------------- (35)}$$

This gives

$u = (\propto_a + \beta_a x + \gamma_a y)/(2\ \Delta_{abc})$

$v = (\propto_b + \beta_b x + \gamma_b y)/(2\ \Delta_{abc})$ ------------------------ (36)

$\propto_a = (x_b y_c - x_c y_b)$, $\propto_b = (x_c y_a - x_a y_c)$,

$\beta_a = (y_b - y_c)$, $\beta_b = (y_c - y_a)$,

$\gamma_a = (x_c - x_b)$, $\gamma_b = (x_a - x_c)$,

$$\frac{\partial(x,y)}{\partial(u,v)} = 2\Delta_{abc} = \begin{vmatrix} 1 & x_a & y_a \\ 1 & x_b & y_b \\ 1 & x_c & y_c \end{vmatrix} = 2 * \text{area of the triangle } \Delta ABC$$

$= (\gamma_b \beta_a - \gamma_a\ \beta_b)$ ------------------- (37)

Hence from Eq.(23) and Eq.(22), we obtain

$$\begin{pmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\beta_a}{2\Delta_{abc}} & \frac{\beta_b}{2\Delta_{abc}} \\ \frac{\gamma_a}{2\Delta_{abc}} & \frac{\gamma_b}{2\Delta_{abc}} \end{pmatrix} \begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix}$$

$= \begin{pmatrix} \beta_a{}^* & \beta_b{}^* \\ \gamma_a{}^* & \gamma_b{}^* \end{pmatrix} \begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix}$ ----------------------------- (38)

where $\beta_a{}^* = \frac{\beta_a}{(2\Delta_{abc})}$, $\beta_b{}^* = \frac{\beta_b}{(2\Delta_{abc})}$

$\gamma_a{}^* = \frac{\gamma_a}{(2\Delta_{abc})}$, $\gamma_b{}^* = \frac{\gamma_b}{(2\Delta_{abc})}$ --------------------------- (39)

Letting,

$D_{x,y}^{i,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{pmatrix}$, $P = \begin{pmatrix} \beta_a{}^* & \beta_b{}^* \\ \gamma_a{}^* & \gamma_b{}^* \end{pmatrix}$, $D_{u,v}^{i,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix}$ ---------------------------- (40)

We obtain from Eq.(24),

$D_{x,y}^{i,e} = P\ D_{u,v}^{i,e}$ ----------------------------- (41)

So that from Eq.(30) and Eq.(31) , we obtain

$$G_{x,y}^{i,j,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{pmatrix} \left(\frac{\partial N_j^e}{\partial x}\ \ \frac{\partial N_j^e}{\partial y} \right) = (D_{x,y}^{i,e})\ (D_{x,y}^{j,e})^T$$

$= \begin{pmatrix} \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x} & \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial y} \\ \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial x} & \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y} \end{pmatrix}$ ----------------------------- (42a)

$$G_{u,v}^{i,j,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix} \left(\frac{\partial N_j^e}{\partial u}\ \ \frac{\partial N_j^e}{\partial v} \right) = (D_{u,v}^{i,e})\ (D_{u,v}^{j,e})^T$$

$= \begin{pmatrix} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial u} & \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial v} \\ \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial u} & \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial v} \end{pmatrix}$ ----------------------------- (42b)

We have now from Eq.(27) and Eq.(28)

$G_{x,y}^{i,j,e} = (P\ D_{u,v}^{i,e})\ (\left(D_{u,v}^{j,e}\right)^T P^T)$

$= P\ G_{u,v}^{i,j,e}\ P^T$ --------------------------- (43)

We now define the submatrices of global derivative integrals in (x,y) and (u,v) space associated with the nodes i and j as ;

$S^{i,j,e} = \iint_{Q_e} G_{x,y}^{i,j,e}\ \ dx\,dy$, (e=1,2,3) --------------------------- (44)

$K^{i,j,e} = \iint_{\widehat{Q}} G_{u,v}^{i,j,e}\ \ du\ dv$ --------------------------- (45)

where, we have already defined the quadrilaterals Q_e (e=1,2,3) in (x,y) space and $\widehat{Q}$ in (u,v) space in Fig 3a-b. From Eqs.(28)-(33) , we obtain the following relations connecting the submatrices $S^{i,j,e}$ and $K^{i,j,e}$

We now obtain the submatrices $S^{i,j,e}$ and $K^{i,j,e}$ in an explicit form from Eqs.(32a)- (32b) as

$$S^{i,j,e} = \iint_{Q_e} G_{x,y}^{i,j,e}\ \ dx\,dy = \begin{pmatrix} \iint_{Q_e} \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x}\ dxdy & \iint_{Q_e} \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial y}\ dxdy \\ \iint_{Q_e} \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial x}\ dxdy & \iint_{Q_e} \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y}\ dxdy \end{pmatrix}$$

$= \begin{pmatrix} S_{2i-1,2j-1}^e & S_{2i-1,2j}^e \\ S_{2i,2j-1}^e & S_{2i,2j}^e \end{pmatrix}$ (say) ------------------(46)

and in similar manner

$$K^{i,j,e} = \iint_{\widehat{Q}} G_{u,v}^{i,j,e}\ \ du\,dv = \begin{pmatrix} \iint_{\widehat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial u}\ dudv & \iint_{\widehat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial v}\ dudv \\ \iint_{\widehat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial u}\ dudv & \iint_{\widehat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial v}\ dudv \end{pmatrix}$$

$= \begin{pmatrix} K_{2i-1,2j-1}^e & K_{2i-1,2j}^e \\ K_{2i,2j-1}^e & K_{2i,2j}^e \end{pmatrix}$ (say) ------------------(47)

We now obtain from the above Eq.(23)-(33)

$S^{i,j,e} = \iint_{Q_e} G_{x,y}^{i,j,e}\ \ dx\,dy = \iint_{\widehat{Q}} (P\ G_{u,v}^{i,j,e} P^T)\ \frac{\partial(x,y)}{\partial(u,v)}\ \ du\ dv$

$= 2\Delta_{abc}\ \iint_{\widehat{Q}} (P\ G_{u,v}^{i,j,e} P^T)\ du\ dv$

$= \mathbf{2\Delta_{abc}\ P\ (\iint_{\widehat{Q}}\ G_{u,v}^{i,j,e}\ du\ dv)\ \ P^T}$

$= \mathbf{2\Delta_{abc}\ \ P\ (K^{i,j,e})\ \ P^T}$ **-----------------------(48)**

We can thus obtain the global derivative integrals in the physical space or Cartesian space (x,y) by using the matrix triple product established in eqn.(43). From Eq.(35) and noting the fact that $\widehat{Q}$ is the quadrilateral in (u, v) space spanned by the vertices (1/3, 1/3), (0, 1/2), (0, 0)and (1/2, 0) we obtain

$\mathbf{K^{i,j,e} = \iint_{\widehat{Q}}\ G_{u,v}^{i,j,e}\ \ du\ dv}$

$\mathbf{= \int_{-1}^{1}\int_{-1}^{1} G_{u,v}^{i,j,e}\ \frac{\partial(u,v)}{\partial(\xi,\eta)}\ \ d\xi\ d\eta}$ **-----------------------(49)**

We now refer to section 5 of this paper, in this section we have derived the necessary relations to integrate the integrals of Eq.(39), As in Eq.(32a-d) , we use the transformation

u(ξ, η) = $\frac{1}{3}N_1(\xi,\ \eta)+\frac{1}{2}N_4(\xi,\ \eta)$

v(ξ, η) = $\frac{1}{3}N_1(\xi,\ \eta)+\frac{1}{2}N_2(\xi,\ \eta)$ **-----------------------(50)**

to map the quadrilateral $\hat{Q}$ **to the 2-square -1≤** $\xi,\eta\le 1$ **Using Eq.(40) in Eq.(39), we obtain**

$K^{i,j,e}=\iint_{\hat{Q}} G^{i,j,e}_{u,v}\ (\frac{4+\xi+\eta}{96})\ d\xi d\eta$ **----------------------- (51)**

The submatrices for the quadrilateral Q_e **is expressed from Eq.(38) as**

$S^{i,j,e}=\ (2\Delta_{abc})\ P\ (K^{i,j,e})\ P^T$ **---------------------- (52)**

In eqn.(38) , $2\Delta_{abc}=2$ **x area of the triangle spanning vertices A(** x_a **,** y_a**) ,B(**x_b **,**y_b**) , C(**x_c **,** y_c**) which is scalar.**

The matrices P, P^T **depend purely on the nodel coordinates (** x_a **,**y_a**) ,(**x_b **,**y_b**) , (**x_c**,** y_c**) the matrix** $K^{i,j,e}$ **can be explicity computed by the relations obtained in section 2 and 3 . We find that** $K^{i,j,e}$ **is a (2X2) matrix of integrals whose integrands are rational functions with polynomial numerator and the linear denominator** $(4+\xi+\eta)$**. Hence these integrals can be explicity computed. The explicit values of these integrals are expressible in terms of logarithmic constants. We have used symbolic mathematics software of MATLAB to compute the explicit values and their conversion to any number of digits can be obtained by using variable precision arithmetic (vpa) command. The matrix** K^e **=[**$K^{i,j,e}$**] is of order (2x**n_{de}**) x (2x**n_{de}**) . We have computed** K^e **for the four noded isoparametric quadrilateral element. This is listed in Table 1,1 and Table 1.2 ,**

Table-1.1

Values of integrals of the product of global derivatives over the quadrilateral {(xk,yk),k=1,2,3,4}={(1/3,1/3),(0,1/2),(0,0),(1/2,0)}, in the interior of the standard triangle (see eqn (37))

```
IntJdnidnjuvrs=[ Ke (2*i-1,2*j-1)  Ke (2*i-1,2*j)
                 Ke (2*i,2*j-1)     Ke (2*i,2*j)]
where,  (i,j=1,2,3,4,5,6,7,8)
                  ANALYTICAL VALUES

IntJdn1dn1uvrs = [  -11/2-34*log (2)+27*log (3), -1/2-20*log (2)+27/2*log (3);...
          -1/2-20*log (2)+27/2*log (3),  -11/2-34*log (2)+27*log (3) ]

IntJdn1dn2uvrs =  [11/3+68/3*log (2)-18*log (3),   5/6+40/3*log (2)-9*log (3);...
           1/3+40/3*log (2)-9*log (3), 25/6+68/3*log (2)-18*log (3) ]

IntJdn1dn3uvrs = [   -7/3-34/3*log (2)+9*log (3), -2/3-20/3*log (2)+9/2*log (3);...
          -2/3-20/3*log (2)+9/2*log (3),   -7/3-34/3*log (2)+9*log (3)]

IntJdn1dn4uvrs = [ 25/6+68/3*log (2)-18*log (3),   1/3+40/3*log (2)-9*log (3);...
           5/6+40/3*log (2)-9*log (3), 11/3+68/3*log (2)-18*log (3)]

IntJdn2dn1uvrs = [ 11/3+68/3*log (2)-18*log (3),   1/3+40/3*log (2)-9*log (3);...
           5/6+40/3*log (2)-9*log (3), 25/6+68/3*log (2)-18*log (3)]

IntJdn2dn2uvrs = [ -22/9-136/9*log (2)+12*log (3),    -5/9-80/9*log (2)+6*log (3);...
           -5/9-80/9*log (2)+6*log (3), -22/9-136/9*log (2)+12*log (3) ]

IntJdn2dn3uvrs =[   14/9+68/9*log (2)-6*log (3),   4/9+40/9*log (2)-3*log (3);...
          -1/18+40/9*log (2)-3*log (3), 19/18+68/9*log (2)-6*log (3)]

IntJdn2dn4uvrs = [ -25/9-136/9*log (2)+12*log (3),    -2/9-80/9*log (2)+6*log (3);...
           -2/9-80/9*log (2)+6*log (3), -25/9-136/9*log (2)+12*log (3)]

IntJdn3dn1uvrs = [   -7/3-34/3*log (2)+9*log (3), -2/3-20/3*log (2)+9/2*log (3);...
          -2/3-20/3*log (2)+9/2*log (3),   -7/3-34/3*log (2)+9*log (3) ]

IntJdn3dn2uvrs = [ 14/9+68/9*log (2)-6*log (3), -1/18+40/9*log (2)-3*log (3);...
           4/9+40/9*log (2)-3*log (3), 19/18+68/9*log (2)-6*log (3)]

IntJdn3dn3uvrs =[   -5/18-34/9*log (2)+3*log (3), 5/18-20/9*log (2)+3/2*log (3);...
          5/18-20/9*log (2)+3/2*log (3),  -5/18-34/9*log (2)+3*log (3) ]

IntJdn3dn4uvrs = [ 19/18+68/9*log (2)-6*log (3),   4/9+40/9*log (2)-3*log (3);...
          -1/18+40/9*log (2)-3*log (3),  14/9+68/9*log (2)-6*log (3)]

IntJdn4dn1uvrs =  [25/6+68/3*log (2)-18*log (3),   5/6+40/3*log (2)-9*log (3);...
           1/3+40/3*log (2)-9*log (3), 11/3+68/3*log (2)-18*log (3)]

IntJdn4dn2uvrs =  -25/9-136/9*log (2)+12*log (3),    -2/9-80/9*log (2)+6*log (3);...
           -2/9-80/9*log (2)+6*log (3), -25/9-136/9*log (2)+12*log (3)

 IntJdn4dn3uvrs = [ 19/18+68/9*log (2)-6*log (3), -1/18+40/9*log (2)-3*log (3);...
           4/9+40/9*log (2)-3*log (3),  14/9+68/9*log (2)-6*log (3)]

IntJdn4dn4uvrs = [ -22/9-136/9*log (2)+12*log (3),    -5/9-80/9*log (2)+6*log (3);...
           -5/9-80/9*log (2)+6*log (3), -22/9-136/9*log (2)+12*log (3)]
```

Table-1.2
Values of integrals of the product of global derivatives over the quadrilateral {(xk,yk),k=1,2,3,4}={(1/3,1/3),(0,1/2),(0,0),(1/2,0)}, in the interior of the standard triangle (see eqn ()).

```
IntJdnidnjuvrs=[ K_e (2*i-1,2*j-1)  K_e (2*i-1,2*j)
                 K_e (2*i,2*j-1)      K_e (2*i,2*j)
  where,  (i,j=1,2,3,4,5,6,7,8)
```

NUMERICAL VALUES IN VARIABLE PRECISION ARITHMETIC

```
intJdn1dn1uvrs = vpa (sym (' .59552765500082114748572926733O')), vpa (sym ('.468322285820574645491168269290'));...
                 vpa (sym (' .46832228582057464549116826929')), vpa (sym ('.595527655000821147485729267330'))  ;
intJdn1dn2uvrs = vpa (sym ('  -.397018436667214098323819511552')),  vpa (sym ('.187785142786283569672554487139 5'));...
                 vpa (sym (' -.312214857213716430327445512860 4')),   vpa (sym ('.102981563332785901676180488448'))  ;
intJdn1dn3uvrs = vpa (sym (' -.301490781666392950838090244223 5')),vpa (sym (' -.343892571393141784836277243570 0'));...
        vpa (sym (' -.343892571393141784836277243570 0')), vpa (sym ('-.301490781666392950838090244223 5'))  ;
intJdn1dn4uvrs = vpa (sym ('   .102981563332785901676180488448')), vpa (sym ('-.312214857213716430327445512860 4'));...
        vpa (sym ('  .187785142786283569672554487139 5')),  vpa (sym ('-.397018436667214098323819511552'))  ;

intJdn2dn1uvrs = vpa (sym ('  -.397018436667214098323819511552')), vpa (sym ('-.312214857213716430327445512860 4'));...
        vpa (sym ('  .187785142786283569672554487139 5')),  vpa (sym (' .102981563332785901676180488448'))  ;
intJdn2dn2uvrs = vpa (sym ('   .264678957778142732215879674369')), vpa (sym ('-.125190095190855713115036324760 0'));...
        vpa (sym (' -.125190095190855713115036324760 0')),  vpa (sym (' .264678957778142732215879674369'))  ;
intJdn2dn3uvrs = vpa (sym ('  .200993854444261967225393496149 1')),  vpa (sym ('.229261714262094523224184829046 6'));...
        vpa (sym (' -.270738285737905476775815170953 4')), vpa (sym ('-.299006145555738032774606503850 9'))  ;
intJdn2dn4uvrs = vpa (sym (' -.686543755551906011174536589 65e-1')),   vpa (sym ('.208143238142477620218297008573 4'));...
        vpa (sym ('   .208143238142477620218297008573 4')), vpa (sym ('-.686543755551906011174536589 65e-1'))  ;

intJdn3dn1uvrs = vpa (sym (' -.301490781666392950838090244223 5')), vpa (sym ('-.343892571393141784836277243570 0'));...
        vpa (sym (' -.343892571393141784836277243570 0')),vpa (sym (' -.301490781666392950838090244223 5'))  ;
intJdn3dn2uvrs = vpa (sym ('  .200993854444261967225393496149 1')), vpa (sym ('-.270738285737905476775815170953 4'));...
        vpa (sym ('  .229261714262094523224184829046 6')), vpa (sym ('-.299006145555738032774606503850 9'))  ;
intJdn3dn3uvrs = vpa (sym (' .399503072777869016387303251925 4')), vpa (sym ('.385369142868952738387907585476 8'));...
        vpa (sym (' .385369142868952738387907585476 8')), vpa (sym ('.399503072777869016387303251925 4'))  ;
intJdn3dn4uvrs = vpa (sym (' -.299006145555738032774606503850 9')), vpa (sym (' .229261714262094523224184829046 6'));...
        vpa (sym (' -.270738285737905476775815170953 4')),  vpa (sym ('.200993854444261967225393496149 1'))  ;

intJdn4dn1uvrs = vpa (sym ('   .102981563332785901676180488448')),  vpa (sym ('.187785142786283569672554487139 5'));...
        vpa (sym (' -.312214857213716430327445512860 4')), vpa (sym (' -.397018436667214098323819511552'))  ;
intJdn4dn2uvrs = vpa (sym (' -.686543755551906011174536589 65e-1')),  vpa (sym (' .208143238142477620218297008573 4'));...
        vpa (sym ('   .208143238142477620218297008573 4')), vpa (sym ('-.686543755551906011174536589 65e-1'))  ;
intJdn4dn3uvrs = vpa (sym (' -.299006145555738032774606503850 9')), vpa (sym ('-.270738285737905476775815170953 4'));...
        vpa (sym ('  .229261714262094523224184829046 6')),  vpa (sym ('.200993854444261967225393496149 1'))  ;
intJdn4dn4uvrs = vpa (sym ('   .264678957778142732215879674369')), vpa (sym ('-.125190095190855713115036324760 0'));...
        vpa (sym (' -.125190095190855713115036324760 0')),  vpa (sym (' .264678957778142732215879674369'))  ;
```

We may note that In order to compute the local/element stiffness matrices for the Poisson Boundary Value problem, we need to compute the integrals Eqns(22a-b)

$$K^e_{i,j} = \int_{\Omega^e} \nabla\varphi_{\,i} \,.\, \nabla\varphi_j \, dx = \int_{\Omega^e}\{\frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y}\} \, dxdy \,, \qquad ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,(53a)$$

from the above derivations, we can rewrite $K^e_{i,j}$ in the notations of this sections by taking $\varphi_{\,i} = N_{\,i}$ and $\varphi_{\,j} = N_{\,j}$ $and\ \Omega^e = Q_e$ so that

$$K^e_{i,j} = \int_{Q_e} \nabla N_{\,i} \,.\, \nabla N_j \, dx = \int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\} \, dxdy = S^e_{2i-1\,,2j-1} + S^e_{2i\,,2j} \qquad ....(53b\ )$$

DOI: 10.18535/ijecs/v5i10.33

3.2 Computation of $K_{i,j}^{e}$

The explicit integration scheme explained above compute four derivative product integrals as given in eqn(36) and they are necessary to compute the stiffness matrix entries of plane stress/plane strain problems in elasticity and sevral other applications in continuum mechanics.But this computation requires matrix triple product as given in eqn (52).Since,we only need the sum of two of these integrals viz : $\mathrm{S}_{2\mathrm{i}-1,2\mathrm{j}-1}^{\mathrm{e}}+\mathrm{S}_{2\mathrm{i},2\mathrm{j}}^{\mathrm{e}}$ .We now present an efficient method to compute this sum by using matrix product.

Let $F_{\alpha.\beta}^{i,j}=\frac{\partial N_i}{\partial\alpha}\frac{\partial N_j}{\partial\beta}$, $I_{\alpha.\beta}^{i,j}=\iint_{Q_{p,q}}F_{\alpha.\beta}^{i,j}\,dpdq$ then we have from eqns(36-37) :

Where $Q_{p,q}$ *is the quadrilateral in* (p,q)*space, we note that* $Q_{p,q}$*is denoted as* $\mathrm{Q_e}$ *in* (x,y)*space and*

$Q_{p,q}$ *is denoted as* $\widehat{\mathrm{Q}}$ in $(\mathrm{u,v})$space

$$\mathrm{S}^{\mathrm{i,j,e}}=\iint_{\mathrm{Q_e}}\mathrm{G}_{\mathrm{x},\mathrm{y}}^{\mathrm{i,j,e}}\ \mathrm{dx\,dy}=\begin{pmatrix}\iint_{\mathrm{Q_e}}\frac{\partial\mathrm{N_i^e}}{\partial\mathrm{x}}\frac{\partial\mathrm{N_j^e}}{\partial\mathrm{x}}\,\mathrm{dxdy} & \iint_{\mathrm{Q_e}}\frac{\partial\mathrm{N_i^e}}{\partial x}\frac{\partial\mathrm{N_j^e}}{\partial y}\,\mathrm{dxdy}\\ \iint_{\mathrm{Q_e}}\frac{\partial\mathrm{N_i^e}}{\partial y}\frac{\partial\mathrm{N_j^e}}{\partial x}\,\mathrm{dxdy} & \iint_{\mathrm{Q_e}}\frac{\partial\mathrm{N_i^e}}{\partial y}\frac{\partial\mathrm{N_j^e}}{\partial y}\,\mathrm{dxdy}\end{pmatrix}$$

$$=\begin{pmatrix}\mathrm{S}_{2\mathrm{i}-1,2\mathrm{j}-1}^{\mathrm{e}} & \mathrm{S}_{2\mathrm{i}-1,2\mathrm{j}}^{\mathrm{e}}\\ \mathrm{S}_{2\mathrm{i},2\mathrm{j}-1}^{\mathrm{e}} & \mathrm{S}_{2\mathrm{i},2\mathrm{j}}^{\mathrm{e}}\end{pmatrix}\ \text{(say)}$$

$$=\begin{pmatrix}I_{x,x}^{i,j} & I_{x,y}^{i,j}\\ I_{y,x}^{i,j} & I_{y,y}^{i,j}\end{pmatrix}\qquad\text{...(54a)}$$

$$\mathbf{K}^{\mathrm{i,j,e}}=\iint_{\widehat{\mathrm{Q}}}\mathbf{G}_{\mathrm{u},\mathrm{v}}^{\mathrm{i,j,e}}\ \mathbf{du\,dv}=\begin{pmatrix}\iint_{\widehat{\mathrm{Q}}}\frac{\partial\mathrm{N_i^e}}{\partial\mathrm{u}}\frac{\partial\mathrm{N_j^e}}{\partial\mathrm{u}}\,\mathbf{dudv} & \iint_{\widehat{\mathrm{Q}}}\frac{\partial\mathrm{N_i^e}}{\partial\mathrm{u}}\frac{\partial\mathrm{N_j^e}}{\partial\mathrm{v}}\,\mathbf{dudv}\\ \iint_{\widehat{\mathrm{Q}}}\frac{\partial\mathrm{N_i^e}}{\partial\mathrm{v}}\frac{\partial\mathrm{N_j^e}}{\partial\mathrm{u}}\,\mathbf{dudv} & \iint_{\widehat{\mathrm{Q}}}\frac{\partial\mathrm{N_i^e}}{\partial\mathrm{v}}\frac{\partial\mathrm{N_j^e}}{\partial\mathrm{v}}\,\mathbf{dudv}\end{pmatrix}$$

$$=\begin{pmatrix}\mathbf{K}_{2\mathbf{i}-1,2\mathbf{j}-1}^{\mathbf{e}} & \mathbf{K}_{2\mathbf{i}-1,2\mathbf{j}}^{\mathbf{e}}\\ \mathbf{K}_{2\mathbf{i},2\mathbf{j}-1}^{\mathbf{e}} & \mathbf{K}_{2\mathbf{i},2\mathbf{j}}^{\mathbf{e}}\end{pmatrix}\ \textbf{(say)}$$

$$=\begin{pmatrix}I_{u,u}^{i,j} & I_{u,v}^{i,j}\\ I_{v,u}^{i,j} & I_{v,v}^{i,j}\end{pmatrix}\qquad\textbf{..(54b)}$$

Let $P=\begin{pmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{pmatrix}$, $P^T=\begin{pmatrix}P_{11} & P_{21}\\ P_{12} & P_{22}\end{pmatrix}$ **..(55)**

From eqns(44a-b) and (45)

$$\mathbf{S}^{\mathrm{i,j,e}}=\iint_{\mathrm{Q_e}}\mathbf{G}_{\mathrm{x},\mathrm{y}}^{\mathrm{i,j,e}}\ \mathbf{dx\,dy}\ =2\Delta_{\mathrm{abc}}\,\mathbf{P}\left(\iint_{\widehat{\mathrm{Q}}}\mathbf{G}_{\mathrm{u},\mathrm{v}}^{\mathrm{i,j,e}}\,\mathbf{du\,dv}\right)\ \mathbf{P}^{\mathrm{T}}$$

$$=2\Delta_{\mathrm{abc}}\begin{pmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{pmatrix}\begin{pmatrix}I_{u,u}^{i,j} & I_{u,v}^{i,j}\\ I_{v,u}^{i,j} & I_{v,v}^{i,j}\end{pmatrix}\begin{pmatrix}P_{11} & P_{21}\\ P_{12} & P_{22}\end{pmatrix}$$

$$=2\Delta_{\mathrm{abc}}\begin{pmatrix}\{\mathbf{P}_{11}(\mathbf{P}_{11}\mathbf{I}_{\mathrm{u.u}}^{\mathrm{i,j}}+\mathbf{P}_{12}\mathbf{I}_{\mathrm{u,v}}^{\mathrm{i,j}})+\mathbf{P}_{12}(\mathbf{P}_{11}\mathbf{I}_{\mathrm{v,u}}^{\mathrm{i,j}}+\mathbf{P}_{12}\mathbf{I}_{\mathrm{v,v}}^{\mathrm{i,j}})\ \} & \{\mathbf{P}_{11}(\mathbf{P}_{21}\mathbf{I}_{\mathrm{u,u}}^{\mathrm{i,j}}+\mathbf{P}_{22}\mathbf{I}_{\mathrm{u.v}}^{\mathrm{i,j}})+\mathbf{P}_{12}(\mathbf{P}_{21}\mathbf{I}_{\mathrm{v,u}}^{\mathrm{i.j}}+\mathbf{P}_{22}\mathbf{I}_{\mathrm{v.v}}^{\mathrm{i,j}})\}\\ \{\mathbf{P}_{21}(\mathbf{P}_{11}\mathbf{I}_{\mathrm{u,u}}^{\mathrm{i,j}}+\mathbf{P}_{12}\mathbf{I}_{\mathrm{u,v}}^{\mathrm{i,j}})+\mathbf{P}_{22}(\mathbf{P}_{11}\mathbf{I}_{\mathrm{u.u}}^{\mathrm{i,j}}+\mathbf{P}_{12}\mathbf{I}_{\mathrm{v,v}}^{\mathrm{i.j}})\ \} & \{\mathbf{P}_{21}(\mathbf{P}_{21}\mathbf{I}_{\mathrm{u,u}}^{\mathrm{i,j}}+\mathbf{P}_{22}\mathbf{I}_{\mathrm{u,v}}^{\mathrm{i.j}})+\mathbf{P}_{22}(\mathbf{P}_{21}\mathbf{I}_{\mathrm{v,u}}^{\mathrm{i,j}}+\mathbf{P}_{22}\mathbf{I}_{\mathrm{v,v}}^{\mathrm{i,j}})\}\end{pmatrix}$$

..(56)

From eqn(43b) ,eqn(44a) and eqn(46) , we find

trace $(\mathrm{S}^{\mathrm{i,j,e}})$**= trace(**$\iint_{\mathrm{Q_e}}\mathbf{G}_{\mathrm{x},\mathrm{y}}^{\mathrm{i,j,e}}$ **)=(** $\mathbf{S}_{2\mathrm{i}-1,2\mathrm{j}-1}^{\mathrm{e}}+\mathbf{S}_{2\mathrm{i},2\mathrm{j}}^{\mathrm{e}}$**)=** $K_{i,j}^{e}=\int_{Q_e}\nabla\mathrm{N}_{\,i}\,.\,\nabla N_j\,d\mathbf{x}=\int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\}\,dxdy$

$$=(P_{11}^2+P_{21}^2)I_{u,u}^{i,j}+(P_{11}\,P_{12}+P_{21}\,P_{22})\,(I_{u,v}^{i,j}+I_{v,u}^{i,j})+=(P_{12}^2+P_{22}^2)I_{v,v}^{i,j}\qquad\textbf{..........(57)}$$

W e can obtain the above integraql $\int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\}\,dxdy$ **by use of matrix operations which doesnot need the computation matrix triple product.This procedure is presented below**

From eqn (44b) and eqn(45),let us do the following:

$$(P^T\,P)\,.*\begin{pmatrix}I_{u,u}^{i,j} & I_{u,v}^{i,j}\\ I_{v,u}^{i,j} & I_{v,v}^{i,j}\end{pmatrix}=\begin{bmatrix}(P_{11}^2+P_{21}^2)\,I_{u,u}^{i,j} & (\,P_{11}\,P_{12}+P_{21}\,P_{22})I_{u,v}^{i,j}\\ (\,P_{11}\,P_{12}+P_{22}\,P_{21})I_{v,u}^{i,j} & (P_{12}^2+P_{22}^2)I_{v,v}^{i,j}\end{bmatrix}\qquad\textbf{...................................(58)}$$

We observe from eqn(48) that sum of all the entries gives us the value of the integral i.e

$$\int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\}dxdy=(2\Delta_{abc})sum(sum\left(\left(P^T\,P\right).*\begin{pmatrix}I_{u,u}^{i,j} & I_{u,v}^{i,j}\\ I_{v,u}^{i,j} & I_{v,v}^{i,j}\end{pmatrix}\right)) \qquad \text{..........................(59)}$$

Where,sum is aq Matlab function.We note that S=sum(X) gives the sum of the elements of vector X. If X is a matrix then S is a row vector with the sum over each column.It is clear that sum(sum(X)) gives the sum of all the entries in a matrix X .

3.2 Computing of Force Vector Integrals $\int_{\Omega^e} f\varphi_i\ \mathrm{dxdy}$

We shall now propose numerical integration for the complicated integrands in the force vector integrals over the domain Ω^e which is an arbitrary linear triangle and $\phi(x,y) = f\varphi_i$ .We also refer to the section 2 for the theory necessary to derive the composite numerical integration formla
We shall now establish a composite integration formula for an arbitrary linear triangular region Δ PQR shown in Fig 2a or Fig 3a. We have for an arbitrary smooth function ϕ(x, y)

$$\mathrm{II}_{\Delta PQR} = \iint_{\Delta PQR}\boldsymbol{\phi}(\mathbf{x},\mathbf{y})\ \mathbf{dxdy} = \sum_{e=1}^{3}\iint_{Q_e}\boldsymbol{\phi}(\mathbf{x},\mathbf{y})\ \mathbf{dxdy} \qquad \text{--------------------- (60)}$$

$$= \iint_{\hat{Q}}\ \sum_{e=1}^{3}[\boldsymbol{\phi}\ (\mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\mathbf{y}^{(e)}(\mathbf{u},\mathbf{v}))\frac{\partial\left(\mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\mathbf{y}^{(e)}(\mathbf{u},\mathbf{v})\right)}{\partial(\mathbf{u},\mathbf{v})}]\ \mathbf{dudv}$$

$$= (2\ \Delta_{pqr})\ \iint_{\hat{Q}}\ \{\sum_{e=1}^{3}[\boldsymbol{\phi}\ (\mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\ \ \mathbf{y}^{(e)}(\mathbf{u},\mathbf{v}))\ ]\}\ \mathbf{dudv} \qquad \text{-------------------- (61)}$$

Where $(\mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\ \mathbf{y}^{(e)}(\mathbf{u},\mathbf{v})), \mathbf{e} = 1,2,3)$ are the transformations of Eqs.(18)–(20) and $\hat{Q}$ is the quadrilateral in uv- plane spanned by vertices G(1/3 ,1/3) , E(0, ½) , C(0, 0) and F(1/2,0) , and Δ_{pqr} is the area of triangle Δ PQR, Now using the transformations defined in Eqs.(1)–(2) we obtain

$$\mathrm{II}_{\Delta PQR} = (2\ \Delta_{pqr})\ \iint_{\hat{Q}}\ \{\sum_{e=1}^{3}[\boldsymbol{\phi}\ (\mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\ \ \mathbf{y}^{(e)}(\mathbf{u},\mathbf{v}))\frac{\partial(\mathbf{u},\mathbf{v})}{\partial(\xi,\eta)}\}\ \mathbf{d\xi\ d\eta} \qquad \text{--------------- (62)}$$

In Eq.(14) we have used the transformation

$$\mathbf{u}(\xi,\ \eta) = \frac{1}{3}\mathbf{N}_1(\xi,\ \eta) + \frac{1}{2}\mathbf{N}_4(\xi,\ \eta)$$

$$\mathbf{v}(\xi,\ \eta) = \frac{1}{3}\mathbf{N}_1(\xi,\ \eta) + \frac{1}{2}\mathbf{N}_2(\xi,\ \eta) \qquad \text{---------------------- (63)}$$

to map the quadrilateral $\hat{Q}$ into a 2 – square in ξη – plane.
We can now obtain from Eqs.(14)–(15)

$$\mathrm{II}_{\Delta PQR} = (2\ \Delta_{pqr})\int_{-1}^{1}\int_{-1}^{1}\ [\ \sum_{e=1}^{3}\ \left(\frac{4+\xi+\eta}{96}\right)\boldsymbol{\phi}(\ \mathbf{x}^{(e)}(\mathbf{u},\mathbf{v}),\ \ \mathbf{y}^{(e)}(\mathbf{u},\mathbf{v}))\ ]\mathbf{d\xi\ d\eta} \qquad \text{------------- (64)}$$

We can evaluate Eq.(16) either analytically or numerically depending on the form of the integrand.
Using Numerical Integration ;

$$\mathrm{II}_{\Delta PQR}=2\Delta_{pqr}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\frac{W_i^{(N)}W_j^{(N)}(4+\xi_i^{(N)}+\eta_j^{(N)})}{96}\right)\sum_{e=1}^{3}\boldsymbol{\phi}(\mathbf{x}^{(e)}\left(\mathbf{u}_{i,j}^{(N)},\mathbf{v}_{i,j}^{(N)}\right),\mathbf{y}^{(e)}\left(\mathbf{u}_{i,j}^{(N)},\mathbf{v}_{i,j}^{(N)}\right)) \qquad \text{----------------------- (65)}$$

Where,

$$\mathbf{u}_{i,j}^{(N)} = \mathbf{u}(\xi_i^{(N)},\ \eta_j^{(N)})\ \ \text{and}\ \ \mathbf{v}_{i,j}^{(N)} = \mathbf{v}(\xi_i^{(N)},\ \eta_j^{(N)}) \qquad \text{---------------------- (66)}$$

and $(W_i^{(N)},\ \xi_i^{(N)})$, $(W_j^{(N)},\ \xi_j^{(N)})$ are the weight coefficients and sampling points of N^th order Gauss Legendre Quadrature rules.
The above composite rule is applied to numerical Integration over polygonal domains using convex quadrangulation and Gauss Legendre Quadrature Rules[27].

The above method will help in integrating $\int_{\Omega^e} f\varphi_i$ dxdy,when the intgrand $f\varphi_i$ is complicated

4.0 A NEW APPROACH TO MESH GENERATION

The first step in implimenting finite element method isto generate a mesh.In a recent work the author and his co-workers have proposed a new approach to mesh generation which can discretise a convex polygon into an all quadrilateral mesh.This will be presented next.This new approach to mesh generation meets the necessary requirements of regularity on the shape of elements.There are two types of them which usually suffice in finite element computations.The first is called shape regularity. It says that the ratio of the diameter of the element to the radius of the inner circle must be less than some constant. For triangles,the diameter of the triangle is related to the smallest circle which contain the triangle.The inner circle refers to the largest circle which fits inside the triangle. Shape regularity focuses on the shape of individual triangles and doesnot refer to how the shapes of different elements relate to each other. So some elements can be large wshile others might be very small. There is a second type of requirement on the shape of elements.This requirement says that ratio of the maximum diameter of elements to the radius of the inner circle of an element must be less than some constant .If a mesh satisfies this requirement,it is called quasiuniform.This requirement is more important when we perform refinements.We must note that a mesh generation gives us the nodes on a particular element as well as the coordinates of the nodes.We now give an account of this novel mesh generation technique with an aim to use it further in the solution of Poisson problem. Stated in eqn(17a-b).

In our recent paper[ ], the explicit finite element integration scheme is presented by using the isoparametric transformation over the 4 node linear convex quadrilateral element which is applied to torison of square shaft, on considering symmetry of the problem domain, mesh generation for 1/8 of the cross section which is a triangle was discritised into an all quadrilateral mesh. **In this paper we consider applications to polygonal domains**.

4.1 An automatic indirect quadrilateral mesh generator

A wide range of problems in applied science and engineering can be simulated by partial derivative equations(PDE).In the last few decade,one of the most relevant techniques to solve is the Finite Element Method(FEM).It is well known that a good quality mesh is required in order to obtain an accurate solution.Hence the construction of a mesh is on e of the most important steps.

In the next few sections , we present a novel mesh generation scheme of all quadrilateral elements for convex polygonal domains. This scheme converts the elements in background triangular mesh into quadrilaterals through the operation of splitting. We first decompose the convex polygon into simple subregions in the shape of triangles. These simple subregions are then triangulated to generate a fine mesh of triangles. We propose then an automatic triangular to quadrilateral conversion scheme in which each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of edges and a vertex at the barrycentre of the triangular element. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this fully discretizes the given convex polygonal domain into all quadrilaterals, thus propogating uniform refinement. In section 4.2, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of six node triangular elements. In section 4.3, we explain the procedure to split these triangles into quadrilaterals. In section 4.4,we have presented a method of piecing together of all triangular subregions and eventually creating a all quadrilateral mesh for the given convex polygonal domain.

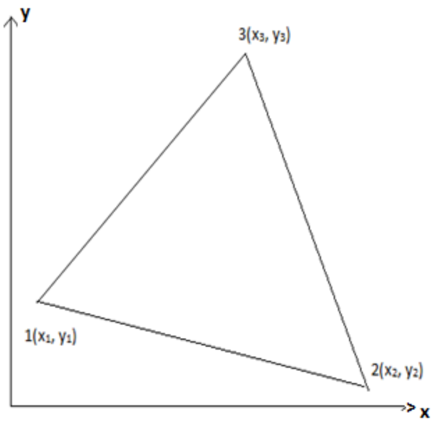
4.2 Division of an Arbitrary Triangle

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. 4a, b. The necessary transformation is given by the equations.

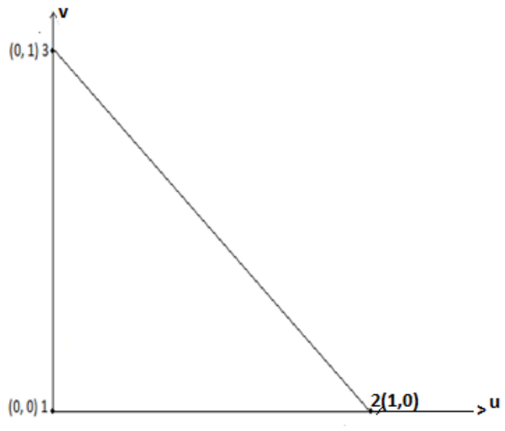
$$x = x_1 + (x_2 - x_1)u + (x_3 - x_1)v$$

$$y = y_1 + (y_2 - y_1)u + (y_3 - y_1)v \qquad (67)$$

The mapping of eqn.(1) describes a unique relation between the coordinate systems. This is illustrated by using the area coordinates and division of each side into three equal parts in Fig. 5a Fig. 5b. 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.



4.a

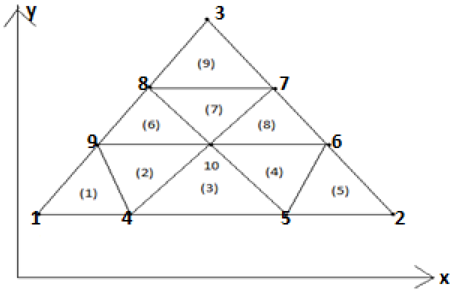


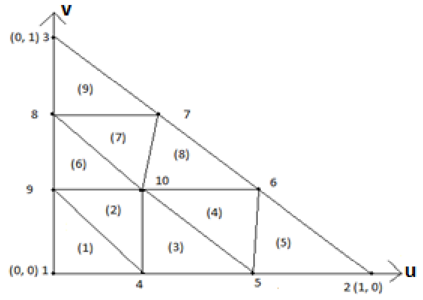
4 b

Fig. 4a An Arbitrary Linear Triangle in the (x, y) space

Fig. 4b A Right Isosceles Triangle in the (u, v) space

DOI: 10.18535/ijecs/v5i10.33





5a

5b

Fig. 5a Division of an arbitrary triangle into Nine triangles in Cartesian space

Fig. 5b Division of a right isosceles triangle into Nine right isosceles triangles in (u, v) space

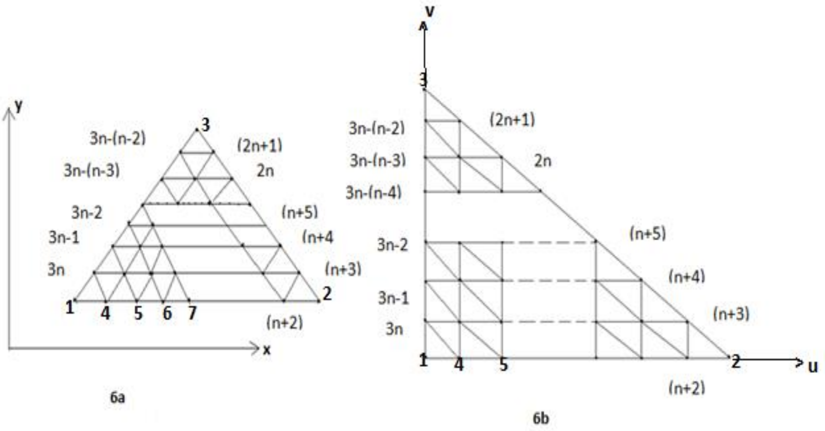


Fig.6a Division of an arbitrary triangle into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisions of equal length

Fig.6b Division of a right isosceles triangle into n^2 right isosceles triangle in (u, v) space, where each side is divided into n divisions of equal length

We have shown the division of an arbitrary triangle in Fig. 6a , Fig. 6b, We divided each side of the triangles (either in Cartesian space or natural space) into n equal parts and draw lines parallel to the sides of the triangles. This creates (n+1) (n+2) nodes. These nodes are numbered from triangle base line l_{12} (letting l_{ij} as the line joining the vertex (x_i, y_i) and (x_j, y_j)) along the line $v = 0$ and upwards up to the line $v = 1$. The nodes 1, 2, 3 are numbered anticlockwise and then nodes 4, 5, ------, (n+2) are along line $v = 0$ and the nodes (n+3), (n+4), ------, 2n, (2n+1) are numbered along the line l_{23} i.e. $u + v = 1$ and then the node (2n+2), (2n+3), -------, 3n are numbered along the line $u = 0$. Then the interior nodes are numbered in increasing order from left to right along the line $v = \frac{1}{n}, \frac{2}{n}, - - -, \frac{n-1}{n}$ bounded on the right by the line $+v = 1$. Thus the entire triangle is covered by (n+1) (n+2)/2 nodes. This is shown in the $\underline{rr}$ matrix of size $(n + 1) \times (n + 1)$, only nonzero entries of this matrix refer to the nodes of the triangles

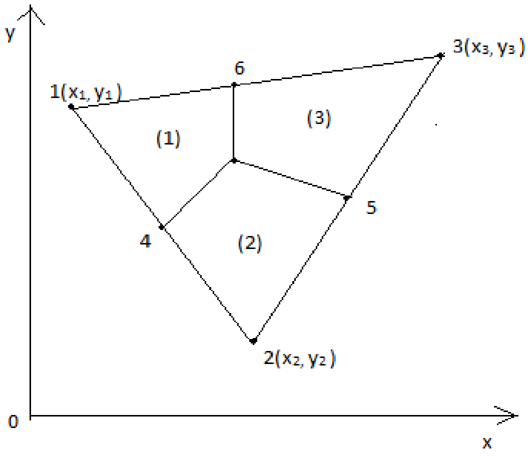
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, & (n+2) & 2 \\ 3n, & (3n+1), \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots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

....(68)

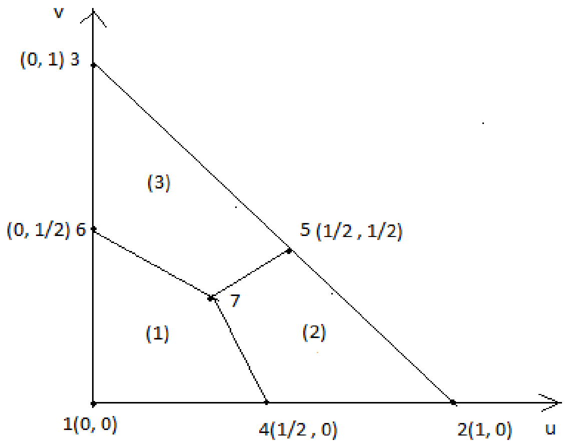
4.3. Quadrangulation of an Arbitrary Triangle

We now consider the quadrangulation of an arbitrary triangle. We first divide the arbitrary triangle into a number of equal size six node triangles. Let us define l_{ij} as the line joining the points (x_i, y_i) and (x_j, y_j) in the Cartesian space (x, y). Then the arbitrary triangle with vertices at $((x_i, y_i), i = 1,2,3)$ is bounded by three lines l_{12}, l_{23}, and l_{31} . By dividing the sides l_{12}, l_{23}, l_{31} into $n = 2m$ divisions (m, an integer) creates m^2 six node triangular divisions. Then by joining the centroid of these six node triangles to the midpoints of their sides, we obtain three quadrilaterals for each of these triangle. We have illustrated this process for the two and four divisions of l_{12}, l_{23}, and l_{31} sides of the arbitrary and standard triangles in Figs. 4 and 5

Two Divisions of Each side of an Arbitrary Triangle



7(a)

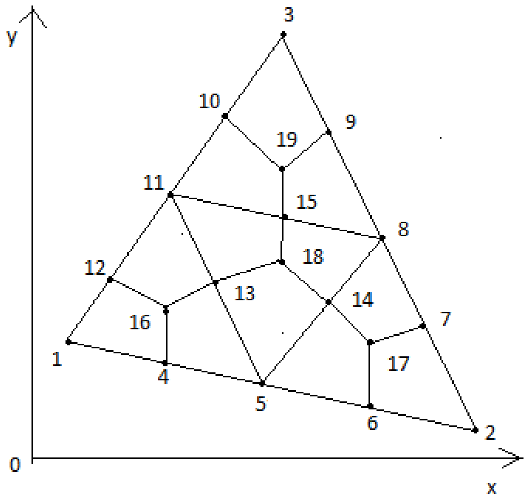


7(b)

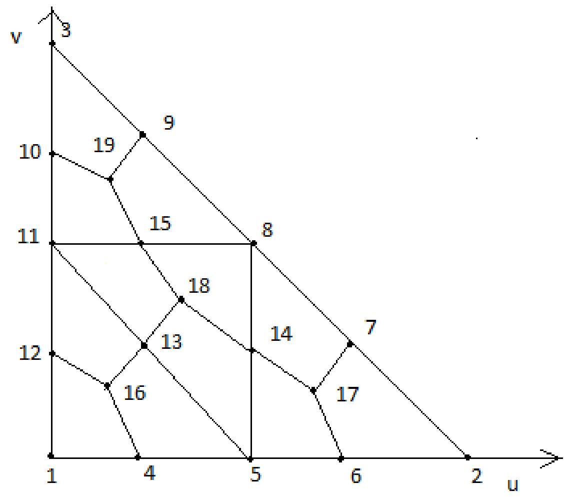
Fig 7(a). Division of an arbitrary triangle into three quadrilaterals

Fig 7(b). Division of a standard triangle into three quadrilaterals

Four Divisions of Each side of an Arbitrary Triangle



8a



8b

Fig 8a. Division of an arbitrary triangle into 4 six node triangles

Fig 8b. Division of a standard triangle into 4 right isosceles triangle

In general, we note that to divide an arbitrary triangle into equal size six node triangle, we must divide each side of the triangle into an even number of divisions and locate points in the interior of triangle at equal spacing. We also do similar divisions and locations of interior points for the standard triangle. Thus n (even) divisions creates (n/2)^2 six node triangles in both the spaces. If the entries of the sub matrix $\underline{rr}\,(i\,;\,i+2,\ j\,;\,j+2)$ are nonzero then two six node triangles can be formed. If $\underline{rr}\,(i+1,\ j+2) = \underline{rr}\ \ (i+2,\ j+1;\,j+2) = 0$ then one six node triangle can be formed. If the sub matrices $\underline{rr}\,(i\,;\,i+2,\ j\,;\,j+2)$ is a (3×3) zero matrix , we cannot form the six node triangles. We now explain the creation of the six node triangles using the $\underline{rr}$ matrix of eqn.( ). We can form six node triangles by using node points of three consecutive rows and columns of $\underline{rr}$ matrix. This procedure is depicted in Fig. 9 for three consecutive rows $i\,, i+1$, $i+2$ and three consecutive columns $j\,,\ j+1,\ j+2$ of the $\underline{rr}$ sub matrix

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

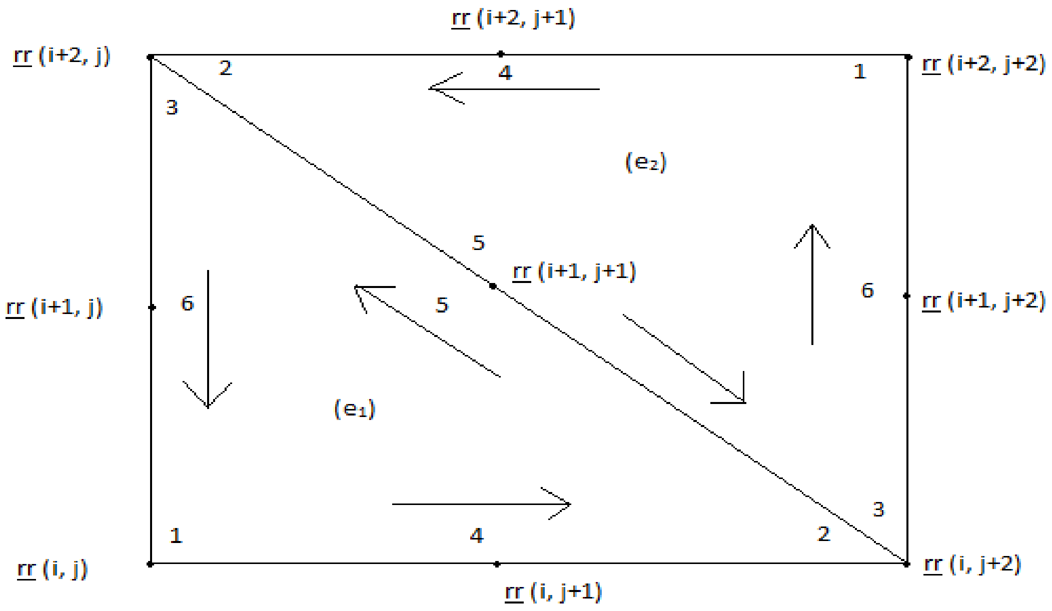


Fig. 9 Six node triangle formation for non zero sub matrix $\underline{rr}$

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

(e$_1$) $<\underline{rr}\ \ (i,\ j), \underline{rr}\,(i, i+2), \underline{rr}\,(i+2,\ j), \underline{rr}\,(i, j+1), \underline{rr}\,(i+1, j+1), \underline{rr}\,(i+1, j) >$

(e$_2$) $< \underline{rr}\ \ (i+2, j+2), \underline{rr}\,(i+2, j), \underline{rr}\,(i, j+2), \underline{rr}\,(i+2, j+1), \underline{rr}\,(i+1, j+1), \underline{rr}\ \ i+1, j+2) >$ ....(69)

If the elements of sub matrix ($(\underline{rr}\ (k,l), k = i, i+1, i+2),\ l = j,\ j+1,\ j+2)$ are nonzero, then as standard earlier, we can construct two six node triangles. We can create three quadrilaterals in each of these six node triangles. The nodal connectivity for the 3 quadrilaterals created in (e_1) are given as

Q_{3n_1-2} < c_1 , $\underline{rr}\ (i+1,\ j),\ \underline{rr}\,(i,\ j)$, $\underline{rr}\,(i, j+1)$ >

Q_{3n_1-1} < c_1 , $\underline{rr}\ (i,\ j+1),\ \underline{rr}\,(i,\ j+2)$, $\underline{rr}\,(i+1, j+1)$ >

Q_{3n_1} < c_1 , $\underline{rr}\ (i+1,\ j+1),\ \underline{rr}\,(i+2,\ j)$, $\underline{rr}\,(i+1, j)$ > ....(70)

and the nodal connectivity for the 3 quadrilaterals created in (e_2) are given as

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

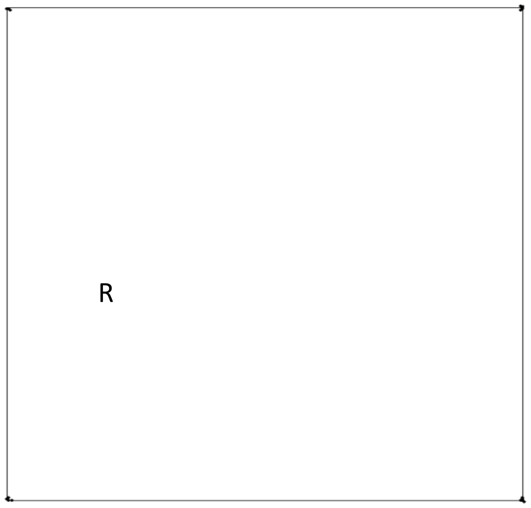
Q_{3n_2-1} < c_2 , $\underline{rr}\ (i+2,\ j+1),\ \underline{rr}\,(i+2,\ j)$, $\underline{rr}\,(i+1,\ j+1)$ >

Q_{3n_1} < c_2 , $\underline{rr}\ (i+1,\ j+1),\ \underline{rr}\,(i,\ j+2)$, $\underline{rr}\,(i+1, j+2)$ > -------------------- (71)

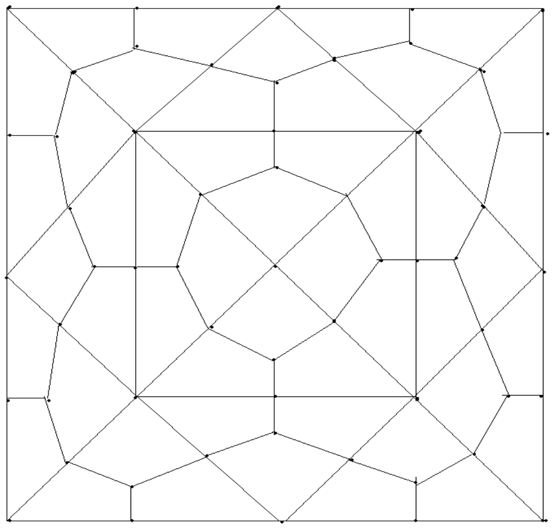
4.4 Quadrangulation of the Polygonal Domain

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

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

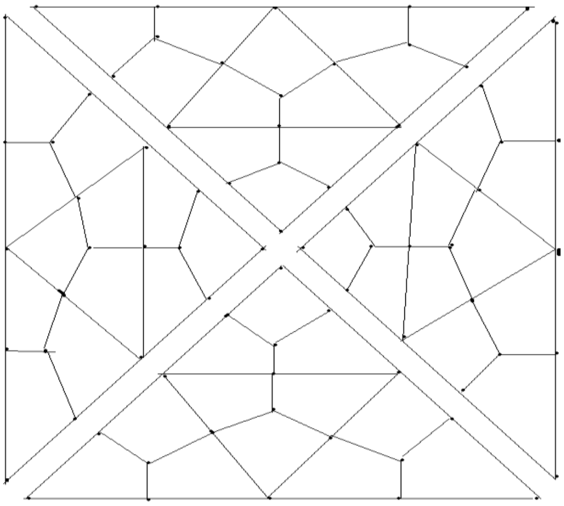


10a

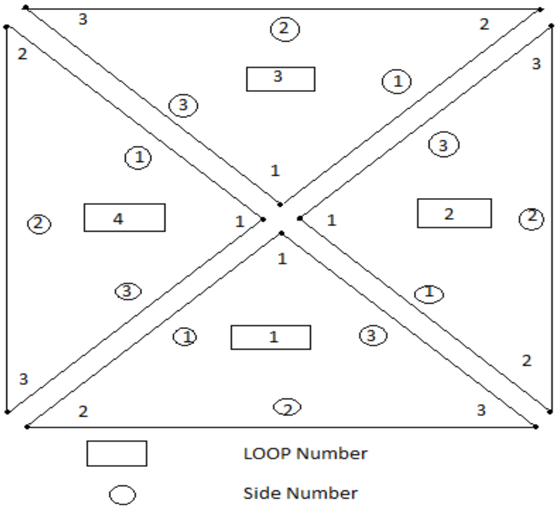


10b

(i)Fig.10a: Region R to be analyzed (ii) Fig.10b: Example of completed mesh



10c



10d

(iii)Fig.10c:Exploded view showing four loops (iv)Fig.10d:Example of a loop and side numbering scheme

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

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

In the present mesh generation code, we aim to create a convex polygon. This requires a simple procedure. We join side 3 0f LOOP 1 to side 1 of LOOP 2, side 3 of LOOP 2 will joined to side 1 of LOOP 3, side 3 of LOOP 3 will be joined to side 1 of LOOP 4. Finally side 3 of LOOP 4 will be joined to side 1 of LOOP 1.

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

5.0 Application Examples

Let us use the explicit integration scheme and the auto mesh generation techniques which are developed in the previous sections to solve the Poisson Equation with Dirichlet boundary value problem:

$\nabla^2 u = -1,\ x\epsilon\Omega \subset \mathcal{R}^2$ ....(9)

$u = 0, \quad x\epsilon\partial\Omega$ ....(10)

Where Ω is a regular polygonal domain and ∇^2 is the standard Laplace operator

An analytic solution for a regular polygon with n sides was developed by Seth in 1934 [].He does not give any numerical results.The solution the torsion problem of uniform bars with polygon cross section by means of the Trefftgz integral for the complex torsion is presented by Hassenpflug[29]

We would like to consider the domain Ω for the linear elastic torsion of a equilateral triangle, a square and the regular polygons with sides n= 5,6,7,8,9,10,..........,18,19,20;which are inscribed in a circle of unit radius.Using the basic results given in Hassenpflug[29],we obtain the following table of analytical values which are correct at least upto five significant digits.We use these values of torisonal constants to compare our computed solution given by finite element technique developed in this paper.We also refer to the recent work of Abdelkader[ ]et al who has found FEM solution by using a mesh of rectangles to discretise the domain Ω ,they have done the analysis for regular polygons up to n=14.They had access to exact solutions only for n=3,4,6,8 only.

TABLE-1.3

Exact Values of Torsonal Constant T for Regular Polygons

NO. OF SIDES	TORISONAL CONSTANTS	
	T (5-DIGIT)	T (ASYMPTOTIC EXPANSION)
REFERENCES:	HASSENPFLUG[29]	& OTHERS[30,31]

3	0.194855625	0.194855715851499
4	0.56232	0.562326022367581
5	0.844754404592026	0.844731922422194
6	1.03545	1.03549324087029
7	1.1640752644062	1.16411801946753
8	1.25312	1.25312573223516
9	1.31660156796861	1.3166153234439
10	1.36322309722253	1.36321828704708
11	1.39833824986799	1.39830995233626
12	1.42533	1.42533025866039
13	1.44661919299975	1.44654464667697
14	1.46355914690879	1.46348634544873
15	1.47728014426494	1.47721934132055
16	1.48855363238248	1.48849890460815
17	1.49787447622668	1.49787220869486
18	1.50570404053051	1.50574300731012
19	1.51241473591888	1.5124141944363
20	1.51812391171354	1.51811642128267

In the finite element studies conducted in this paper,we developed MATLAB Programs to discritise the mesh and compute the element stiffness matrix by using explicit integration scheme explained in the previous sections of this paper.The following programs are written to generate the all quadrilateral finite element meshes ,to compute the element stiffness matrices using explicit integration scheme and to draw contour level curve of computed and exact values of Prandtl stress function. If necessary other physical parameters can be computed by making small changes in the listed programs.

(1)D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria.m
(2)D2PoissonEquationQ4MoinEx_MeshgridContour.m
(3)quadrilateral_mesh4MOINEX_q4.m
(4)polygonal_domain_coordinates.m
(5)nodaladdresses_special_convex_quadrilaterals.m
(6)nodaladdresses_special_convex_quadrilaterals_trial.m
(7)generate_area_coordinate_over_the_standard_triangle.m

Conclusions:

This paper proposes the explicit integration scheme for a unique linear convex quadrilateral which can be obtained from an arbitrary linear triangle by joining the centroid to the midpoints of sides of the triangle. The explicit integration scheme proposed for these unique linear convex quadrilaterals is derived by using the standard transformations in two steps. We first map an arbitrary triangle into a standard right isosceles triangle by using a affine linear transformation from global (x, y) space into a local space (u, v). We discritise this standard right isosceles triangle in (u, v) space into three unique linear convex quadrilaterals. We have shown that any unique linear convex quadrilateral in (x, y) space can be mapped into one of the unique quadrilaterals in (u, v) space. We can always map these linear convex quadrilaterals into a 2-aquare in (ξ, η) space by an approximate transformation. Using these two mapping, we have established an integral derivative product relation between the linear convex quadrilaterals in the (x, y) space interior to the arbitrary triangle and the linear convex quadrilaterals of the local space (u, v) interior to the standard right isosceles triangle. Further , we have shown that the product of global derivative integrals $S^{i,j,e}$ in (x, y) space can be expressed as a matrix triple product P $(K^{i,j,e})$ P^T X (2 * area of the arbitrary triangle in (x, y) space) in which P is a geometric properties matrix and $K^{i,j,e}$ is the product of global derivative integrals in (u, v) space.We have shown that the explicit integration of the product of local derivative integrals in (u, v) space over the unique quadrilateral spanning vertices (1/3 ,1/3), (0, ½), (0, 0) , (1/2, 0) is now possible by use of symbolic processing capabilities in MATLAB which are based on Maple – V software package. The proposed explicit integration scheme is a useful technique for boundary value problems governed by either a single equation or a system of second order partial differential equations. The physical applications of such problems are numerous in science, and engineering, the examples of single equations are the well known Laplace and Poisson equations with suitable boundary conditions and the examples of the system of equations are plane stress, plane stress and axisymmetric stress analysis etc in linear elasticity. We have demonstrated the proposed explicit integration scheme to solve the Poisson Boundary Value Problem for a linear elastic torsion of a non-circular bar for cases of equilateral triangler, a square and regular polygon profiles of pentagon(5-gon)to icosagon(20-gon). Monotonic convergence from below is observed with known analytical solutions for the governing unknown function of Poisson Boundary Value Problem.We have shown the solutions in Tables which list both the FEM and exact solutions.The graphical solutions of contour level curves are also displayed.

REFERENCES:

[1] Zienkiewicz O.C, Taylor R.L and J.Z Zhu, Finite Element Method, its basis and fundamentals, Elservier, (2005)

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

[3] Reddy J.N, Finite Element Method, Third Edition, Tata Mc Graw-Hill (2005)

[4] Burden R.L and J.D Faires, Numerical Analysis, 9th Edition, Brooks/Cole, Cengage Learning (2011)

[5] Stroud A.H and D.Secrest, Gaussian quadrature formulas, Prentice Hall,Englewood Cliffs, N J, (1966)

[6] Stoer J and R. Bulirsch, Introduction to Numerical Analysis, Springer-Verlag, New York (1980)

[7] Chung T.J, Finite Element Analysis in Fluid Dynamics, pp.191-199, Mc Graw Hill, Scarborough, C A , (1978)

[8] Rathod H.T, Some analytical integration formulae for four node isoparametric element, Computer and structures 30(5), pp.1101-1109, (1988)

[9] Babu D.K and G.F Pinder, Analytical integration formulae for linear isoparametric finite elements, Int. J. Numer. Methods Eng 20, pp.1153-1166

[10] Mizukami A, Some integration formulas for four node isoparametric element, Computer Methods in Applied Mechanics and Engineering. 59 pp. 111-121(1986)

[11] Okabe M, Analytical integration formulas related to convex quadrilateral finite elements, Computer methods in Applied mechanics and Engineering. 29, pp.201-218 (1981)

[12] Griffiths D.V, Stiffness matrix of the four node quadrilateral element in closed form, International Journal for Numerical Methods in Engineering. 28, pp.687- 703(1996)

[13] Rathod H.T and Md. Shafiqul Islam, Integration of rational functions of bivariate polynomial numerators with linear denominators over a (-1,1) square in a local parametric two dimensional space, Computer Methods in Applied Mechanics and
Engineering. 161 pp.195-213 (1998)

[14] Rathod H.T and Md. Sajedul Karim, An explicit integration scheme based on recursion and matrix multiplication for the linear convex quadrilateral elements, International Journal of Computational Engineering Science. 2(1) pp. 95- 135(2001)

[15] Yagawa G, Ye G.W and S. Yoshimura, A numerical integration scheme for finite element method based on symbolic manipulation, International Journal for Numerical Methods in Engineering. 29, pp.1539-1549 (1990)

[16] Rathod H.T and Md. Shafiqul Islam , Some pre-computed numeric arrays for linear convex quadrilateral finite elements, Finite Elements in Analysis and Design 38, pp. 113-136 (2001)

[17] Hanselman D and B. Littlefield, Mastering MATLAB 7 , Prentice Hall, Happer Saddle River, N J . (2005)

[18] Hunt B.H, Lipsman R.L and J.M Rosenberg , A Guide to MATLAB for beginners and experienced users, Cambridge University Press (2005)

[19] Char B, Geddes K, Gonnet G, Leong B, Monagan M and S.Watt , First Leaves; A tutorial Introduction to Maple V , New York : Springer–Verlag (1992)

[20] Eugene D, Mathematica , Schaums Outlines Theory and Problems, Tata Mc Graw Hill (2001)

[21] Ruskeepaa H, Mathematica Navigator, Academic Press (2009)

[22] Timoshenko S.P and J.N Goodier , Theory of Elasticity, 3rd Edition, Tata Mc
Graw Hill Edition (2010)

[23] Budynas R.G, Applied Strength and Applied Stress Analysis, Second Edition,
Tata Mc Graw Hill Edition (2011)

[24] Roark R.J, Formulas for stress and strain, Mc Graw Hill, New York (1965)

[25] Nguyen S.H, An accurate finite element formulation for linear elastic torsion
calculations, Computers and Structures. 42, pp.707-711 (1992)

[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] 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)

[28] RathodH.T, Bharath Rathod, Shivaram K.T , H. Y. Shrivalli , Tara Rathod ,K. Sugantha Devi , An explicit finite element integration scheme usingautomatic mesh generation technique for linearconvex quadrilaterals over plane regions International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 3 Issue 4 April, 2014 Page No. 5400-5435

[29]Hassenpflug. W.C Torsion of uniform bars with polygon cross section,Computers and Mathematics with Applications,46,(2003),313-392

[30] Abdelkader,K, Toufik.Z and Mohamed,B, Torsional stress in non-circular cross sections by the finite element method, Advances in Mechanical Engineering,2015, Vol. 7(5) 1–20

[31] .Francu,J,.Novackova.P.Janicek,P,Torsion of NonCircular Bar, *Engineering MECHANICS, Vol. 19, 2012, No. 1, p. 45–60* **[32]Seth,B.R,** Torsion of beams whose cross-section is a regular polygon of n sides, Mathematical Proceedings of the Cambridge Philosophical Society, **Vol 30**, **2** , 1934, pp. 139-149

[33] Rathod H.T,Sugantha Devi.K, Finite Element Solution of Poisson Equation over Polygonal Domains using an Explicit Integration Scheme and a novel Auto Mesh Generation Technique *International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume 5 Issues 8 Aug 2016, Page No. 17397-17481*

TABLES ON TORSION CONSTANT VALUES

TABLE-A(EQUILATERAL TRIANGLE)
EACH SIDE LENGTH=2*sqrt(3) units
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
7	3	1.8722884497	3.1176914536	1.2454030039
19	12	2.5125294041	3.1176914536	0.6051620495

37	27	2.8307089573	3.1176914536	0.2869824963
61	48	2.9539197663	3.1176914536	0.1637716873
91	75	3.0125356171	3.1176914536	0.1051558365
127	108	3.0446697570	3.1176914536	0.0730216966
169	147	3.0641026510	3.1176914536	0.0535888026
217	192	3.0767211933	3.1176914536	0.0409702604
271	243	3.0853673364	3.1176914536	0.0323241172
331	300	3.0915454781	3.1176914536	0.0261459755

TABLE-B(EQUILATERAL TRIANGLE)
EACH SIDE LENGTH=sqrt(3) units
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
7	3	0.1170180281	0.1948557159	0.0778376877
19	12	0.1570330878	0.1948557159	0.0378226281
27		0.1769193098	0.1948557159	0.0179364060
61	48	0.1846199854	0.1948557159	0.0102357305
91	75	0.1882834761	0.1948557159	0.0065722398
127	108	0.1902918598	0.1948557159	0.0045638560
169	147	0.1915064157	0.1948557159	0.0033493002
217	192	0.1922950746	0.1948557159	0.0025606413
271	243	0.1928354585	0.1948557159	0.0020202573
331	300	0.1932215924	0.1948557159	0.0016341235

TABLE-C(SQUARE)
EACH SIDE LENGTH=sqrt(2) units
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
17	12	0.5108825061	0.5623080598	0.0514255537
57	48	0.5426868571	0.5623080598	0.0196212027
121	108	0.5529117923	0.5623080598	0.0093962675
209	192	0.5568853490	0.5623080598	0.0054227108
321	300	0.5587965664	0.5623080598	0.0035114934

DOI: 10.18535/ijecs/v5i10.33

```
   457  432        0.5598541695       0.5623080598       0.0024538904
_____________________________________________________________________
   617  588        0.5604985015       0.5623080598       0.0018095583
_____________________________________________________________________
   801  768        0.5609193581       0.5623080598       0.0013887017
___________________________________________________________________
  1009  972        0.5612090954       0.5623080598       0.0010989645
_____________________________________________________________________
  1241 1200        0.5614169335       0.5623080598       0.0008911263
=====================================================================
```

```
                    TABLE-D(REGULAR PENTAGON)
                    TORISON CONSTANT VALUES
==================================================================
nodes elements        FEM SOL        EXACT SOL        absolute error
==================================================================
   21   15        0.7852864069       0.8447544046       0.0594679977
_____________________________________________________________________
   71   60        0.8222617215       0.8447544046       0.0224926830
_____________________________________________________________________
  151  135        0.8339760011       0.8447544046       0.0107784035
_____________________________________________________________________
  261  240        0.8385247648       0.8447544046       0.0062296398
_____________________________________________________________________
  401  375        0.8407165724       0.8447544046       0.0040378321
_____________________________________________________________________
  571  540        0.8419312399       0.8447544046       0.0028231647
_____________________________________________________________________
  771  735        0.8426720465       0.8447544046       0.0020823581
_____________________________________________________________________
 1001  960        0.8431562810       0.8447544046       0.0015981236
_____________________________________________________________________
 1261 1215        0.8434898303       0.8447544046       0.0012645743
_____________________________________________________________________
 1551 1500        0.8437291903       0.8447544046       0.0010252143
```

```
                        TABLE-E(REGULAR HEXAGON)
                         TORISON CONSTANT VALUES
=================================================================
nodes elements        FEM SOL        EXACT SOL      absolute error
==================================================================
   25   18        0.9740536279       1.0383137600       0.0642601321
_____________________________________________________________________
   85   72        1.0122110516       1.0383137600       0.0261027084
_____________________________________________________________________
  181  162        1.0242966106       1.0383137600       0.0140171494
_____________________________________________________________________
  313  288        1.0289990132       1.0383137600       0.0093147468
_____________________________________________________________________
  481  450        1.0312680826       1.0383137600       0.0070456774
_____________________________________________________________________
  685  648        1.0325268439       1.0383137600       0.0057869161
_____________________________________________________________________
  925  882        1.0332950925       1.0383137600       0.0050186675
_____________________________________________________________________
 1201 1152        1.0337975197       1.0383137600       0.0045162403
_____________________________________________________________________
 1513 1458        1.0341437261       1.0383137600       0.0041700339
_____________________________________________________________________
 1861 1800        1.0343922338       1.0383137600       0.0039215262
_____________________________________________________________________
```

DOI: 10.18535/ijecs/v5i10.33

TABLE-F(REGULAR HEPTAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
29	21	1.1021611292	1.1635500000	0.0613888708
99	84	1.1407369769	1.1635500000	0.0228130231
211	189	1.1528272355	1.1635500000	0.0107227645
365	336	1.1575527040	1.1635500000	0.0059972960
561	525	1.1598387647	1.1635500000	0.0037112353
799	756	1.1611090892	1.1635500000	0.0024409108
1079	1029	1.1618853234	1.1635500000	0.0016646766
1401	1344	1.1623934236	1.1635500000	0.0011565764
1765	1701	1.1627437763	1.1635500000	0.0008062237
2171	2100	1.1629953931	1.1635500000	0.0005546069

TABLE-G(REGULAR OCTAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
33	24	1.1906026600	1.2551051100	0.0645024500
113	96	1.2298195763	1.2551051100	0.0252855337
241	216	1.2418363822	1.2551051100	0.0132687278
417	384	1.2465517251	1.2551051100	0.0085533849
641	600	1.2488398525	1.2551051100	0.0062652575
913	864	1.2501140486	1.2551051100	0.0049910614
1233	1176	1.2508938883	1.2551051100	0.0042112217
1601	1536	1.2514049781	1.2551051100	0.0037001319
2017	1944	1.2517577388	1.2551051100	0.0033473712
2481	2400	1.2520112881	1.2551051100	0.0030938219

TABLE-H(REGULAR NONAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
37	27	1.2530412269	1.3161370000	0.0630957731
127	108	1.2933408715	1.3161370000	0.0227961285

271	243	1.3053214277	1.3161370000	0.0108155723
469	432	1.3100308422	1.3161370000	0.0061061578
721	675	1.3123222046	1.3161370000	0.0038147954
1027	972	1.3136009194	1.3161370000	0.0025360806
1387	1323	1.3143848371	1.3161370000	0.0017521629
1801	1728	1.3148992939	1.3161370000	0.0012377061
2269	2187	1.3152547736	1.3161370000	0.0008822264
2791	2700	1.3155105157	1.3161370000	0.0006264843

TABLE-I(REGULAR DECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
41	30	1.2980448713	1.3629210000	0.0648761287
141	120	1.3398529342	1.3629210000	0.0230680658
301	270	1.3518756601	1.3629210000	0.0110453399
521	480	1.3565984423	1.3629210000	0.0063225577
801	750	1.3589004314	1.3629210000	0.0040205686
1141	1080	1.3601873621	1.3629210000	0.0027336379
1541	1470	1.3609775134	1.3629210000	0.0019434866
2001	1920	1.3614967238	1.3629210000	0.0014242762
2521	2430	1.3618558774	1.3629210000	0.0010651226
3101	3000	1.3621145034	1.3629210000	0.0008064966

TABLE-J(REGULAR HENDECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
45	33	1.3310471277	1.3980270000	0.0669798723
155	132	1.3747112826	1.3980270000	0.0233157174
331	297	1.3868630424	1.3980270000	0.0111639576
573	528	1.3916236869	1.3980270000	0.0064033131
881	825	1.3939459214	1.3980270000	0.0040810786
1255	1188	1.3952458097	1.3980270000	0.0027811903
1695	1617	1.3960448916	1.3980270000	0.0019821084

```
 2201 2112        1.3965705432       1.3980270000       0.0014564568

2773 2673        1.3969345015       1.3980270000       0.0010924985

3411 3300        1.3971968087       1.3980270000       0.0008301913
==================================================================


                       TABLE-K(REGULAR DODECAGON)
                        TORISON CONSTANT VALUES
===============================================================
nodes elements        FEM SOL         EXACT SOL      absolute error
================================================================
   49   36        1.3555791239       1.4245820000       0.0690028761

169  144        1.4013615683       1.4245820000       0.0232204317

361  324        1.4137236723       1.4245820000       0.0108583277

625  576        1.4185470940       1.4245820000       0.0060349060

961  900        1.4208995630       1.4245820000       0.0036824370

 1369 1296        1.4222173115       1.4245820000       0.0023646885

1849 1764        1.4230280816       1.4245820000       0.0015539184

2401 2304        1.4235618777       1.4245820000       0.0010201223

3025 2916        1.4239317665       1.4245820000       0.0006502335

3721 3600        1.4241985386       1.4245820000       0.0003834614
==================================================================

                       TABLE-L(REGULAR TRIDECAGON)
                        TORISON CONSTANT VALUES
================================================================
nodes elements        FEM SOL         EXACT SOL      absolute error
================================================================
 53    39        1.3739990071       1.4466450000       0.0726459929

183  156        1.4220851164       1.4466450000       0.0245598836

391  351        1.4347279926       1.4466450000       0.0119170074

677  624        1.4396372290       1.4466450000       0.0070077710

1041  975        1.4420293764       1.4466450000       0.0046156236

1483 1404        1.4433696248       1.4466450000       0.0032753752

2003 1911        1.4441946777       1.4466450000       0.0024503223

2601 2496        1.4447382089       1.4466450000       0.0019067911

3277 3159        1.4451150711       1.4466450000       0.0015299289

4031 3900        1.4453870278       1.4466450000       0.0012579722
```

DOI: 10.18535/ijecs/v5i10.33

TABLE-L(REGULAR TETRADECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
57	42	1.3879236167	1.4635300000	0.0756063833
197	168	1.4384342067	1.4635300000	0.0250957933
421	378	1.4514160427	1.4635300000	0.0121139573
729	672	1.4564312472	1.4635300000	0.0070987528
1121	1050	1.4588715289	1.4635300000	0.0046584711

TABLE-M(REGULAR PENTADECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
61	45	1.3984886179	1.4772800000	0.0787913821
211	180	1.4514918675	1.4772800000	0.0257881325
451	405	1.4648590116	1.4772800000	0.0124209884
781	720	1.4699971311	1.4772800000	0.0072828689
1201	1125	1.4724928297	1.4772800000	0.0047871703

TABLE-N(REGULAR HEXADECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
65	48	1.4065090843	1.4885300000	0.0820209157
225	192	1.4620310253	1.4885300000	0.0264989747
481	432	1.4758191034	1.4885300000	0.0127108966
833	768	1.4810939150	1.4885300000	0.0074360850
1281	1200	1.4836510988	1.4885300000	0.0048789012

TABLE-O(REGULAR HEPTADECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
69	51	1.4125811889	1.4978700000	0.0852888111
239	204	1.4706146014	1.4978700000	0.0272553986

511	459	1.4848498624	1.4978700000	0.0130201376
885	816	1.4902721877	1.4978700000	0.0075978123
1361	1275	1.4928957515	1.4978700000	0.0049742485

TABLE-P(REGULAR OCTADECAGON)
TORISON CONSTANT VALUES

nodes	elements	FEM SOL	EXACT SOL	absolute error
73	54	1.4171482341	1.5057400000	0.0885917659
253	216	1.4776600718	1.5057400000	0.0280799282
541	486	1.4923607369	1.5057400000	0.0133792631
937	864	1.4979387377	1.5057400000	0.0078012623
1441	1350	1.5006324859	1.5057400000	0.0051075141

TABLE-Q(REGULAR ENNEADECAGON)
TORISON CONSTANT VALUES

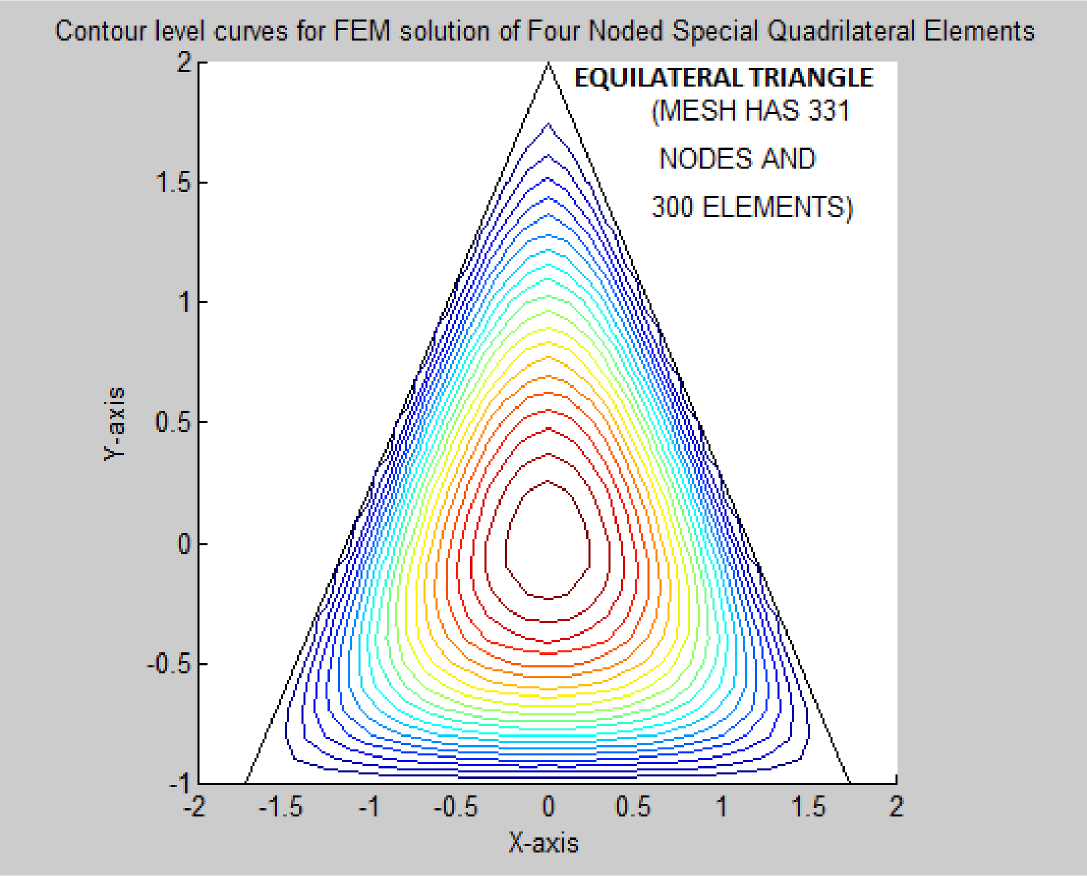
nodes	elements	FEM SOL	EXACT SOL	absolute error
77	57	1.4205445657	1.5124147000	0.0918701343
267	228	1.4834820991	1.5124147000	0.0289326009
571	513	1.4986596184	1.5124147000	0.0137550816
989	912	1.5043991164	1.5124147000	0.0080155836
1521	1425	1.5071658685	1.5124147000	0.0052488315

TABLE-R(REGULAR ICOSAGON)
TORISON CONSTANT VALUES

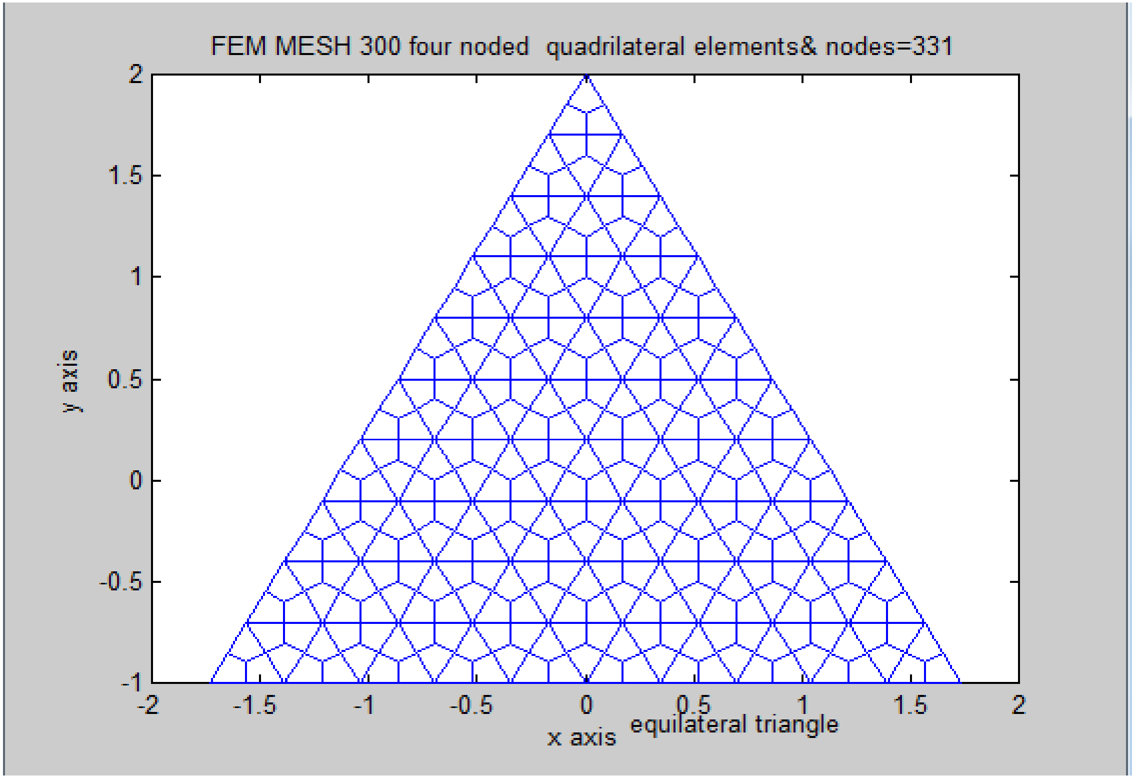
nodes	elements	FEM SOL	EXACT SOL	absolute error
81	60	1.4230254369	1.5181239000	0.0950984631
281	240	1.4883213032	1.5181239000	0.0298025968
601	540	1.5039814835	1.5181239000	0.0141424165
1041	960	1.5098862774	1.5181239000	0.0082376226
1601	1500	1.5127279802	1.5181239000	0.0053959198

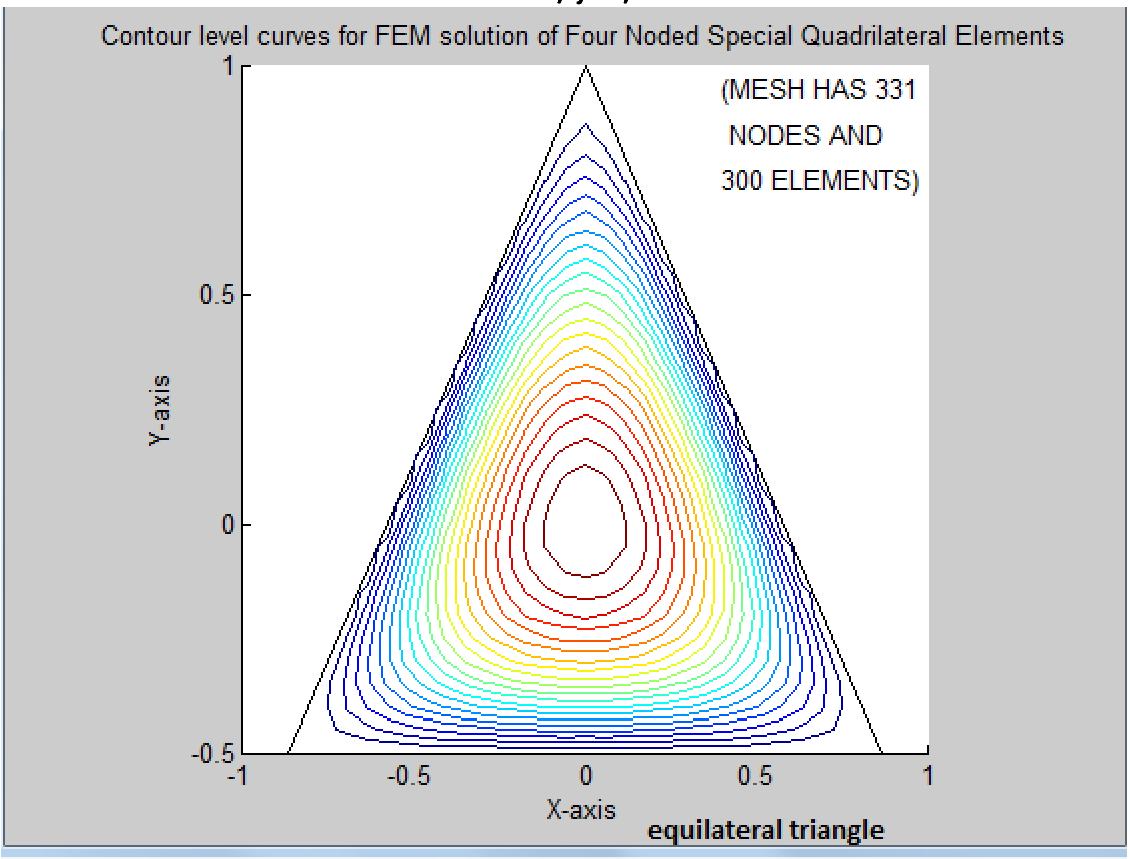
FIGURES

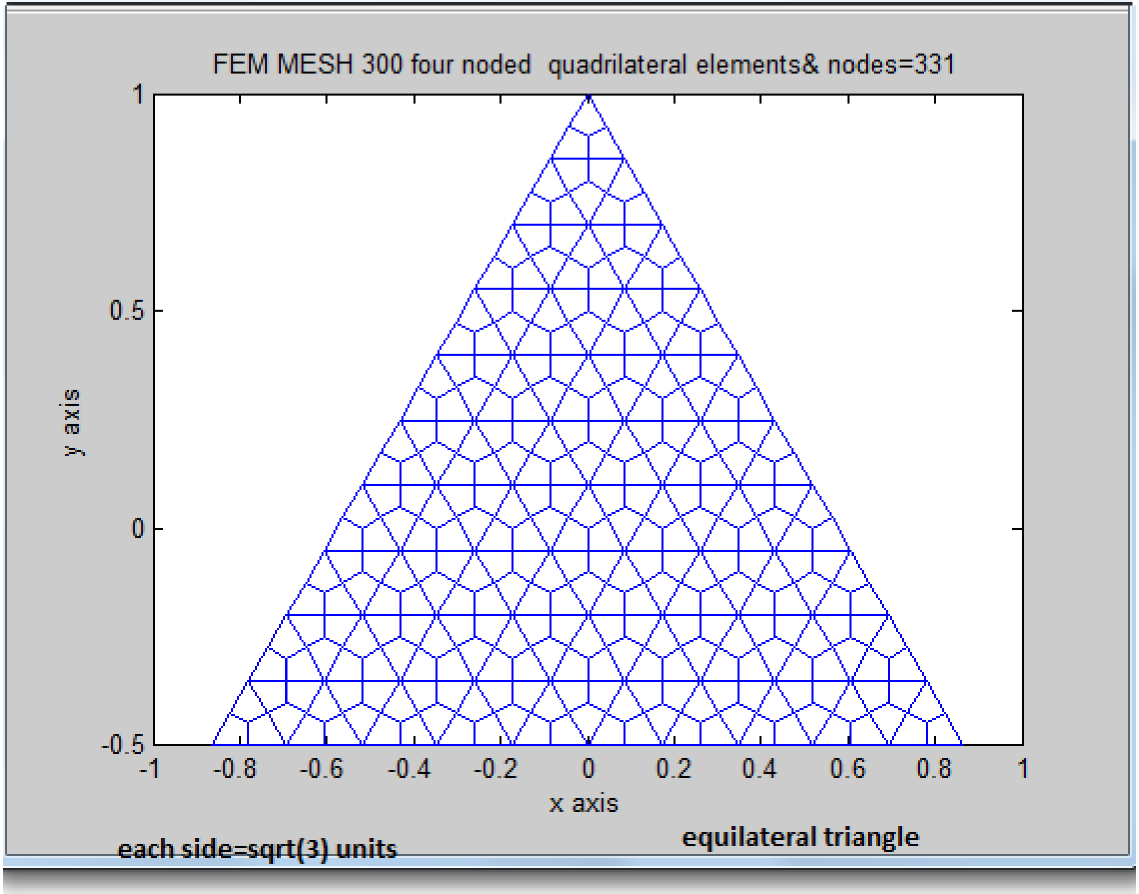
FIGURES:CONTOUR LEVEL CURVES AND FEM MESHES

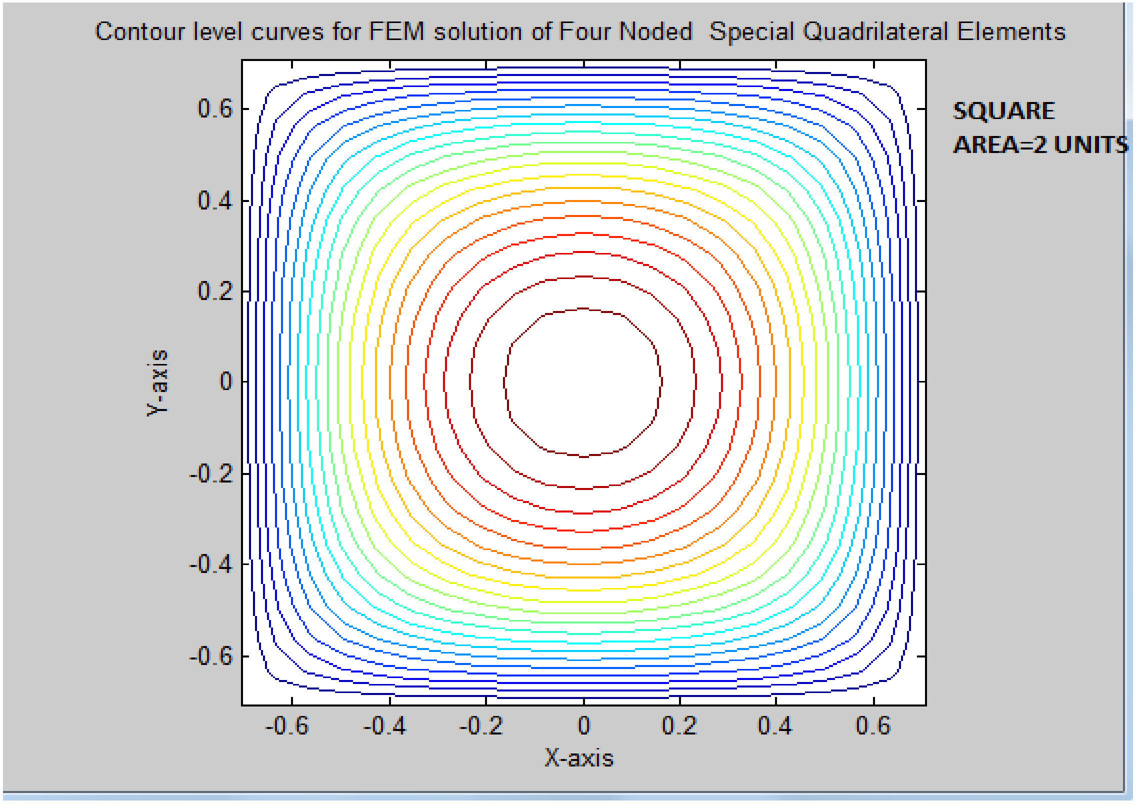

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
EQUILATERAL TRIANGLE
(MESH HAS 331
NODES AND
300 ELEMENTS)
Y-axis
X-axis
2
1.5
1
0.5
0
-0.5
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

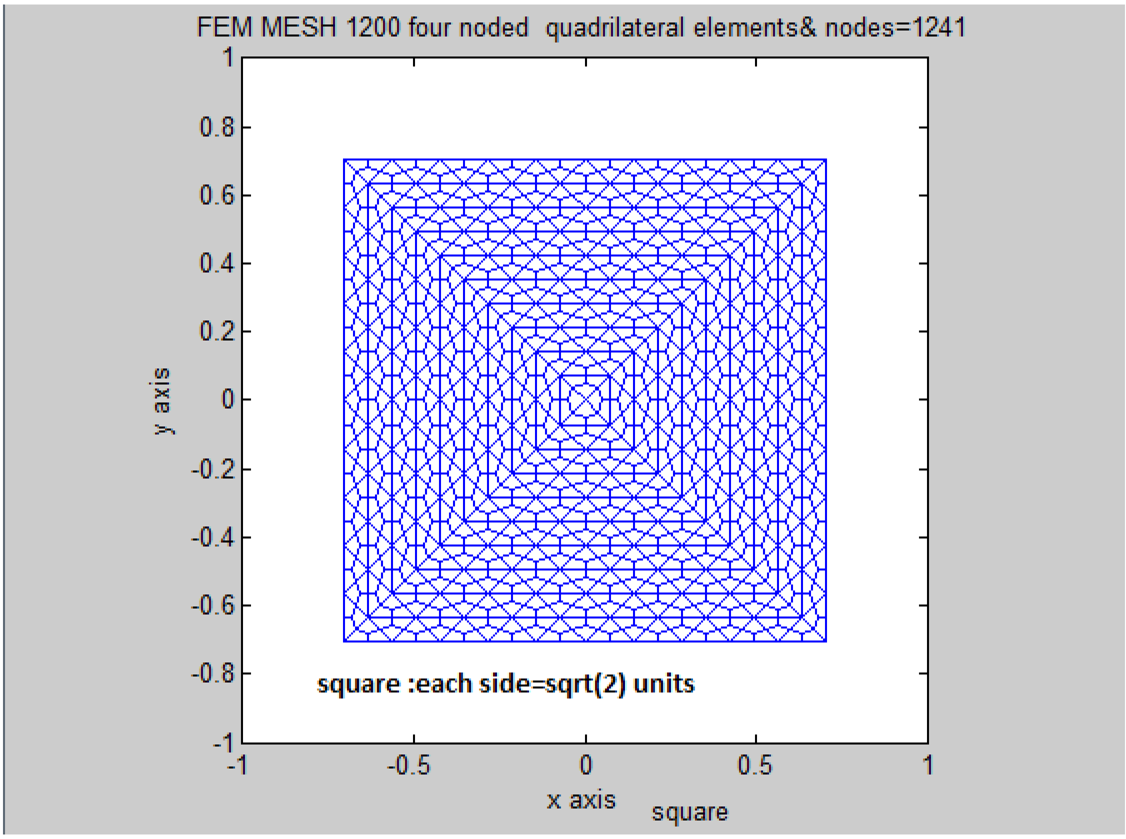
DOI: 10.18535/ijecs/v5i10.33



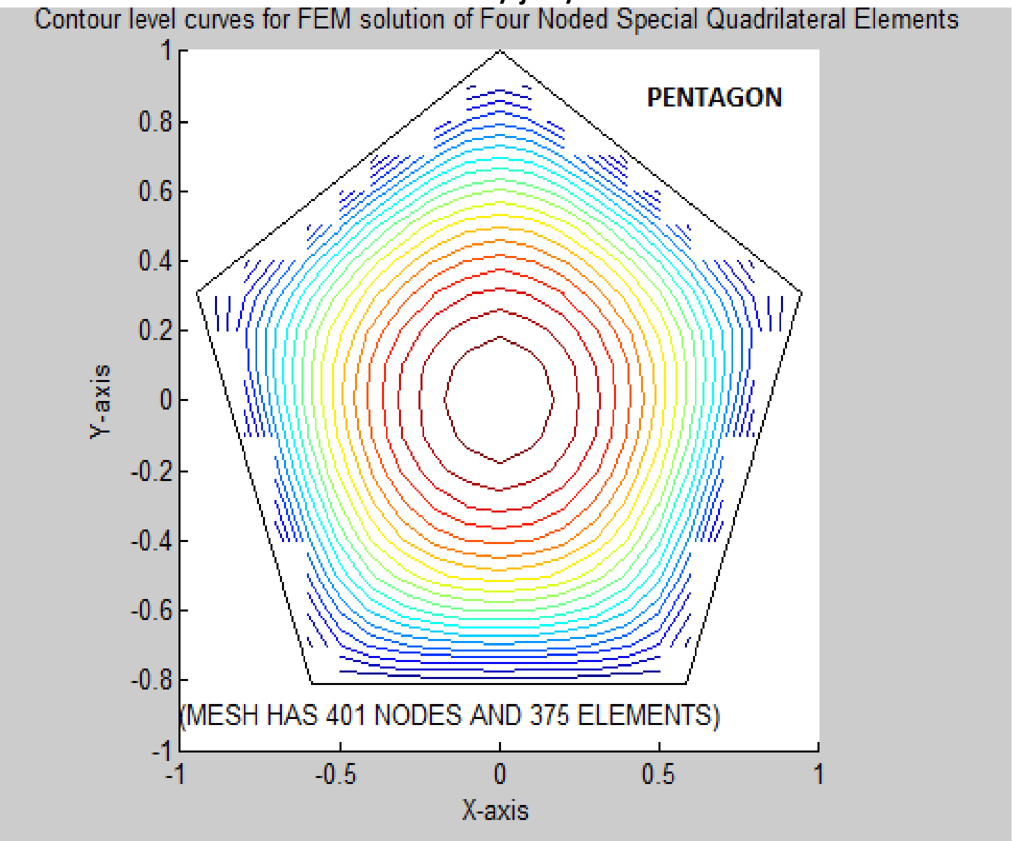

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
(MESH HAS 331
NODES AND
300 ELEMENTS)
Y-axis
1
0.5
0
-0.5
-1
-0.5
0
0.5
1
X-axis
equilateral triangle

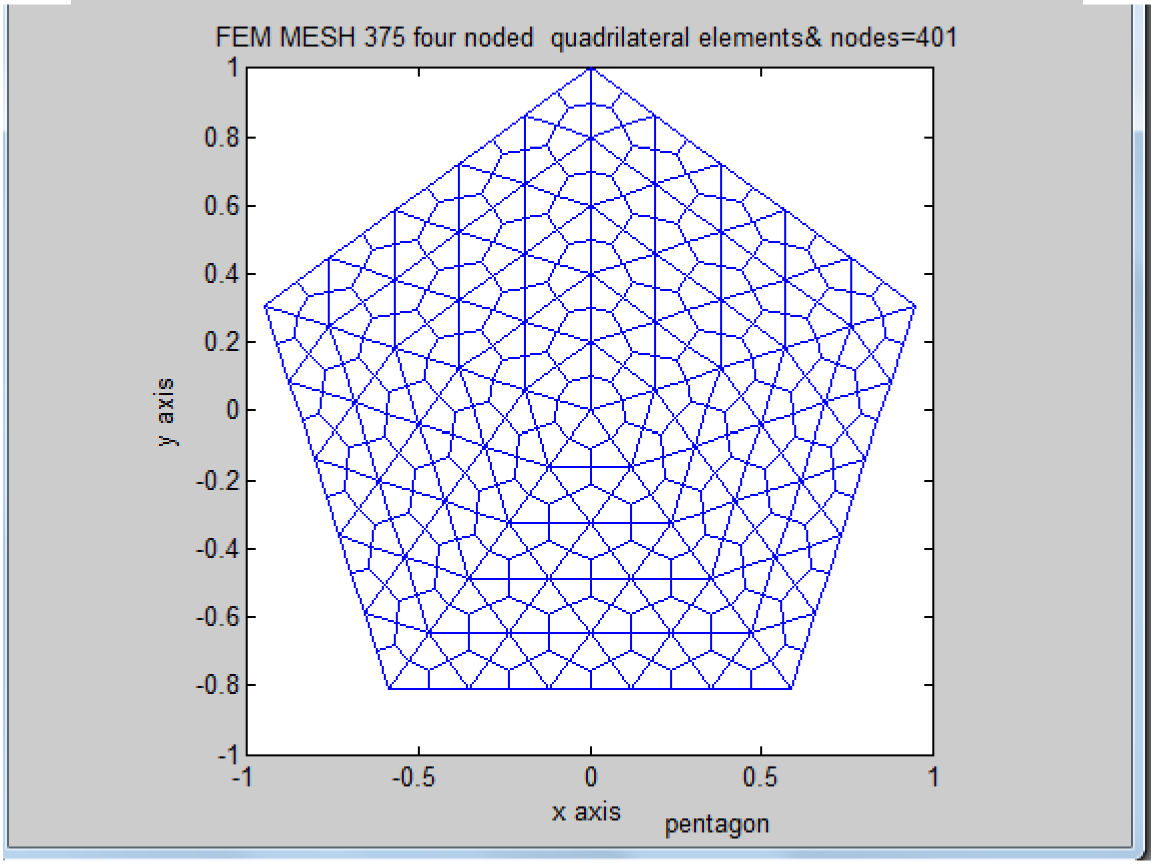

FEM MESH 300 four noded quadrilateral elements& nodes=331
y axis
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
x axis
each side=sqrt(3) units
equilateral triangle


Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
SQUARE
AREA=2 UNITS
Y-axis
X-axis
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

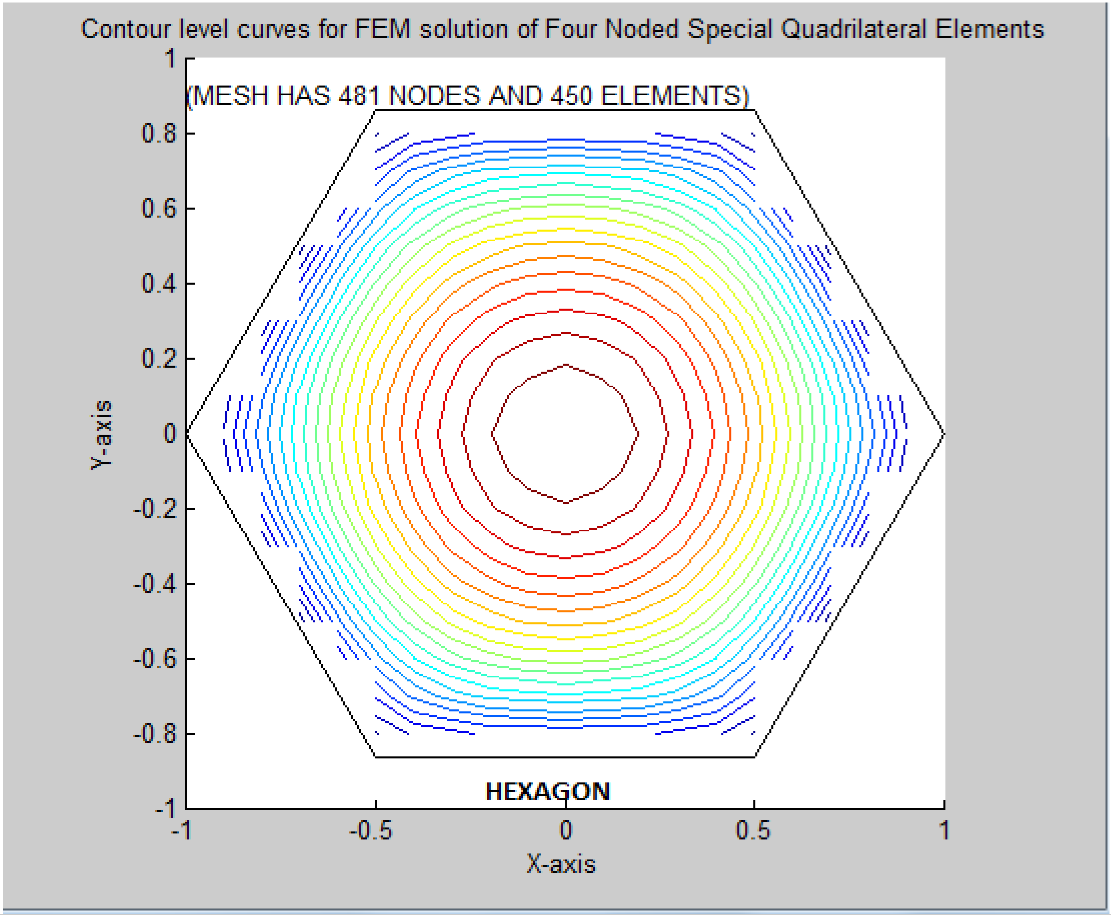

FEM MESH 1200 four noded quadrilateral elements& nodes=1241
square :each side=sqrt(2) units
y axis
x axis
square
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

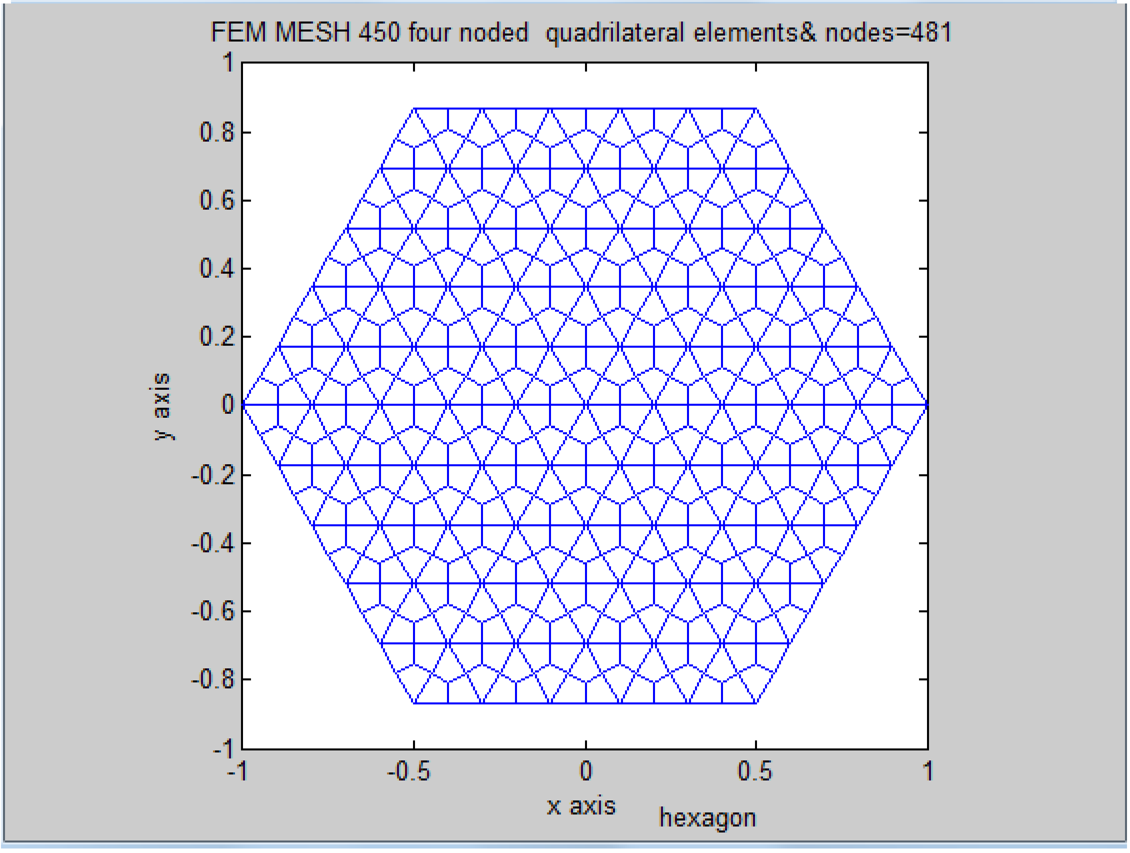
DOI: 10.18535/ijecs/v5i10.33

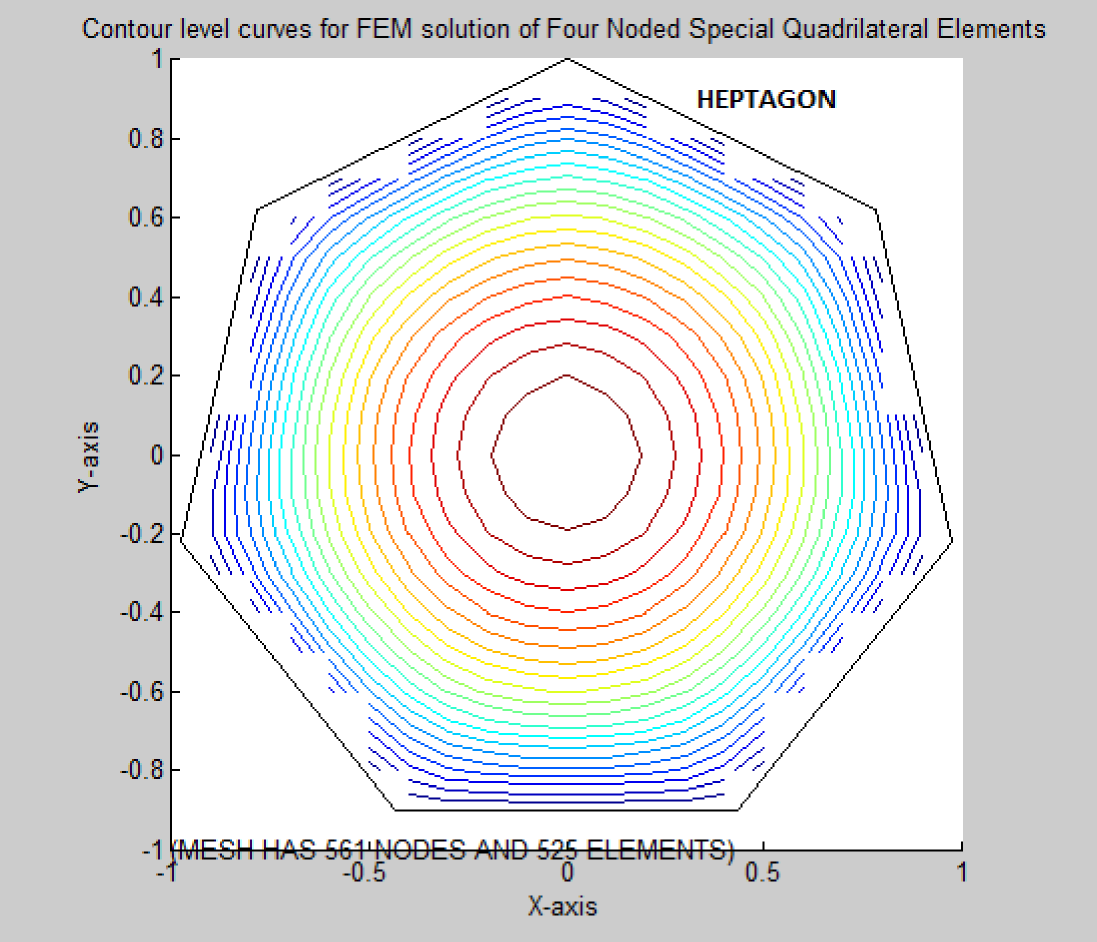


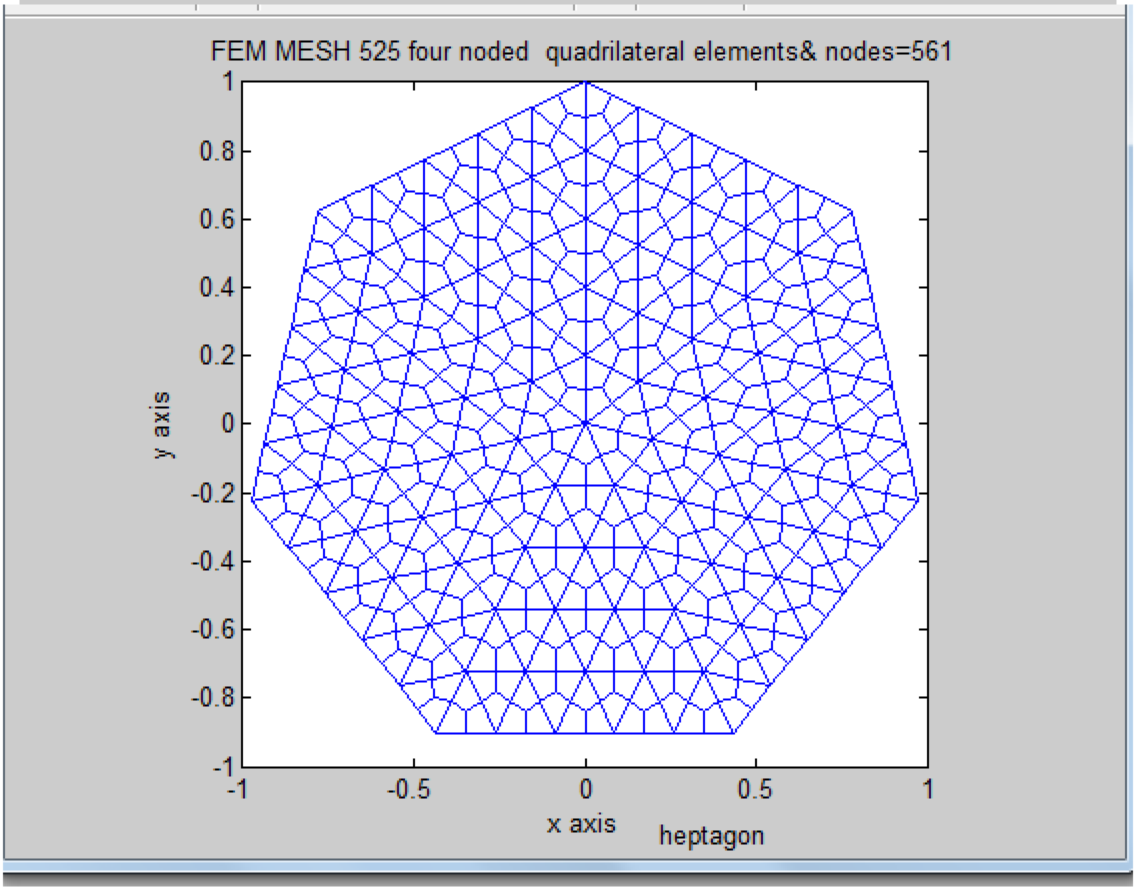


DOI: 10.18535/ijecs/v5i10.33

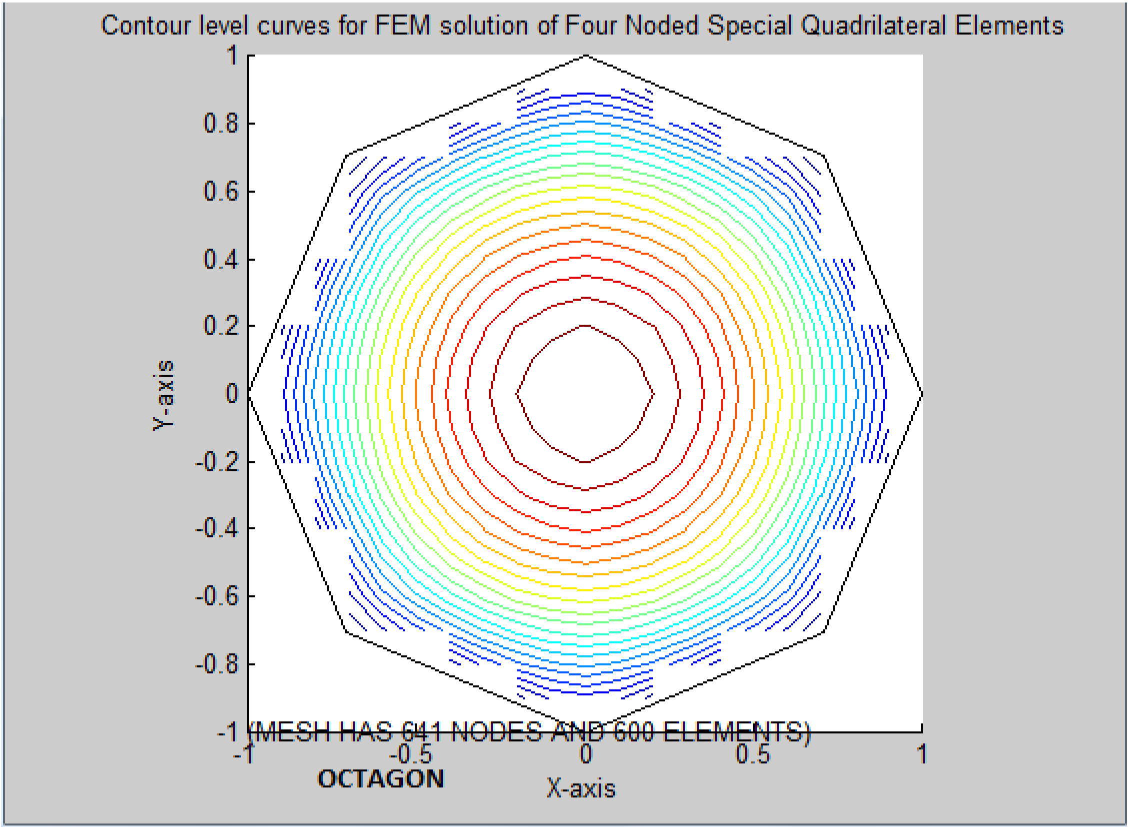


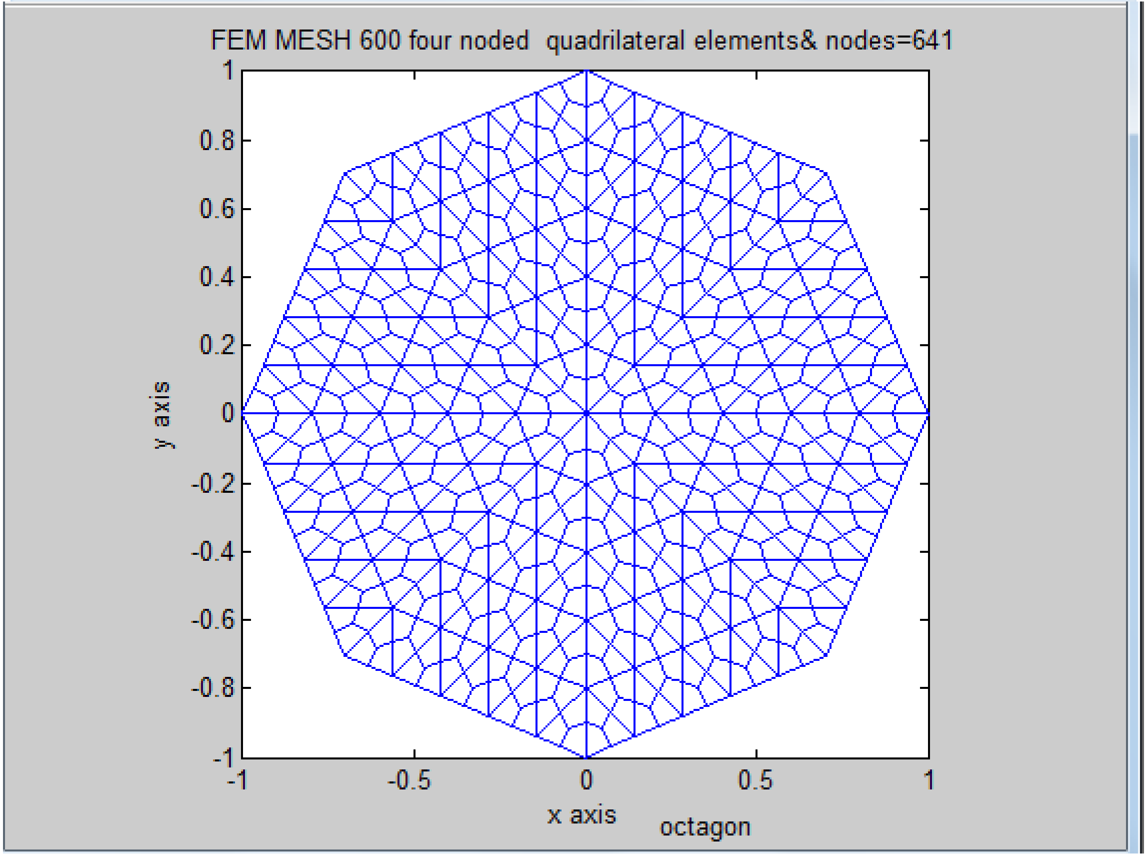


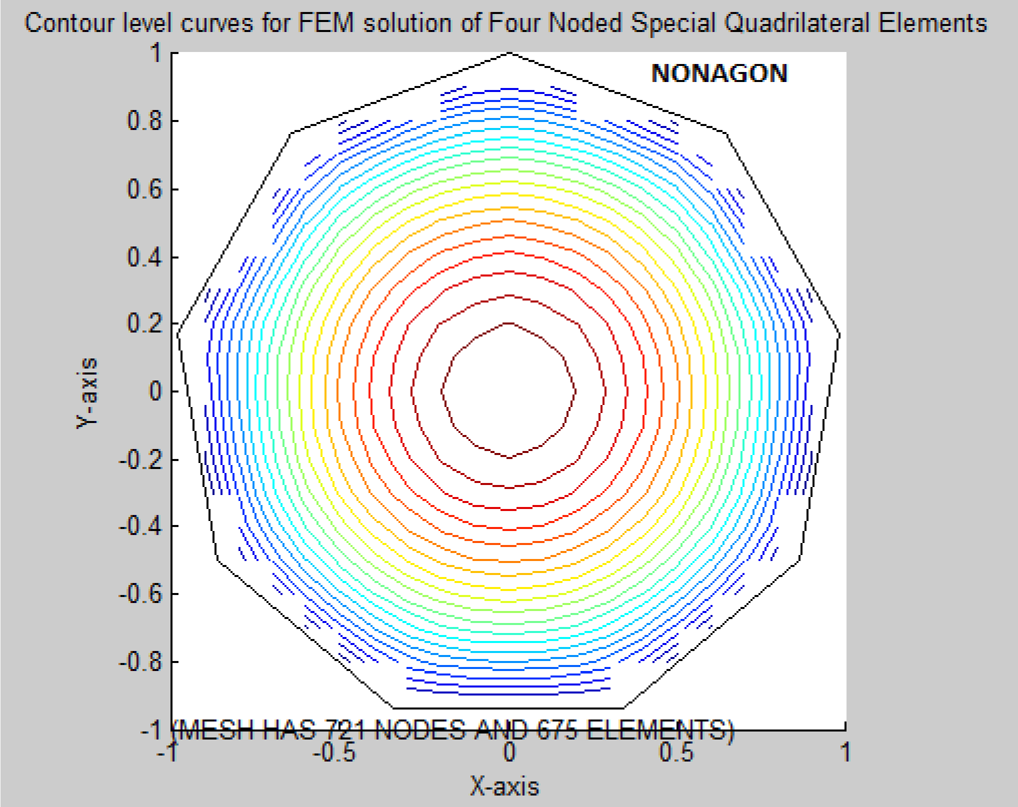

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
HEPTAGON
Y-axis
X-axis
(MESH HAS 561 NODES AND 525 ELEMENTS)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

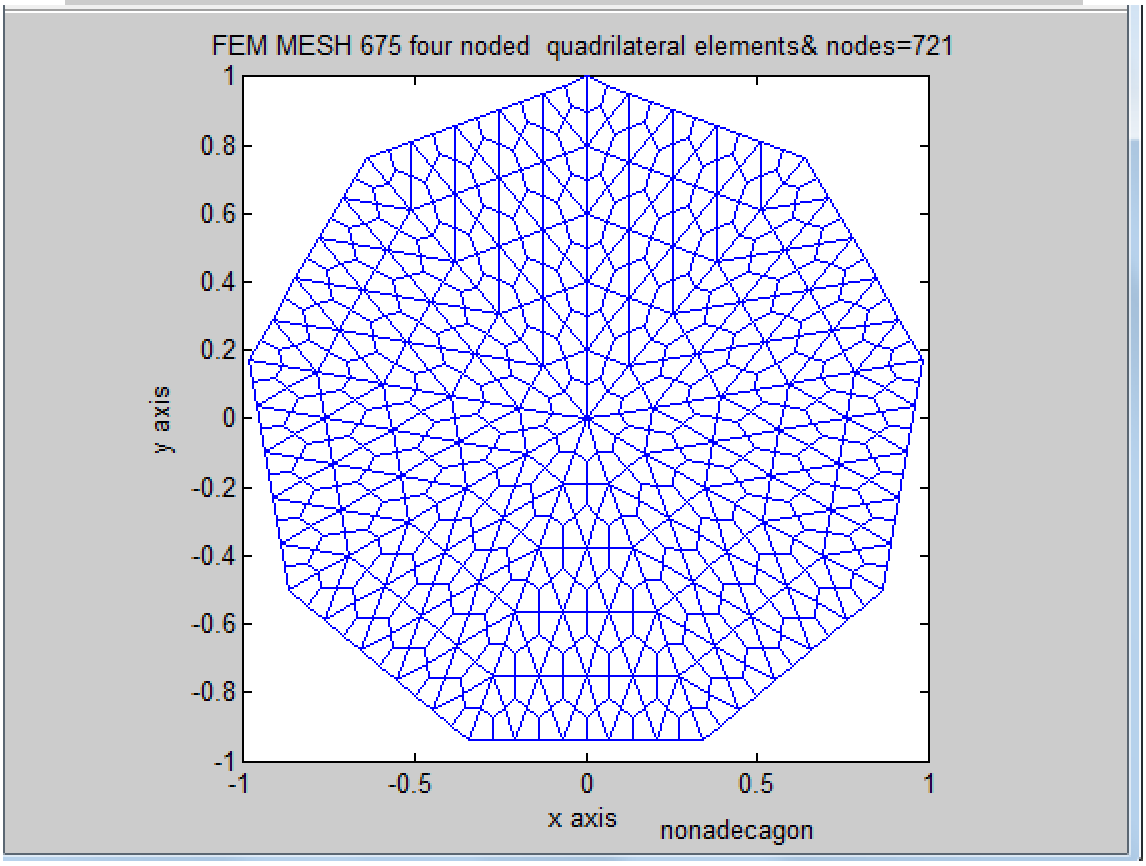

FEM MESH 525 four noded quadrilateral elements& nodes=561
y axis
x axis
heptagon
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

DOI: 10.18535/ijecs/v5i10.33

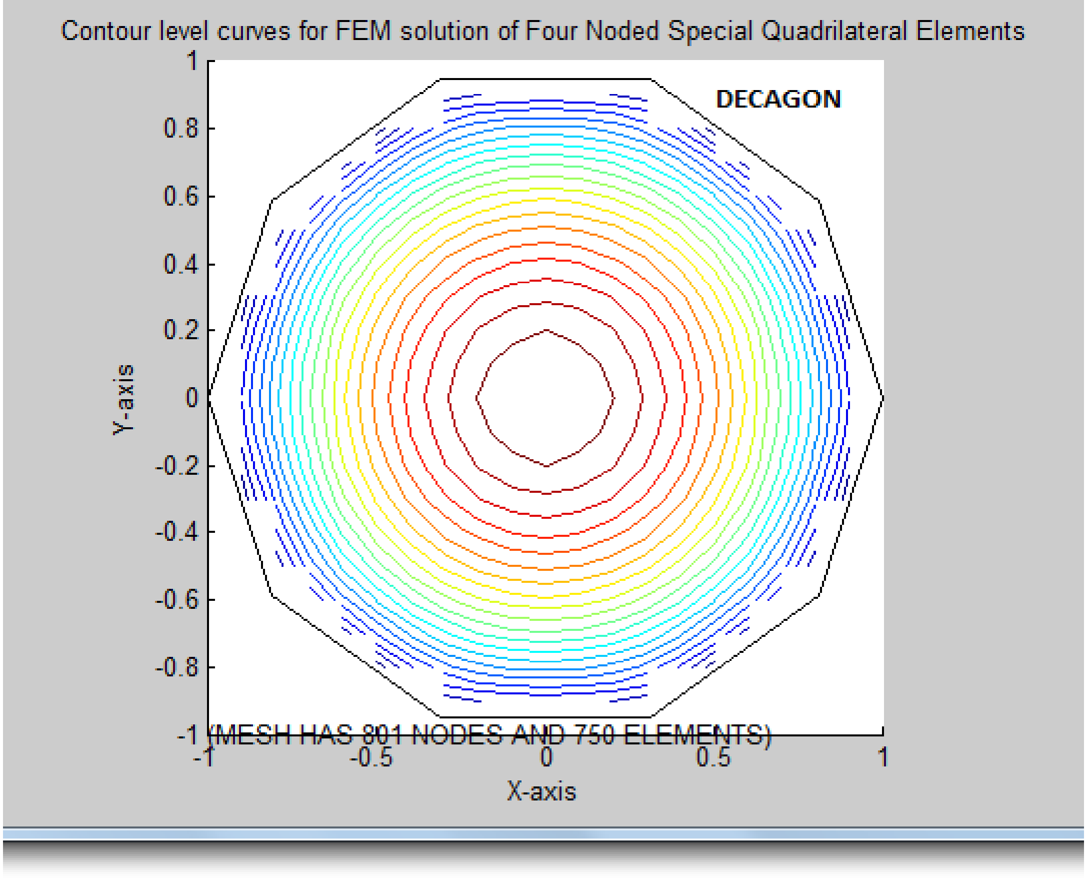


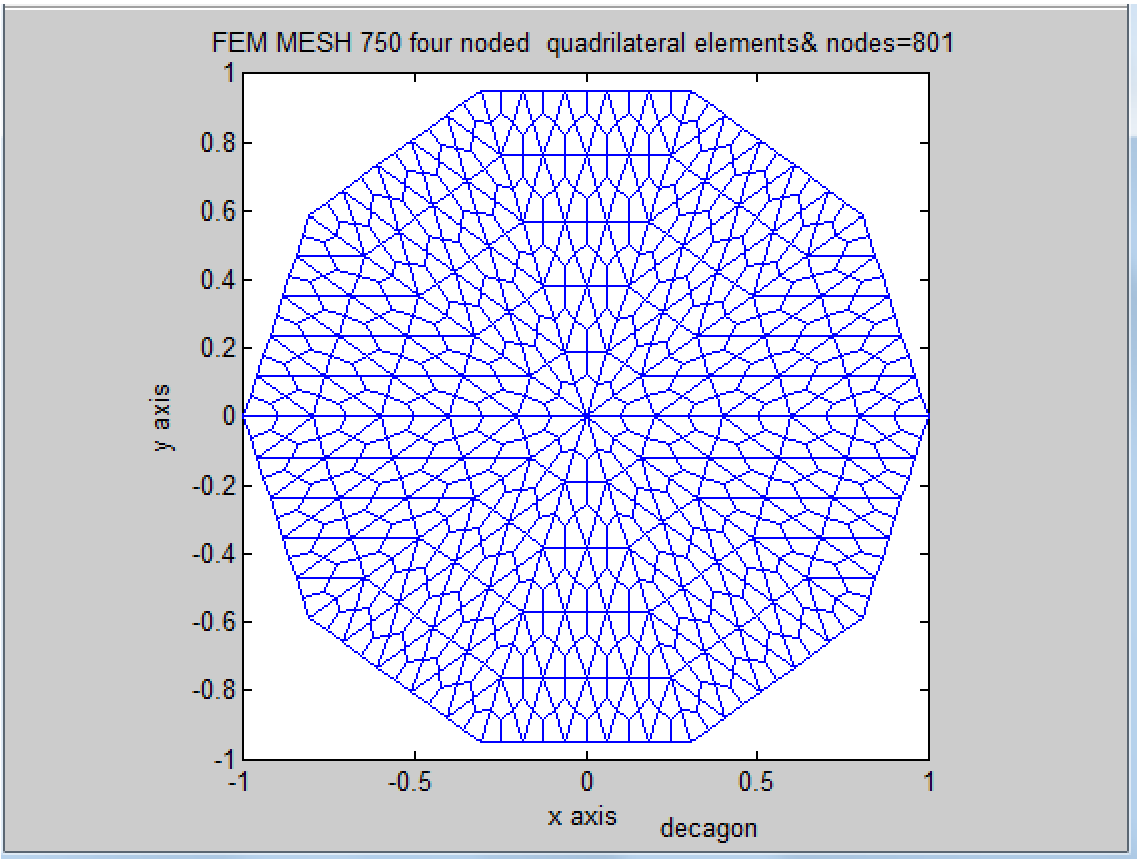



Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
NONAGON
Y-axis
X-axis
(MESH HAS 721 NODES AND 675 ELEMENTS)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

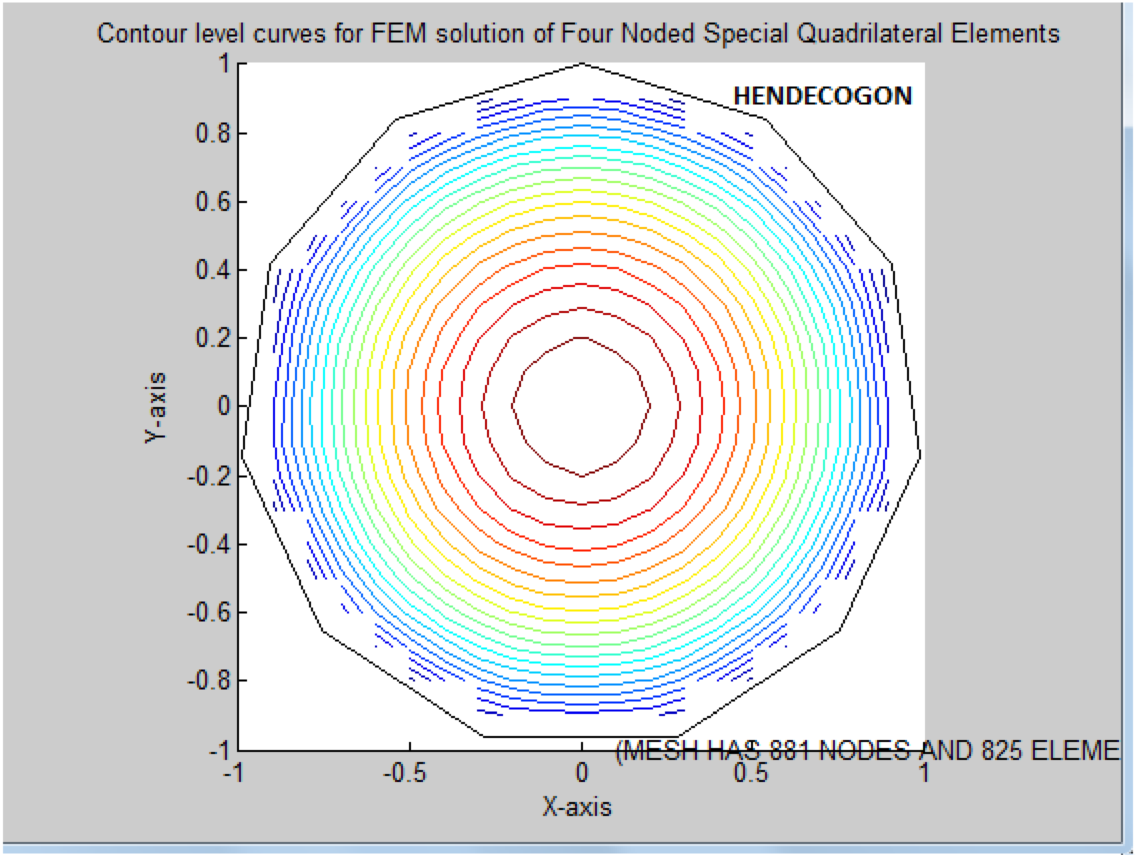

FEM MESH 675 four noded quadrilateral elements& nodes=721
y axis
x axis
nonadecagon
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

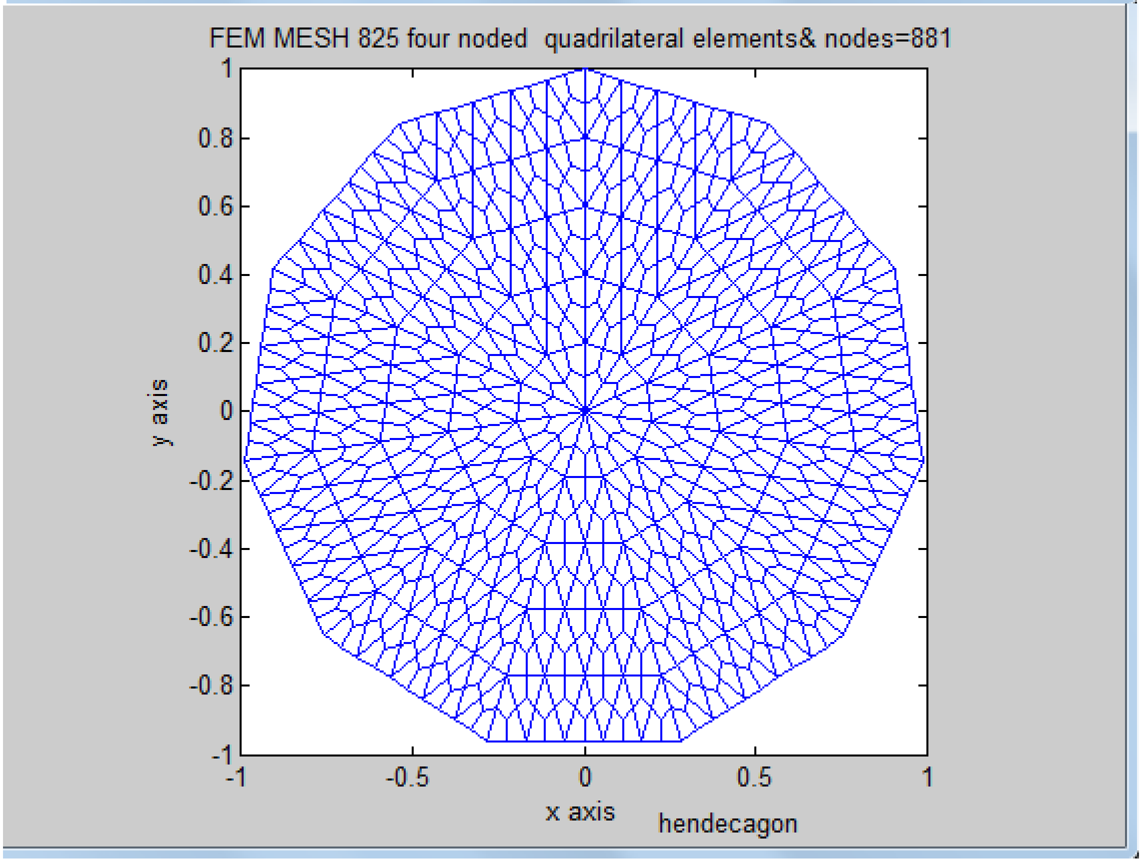
DOI: 10.18535/ijecs/v5i10.33



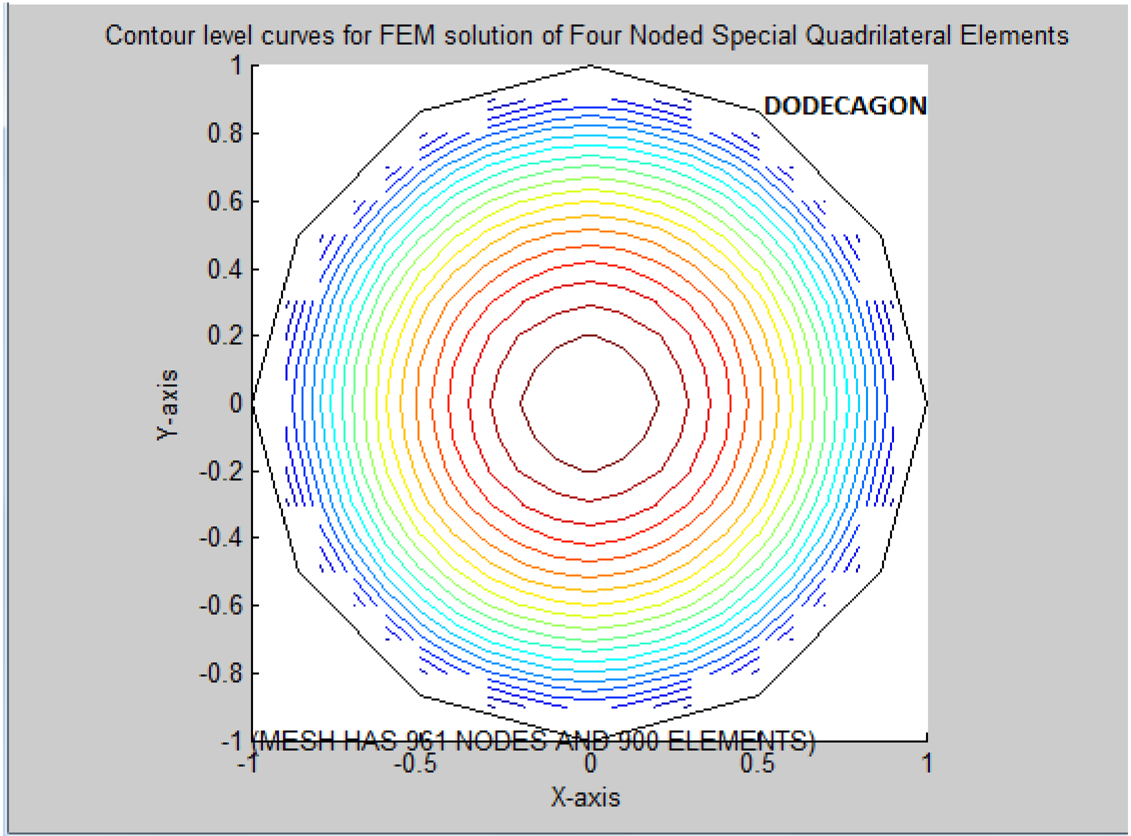


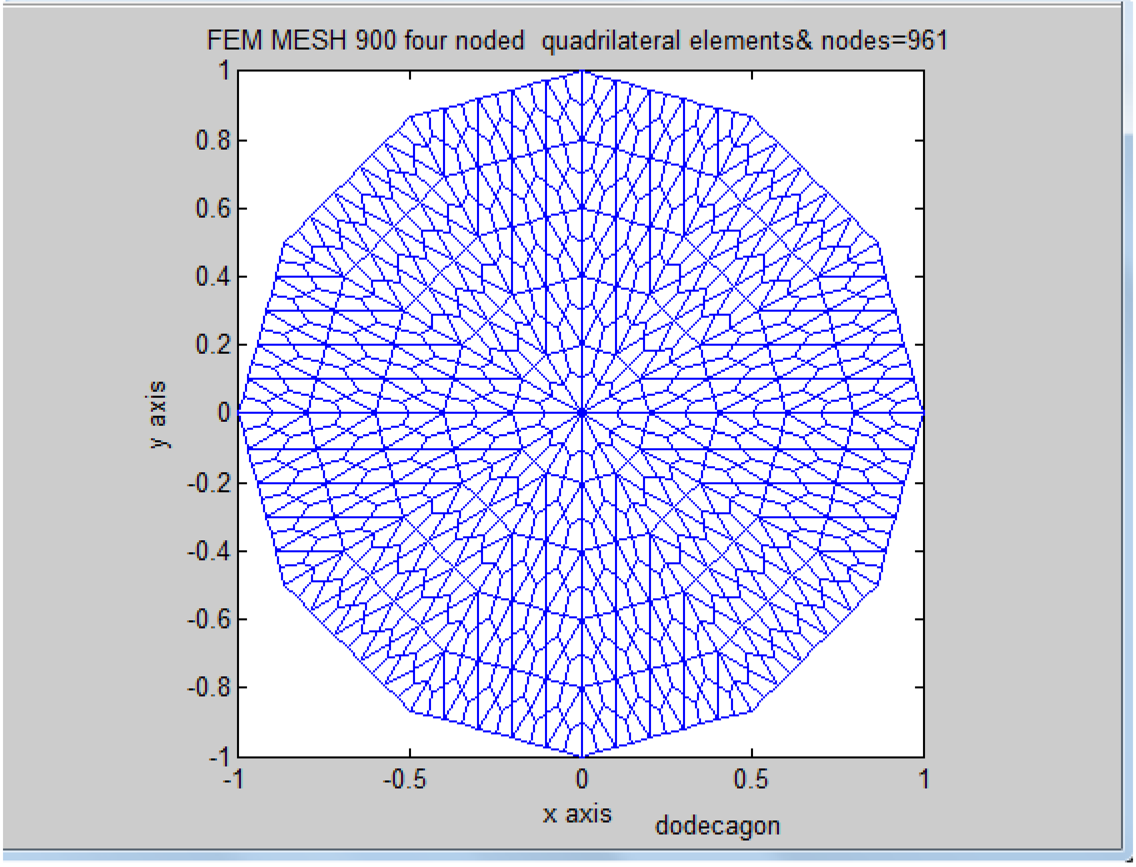
DOI: 10.18535/ijecs/v5i10.33

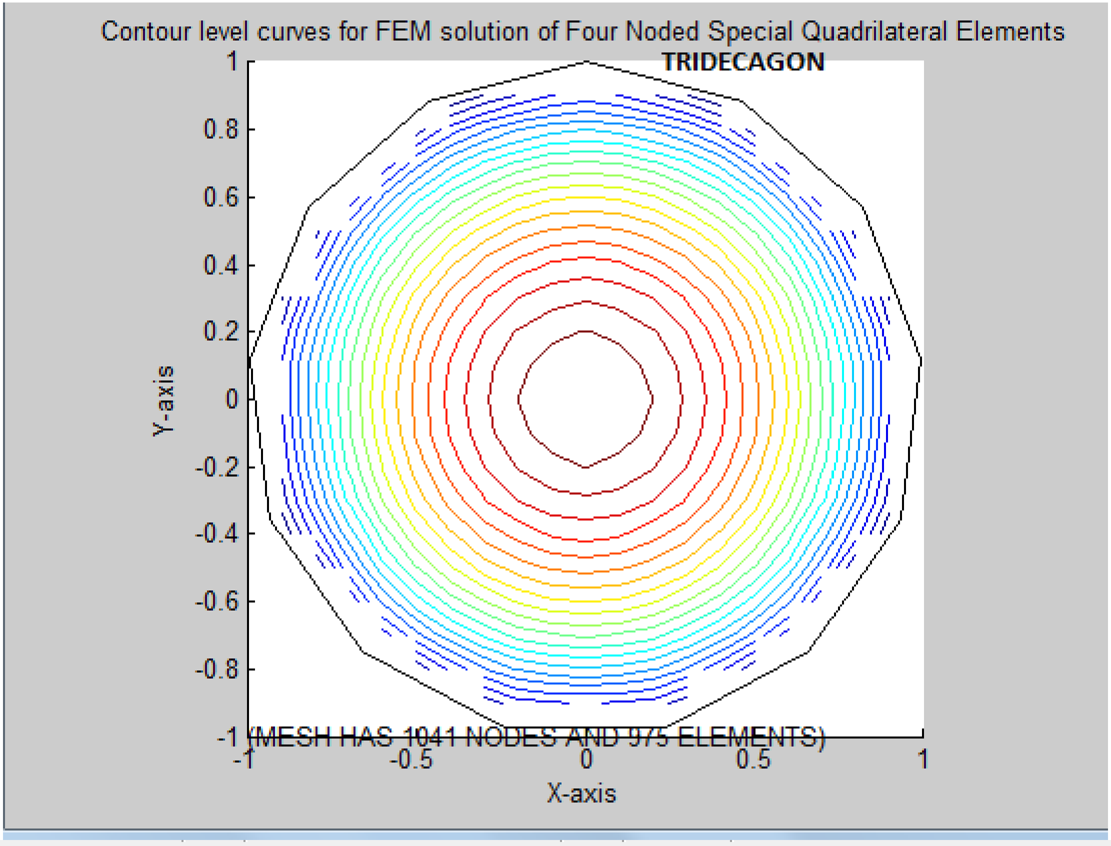


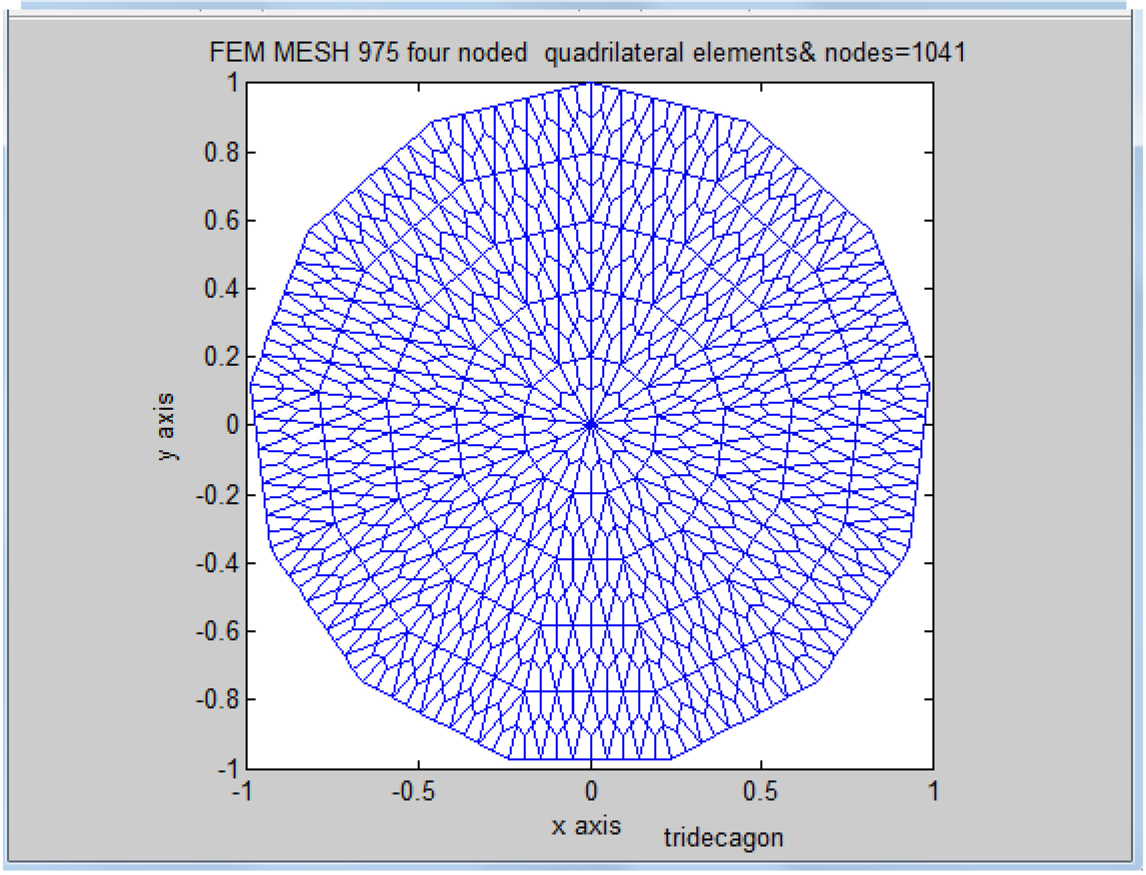


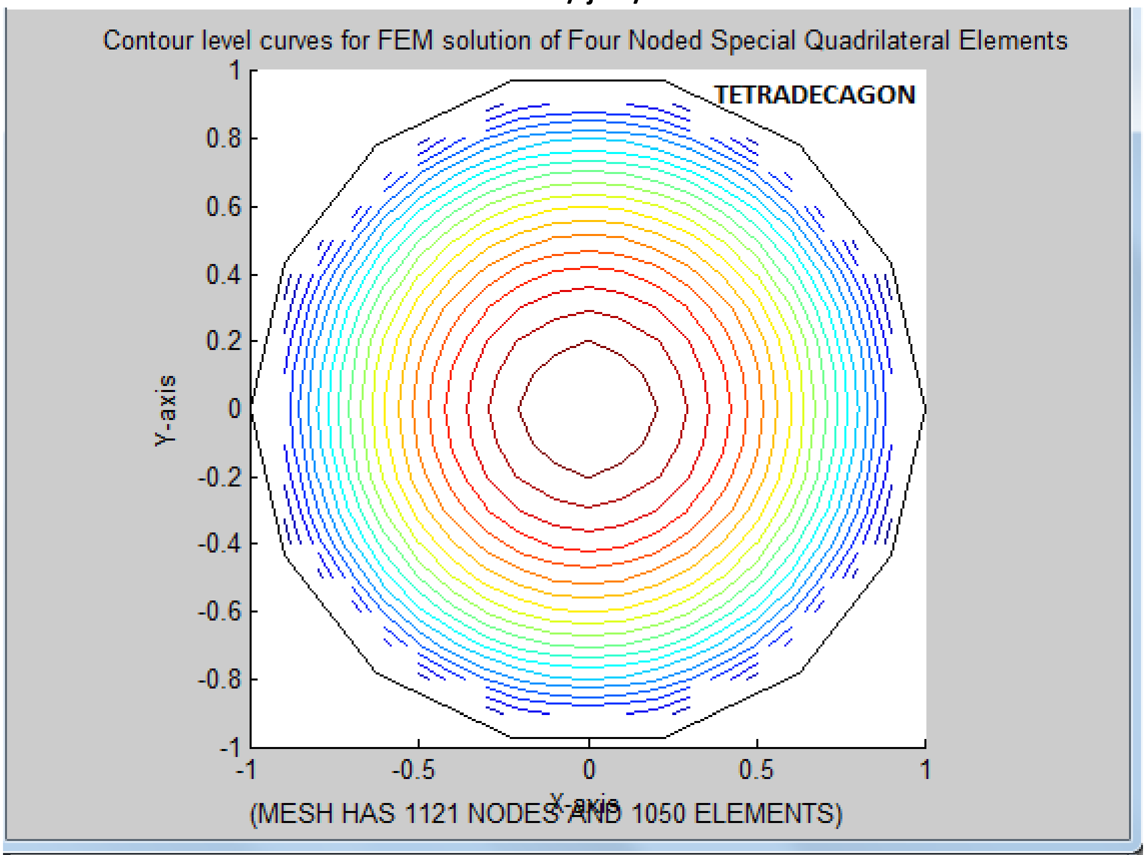
DOI: 10.18535/ijecs/v5i10.33

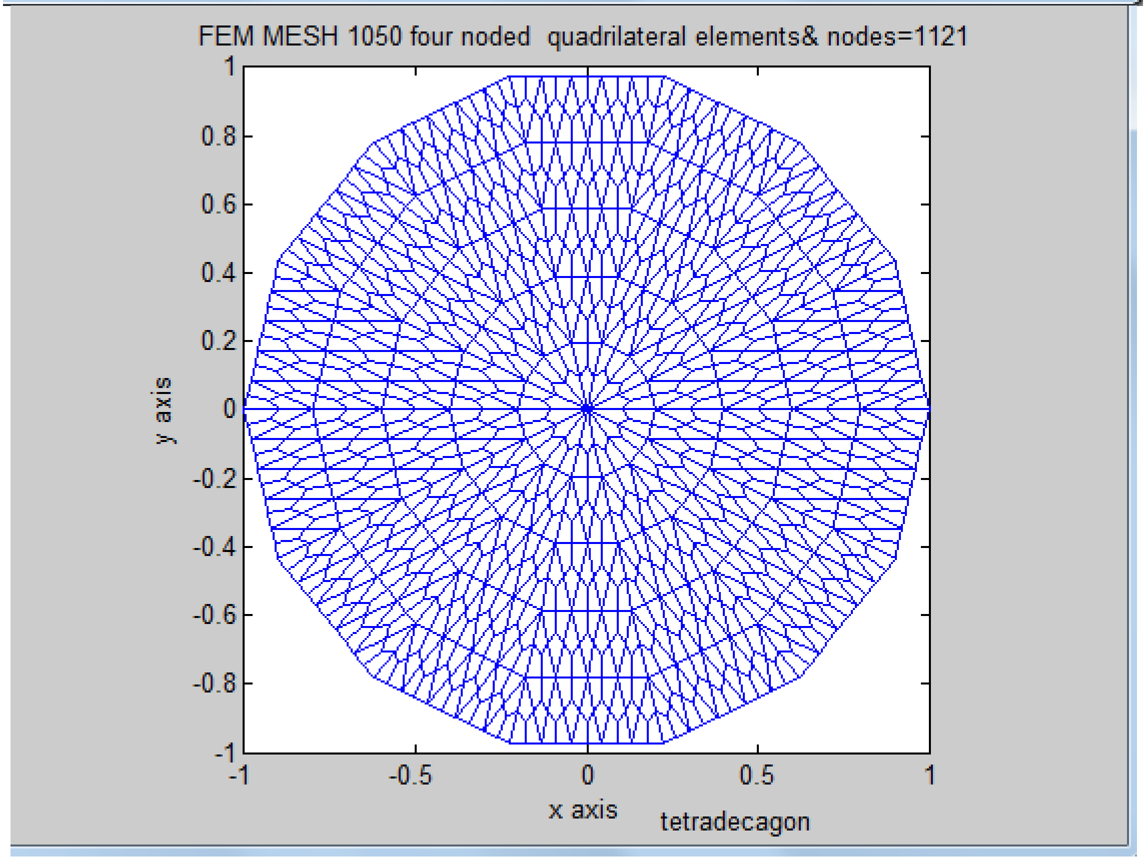


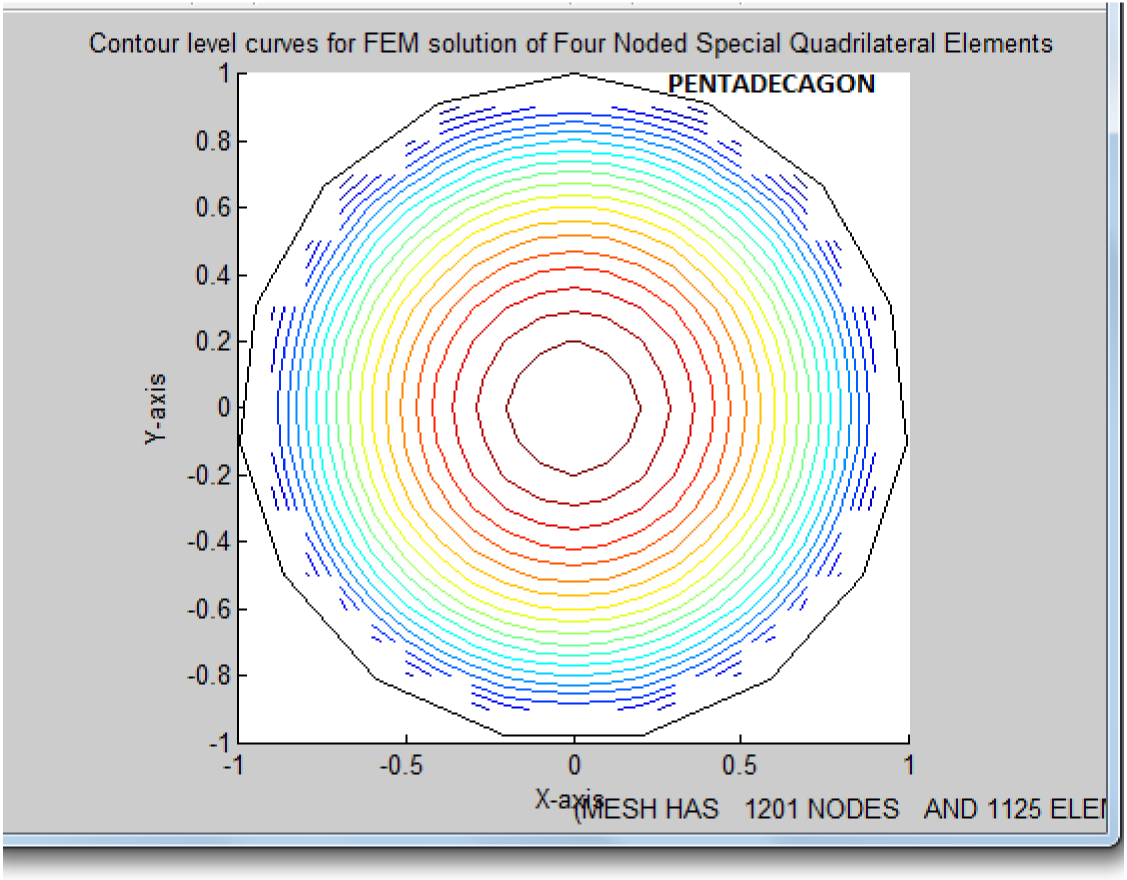


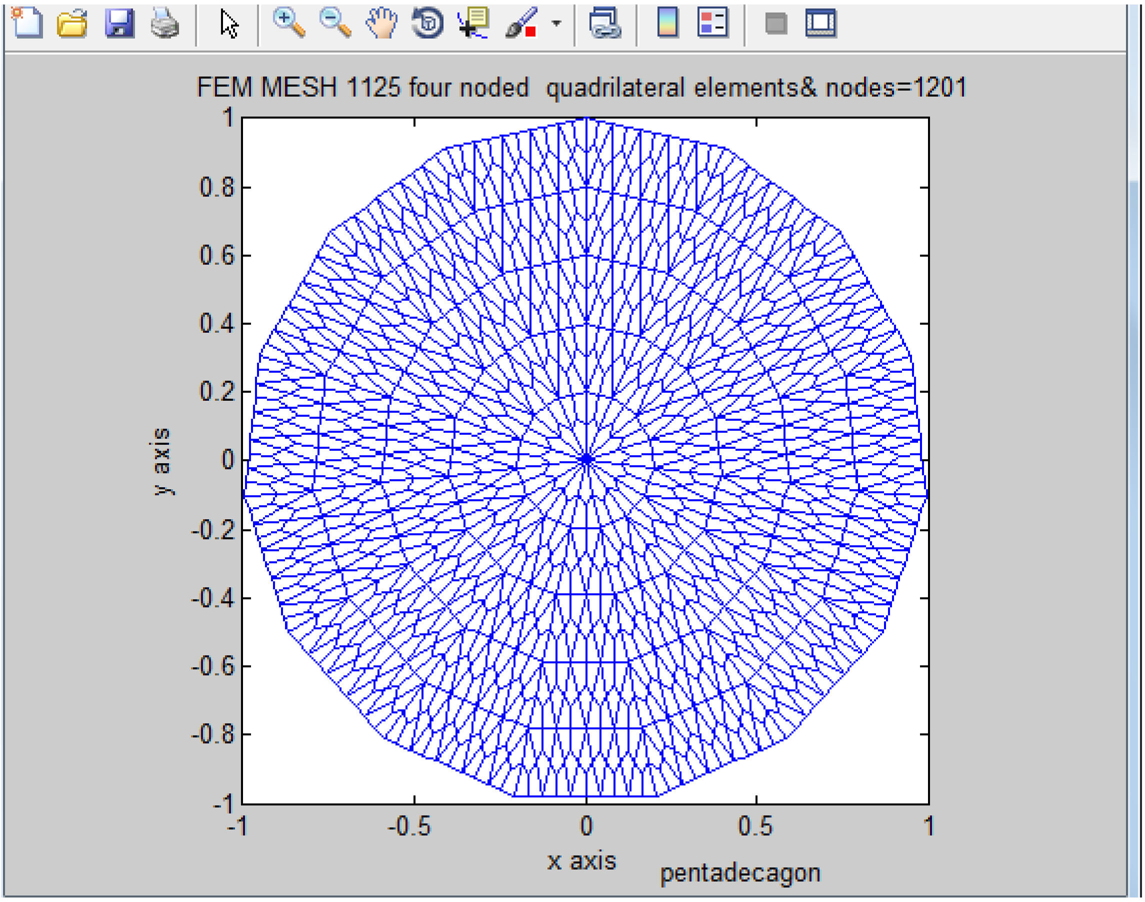

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
TRIDECAGON
(MESH HAS 1041 NODES AND 975 ELEMENTS)
Y-axis
X-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


FEM MESH 975 four noded quadrilateral elements& nodes=1041
y axis
x axis
tridecagon
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

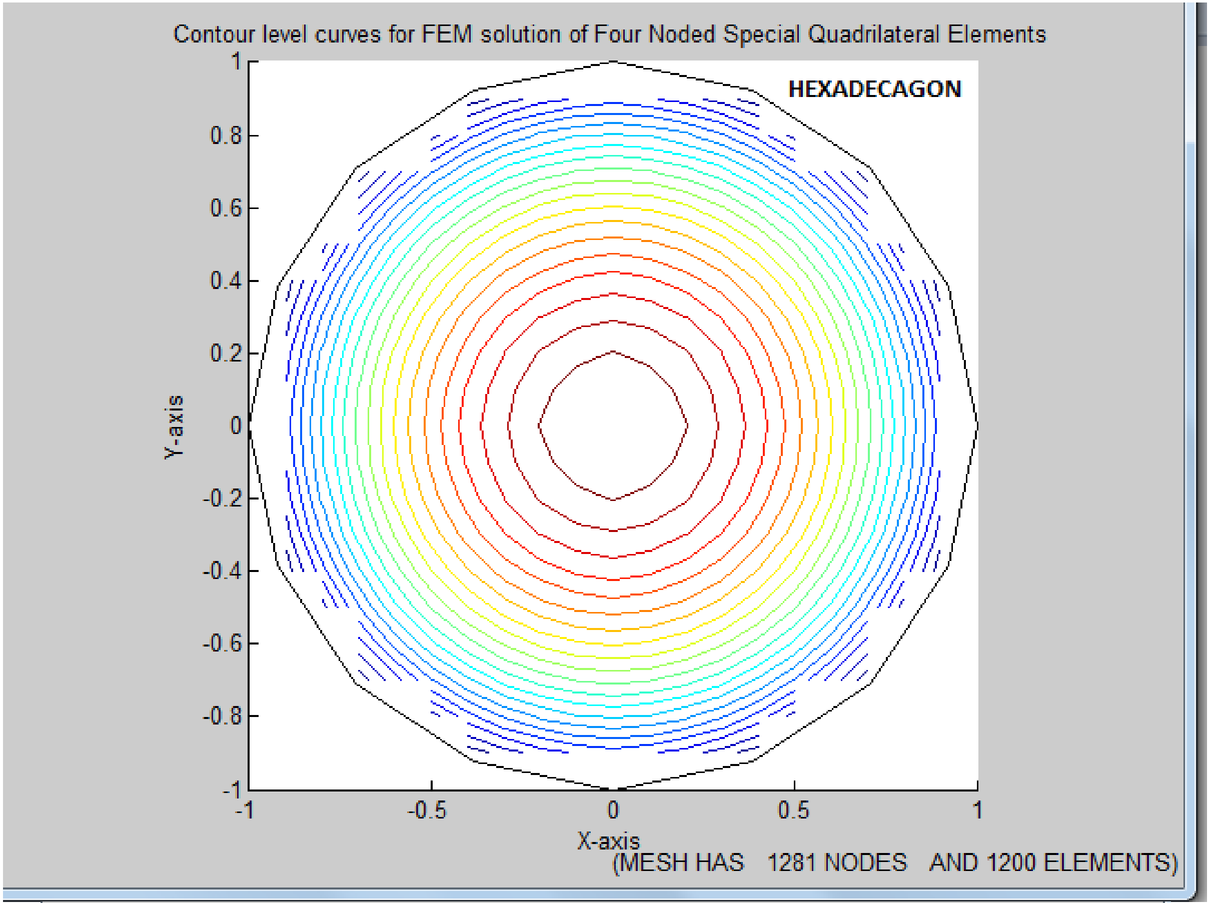

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
TETRADECAGON
Y-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
X-axis
(MESH HAS 1121 NODES AND 1050 ELEMENTS)

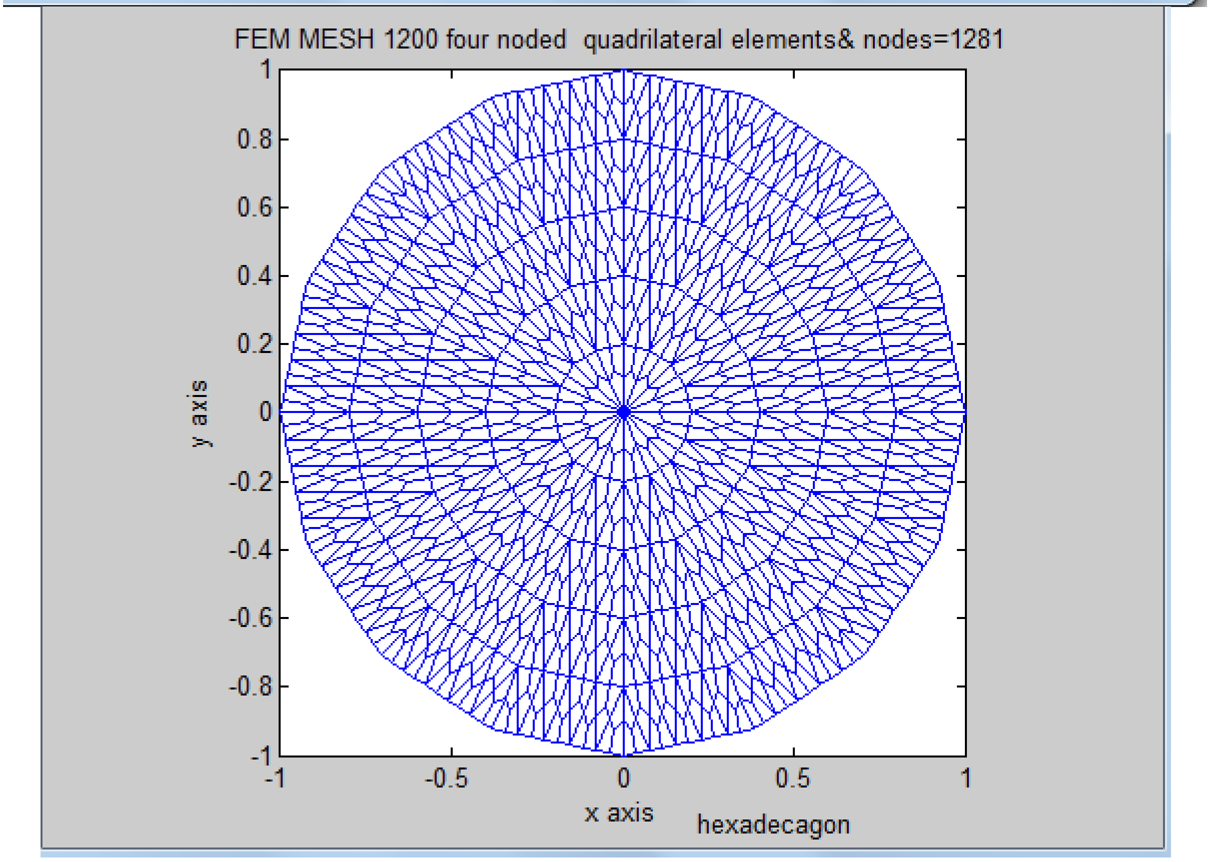

FEM MESH 1050 four noded quadrilateral elements& nodes=1121
y axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
x axis
tetradecagon


Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
PENTADECAGON
Y-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
X-axis
(MESH HAS 1201 NODES AND 1125 ELE

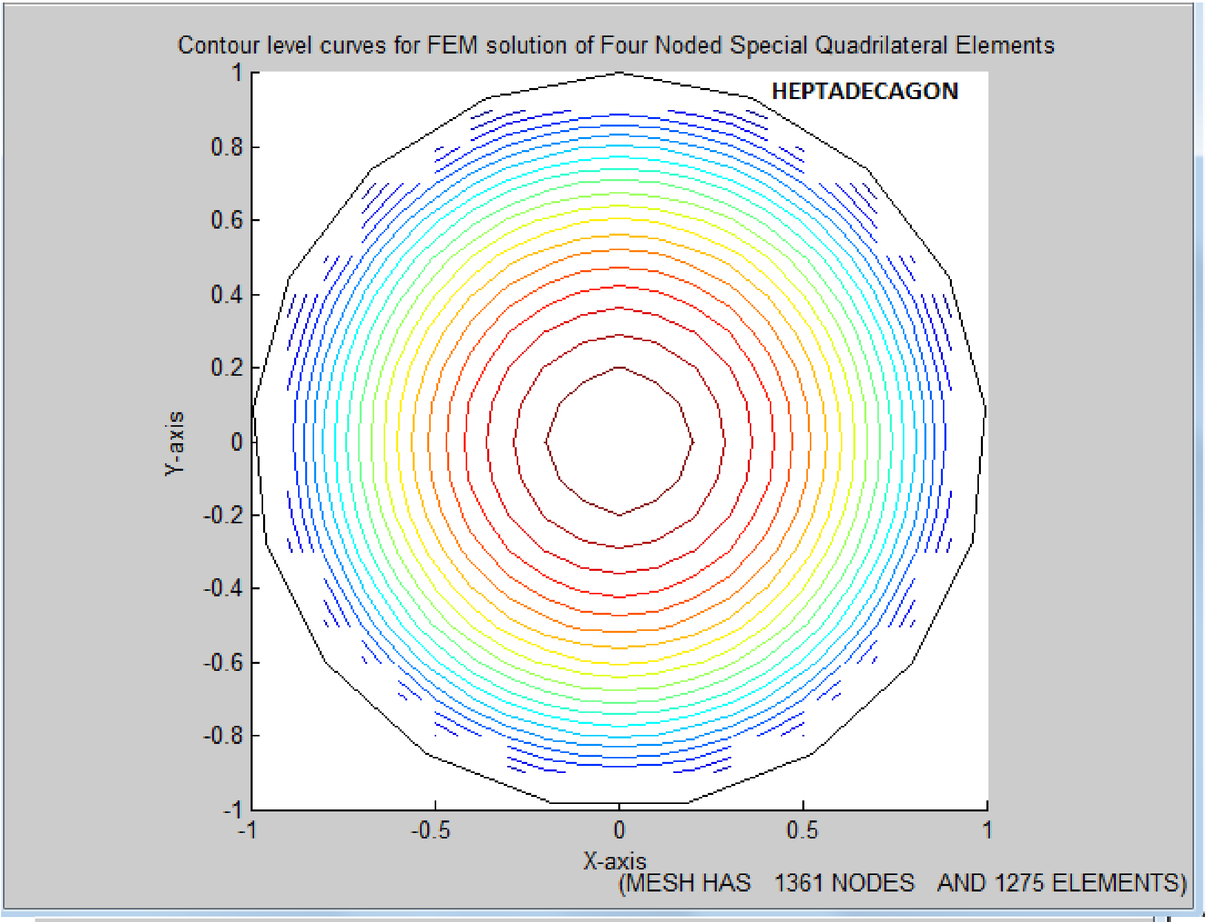

FEM MESH 1125 four noded quadrilateral elements& nodes=1201
y axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
x axis
pentadecagon

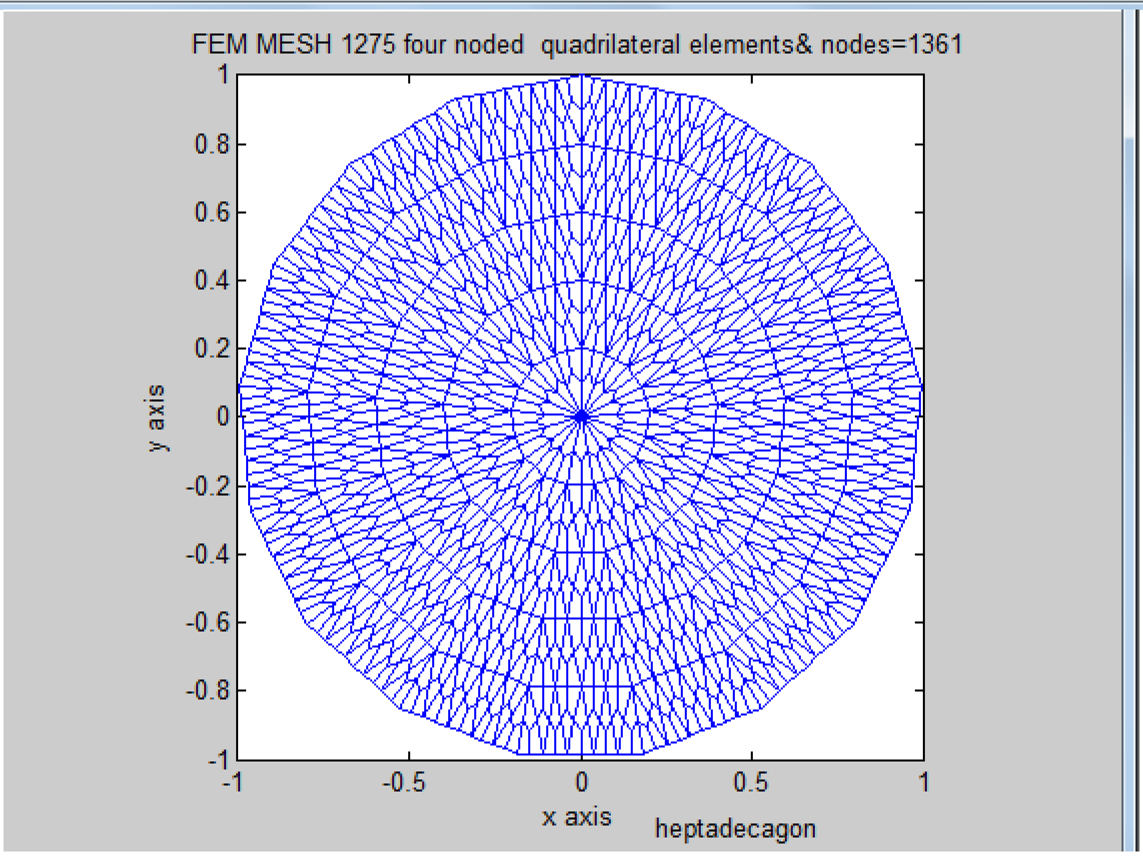
DOI: 10.18535/ijecs/v5i10.33

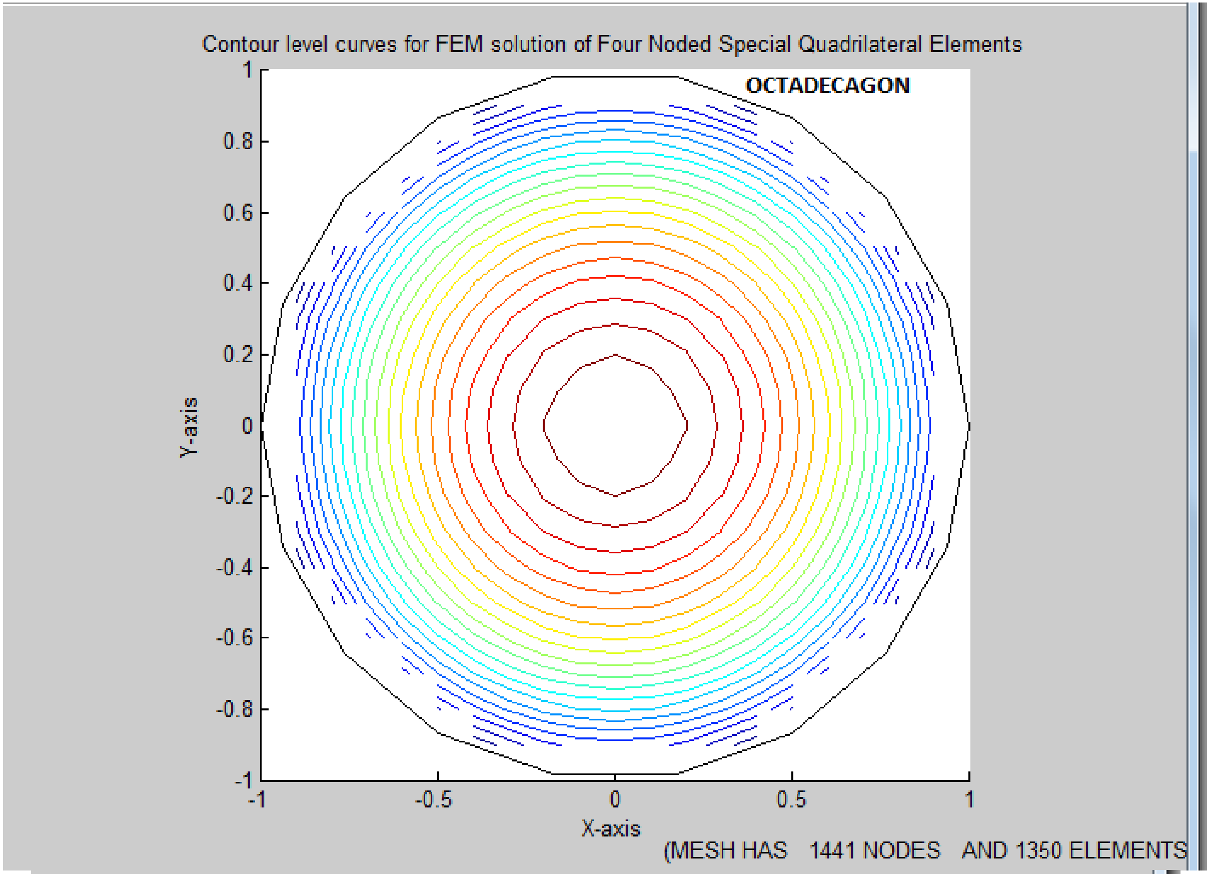


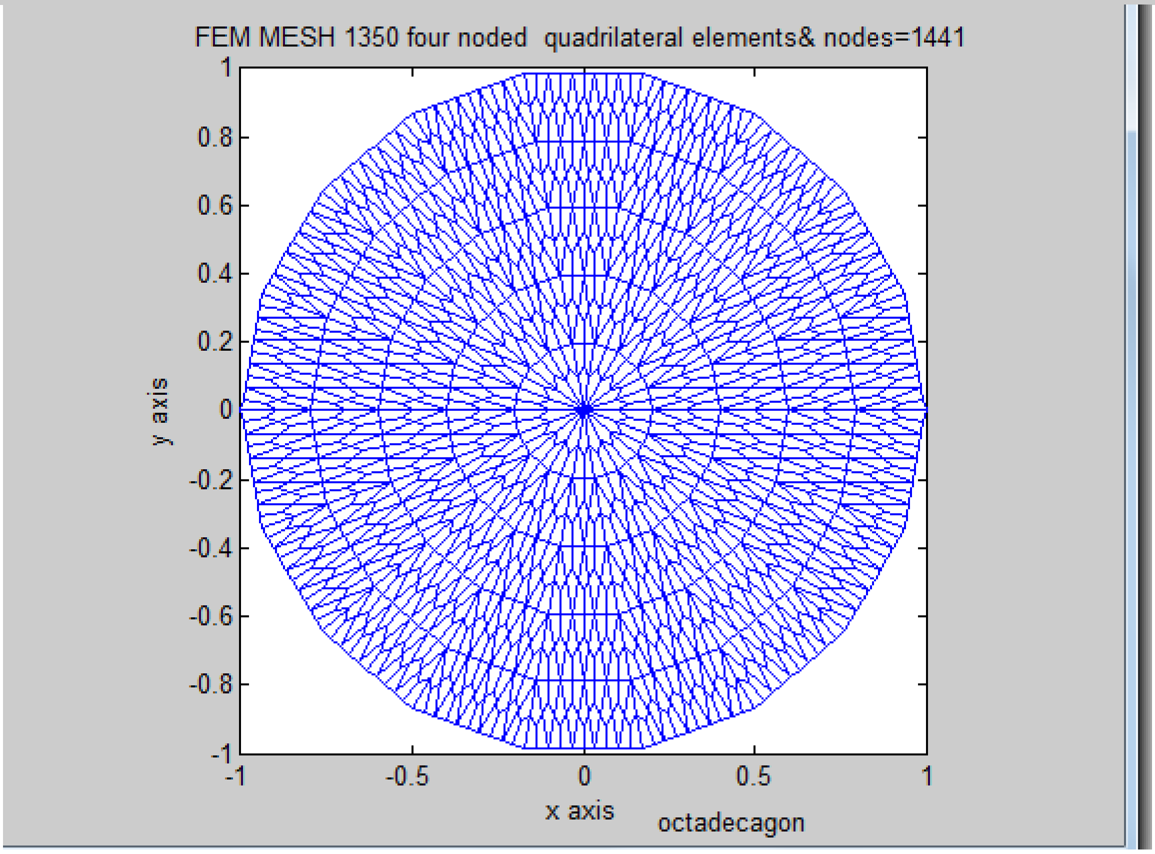


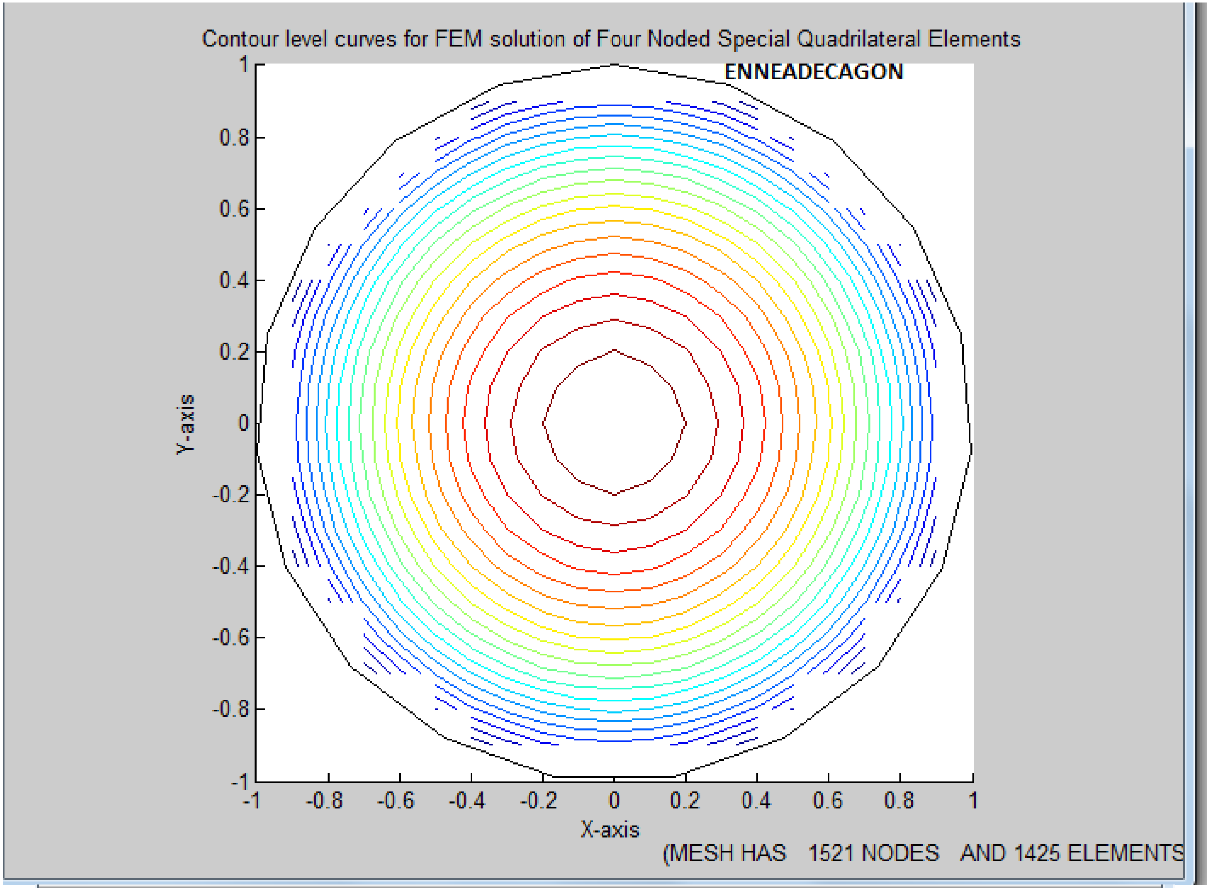
DOI: 10.18535/ijecs/v5i10.33

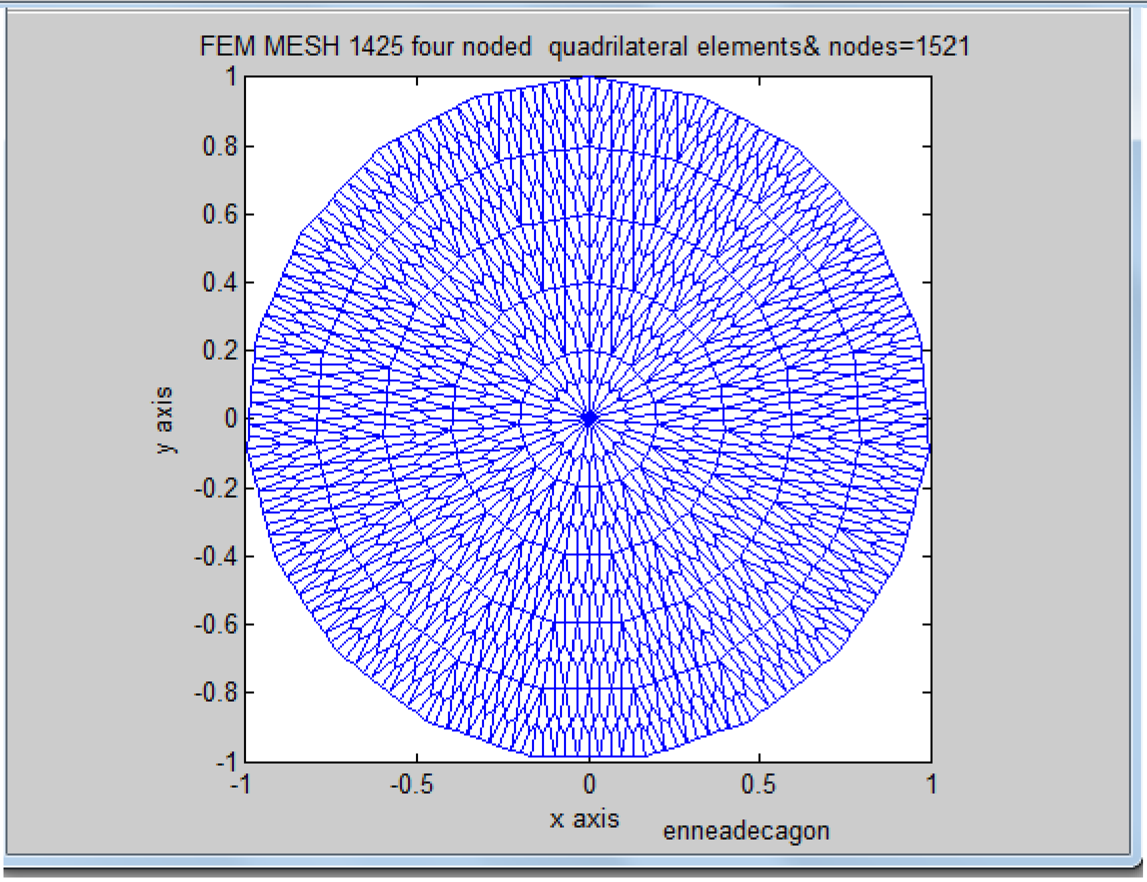


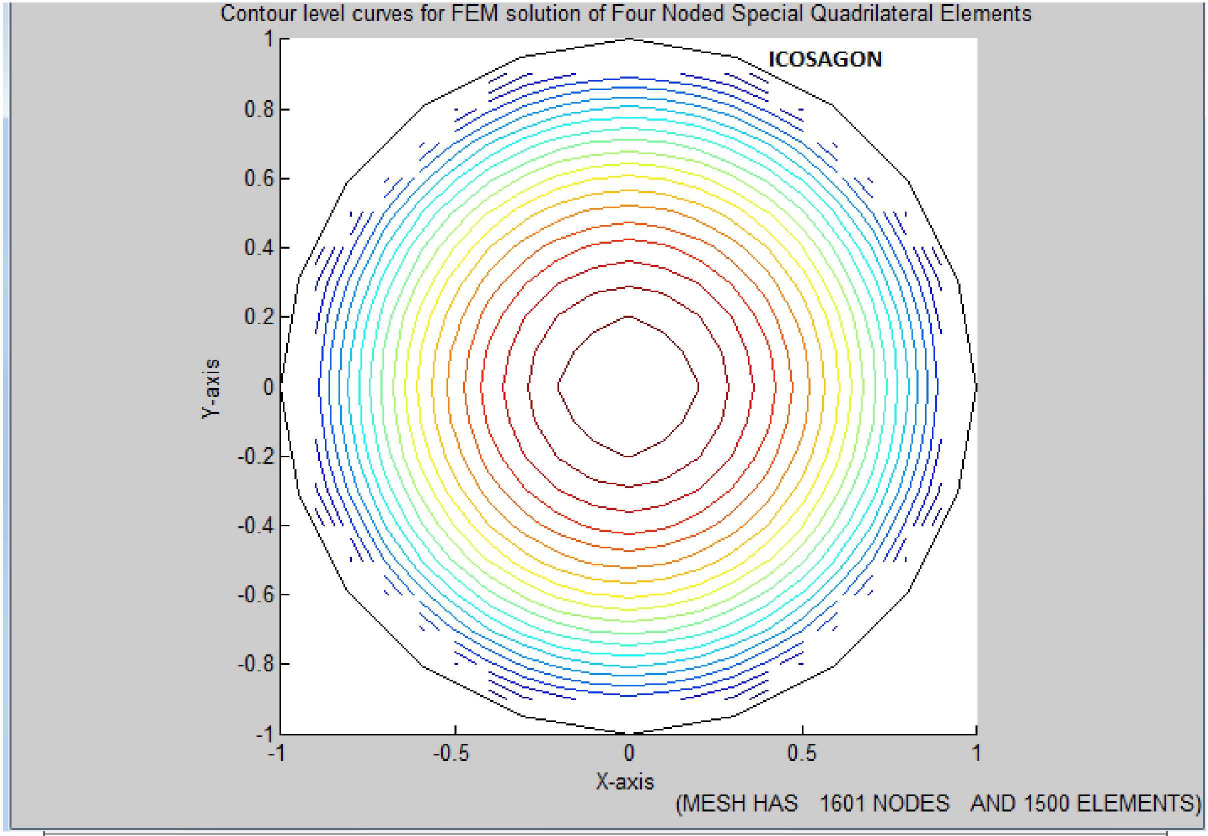


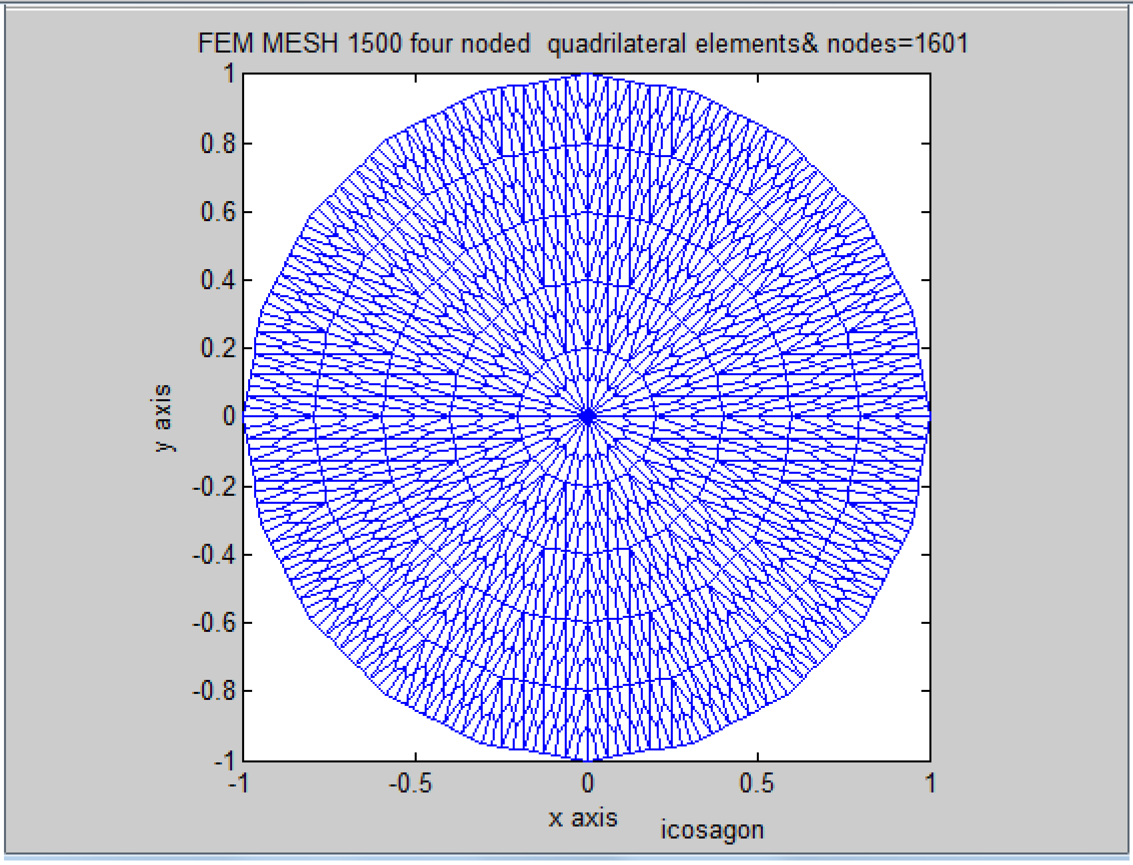

Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
OCTADECAGON
Y-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
X-axis
(MESH HAS 1441 NODES AND 1350 ELEMENTS


FEM MESH 1350 four noded quadrilateral elements& nodes=1441
y axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
x axis
octadecagon


Contour level curves for FEM solution of Four Noded Special Quadrilateral Elements
ENNEADECAGON
Y-axis
X-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
(MESH HAS 1521 NODES AND 1425 ELEMENTS


FEM MESH 1425 four noded quadrilateral elements& nodes=1521
y axis
x axis
enneadecagon
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1





COMPUTER PROGRAMS

(1) PROGRAM-1

```
function[nnode,nel,T,k1,errorTK1]=D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationo
verstdtria(n1,n2,n3,nmax,numtri,ndiv,mesh)
%note that input vlues of X and Y must be symbolic constants
%for the example triangle input for X is sym([-1/2 1/2 0])
%for the example triangle input for Y is sym([0 0 sqrt(3/4)])
%LaplaceEquationQ4twoD(3,sym([-1/2 1/2 0]),sym([0 0 sqrt(3/4)]))
%ndiv=2,4,6,8,10,......
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria(1,2,3,3,1,2,17)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1],[2;3;4;5],[3
;4;5;2],5,1,2,20)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1],[2;3;4;5;6
],[3;4;5;6;2],6,1,2,21)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1],[2;3;4;5
;6;7],[3;4;5;6;7;2],7,1,2,22)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4,20,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,9,6,21,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,1,2,22,1,1)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1],[2;3;4;5
;6;7],[3;4;5;6;7;2],7,25,10,22)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1],[2;3;4;5
;6;7],[3;4;5;6;7;2],7,100,20,22)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1],[2;3;4
;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,23)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1],[2;3
;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,24)
%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
,2,24,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10],[3;4;5;6;7;8;9;1
0;2],10,25,10,25,1,1)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1],
[2;3;4;5;6;7;8;9;10;11],[3;4;5;6;7;8;9;10;11;2],11,1,2,26)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
],[2;3;4;5;6;7;8;9;10;11;12],[3;4;5;6;7;8;9;10;11;12;2],12,1,2,27)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11],[3;4;5;6;7;
8;9;10;11;2],11,1,2,26,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12],[3;4;5
;6;7;8;9;10;11;12;2],12,1,2,27,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13],[
3;4;5;6;7;8;9;10;11;12;13;2],13,1,2,28,1,1)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1],[2;3;4;5;6;7;8;9;10;11;12;13],[3;4;5;6;7;8;9;10;11;12;13;2],13,1,2,28)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14],[3;4;5;6;7;8;9;10;11;12;13;14;2],14,1,2,29)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15],[3;4;5;6;7;8;9;10;11;12;13;14;15;2],15,1,2,
30)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;2]
,16,1,2,31)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17],[3;4;5;6;7;8;9;10;11;12;13;14;15;
16;17;2],17,1,2,32)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18],[3;4;5;6;7;8;9;10;11;12;13;1
4;15;16;17;18;2],18,1,2,33)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19],[3;4;5;6;7;8;9;10;11;12
;13;14;15;16;17;18;19;2],19,1,2,34)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20],[3;4;5;6;7;8;9;10;
11;12;13;14;15;16;17;18;19;20;2],20,1,2,35)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria([1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21],[3;4;5;6;7;8;
9;10;11;12;13;14;15;16;17;18;19;20;21;2],21,1,2,36)
syms coord xnn ynn xni yni xnj ynj tt
```

```
nnel=4;
ndof=1;
format long g
if nmax==3
nc=(ndiv/2)^2;
nnode=(ndiv+1)*(ndiv+2)/2+nc;
nel=3*nc;
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
for i=1:nel
N(i,1)=i;
end
end

[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtr
i,ndiv,mesh)
  if nmax>3
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
for i=1:nel
N(i,1)=i;
end

  end
  switch mesh
   case 17% torsion of a square cross section(one triangle required due to symmetry)
    nnn=0;
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn)==sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

    bcdof;
    mm=length(bcdof);

format long g
k1 =4*double(0.14057701495515551037840396020329);
 xi=(zeros(nnode,1));
 a2=sqrt(2);aa=1/4;
a0=32*aa/pi^3;
for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));rr=(0);

for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*y/a2)/cosh(n*pi/2)))*cos(n*pi*x/a2)/n^3;
end
xi(m,1)=(a0*rr);
end
  case 18%torsion of an equilateral triangle

 nnn=0;
 %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+1)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
```

```
%boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym(-(sqrt(3))*xnn-ynn+2)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym((sqrt(3))*xnn-ynn+2)==0)
            nnn=nnn+1
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  bcdof
  bcval
  mm=length(bcdof);
  for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));
    xi(m,1)=((y+1)*((sqrt(3))*x-y+2)*(-(sqrt(3))*x-y+2))/12;
  end
format long g
k1 =9*sqrt(3)/5;
 case 19%torsion of an equilateral triangle
  nnn=0;
 %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+1/2)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym(-(sqrt(3))*xnn-ynn+1)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym((sqrt(3))*xnn-ynn+1)==0)
            nnn=nnn+1
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  bcdof
  bcval
  mm=length(bcdof);
  for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));
    xi(m,1)=((y+1/2)*((sqrt(3))*x-y+1)*(-(sqrt(3))*x-y+1))/6;
  end
format long g
k1 =9*sqrt(3)/5/16;
```

```
 case 20%  torsion of a square cross section(without symmetry considerations)
 %************************************************

    nnn=0;
    %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn)==sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
     %boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((xnn)==-sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

     %boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn)==-sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

      %boundary conditions on side 4
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((xnn)==sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

    bcdof;
    mm=length(bcdof);

format long g
k1 =4*double(0.14057701495515551037840396020329);
 xi=(zeros(nnode,1));
 a2=sqrt(2);aa=1/4;
a0=32*aa/pi^3;
for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));rr=(0);

for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*y/a2)/cosh(n*pi/2)))*cos(n*pi*x/a2)/n^3;
end
xi(m,1)=(a0*rr);
end
case 21%  torsion of a pentagon cross section(without symmetry considerations)

 %=========================================
   syms COORD
 nside=5;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  %========================================
    bcdof;
    mm=length(bcdof);

format long g
%k1 =0.843477%ABDELKAR
 k1=0.844754404592%
xi=(zeros(nnode,1));

for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
case 22%  torsion of a hexagonal cross section(without symmetry considerations)
  nnn=0;
   snc=sym(sin(pi/3));
   csc=sym(cos(pi/3));
   %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((csc-1)*ynn-snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
   %boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn-snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
 %boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((csc-1)*ynn+snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 4
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((-(csc-1)*ynn+snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
   %boundary conditions on side 5
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 6
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((-(csc-1)*ynn-snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

    bcdof;
    mm=length(bcdof);

format long g
k1 =1.03831376

xi=(zeros(nnode,1));

for m=1:nnode
xi(m,1)=1;%exact solutions not known
end

case 23%  torsion of a heptagonal cross section(without symmetry considerations)
 coord
  nnn=0;
 %boundary conditions on side 1
  xni=coord(2,1);yni=coord(2,2);xnj=coord(3,1);ynj=coord(3,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
         if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
 %boundary conditions on side 2
 xni=coord(3,1);yni=coord(3,2);xnj=coord(4,1);ynj=coord(4,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
  %boundary conditions on side 3
   xni=coord(4,1);yni=coord(4,2);xnj=coord(5,1);ynj=coord(5,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
%boundary conditions on side 4
 xni=coord(5,1);yni=coord(5,2);xnj=coord(6,1);ynj=coord(6,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
%boundary conditions on side 5
xni=coord(6,1);yni=coord(6,2);xnj=coord(7,1);ynj=coord(7,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  %boundary conditions on side 6
  xni=coord(7,1);yni=coord(7,2);xnj=coord(8,1);ynj=coord(8,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
     if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]

 %boundary conditions on side 7
 xni=coord(8,1);yni=coord(8,2);xnj=coord(2,1);ynj=coord(2,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
   if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
 bcdof;
 mm=length(bcdof);
 format long g
k1 =1.163550%ABDELKAR
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
 case 24%  torsion of a octagonal cross section(without symmetry considerations)
 syms COORD
 nside=8;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.25510511;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%=================================================
 case 25%nonagon
  syms COORD
 nside=9;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.316137
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%================
case 26%decagon
  syms COORD
 nside=10;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.362921;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
case 27%hendecagon
 syms COORD
 nside=11;
```

```
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.398027;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
 %***************************
   case 28%dodecagon
 syms COORD
 nside=12;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn
  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.424582;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end

 %*****************************
 case 29%tridecagon
  syms COORD
 nside=13;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.446645;%Abdelkader
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%*******************************
case 30%tetradecagon
syms COORD
 nside=14;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
```

```
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.463530;%Abdelkader
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%*************************************
 case 31%pentadecagon
 syms COORD
 nside=15;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.47728 ;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
```

```
%***************************************
 case 32%hexadecagon
syms COORD
 nside=16;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.48853 ;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
  case 33%(17-gon)heptadecagon
 syms COORD
 nside=17;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.49787;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%***************************************************
 case 34%(18-gon)octadecagon
 syms COORD
 nside=18;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.50574;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%****************************************************
case 35%(19-gon)enneadecagon
syms COORD
 nside=19;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
```

```
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.5124147;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%***********************************************
case 36%(20-gon)icosagon
syms COORD
 nside=20;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.5181239;%Hassenpflug
xi=(zeros(nnode,1));
```

```
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%***************************************************************


 end%switch
 for L=1:nel
   for M=1:3
      LM=nodetel(L,M);
      xx(L,M)=gcoord(LM,1);
      yy(L,M)=gcoord(LM,2);
    end
  end
%_________________________________________________________________
table1=[N nodes];
table2=[N xx yy];
%disp([xx yy])
intJdn1dn1uvrs =[vpa(sym(' .59552765500082114748572926733O')),
vpa(sym('.46832228582057464549116826929O'));vpa(sym('
.46832228582057464549116826929')), vpa(sym('.59552765500082114748572926733O'))];
intJdn1dn2uvrs =[vpa(sym('  -.39701843666721409832381951155 2')),
vpa(sym('.18778514278628356967255448713 95'));vpa(sym(' -
.31221485721371643032744551286 04')),   vpa(sym('.10298156333278590167618048844 8'))];
intJdn1dn3uvrs =[vpa(sym(' -.30149078166639295083809024422 35')),vpa(sym(' -
.34389257139314178483627724357 00'));vpa(sym(' -.34389257139314178483627724357 00')),
vpa(sym('-.30149078166639295083809024422 35'))];
intJdn1dn4uvrs =[vpa(sym('   .10298156333278590167618048844 8')), vpa(sym('-
.31221485721371643032744551286 04'));vpa(sym('  .18778514278628356967255448713 95')),
vpa(sym('-.39701843666721409832381951155 2'))];
%
intJdn2dn1uvrs =[vpa(sym('  -.39701843666721409832381951155 2')), vpa(sym('-
.31221485721371643032744551286 04'));vpa(sym('  .18778514278628356967255448713 95')),
vpa(sym(' .10298156333278590167618048844 8'))];
intJdn2dn2uvrs =[vpa(sym('   .26467895777814273221587967436 9')), vpa(sym('-
.12519009519085571311503632476 00'));vpa(sym(' -.12519009519085571311503632476 00')),
vpa(sym(' .26467895777814273221587967436 9'))];
intJdn2dn3uvrs =[vpa(sym('  .20099385444426196722539349614 91')),
vpa(sym('.22926171426209452322418482904 66'));vpa(sym(' -
.27073828573790547677581517095 34')), vpa(sym('-.29900614555573803277460650385 09'))];
intJdn2dn4uvrs =[vpa(sym(' -.68654375555190601117453658965e-1')),
vpa(sym('.20814323814247762021829700857 34'));vpa(sym('
.20814323814247762021829700857 34')), vpa(sym('-.68654375555190601117453658965e-1'))];
%
intJdn3dn1uvrs =[vpa(sym(' -.30149078166639295083809024422 35')), vpa(sym('-
.34389257139314178483627724357 00'));vpa(sym(' -
.34389257139314178483627724357 00')),vpa(sym(' -.30149078166639295083809024422 35'))];
intJdn3dn2uvrs =[vpa(sym('  .20099385444426196722539349614 91')), vpa(sym('-
.27073828573790547677581517095 34'));vpa(sym('  .22926171426209452322418482904 66')),
vpa(sym('-.29900614555573803277460650385 09'))];
intJdn3dn3uvrs =[vpa(sym(' .39950307277786901638730325192 54')),
vpa(sym('.38536914286895273838790758547 68'));vpa(sym('
.38536914286895273838790758547 68')), vpa(sym('.39950307277786901638730325192 54'))];
intJdn3dn4uvrs =[vpa(sym(' -.29900614555573803277460650385 09')), vpa(sym('
.22926171426209452322418482904 66'));vpa(sym(' -.27073828573790547677581517095 34')),
vpa(sym('.20099385444426196722539349614 91'))];
%
intJdn4dn1uvrs =[vpa(sym('   .10298156333278590167618048844 8')),
vpa(sym('.18778514278628356967255448713 95'));vpa(sym(' -
.31221485721371643032744551286 04')), vpa(sym(' -.39701843666721409832381951155 2'))];
intJdn4dn2uvrs =[vpa(sym(' -.68654375555190601117453658965e-1')),  vpa(sym('
.20814323814247762021829700857 34'));vpa(sym('   .20814323814247762021829700857 34')),
vpa(sym('-.68654375555190601117453658965e-1'))];
```

```
intJdn4dn3uvrs =[vpa(sym(' -.2990061455557380327746065038509')), vpa(sym('-
.2707382857379054767758151709534'));vpa(sym('  .2292617142620945232241848290466')),
vpa(sym('.2009938544442619672253934961491'))];
intJdn4dn4uvrs =[vpa(sym('   .2646789577781427322158796743 69')), vpa(sym('-
.1251900951908557131150363247600'));vpa(sym(' -.1251900951908557131150363247600')),
vpa(sym(' .2646789577781427322158796743 69'))];
%
%
intJdndn=double([intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs;...
          intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs;...
          intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs;...
          intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs]);


%
for iel=1:nel
index=zeros(nnel*ndof,1);

X=xx(iel,1:3);
Y=yy(iel,1:3);
%disp([X Y])
xa=X(1,1);
xb=X(1,2);
xc=X(1,3);
ya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
bta=yb-yc;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;
sk(1:4,1:4)=(zeros(4,4));SK(1:8,1:8)=(zeros(8,8));
for i=1:4
 for j=i:4
 sk(i,j)=(delabc*sum(sum(Q.*(intJdndn(2*i-1:2*i,2*j-1:2*j)))));
 sk(j,i)=sk(i,j);
 end
end
f =[5/144;1/24;7/144;1/24]*(delabc);



%______________________________________
 edof=nnel*ndof;
 k=0;
 for i=1:nnel
     nd(i,1)=nodes(iel,i);
     start=(nd(i,1)-1)*ndof;
     for j=1:ndof
         k=k+1;
         index(k,1)=start+j;
     end
 end
 %-------------------------------------------------------------------------
for i=1:edof
    ii=index(i,1);
    ff(ii,1)=ff(ii,1)+f(i,1);
 for j=1:edof
     jj=index(j,1);
     ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
 %-------------------------------------------------------------------------
```

```
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,1:nnode)=zeros(1,nnode);
    ss(1:nnode,kk)=zeros(nnode,1);
    ff(kk,1)=0;
end
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,kk)=1;
end
phi=ss\ff;
for I=1:nnode
NN(I,1)=I;
phi_xi(I,1)=phi(I,1)-xi(I,1);
end
MAXPHI_XI=max(abs(phi_xi));
%disp('__________________________________________________________________')
%disp('number of nodes,elements & nodes per element')
%[nnode nel nnel ndof]
%disp('element number    nodal connectivity for quadrilateral element')
%table1
%disp('________________________________________________________________________________
_____')
%disp('element number   coordinates of the triangle spanning the quadrilateral
element')
%table2
disp('_________________________________________________________________________________
__________________________')
disp('                                     Prandtl Stress Values')
disp('                  node number        computed values
anlytical(theoretical)        error           ')
disp('_________________________________________________________________________________
__________________________')
disp([NN phi xi phi_xi])
disp('_________________________________________________________________________________
__________________________')
nodes
nel
nnel
t=0;
for iel=1:nel
X=xx(iel,1:3);
Y=yy(iel,1:3);
%disp([X Y])
xa=X(1,1);
xb=X(1,2);
xc=X(1,3);
ya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
bta=yb-yc;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
for j=1:nnel
fe =[5/144;1/24;7/144;1/24]*(delabc);
en=nodes(iel,j);
t=t+phi(en,1)*fe(j,1);
end
end
switch mesh
    case 17
        T=16*t;
    case 18
        T=4*t;
    case 19
      T=4*t;
    case 20
        T=4*t;
    case 21
        T=4*t;
    case 22
       T=4*t;
     case 23
       T=4*t;
        case 24
       T=4*t;
       case 25
       T=4*t;
       case 26
       T=4*t;
       case 27
       T=4*t;
       case 28
       T=4*t;
      case 29
       T=4*t;
       case 30
       T=4*t;
         case 31
       T=4*t
          case 32
       T=4*t
           case 33
       T=4*t
              case 34
       T=4*t
               case 35
       T=4*t
               case 36
       T=4*t
end
%
disp('                              torisonal constants')
disp('                  fem=phi               exact=xi
error=(max(abs(phi_xi))')
disp('---------------------------------------------------------------------------------
--------------------------')
disp([T k1 MAXPHI_XI ])
disp('---------------------------------------------------------------------------------
--------------------------')
disp('number of nodes,elements & nodes per element')
disp([nnode nel nnel ])
disp('---------------------------------------------------------------------------------
--------------------------')
errorTK1=abs(T-k1)
[bcdof bcval]
length(bcdof)
```

(2)PROGRAM-2

```
function[]=D2PoissonEquationQ4MoinEx_MeshgridContour(n1,n2,n3,nmax,numtri,ndiv,mesh)

%note that input vlues of X and Y must be symbolic constants
%for the example triangle input for X is sym([-1/2 1/2 0])
%for the example triangle input for Y is sym([0 0 sqrt(3/4)])
%LaplaceEquationQ4twoD(3,sym([-1/2 1/2 0]),sym([0 0 sqrt(3/4)]))
%ndiv=2,4,6,8,......
%polygonal_domain_coordinates([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2)
%polygonal_domain_coordinates([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4)
%D2LaplaceEquationQ4MoinExautomeshgen(n1,n2,n3,nmax,numtri,ndiv)
```

```
%D2LaplaceEquationQ4MoinExautomeshgen([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)
%D2LaplaceEquationQ4MoinExautomeshgen([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)
%D2LaplaceEquationQ4MoinExautomeshgen([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)
%D2LaplaceEquationQ4MoinExautomeshgen([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)
%quadrilateral_mesh4MOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)([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)
%D2POISSONEQUATION_NODALINTERPOLATION_VALUES(n1,n2,n3,nmax,numtri,ndiv,mesh)([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)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;
9;2],9,100,20,2)
%D2PoissonEquationQ4MoinEx_MeshgridContour([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)
%D2PoissonEquationQ4MoinEx_MeshgridContour([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)
%D2PoissonEquationQ4MoinEx_MeshgridContour([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)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8
;2],8,9,6,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8
;2],8,16,8,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8
;2],8,25,10,1)
%D2PoissonEquationQ4MoinExNew_MeshgridContour([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)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria(1,2,3,3,1,2,17)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria(1,2,3,3,1,2,18)
%D2LaplaceEquationQ4Ex3automeshgenTcomputenewequationoverstdtria(1,2,3,3,1,2,19)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1],[2],[3],3,1,2,18)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1],[2],[3],3,1,2,19)
%D2PoissonEquationQ4MoinExNew_MeshgridContour([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,20)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,1,2,21
)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,
1,2,22)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8
;2],8,1,2,23)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6
;7;8;9;2],9,1,2,24)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10],[3;
4;5;6;7;8;9;10;2],10,1,2,25)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11
],[3;4;5;6;7;8;9;10;11;2],11,1,2,26)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;
11;12],[3;4;5;6;7;8;9;10;11;12;2],12,1,2,27)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;1
0;11;12;13],[3;4;5;6;7;8;9;10;11;12;13;2],13,1,2,28)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9
;10;11;12;13;14],[3;4;5;6;7;8;9;10;11;12;13;14;2],14,1,2,29)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8
;9;10;11;12;13;14;15],[3;4;5;6;7;8;9;10;11;12;13;14;15;2],15,1,2,30)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7
;8;9;10;11;12;13;14;15;16],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;2],16,1,2,31)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6
;7;8;9;10;11;12;13;14;15;16;17],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;2],17,1,2,32)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5
;6;7;8;9;10;11;12;13;14;15;16;17;18],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;2],18,1,
2,33)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4
;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;
2],19,1,2,34)
```

```
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3
;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;1
8;19;20;2],20,1,2,35)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2
;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21],[3;4;5;6;7;8;9;10;11;12;13;14;15;16
;17;18;19;20;21;2],21,25,10,36)
%clf
syms coord
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtr
i,ndiv,mesh)
nnel=4;
ndof=1;
%nc=(ndiv/2)^2;
%nnode=(ndiv+1)*(ndiv+2)/2+nc;
%nel=3*nc;
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));

disp([nel nnode nnel ndof])
format long g
for i=1:nel
N(i,1)=i;
end
for i=1:nel
NN(i,1)=i;
end
    %[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh(ndiv);
    %[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
    %
    %bcdof=[2;5;3]
   %boundary conditions-1
   switch mesh
       case 1
    nnn=0;
    for nn=1:nnode
        xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (xnn==0)&((ynn>=0)&(ynn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-2
     for nn=1:nnode
        xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (ynn==0)&((xnn>=0)&(xnn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-3
    for nn=1:nnode
        xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (ynn==1)&((xnn>=0)&(xnn<=1/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
   %boundary conditions-4
   for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (xnn==1)&((ynn>=0)&(ynn<=1/2))
```

```
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
  %boundary conditions-5
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((xnn+ynn)==3/2)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=double((sin(pi*xnn))*(sin(pi*ynn)))
        end
  end
  format long g
%analytical solution

 xi=(zeros(nnode,1));
for m=1:nnode
    xm=(gcoord(m,1));ym=(gcoord(m,2));
    xi(m,1)=sin(pi*xm)*sin(pi*ym);
end


case 2
      nnn=0;
    for nn=1:nnode
        xnn=coord(nn,1);ynn=gcoord(nn,2);
        if (xnn==0)&((ynn>=0)&(ynn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-2
     for nn=1:nnode
        xnn=gcoord(nn,1);ynn=coord(nn,2);
        if (ynn==0)&((xnn>=0)&(xnn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-3
    for nn=1:nnode
        xnn=gcoord(nn,1);ynn=coord(nn,2);
        if (ynn==1)&((xnn>=0)&(xnn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
   %boundary conditions-4
   for nn=1:nnode
      xnn=coord(nn,1);ynn=gcoord(nn,2);
        if (xnn==1)&((ynn>=0)&(ynn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
  format long g
%analytical solution
```

```
 xi=(zeros(nnode,1));
for m=1:nnode
    xm=(gcoord(m,1));ym=(gcoord(m,2));
    xi(m,1)=sin(pi*xm)*sin(pi*ym);
end
%**************************************
case 18%torsion of an equilateral triangle

 nnn=0;
 %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+1)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym(-(sqrt(3))*xnn-ynn+2)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym((sqrt(3))*xnn-ynn+2)==0)
            nnn=nnn+1
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  bcdof
  bcval
  mm=length(bcdof);
  for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));
    xi(m,1)=((y+1)*((sqrt(3))*x-y+2)*(-(sqrt(3))*x-y+2))/12;
  end
format long g
k1 =9*sqrt(3)/5;



%****************************
      case 19%torsion of an equilateral triangle
  nnn=0;
 %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+1/2)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym(-(sqrt(3))*xnn-ynn+1)==0)
```

```
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (sym((sqrt(3))*xnn-ynn+1)==0)
            nnn=nnn+1
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  bcdof
  bcval
  mm=length(bcdof);
  for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));
    xi(m,1)=((y+1/2)*((sqrt(3))*x-y+1)*(-(sqrt(3))*x-y+1))/6;
  end
format long g
k1 =9*sqrt(3)/5/16;




 case 20%  torsion of a square cross section(without symmetry considerations)
 %**********************************************

    nnn=0;
    %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn)==sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
     %boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((xnn)==-sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

     %boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn)==-sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

      %boundary conditions on side 4
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
```

```
        if ((xnn)==sym(sqrt(2)/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end

    bcdof;
    mm=length(bcdof);

format long g
k1 =4*double(0.14057701495515551037840396020329);
 xi=(zeros(nnode,1));
 a2=sqrt(2);aa=1/4;
a0=32*aa/pi^3;
for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));rr=(0);

for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*y/a2)/cosh(n*pi/2)))*cos(n*pi*x/a2)/n^3;
end
xi(m,1)=(a0*rr);
end
%*************************************************
case 21%  torsion of a pentagon cross section(without symmetry considerations)
 %========================================
 syms COORD
 nside=5;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  %========================================
    bcdof;
    mm=length(bcdof);

format long g
%k1 =0.843477%ABDELKAR
 k1=0.844754404592%
```

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

for m=1:nnode
xi(m,1)=NaN;%exact solutions not known
end
%****************************************************
case 22%  torsion of a hexagonal cross section(without symmetry considerations)
  nnn=0;
   snc=sym(sin(pi/3));
   csc=sym(cos(pi/3));
   %boundary conditions on side 1
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((csc-1)*ynn-snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
   %boundary conditions on side 2
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn-snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
 %boundary conditions on side 3
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((csc-1)*ynn+snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 4
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((-(csc-1)*ynn+snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
   %boundary conditions on side 5
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((ynn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
%boundary conditions on side 6
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((-(csc-1)*ynn-snc*xnn+snc)==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
```

```
    bcdof;
    mm=length(bcdof);

format long g
k1 =1.03831376

xi=(zeros(nnode,1));

for m=1:nnode
xi(m,1)=NaN;%exact solutions not known
end
%*****************************************************
case 23%  torsion of a heptagonal cross section(without symmetry considerations)
 coord
  nnn=0;
 %boundary conditions on side 1
  xni=coord(2,1);yni=coord(2,2);xnj=coord(3,1);ynj=coord(3,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
         if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
 %boundary conditions on side 2
 xni=coord(3,1);yni=coord(3,2);xnj=coord(4,1);ynj=coord(4,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
  %boundary conditions on side 3
   xni=coord(4,1);yni=coord(4,2);xnj=coord(5,1);ynj=coord(5,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
%boundary conditions on side 4
 xni=coord(5,1);yni=coord(5,2);xnj=coord(6,1);ynj=coord(6,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
%boundary conditions on side 5
xni=coord(6,1);yni=coord(6,2);xnj=coord(7,1);ynj=coord(7,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
```

```
       if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  %boundary conditions on side 6
  xni=coord(7,1);yni=coord(7,2);xnj=coord(8,1);ynj=coord(8,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
     if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]

 %boundary conditions on side 7
 xni=coord(8,1);yni=coord(8,2);xnj=coord(2,1);ynj=coord(2,2);
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
   if (((xnj-xni)*(ynn-yni)-(ynj-yni)*(xnn-xni))==0)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
  end
  [bcdof bcval]
 bcdof;
 mm=length(bcdof);
 format long g
k1 =1.163550%ABDELKAR
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%**********************************************************
 case 24%  torsion of a octagonal cross section(without symmetry considerations)
 syms COORD
 nside=8;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
```

```
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.25510511;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end

%****************************************************
case 25%nonagon
  syms COORD
 nside=9;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.316137
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%===============
case 26%decagon
  syms COORD
 nside=10;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.362921;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%*************************************
case 27%hendecagon
 syms COORD
 nside=11;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.398027;
xi=(zeros(nnode,1));
 for m=1:nnode
```

```
xi(m,1)=1;%exact solutions not known
end
 %****************************case 28
  case 28%dodecagon
 syms COORD
 nside=12;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.424582;
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end

 %*****************************case 29
 case 29%tridecagon
  syms COORD
 nside=13;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.446645;%Abdelkader
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%*******************************case 30
%*******************************
case 30%tetradecagon
syms COORD
 nside=14;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.463530;%Abdelkader
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%************************************case 15
 %************************************
 case 31%pentadecagon
 syms COORD
 nside=15;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.47728 ;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end

%****************************************CASE32
case 32%hexadecagon
syms COORD
 nside=16;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn
  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.48853 ;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%*************************************case 33
 case 33%(17-gon)heptadecagon
 syms COORD
 nside=17;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.49787;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
 end
%************************************************
case 34%(18-gon)octadecagon
 syms COORD
 nside=18;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.50574;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%****************************************************case 35
case 35%(19-gon)enneadecagon
syms COORD
 nside=19;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.5124147;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%**********************************************case 36
%**********************************************
```

DOI: 10.18535/ijecs/v5i10.33

```
case 36%(20-gon)icosagon
syms COORD
 nside=20;
  coord
  nnn=0; tt(1:ndiv+1,1)=linspace(0,1,ndiv+1);
 % generate nodal coordinates for (nside+1) nodes of the boundary
 for side=1:nside
     COORD(side,1)=coord(side+1,1);
     COORD(side,2)=coord(side+1,2);
 end
  COORD(nside+1,1)=coord(2,1);
  COORD(nside+1,2)=coord(2,2);
  for jj=1:nside
   xni=COORD(jj,1);yni=COORD(jj,2);
   xnj=COORD(jj+1,1);ynj=COORD(jj+1,2);
   xtt(1:ndiv+1,jj)=xni+(xnj-xni)*tt;
   ytt(1:ndiv+1,jj)=yni+(ynj-yni)*tt;

  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
      for ii=1:ndiv+1
   if(xtt(ii,jj)==xnn)&(ytt(ii,jj)==ynn)
       nnn=nnn+1;
       bcdof(nnn,1)=nn;
       bcval(nnn,1)=0;
   end%if
      end%ii
  end%nn

  end%jj
  [bcdof(1:nnn,1) bcval(1:nnn,1)]
  mm=length(bcdof);
 format long g
k1 =1.5181239;%Hassenpflug
xi=(zeros(nnode,1));
 for m=1:nnode
xi(m,1)=1;%exact solutions not known
end
%**************************************************************



 end%mesh
    bcdof
    mm=length(bcdof);

 for L=1:nel
   for M=1:3
      LM=nodetel(L,M);
      xx(L,M)=gcoord(LM,1);
      yy(L,M)=gcoord(LM,2);
    end
  end
%________________________________________________________________
ng=10
 [sp,wt]=glsampleptsweights(ng)
table2=[N xx yy];
%disp([xx yy])
intJdn1dn1uvrs =[vpa(sym(' .595527655000821147485729267330')),
vpa(sym('.468322285820574645491168269290'));vpa(sym('
.46832228582057464549116826929')), vpa(sym('.595527655000821147485729267330'))];
```

```
intJdn1dn2uvrs =[vpa(sym('  -.397018436667214098323819511552')),
vpa(sym('.1877851427862835696725544871395'));vpa(sym(' -
.3122148572137164303274455128604')),   vpa(sym('.102981563332785901676180488448'))];
intJdn1dn3uvrs =[vpa(sym(' -.3014907816663929508380902442235')),vpa(sym(' -
.3438925713931417848362772435700'));vpa(sym(' -.3438925713931417848362772435700')),
vpa(sym('-.3014907816663929508380902442235'))];
intJdn1dn4uvrs =[vpa(sym('   .102981563332785901676180488448')), vpa(sym('-
.3122148572137164303274455128604'));vpa(sym('  .1877851427862835696725544871395')),
vpa(sym('-.397018436667214098323819511552'))];
%
intJdn2dn1uvrs =[vpa(sym('  -.397018436667214098323819511552')), vpa(sym('-
.3122148572137164303274455128604'));vpa(sym('  .1877851427862835696725544871395')),
vpa(sym(' .102981563332785901676180488448'))];
intJdn2dn2uvrs =[vpa(sym('   .264678957778142732215879674369')), vpa(sym('-
.1251900951908557131150363247600'));vpa(sym(' -.1251900951908557131150363247600')),
vpa(sym(' .264678957778142732215879674369'))];
intJdn2dn3uvrs =[vpa(sym('  .2009938544442619672253934961491')),
vpa(sym('.2292617142620945232241848290466'));vpa(sym(' -
.2707382857379054767758151709534')), vpa(sym('-.2990061455557380327746065038509'))];
intJdn2dn4uvrs =[vpa(sym(' -.68654375555190601117453658965e-1')),
vpa(sym('.2081432381424776202182970085734'));vpa(sym('
.2081432381424776202182970085734')), vpa(sym('-.68654375555190601117453658965e-1'))];
%
intJdn3dn1uvrs =[vpa(sym(' -.3014907816663929508380902442235')), vpa(sym('-
.3438925713931417848362772435700'));vpa(sym(' -
.3438925713931417848362772435700')),vpa(sym(' -.3014907816663929508380902442235'))];
intJdn3dn2uvrs =[vpa(sym('  .2009938544442619672253934961491')), vpa(sym('-
.2707382857379054767758151709534'));vpa(sym('  .2292617142620945232241848290466')),
vpa(sym('-.2990061455557380327746065038509'))];
intJdn3dn3uvrs =[vpa(sym(' .3995030727778690163873032519254')),
vpa(sym('.3853691428689527383879075854768'));vpa(sym('
.3853691428689527383879075854768')), vpa(sym('.3995030727778690163873032519254'))];
intJdn3dn4uvrs =[vpa(sym(' -.2990061455557380327746065038509')), vpa(sym('
.2292617142620945232241848290466'));vpa(sym(' -.2707382857379054767758151709534')),
vpa(sym('.2009938544442619672253934961491'))];
%
intJdn4dn1uvrs =[vpa(sym('   .102981563332785901676180488448')),
vpa(sym('.1877851427862835696725544871395'));vpa(sym(' -
.3122148572137164303274455128604')), vpa(sym(' -.397018436667214098323819511552'))];
intJdn4dn2uvrs =[vpa(sym(' -.68654375555190601117453658965e-1')),  vpa(sym('
.2081432381424776202182970085734'));vpa(sym('   .2081432381424776202182970085734')),
vpa(sym('-.68654375555190601117453658965e-1'))];
intJdn4dn3uvrs =[vpa(sym(' -.2990061455557380327746065038509')), vpa(sym('-
.2707382857379054767758151709534'));vpa(sym('  .2292617142620945232241848290466')),
vpa(sym('.2009938544442619672253934961491'))];
intJdn4dn4uvrs =[vpa(sym('   .264678957778142732215879674369')), vpa(sym('-
.1251900951908557131150363247600'));vpa(sym(' -.1251900951908557131150363247600')),
vpa(sym(' .264678957778142732215879674369'))];
%
%
intJdndn=double([intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs;...
          intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs;...
          intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs;...
          intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs]);



%
for iel=1:nel
index=zeros(nnel*ndof,1);

X=xx(iel,1:3);
Y=yy(iel,1:3);
%disp([X Y])
```

```
xa=X(1,1);
xb=X(1,2);
xc=X(1,3);
ya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
bta=yb-yc;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;
sk(1:4,1:4)=(zeros(4,4));
for i=1:4
 for j=i:4
 sk(i,j)=(delabc*sum(sum(Q.*(intJdndn(2*i-1:2*i,2*j-1:2*j)))));
 sk(j,i)=sk(i,j);
 end
end
%f =[5/144;1/24;7/144;1/24]*(2*delabc);

 xe(1,1)=(xa+xb+xc)/3;
 xe(2,1)=(xa+xc)/2;
 xe(3,1)=xa;
 xe(4,1)=(xa+xb)/2;
 %
 ye(1,1)=(ya+yb+yc)/3;
 ye(2,1)=(ya+yc)/2;
 ye(3,1)=ya;
 ye(4,1)=(ya+yb)/2;
 %
 [sp,wt]=glsampleptsweights(ng)
 %for j=1:4
 %    qe(j,1)=(2*pi^2)*sin(pi*xe(j,1))*sin(pi*ye(j,1));
 %end
 %II =([  1/72, 7/864, 1/216, 7/864;...
 %     7/864,  1/54,  1/96, 1/216;...
  %    1/216,  1/96, 5/216,  1/96;...
   %   7/864, 1/216,  1/96,  1/54]);
 %f=(2*delabc)*(II*qe);
 xe1=xe(1,1);xe2=xe(2,1);xe3=xe(3,1);xe4=xe(4,1);
 ye1=ye(1,1);ye2=ye(2,1);ye3=ye(3,1);ye4=ye(4,1);
 f(1:4,1)=zeros(4,1)

 if (mesh==1)|(mesh==2)
 for i=1:ng
     si=sp(i,1);wi=wt(i,1);
     for j=1:ng
         sj=sp(j,1);wj=wt(j,1);
         n1ij=(1-si)*(1-sj)/4;
         n2ij=(1+si)*(1-sj)/4;
         n3ij=(1+si)*(1+sj)/4;
         n4ij=(1-si)*(1+sj)/4;
         xeij=xe1*n1ij+xe2*n2ij+xe3*n3ij+xe4*n4ij;
         yeij=ye1*n1ij+ye2*n2ij+ye3*n3ij+ye4*n4ij;

        f1i=n1ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
        f2i=n2ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
        f3i=n3ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
        f4i=n4ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;


          f(1,1)=f(1,1)+f1i*wi*wj;
          f(2,1)=f(2,1)+f2i*wi*wj;
          f(3,1)=f(3,1)+f3i*wi*wj;
          f(4,1)=f(4,1)+f4i*wi*wj;
      end
  end
 f=(delabc)*f;
 end%(mesh==1)|(mesh==2)
 if
(mesh==18)|(mesh==19)|(mesh==20)|(mesh==21)|(mesh==22)|(mesh==23)|(mesh==24)|(mesh==25)
|(mesh==26)|(mesh==27)|(mesh==28)|(mesh==29)|(mesh==30)|(mesh==31)|(mesh==32)|(mesh==33
)|(mesh==34)|(mesh==35)|(mesh==36)
     f =[5/144;1/24;7/144;1/24]*(delabc);
 end

%_______________________________________
 edof=nnel*ndof;
 k=0;
 for i=1:nnel
     nd(i,1)=nodes(iel,i);
     start=(nd(i,1)-1)*ndof;
     for j=1:ndof
         k=k+1;
         index(k,1)=start+j;
     end
 end
 %------------------------------------------------------------------------
for i=1:edof
    ii=index(i,1);
    ff(ii,1)=ff(ii,1)+f(i,1);
 for j=1:edof
     jj=index(j,1);
     ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
 %------------------------------------------------------------------------
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
%apply boundary conditions

%
mm=length(bcdof);
sdof=size(ss);
%
for i=1:mm
c=bcdof(i,1);
for j=1:sdof
ss(c,j)=0;
end
%
ss(c,c)=1;
ff(c,1)=bcval(i,1);
end
%solve the equations

phi=ss\ff;
for I=1:nnode
NN(I,1)=I;
end

disp('__________________________________________________________________')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
disp('_________________________________________________________________________________
____')
```

```
disp('                         fem-computed values             anlytical(theoretical)-
values           ')

disp([NN phi xi])
disp('_________________________________________________________________________________
____')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
nodes
gcoord
%for (mesh==1) & (mesh==2) generate meshgrid coordinates
if (mesh==1)|(mesh==2)
[x,y]=meshgrid(0:0.1:1,0:0.1:1);

for i=1:11
    for j=1:11
      for iel=1:nel
 %node numbers of quadrilateral
       nd1=nodes(iel,1);nd2=nodes(iel,2);nd3=nodes(iel,3);nd4=nodes(iel,4);
 %coordinates of quadrilateral(u,v)

u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);u(3,1)=gcoord(nd3,1);u(4,1)=gcoord(nd4,1);

v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);v(3,1)=gcoord(nd3,2);v(4,1)=gcoord(nd4,2);
 %coordinates of the grid(x,y)

          in=inpolygon(x(i,j),y(i,j),u,v);
          if (in==1)
           X=x(i,j);Y=y(i,j);
           [t]=convexquadrilateral_coordinates(u,v,X,Y);
           r=t(1,1);
           s=t(2,1);
             PHI(i,j)=(1-r)*(1-s)*phi(nd1,1)/4+(1+r)*(1-
s)*phi(nd2,1)/4+(1+r)*(1+s)*phi(nd3,1)/4+(1-r)*(1+s)*phi(nd4,1)/4;
              break
           end%if (in==1)
        end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
  z=sin(pi*x).*sin(pi*y);

for i=1:11
    for j=1:11
        if (abs(PHI(i,j))<=1e-5)
            PHI(i,j)=0;
        end
        if (abs(z(i,j))<=1e-5)
            z(i,j)=0;
        end

    end
end
end%if (mesh==1)&(mesh==2)
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
if (mesh==18)

[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
 z=(zeros(31,31));
for i=1:31
    for j=1:31
      for iel=1:nel
 %node numbers of quadrilateral
       nd1=nodes(iel,1);nd2=nodes(iel,2);nd3=nodes(iel,3);nd4=nodes(iel,4);
 %coordinates of quadrilateral(u,v)

u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);u(3,1)=gcoord(nd3,1);u(4,1)=gcoord(nd4,1);

v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);v(3,1)=gcoord(nd3,2);v(4,1)=gcoord(nd4,2);
 %coordinates of the grid(x,y)

          in=inpolygon(x(i,j),y(i,j),u,v);
          if (in==1)
           X=x(i,j);Y=y(i,j);
           [t]=convexquadrilateral_coordinates(u,v,X,Y);
           r=t(1,1);
           s=t(2,1);
             PHI(i,j)=(1-r)*(1-s)*phi(nd1,1)/4+(1+r)*(1-
s)*phi(nd2,1)/4+(1+r)*(1+s)*phi(nd3,1)/4+(1-r)*(1+s)*phi(nd4,1)/4;
             z(i,j)=((Y+1)*((sqrt(3))*X-Y+2)*(-(sqrt(3))*X-Y+2))/12;;
              break
           end%if (in==1)
        end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
 % z=sin(pi*x).*sin(pi*y);
  %z=(zeros(31,31));

%for ii=1:31
 %   for jj=1:31
  %  xx=(x(ii,jj));yy=(y(ii,jj));
%z(ii,jj)=((yy+1/2)*((sqrt(3))*xx-yy+1)*(-(sqrt(3))*xx-yy+1))/6;;
%end %ii
%end%jj

for i=1:31
    for j=1:31
        if (abs(PHI(i,j))<=1e-5)
            PHI(i,j)=0;
        end
        if (abs(z(i,j))<=1e-5)
            z(i,j)=0;
        end

    end
end









end%(mesh==18)
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
if (mesh==19)
 [x,y]=meshgrid(-sqrt(3)/2:(1/15)*sqrt(3)/2:sqrt(3)/2,-1/2:(0.1)*(1/2):1);
 z=(zeros(31,31));
for i=1:31
    for j=1:31
      for iel=1:nel
 %node numbers of quadrilateral
```

```
       nd1=nodes(iel,1);nd2=nodes(iel,2);nd3=nodes(iel,3);nd4=nodes(iel,4);
 %coordinates of quadrilateral(u,v)

u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);u(3,1)=gcoord(nd3,1);u(4,1)=gcoord(nd4,1);

v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);v(3,1)=gcoord(nd3,2);v(4,1)=gcoord(nd4,2);
 %coordinates of the grid(x,y)

          in=inpolygon(x(i,j),y(i,j),u,v);
          if (in==1)
           X=x(i,j);Y=y(i,j);
           [t]=convexquadrilateral_coordinates(u,v,X,Y);
           r=t(1,1);
           s=t(2,1);
             PHI(i,j)=(1-r)*(1-s)*phi(nd1,1)/4+(1+r)*(1-
s)*phi(nd2,1)/4+(1+r)*(1+s)*phi(nd3,1)/4+(1-r)*(1+s)*phi(nd4,1)/4;
             z(i,j)=((Y+1/2)*((sqrt(3))*X-Y+1)*(-(sqrt(3))*X-Y+1))/6;;
              break
           end%if (in==1)
        end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
 % z=sin(pi*x).*sin(pi*y);
  %z=(zeros(31,31));

%for ii=1:31
 %   for jj=1:31
  %  xx=(x(ii,jj));yy=(y(ii,jj));
%z(ii,jj)=((yy+1/2)*((sqrt(3))*xx-yy+1)*(-(sqrt(3))*xx-yy+1))/6;;
%end %ii
%end%jj

for i=1:31
    for j=1:31
        if (abs(PHI(i,j))<=1e-5)
            PHI(i,j)=0;
        end
        if (abs(z(i,j))<=1e-5)
            z(i,j)=0;
        end

    end
end


end%(mesh==19)
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if (mesh==20)
%******
[x,y]=meshgrid(-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2,-
sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2);

for i=1:21
    for j=1:21
      for iel=1:nel
 %node numbers of quadrilateral
       nd1=nodes(iel,1);nd2=nodes(iel,2);nd3=nodes(iel,3);nd4=nodes(iel,4);
 %coordinates of quadrilateral(u,v)

u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);u(3,1)=gcoord(nd3,1);u(4,1)=gcoord(nd4,1);
```

```
v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);v(3,1)=gcoord(nd3,2);v(4,1)=gcoord(nd4,2);
 %coordinates of the grid(x,y)

          in=inpolygon(x(i,j),y(i,j),u,v);
          if (in==1)
           X=x(i,j);Y=y(i,j);
           [t]=convexquadrilateral_coordinates(u,v,X,Y);
           r=t(1,1);
           s=t(2,1);
             PHI(i,j)=(1-r)*(1-s)*phi(nd1,1)/4+(1+r)*(1-
s)*phi(nd2,1)/4+(1+r)*(1+s)*phi(nd3,1)/4+(1-r)*(1+s)*phi(nd4,1)/4;
              break
           end%if (in==1)
        end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
 % z=sin(pi*x).*sin(pi*y);
  z=(zeros(21,21));
 a2=sqrt(2);aa=1/4;
a0=32*aa/pi^3;
for ii=1:21
    for jj=1:21
    xx=(x(ii,jj));yy=(y(ii,jj));rr=(0);

for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*yy/a2)/cosh(n*pi/2)))*cos(n*pi*xx/a2)/n^3;
end
z(ii,jj)=(a0*rr);
end %ii
end%jj

for i=1:21
    for j=1:21
        if (abs(PHI(i,j))<=1e-5)
            PHI(i,j)=0;
        end
        if (abs(z(i,j))<=1e-5)
            z(i,j)=0;
        end

    end
end

%*******
end%(mesh==20)
%********************************
%regular polygons inscribed in a circle of radius =1
%number in the bracket mentioned to indicate mesh
%regular pentagon(21),hexagon(22),heptagon(23),  inscribed in a circle of radius =1
if
(mesh==21)|(mesh==22)|(mesh==23)|(mesh==24)|(mesh==25)|(mesh==26)|(mesh==27)|(mesh==28)
|(mesh==29)|(mesh==30)|(mesh==31)|(mesh==32)|(mesh==33)|(mesh==34)|(mesh==35)|(mesh==36
)
 [x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
 %z=(zeros(21,21));PHI(i,j)=(zeros(21,21));
for i=1:21
    for j=1:21
      for iel=1:nel
 %node numbers of quadrilateral
       nd1=nodes(iel,1);nd2=nodes(iel,2);nd3=nodes(iel,3);nd4=nodes(iel,4);
 %coordinates of quadrilateral(u,v)
```

```
u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);u(3,1)=gcoord(nd3,1);u(4,1)=gcoord(nd4,1);

v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);v(3,1)=gcoord(nd3,2);v(4,1)=gcoord(nd4,2);
 %coordinates of the grid(x,y)

          in=inpolygon(x(i,j),y(i,j),u,v);
          if in==0
              PHI(i,j)=NaN;
          end
          if (in==1)
           X=x(i,j);Y=y(i,j);
           [t]=convexquadrilateral_coordinates(u,v,X,Y);
           r=t(1,1);
           s=t(2,1);
             PHI(i,j)=(1-r)*(1-s)*phi(nd1,1)/4+(1+r)*(1-
s)*phi(nd2,1)/4+(1+r)*(1+s)*phi(nd3,1)/4+(1-r)*(1+s)*phi(nd4,1)/4;

              break
           end%if (in==1)
        end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
 % z=sin(pi*x).*sin(pi*y);
  %z=(zeros(31,31));

%for ii=1:31
 %   for jj=1:31
  %  xx=(x(ii,jj));yy=(y(ii,jj));
%z(ii,jj)=((yy+1/2)*((sqrt(3))*xx-yy+1)*(-(sqrt(3))*xx-yy+1))/6;;
%end %ii
%end%jj
 [M,N]=size(PHI)
for i=1:M
    for j=1:N
        if (abs(PHI(i,j))<=1e-5)
            PHI(i,j)=0;
        end
       % if (abs(z(i,j))<=1e-5)
        %    z(i,j)=0;
        %end

    end
end

end%if
(mesh==21)|(mesh==22)|(mesh==23)|(mesh==24)|(mesh==25)|(mesh==26)|(mesh==27)|(mesh==28)
|(mesh==29)|(mesh==30)ETC
%********************************8

switch mesh
case 1
 clf
figure(1)
 x=[0.0 1.0 1.0 0.5 0.0];
 y=[0.0 0.0 0.5 1.0 1.0];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.1:1,0:0.1:1)
y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
[c,h]=contour(x,y,PHI)
xlabel('X-axis');
```

```
ylabel('Y-axis');
clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
%
figure(2)
 x=[0.0 1.0 1.0 0.5 0.0];
 y=[0.0 0.0 0.5 1.0 1.0];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.1:1,0:0.1:1)
y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
[c,h]=contour(x,y,z)
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
title('contour level curves for exact solution: sin(pi*x)*sin(pi*y)')
hold off
%///////////////////////////////////////////////

figure(3)
 x=[0.0 1.0 1.0 0.5 0.0];
 y=[0.0 0.0 0.5 1.0 1.0];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.1:1,0:0.1:1)
y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
[c,h]=contour(x,y,PHI,'r:')

xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1=' NODES='
sst2=num2str(nnode)
sst3=' ELEMENTS='
sst4=num2str(nel)
text(0.7,.85,[sst1 sst2])
text(0.7,.8,[sst3 sst4])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,z,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
clabel(c,h);
```

```
axis square
text(0.65,.99,'{ SUPERPOSITION OF }')
text(0.65,.95,'{ FEM AND EXACT SOLUTIONS}')
text(0.65,.9,'{...(red)FEM and--(green)EXACT}')

 mm=0;
 for i=1:11
  for j=1:11
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
  end
end

 [femsoln exactsoln]
case 2
 clf
figure(1)
[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,PHI)
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded  ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
figure(2)
[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,z)
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
title('contour level curves for exact solution: sin(pi*x)*sin(pi*y)')
hold off
%
figure(3)
[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,PHI,'r:')
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,z,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
clabel(c,h);
axis square

 mm=0;
 for i=1:11
  for j=1:11
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
  end
 end
 [femsoln exactsoln]
%+++++++++++++++++++++++++++++++++++++++++++
    case 18
  clf
figure(1)
 x=[-sqrt(3);sqrt(3);0];
 y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
%[c,h]=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.6,1.8,[sst1 sst2])
text(0.6,1.6,[sst3 sst4])
text(0.6,1.4,[sst5 sst6])
%text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
%
figure(2)
 %x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
 x=[-sqrt(3);sqrt(3);0];
 y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
```

```
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%[c,h]=contour(x,y,z)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);

contour(x,y,z,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
title('contour level curves for exact solution: ')
hold off

figure(3)
% x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
x=[-sqrt(3);sqrt(3);0];
y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
contour(x,y,PHI,'r-')

xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1=' NODES='
sst2=num2str(nnode)
sst3=' ELEMENTS='
sst4=num2str(nel)
text(0.6,1.1,[sst1 sst2])
text(0.6,.9,[sst3 sst4])

hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,'b-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square
text(0.6,1.9,'{ SUPERPOSITION OF }')
text(0.6,1.7,'{ FEM/EXACT SOLUTIONS}')
text(0.6,1.5,'--(red)FEM ')
text(0.6,1.3,'--(blue)EXACT')

 mm=0;
 for i=1:31
  for j=1:31
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
  end
end
```

```
 [femsoln exactsoln]





%+++++++++++++++++++++++++++++++++++++++++++++++
case 19
%***************************
  clf
figure(1)
 x=[-sqrt(3)/2;sqrt(3)/2;0];
 y=[      -1/2;     -1/2;1];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[x,y]=meshgrid(-sqrt(3)/2:(1/15)*sqrt(3)/2:sqrt(3)/2,-1/2:(0.1)*(1/2):1);

%[c,h]=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
%text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
text(0.4,.95,[sst1 sst2])
text(0.4,.85,[sst3 sst4])
text(0.4,.75,[sst5 sst6])

%
figure(2)
 %x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
 x=[-sqrt(3)/2;sqrt(3)/2;0];
 y=[      -1/2;     -1/2;1];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%[c,h]=contour(x,y,z)
[x,y]=meshgrid(-sqrt(3)/2:(1/15)*sqrt(3)/2:sqrt(3)/2,-1/2:(0.1)*(1/2):1);


contour(x,y,z)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
title('contour level curves for exact solution: sin(pi*x)*sin(pi*y)')
hold off
```

```
figure(3)
% x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
x=[-sqrt(3)/2;sqrt(3)/2;0];
y=[      -1/2;     -1/2;1];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-sqrt(3)/2:(1/15)*sqrt(3)/2:sqrt(3)/2,-1/2:(0.1)*(1/2):1);

contour(x,y,PHI,'r-')

xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1=' NODES='
sst2=num2str(nnode)
sst3=' ELEMENTS='
sst4=num2str(nel)
%text(0.7,.85,[sst1 sst2])
%text(0.7,.8,[sst3 sst4])
text(0.4,.55,[sst1 sst2])
text(0.4,.45,[sst3 sst4])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,'b-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square
%text(0.65,.99,'{ SUPERPOSITION OF }')
%text(0.65,.95,'{ FEM AND EXACT SOLUTIONS}')
%text(0.65,.9,'{...(red)FEM and--(green)EXACT}')
text(0.4,.95,'{ SUPERPOSITION OF }')
text(0.4,.85,'{ FEM/EXACT SOLUTIONS}')
text(0.4,.75,'--(red)FEM ')
text(0.4,.65,'--(blue)EXACT')

 mm=0;
 for i=1:31
  for j=1:31
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
  end
end

[femsoln exactsoln]

%***********************
  case 20
  clf
figure(1)
[x,y]=meshgrid(-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2,-
sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2);
```

```
%[c,h]=contour(x,y,PHI,'k')
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded  ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
figure(2)
[x,y]=meshgrid(-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2,-
sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2);
%[c,h]=contour(x,y,z,'b')
contour(x,y,z,'g-')
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
title('contour level curves for exact solution:')
hold off

figure(3)
[x,y]=meshgrid(-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2,-
sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2);
%[c,h]=contour(x,y,PHI,'r-')
contour(x,y,PHI,'r-')
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square

 mm=0;
 for i=1:21
```

```
  for j=1:21
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
  end
 end
 [femsoln exactsoln]
 %*******************
 case 21%pentagon
   %clf
figure(1)
 %x=[-sqrt(3)/2;sqrt(3)/2;0];
 %y=[      -1/2;     -1/2;1];

  %       2    3       4           5            6
x=([cos(pi/10);0;-cos(pi/10);-cos(3*pi/10); cos(3*pi/10)])
y=([sin(pi/10);1; sin(pi/10);-sin(3*pi/10);-sin(3*pi/10)])
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
%[c,h]=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-.9,[sst1 sst2 sst3 sst4 sst5 sst6])
hold on
%************************************
 case 22%hexagon
   %clf
figure(2)
 %         2          3      4        5        6           7
    x=sym([1;cos(pi/3);-cos(pi/3);-1; -cos(pi/3); cos(pi/3)])
    y=sym([0;sin(pi/3); sin(pi/3); 0; -sin(pi/3);-sin(pi/3)])
patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
```

```
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,.9,[sst1 sst2 sst3 sst4 sst5 sst6])
%*************************************
 case 23%regular heptagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(3)
 %         2            3              4            5       6       7         8
x=sym([-cos(5*pi/14); cos(5*pi/14); cos(pi/14);cos(3*pi/14);0;-cos(3*pi/14);-
cos(pi/14)])
y=sym([-sin(5*pi/14);-sin(5*pi/14);-sin(pi/14);sin(3*pi/14);1; sin(3*pi/14);-
sin(pi/14)])
patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
%**********************************************
  case 24%regular octagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(4)
  %    2     3     4     5       6       7     8     9
x=sym([1;cos(pi/4);0;-cos(pi/4);-1;-cos(pi/4); 0; cos(pi/4)])
y=sym([0;sin(pi/4);1; sin(pi/4); 0;-sin(pi/4);-1;-sin(pi/4)])
patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
```

```
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
%*****************************************
  case 25%regular nonagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(5)
 %          2              3      4          5         6            7             8
9              10
  x=sym([cos(pi/18); cos(5*pi/18);0;-cos(5*pi/18);-cos(pi/18);-cos(3*pi/18);-
cos(7*pi/18); cos(7*pi/18); cos(3*pi/18)])
  y=sym([sin(pi/18); sin(5*pi/18);1; sin(5*pi/18); sin(pi/18);-sin(3*pi/18);-
sin(7*pi/18);-sin(7*pi/18);-sin(3*pi/18)])
patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
%*******************************************
 case 26%regular decaagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(6)
 %       2     3          4          5              6        7     8           9
10       11
  x=sym([1;cos(pi/5); cos(2*pi/5);-cos(2*pi/5);-cos(pi/5);-1;-cos(pi/5);-cos(2*pi/5);
cos(2*pi/5); cos(pi/5)])
  y=sym([0;sin(pi/5); sin(2*pi/5); sin(2*pi/5); sin(pi/5); 0;-sin(pi/5);-sin(2*pi/5);-
sin(2*pi/5);-sin(pi/5)])
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
```

```
%*****************************************
case 27%regular hendecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(7)
 %             2          3       4          5          6               7        8
9              10                11      12
  x=sym([cos(3*pi/22);cos(7*pi/22);0;-cos(7*pi/22);-cos(3*pi/22);-cos(pi/22);-
cos(5*pi/22);-cos(9*pi/22); cos(9*pi/22); cos(5*pi/22); cos(pi/22)])
  y=sym([sin(3*pi/22);sin(7*pi/22);1; sin(7*pi/22); sin(3*pi/22);-sin(pi/22);-
sin(5*pi/22);-sin(9*pi/22);-sin(9*pi/22);-sin(5*pi/22);-sin(pi/22)])
  patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1.1,[sst1 sst2 sst3 sst4 sst5 sst6])
%********************************************
case 28%regular dodecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(8)
  %      2   3           4     5      6          7      8       9         10    11
12     13
  x=sym([1;cos(pi/6);cos(pi/3);0;-cos(pi/3);-cos(pi/6);-1;-cos(pi/6);-cos(pi/3); 0;
cos(pi/3); cos(pi/6)])
  y=sym([0;sin(pi/6);sin(pi/3);1; sin(pi/3); sin(pi/6); 0;-sin(pi/6);-sin(pi/3);-1;-
sin(pi/3);-sin(pi/6)])
  patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
%**********************************************
```

```
 case 29%regular tridecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(9)
  %            2          3            4         5      6                  7             8
9                 10             11                12           13             14
  x=sym([cos(pi/26);cos(5*pi/26);cos(9*pi/26);0;-cos(9*pi/26);-cos(5*pi/26);-
cos(pi/26);-cos(3*pi/26);-cos(7*pi/26);-cos(11*pi/26); cos(11*pi/26); cos(7*pi/26);
cos(3*pi/26)])
  y=sym([sin(pi/26);sin(5*pi/26);sin(9*pi/26);1; sin(9*pi/26); sin(5*pi/26);
sin(pi/26);-sin(3*pi/26);-sin(7*pi/26);-sin(11*pi/26);-sin(11*pi/26);-sin(7*pi/26);-
sin(3*pi/26)])
  patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1,[sst1 sst2 sst3 sst4 sst5 sst6])
%************************************************case 30
case 30%regular tetradecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(10)
 %      2          3          4             5          6                  7             8
9       10             11               12           13             14             15
  x=sym([1;cos(2*pi/14);cos(4*pi/14);cos(6*pi/14);-cos(6*pi/14);-cos(4*pi/14);-
cos(2*pi/14);-1;-cos(2*pi/14);-cos(4*pi/14);-cos(6*pi/14); cos(6*pi/14); cos(4*pi/14);
cos(2*pi/14)])
  y=sym([0;sin(2*pi/14);sin(4*pi/14);sin(6*pi/14); sin(6*pi/14); sin(4*pi/14);
sin(2*pi/14); 0;-sin(2*pi/14);-sin(4*pi/14);-sin(6*pi/14);-sin(6*pi/14);-sin(4*pi/14);-
sin(2*pi/14)])
  patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
```

```
%**************************************************case 31
case 31%regular pentadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(11)
  %            2         3              4             5       6    7                  8
9       10            11              12          13            14            15
16
   x=sym([ cos(pi/30);cos(3*pi/30);cos(7*pi/30);cos(11*pi/30);0;-cos(11*pi/30);-
cos(7*pi/30);-cos(3*pi/30);-cos(pi/30);-cos(5*pi/30);-cos(9*pi/30);-cos(13*pi/30);
cos(13*pi/30); cos(9*pi/30); cos(5*pi/30)])
   y=sym([-sin(pi/30);sin(3*pi/30);sin(7*pi/30);sin(11*pi/30);1; sin(11*pi/30);
sin(7*pi/30); sin(3*pi/30);-sin(pi/30);-sin(5*pi/30);-sin(9*pi/30);-sin(13*pi/30);-
sin(13*pi/30);-sin(9*pi/30);-sin(5*pi/30)])
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES  '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
%**************************************************CASE 32
case 32%regular hexadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(12)
   %      2     3       4           5       6      7              8        9     10
11          12          13    14       15           16        17
   x=sym([1;cos(pi/8);cos(pi/4);cos(3*pi/8);0;-cos(3*pi/8);-cos(pi/4);-cos(pi/8);-1;-
cos(pi/8);-cos(pi/4);-cos(3*pi/8); 0; cos(3*pi/8); cos(pi/4); cos(pi/8)]);
   y=sym([0;sin(pi/8);sin(pi/4);sin(3*pi/8);1; sin(3*pi/8); sin(pi/4); sin(pi/8); 0;-
sin(pi/8);-sin(pi/4);-sin(3*pi/8);-1;-sin(3*pi/8);-sin(pi/4);-sin(pi/8)]);
  patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES  '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
```

```
%*********************************************case 33
case 33%regular heptadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(13)
 %
  %            2            3        4              5       6           7              8
9              10           11           12                13               14
15             16               17       18
   x=sym([cos(pi/34);cos(5*pi/34);cos(9*pi/34);cos(13*pi/34);0;-cos(13*pi/34);-
cos(9*pi/34);-cos(5*pi/34);-cos(pi/34);-cos(3*pi/34);-cos(7*pi/34);-cos(11*pi/34);-
cos(15*pi/34); cos(15*pi/34); cos(11*pi/34); cos(7*pi/34); cos(3*pi/34)]);
   y=sym([sin(pi/34);sin(5*pi/34);sin(9*pi/34);sin(13*pi/34);1; sin(13*pi/34);
sin(9*pi/34); sin(5*pi/34); sin(pi/34);-sin(3*pi/34);-sin(7*pi/34);-sin(11*pi/34);-
sin(15*pi/34);-sin(15*pi/34);-sin(11*pi/34);-sin(7*pi/34);-sin(3*pi/34)]);
   patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='(MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES  '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
%******************************************************case 34
case 34%regular octadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(14)
%      2           3       4              5              6           7              8
9             10        11          12            13               14            15
16               17       18              19
 x=sym([1;cos(4*pi/36);cos(8*pi/36);cos(12*pi/36);cos(16*pi/36);-cos(16*pi/36);-
cos(12*pi/36);-cos(8*pi/36);-cos(4*pi/36);-1;-cos(4*pi/36);-cos(8*pi/36);-
cos(12*pi/36);-cos(16*pi/36); cos(16*pi/36); cos(12*pi/36); cos(8*pi/36);
cos(4*pi/36)]);
 y=sym([0;sin(4*pi/36);sin(8*pi/36);sin(12*pi/36);sin(16*pi/36); sin(16*pi/36);
sin(12*pi/36); sin(8*pi/36); sin(4*pi/36); 0;-sin(4*pi/36);-sin(8*pi/36);-
sin(12*pi/36);-sin(16*pi/36);-sin(16*pi/36);-sin(12*pi/36);-sin(8*pi/36);-
sin(4*pi/36)]);
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='  (MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES  '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
%***********************************************************case 35
case 35%regular enneadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(15)
   %          2            3           4             5        6      7               8
9             10              11         12            13              14             15
16                  17        18              19             20
 x=sym([cos(3*pi/38);cos(7*pi/38);cos(11*pi/38);cos(15*pi/38);0;-cos(15*pi/38);-
cos(11*pi/38);-cos(7*pi/38);-cos(3*pi/38);-cos(pi/38);-cos(5*pi/38);-cos(9*pi/38);-
cos(13*pi/38);-cos(17*pi/38); cos(17*pi/38); cos(13*pi/38); cos(9*pi/38); cos(5*pi/38);
cos(pi/38)]);
 y=sym([sin(3*pi/38);sin(7*pi/38);sin(11*pi/38);sin(15*pi/38);1; sin(15*pi/38);
sin(11*pi/38); sin(7*pi/38); sin(3*pi/38);-sin(pi/38);-sin(5*pi/38);-sin(9*pi/38);-
sin(13*pi/38);-sin(17*pi/38);-sin(17*pi/38);-sin(13*pi/38);-sin(9*pi/38);-
sin(5*pi/38);-sin(pi/38)]);
patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='  (MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES  '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])
%***********************************************************case 36
case 36%regular icosadecagon inscribed in a circle of radius=1,required for torsion
analysis
 figure(16)
   %  2     3           4              5           6      7           8              9
10          11   12      13            14            15           16      17        18
19             20      21
x=sym([1;cos(pi/10);cos(2*pi/10);cos(3*pi/10);cos(4*pi/10);0;-cos(4*pi/10);-
cos(3*pi/10);-cos(2*pi/10);-cos(pi/10);-1;-cos(pi/10);-cos(2*pi/10);-cos(3*pi/10);-
cos(4*pi/10); 0; cos(4*pi/10); cos(3*pi/10); cos(2*pi/10); cos(pi/10)]);
y=sym([0;sin(pi/10);sin(2*pi/10);sin(3*pi/10);sin(4*pi/10);1; sin(4*pi/10);
sin(3*pi/10); sin(2*pi/10); sin(pi/10); 0;-sin(pi/10);-sin(2*pi/10);-sin(3*pi/10);-
sin(4*pi/10);-1;-sin(4*pi/10);-sin(3*pi/10);-sin(2*pi/10);-sin(pi/10)]);

patch(x,y,'w')
hold on
[x,y]=meshgrid(-1:0.1:1,-1:(0.1):1);
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
```

```
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Four Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1,st2,st3,st4,st5])
sst1='  (MESH HAS    '
sst2=num2str(nnode)
sst3=' NODES   '
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.1,-1.2,[sst1 sst2 sst3 sst4 sst5 sst6])


end%switch mesh
%[femsoln exactsoln]
 disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
[1 phi(1,1) xi(1,1)]
```

(3) PROGRAM-3

```
function[]=quadrilateral_mesh4MOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
clf
%(1)=generate 2-D quadrilateral mesh
%for a rectangular shape of domain
%quadrilateral_mesh_q4(xlength,ylength)
%xnode=number of nodes along x-axis
%ynode=number of nodes along y-axis
%xzero=x-coord of bottom left corner
%yzero=y-coord of bottom left corner
%xlength=size of domain alog x-axis
%ylength=size of domain alog y-axis
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1],[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],[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],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,9,6,1,1
,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4
,4,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,9
,6,2,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1
,2,4,1,1)
%quadrilateral_mesh4MOINEX_q4([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_mesh4MOINEX_q4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,4
,4,9,1,1)
%quadrilateral_mesh4MOINEX_q4([9;9;9;9;9;9;9;9],[1;2;3;4;5;6;7;8],[2;3;4;5;6;7;8;1],9,9
,6,9,1,1)
%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
,2,10,1,1)
%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
,4,10,1,1)
%quadrilateral_mesh4MOINEX_q4([1],[2],[3],3,1,2,6,1,1)
%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],9,2
5,10,2,1,1)
%%quadrilateral_mesh4MOINEX_q4([1],[2],[3],3,1,2,17,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1],[2;3;4;5;6],[3;4;5;6;2],6,9,6,21,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1],[2;3;4;5;6;7],[3;4;5;6;7;2],7,9,6,22,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,23,
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],9,2
5,10,24,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10],[3;4;5;6;7;8;9;1
0;2],10,1,2,25,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11],[3;4;5;6;7;
8;9;10;11;2],11,1,2,26,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12],[3;4;5
;6;7;8;9;10;11;12;2],12,9,6,27,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13],[
3;4;5;6;7;8;9;10;11;12;13;2],13,9,6,28,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;13;
14],[3;4;5;6;7;8;9;10;11;12;13;14;2],14,1,2,29,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12;1
3;14;15],[3;4;5;6;7;8;9;10;11;12;13;14;15;2],15,1,2,30,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;12
;13;14;15;16],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;2],16,1,2,31,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;11;
12;13;14;15;16;17],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;2],17,1,2,32,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;1
1;12;13;14;15;16;17;18],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;2],18,1,2,33,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10;1
1;12;13;14;15;16;17;18],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;2],18,1,2,33,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;10
;11;12;13;14;15;16;17;18;19],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;2],19,1,2,34,
1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9;
10;11;12;13;14;15;16;17;18;19;20],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;2],20
,1,2,35,1,1)
%quadrilateral_mesh4MOINEX_q4([1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;
9;10;11;12;13;14;15;16;17;18;19;20;21],[3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;
21;2],21,1,2,36,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtr
i,ndiv,mesh)
[nel,nnel]=size(nodes);

disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where i->node no. and j->x or y
%_______________________________________________________________
%
%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
%extract coordinates for each element
%clf
figure(ndiv/2)
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
axis equal
%axis tight
rtext=0;
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 12
    axis([-xlength xlength -ylength ylength])

case 17
    axis([-xlength xlength 0 ylength])
     rpoly=' one isosceles triangle in a  square';
    rtext=1;
case 18
    axis([-2*xlength 2*xlength -ylength 2*ylength])
    rpoly='   equilateral triangle';
    rtext=1;
case 19
    axis([-xlength xlength -ylength/2 ylength])
     rpoly='   equilateral triangle';
    rtext=1;

 case 20
    axis([-xlength xlength -ylength ylength])
     rpoly='    square';
    rtext=1;
 case 21
    axis([-xlength xlength -ylength ylength])
     rpoly='     pentagon';
    rtext=1;
 case 22
    axis([-xlength xlength -ylength ylength])
     rpoly='     hexagon';
    rtext=1;
  case 23
    axis([-xlength xlength -ylength ylength])
     rpoly='     heptagon';
    rtext=1;
    case 24
    axis([-xlength xlength -ylength ylength])
     rpoly='     octagon';
    rtext=1;
     case 25
    axis([-xlength xlength -ylength ylength])
     rpoly='     nonadecagon';
    rtext=1;
     case 26
    axis([-xlength xlength -ylength ylength])
     rpoly='     decagon';
     rtext=1;
     case 27
    axis([-xlength xlength -ylength ylength])
     rpoly='     hendecagon';
    rtext=1;
     case 28
    axis([-xlength xlength -ylength ylength])
     rpoly='     dodecagon';
    rtext=1;
     case 29
    axis([-xlength xlength -ylength ylength])
     rpoly='     tridecagon';
    rtext=1;
     case 30
    axis([-xlength xlength -ylength ylength])
     rpoly='     tetradecagon';
    rtext=1;
     case 31
    axis([-xlength xlength -ylength ylength])
     rpoly='     pentadecagon';
    rtext=1;
      case 32
    axis([-xlength xlength -ylength ylength])
     rpoly='     hexadecagon';
    rtext=1;
      case 33
    axis([-xlength xlength -ylength ylength])
     rpoly='     heptadecagon';
    rtext=1;
     case 34
    axis([-xlength xlength -ylength ylength])
    rpoly='     octadecagon';
    rtext=1;
     case 35
    axis([-xlength xlength -ylength ylength])
    rpoly='     enneadecagon';
    rtext=1;
  case 36
    axis([-xlength xlength -ylength ylength])
    rpoly='     icosagon';
    rtext=1;

end
plot(xvec,yvec);%plot element
hold on;
%place element number
if ndiv<=4
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['[',num2str(i),']']);
end
end;%i loop

xlabel('x axis')
ylabel('y axis')
st1='FEM MESH ';
st2=num2str(nel);
st3=' four noded  ';
st4='quadrilateral';
st5=' elements'
st6='& nodes='
st7=num2str(nnode);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
if ndiv<=4
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end
if rtext==1
  text(0.1,-1.2,rpoly)
end
%axis off
```

(4) PROGRAM-4

```
function[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nm
ax,numtri,n,mesh)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.......i.e number of divisions
```

```
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the
polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=
2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,9,6)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4
;5;2],5,16,8)
%PARVIZ MOIN EXAMPLE
syms U V W xi yi

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



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

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

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

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

```
y=([0;0;1;1;0.5])
case 12 %a 2-square [-1,1]x[-1,1]
 %       1    2   3   4   5   6    7   8    9
  x=sym([0;   0;  1;  1;  1;  0;  -1;  -1; -1])
  y=sym([0;  -1; -1;  0;  1;  1;   1;   0; -1])
case 13%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 14%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 15%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 SQUAREpi

case 16 %a 2-square [-1,1]x[-1,1]
 %       1    2   3   4   5   6    7   8    9
  x=sym([0;   0;  1;  1;  1;  0;  -1;  -1; -1])
  y=sym([0;  -1; -1;  0;  1;  1;   1;   0; -1])

case 17%isosceles triangle(one triangle in a square-required for torsion analysis)
x=sym([0;sqrt(2)/2; -sqrt(2)/2])
y=sym([0;sqrt(2)/2;sqrt(2)/2])
case 18%isoscles triangle(torsion of an equilateral triangle,each side=2*sqrt(3))
 x=sym([-sqrt(3);sqrt(3); 0])
 y=sym([     -1;      -1; 2])
 case 19%triangle inscribed in a circle of radius=1(torsion of an equilateral triangle
each side=sqrt(3))
 x=sym([-sqrt(3)/2;sqrt(3)/2; 0])
 y=sym([     -1/2;      -1/2; 1])
 case 20%square inscribed in a circle of radius=1,required for torsion analysis
%      1      2          3       4          5
x=sym([0;-sqrt(2)/2; sqrt(2)/2;sqrt(2)/2;-sqrt(2)/2])
y=sym([0;-sqrt(2)/2;-sqrt(2)/2;sqrt(2)/2; sqrt(2)/2])
case 21%regular pentagon inscribed in a circle of radius=1,required for torsion
analysis
 %      1      2    3       4           5            6
x=sym([0;cos(pi/10);0;-cos(pi/10);-cos(3*pi/10); cos(3*pi/10)])
y=sym([0;sin(pi/10);1; sin(pi/10);-sin(3*pi/10);-sin(3*pi/10)])
case 22%regular hexagon inscribed in a circle of radius=1,required for torsion analysis
    %      1 2   3          4        5     6           7
    x=sym([0;1;cos(pi/3);-cos(pi/3);-1; -cos(pi/3); cos(pi/3)])
    y=sym([0;0;sin(pi/3); sin(pi/3); 0; -sin(pi/3);-sin(pi/3)])
    case 23%regular heptagon inscribed in a circle of radius=1,required for torsion
analysis
 %     1     2            3              4            5       6       7         8
x=sym([0;-cos(5*pi/14); cos(5*pi/14); cos(pi/14);cos(3*pi/14);0;-cos(3*pi/14);-
cos(pi/14)])
y=sym([0;-sin(5*pi/14);-sin(5*pi/14);-sin(pi/14);sin(3*pi/14);1; sin(3*pi/14);-
sin(pi/14)])
    case 24 %regular octagon inscribed in a circle of radius=1,required for torsion
analysis
 %     1 2     3     4     5       6       7     8     9
x=sym([0;1;cos(pi/4);0;-cos(pi/4);-1;-cos(pi/4); 0; cos(pi/4)])
y=sym([0;0;sin(pi/4);1; sin(pi/4); 0;-sin(pi/4);-1;-sin(pi/4)])
    case 25%(9-gon)nonagon
 %       1     2             3      4          5         6            7             8
9              10
  x=sym([0;cos(pi/18); cos(5*pi/18);0;-cos(5*pi/18);-cos(pi/18);-cos(3*pi/18);-
cos(7*pi/18); cos(7*pi/18); cos(3*pi/18)])
  y=sym([0;sin(pi/18); sin(5*pi/18);1; sin(5*pi/18); sin(pi/18);-sin(3*pi/18);-
sin(7*pi/18);-sin(7*pi/18);-sin(3*pi/18)])
    case 26%(10-gon)decagon
 %       1 2     3          4          5              6        7     8           9
10       11
  x=sym([0;1;cos(pi/5); cos(2*pi/5);-cos(2*pi/5);-cos(pi/5);-1;-cos(pi/5);-cos(2*pi/5);
cos(2*pi/5); cos(pi/5)])
  y=sym([0;0;sin(pi/5); sin(2*pi/5); sin(2*pi/5); sin(pi/5); 0;-sin(pi/5);-
sin(2*pi/5);-sin(2*pi/5);-sin(pi/5)])
    case 27%(11-gon)hendecagon
  %      1        2          3       4          5           6              7        8
9              10                11       12
  x=sym([0;cos(3*pi/22);cos(7*pi/22);0;-cos(7*pi/22);-cos(3*pi/22);-cos(pi/22);-
cos(5*pi/22);-cos(9*pi/22); cos(9*pi/22); cos(5*pi/22); cos(pi/22)])
  y=sym([0;sin(3*pi/22);sin(7*pi/22);1; sin(7*pi/22); sin(3*pi/22);-sin(pi/22);-
sin(5*pi/22);-sin(9*pi/22);-sin(9*pi/22);-sin(5*pi/22);-sin(pi/22)])
  case 28%(12-gon)dodecagon
  %      1 2   3           4     5      6          7      8       9         10    11
12      13
  x=sym([0;1;cos(pi/6);cos(pi/3);0;-cos(pi/3);-cos(pi/6);-1;-cos(pi/6);-cos(pi/3); 0;
cos(pi/3); cos(pi/6)])
  y=sym([0;0;sin(pi/6);sin(pi/3);1; sin(pi/3); sin(pi/6); 0;-sin(pi/6);-sin(pi/3);-1;-
sin(pi/3);-sin(pi/6)])
  case 29%(13-gon)tridecagon
  %      1       2         3           4        5      6                 7            8
9                 10            11              12          13           14
  x=sym([0;cos(pi/26);cos(5*pi/26);cos(9*pi/26);0;-cos(9*pi/26);-cos(5*pi/26);-
cos(pi/26);-cos(3*pi/26);-cos(7*pi/26);-cos(11*pi/26); cos(11*pi/26); cos(7*pi/26);
cos(3*pi/26)])
  y=sym([0;sin(pi/26);sin(5*pi/26);sin(9*pi/26);1; sin(9*pi/26); sin(5*pi/26);
sin(pi/26);-sin(3*pi/26);-sin(7*pi/26);-sin(11*pi/26);-sin(11*pi/26);-sin(7*pi/26);-
sin(3*pi/26)])
    case 30%(14-gon)tetradecagon
  %      1 2         3          4             5         6                  7
8       9        10            11              12          13           14            15
  x=sym([0;1;cos(2*pi/14);cos(4*pi/14);cos(6*pi/14);-cos(6*pi/14);-cos(4*pi/14);-
cos(2*pi/14);-1;-cos(2*pi/14);-cos(4*pi/14);-cos(6*pi/14); cos(6*pi/14); cos(4*pi/14);
cos(2*pi/14)])
  y=sym([0;0;sin(2*pi/14);sin(4*pi/14);sin(6*pi/14); sin(6*pi/14); sin(4*pi/14);
sin(2*pi/14); 0;-sin(2*pi/14);-sin(4*pi/14);-sin(6*pi/14);-sin(6*pi/14);-sin(4*pi/14);-
sin(2*pi/14)])
    case 31%(15-gon)pentadecagon
  %      1       2         3              4             5       6     7
8               9        10            11              12          13           14
15               16
   x=sym([0; cos(pi/30);cos(3*pi/30);cos(7*pi/30);cos(11*pi/30);0;-cos(11*pi/30);-
cos(7*pi/30);-cos(3*pi/30);-cos(pi/30);-cos(5*pi/30);-cos(9*pi/30);-cos(13*pi/30);
cos(13*pi/30); cos(9*pi/30); cos(5*pi/30)])
   y=sym([0;-sin(pi/30);sin(3*pi/30);sin(7*pi/30);sin(11*pi/30);1; sin(11*pi/30);
sin(7*pi/30); sin(3*pi/30);-sin(pi/30);-sin(5*pi/30);-sin(9*pi/30);-sin(13*pi/30);-
sin(13*pi/30);-sin(9*pi/30);-sin(5*pi/30)])
    case 32%(16-gon)hexadecagon
    %  1 2     3       4           5       6      7              8         9     10
11           12          13    14       15           16        17
   x=sym([0;1;cos(pi/8);cos(pi/4);cos(3*pi/8);0;-cos(3*pi/8);-cos(pi/4);-cos(pi/8);-1;-
cos(pi/8);-cos(pi/4);-cos(3*pi/8); 0; cos(3*pi/8); cos(pi/4); cos(pi/8)]);
   y=sym([0;0;sin(pi/8);sin(pi/4);sin(3*pi/8);1; sin(3*pi/8); sin(pi/4); sin(pi/8); 0;-
sin(pi/8);-sin(pi/4);-sin(3*pi/8);-1;-sin(3*pi/8);-sin(pi/4);-sin(pi/8)]);
    case 33%(17-gon)heptadecogon
   %   1          2           3       4              5       6          7
8           9              10           11          12               13              14
15             16              17       18
   x=sym([0;cos(pi/34);cos(5*pi/34);cos(9*pi/34);cos(13*pi/34);0;-cos(13*pi/34);-
cos(9*pi/34);-cos(5*pi/34);-cos(pi/34);-cos(3*pi/34);-cos(7*pi/34);-cos(11*pi/34);-
cos(15*pi/34); cos(15*pi/34); cos(11*pi/34); cos(7*pi/34); cos(3*pi/34)]);
   y=sym([0;sin(pi/34);sin(5*pi/34);sin(9*pi/34);sin(13*pi/34);1; sin(13*pi/34);
sin(9*pi/34); sin(5*pi/34); sin(pi/34);-sin(3*pi/34);-sin(7*pi/34);-sin(11*pi/34);-
sin(15*pi/34);-sin(15*pi/34);-sin(11*pi/34);-sin(7*pi/34);-sin(3*pi/34)]);
```

```
 case 34%(18-gon)octadecogon
  %  1 2           3       4              5              6          7              8
9             10        11         12           13              14            15
16              17       18              19
 x=sym([0;1;cos(4*pi/36);cos(8*pi/36);cos(12*pi/36);cos(16*pi/36);-cos(16*pi/36);-
cos(12*pi/36);-cos(8*pi/36);-cos(4*pi/36);-1;-cos(4*pi/36);-cos(8*pi/36);-
cos(12*pi/36);-cos(16*pi/36); cos(16*pi/36); cos(12*pi/36); cos(8*pi/36);
cos(4*pi/36)]);
 y=sym([0;0;sin(4*pi/36);sin(8*pi/36);sin(12*pi/36);sin(16*pi/36); sin(16*pi/36);
sin(12*pi/36); sin(8*pi/36); sin(4*pi/36); 0;-sin(4*pi/36);-sin(8*pi/36);-
sin(12*pi/36);-sin(16*pi/36);-sin(16*pi/36);-sin(12*pi/36);-sin(8*pi/36);-
sin(4*pi/36)]);
case 35%(19-gon)enneadecogon
  %  1        2           3           4             5        6      7              8
9             10           11         12          13             14            15
16                  17       18              19             20
 x=sym([0;cos(3*pi/38);cos(7*pi/38);cos(11*pi/38);cos(15*pi/38);0;-cos(15*pi/38);-
cos(11*pi/38);-cos(7*pi/38);-cos(3*pi/38);-cos(pi/38);-cos(5*pi/38);-cos(9*pi/38);-
cos(13*pi/38);-cos(17*pi/38); cos(17*pi/38); cos(13*pi/38); cos(9*pi/38); cos(5*pi/38);
cos(pi/38)]);
 y=sym([0;sin(3*pi/38);sin(7*pi/38);sin(11*pi/38);sin(15*pi/38);1; sin(15*pi/38);
sin(11*pi/38); sin(7*pi/38); sin(3*pi/38);-sin(pi/38);-sin(5*pi/38);-sin(9*pi/38);-
sin(13*pi/38);-sin(17*pi/38);-sin(17*pi/38);-sin(13*pi/38);-sin(9*pi/38);-
sin(5*pi/38);-sin(pi/38)]);
  case 36%(20-gon)icosagon
  % 1 2     3           4             5           6      7           8             9
10          11   12      13           14            15            16      17       18
19             20      21
x=sym([0;1;cos(pi/10);cos(2*pi/10);cos(3*pi/10);cos(4*pi/10);0;-cos(4*pi/10);-
cos(3*pi/10);-cos(2*pi/10);-cos(pi/10);-1;-cos(pi/10);-cos(2*pi/10);-cos(3*pi/10);-
cos(4*pi/10); 0; cos(4*pi/10); cos(3*pi/10); cos(2*pi/10); cos(pi/10)]);
y=sym([0;0;sin(pi/10);sin(2*pi/10);sin(3*pi/10);sin(4*pi/10);1; sin(4*pi/10);
sin(3*pi/10); sin(2*pi/10); sin(pi/10); 0;-sin(pi/10);-sin(2*pi/10);-sin(3*pi/10);-
sin(4*pi/10);-1;-sin(4*pi/10);-sin(3*pi/10);-sin(2*pi/10);-sin(pi/10)]);


end
if (nmax>3)
[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial(n1,n2,n
3,nmax,numtri,n);
end
if (nmax==3)
    [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals(n);
end
[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);
ss1='number of 6-node triangles with centroid=';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
%
eln
%
ss2='number of special convex quadrilaterals elements&nodes per element =';
[nel,nnel]=size(spqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
spqd
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])



nitri=nmax-1;
```

```
if (nmax==3)nitri=1
end
for itri=1:nitri
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
x1=x(n1(itri,1),1)
x2=x(n2(itri,1),1)
x3=x(n3(itri,1),1)
%
y1=y(n1(itri,1),1)
y2=y(n2(itri,1),1)
y3=y(n3(itri,1),1)
rrr(:,:,itri)
U'
V'
W'
kk=0;
for ii=1:n+1
    for jj=1:(n+1)-(ii-1)
        kk=kk+1;
        mm=rrr(ii,jj,itri);
        uu=U(kk,1);vv=V(kk,1);ww=W(kk,1);
        xi(mm,1)=x1*ww+x2*uu+x3*vv;
        yi(mm,1)=y1*ww+y2*uu+y3*vv;
    end
end
[xi yi]
%add coordinates of centroid
 ne=(n/2)^2;
% stdnode=kk;
 for iii=1+(itri-1)*ne:ne*itri
     %kk=kk+1;
     node1=eln(iii,1)
     node2=eln(iii,2)
     node3=eln(iii,3)
     mm=eln(iii,7)
     xi(mm,1)=(xi(node1,1)+xi(node2,1)+xi(node3,1))/3;
     yi(mm,1)=(yi(node1,1)+yi(node2,1)+yi(node3,1))/3;
 end

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

The remaining programs(5),(6),(7) are already listed in our earlier works published in this journal