www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 10 October 2017, Page No. 22632-22787
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i10.13

Finite element solution of Poisson Equation over Polygonal Domains using a novel auto mesh generation technique and an explicit integration scheme for eight node linear convex quadrilaterals

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

[a]Department of Mathematics, Central College Campus, Bangalore University, Bangalore -560001, Karnataka State, India.
[b] Xavier Institute of Management and Entrepreneurship, Hosur Road, Electronic City Phase II, Bangalore, Karnataka 560034, Karnataka State, India.
[c]Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post, Kolar Gold Field, Kolar District, 563120 , Karnataka State, India.
[d]Department of Mathematics,Rajarajeswari College of Engineering, Bangalore -560074, Karnataka State, India.

Abstract :
This paper presents an explicit integration scheme to compute the stiffness matrix of an eight node linear convex quadrilateral element for plane problems using symbolic mathematics and a novel auto mesh generation technique , In finite element analysis, the boundary value problems governed by second order linear partial differential equations, the element stiffness matrices are expressed as integrals of the product of global derivatives over the linear convex quadrilateral region. These matrices can be shown to depend on the material properties and the matrix of integrals with integrands as rational functions with polynomial numerator and the linear denominator $(4+\xi+\eta)$ in the bivariates ξ and η over an eight node 2-square $(-1\leq \xi,\eta \leq 1)$. In this paper, we have computed these integrals in exact forms using the 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 a novel auto mesh generation technique of all quadrilaterals for a convex polygonal domain which provides the nodal coordinates and element connectivity. We have used the explicit integration scheme and this novel auto mesh generation technique to solve the Poisson equation with given Dirichlet boundary conditions over convex polygonal domains.

Key words: Explicit Integration, Finite Element Method, Matlab Symbolic Mathematics, All Quadrilateral Mesh Generation Technique,Poisson Equation,Dirichlet Boundary Conditions ,Polygonal Domain, Gauss Legendre Quadrature Rules

1. Introduction :

In recent years, the finite element method (FEM) has emerged as a powerful tool for the approximate solution of differential equations governing diverse physical phenomena. Today, finite element analysis is an integral and major component in many fields of engineering design and manufacturing. Its use in industry and research is extensive, and indeed without it many practical problem in science, engineering and emerging technologies such as nanotechnology, biotechnology, aerospace, chemical. etc would be incapable of solution [1,2,3]. In FEM, various integrals are to be determined numerically in the evaluation of stiffness matrix, mass matrix, body force vector, etc. The algebraic integration needed to derive explicit finite element relations for second order continuum mechanics problems generally defies our analytic skill and in most cases, it appears to be a prohibitive task. Hence, from a practical point of view, numerical integration scheme is not only necessary but very important as well. Among various numerical integration schemes, Gauss Legendre quadrature, which can evaluate exactly the $(2n-1)^{th}$ degree polynomial with 'n' Gaussian

integration points, is mostly used in view of the accuracy and efficiency of calculation. However, the integrands of global derivative products in stiffness matrix computations of practical applications are not always simple polynomials but rational expressions which the Gaussian quadrature cannot evaluate exactly [7-15]. The integration points have to be increased in order improve the integration accuracy but it is also desirable to make these evaluations by using as few Gaussian points as possible, from the point of view of the computational efficiency. Thus it is an important task to strike a proper balance between accuracy and economy in computation. Therefore analytical integration is essential to generate a smaller error as well as to save the computational costs of Gaussian quadrature commonly applied for science, engineering and technical problems. In explicit integration of stiffness matrix, complications arise from two main sources, firstly the large number of integrations that need to be performed and secondly, in methods which use isoparametric or equivalently the subparametric finite elements, the presence of determinant of the Jacobian matrix (we refer this as Jacobian here after) in the denominator of the element matrix integrands. This problem is considered in the recent work [16] for the four node linear convex quadrilateral which proposes a new discretisation method and use of pre computed universal numeric arrays which do not depend on element size and shape. In this method a linear polygon is discretized into a set of linear triangles and then each of these triangles is further discretised into three linear four node convex quadrilateral elements by joining the centroid to the mid-point of sides. These quadrilateral elements are then mapped into 2-squares ($-1\leq \xi,\eta \leq 1$) in the natural space (ξ,η) to obtain the same expression of the Jacobian, namely $c(4+\xi+\eta)$ where c is some appropriate constant which depends on the geometric data for the triangle.
Many important problems in engineering,science and applied mathematics are formulated by appropriate differential equations with some boundary conditions imposed on the desired unknown function or the set of functions. There exists a large literature which demonstrates numerical accuracy of the finite element method to deal with such issues [1]. Clough seems to be the first who introduced the fnite elements to standard computational procedures [2]. A further historical development and present day concepts of finite element analysis are widely described in references [1, 3]. In this paper the well-known Laplace and Poisson equations will be examined by means of the finite element method applied to an appropriate 'mesh'. The class of physical situations in which we meet these equations is really broad. Let's recall such problems like heat conduction, seepage through porous media, irrotational flow of ideal fluids, distribution of electrical or magnetic potential, torsion of prismatic shafts, lubrication of pad bearings and others [4]. Therefore, in physics and engineering arises a need of some computational methods that allow to solve accurately such a large variety of physical situations.The considered method completes the above-mentioned task. Particularly, it refers to a standard discrete pattern allowing to find an approximate solution to continuum problem. At the beginning, the continuum domain is discretized by dividing it into a finite number of elements which properties must be determined from an analysis of the physical problem (e. g. as a result of experiments). These studies on particular problem allow to construct so{called the stiffness matrix for each element that, for instance, in elasticity comprising material properties like stress strain relationships [2, 5]. Then the corresponding nodal loads associated with elements must be found. The construction of accurate elements constitutes the subject of a mesh generation recipe proposed by the author within the presented article. In many realistic situations, mesh generation is a time consuming and error prone process because of various levels of geometrical complexity. Over the years, there were developed both semi automatic and fully automatic mesh generators obtained, respectively, by using the mapping methods or, on the contrary, algorithms based on the Delaunay triangulation method [6], the advancing front method [7] and tree methods [8]. It is worth mentioning that the first attempt to create fully automatic mesh generator capable to produce valid finite element meshes over arbitrary domains has been made by Zienkiewicz and Phillips [9].
In the present paper, we propose a similar discretisation method for linear polygon in Cartesian two space (x,y). This discrtisation is carried in two steps, We first discretise the linear polygon into a set of linear triangles in the Cartesian space (x,y) and these linear triangles are then mapped into a standard triangle in a local space (u,v). We further discretise the standard triangles into three linear quadrilaterals by joining the centroid to the midpoints of triangles in (u,v) space which are finally mapped into 2-square in the local (ξ,η) space. We then establish a derivative product relation between the linear convex quadrilaterals in the Cartesian space, (x,y) which are interior to an arbitrary triangle and the linear quadrilaterals in the local

space (u,v) interior to the standard triangle. In this procedure, all computations in the local space (u,v) for product of global derivative integrals are free from geometric properties and hence they are pure numbers. We then propose a numerical scheme to integrate the products of global derivatives. We have shown that the matrix product of global derivative integrals is expressible as matrix triple product comprising of geometric properties matrices and the product of local derivative integrals matrix. We have obtained explicit integration of the product of local derivatives which is now possible by use of symbolic integration commands available in leading mathematical softwares MATLAB, MAPLE, MATHEMATIKA etc. In this paper, we have used the MATLAB symbolic mathematics to compute the integrals of the products of local derivatives in (u, v) space .The proposed explicit integration scheme is shown as a useful technique in the formation of element stiffness matrices for second order boundary problems governed by partial differential equations.

2. Poisson Equation

2.1 Statement of the Problem

The Poisson equation

$$-\nabla^2 u= f \qquad \text{...(1)}$$

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 $\partial\Omega\ of\ \Omega$;for example

$$\alpha u + \beta \frac{\partial u}{\partial n} = g \quad \text{on} \quad \partial\Omega \qquad \text{...(2)}$$

where $\partial u /\partial n$ 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 $\partial\Omega_D$ and Neumann conditions(or indeed mixed conditions where α and β are both nonzero) hold on remainder $\partial\Omega\backslash\partial\Omega_D$. 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

$$-\int_{\partial\Omega}\frac{\partial u}{\partial n} = --\int_{\Omega}\nabla^2 u = \int_{\Omega} f \qquad \text{..(3)}$$

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:

$$\int_{\partial\Omega} g + \int_{\Omega} f = 0 \qquad \text{---(4)}$$

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,

$$\int_{\Omega}(\nabla^2 u + f)\, v = \text{...(5)}$$

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

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

so that

$$\int_\Omega \nabla u \,.\nabla v = \int_\Omega vf + \int_{\partial\Omega} v\,\frac{\partial u}{\partial n}$$
....(6a)

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(6)is equivalent to eqns (1) and (2) 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

$$\int_\Omega \nabla u \,.\nabla v = \int_\Omega vf + \int_{\partial\Omega} vg$$
....(6b)

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:

$-\Delta u(x)=f(x)$ for all $x\in \Omega$
............................(7a)

$u = g(x)$ on $\partial\Omega$
...............................(7b)

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

$$\int_\Omega \nabla u \,.\nabla v\, d\,x = \int_\Omega vf\, dx$$
..............................(7c)

$u = g(x)$ on $\partial\Omega$
...............................(7d)

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

$$\int_\Omega \nabla u_h \,.\nabla v\, d\,x = \int_\Omega vf\, dx \qquad \text{for all} \qquad v \in V_h$$
................................(8)

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 $u_h = \sum_{j=1}^{j=N}\alpha_j\varphi_j$. Substituting this in eqn(8) and choosing v to be a basis function φ_i gives the following set of equations

$\sum_{j=1}^{N} \alpha_j \int_{\Omega} \nabla\varphi_i . \nabla\varphi_j \, d\,x = \int_{\Omega} f\varphi_i \, dx$,i=1,2,3,...., N
..................(9)
This is really a linear system of the form
$Ku=f$
..................(10)

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

$$K_{i,j} = \int_{\Omega} \nabla\varphi_i . \nabla\varphi_j \, d\,x$$
.....................(11a)

$$f_i = \int_{\Omega} f\varphi_i \, dx$$
.......................(11b)

and K is called stiffness matrix because the linear system looks like Hookes law if f represents forces and u represents displacemewnts.

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(11) in section (2.3.2)

$$K_{i,j} = \int_{\Omega} \nabla\varphi_i . \nabla\varphi_j \, d\,x$$
....(11a)

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 φ_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(11) as

$$K_{i,j} = \sum_{\Omega^e \; \varepsilon\Omega_h^e} K_{i,j}^e \quad =$$
....(12a)
where

$K_{i,j}^{e}$ $=\int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ ....(12b)

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

$K_{i,j}^{e} = \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ look like the entries $K_{i,j} = \int_{\Omega} \nabla\varphi_i . \nabla\varphi_j \, dx = \sum_{\Omega^e \varepsilon \Omega_h^e} \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ 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

$f_i = \int_{\Omega} f\varphi_i \, dx = \sum_{e=1}^{N_e} f_i^e$ ....(12c)

where

f_i^e = $\int_{\Omega^e} f\varphi_i d\,\mathrm{x}$ ....(12d)

Now further assume that on an element Ω^e , $u_h = \mathrm{u^e} = \sum_{\mathrm{j=1}}^{\mathrm{j=d}} \mathrm{u_j^e} \varphi_\mathrm{j}$

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

$$\sum_{e=1}^{N_e} K^e \, \mathrm{u^e} = \sum_{e=1}^{N_e} f^e$$ ....(12e)

Where

$\mathrm{u^e} = (\mathrm{u_1^e}, \mathrm{u_2^e}, \mathrm{u_3^e}, \ldots\ldots\ldots\ldots \mathrm{u_d^e})^T$, $f^\mathrm{e} = (f_1^\mathrm{e}, f_2^\mathrm{e}, f_3^\mathrm{e}, \ldots\ldots\ldots\ldots f_\mathrm{d}^\mathrm{e})^T$ ....(12f)

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 $K_{i,j}^e$ and f_i^e

In order to compute the local/element stiffness matrices,we need to compute the integrals $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ .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 $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ 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 $K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ is given by

$K_{i,j}^e = \int_{\Omega^e} \nabla\varphi_i . \nabla\varphi_j \, dx$ $=\int_{\Omega^e} \{ \frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y} \} dxdy = \sum_{e=1}^{N_e} \sum_{n=0}^{2} \int_{Q_E} \{ \frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y} \} dxdy$

$=\sum_{e=1}^{N_e} \sum_{n=0}^{2} S_{i,j}^E$ ....(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)

$f_i^e = \int_{\Omega^e} f\varphi_i$ 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

2.4 Finite Element Types

2.4.1 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.

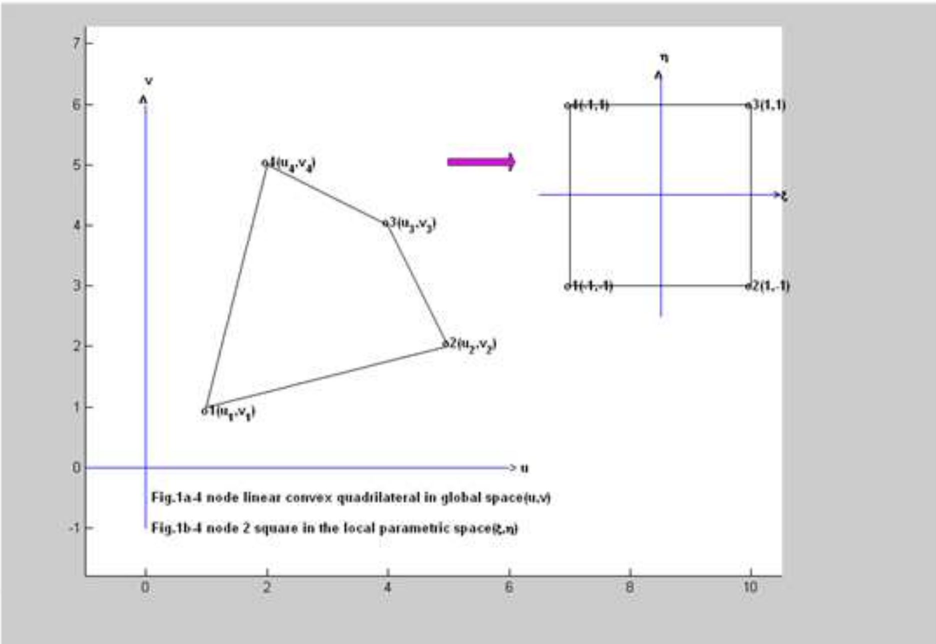


Fig.1a-4 node linear convex quadrilateral in global space(u,v)
Fig.1b-4 node 2 square in the local parametric space(ξ,η)

$$\begin{pmatrix} u \\ v \end{pmatrix} = \sum_{k=1}^{4} \begin{pmatrix} u_k \\ v_k \end{pmatrix} M_k(\xi, \eta) \qquad \text{------------------------ (13)}$$

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

$$M_k(\xi, \eta) = \frac{1}{4}(1 + \xi\xi_k)(1 + \eta\eta_k) \ , \quad k = 1, 2,3,4 \qquad \text{----------------------- (14a)}$$

Where $\{ (\xi_k, \eta_k), k = 1,2,3,4\}$ = {(-1,-1),(1,-1),(1,1),(-1,1)} ----------------------- (14b)

describes a geometric transformation over a linear convex quadrilateral element from the original global space into the local parametric space.

2.4.2 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

$$\begin{pmatrix} \phi \\ \psi \end{pmatrix} = \sum_{k=1}^{4} \begin{pmatrix} \phi_k \\ \psi_k \end{pmatrix} N_k{}^e(\xi, \eta) \quad \text{------------------------ (15)}$$

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.(14a-b)

We have considered the application of explicit stiffness matrix integration scheme and automesh generation technique to find FEM solution of Poisson equation boundary value problems over polygonal domains using linear convex quadrilateral elements under isoparametric transformations[ ].

2.4.3 Subparmetric Transformation :

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

$$\begin{pmatrix} \phi \\ \psi \end{pmatrix} = \sum_{k=1}^{nde} \begin{pmatrix} \phi_k{}^e \\ \psi_k{}^e \end{pmatrix} N_k{}^e(\xi, \eta) \quad \text{------------------------ (16)}$$

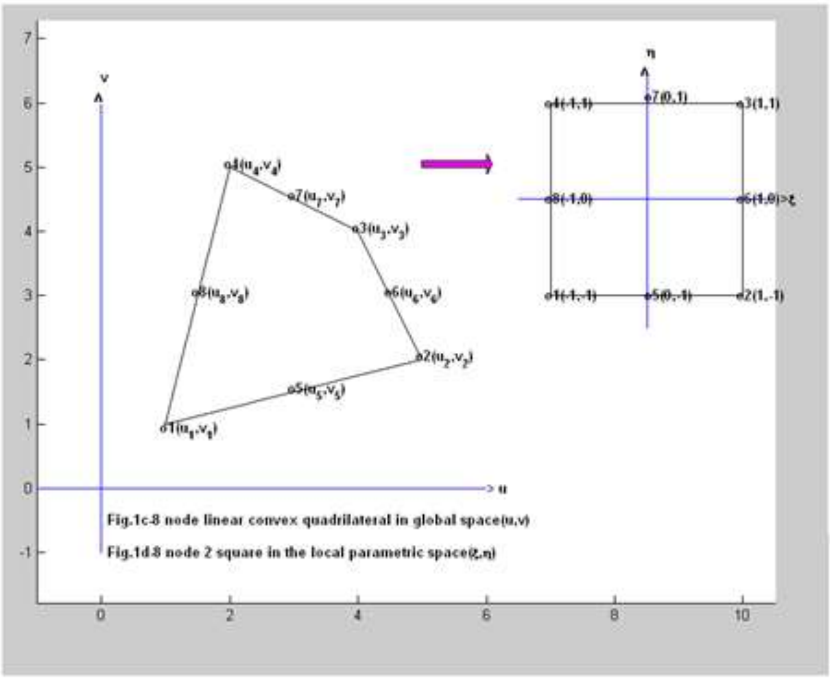
Where ϕ_k , ψ_k refer to unknowns at node k and nde >4

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 for which we set nde=4

In the present paper, we consider the subparametric transformation for a linear convex quadrilateral element for which nde = 8 , a eight noded (serendipity type 2 square)

3. Eight Node Linear Convex Quadrilateral Element :

In this section, we give a brief description of the 8- node quadrilateral element under subparametric transformation as shown in Fig 1c , Fig 1d.



We use the transform of Eqns.(13-14) to define the element geometry i.e.

$\begin{pmatrix} u(\xi,\eta) \\ v(\xi,\eta) \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} = \sum_{k=1}^{4} \begin{pmatrix} u_k \\ v_k \end{pmatrix} M_k(\xi,\eta)$ -------------------------- (13)

Where $M_k(\xi,\eta) = \frac{1}{4}\,(1+\xi\xi_k)(1+\eta\eta_k)$, (k = 1, 2,3,4) ----------------------- (14a)

With $(u(\xi_k,\eta_k), v(\xi_k,\eta_k))$, $k = 1, 2,3,4$) are the vertices of the linear convex quadrilateral in global (u, v) space .

$\{(\xi_k,\eta_k), k = 1,2,3,4\}$ = {(-1,-1),(1,-1),(1,1),(-1,1)} ----------------------- (14b)

Using the transformation of Eqns.(13-14) and from Fig 1c , Fig 1d we see that there is a one to one correspondence between $((\xi_k,\eta_k),\ k = 5,6,7,8) = ((0,1),(1,0),(0,1),(-1,0))$

and $((u_k,v_k) = \big(u(\xi_k,\eta_k), v(\xi_k,\eta_k)\big)$, $k = 5,6,7,8)$, where

$(u_5,\ v_5)$ = ($(u_1+u_2)/2$, $(v_1+v_2)/2$)

$(u_6,\ v_6)$ = ($(u_2+u_3)/2$, $(v_2+v_3)/2$)

$(u_7,\ v_7)$ = ($(u_3+u_4)/2$, $(v_3+v_4)/2$)

$(u_8,\ v_8)$ = ($(u_1+u_4)/2$, $(v_1+v_4)/2$) -------------------------(14c)

We then define the variation of physical variables $\phi^e,\ \psi^e$ (say) over 8- node element of Fig 1c , 1d by Eqn.(16) with nde = 8

$\begin{pmatrix} \phi^e \\ \psi^e \end{pmatrix} = \sum_{k=1}^{8} N_k{}^e(\xi,\eta)\ \begin{pmatrix} \phi_k{}^e \\ \psi_k{}^e \end{pmatrix}$ ------------------------ (16)

Where $\phi_k{}^e$, $\psi_k{}^e$ are the nodal values at node k

The shape functions $N_i{}^e$ of the 8- node element shown in Fig 1c , Fig 1d are given by

$N_i{}^e\,(\xi,\eta) = \frac{1}{4}\,(1+\xi\xi_k)(1+\eta\eta_k)(-1+\xi\xi_k+\eta\eta_k)$, $i = 1, 2,3,4$

$N_i{}^e\,(\xi,\eta) = \frac{1}{2}\,(1-\xi^2)(1+\eta\eta_k)$, $i = 5, 7$

$N_i{}^e\,(\xi,\eta) = \frac{1}{2}\,(1+\xi\xi_k)(1-\eta^2)$, $i = 6, 8$ -------------------------(17a)

and

$\{(\xi_k,\eta_k), k = 1(1)8\}$ = {(-1,-1), (1,-1), (1,1), (-1,1), (0, -1), (1, 0), (0, 1), (-1, 0) } ----------------------(17b)

4. Explicit Form of the Jacobian and Global Derivatives :

4.1 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.(1) and Eq.(2), we have

$$\frac{\partial u}{\partial \xi} = \sum_{k=1}^{4} u_k \frac{\partial M_k}{\partial \xi} = \frac{1}{4}[\,(-u_1 + u_2 + u_3 - u_4) + (u_1 - u_2 + u_3 - u_4)\,\eta\,] \quad \text{------------ (18a)}$$

$$\frac{\partial u}{\partial \eta} = \sum_{k=1}^{4} u_k \frac{\partial M_k}{\partial \eta} = \frac{1}{4}[\,(-u_1 - u_2 + u_3 + u_4) + (u_1 - u_2 + u_3 - u_4)\,\xi\,] \quad \text{------------ (18b)}$$

$$\frac{\partial v}{\partial \xi} = \frac{1}{4}[\,(-v_1 + v_2 + v_3 - v_4) + (v_1 - v_2 + v_3 - v_4)\,\eta\,] \quad \text{------------ (18c)}$$

$$\frac{\partial v}{\partial \eta} = \frac{1}{4}[\,(-v_1 - v_2 + v_3 + v_4) + (v_1 - v_2 + v_3 - v_4)\,\xi\,] \quad \text{------------ (18d)}$$

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

$$J = \frac{\partial(u,v)}{\partial(\xi,\eta)} = \frac{\partial u}{\partial \xi}\frac{\partial v}{\partial \eta} - \frac{\partial u}{\partial \eta}\frac{\partial v}{\partial \xi} = \alpha + \beta\xi + \gamma\,\eta \quad \text{-------------- (19a)}$$

Where

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

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

$$\gamma = \frac{1}{8}[\,(u_4 - u_1)(v_2 - v_3) + (u_3 - u_2)(v_4 - v_1)\,] \quad \text{---------------- (19b)}$$

4.2 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]

$$\begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix} = \frac{1}{J}\begin{bmatrix} \frac{\partial v}{\partial \eta} & -\frac{\partial v}{\partial \xi} \\ -\frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \xi} \end{bmatrix}\begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} \\ \frac{\partial N_i^e}{\partial \eta} \end{bmatrix} \quad \text{--------------------------- (20)}$$

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.(18a)–(18d) while J is defined in Eq.(19a-b) , $(\,i, j = 1,2,3, - - - - -,$ nde) , nde = the number of nodes per element. We may recall that the explicit integration for linear convex quadrilateral with nde = 4 is already presented by the authors in their recent paper [18]. We take nde = 8 for the present study.

5. 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, let a, b, c, d, e, f, g, h, I be the midpoints of sides ZS, ZT, ZU, PU, PS, QS, QT, RT, RU and let Z be the centroid of the triangle Δ PQR. We can obtain three linear convex 8- node quadrilaterals (Q_e, e=1,2,3),where Q_1 =ZcUdPeS , Q_2 =ZaSfQgT and Q_3= ZbThRiU from triangle Δ PQR as shown in Fig 2a,b. If we map each of these 8- node linear convex quadrilaterals into

8- node 2- squares in which the nodes are oriented in counter clockwise from Z, then the Jacobian J^e for each element Q_e, (e=1,2,3) is given by

$$J = J^e = \frac{1}{48}\Delta\, pqr\,(4+\xi+\eta), \qquad e = 1,2,3 \qquad \text{---------------} (21)$$

Where Δ pqr is the area of the triangle Δ PQR

$$2\Delta\, pqr = \begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix} = \left[\, (x_p - x_r)(y_q - y_r) - (x_q - x_r)(\,y_p - y_r)\,\right] \qquad \text{------------} (22)$$

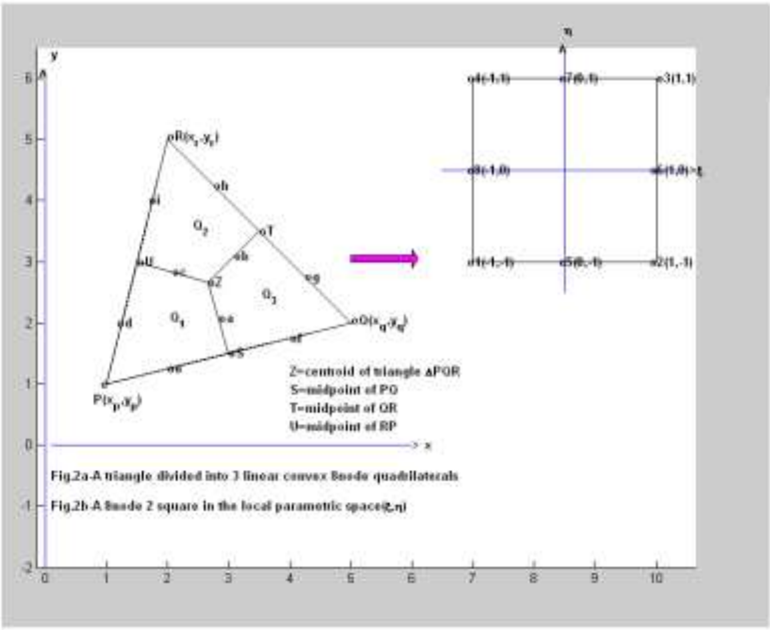


Fig.2a-A triangle divided into 3 linear convex 8node quadrilaterals

Fig.2b-A 8node 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 the arbitrary linear 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 Further, let a, b, c, d, e, f, g, h, I be the midpoints of sides ZS, ZT, ZU, PU, PS, QS, QT, RT, RU and Z be the centroid of the Δ PQR . Then we obtain three linear convex 8- node quadrilaterals Q_e (e=1,2,3),Q_1 =<ZcUdPeSa> , Q_2 =<ZaSfQgTb > and Q_3= <ZbThRiUc> , these quadrilaterals can mapped into the linear conv ex 8- node quadrilateral spanning the vertices GHEICJF with G(1/3, 1/3), H(1/6, 5/12), E(0, ½), I(0, ¼), C(0, 0), J(1/4, 0), F(1/2, 0), K(5/12, 1/6) in the interior of the right isosceles triangle Δ ABC with vertices A(1, 0), B(0, 1) and C(0, 0) in the (u, v) space as shown in Fig3a and Fig3b.

Proof : The sum of the three quadrilaterals Q_1, Q_2, Q_3 is Q_1+ Q_2+ Q_3 = Δ PQR as shown in Fig 2a & Fig 3a.

We know that the linear transformations

$$\begin{pmatrix} x^{(1)} \\ y^{(1)} \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} w + \begin{pmatrix} x_q \\ y_q \end{pmatrix} u + \begin{pmatrix} x_r \\ y_r \end{pmatrix} v \qquad \text{-------------------------} (23)$$

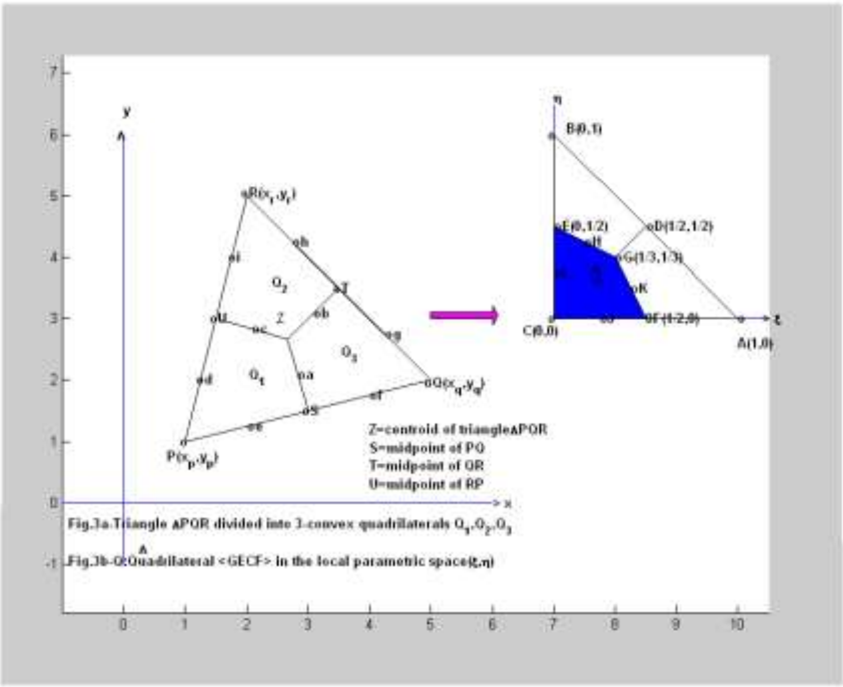
$$\begin{pmatrix} x^{(2)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} x_q \\ y_q \end{pmatrix} w + \begin{pmatrix} x_r \\ y_r \end{pmatrix} u + \begin{pmatrix} x_p \\ y_p \end{pmatrix} v \qquad \text{--------------------------- (24)}$$

$$\begin{pmatrix} x^{(3)} \\ y^{(3)} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} w + \begin{pmatrix} x_p \\ y_p \end{pmatrix} u + \begin{pmatrix} x_q \\ y_q \end{pmatrix} v \qquad \text{------------------------- (25)}$$

with $w = 1 - u - v$

map the arbitrary triangle Δ PQR into a linear right isosceles triangle A(1, 0) , B(0, 1) and C(0, 0) in the uv– plane. We can now verify that the vertices Z, c, U, d, P, e, S in xy palne is mapped into the linear convex 8- node quadrilateral spanning the vertices G, H, E, I, C, J, F, K by use of the transformation given in Eqn.(23).

Similarly, we see that the linear convex 8- node quadrilateral Q_2 spanned by vertices Z, a, S, f, Q, g, T, b is mapped into the linear convex 8- node quadrilateral spanned by the vertices G, H, E, I, C, J, F, K by use of the transformation of Eqn.(24). Finally the quadrilateral Q_3 in xy plane is mapped into the quadrilateral GHEICJFK in uv- plane by use of the linear transformation of Eqn.(25),
This completes the proof.



We have shown in the foregoing Lemma that an arbitrary linear triangle can be discretised into three linear convex 8- node quadrilaterals. Further, each of these quadrilaterals in xy plane can be mapped into a unique linear convex 8- node quadrilateral spanned by the vertices G(1/3, 1/3), H(1/6, 5/12), E(0, ½), I(0, ¼), C(0, 0), J(1/4, 0), F(1/2, 0) and K(5/12, 1/6) (see Fig 3a, Fig 3b) using a proper linear transformation as given Eqn.(23) – (25).

6. Integration over a Triangular Region :

6.1 Composite Integration

We shall now establish a composite integration formula for an arbitrary triangular region Δ PQR shown in Fig 2a or Fig 3a. Let $\phi(x, y)$ be an arbitrary and smooth function defined over the region Δ PQR . We now consider

$$II_{\Delta PQR} = \iint_{\Delta PQR} \phi(x,y)\, dxdy = \sum_{e=1}^{3} \iint_{Q_e} \phi(x,y)\, dxdy \qquad \text{---------------------} (26)$$

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

$$= (2\,\Delta_{pqr}) \iint_{\hat{Q}} \{\sum_{e=1}^{3} [\phi\,(x^{(e)}(u,v),\ y^{(e)}(u,v))\,]\}\, dudv \qquad \text{--------------------} (27)$$

Where $(x^{(e)}(u,v),\ y^{(e)}(u,v)), e = 1,2,3)$ are the linear transformations of Eqs.(23)–(25) and $\hat{Q}$ is the linear convex 8- node quadrilateral GHEICJFK spanning the vertices G(1/3, 1/3), H(1/6, 5/12), E(0, ½), I(0, ¼), C(0, 0), J(1/4, 0), F(1/2, 0) and K(5/12, 1/6) and Δ_{pqr} is the area of triangle Δ PQR, Now, we further use the bilinear transformation of Eqns.(1)–(2) in Eqn.(15) and obtain.

$$II_{\Delta PQR} = (2\,\Delta_{pqr}) \int_{-1}^{1}\int_{-1}^{1} \{\sum_{e=1}^{3} [\phi\,(x^{(e)}(u,v),\ y^{(e)}(u,v)) \frac{\partial(u,v)}{\partial(\xi,\eta)}\}\, d\xi\, d\eta \qquad \text{---------------} (28)$$

In Eq.(16) we have used the bilinear transformation given in Eqns.(13)- (14)

$$u = u(\xi,\ \eta) = \frac{1}{3} M_1(\xi,\ \eta) + \frac{1}{2} M_4(\xi,\ \eta)$$

$$v = v(\xi,\ \eta) = \frac{1}{3} M_1(\xi,\ \eta) + \frac{1}{2} M_2(\xi,\ \eta) \qquad \text{----------------------} (29)$$

to map the arbitrary linear convex 8- noded quadrilateral into a 2 – square in (ξ,η) – plane. Thus on using Eqn.(29), the integral of Eqn.(28) simplifies to the following.

$$II_{\Delta PQR} = (2\,\Delta_{pqr}) \int_{-1}^{1}\int_{-1}^{1} [\sum_{e=1}^{3} \left(\frac{4+\xi+\eta}{96}\right) \phi(x^{(e)}(u,v),\ y^{(e)}(u,v))\,]d\xi\, d\eta \qquad \text{-------------} (30)$$

We can evaluate Eqn.(30) either analytically or numerically depending on the form of the integrand.

Using Numerical Integration , we have from Eqn.(30)

$$II_{\Delta PQR} = 2\Delta_{pqr} \sum_{i=1}^{N}\sum_{j=1}^{N} \left(\frac{W_i^{(N)} W_j^{(N)} (4+\xi_i^{(N)}+\eta_j^{(N)})}{96}\right) \sum_{e=1}^{3} \phi(x^{(e)}\left(u_{i,j}^{(N)}, v_{i,j}^{(N)}\right), y^{(e)}\left(u_{i,j}^{(N)}, v_{i,j}^{(N)}\right))$$

$$\text{-----------------------} (31)$$

Where from Eqn.(29), we write

$$u_{i,j}^{(N)} = u(\xi_i^{(N)},\ \eta_j^{(N)})$$

$$v_{i,j}^{(N)} = v(\xi_i^{(N)},\ \eta_j^{(N)}) \qquad \text{----------------------} (32)$$

and $(W_i^{(N)},\ \xi_i^{(N)})$, $(W_j^{(N)},\ \xi_j^{(N)})$ are the weight coefficients and sampling points along ξ,η directions of the N^{th} order Gauss Legendre quadrature rules. We could also use Gauss Labatto quadrature rules as well to evaluate the integral of Eqn.(18).

The above composite rule is applied to numerical Integration over polygonal domains using convex quadrangulation and Gauss Legendre Quadrature Rules[27].

In the next section 6.2, we shall apply the above derivations and compute the integral of eqn.(26) by assuming the integrand $\phi(x, y)$ as the product of global derivatives, which are not explicit function of global variates (x, y)

6.2 Global Derivative Integrals :

If $N_i^{(e)}$ (i =1(1)8) denotes the basis functions for node i of a linear convex 8- node linear convex quadrilateral element e , then by use of chain rule of partial differentiation

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

We note that to transform 8- node linear convex quadrilateral $Q_e(e = 1,2,3)$ of ΔPQR in Cartesian space (x,y) into $\widehat{Q}$, the 8- node linear convex quadrilateral spanned by vertices (1/3, 1/3), (1/6, 5/12), (0, ½), (0, ¼), (0, 0), (1/4, 0), (1/2, 0) and (5/12, 1/6) in uv-plane.

We must now use the earlier transformations.

$$\begin{pmatrix} x^1 \\ y^1 \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} + \begin{pmatrix} x_q - x_p \\ y_q - y_p \end{pmatrix} u + \begin{pmatrix} x_r - x_p \\ y_r - y_p \end{pmatrix} v \quad \text{for } Q_1 \text{ in } \Delta PQR \quad \text{-------------------------- (23)}$$

$$\begin{pmatrix} x^2 \\ y^2 \end{pmatrix} = \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} x_r - x_q \\ y_r - y_q \end{pmatrix} u + \begin{pmatrix} x_p - x_q \\ y_p - y_q \end{pmatrix} v \quad \text{for } Q_2 \text{ in } \Delta PQR \quad \text{-------------------------- (24)}$$

$$\begin{pmatrix} x^3 \\ y^3 \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \begin{pmatrix} x_p - x_r \\ y_p - y_r \end{pmatrix} u + \begin{pmatrix} x_q - x_r \\ y_q - y_r \end{pmatrix} v \quad \text{for } Q_3 \text{ in } \Delta PQR \quad \text{------------------------ (25)}$$

And we note that the above transformations viz Eqns.(23)-(25) are of the form

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

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 from Eqn.(34)

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

This gives

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

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

where

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

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

$\gamma_a = (x_c - x_b)$, $\gamma_b = (x_a - x_c)$, ------------------------(37a)

and

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

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

From Eqn.(33) and Eqn.(36), we obtain

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

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

$\gamma_a{}^* = \frac{\gamma_a}{(2\Delta_{abc})}$, $\gamma_b{}^* = \frac{\gamma_b}{(2\Delta_{abc})}$ --------------------------- (38b)

Letting,

$$D_{x,y}^{i,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{pmatrix} , \ P = \begin{pmatrix} \beta_a{}^* & \beta_b{}^* \\ \gamma_a{}^* & \gamma_b{}^* \end{pmatrix} , \ D_{u,v}^{i,e} = \begin{pmatrix} \frac{\partial N_i^e}{\partial u} \\ \frac{\partial N_i^e}{\partial v} \end{pmatrix} \quad \text{---------------------------- (39)}$$

We obtain from Eqn.(38) and Eqn.(39)

$$D_{x,y}^{i,e} = P\, D_{u,v}^{i,e} \quad \text{----------------------------- (40)}$$

Hence from Eqn.(39) and Eqn.(40)

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

$$= \begin{pmatrix} \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x} & \frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial y} \\ \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial x} & \frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y} \end{pmatrix} \quad \text{----------------------------- (41a)}$$

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

$$= \begin{pmatrix} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial u} & \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial v} \\ \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial u} & \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial v} \end{pmatrix} \quad \text{----------------------------- (41b)}$$

We have now from Eqn.(40) and Eqn.(41a- b)

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

= P ($D^{i,e}_{u,v}$) $\left(D^{j,e}_{u,v}\right)^T P^T$

= P $G^{i,j,e}_{u,v}$ P^T --------------------------- (41c)

We now define the submatrices of global derivative integrals in (x,y) and (u,v) space associated with the nodes i and j $(i, j = 1, 2, 3, 4, 5, 6, 7, 8)$ as

$$S^{i,j,e} = \iint_{Q_e} G^{i,j,e}_{x,y}\ \ dx\,dy\ , \qquad \text{---------------------------}\ (42)$$

$$K^{i,j,e} = \iint_{\hat{Q}} G^{i,j,e}_{u,v}\ \ du\,dv \qquad \text{---------------------------}\ (43)$$

where, we have already defined the 8- node linear convex quadrilaterals Q_e (e=1,2,3) in (x,y) space and $\hat{Q}$ in (u,v) space in Fig 3a- 3b. From Eqns.(41)-(43) , 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.(41a)- (41b)

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

$$= \begin{pmatrix} S^e_{2i-1\,,2j-1} & S^e_{2i-1\,,2j} \\ S^e_{2i\,,2j-1} & S^e_{2i\,,2j} \end{pmatrix}\ \ \text{(say)} \qquad \text{------------------}(44)$$

and in similar manner

$$K^{i,j,e} = \iint_{\hat{Q}} G^{i,j,e}_{u,v}\ \ du\,dv = \begin{pmatrix} \iint_{\hat{Q}} \frac{\partial N^e_i}{\partial u}\frac{\partial N^e_j}{\partial u}\ dudv & \iint_{\hat{Q}} \frac{\partial N^e_i}{\partial u}\frac{\partial N^e_j}{\partial v}\ dudv \\ \iint_{\hat{Q}} \frac{\partial N^e_i}{\partial v}\frac{\partial N^e_j}{\partial u}\ dudv & \iint_{\hat{Q}} \frac{\partial N^e_i}{\partial v}\frac{\partial N^e_j}{\partial v}\ dudv \end{pmatrix}$$

$$= \begin{pmatrix} K^e_{2i-1\,,2j-1} & K^e_{2i-1\,,2j} \\ K^e_{2i\,,2j-1} & K^e_{2i\,,2j} \end{pmatrix}\ \ \text{(say)} \qquad \text{------------------}(45)$$

We have now from the above Eqns.(41)-(45)

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

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

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

$$= 2\Delta_{abc}\ \ P\,(K^{i,j,e})\ \ P^T\ \ ,\ (i, j = 1, 2, 3, 4, 5, 6, 7, 8) \qquad \text{----------------}(46)$$

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.(46).

We note that $\widehat{Q}$ is the 8- node linear convex quadrilateral in (u, v) space spanned by the vertices (1/3, 1/3), (1/6, 5/12), (0, ½), (0, ¼), (0, 0), (1/4, 0), (1/2, 0) and (5/12, 1/6) in uv- plane hence from Eqn.(45)

$$K^{i,j,e} = \iint_{\widehat{Q}} G^{i,j,e}_{u,v} \; du\, dv \qquad \text{-------------------------(47)}$$

$$= \int_{-1}^{1}\int_{-1}^{1} G^{i,j,e}_{u,v} \frac{\partial(u,v)}{\partial(\xi,\eta)} \; d\xi\, d\eta \qquad \text{------------------------(48)}$$

We now refer to section 6.1 of this paper, in this section, we have derived the necessary relations to integrate Eq.(47). As in Eqns.(27)-(28), we use the transformation of Eqn.(29) to map the 8- node quadrilateral $\widehat{Q}$ to the 8- node 2-square $-1\leq \xi,\eta \leq 1$ Using Eqn.(29) in Eqn.(48), we obtain

$$K^{i,j,e} = \iint_{\widehat{Q}} G^{i,j,e}_{u,v} \left(\frac{4+\xi+\eta}{96}\right) d\xi d\eta \qquad \text{----------------------- (49)}$$

Thus, we have from Eq.(46)

$$S^{i,j,e} = (2\Delta_{abc}) \; P\,(K^{i,j,e}) \; P^T \qquad \text{---------------------- (50)}$$

Where $K^{i,j,e}$ is given in Eqn.(49)

In Eqn.(50) , $2\Delta_{abc}$= 2 * area of the triangle spanning vertices A(x_a ,y_a) ,B(x_b ,y_b) , C(x_c , y_c) is a 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 – 6. 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 as noted in Eqn.(45) is of order ($2xn_{de}$) x ($2xn_{de}$), n_{de} = 8 = for 8-node convex quadrilateral element.

We have computed K^e for the four node element n_{de}= 4 in our resent paper [18]. In the present paper, we have computed K^e for the 8- node linear convex quadrilateral $\widehat{Q}$ in uv – space. This is listed in Table 1A and Table 1B.

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(12a-b)

$$\boldsymbol{K^e_{i,j} = \int_{\Omega^e} \nabla\varphi_i \,.\, \nabla\varphi_j \, dx = \int_{\Omega^e}\left\{\frac{\partial\varphi_i}{\partial x}\frac{\partial\varphi_j}{\partial x} + \frac{\partial\varphi_i}{\partial y}\frac{\partial\varphi_j}{\partial y}\right\} dxdy}\ ,$$

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,(51a)

from the above derivations, we can rewrite $\boldsymbol{K^e_{i,j}}$ in the notations of this sections by taking $\boldsymbol{\varphi_i = N_i}$ and $\boldsymbol{\varphi_j = N_j \; and \; \Omega^e = Q_e}$ so that

$$\boldsymbol{K^e_{i,j} = \int_{Q_e} \nabla N_i \,.\, \nabla N_j \, dx = \int_{Q_e}\left\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\right\} dxdy = S^e_{2i-1,2j-1} + S^e_{2i,2j}}$$

....................................(51b)

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

The explicit integration scheme explained above compute four derivative product integrals as given in eqn(44) 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 (50).Since,we only need the sum of two of these integrals viz : $S_{2i-1,2j-1}^e + S_{2i,2j}^e$ .We now present an efficient method to compute this sum by using matrix product.

Let $F_{p.q}^{i,j} = \frac{\partial N_i}{\partial p}\frac{\partial N_j}{\partial q}$, $I_{p.q}^{i,j} = \int_{Q_e} F_{p.q}^{i,j}\, dpdq$, then we have from eqns(44-45) :

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

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

$$= \begin{pmatrix} I_{x,x}^{i,j} & I_{x,y}^{i,j} \\ I_{y,x}^{i,j} & I_{y,y}^{i,j} \end{pmatrix}$$

...(52a)

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

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

$$= \begin{pmatrix} I_{u,u}^{i,j} & I_{u,v}^{i,j} \\ I_{v,u}^{i,j} & I_{v,v}^{i,j} \end{pmatrix}$$

...(52b)

Let $P=\begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$, $P^T=\begin{pmatrix} P_{11} & P_{21} \\ P_{12} & P_{22} \end{pmatrix}$

...(53)

From eqns(44) , (46) and (52a-b)

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

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

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

...(54)

From eqn(51a-b) and eqn(46) , we find

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

$$=(P^2_{11}+P^2_{21})I^{i,j}_{u,u}+(P_{11}\,P_{12}+P_{21}\,P_{22})\left(I^{i,j}_{u,v}+I^{i,j}_{v,u}\right)+(P^2_{12}+P^2_{22})I^{i,j}_{v,v}$$

..........(55)

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:

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

.................................(56)

We observe from eqn(56) that sum of all the entries gives us the value of the integral i.e

$$\int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\}\, dxdy = sum(sum\left(\ (P^T\,P).*\begin{pmatrix} I^{i,j}_{u,u} & I^{i,j}_{u,v} \\ I^{i,j}_{v,u} & I^{i,j}_{v,v}\end{pmatrix}\ \right))$$

............................(57)

Where,sum is a 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 .

6.4 Computing of Force Vector Integrals $\int_{\Omega^e} f\varphi_i$ **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** $\phi(x, y)$

$$II_{\,\Delta PQR} = \iint_{\Delta PQR} \phi(x,y)\, dxdy = \sum_{e=1}^{3}\iint_{Q_e} \phi(x,y)\, dxdy$$

--------------------- (58)

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

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

-------------------- (59)

DOI: 10.18535/ijecs/v6i10.13

Where $(x^{(e)}(u,v),\ y^{(e)}(u,v)), e = 1, 2, 3)$ are the transformations of Eqs.(8)–(10) and $\widehat{Q}$ 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

$$II_{\Delta PQR} = (2\,\Delta_{pqr}) \iint_{\widehat{Q}} \{ \textstyle\sum_{e=1}^{3} [\phi\,(x^{(e)}(u,v),\ y^{(e)}(u,v)) \frac{\partial(u,v)}{\partial(\xi,\eta)} \}\, d\xi\, d\eta$$

--------------- (60)

In Eq.(14) we have used the transformation

$$u(\xi,\ \eta) = \frac{1}{3} N_1(\xi,\ \eta) + \frac{1}{2} N_4(\xi,\ \eta)$$
$$v(\xi,\ \eta) = \frac{1}{3} N_1(\xi,\ \eta) + \frac{1}{2} N_2(\xi,\ \eta)$$

---------------------- (61)

to map the quadrilateral $\widehat{Q}$ into a 2 – square in ξη – plane.

We can now obtain from Eqs.(14)–(15)

$$II_{\Delta PQR} = (2\,\Delta_{pqr}) \int_{-1}^{1}\int_{-1}^{1} [\textstyle\sum_{e=1}^{3} \left(\frac{4+\xi+\eta}{96}\right) \phi(x^{(e)}(u,v),\ y^{(e)}(u,v))\,] d\xi\, d\eta$$

------------- (62)

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

$$II_{\Delta PQR} = 2\Delta_{pqr} \sum_{i=1}^{N}\sum_{j=1}^{N} \left(\frac{W_i^{(N)} W_j^{(N)} (4+\xi_i^{(N)}+\eta_j^{(N)})}{96}\right) \sum_{e=1}^{3} \phi(x^{(e)}\left(u_{i,j}^{(N)}, v_{i,j}^{(N)}\right), y^{(e)}\left(u_{i,j}^{(N)}, v_{i,j}^{(N)}\right))$$

----------------------- (63)

Where,

$$u_{i,j}^{(N)} = u(\xi_i^{(N)},\ \eta_j^{(N)}) \quad \text{and} \quad v_{i,j}^{(N)} = v(\xi_i^{(N)},\ \eta_j^{(N)})$$

---------------------- (64)

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

7 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(7a-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.

7.1 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. In section 4.5,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for standard and arbitrary triangles,rectangles and convex polygonal domains.

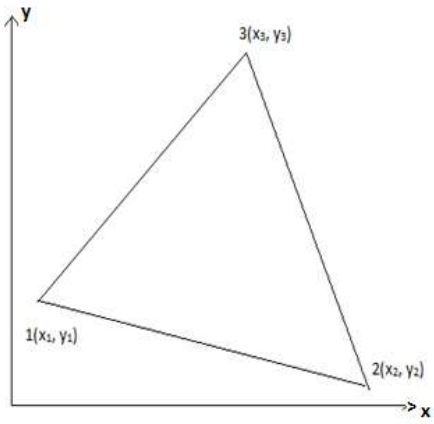
7.2 Division of an Arbitrary Triangle

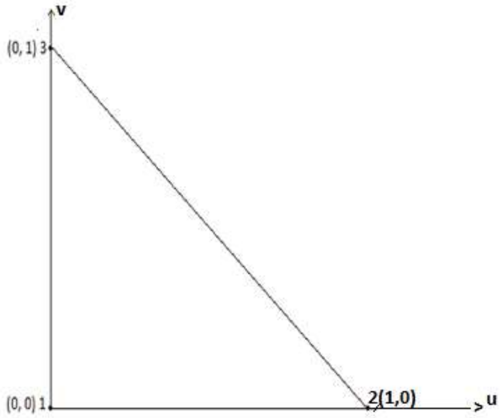
We can map an arbitrary triangle with vertices ($(x_i, y_i),\ i = 1,2,3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 4a, b. The necessary transformation is given by the equations.

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

$$y = y_1 + (y_2 - y_1)u + (y_3 - y_1)v$$
(57)

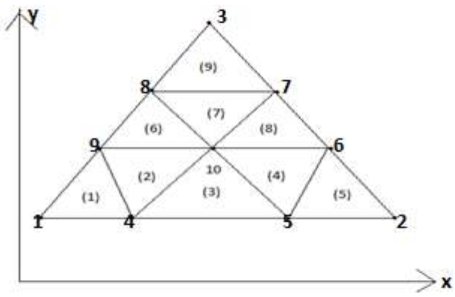
The mapping of eqn.(1) describes a unique relation between the coordinate systems. This is illustrated by using the area coordinates and division of each side into three equal parts in Fig. 5a Fig. 5b. It is clear that all the coordinates of this division can be determined by knowing the coordinates ((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.

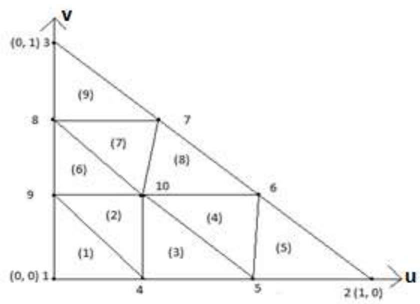




4.a **4 b**

Fig. 4a An Arbitrary Linear Triangle in the (x, y) space **Fig. 4b A Right Isosceles Triangle in the (u, v) space**





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

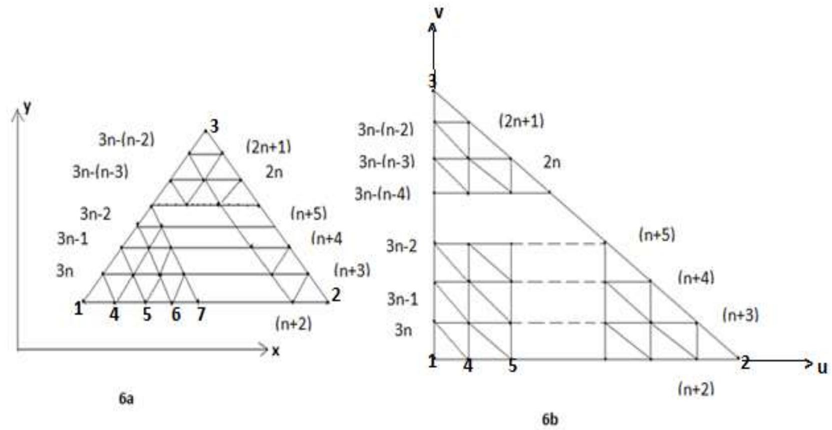


Fig.6a Division of an arbitrary triangle into n^2 triangle in Cartesian space (x, y), where each side is divided into n divisions of equal length

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

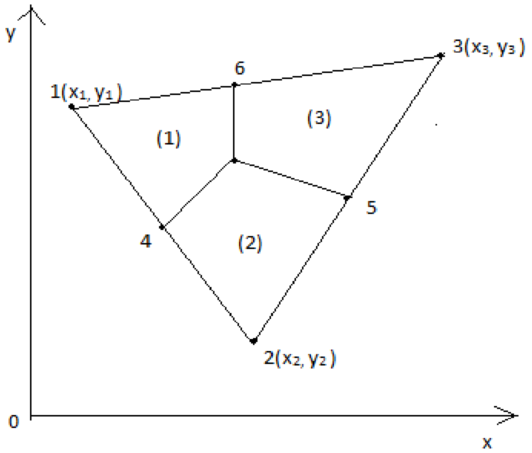
$$\underline{rr} = \begin{bmatrix} 1, & 4, & 5, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, & (n+2) & 2 \\ 3n, & (3n+1), \dots\dots\dots\dots\dots\dots\dots\dots, & 3n+(n-2), & (n+3) & 0 \\ 3n-1, & 3n+(n-1) \dots\dots\dots\dots & ,3n+(n-2)+(n-3), & (n+4) \quad 0 & 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots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

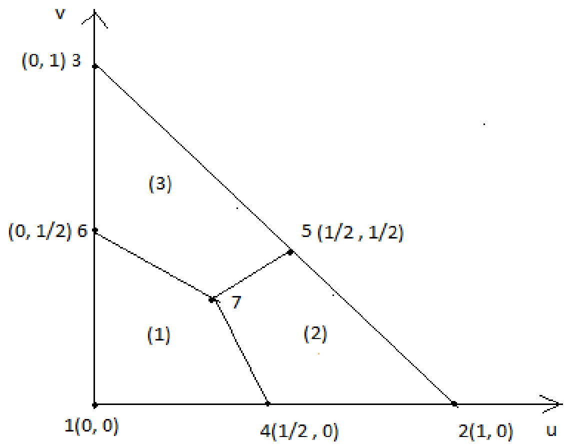
....(58)

7.3. 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



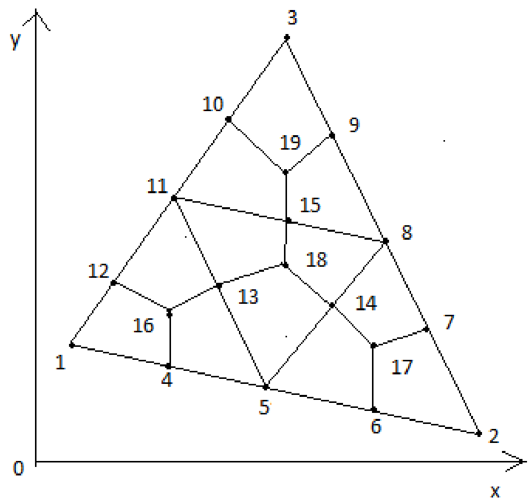


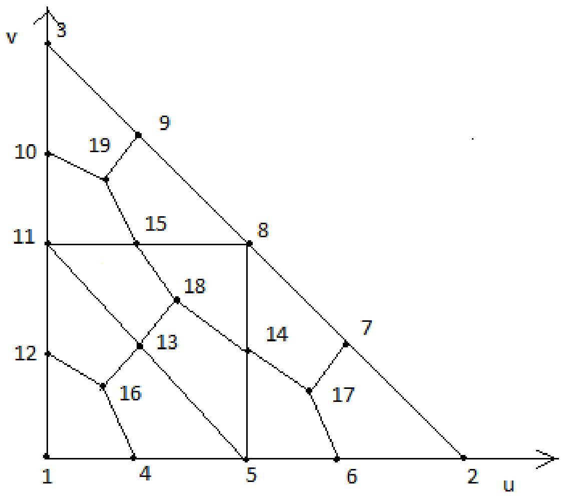
7(a) 7(b)

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 triangles using sub matrix $\underline{rr}$

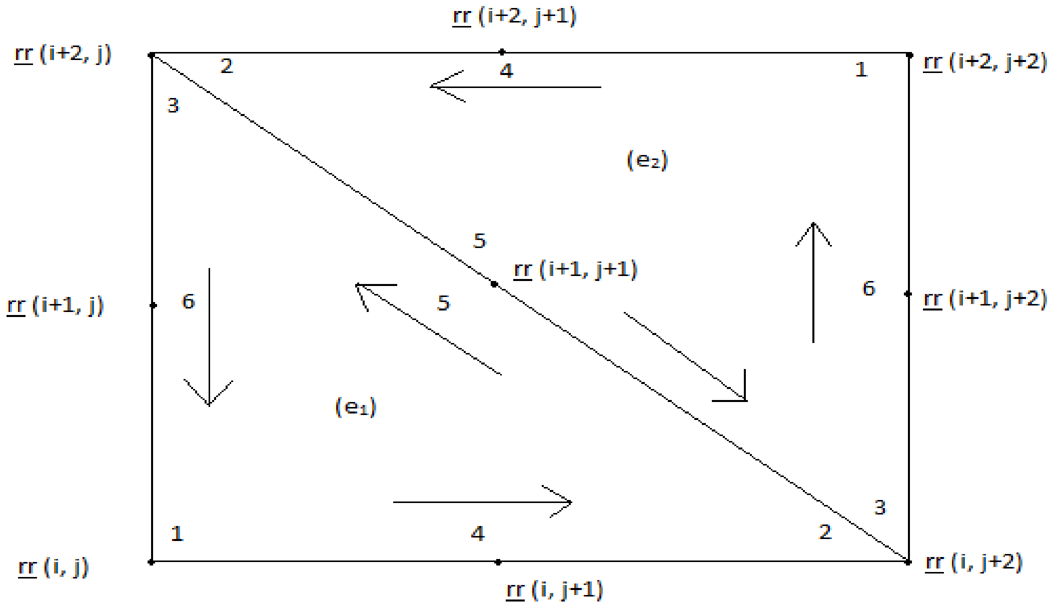


Fig. 9 Six node triangle formation for non zero sub matrix $\underline{rr}$

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) $<\underline{rr}\ (i+2, j+2), \underline{rr}\ (i+2, j), \underline{rr}\ (i, j+2), \underline{rr}\ (i+2, j+1), \underline{rr}\ (i+1, j+1), \underline{rr}\ \ i+1, j+2)>$ ....(59)

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$_1$, $\underline{rr}$ $(i+1,\ j+1)$, $\underline{rr}$ $(i+2,\ j)$, $\underline{rr}$ $(i+1, j)$ > ....(60)

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

Q_{3n_2-2} < c$_2$, $\underline{rr}$ $(i+1,\ j+2)$, $\underline{rr}\ (i+2,\ j+2)$, $\underline{rr}\ (i+2,\ j+1)$ >

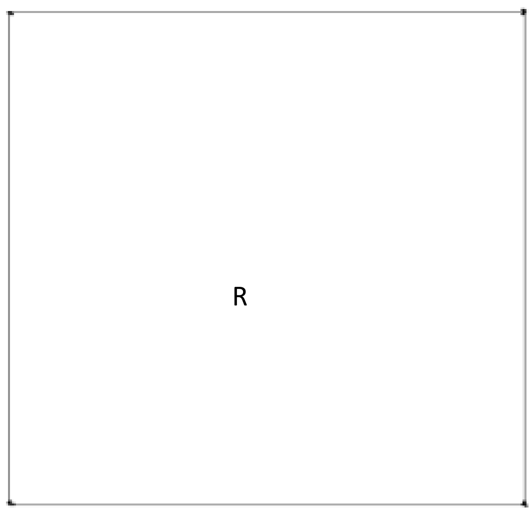
Q_{3n_2-1} < c$_2$, $\underline{rr}$ $(i+2,\ j+1)$, $\underline{rr}\ (i+2,\ j)$, $\underline{rr}\ (i+1,\ j+1)$ >

Q_{3n_1} < c$_2$, $\underline{rr}$ $(i+1,\ j+1)$, $\underline{rr}$ $(i,\ j+2)$, $\underline{rr}$ $(i+1, j+2)$ > --------------------- (61)

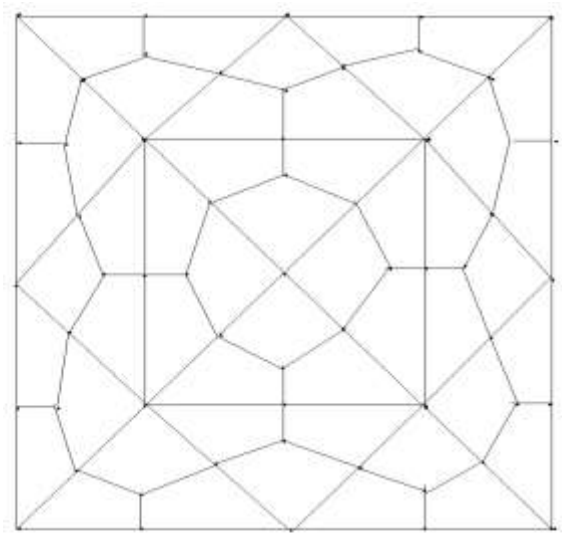
7.4 Quadrangulation of the Polygonal Domain

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)



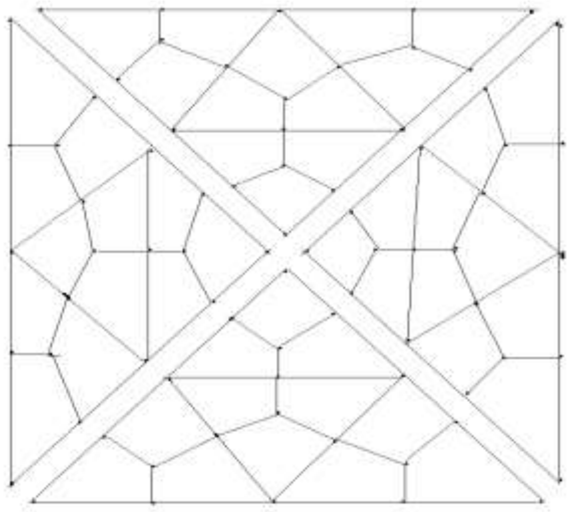
10a



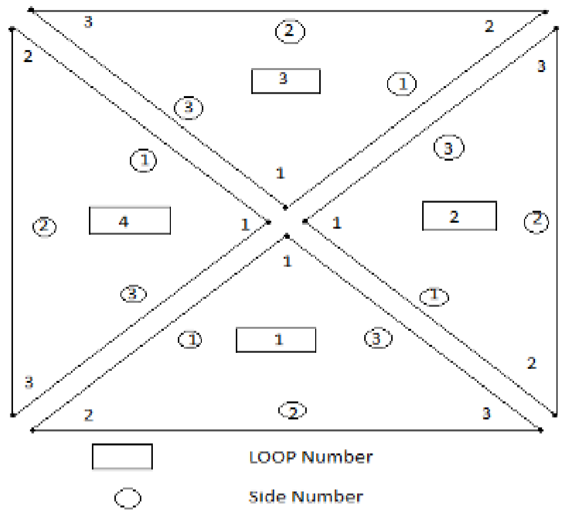
10b

(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.

The auto mesh generation technique discritises a polygonal domain into all four node special quadrilateral elements.We can convert these into eight node special quadrilateral elements by adding one node at the midpoint of each side of the four node special quadrilateral elements.We have written codes to carry this conversion schemes in the programs of all four node special quadrilaterals proposed in[ ].We include here some meshes all eight node special quadrilaterals at initial stagtes of mesh generation which is self explainatory.

Exam ple1: equilateral triangle,each side=2*sqrt(3)
x=sym([-sqrt(3);sqrt(3); 0])
y=sym([-1; -1; 2])

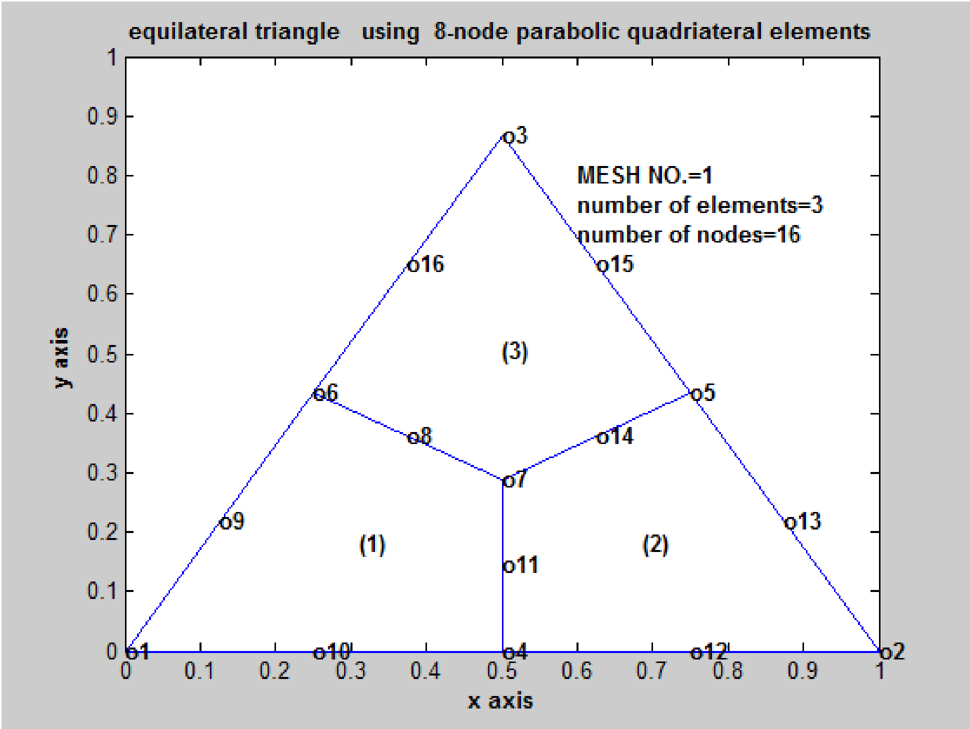


Fig.11a discritisation of equilateral triangle(initial mesh)

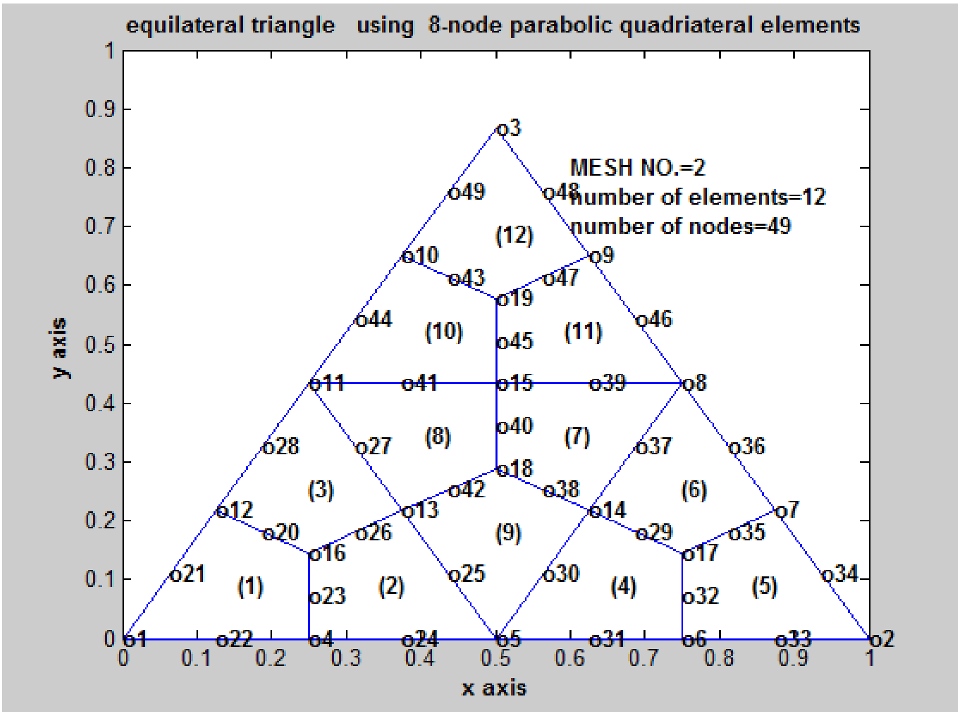


Fig.11a discritisation of equilateral triangle(first refinement of initial mesh)

Example 2:pentagonal domain with seven triangles(8-nodes)

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

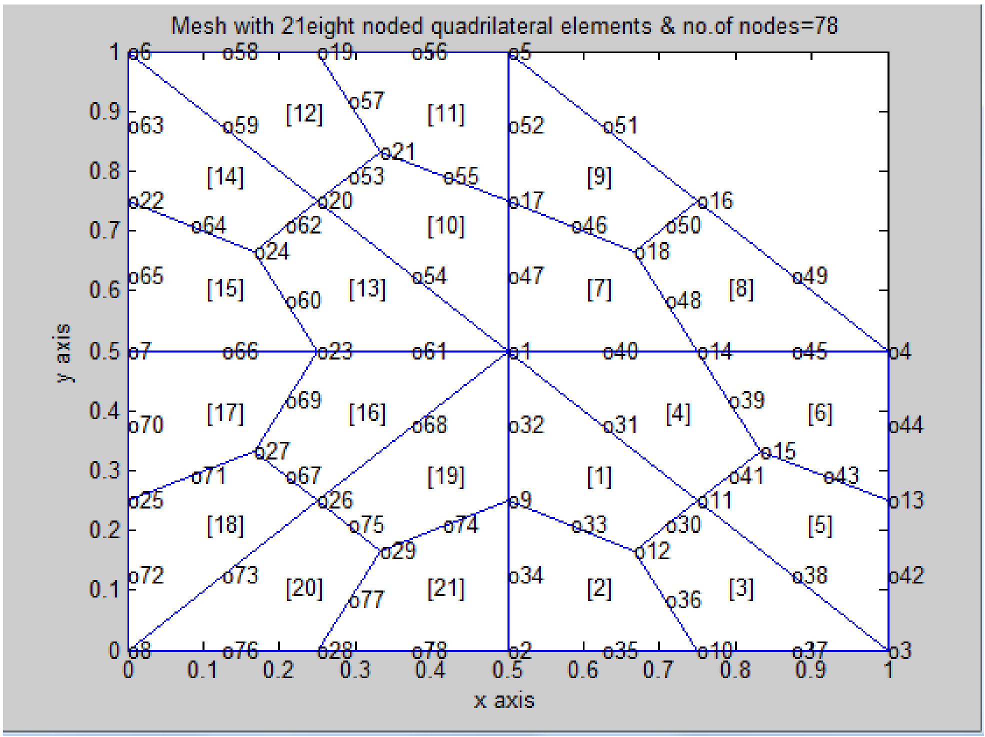


Fig 12: pentagonal domain(initial mesh)

Example 3 :a square domain with eight triangles(9-nodes)

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

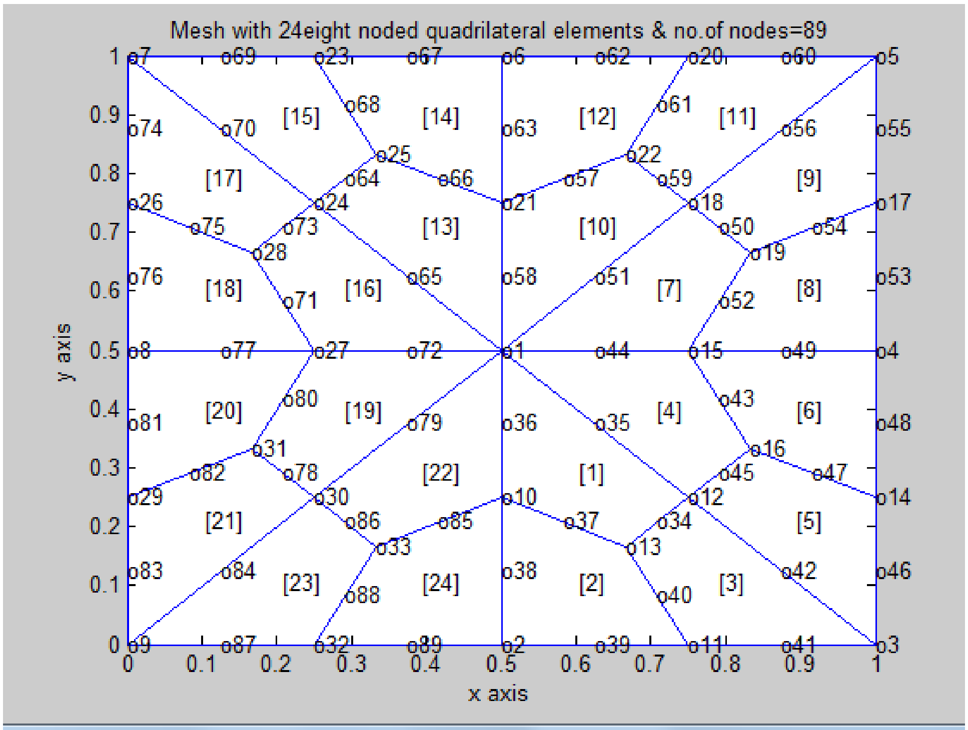


Fig 12: square domain(initial mesh)

7.5 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:
$-\Delta u = f,\ x\epsilon\Omega \subset \mathcal{R}^2$ ....(1)
$u = g,$ $\boldsymbol{x\epsilon\partial\Omega}$ **..(2)**
Where Ω is a polygonal domain and Δ is the standard standard Laplace operator

In this section, we examine the application of the proposed explicit integration scheme to the Saint Venant Torsion problem [24]. Exact solutions of this problem for simple cross sections such as circle, ellipse, equilateral triangle and rectangle have been rigorously derived. These problems are described by the following boundary value problem ;

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + 2G\theta = 0 \quad \text{in R} \qquad \text{-------------------- (39)}$$

$$\phi = 0 \quad \text{on} \quad \partial R \text{, the boundary of R} \qquad \text{--------------------- (40)}$$

where $\phi(x,y)$ is known as Prandtl stress function, G is the shear modulus, θ is the angle of twist per unit length, R is the cross sectional region and ∂R is the boundary of R. We choose $G\theta = 1$ for the sake of simplicity. Then the corresponding torisonal constant is given by the equation

$$t_c = 2\iint_R \phi(x,y)dx\,dy$$

7.5.1 Mesh Generation Over an Arbitrary Triangle

In applications to boundary value problems due to symmetry considerations or otherwise also, we may have to discretize an arbitrary triangle. Our purpose is to have a code which automatically generates convex quadrangulations of the domain by assuming the input as coordinates of the boundary vertices. We use the theory and procedure developed in section 7.2 and section 7.3 for this purpose..

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)
$u = 0,$ $x\epsilon\partial\Omega$ ....(10)
Where Ω is a regular polygonal domain and ∇^2 is the standard Laplace operator
In a recent paper[26] a new approach to automatic generation of all quadrilateral mesh for finite analysis is proposed and it was applied to discretise the 1/8-th of the square cross section a triangular region into an all quadrilateral mesh. We have demonstrated the proposed explicit integration scheme to solve the St. Venant Torsion problem for a square cross section. Monotonic convergence from below is observed with known analytical solutions for the Prandtl stress function and the torisonal constant which are expessed in terms of infinite series.This triangular domain is a right isosceles triangle and it was discretised by 8-noded special linear convex quadrilaterals.We would like to now illustrate the t St. Venant Torsion problem for an arbitrary triangular cross section using 8-node special linear convex quadrilaterals

We would like to consider the domain Ω for the linear elastic torsion of an equilateral triangle which is inscribed in a circle of unit radius. The following MATLAB codes are written to achieve this objective.

(1)quadrilateralmesh_over_arbitrarytriangle_q8automeshgen.m

(2)nodaladdresses_special_convex_quadrilaterals_2nd_order.m
(3)coordinate_arbitrarytriangle_2ndorder.m
(4)D2LaplaceEquationQ8Ex3automeshgenNew.m
(5)coordinate_special_quadrilaterals_in_stdtriangle_2nd_order.m

7.5.2 Mesh Generation over a Convex Polygonal Domain

In several physical applications in science and engineering, the boundary value problem require meshes generated over convex polygons. Again our aim is to have a code which automatically generates a mesh of convex quadrilaterals for the complex domains such as those in [21,22]. We use the theory and procedure developed in sections 7.2, 7.3 and 7.4 for this purpose. The following MATLAB codes are written for this purpose.

Example 1

$-\Delta u = 2\pi^2 sin(\pi x) sin(\pi y)$, $(x, y)\epsilon\Omega \subset \mathcal{R}^2$

$u(x, 0) = 0, on\ \ y = 0, 0 \le x \le 1$

$u(x, 1) = 0, on\ \ y = 1, 0 \le x \le 1$,

$u(1, y) = 0, on\ \ x = 1, 0 \le y \le 1/2$,

$u(x, y) = sin(\pi x) sin(\pi y)$, on the line x= 1 - 0.5t ,y=0..5+ 0.5t ,0≤ $t \le 1$
....(62)

Where Δ is a standard Laplace operator and Ω is a pentagonal domain joining the vertices
{(0,0),(1,0),(1,0.5),(0.5,1),(0,1)}

The exact solution of the above boundary value problem is $u(x, y) = sin(\pi x) sin(\pi y)$.

Example 2

$-\Delta u = 2\pi^2 sin(\pi x) sin(\pi y)$, $(x, y)\epsilon\Omega \subset \mathcal{R}^2$

u =0 , on the boundary $\partial\Omega$
....(63)

Where Δ is a standard Laplace operator and Ω is a square domain $[0,1]^2$.

We have written the following codes to solve the Poisson Equations with Dirichlet Boundary Conditions over linear convex polygonal domains

(1)quadrilateral_mesh4MOINEX_q8.m
(2)polygonal_domain_coordinates_2nd_order.m
(3)nodaladdresses_special_convex_quadrilaterals_trial_2nd_order.m
(4)generate_area_coordinate_over_the_standard_triangle.m
(5)D2LaplaceEquationQ8MoinExautomeshgen.m
(6)glsampleptsweights.m
(7)D2PoissonEquationQ8MoinEx_MeshgridContour.m

8.0 Conclusions

This paper presents the explicit integration scheme for a unique(special) linear convex 8- node 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 8- node quadrilaterals is derived by using the standard transformations in two steps. We first map an arbitrary linear triangle into a standard right isosceles triangle by using the affine linear transformation from global (x, y) space into a local space (u, v). We then discritise this standard right isosceles triangle in (u, v) space into three unique linear convex 8- node quadrilaterals. We have shown by proving a lemma that any unique linear convex 8-node quadrilateral in (x, y) space can be mapped into one of the unique 8-node quadrilaterals in (u, v) space. We have then mapped these linear convex 8- node quadrilaterals into a 2-square in the local (ξ,η) space by use of the bilinear transformation between (u, v) and (ξ,η) space. Using these two mappings, we have established an integral derivative product relation between the linear convex 8- node quadrilaterals in the global (x ,y) space interior to the arbitrary triangle and the linear convex 8-node quadrilaterals in the local (u, v) space which are interior to the standard right isosceles triangle. We have then shown that the product of global derivative integrals $S^{i,j,e}$ in global (x, y) space can be expressed as a matrix triple product $P*(K^{i,j,e})*P^T*(2*$ 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, and $(i,j=1,2,3,4,5,6,7,8)$. We have shown that the explicit integration of the global derivative products in (u, v) space over the unique 8- node quadrilateral spanning vertices {(1/3, 1/3), (1/6, 5/12), (0, ½), (0, ¼), (0, 0), (1/4, 0), (1/2, 0) and (5/12, 1/6)} is now possible by application of symbolic processing capabilities in MATLAB which are based on MAPLE –V mathematical software package. The proposed explicit integration scheme is a useful technique for boundary value problems governed by either a single or a system of partial differential equations. The physical applications of such problems are numerous in science, and engineering and business , the well known examples are the Laplace and Poisson equations with suitable boundary conditions and the examples of system of equations are the plane stress, plane stress and axisymmetric stress analysis, flow through porous media, shallow water circulation, dispersion and viscous incompressible flow etc in the areas of solid and fluid mechanics. We have first demonstrated the proposed explicit integration scheme to solve the St. Venant Torsion problem for an equilateral triangular cross section. Monotonic convergence from below is observed with known analytical solutions for the Prandtl stress function and the torisonal constant. We have demonstrated the proposed explicit integration scheme to solve the Poisson Boundary Value Problem for a pentagonal and square domains which are to be considered as simple polygonal domains. 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 eight noded quadrilateral meshes and contour level curves for FEM and exact solutions are also displayed. We conclude that efficient scheme on explicit integration of stiffness matrix and a novel automesh generation technique developed in this paper will be useful for the solution of many physical problems governed by second order partial differential equations.

We hope that the scheme developed in this paper will be useful for the solution of boundary value problems governed by second order partial differential equations.

REFERENCES:

[1] Zienkiewicz O.C, Taylor R.L and Zhu J.Z 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 Faires J.D Numerical Analysis, 9th Edition, Brooks/Cole, Cengage Learning (2011)

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

[6] Stoer J and Bulirsch R, 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 Pinder G.F, 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 Shafiqul Islam. Md , 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 Sajedul Karim, Md 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 Yoshimura S, 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 Shafiqul Islam Md, 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 Littlefield B, Mastering MATLAB 7 , Prentice Hall, Happer Saddle River, N J . (2005)
[18] Hunt B.H, Lipsman R.L and Rosenberg J.M, 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 Watt S, 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 Goodier J.N, 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] Rathod.H.T, Bharath Rathod, Shivaram K.T , H. Y. Shrivalli , Tara Rathod , K. Sugantha Devi,An explicit finite element integration scheme using automatic mesh generation technique for linear convex quadrilaterals over plane regions , international Journal of Engineering and Computer Science, Vol. 3,issue 4,pp5400-5435 (2014)
[29] Rathod.H.T, Bharath Rathod, K.T.Shivaram,Sugantha Devi.K,Tara Rathod ,An explicit finite element integration scheme for linear eight node convex quadrilaterals using automatic mesh generation technique over plane regions, international Journal of Engineering and Computer Science, Vol. 3,issue 4,pp5657-5713 (2014)
[30] 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),issue 8,August(2016), pp. 17397-17481

[31] Rathod.H.T, Sugantha Devi.K,Nagabhushana.C.S , Chudamani.H.M ,Finite element analysis of linear elastic torsion for regular polygons, International Journal of Engineering and Computer Science, ISSN:2319-7242,Volume(5),issue 10,October(2016), pp. 18413-18427

TABLE-1

EIGHT NODE SERENDIPITY ELEMENT

$(K^{e}_{i,j}, (i=1(1)16,\ j=1(1)16))$

ANALYTICAL VALUES FOR PRODUCTS OF GLOBAL DERIVATIVE INTEGRALS WITH 32-DIGITS PRECISION

OVER THE EIGHT NODE QUADRILATERAL $((u_k,v_k),k=1,2,3,4)=((1/3,1/3),(0,1/2),(0,0),(1/2,0))$,WITH

$(u_5,v_5),(u_6,v_6),(u_7,v_7)$ AND (u_8,v_8) AS THE MIDPOINT OF SIDES 1-2,2-3,3-4,AND 4-1 RESPECTIVELY IN

THE INTERIOR OF THE STANDARD TRIANGLE IN (u,v) SPACE (see eqn ())

$K^{p,q,e}=[K^{e}_{2p-1,2q-1}\ K^{e}_{2p-1,2q}]$

$[K^{e}_{2p,2q-1}\quad K^{e}_{2p,2q}]$

where, (p,q=1,2,3,4,5,6,7,8)

p	$K^{p,1,e}$	
1	-58259/630-93266/105*log(2)+22599/35*log(3)	-545/126-13124/21*log(2)+5589/14*log(3)
	-545/126-13124/21*log(2)+5589/14*log(3)	-58259/630-93266/105*log(2)+22599/35*log(3)
2	9223/135+1868/15*log(2)-702/5*log(3)	-479/270-1112/15*log(2)+243/5*log(3)
	-262/135-1112/15*log(2)+243/5*log(3)	-26893/270-5284/15*log(2)+1566/5*log(3)
3	-359/189+4306/21*log(2)-891/7*log(3)	-5954/945+18916/105*log(2)-7533/70*log(3)
	-5954/945+18916/105*log(2)-7533/70*log(3)	-359/189+4306/21*log(2)-891/7*log(3)
4	-26893/270-5284/15*log(2)+1566/5*log(3)	-262/135-1112/15*log(2)+243/5*log(3)
	-479/270-1112/15*log(2)+243/5*log(3)	9223/135+1868/15*log(2)-702/5*log(3)
5	-18082/315+37784/105*log(2)-6156/35*log(3)	961/630+53944/105*log(2)-11421/35*log(3)
	1381/630+53944/105*log(2)-11421/35*log(3)	1367/7+22576/21*log(2)-5994/7*log(3)
6	-45224/945-40024/105*log(2)+9936/35*log(3)	10039/1890-22808/105*log(2)+4617/35*log(3)
	10039/1890-22808/105*log(2)+4617/35*log(3)	33673/945-14992/105*log(2)+1998/35*log(3)
7	33673/945-14992/105*log(2)+1998/35*log(3)	10039/1890-22808/105*log(2)+4617/35*log(3)
	10039/1890-22808/105*log(2)+4617/35*log(3)	-45224/945-40024/105*log(2)+9936/35*log(3)
8	1367/7+22576/21*log(2)-5994/7*log(3)	1381/630+53944/105*log(2)-11421/35*log(3)
	961/630+53944/105*log(2)-11421/35*log(3)	-18082/315+37784/105*log(2)-6156/35*log(3)

DOI: 10.18535/ijecs/v6i10.13

p	$K^{p,2,e}$	
1	9223/135+1868/15*log(2)-702/5*log(3)	-262/135-1112/15*log(2)+243/5*log(3)
	-479/270-1112/15*log(2)+243/5*log(3)	-26893/270-5284/15*log(2)+1566/5*log(3)
2	-1892/45-5288/45*log(2)+564/5*log(3)	-703/45-2512/45*log(2)+246/5*log(3)
	-703/45-2512/45*log(2)+246/5*log(3)	-4352/45-12008/45*log(2)+1284/5*log(3)
3	-268/135+44/5*log(2)-18/5*log(3)	-53/135+104/5*log(2)-63/5*log(3)
	-151/270+104/5*log(2)-63/5*log(3)	523/270+188/5*log(2)-126/5*log(3)
4	323/5+7288/45*log(2)-804/5*log(3)	72/5+1712/45*log(2)-186/5*log(3)
	72/5+1712/45*log(2)-186/5*log(3)	323/5+7288/45*log(2)-804/5*log(3)
5	3746/135+592/5*log(2)-504/5*log(3)	3268/135+656/5*log(2)-522/5*log(3)
	3178/135+656/5*log(2)-522/5*log(3)	26528/135+2976/5*log(2)-2772/5*log(3)
6	4666/135+3248/45*log(2)-384/5*log(3)	202/27-16/9*log(2)-6*log(3)
	220/27-16/9*log(2)-6*log(3)	4024/135+1952/45*log(2)-276/5*log(3)
7	-526/27-608/9*log(2)+60*log(3)	-964/135-2192/45*log(2)+186/5*log(3)
	-964/135-2192/45*log(2)+186/5*log(3)	-6782/135-7696/45*log(2)+768/5*log(3)
8	-17782/135-1504/5*log(2)+1548/5*log(3)	-2834/135-48/5*log(2)+126/5*log(3)
	-2834/135-48/5*log(2)+126/5*log(3)	-1250/27-48*log(2)+72*log(3)

p	$K^{p,3,e}$	
1	-359/189+4306/21*log(2)-891/7*log(3)	-5954/945+18916/105*log(2)-7533/70*log(3)
	-5954/945+18916/105*log(2)-7533/70*log(3)	-359/189+4306/21*log(2)-891/7*log(3)
2	-268/135+44/5*log(2)-18/5*log(3)	-151/270+104/5*log(2)-63/5*log(3)
	-53/135+104/5*log(2)-63/5*log(3)	523/270+188/5*log(2)-126/5*log(3)
3	2617/1890-6914/105*log(2)+1431/35*log(3)	7211/1890-5812/105*log(2)+2241/70*log(3)
	7211/1890-5812/105*log(2)+2241/70*log(3)	2617/1890-6914/105*log(2)+1431/35*log(3)
4	523/270+188/5*log(2)-126/5*log(3)	-53/135+104/5*log(2)-63/5*log(3)
	-151/270+104/5*log(2)-63/5*log(3)	-268/135+44/5*log(2)-18/5*log(3)
5	4042/945-15608/105*log(2)+3132/35*log(3)	2117/378-3128/21*log(2)+621/7*log(3)
	2117/378-3128/21*log(2)+621/7*log(3)	-1019/945-20144/105*log(2)+4266/35*log(3)
6	88/945+8888/105*log(2)-1872/35*log(3)	-6707/1890+6904/105*log(2)-1341/35*log(3)
	-7967/1890+6904/105*log(2)-1341/35*log(3)	-2579/945+7376/105*log(2)-1494/35*log(3)
7	-2579/945+7376/105*log(2)-1494/35*log(3)	-7967/1890+6904/105*log(2)-1341/35*log(3)
	-6707/1890+6904/105*log(2)-1341/35*log(3)	88/945+8888/105*log(2)-1872/35*log(3)
8	-1019/945-20144/105*log(2)+4266/35*log(3)	2117/378-3128/21*log(2)+621/7*log(3)
	2117/378-3128/21*log(2)+621/7*log(3)	4042/945-15608/105*log(2)+3132/35*log(3)

DOI: 10.18535/ijecs/v6i10.13

p	$K^{p,4,e}$	
1	-26893/270-5284/15*log(2)+1566/5*log(3)	-479/270-1112/15*log(2)+243/5*log(3)
	-262/135-1112/15*log(2)+243/5*log(3)	9223/135+1868/15*log(2)-702/5*log(3)
2	323/5+7288/45*log(2)-804/5*log(3)	72/5+1712/45*log(2)-186/5*log(3)
	72/5+1712/45*log(2)-186/5*log(3)	323/5+7288/45*log(2)-804/5*log(3)
3	523/270+188/5*log(2)-126/5*log(3)	-151/270+104/5*log(2)-63/5*log(3)
	-53/135+104/5*log(2)-63/5*log(3)	-268/135+44/5*log(2)-18/5*log(3)
4	-4352/45-12008/45*log(2)+1284/5*log(3)	-703/45-2512/45*log(2)+246/5*log(3)
	-703/45-2512/45*log(2)+246/5*log(3)	-1892/45-5288/45*log(2)+564/5*log(3)
5	-.4672761790697733678657810678869	.38224148468296660162159011855e-1
	.38224148468296660162159011855e-1	-.86825859303307135254015301930e-1
6	-6782/135-7696/45*log(2)+768/5*log(3)	-964/135-2192/45*log(2)+186/5*log(3)
	-964/135-2192/45*log(2)+186/5*log(3)	-526/27-608/9*log(2)+60*log(3)
7	4024/135+1952/45*log(2)-276/5*log(3)	220/27-16/9*log(2)-6*log(3)
	202/27-16/9*log(2)-6*log(3)	4666/135+3248/45*log(2)-384/5*log(3)
8	26528/135+2976/5*log(2)-2772/5*log(3)	3178/135+656/5*log(2)-522/5*log(3)
	3268/135+656/5*log(2)-522/5*log(3)	3746/135+592/5*log(2)-504/5*log(3)

p	$K^{p,5,e}$	
1	-18082/315+37784/105*log(2)-6156/35*log(3)	1381/630+53944/105*log(2)-11421/35*log(3)
	961/630+53944/105*log(2)-11421/35*log(3)	1367/7+22576/21*log(2)-5994/7*log(3)
2	3746/135+592/5*log(2)-504/5*log(3)	3178/135+656/5*log(2)-522/5*log(3)
	3268/135+656/5*log(2)-522/5*log(3)	26528/135+2976/5*log(2)-2772/5*log(3)
3	4042/945-15608/105*log(2)+3132/35*log(3)	2117/378-3128/21*log(2)+621/7*log(3)
	2117/378-3128/21*log(2)+621/7*log(3)	-1019/945-20144/105*log(2)+4266/35*log(3)
4	-1250/27-48*log(2)+72*log(3)	-2834/135-48/5*log(2)+126/5*log(3)
	-2834/135-48/5*log(2)+126/5*log(3)	-17782/135-1504/5*log(2)+1548/5*log(3)
5	-3268/315-45344/105*log(2)+9936/35*log(3)	-10522/315-55456/105*log(2)+12744/35*log(3)
	-10522/315-55456/105*log(2)+12744/35*log(3)	-41204/105-160256/105*log(2)+46224/35*log(3)
6	-26512/945+13088/105*log(2)-1872/35*log(3)	-15824/945+15136/105*log(2)-2664/35*log(3)
	-15824/945+15136/105*log(2)-2664/35*log(3)	-61288/945+4352/105*log(2)+1152/35*log(3)
7	9944/945+21824/105*log(2)-4896/35*log(3)	5848/945+22528/105*log(2)-4932/35*log(3)
	5848/945+22528/105*log(2)-4932/35*log(3)	93272/945+51392/105*log(2)-13968/35*log(3)
8	10456/105-19136/105*log(2)+864/35*log(3)	706/21-6656/21*log(2)+1188/7*log(3)
	706/21-6656/21*log(2)+1188/7*log(3)	10456/105-19136/105*log(2)+864/35*log(3)

P	$K^{p,6,e}$	
1	-45224/945-40024/105*log(2)+9936/35*log(3)	10039/1890-22808/105*log(2)+4617/35*log(3)
	10039/1890-22808/105*log(2)+4617/35*log(3)	33673/945-14992/105*log(2)+1998/35*log(3)
2	4666/135+3248/45*log(2)-384/5*log(3)	220/27-16/9*log(2)-6*log(3)
	202/27-16/9*log(2)-6*log(3)	4024/135+1952/45*log(2)-276/5*log(3)
3	88/945+8888/105*log(2)-1872/35*log(3)	-7967/1890+6904/105*log(2)-1341/35*log(3)
	-6707/1890+6904/105*log(2)-1341/35*log(3)	-2579/945+7376/105*log(2)-1494/35*log(3)
4	-6782/135-7696/45*log(2)+768/5*log(3)	-964/135-2192/45*log(2)+186/5*log(3)
	-964/135-2192/45*log(2)+186/5*log(3)	-526/27-608/9*log(2)+60*log(3)
5	-26512/945+13088/105*log(2)-1872/35*log(3)	-15824/945+15136/105*log(2)-2664/35*log(3)
	-15824/945+15136/105*log(2)-2664/35*log(3)	-61288/945+4352/105*log(2)+1152/35*log(3)
6	-4604/189-10720/63*log(2)+912/7*log(3)	382/945-28384/315*log(2)+1992/35*log(3)
	382/945-28384/315*log(2)+1992/35*log(3)	-5716/945-32768/315*log(2)+2544/35*log(3)
7	16208/945-15296/315*log(2)+528/35*log(3)	7606/945-20992/315*log(2)+1236/35*log(3)
	7606/945-20992/315*log(2)+1236/35*log(3)	16208/945-15296/315*log(2)+528/35*log(3)
8	93272/945+51392/105*log(2)-13968/35*log(3)	5848/945+22528/105*log(2)-4932/35*log(3)
	5848/945+22528/105*log(2)-4932/35*log(3)	9944/945+21824/105*log(2)-4896/35*log(3)

P	$K^{p,7,e}$	
1	33673/945-14992/105*log(2)+1998/35*log(3)	10039/1890-22808/105*log(2)+4617/35*log(3)
	10039/1890-22808/105*log(2)+4617/35*log(3)	-45224/945-40024/105*log(2)+9936/35*log(3)
2	-526/27-608/9*log(2)+60*log(3)	-964/135-2192/45*log(2)+186/5*log(3)
	-964/135-2192/45*log(2)+186/5*log(3)	-6782/135-7696/45*log(2)+768/5*log(3)
3	-2579/945+7376/105*log(2)-1494/35*log(3)	-6707/1890+6904/105*log(2)-1341/35*log(3)
	-7967/1890+6904/105*log(2)-1341/35*log(3)	88/945+8888/105*log(2)-1872/35*log(3)
4	4024/135+1952/45*log(2)-276/5*log(3)	202/27-16/9*log(2)-6*log(3)
	220/27-16/9*log(2)-6*log(3)	4666/135+3248/45*log(2)-384/5*log(3)
5	9944/945+21824/105*log(2)-4896/35*log(3)	5848/945+22528/105*log(2)-4932/35*log(3)
	5848/945+22528/105*log(2)-4932/35*log(3)	93272/945+51392/105*log(2)-13968/35*log(3)
6	16208/945-15296/315*log(2)+528/35*log(3)	7606/945-20992/315*log(2)+1236/35*log(3)
	7606/945-20992/315*log(2)+1236/35*log(3)	16208/945-15296/315*log(2)+528/35*log(3)
7	-5716/945-32768/315*log(2)+2544/35*log(3)	382/945-28384/315*log(2)+1992/35*log(3)
	382/945-28384/315*log(2)+1992/35*log(3)	-4604/189-10720/63*log(2)+912/7*log(3)
8	-61288/945+4352/105*log(2)+1152/35*log(3)	-15824/945+15136/105*log(2)-2664/35*log(3)
	-15824/945+15136/105*log(2)-2664/35*log(3)	-26512/945+13088/105*log(2)-1872/35*log(3)

DOI: 10.18535/ijecs/v6i10.13

p	$K^{p,4,e}$	
1	1367/7+22576/21*log(2)-5994/7*log(3)	961/630+53944/105*log(2)-11421/35*log(3)
	1381/630+53944/105*log(2)-11421/35*log(3)	-18082/315+37784/105*log(2)-6156/35*log(3)
2	-17782/135-1504/5*log(2)+1548/5*log(3)	-2834/135-48/5*log(2)+126/5*log(3)
	-2834/135-48/5*log(2)+126/5*log(3)	-1250/27-48*log(2)+72*log(3)
3	-1019/945-20144/105*log(2)+4266/35*log(3)	2117/378-3128/21*log(2)+621/7*log(3)
	2117/378-3128/21*log(2)+621/7*log(3)	4042/945-15608/105*log(2)+3132/35*log(3)
4	26528/135+2976/5*log(2)-2772/5*log(3)	3268/135+656/5*log(2)-522/5*log(3)
	3178/135+656/5*log(2)-522/5*log(3)	3746/135+592/5*log(2)-504/5*log(3)
5	10456/105-19136/105*log(2)+864/35*log(3)	706/21-6656/21*log(2)+1188/7*log(3)
	706/21-6656/21*log(2)+1188/7*log(3)	10456/105-19136/105*log(2)+864/35*log(3)
6	93272/945+51392/105*log(2)-13968/35*log(3)	5848/945+22528/105*log(2)-4932/35*log(3)
	5848/945+22528/105*log(2)-4932/35*log(3)	9944/945+21824/105*log(2)-4896/35*log(3)
7	-61288/945+4352/105*log(2)+1152/35*log(3)	-15824/945+15136/105*log(2)-2664/35*log(3)
	-15824/945+15136/105*log(2)-2664/35*log(3)	-26512/945+13088/105*log(2)-1872/35*log(3)
8	-41204/105-160256/105*log(2)+46224/35*log(3)	-10522/315-55456/105*log(2)+12744/35*log(3)
	-10522/315-55456/105*log(2)+12744/35*log(3)	-3268/315-45344/105*log(2)+9936/35*log(3)

TABLE-2

EIGHT NODE SERENDIPITY ELEMENT

$(K^{e}_{i,j}, (i=1(1)16,\ j=1(1)16))$

NUMERICAL VALUES FOR PRODUCTS OF GLOBAL DERIVATIVE INTEGRALS WITH 32-DIGITS PRECISION

OVER THE EIGHT NODE QUADRILATERAL $((u_k,v_k),k=1,2,3,4)=((1/3,1/3),(0,1/2),(0,0),(1/2,0))$,WITH

$(u_5,v_5),(u_6,v_6),(u_7,v_7)$ AND (u_8,v_8) AS THE MIDPOINT OF SIDES 1-2,2-3,3-4,AND 4-1 RESPECTIVELY IN

THE INTERIOR OF THE STANDARD TRIANGLE IN (u,v) SPACE (see eqn ())

$K^{p,q,e}=[K^{e}_{2p-1,2q-1}\ K^{e}_{2p-1,2q}]$

$[K^{e}_{2p,2q-1}\quad K^{e}_{2p,2q}]$

where, (p,q=1,2,3,4,5,6,7,8)

p	$K^{p,1,e}$	
1	1.19732437518704939126225670841	1.07234243081152493747384516139
	1.07234243081152493747384516139	1.19732437518704939126225670841
2	.393282075247773712718727444686	.233172169685444656270703169594
	.66505503018777989604036502927e-1	.30901830189818397724346918143
3	.39105823773230516010581465792	.34520910581966151866889237468
	.34520910581966151866889237468	.39105823773230516010581465792
4	.30901830189818397724346918143	.66505503018777989604036502927e-1
	.233172169685444656270703169594	.393282075247773712718727444686
5	-1.20601914456199564761927515849	.86147306517224684195667929589
	-.19480639850558017529001262923	-.27406792898205548885083220480
6	-.19025279048691687518778377091	-.33047487282879104238539264173
	-.33047487282879104238539264173	-.620343126034344229672376860410
7	-.620343126034344229672376860410	-.33047487282879104238539264173
	-.33047487282879104238539264173	-.19025279048691687518778377091
8	-.27406792898205548885083220480	-.19480639850558017529001262923
	-.86147306517224684195667929589	-1.20601914456199564761927515849

p	$K^{p,2,e}$	
1	.393282075247773712718727444686	.66505503018777989604036502927e-1
	.233172169685444656270703169594	.30901830189818397724346918143
2	.42652636618519994053203031884	-.263513565675283566822536323478
	-.263513565675283566822536323478	.45005030410782929514068096460
3	.1595057645371386486635746307258	.182353925836087731705745548513
	.15687259169421065039078881847e-1	.314341351654616447964784833418
4	.20218069152088195770651481813	-.97977735817538970518647437153e-1
	-.97977735817538970518647437153e-1	.20218069152088195770651481813
5	-.92334437129978410949228855299	.45319455992158022128465900801
	-.21347210674508644538200765866	-.574726461686104068369695422e-2
6	.219362359001080107967460556119	-.3424539048559683280761803782014
	.3242127618106983385904862884652	-.768806561449731024667082979907
7	-.390687025888983122842003915751	-.36332930895951737339235932476e-1
	-.36332930895951737339235932476e-1	-.3376064404514624483888879556e-1
8	-.8682585930330713525401530193e-1	.38224148468296660162159011855e-1
	.38224148468296660162159011855e-1	-.467276179069773367865781067869

p	$K^{p,3,e}$	
1	.39105823773230516010581465792	.34520910581966151866889237468
	.34520910581966151866889237468	.39105823773230516010581465792
2	.1595057645371386486635746307258	.15687259169421065039078881847e-1
	.182353925836087731705745548513	.314341351654616447964784833418
3	.660112740472589759408893364145	.619284819567235241935951286773
	.619284819567235241935951286773	.660112740472589759408893364145
4	.314341351654616447964784833418	.182353925836087731705745548513
	.15687259169421065039078881847e-1	.1595057645371386486635746307258
5	-.447323988028158265936415185565	-.182789328176641032221385163017
	-.182789328176641032221385163017	-.15179959092659718711045271183
6	.6317194613977723832758311823e-2	-.65144893686228413120115549553e-1
	-.731811560352895079786782216220	-.932211710055872286928957900634
7	-.932211710055872286928957900634	-.731811560352895079786782216220
	-.65144893686228413120115549553e-1	.6317194613977723832758311823e-2
8	-.15179959092659718711045271183	-.182789328176641032221385163017
	-.182789328176641032221385163017	-.447323988028158265936415185565

p	$K^{p,4,e}$	
1	.30901830189818397724346918143	.233172169685444656270703169594
	.66505503018777989604036502927e-1	.393282075247773712718727444686
2	.20218069152088195770651481813	-.97977735817538970518647437153e-1
	-.97977735817538970518647437153e-1	.20218069152088195770651481813
3	.314341351654616447964784833418	.15687259169421065039078881847e-1
	.182353925836087731705745548513	.1595057645371386486635746307258
4	.45005030410782929514068096460	-.263513565675283566822536323478
	-.263513565675283566822536323478	.42652636618519994053203031884
5	-.467276179069773367865781067869	.38224148468296660162159011855e-1
	.38224148468296660162159011855e-1	-.8682585930330713525401530193e-1
6	-.3376064404514624483888879556e-1	-.36332930895951737339235932476e-1
	-.36332930895951737339235932476e-1	-.390687025888983122842003915751
7	-.768806561449731024667082979907	.3242127618106983385904862884652
	-.3424539048559683280761803782014	.219362359001080107967460556119
8	-.574726461686104068369695422e-2	-.21347210674508644538200765866
	.45319455992158022128465900801	-.92334437129978410949228855299

P	$K^{p,5,e}$	
1	-1.2060191445619956476192751584 -.86147306517224684195667929589	-.19480639850558017529001262923 -.27406792898205548885083220480
2	-.92334437129978410949228855299 .45319455992158022128465900801	-.21347210674508644538200765866 -.574726461686104068369695422e-2
3	-.447323988028158265936415185565 -.182789328176641032221385163017	-.182789328176641032221385163017 -.15179959092659718711045271183
4	-.467276179069773367865781067869 .38224148468296660162159011855e-1	.38224148468296660162159011855e-1 -.8682585930330713525401530193e-1
5	2.17177154262400226248727022336 .52992039898580294331102892478	.52992039898580294331102892478 .5882735844715392304142536963
6	-.415943731136358047626104977997 -.446246745737358785845683479200	-.446246745737358785845683479200 .34312587200869545409370986785e-1
7	.91157772830419213711619683500 .9457179332261632804889985221e-1	.9457179332261632804889985221e-1 -.48070367101146296286102539491
8	.376558143167875038936397884560 .37459823838795050721700114125	.37459823838795050721700114125 .376558143167875038936397884560

P	$K^{p,6,e}$	
1	-.19025279048691687518778377091 -.33047487282879104238539264173	-.33047487282879104238539264173 -.620343126034344229672376860410
2	.219362359001080107967460556119 -.3424539048559683280761803782014	.3242127618106983385904862884652 -.768806561449731024667082979907
3	.6317194613977723832758311823e-2 -.65144893686228413120115549553e-1	-.731811560352895079786782216220 -.932211710055872286928957900634
4	-.3376064404514624483888879556e-1 -.36332930895951737339235932476e-1	-.36332930895951737339235932476e-1 -.390687025888983122842003915751
5	-.415943731136358047626104977997 -.446246745737358785845683479200	-.446246745737358785845683479200 .34312587200869545409370986785e-1
6	.82865433521530100559964914224 .472904354164178470376423509241	.472904354164178470376423509241 1.699831160074343688470918905720
7	.66326947849525293113934929200e-1 .653177200517503508341284619711	.653177200517503508341284619711 .66326947849525293113934929200e-1
8	-.48070367101146296286102539491 .9457179332261632804889985221e-1	.9457179332261632804889985221e-1 .91157772830419213711619683500

P	$K^{p,7,e}$	
1	-.620343126034344229672376860410 -.33047487282879104238539264173	-.33047487282879104238539264173 -.19025279048691687518778377091
2	-.390687025888983122842003915751 -.36332930895951737339235932476e-1	-.36332930895951737339235932476e-1 -.3376064404514624483888879556e-1
3	-.932211710055872286928957900634 -.731811560352895079786782216220	-.65144893686228413120115549553e-1 .6317194613977723832758311823e-2
4	-.768806561449731024667082979907 .3242127618106983385904862884652	-.3424539048559683280761803782014 .219362359001080107967460556119
5	.91157772830419213711619683500 .9457179332261632804889985221e-1	.9457179332261632804889985221e-1 -.48070367101146296286102539491
6	.66326947849525293113934929200e-1 .653177200517503508341284619711	.653177200517503508341284619711 .66326947849525293113934929200e-1
7	1.699831160074343688470918905720 .472904354164178470376423509241	.472904354164178470376423509241 .82865433521530100559964914224
8	.34312587200869545409370986785e-1 -.446246745737358785845683479200	-.446246745737358785845683479200 -.415943731136358047626104977997

Some Sample Results (Tables& Figures)

Table-3a(Refer Section 7.5.1)

(I)Example Solution:Torsion Of An Equilateral Triangular Cross Section(Eight Noded Elements)

Torisonal Constant Values

Mesh No Nodes Elements Fem Sol Exact Sol Absolute Error

----

1 16 3 3.01771523127018 3.11769145362398 0.0326698575267504

2	**49**	**12**	**3.10973432469345**	**3.11769145362398**	**0.00644969161285464**
3	**100**	**27**	**3.11557019336415**	**3.11769145362398**	**0.00209212772860204**
4	**169**	**48**	**3.1167795362198**	**3.11769145362398**	**0.00131858829294726**
5	**256**	**75**	**3.11719193923821**	**3.11769145362398**	**0.000908905458018264**

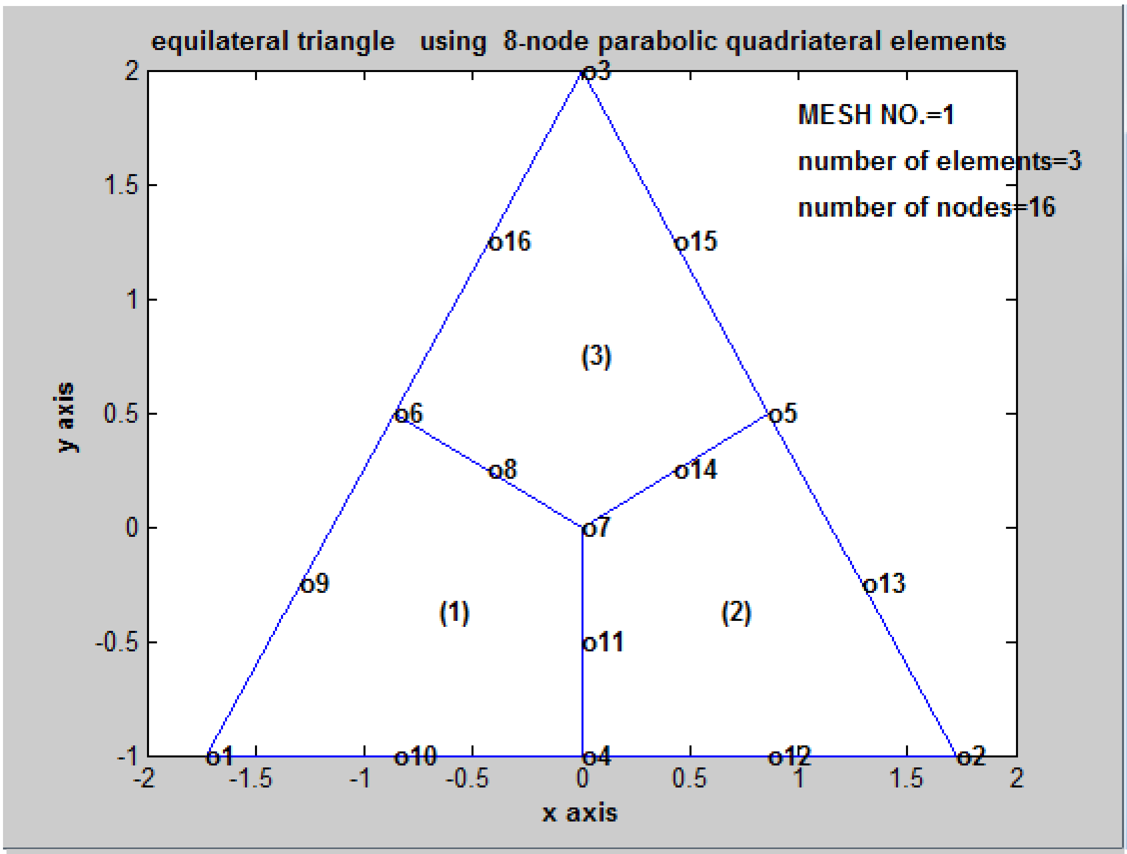
(Ii)Example Solution:Torsion Of An Equilateral Triangular Cross Section(Four Noded Elements)

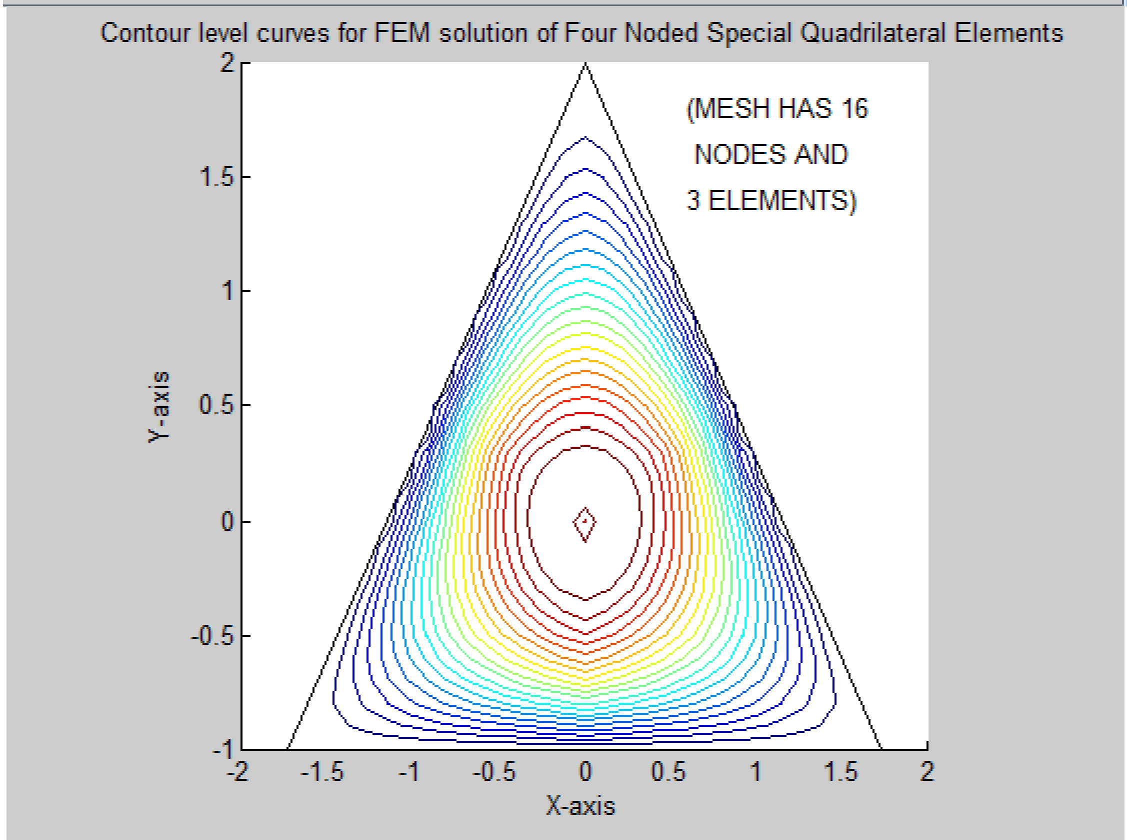
Table-3b

Torisonal Constant Values

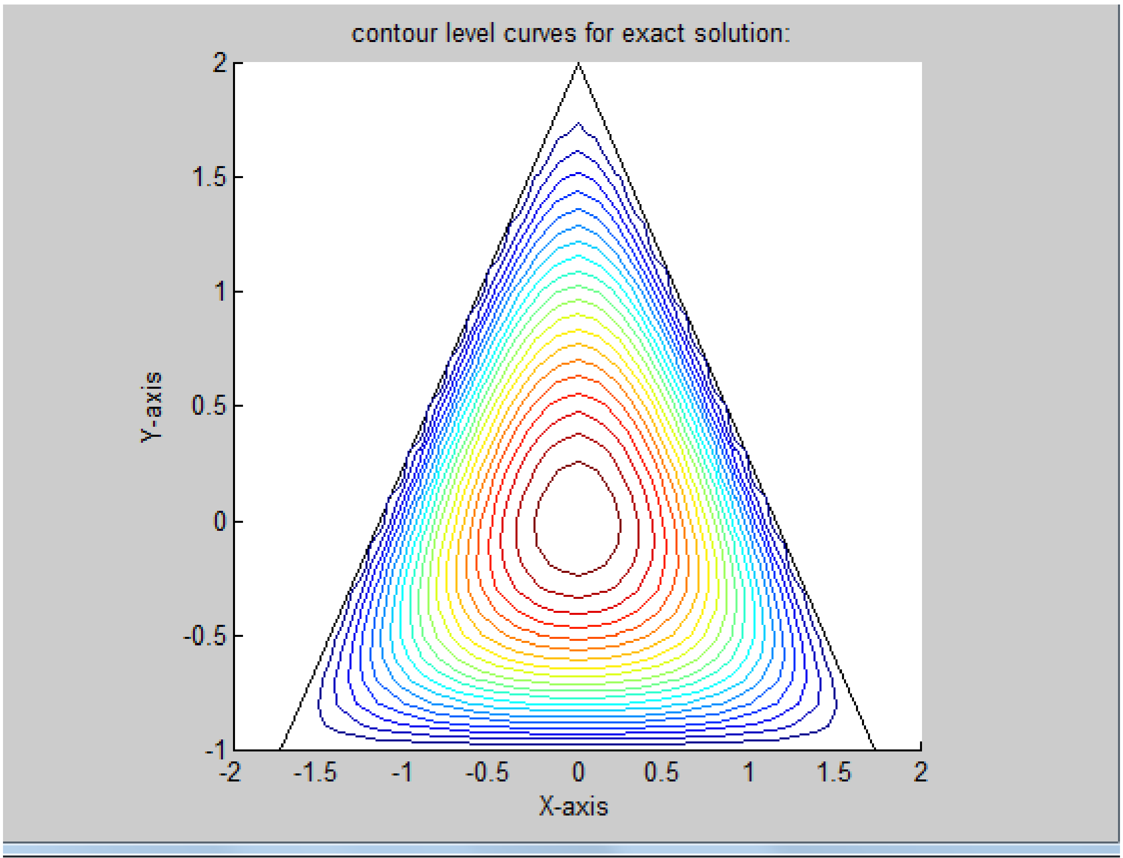
Mesh No.	Nodes	Elements	Fem Sol	Exact Sol	Absolute Error
1	7	3	1.8722884497	3.1176914536	1.2454030039
2	19	12	2.5125294041	3.1176914536	0.6051620495
3	37	27	2.8307089573	3.1176914536	0.2869824963
4	61	48	2.9539197663	3.1176914536	0.1637716873
5	91	75	3.0125356171	3.1176914536	0.1051558365
6	127	108	3.0446697570	3.1176914536	0.0730216966
7	169	147	3.0641026510	3.1176914536	0.0535888026
8	217	192	3.0767211933	3.1176914536	0.0409702604
9	271	243	3.0853673364	3.1176914536	0.0323241172
10	331	300	3.0915454781	3.1176914536	0.0261459755

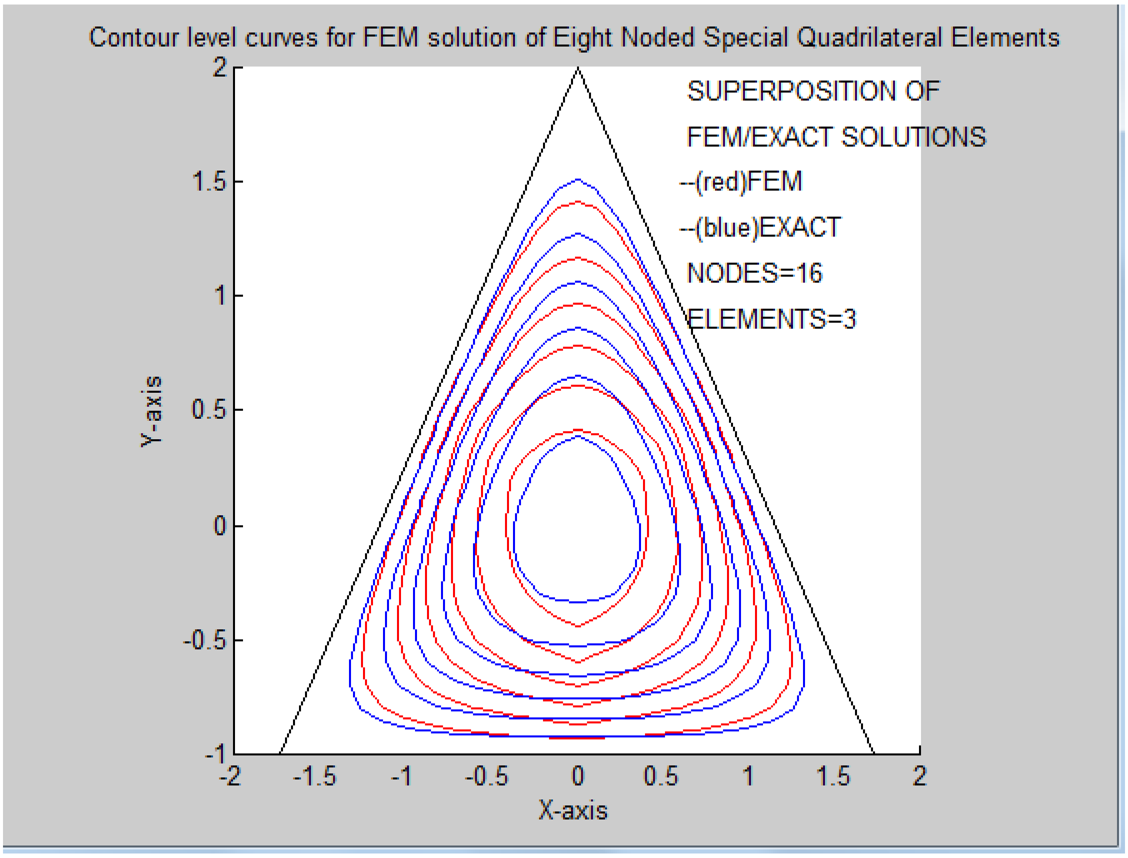
DOI: 10.18535/ijecs/v6i10.13

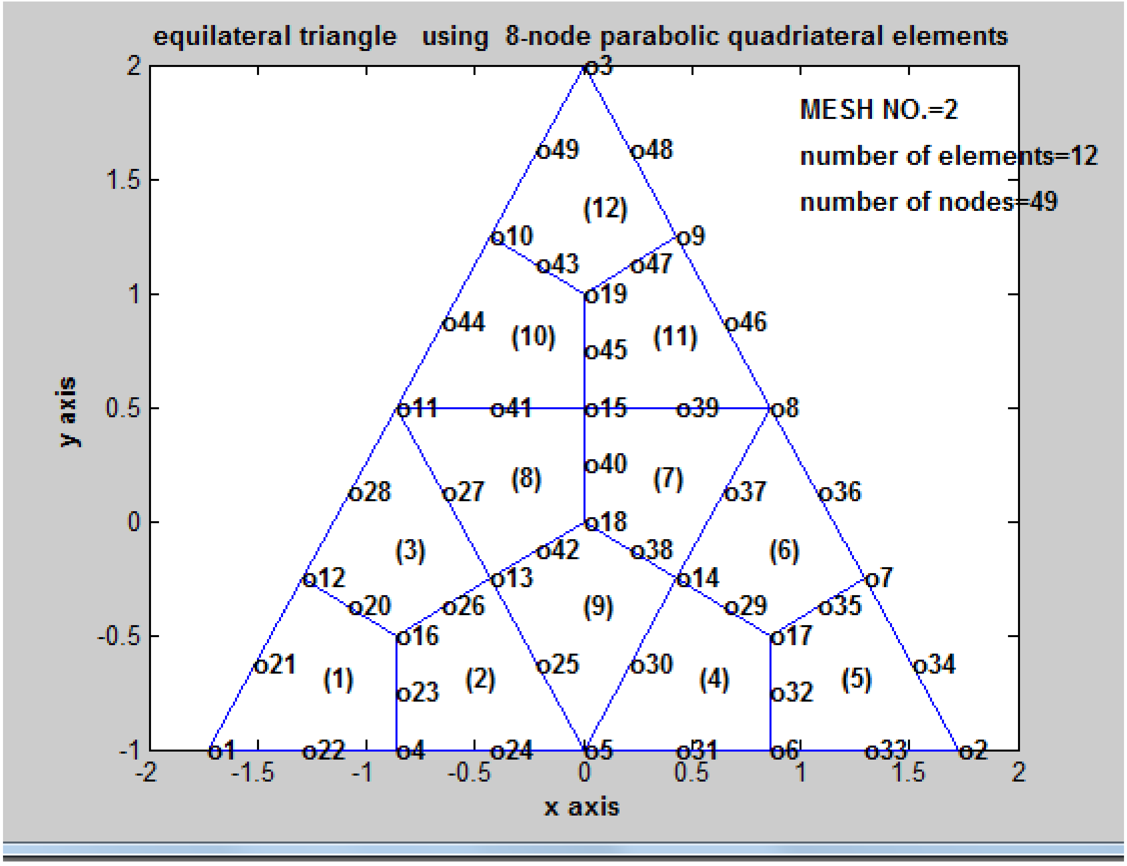


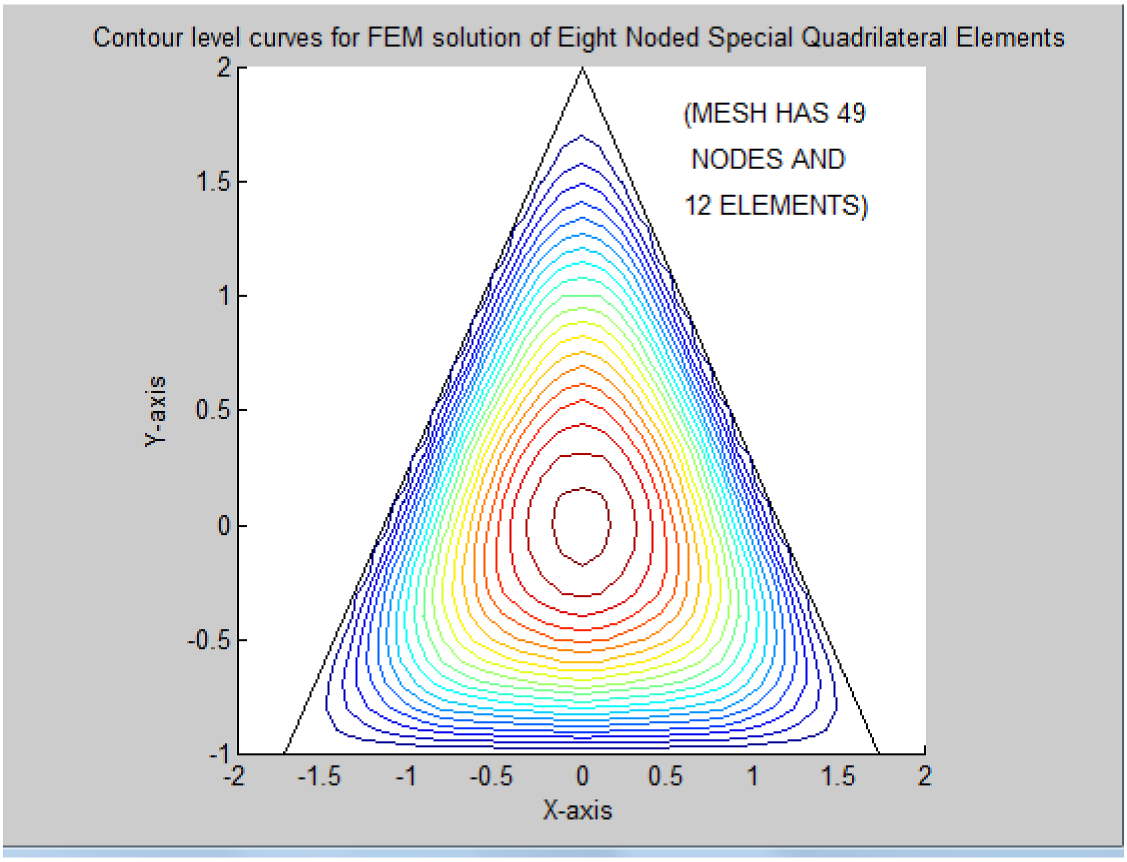


DOI: 10.18535/ijecs/v6i10.13

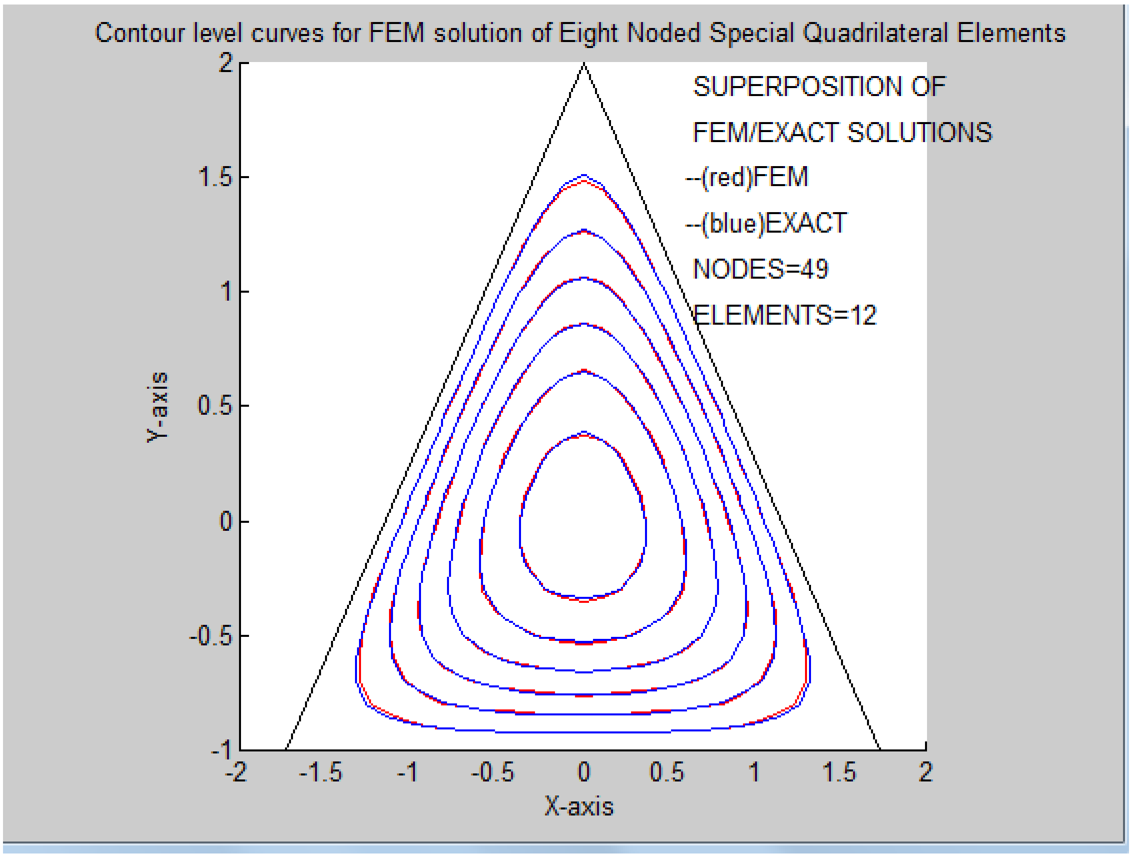





equilateral triangle using 8-node parabolic quadriateral elements
MESH NO.=2
number of elements=12
number of nodes=49
x axis
y axis


Contour level curves for FEM solution of Eight Noded Special Quadrilateral Elements
(MESH HAS 49
NODES AND
12 ELEMENTS)
X-axis
Y-axis

DOI: 10.18535/ijecs/v6i10.13



Solution of Poisson Boundary Value Problems Over Polygonal Domains

Example 1(pentagonal domain,refer section 7.5.2)

TABLE -4a(MESH NO.1- pentagonal domain)

(NUMBER OF NODES=1646,NUMBER OF EIGHT NODE ELEMENTS= 525)
FEM COMPUTED VALUES AND EXACT VALUES AT CENTROID POINTS

node number	fem-computed values	anlytical(theoretical)-values	node number	fem-computed values	anlytical(theoretical)-values
1	0.984054552974264	1	-------------------------	--	
72	0.95731034161468	0.972789205831714	73	0.846731417487447	0.861281226008774
74	0.653454111877992	0.665465038884934	75	0.396534048073442	0.404508497187474
76	0.101677630730026	0.10395584540888	77	0.879296988929729	0.893582297554377
78	0.714694375686168	0.726905328038456	79	0.480400493551786	0.489073800366903
80	0.199511722819656	0.2033683215379	81	0.779093710288116	0.791153573830373
82	0.601743285016065	0.611281226008774	83	0.365711253990309	0.371572412738697
84	0.0940734015934877	0.0954915028125263	85	0.634183627031335	0.643582297554377

86 0.426726726183634 0.433012701892219 87
0.177526027000476 0.180056805991955
88 0.490537978964109 0.497260947684137 89
0.298400440538362 0.302264231633827
90 0.0768480127109979 0.0776797865924606 91
0.330629587069012 0.334565303179429
92 0.137696267998473 0.139120075745983 93
0.201350071842066 0.2033683215379
94 0.0519257119011615 0.0522642316338267 95
0.0840718933767792 0.0845653031794291
96 0.0217104740359334 0.0217326895365599 151
0.958062340925544 0.972789205831713
152 0.779792505027551 0.791153573830373 153
0.490994266704313 0.497260947684137
154 0.201573369158469 0.2033683215379 155
0.0217179814800985 0.0217326895365599
156 0.880492337598912 0.893582297554377 157
0.634753435959794 0.643582297554377
158 0.330687355067233 0.334565303179429 159
0.0839689995966966 0.0845653031794291
160 0.849470299675736 0.861281226008774 161
0.603479320566397 0.611281226008774
162 0.299084415371746 0.302264231633827 163
0.0519780040247773 0.0522642316338269
164 0.717228290055083 0.726905328038456 165
0.427681281779308 0.433012701892219
166 0.137727448010789 0.139120075745983 167
0.657430315402602 0.665465038884933
168 0.367457138957961 0.371572412738697 169
0.0770540478919576 0.0776797865924607

170 0.48324347827485 0.489073800366903 171
0.178006505443163 0.180056805991955
172 0.400453968402124 0.404508497187474 173
0.0946309079026146 0.0954915028125265
174 0.201134241368618 0.2033683215379 175
0.103174472539396 0.10395584540888
230 0.974330203264399 0.989073800366903 231
0.896136875815417 0.908540960039796
232 0.730204658640881 0.739073800366903 233
0.492499922392153 0.497260947684137
234 0.205668506621643 0.2067727288213 235
0.943341717284612 0.956772728821301
236 0.836575836179908 0.847100670886274 237
0.64796706047784 0.654508497187474
238 0.395624589370595 0.397848471555116 239
0.896202291174062 0.908540960039796
240 0.824580694221978 0.834565303179429 241
0.672715423288068 0.678896579685477
242 0.455080745902478 0.4567727288213 243
0.836511569210507 0.847100670886274

244	0.742473159322108	0.75
245	0.57642004569961	0.579484103556456
246	0.730330173970336	0.739073800366903
247	0.672781111938688	0.678896579685477
248	0.55040257053725	0.552264231633827
249	0.647845537697493	0.654508497187474
250	0.576355500766373	0.579484103556456
251	0.492674744680149	0.497260947684137
252	0.455204988799201	0.4567727288213
253	0.395457955117554	0.397848471555116
254	0.205876620106131	0.2067727288213
309	0.957996644729014	0.972789205831713
310	0.849347574939431	0.861281226008774
311	0.657263449074805	0.665465038884933
312	0.400259440479858	0.404508497187474
313	0.103066349242338	0.10395584540888
314	0.880615935579504	0.893582297554377
315	0.717400773295887	0.726905328038456
316	0.48344858244007	0.489073800366903
317	0.201312666942761	0.2033683215379
318	0.779622695286169	0.791153573830373
319	0.603282713460405	0.611281226008774
320	0.36725423531851	0.371572412738697
321	0.0945734156042405	0.0954915028125265
322	0.634955123667848	0.643582297554377
323	0.427898430914042	0.433012701892219
324	0.178199094985638	0.180056805991955
325	0.490785585796504	0.497260947684137
326	0.298895383796607	0.302264231633827
327	0.077019828598804	0.0776797865924607
328	0.330893141181613	0.334565303179429
329	0.137904084749133	0.139120075745983
330	0.20140876017037	0.2033683215379
331	0.0519656373164292	0.0522642316338269
332	0.084102815022196	0.0845653031794291
333	0.0217126313366155	0.0217326895365599
388	0.95736191567378	0.972789205831714
389	0.779263018753169	0.791153573830373
390	0.490746385223315	0.497260947684137
391	0.201514454385377	0.2033683215379
392	0.0217157987086859	0.0217326895365599
393	0.879175636367432	0.893582297554377
394	0.63398177210489	0.643582297554377
395	0.330423473815937	0.334565303179429
396	0.0839379752211733	0.0845653031794291
397	0.84682762714807	0.861281226008774
398	0.601940050576254	0.611281226008774
399	0.298589050686083	0.302264231633827
400	0.0519380058794701	0.0522642316338267
401	0.714528770897719	0.726905328038456
402	0.42650920272216	0.433012701892219
403	0.13751937517853	0.139120075745983

404	0.653585381837678	0.665465038884934
405	0.365913549933321	0.371572412738697
406	0.0768820030756825	0.0776797865924606
407	0.480205987268531	0.489073800366903
408	0.177332396433984	0.180056805991955
409	0.396688343626487	0.404508497187474
410	0.0941295665513121	0.0954915028125263
411	0.199334342441401	0.2033683215379
412	0.101690668171048	0.10395584540888
467	0.956755120877432	0.972789205831713
468	0.846412590528498	0.861281226008774
469	0.653281088801857	0.665465038884934
470	0.396452400769504	0.404508497187474
471	0.101659105849705	0.10395584540888
472	0.878539447955784	0.893582297554377
473	0.714273926615434	0.726905328038456
474	0.480189226582633	0.489073800366903
475	0.199437603227747	0.2033683215379
476	0.777914091404968	0.791153573830373
477	0.601111493756886	0.611281226008774
478	0.365415467830288	0.371572412738697
479	0.0940064898810869	0.0954915028125263
480	0.633255101877495	0.643582297554376
481	0.426266328736289	0.433012701892219
482	0.177365499735863	0.180056805991955
483	0.489654219478838	0.497260947684137
484	0.297993492159467	0.302264231633827
485	0.076756538142967	0.0776797865924605
486	0.33008168650625	0.334565303179429
487	0.137507270526781	0.139120075745983
488	0.200984334238908	0.2033683215379
489	0.0518441477451137	0.0522642316338267
490	0.0839271550236313	0.0845653031794291
491	0.0216715014269189	0.0217326895365599
537	0.956807686230813	0.972789205831713
538	0.778083724714853	0.791153573830373
539	0.489862771705492	0.497260947684137
540	0.201148815295151	0.2033683215379
541	0.0216768381412624	0.0217326895365599
542	0.878418862196617	0.893582297554377
543	0.633053454897349	0.643582297554376
544	0.32987569699646	0.334565303179429
545	0.0837932785515178	0.0845653031794291
546	0.846509421295818	0.861281226008774
547	0.60130849198806	0.611281226008774
548	0.298182207945904	0.302264231633827
549	0.0518564644312021	0.0522642316338267
550	0.714108693190651	0.726905328038456
551	0.426048955054831	0.433012701892219
552	0.13733042935964	0.139120075745983
553	0.65341268182991	0.665465038884934
554	0.365617896757373	0.371572412738697

DOI: 10.18535/ijecs/v6i10.13

555 0.0767905558734775 0.0776797865924605 556
0.479994918028663 0.489073800366903
557 0.177171931041807 0.180056805991955 558
0.396606856093206 0.404508497187474
559 0.0940626879519265 0.0954915028125263 560
0.199260297131279 0.2033683215379
561 0.101672181200666 0.10395584540888

TABLE -4b(MESH NO.2- pentagonal domain)
(NUMBER OF NODES=6441,NUMBER OF EIGHT NODE ELEMENTS= 2400)
FEM COMPUTED VALUES AND EXACT VALUES AT CENTROID POINTS

----

node number	fem-computed values	anlytical(theoretical)-values	node number	fem-computed values	anlytical(theoretical)-values
1	0.996073245850932	1	--		
239	0.908417735923891	0.912293475342785	240	0.833840642781965	0.837521199079693
237	0.989208459901653	0.993158937674856	238	0.960636141837976	0.96460205851448
241	0.738743343730623	0.742126371321759	242	0.625471194936228	0.62845792932566
243	0.496820263354389	0.499314767377287	244	0.355971591464014	0.357876818725374
245	0.20642067360548	0.207626755071376	246	0.0519282823505573	0.0522642316338267
247	0.968885775827469	0.972789205831714	248	0.924624697341631	0.928466175238718
249	0.857607502573432	0.861281226008774	250	0.769485674178439	0.772888674565986
251	0.662431555608889	0.665465038884934	252	0.539086087175029	0.541655445394692
253	0.402495888306543	0.404508497187474	254	0.256042580900918	0.257401207292766
255	0.103365901089715	0.10395584540888	256	0.941092708809802	0.944818029471471
257	0.889963870784639	0.893582297554377	258	0.816936545388237	0.820343603841875
259	0.723809077395573	0.726905328038456	260	0.612876055701065	0.615568230598259
261	0.486872746829154	0.489073800366903	262	0.348908997350752	0.350536750027629
263	0.202393300267115	0.2033683215379	264	0.0509518251403407	0.0511922900311449
265	0.898219332053696	0.90176944487161	266	0.833146027631102	0.836516303737808
267	0.747574539644006	0.750665354967537	268	0.643611963873639	0.646330533842485
269	0.523819814092861	0.526080909926171	270	0.391151620655057	0.392877428045034
271	0.248880184397497	0.25	272	0.100509424800555	0.100966742252535
273	0.849580764343302	0.852868157970561	274	0.779893899730008	0.782966426513139
275	0.691021564435995	0.693785463118374	276	0.585150394583272	0.587521199079693
277	0.464886515127692	0.466790213248601	278	0.333190990012904	0.334565303179429
279	0.19330464732395	0.194102284987398	280	0.0486717020297113	0.048859824350426
281	0.788140401255575	0.791153573830373	282	0.707219417207652	0.709958163014307
283	0.608901149109714	0.611281226008774	284	0.495604089439074	0.497552516492827
285	0.370115783794626	0.371572412738697	286	0.235522604376122	0.236442963004803
287	0.0951270563252322	0.0954915028125263	288	0.723616350425816	0.72631001724706
289	0.641181900806802	0.643582297554377	290	0.542974226146589	0.545007445768716
291	0.43140644951027	0.433012701892219	292	0.309219720544439	0.31035574820837
293	0.179413668467368	0.180056805991955	294	0.0451785107967192	0.0453242676377401
295	0.649414906854648	0.65176944487161	296	0.559157395300991	0.56118014566465
297	0.455141638066962	0.4567727288213	298	0.339921897134717	0.341118051452597
299	0.216326036609548	0.217063915551223	300	0.0873813502312774	0.0876649456551453

DOI: 10.18535/ijecs/v6i10.13

301	0.575517118608351	0.577531999897402	302	0.487388788303937	0.489073800366903
303	0.38726391357419	0.388572980728485	304	0.277598009292733	0.278504204704746
305	0.161078615879269	0.161577730858712	306	0.0405653211981328	0.0406726770331925
307	0.495606728768993	0.497260947684137	308	0.403433274200725	0.404745680624419
309	0.30132213042003	0.302264231633827	310	0.191775072220009	0.192340034102939
311	0.0774712315058161	0.0776797865924606	312	0.419763617731523	0.421097534857171
313	0.3335492484069	0.334565303179429	314	0.239110288272727	0.239794963378827
315	0.138757078426381	0.139120075745983	316	0.0349478355028621	0.0350195901352117
317	0.341753711867582	0.342752450496663	318	0.255270444902929	0.255967663274761
319	0.162478651271768	0.162880102675064	320	0.065643341737255	0.0657818933794382
321	0.271584911586663	0.272319517507514	322	0.194705187309113	0.195181174220666
323	0.112999957126098	0.113236822655327	324	0.0284648212588135	0.0285042047047456
325	0.20290006303692	0.2033683215379	326	0.12915810223945	0.12940952255126
327	0.0521890446534149	0.0522642316338267	328	0.145472957078296	0.145761376784013
329	0.0844392934418026	0.0845653031794291	330	0.0212752995331689	0.0212869511544169
331	0.0926299663544781	0.092752450496663	332	0.0374377147575889	0.0374596510503502
333	0.0537719908914414	0.0538115052831031	334	0.0135542970874276	0.013545542219326
335	0.0217473545150456	0.0217326895365599	336	0.00548580193121442	0.00547059707971809
546	0.989300132912557	0.993158937674856	547	0.941193035584251	0.944818029471471
548	0.849674595896623	0.852868157970561	549	0.723697926556827	0.72631001724706
550	0.575583092822763	0.577531999897402	551	0.41981341338645	0.421097534857172
552	0.271620058984985	0.272319517507513	553	0.145495794384609	0.145761376784013
554	0.0537844981235693	0.053811505283103	555	0.00548617883702121	0.00547059707971812
556	0.96906460486043	0.972789205831713	557	0.898368688235235	0.90176944487161
558	0.788252850007759	0.791153573830373	559	0.64948739371908	0.65176944487161
560	0.495642777447231	0.497260947684137	561	0.341762388343014	0.342752450496663
562	0.202893035033354	0.2033683215379	563	0.0926181291632502	0.092752450496663
564	0.0217388664526258	0.0217326895365599	565	0.961010370197332	0.96460205851448
566	0.890310387880826	0.893582297554377	567	0.780183591367895	0.782966426513139
568	0.641402280910384	0.643582297554377	569	0.48753959420676	0.489073800366903
570	0.333641134525872	0.334565303179429	571	0.194754438012601	0.195181174220666
572	0.084461644738973	0.0845653031794291	573	0.0135556003902938	0.0135455422193261
574	0.925059221777242	0.928466175238718	575	0.833505250243331	0.836516303737808
576	0.707486404737171	0.709958163014308	577	0.55932917720214	0.561180145664649
578	0.403522885120579	0.404745680624419	579	0.255301865025869	0.255967663274761
580	0.129157704822438	0.12940952255126	581	0.0374280451707718	0.0374596510503503
582	0.909050953621805	0.912293475342785	583	0.817487577084702	0.820343603841875
584	0.691452774497855	0.693785463118375	585	0.543275809384325	0.545007445768716
586	0.38744873885157	0.388572980728485	587	0.239207117905952	0.239794963378828
588	0.11304129530476	0.113236822655327	589	0.0212798820129959	0.0212869511544171
590	0.858264942654166	0.861281226008774	591	0.748094501368848	0.750665354967537
592	0.609265317073151	0.611281226008774	593	0.455357566883276	0.456772728821301
594	0.301421596632842	0.302264231633827	595	0.162504693103055	0.162880102675064
596	0.0521829968284905	0.0522642316338269	597	0.834698027200977	0.837521199079693
598	0.724511843036489	0.726905328038456	599	0.58566119475182	0.587521199079693
600	0.431729829507049	0.433012701892219	601	0.277770539746609	0.278504204704746
602	0.138830051877114	0.139120075745983	603	0.0284751178204234	0.0285042047047458
604	0.770320661209626	0.772888674565986	605	0.644233023321532	0.646330533842485
606	0.496002283353717	0.497552516492828	607	0.340127935686681	0.341118051452596
608	0.191847776863326	0.192340034102939	609	0.0656473359221466	0.0657818933794384
610	0.739775640954941	0.742126371321759	611	0.6136648972734	0.615568230598259
612	0.465407002196462	0.466790213248601	613	0.309505900223892	0.31035574820837

DOI: 10.18535/ijecs/v6i10.13

614	0.161200624981383	0.161577730858712	615	0.0349671333987617	0.035019590135212
616	0.66338380409327	0.665465038884933	617	0.524470177339141	0.526080909926171
618	0.370479978017658	0.371572412738697	619	0.216473087229068	0.217063915551224
620	0.0774940152175174	0.0776797865924607	621	0.626611665378016	0.628457929325666
622	0.487666684740177	0.489073800366903	623	0.333644352531367	0.334565303179429
624	0.179609570442655	0.180056805991955	625	0.0405981882581289	0.0406726770331929
626	0.540076896748232	0.541655445394692	627	0.391745762018007	0.392877428045034
628	0.235784061880465	0.236442963004804	629	0.0874352238133717	0.0876649456551455
630	0.497980037877879	0.499314767377287	631	0.349608903930361	0.350536750027629
632	0.193612908539903	0.194102284987398	633	0.0452316534156568	0.0453242676377405
634	0.403422740558437	0.404508497187474	635	0.249319283987149	0.25
636	0.0952307509748933	0.0954915028125265	637	0.357031867890116	0.357876818725374
638	0.202880984584568	0.2033683215379	639	0.0487561069128489	0.0488598243504268
640	0.256769865985167	0.257401207292766	641	0.100695948112821	0.100966742252535
642	0.207216479171139	0.207626755071376	643	0.0510895942232839	0.0511922900311454
644	0.103712117029774	0.10395584540888	645	0.0521859928436363	0.0522642316338272
855	0.993425401825099	0.997260947684137	856	0.973191963685092	0.976807083442103
857	0.928997961256502	0.932300986968877	858	0.861927715076633	0.864838546066896
859	0.773626487913368	0.776080909926171	860	0.666259294468426	0.668213586118192
861	0.542456952909081	0.543892626146237	862	0.405250309017319	0.406179224642423
863	0.257993977804265	0.258464342596354	864	0.104274464136595	0.104385210641588
865	0.98536785615636	0.989073800366903	866	0.957184847067251	0.96063438354617
867	0.905438214423681	0.908540960039796	868	0.831397462159804	0.834076242822669
869	0.736878403029274	0.739073800366903	870	0.624197647393737	0.625872908100787
871	0.496114718074032	0.497260947684137	872	0.355762667524841	0.356404772421034
873	0.206570044433268	0.2067727288213	874	0.973196089867784	0.976807083442103
875	0.953373175300258	0.956772728821301	876	0.910089747943593	0.913179454270281
877	0.84440788858283	0.847100670886274	878	0.757938641322136	0.760163457619103
879	0.652801352359828	0.654508497187474	880	0.531570470877153	0.532737365365924
881	0.397211640489236	0.397848471555116	882	0.252999705143515	0.253163227991246
883	0.957180702996505	0.96063438354617	884	0.929810436754569	0.933012701892219
885	0.879563241851682	0.882417151026054	886	0.807672244452339	0.810093561327006
887	0.715900435525955	0.717822779601698	888	0.606496281281425	0.607876818725374
889	0.482137117876969	0.482962913144534	890	0.34586529024744	0.346156857780064
891	0.929006149304298	0.932300986968877	892	0.910093898902045	0.913179454270281
893	0.868800196467832	0.871572412738697	894	0.806139585309414	0.808504365822488
895	0.723649635515234	0.725528258147577	896	0.623352032393114	0.624687236866834
897	0.507702626575375	0.508464342596354	898	0.37951898661039	0.379721368714746
899	0.905430054266162	0.908540960039796	900	0.879559114807923	0.882417151026054
901	0.832063295086486	0.834565303179429	902	0.764109560018339	0.766163687805082
903	0.677365046943139	0.678896579685477	904	0.573954752016994	0.574912784645444
905	0.456412572774209	0.4567727288213	906	0.861939776426569	0.864838546066896
907	0.844416101435488	0.847100670886274	908	0.806143743212269	0.808504365822488
909	0.748065209780585	0.75	910	0.671607112244574	0.673028145070219
911	0.578645041043814	0.579484103556456	912	0.471442941340965	0.471671240215656
913	0.831385498266943	0.834076242822669	914	0.807664100853283	0.810093561327006
915	0.764105436073146	0.766163687805082	916	0.701782069660152	0.7033683215379
917	0.622224397104039	0.623253692848829	918	0.527387060029069	0.527792489781403
919	0.773642134193344	0.776080909926171	920	0.757950720067617	0.760163457619103
921	0.723657850411695	0.725528258147577	922	0.671611266471013	0.673028145070219
923	0.603093053667256	0.60395584540888	924	0.519772510655331	0.520012148419041
925	0.73686293722567	0.739073800366903	926	0.715888478973598	0.717822779601698

DOI: 10.18535/ijecs/v6i10.13

927	0.677356898629872	0.678896579685477	928	0.622220266518589	0.62325369284882
929	0.551840440585182	0.552264231633827	930	0.666278148194785	0.668213586118192
931	0.652817001516288	0.654508497187474	932	0.62336409802753	0.624687236866834
933	0.578653238934691	0.579484103556456	934	0.519776653700955	0.520012148419041
935	0.624179066040366	0.625872908100787	936	0.6064808043591	0.607876818725374
937	0.573942772236034	0.574912784645444	938	0.52737888971873	0.527792489781403
939	0.542478558359223	0.543892626146237	940	0.531589302964966	0.532737365365924
941	0.507718234317908	0.508464342596354	942	0.471454966660728	0.471671240215656
943	0.49609348553538	0.497260947684137	944	0.482118492790477	0.482962913144534
945	0.456397039697703	0.4567727288213	946	0.405274156606816	0.406179224642423
947	0.397233181959841	0.397848471555116	948	0.379537724961225	0.379721368714746
949	0.355739331542845	0.356404772421034	950	0.345843946086954	0.346156857780064
951	0.258019628885847	0.258464342596354	952	0.253023365120809	0.253163227991246
953	0.206545232286517	0.2067727288213	954	0.104301511024142	0.104385210641588
1164	0.989295921651703	0.993158937674856	1165	0.961002162205927	0.96460205851448
1166	0.909038952245825	0.912293475342785	1167	0.83468253317779	0.837521199079693
1168	0.739757045991632	0.742126371321759	1169	0.626590438792648	0.628457929325666
1170	0.497956713955061	0.499314767377287	1171	0.357007036299771	0.357876818725374
1172	0.207190652154935	0.207626755071376	1173	0.0521716932641899	0.0522642316338272
1174	0.969072726049803	0.972789205831713	1175	0.925071235480818	0.928466175238718
1176	0.858280549230431	0.861281226008774	1177	0.770339486335847	0.772888674565986
1178	0.663405397485869	0.665465038884933	1179	0.54010073916479	0.541655445394692
1180	0.403448248970278	0.404508497187474	1181	0.256796333669271	0.257401207292766
1182	0.103733665835149	0.10395584540888	1183	0.941180971061724	0.944818029471471
1184	0.890294834732729	0.893582297554377	1185	0.817468926931309	0.820343603841875
1186	0.724490571054695	0.726905328038456	1187	0.613641546195002	0.615568230598259
1188	0.487641843036863	0.489073800366903	1189	0.34958316259184	0.350536750027629
1190	0.202854728498343	0.2033683215379	1191	0.0510801986519014	0.0511922900311454
1192	0.898384228951361	0.90176944487161	1193	0.833524009200939	0.836516303737808
1194	0.748116033232795	0.750665354967537	1195	0.644256820133513	0.646330533842485
1196	0.524495674465009	0.526080909926171	1197	0.391772336007349	0.392877428045034
1198	0.249346215533084	0.25	1199	0.100719042841391	0.100966742252535
1200	0.849655865559591	0.852868157970561	1201	0.780162230824065	0.782966426513139
1202	0.691429336169093	0.693785463118375	1203	0.585636288972206	0.587521199079693
1204	0.465381276201418	0.466790213248601	1205	0.33361846474604	0.334565303179429
1206	0.193587450316622	0.194102284987398	1207	0.0487477751849071	0.0488598243504268
1208	0.788274284216081	0.791153573830373	1209	0.707510095656525	0.709958163014308
1210	0.609290718255751	0.611281226008774	1211	0.496028812187687	0.497552516492828
1212	0.370507024879278	0.371572412738697	1213	0.235811000475566	0.236442963004804
1214	0.095253689160531	0.0954915028125265	1215	0.723674357211869	0.72631001724706
1216	0.641377226313847	0.643582297554377	1217	0.543249939612822	0.545007445768716
1218	0.431703846144208	0.433012701892219	1219	0.309480517364245	0.31035574820837
1220	0.179585437164381	0.180056805991955	1221	0.0452243724013542	0.0453242676377405
1222	0.649512627909302	0.65176944487161	1223	0.559355524441261	0.561180145664649
1224	0.455384448850355	0.456772728821301	1225	0.340154775240758	0.341118051452596
1226	0.216499314659671	0.217063915551224	1227	0.087457298092112	0.0876649456551455
1228	0.575556991115876	0.577531999897402	1229	0.487513345821429	0.489073800366903
1230	0.387423087031282	0.388572980728485	1231	0.277746233096526	0.278504204704746
1232	0.161178340234852	0.161577730858712	1233	0.0405921015069459	0.0406726770331929
1234	0.495669361078025	0.497260947684137	1235	0.403549393483726	0.404745680624419
1236	0.301447501658208	0.302264231633827	1237	0.191872577971011	0.192340034102939
1238	0.0775146348273959	0.0776797865924607	1239	0.419787346229175	0.42109753485717

DOI: 10.18535/ijecs/v6i10.13

1240	0.333616350956359	0.334565303179429	1241	0.239184391672041	0.239794963378828
1242	0.13881008943992	0.139120075745983	1243	0.034962367717507	0.035019590135212
1244	0.341787762386894	0.342752450496663	1245	0.255326084974346	0.255967663274761
1246	0.162527360186339	0.162880102675064	1247	0.0656659449234731	0.065781893379438
1248	0.271596606652092	0.272319517507513	1249	0.19473370039583	0.195181174220666
1250	0.113024029304349	0.113236822655327	1251	0.0284717560343298	0.0285042047047458
1252	0.202914795238728	0.2033683215379	1253	0.129177518524575	0.12940952255126
1254	0.0521990551171121	0.0522642316338269	1255	0.145477308639343	0.145761376784013
1256	0.08444728275464	0.0845653031794291	1257	0.0212779370042533	0.0212869511544171
1258	0.0926343061776432	0.092752450496663	1259	0.0374409820706987	0.0374596510503503
1260	0.0537729371171268	0.053811505283103	1261	0.013554928470552	0.0135455422193261
1262	0.0217478467353385	0.0217326895365599	1263	0.00548583638607199	0.00547059707971812
1473	0.989211729460003	0.993158937674856	1474	0.941104656564619	0.944818029471471
1475	0.849599441464903	0.852868157970561	1476	0.72363990544461	0.72631001724706
1477	0.57554322212952	0.577531999897402	1478	0.419789689329642	0.421097534857171
1479	0.271608366803596	0.272319517507514	1480	0.14549144418536	0.145761376784013
1481	0.0537835523649264	0.0538115052831031	1482	0.00548614442865672	0.00547059707971809
1483	0.968877720517919	0.972789205831714	1484	0.898203711354294	0.90176944487161
1485	0.788118937567523	0.791153573830373	1486	0.649389669572493	0.65176944487161
1487	0.495580149610782	0.497260947684137	1488	0.341728341654493	0.342752450496663
1489	0.202878304842337	0.2033683215379	1490	0.092613790164134	0.092752450496663
1491	0.0217383744179899	0.0217326895365599	1492	0.9606425007782	0.96460205851448
1493	0.889979354255774	0.893582297554377	1494	0.779915235446353	0.782966426513139
1495	0.641206955515444	0.643582297554377	1496	0.487415042474032	0.489073800366903
1497	0.333574035934151	0.334565303179429	1498	0.194725926814014	0.195181174220666
1499	0.0844536561610197	0.0845653031794291	1500	0.0135549691224094	0.013545542219326
1501	0.924613015156685	0.928466175238718	1502	0.833127226867596	0.836516303737808
1503	0.707195719983576	0.709958163014307	1504	0.559131054372718	0.56118014566465
1505	0.403406771380652	0.404745680624419	1506	0.255246227667243	0.255967663274761
1507	0.129138289677014	0.12940952255126	1508	0.037424778175094	0.0374596510503502
1509	0.908427046983898	0.912293475342785	1510	0.816955174289703	0.820343603841875
1511	0.691045005048807	0.693785463118374	1512	0.543000106721456	0.545007445768716
1513	0.387289571782249	0.388572980728485	1514	0.239133016924075	0.239794963378827
1515	0.113017224040879	0.113236822655327	1516	0.021277244717292	0.0212869511544169
1517	0.857592480137639	0.861281226008774	1518	0.747553004257123	0.750665354967537
1519	0.608875761393405	0.611281226008774	1520	0.455114766595143	0.4567727288213
1521	0.301296229245677	0.302264231633827	1522	0.162455985470004	0.16288010267506
1523	0.0521729867702466	0.0522642316338267	1524	0.833852673042778	0.837521199079693
1525	0.72383037686367	0.726905328038456	1526	0.585175324778449	0.587521199079693
1527	0.431432446289016	0.433012701892219	1528	0.277622319075679	0.278504204704746
1529	0.138777041614868	0.139120075745983	1530	0.0284681832401898	0.028504204704745
1531	0.76946767334382	0.772888674565986	1532	0.643588199619926	0.646330533842485
1533	0.495577584847013	0.497552516492827	1534	0.33989506437604	0.341118051452597
1535	0.191750271912676	0.192340034102939	1536	0.0656247330500414	0.0657818933794382
1537	0.738757791044872	0.742126371321759	1538	0.612899480321396	0.615568230598259
1539	0.464912274700475	0.466790213248601	1540	0.309245108377733	0.31035574820837
1541	0.161100899885954	0.161577730858712	1542	0.0349526012365779	0.0350195901352117
1543	0.662411009082147	0.665465038884934	1544	0.523794376859607	0.526080909926171
1545	0.370088753936604	0.371572412738697	1546	0.216299807225364	0.217063915551223
1547	0.0774506113919487	0.0776797865924606	1548	0.625487697225502	0.628457929325666
1549	0.486897697204478	0.489073800366903	1550	0.333216893023385	0.334565303179429
1551	0.179437794288198	0.180056805991955	1552	0.0405714071769853	0.0406726770331925

1553	0.539063489891869	0.541655445394692	1554	0.391125107046739	0.392877428045034
1555	0.235495654039938	0.236442963004803	1556	0.0873592714920214	0.0876649456551453
1557	0.496838408951993	0.499314767377287	1558	0.348934843637271	0.350536750027629
1559	0.193330072314947	0.194102284987398	1560	0.0451857870758743	0.0453242676377401
1561	0.40247179173597	0.404508497187474	1562	0.248853229886741	0.25
1563	0.0951040944987861	0.0954915028125263	1564	0.355990939286488	0.357876818725374
1565	0.202419487862486	0.2033683215379	1566	0.0486800077818413	0.0488598243504264
1567	0.256017629045547	0.257401207292766	1568	0.100486195656722	0.100966742252535
1569	0.206440961876334	0.207626755071376	1570	0.0509610481252452	0.0511922900311449
1571	0.103344475352944	0.10395584540888	1572	0.0519304236173952	0.0522642316338267
1782	0.989125235574475	0.993158937674856	1783	0.960572394413234	0.96460205851448
1784	0.908368995068934	0.912293475342785	1785	0.833803936174924	0.837521199079693
1786	0.738716173804794	0.742126371321759	1787	0.625451544480813	0.628457929325666
1788	0.496806574058269	0.499314767377287	1789	0.355962724366778	0.357876818725374
1790	0.206415866145334	0.207626755071376	1791	0.0519271108558334	0.0522642316338267
1792	0.968747282570441	0.972789205831713	1793	0.924518315248889	0.928466175238718
1794	0.857527002599048	0.861281226008774	1795	0.769425741986579	0.772888674565986
1796	0.662387841812248	0.665465038884934	1797	0.539055182312928	0.541655445394692
1798	0.40247525046764	0.404508497187474	1799	0.256030448050792	0.257401207292766
1800	0.103361211943156	0.10395584540888	1801	0.940841604845109	0.944818029471471
1802	0.889772271344953	0.893582297554377	1803	0.816792537094134	0.820343603841875
1804	0.723702663053038	0.726905328038456	1805	0.612799196639984	0.615568230598259
1806	0.486819258338097	0.489073800366903	1807	0.348874374111794	0.350536750027629
1808	0.202374533375447	0.2033683215379	1809	0.0509472508245047	0.0511922900311449
1810	0.897959438568805	0.90176944487161	1811	0.832949936176764	0.836516303737808
1812	0.747428916226336	0.750665354967537	1813	0.643505966563885	0.646330533842485
1814	0.523744995535456	0.526080909926171	1815	0.391101715085857	0.392877428045034
1816	0.248850865471984	0.25	1817	0.100498093959087	0.100966742252535
1818	0.849258257339191	0.852868157970561	1819	0.779652588262356	0.782966426513139
1820	0.690843921864592	0.693785463118374	1821	0.585022478551723	0.587521199079693
1822	0.464797699268288	0.466790213248601	1823	0.333133592263342	0.334565303179429
1824	0.193273566063226	0.194102284987398	1825	0.0486641274700553	0.0488598243504264
1826	0.787842068345843	0.791153573830373	1827	0.70699892418536	0.709958163014307
1828	0.608741276885113	0.611281226008774	1829	0.495491580310044	0.497552516492827
1830	0.370040900130993	0.371572412738697	1831	0.235478672761962	0.236442963004803
1832	0.095110088846801	0.0954915028125263	1833	0.723295255236095	0.72631001724706
1834	0.640946923212312	0.643582297554376	1835	0.542805814433453	0.545007445768716
1836	0.431289925824229	0.433012701892219	1837	0.309144601029561	0.31035574820837
1838	0.179373051818445	0.180056805991955	1839	0.0451686178275864	0.0453242676377401
1840	0.649136711033313	0.65176944487161	1841	0.558956828247415	0.56118014566465
1842	0.455001097583764	0.4567727288213	1843	0.339828644427941	0.341118051452596
1844	0.216271437568563	0.217063915551223	1845	0.0873602833744188	0.0876649456551453
1846	0.575244252521064	0.577531999897402	1847	0.487194441204915	0.489073800366903
1848	0.387130061900311	0.388572980728485	1849	0.277511991901579	0.278504204704746
1850	0.161032196543329	0.161577730858712	1851	0.0405540235750894	0.0406726770331925
1852	0.495382927293562	0.497260947684137	1853	0.403277311987097	0.404745680624419
1854	0.30121904305439	0.302264231633827	1855	0.191714863651767	0.192340034102939
1856	0.0774480287184385	0.0776797865924605	1857	0.419561163600188	0.421097534857172
1858	0.333410572185011	0.334565303179429	1859	0.239021498410448	0.239794963378828
1860	0.138709269169698	0.139120075745983	1861	0.0349362100778989	0.0350195901352117
1862	0.341597096876498	0.342752450496663	1863	0.255167378097686	0.255967663274761

1864	0.162418620220194	0.162880102675064	1865	0.0656202388756059	0.0657818933794383
1866	0.271455083410393	0.272319517507513	1867	0.194622386230989	0.195181174220666
1868	0.112955475474922	0.113236822655327	1869	0.0284540149895751	0.0285042047047456
1870	0.202807348485452	0.2033683215379	1871	0.129104250932692	0.12940952255126
1872	0.0521683481458166	0.0522642316338267	1873	0.145404683881683	0.145761376784013
1874	0.0844026949709551	0.0845653031794291	1875	0.0212664156344509	0.0212869511544169
1876	0.0925878889680897	0.0927524504966629	1877	0.0374215622100085	0.0374596510503502
1878	0.0537471507052501	0.053811505283103	1879	0.0135482705154839	0.013545542219326
1880	0.0217374633260827	0.0217326895365599	1881	0.00548328212804241	0.00547059707971807
2072	0.989128589145629	0.993158937674856	2073	0.940853610363781	0.944818029471471
2074	0.849276962271101	0.852868157970561	2075	0.723318819286489	0.72631001724706
2076	0.57527035666783	0.577531999897402	2077	0.41958723379089	0.421097534857172
2078	0.271478537491089	0.272319517507513	2079	0.145423170396999	0.145761376784013
2080	0.0537587119626278	0.053811505283103	2081	0.005483624602861	0.00547059707971807
2082	0.968739307420798	0.972789205831713	2083	0.897943859329832	0.90176944487161
2084	0.787820622784073	0.791153573830373	2085	0.649111478926859	0.65176944487161
2086	0.49535634824281	0.497260947684137	2087	0.341571725767188	0.342752450496663
2088	0.202785589617079	0.2033683215379	2089	0.0925717124478797	0.0927524504966629
2090	0.0217284831451163	0.0217326895365599	2091	0.9605788221136	0.96460205851448
2092	0.889787794922868	0.893582297554377	2093	0.779673943546372	0.782966426513139
2094	0.640971985295444	0.643582297554376	2095	0.487220697164546	0.489073800366903
2096	0.333435359696516	0.334565303179429	2097	0.194643125409927	0.195181174220666
2098	0.0844170574701913	0.0845653031794291	2099	0.013548942504203	0.013545542219326
2100	0.924506684421635	0.928466175238718	2101	0.832931163968967	0.836516303737808
2102	0.706975240145537	0.709958163014307	2103	0.558930492044804	0.56118014566465
2104	0.403250810286945	0.404745680624419	2105	0.255143160852517	0.255967663274761
2106	0.129084438169633	0.12940952255126	2107	0.0374086255271474	0.0374596510503502
2108	0.908378352409231	0.912293475342785	2109	0.816811193511579	0.820343603841875
2110	0.690867376454651	0.693785463118374	2111	0.542831700924955	0.545007445768716
2112	0.387155722098774	0.388572980728485	2113	0.23904422749213	0.239794963378828
2114	0.112972742354387	0.113236822655327	2115	0.0212683607765631	0.0212869511544169
2116	0.857512014979638	0.861281226008774	2117	0.747407400673161	0.750665354967537
2118	0.608715898844536	0.611281226008774	2119	0.454974230067344	0.4567727288213
2120	0.301193143147616	0.302264231633827	2121	0.162395954642276	0.162880102675064
2122	0.052152290225835	0.0522642316338267	2123	0.83381599789221	0.837521199079693
2124	0.723723981549777	0.726905328038456	2125	0.585047418753757	0.587521199079693
2126	0.431315927114766	0.433012701892219	2127	0.277536303332737	0.278504204704746
2128	0.138729232743328	0.139120075745983	2129	0.0284573769781879	0.0285042047047456
2130	0.769407764986575	0.772888674565986	2131	0.643482216155563	0.646330533842485
2132	0.49546508270919	0.497552516492827	2133	0.339801814648786	0.341118051452596
2134	0.191690064297704	0.192340034102939	2135	0.0656016303222009	0.0657818933794383
2136	0.738730642657496	0.742126371321759	2137	0.61282263442709	0.615568230598259
2138	0.464823465849667	0.466790213248601	2139	0.309169991996609	0.31035574820837
2140	0.161054481571364	0.161577730858712	2141	0.03494097592256	0.0350195901352117
2142	0.662367311655548	0.665465038884934	2143	0.523719567865343	0.526080909926171
2144	0.370013875065274	0.371572412738697	2145	0.216245210080221	0.217063915551223
2146	0.0774274090166313	0.0776797865924605	2147	0.625468061513703	0.628457929325666
2148	0.486844217658188	0.489073800366903	2149	0.333159499867707	0.334565303179429
2150	0.179397179419196	0.180056805991955	2151	0.0405601098176238	0.0406726770331925
2152	0.539032596156848	0.541655445394692	2153	0.391075207804657	0.392877428045034
2154	0.235451725307476	0.236442963004803	2155	0.0873382053994804	0.0876649456551453
2156	0.496824729569446	0.499314767377287	2157	0.348900226123777	0.350536750027629

2158	0.193298993573824	0.194102284987398	2159	0.0451758945504923	0.0453242676377401
2160	0.402451161158745	0.404508497187474	2161	0.248823914674774	0.25
2162	0.0950871281500973	0.0954915028125263	2163	0.355982078481331	0.357876818725374
2164	0.202400724062353	0.2033683215379	2165	0.0486724338407089	0.0488598243504264
2166	0.256005500415231	0.257401207292766	2167	0.100474866252417	0.100966742252535
2168	0.206436157793976	0.207626755071376	2169	0.0509564745628248	0.0511922900311449

Example 2 (square domain,refer section 7.5.2)

TABLE -5a(MESH NO.1- square domain,)

(NUMBER OF NODES=1881,NUMBER OF EIGHT NODE ELEMENTS= 600)

FEM COMPUTED VALUES AND EXACT VALUES AT CENTROID POINTS

node number	fem-computed values	anlytical(theoretical)-values	node number	fem-computed values	anlytical(theoretical)-values
1	0.979379623412904	1	------	---------------	----- ---------------
73	0.953168754967672	0.972789205831714	74	0.843808246439099	0.861281226008774
75	0.651552462747831	0.665465038884934	76	0.395502746695546	0.404508497187474
77	0.101426236899791	0.10395584540888	78	0.875831211406802	0.893582297554377
79	0.712391304414235	0.726905328038456	80	0.479056212422448	0.489073800366903
81	0.198994493036731	0.2033683215379	82	0.775870193315283	0.791153573830373
83	0.599718795975148	0.611281226008774	84	0.364637620983079	0.371572412738697
85	0.0938142916450892	0.0954915028125263	86	0.631794329679959	0.643582297554377
87	0.425371010888716	0.433012701892219	88	0.177011946217157	0.180056805991955
89	0.488638506349959	0.497260947684137	90	0.297418869813156	0.302264231633827
91	0.0766137128681347	0.0776797865924606	92	0.329450576590961	0.334565303179429
93	0.137256222029278	0.139120075745983	94	0.200624116722613	0.2033683215379
95	0.0517542364697151	0.0522642316338267	96	0.0837831649318304	0.0845653031794291
97	0.0216354596436645	0.0217326895365599	152	0.953221209301514	0.972789205831713
153	0.776039794282749	0.791153573830373	154	0.488847050224056	0.497260947684137

DOI: 10.18535/ijecs/v6i10.13

155 0.200788596153649 0.2033683215379 156
0.021640796299794 0.0217326895365599
157 0.875710498596284 0.893582297554377 158
0.631592650916028 0.643582297554377
159 0.329244579423693 0.334565303179429 160
0.0836492876263726 0.0845653031794291
161 0.843904853873921 0.861281226008774 162
0.59991573651432 0.611281226008774
163 0.297607571857853 0.302264231633827 164
0.0517665520937256 0.0522642316338269
165 0.712225944582529 0.726905328038456 166
0.425153603034903 0.433012701892219
167 0.137079375402575 0.139120075745983 168
0.65168391660508 0.665465038884933
169 0.364840010786735 0.371572412738697 170
0.0766477263052678 0.0776797865924607
171 0.478861826537811 0.489073800366903 172
0.176818361823277 0.180056805991955
173 0.395657130315354 0.404508497187474 174
0.0938704795092666 0.0954915028125265
175 0.198817156894259 0.2033683215379 176
0.101439295069483 0.10395584540888
231 0.953168754967675 0.972789205831714 232
0.843808246439106 0.861281226008774
233 0.651552462747833 0.665465038884934 234
0.395502746695548 0.404508497187474
235 0.101426236899792 0.10395584540888 236
0.875831211406811 0.893582297554377
237 0.712391304414235 0.726905328038456 238
0.47905621242245 0.489073800366903
239 0.198994493036732 0.2033683215379 240
0.77587019331529 0.791153573830373
241 0.59971879597515 0.611281226008774 242
0.36463762098308 0.371572412738697
243 0.0938142916450894 0.0954915028125265 244
0.631794329679962 0.643582297554377
245 0.425371010888719 0.433012701892219 246
0.177011946217157 0.180056805991955
247 0.488638506349962 0.497260947684137 248
0.297418869813156 0.302264231633827
249 0.0766137128681354 0.0776797865924608 250
0.329450576590962 0.334565303179429
251 0.137256222029279 0.139120075745983 252
0.200624116722615 0.2033683215379
253 0.0517542364697154 0.0522642316338269 254
0.0837831649318312 0.0845653031794291
255 0.0216354596436647 0.0217326895365599 310
0.953221209301516 0.972789205831714
311 0.776039794282758 0.791153573830373 312
0.488847050224059 0.497260947684137
313 0.20078859615365 0.2033683215379 314
0.0216407962997942 0.0217326895365599

315	0.875710498596292	0.893582297554377
316	0.631592650916031	0.643582297554377
317	0.329244579423695	0.334565303179429
318	0.0836492876263732	0.0845653031794291
319	0.843904853873924	0.861281226008774
320	0.599915736514326	0.611281226008774
321	0.297607571857856	0.302264231633827
322	0.051766552093726	0.0522642316338269
323	0.712225944582535	0.726905328038456
324	0.425153603034907	0.433012701892219
325	0.137079375402576	0.139120075745983
326	0.651683916605085	0.665465038884934
327	0.36484001078674	0.371572412738697
328	0.0766477263052687	0.0776797865924608
329	0.478861826537817	0.489073800366903
330	0.17681836182328	0.180056805991955
331	0.395657130315359	0.404508497187474
332	0.0938704795092679	0.0954915028125265
333	0.198817156894261	0.2033683215379
334	0.101439295069485	0.10395584540888
389	0.953168754967675	0.972789205831713
390	0.843808246439108	0.861281226008774
391	0.651552462747837	0.665465038884933
392	0.395502746695552	0.404508497187474
393	0.101426236899793	0.10395584540888
394	0.875831211406811	0.893582297554377
395	0.712391304414239	0.726905328038456
396	0.479056212422455	0.489073800366903
397	0.198994493036733	0.2033683215379
398	0.775870193315288	0.791153573830373
399	0.59971879597515	0.611281226008774
400	0.364637620983081	0.371572412738697
401	0.0938142916450899	0.0954915028125265
402	0.631794329679962	0.643582297554377
403	0.425371010888721	0.433012701892219
404	0.177011946217158	0.180056805991955
405	0.488638506349961	0.497260947684137
406	0.297418869813159	0.302264231633827
407	0.0766137128681356	0.0776797865924607
408	0.329450576590963	0.334565303179429
409	0.13725622202928	0.139120075745983
410	0.200624116722614	0.2033683215379
411	0.0517542364697155	0.0522642316338269
412	0.0837831649318312	0.0845653031794291
413	0.0216354596436646	0.0217326895365599
468	0.953221209301516	0.972789205831714
469	0.776039794282756	0.791153573830373
470	0.488847050224057	0.497260947684137
471	0.200788596153649	0.2033683215379
472	0.021640796299794	0.0217326895365599
473	0.875710498596286	0.893582297554377
474	0.631592650916034	0.643582297554377

475	0.329244579423692	0.334565303179429
476	0.0836492876263729	0.0845653031794291
477	0.843904853873918	0.861281226008774
478	0.599915736514325	0.611281226008774
479	0.297607571857853	0.302264231633827
480	0.0517665520937258	0.0522642316338267
481	0.712225944582531	0.726905328038456
482	0.425153603034905	0.433012701892219
483	0.137079375402575	0.139120075745983
484	0.651683916605081	0.665465038884934
485	0.364840010786737	0.371572412738697
486	0.0766477263052679	0.0776797865924606
487	0.478861826537814	0.489073800366903
488	0.176818361823278	0.180056805991955
489	0.395657130315356	0.404508497187474
490	0.0938704795092668	0.0954915028125263
491	0.198817156894259	0.2033683215379
492	0.101439295069484	0.10395584540888
547	0.95316875496767	0.972789205831713
548	0.8438082464391	0.861281226008774
549	0.651552462747833	0.665465038884934
550	0.395502746695548	0.404508497187474
551	0.101426236899791	0.10395584540888
552	0.875831211406803	0.893582297554377
553	0.712391304414236	0.726905328038456
554	0.47905621242245	0.489073800366903
555	0.198994493036731	0.2033683215379
556	0.775870193315277	0.791153573830373
557	0.599718795975149	0.611281226008774
558	0.364637620983078	0.371572412738697
559	0.0938142916450888	0.0954915028125263
560	0.63179432967996	0.643582297554376
561	0.425371010888716	0.433012701892219
562	0.177011946217157	0.180056805991955
563	0.48863850634996	0.497260947684137
564	0.297418869813156	0.302264231633827
565	0.0766137128681352	0.0776797865924605
566	0.329450576590961	0.334565303179429
567	0.137256222029279	0.139120075745983
568	0.200624116722614	0.2033683215379
569	0.0517542364697153	0.0522642316338267
570	0.0837831649318308	0.0845653031794291
571	0.0216354596436645	0.0217326895365599
617	0.953221209301512	0.972789205831713
618	0.776039794282745	0.791153573830373
619	0.488847050224056	0.497260947684137
620	0.200788596153649	0.2033683215379
621	0.021640796299794	0.0217326895365599
622	0.875710498596279	0.893582297554377
623	0.631592650916026	0.643582297554376
624	0.329244579423691	0.334565303179429
625	0.0836492876263728	0.0845653031794291

626	0.843904853873911	0.861281226008774	627	0.599915736514321	0.611281226008774
628	0.297607571857851	0.302264231633827	629	0.0517665520937256	0.0522642316338267
630	0.712225944582527	0.726905328038456	631	0.425153603034902	0.433012701892219
632	0.137079375402574	0.139120075745983	633	0.651683916605079	0.665465038884934
634	0.364840010786733	0.371572412738697	635	0.0766477263052677	0.0776797865924605
636	0.478861826537811	0.489073800366903	637	0.176818361823277	0.180056805991955
638	0.395657130315352	0.404508497187474	639	0.0938704795092664	0.0954915028125263
640	0.198817156894257	0.2033683215379	641	0.101439295069483	0.10395584540888

TABLE -5b(MESH NO.2- square domain,)

(NUMBER OF NODES=7361,NUMBER OF EIGHT NODE ELEMENTS= 2400)

FEM COMPUTED VALUES AND EXACT VALUES AT CENTROID POINTS

----

node number	fem-computed values	anlytical(theoretical)-values	node number	fem-computed values	anlytical(theoretical)-values
1	0.994930801992269	1	-------	-------------	------------------------
238	0.98811195239712	0.993158937674856	239	0.959693109629146	0.96460205851448
240	0.907618249217412	0.912293475342785	241	0.833174876762085	0.837521199079693
242	0.738201418926509	0.742126371321759	243	0.625043940727546	0.628457929325666
244	0.496499762060679	0.499314767377287	245	0.355751523594188	0.357876818725374
246	0.206296519449992	0.207626755071376	247	0.0518974284783571	0.0522642316338267
248	0.967859476875116	0.972789205831714	249	0.923753282722191	0.928466175238718
250	0.85687968124559	0.861281226008774	251	0.768890023972355	0.772888674565986
252	0.661957407796414	0.665465038884934	253	0.538724138989495	0.541655445394692

DOI: 10.18535/ijecs/v6i10.13

254	0.40223858204869	0.404508497187474
255	0.255884317139615	0.257401207292766
256	0.103303146785268	0.10395584540888
257	0.940062060623085	0.944818029471471
258	0.889100036180271	0.893582297554377
259	0.816224435736178	0.820343603841875
260	0.723234413206202	0.726905328038456
261	0.612426198171119	0.615568230598259
262	0.486537168624728	0.489073800366903
263	0.348679516809266	0.350536750027629
264	0.202264174994639	0.2033683215379
265	0.0509197744881523	0.0511922900311449
266	0.897280859664448	0.90176944487161
267	0.832370766797058	0.836516303737808
268	0.746946046194954	0.750665354967537
269	0.643115599277149	0.646330533842485
270	0.523443280329412	0.526080909926171
271	0.390885205766652	0.392877428045034
272	0.248716840359405	0.25
273	0.100444765598868	0.100966742252535
274	0.848667889315267	0.852868157970561
275	0.779149859343825	0.782966426513139
276	0.690426874411857	0.693785463118374
277	0.584688497241742	0.587521199079693
278	0.464544055153659	0.466790213248601
279	0.332957841517689	0.334565303179429
280	0.193173829502669	0.194102284987398
281	0.0486392724669075	0.0488598243504264
282	0.787334302303178	0.791153573830373
283	0.706572803040855	0.709958163014307
284	0.608394902680416	0.611281226008774
285	0.495222692133831	0.497552516492827
286	0.369847306037521	0.371572412738697
287	0.235358560975429	0.236442963004803
288	0.0950622380280101	0.0954915028125263
289	0.722859975706579	0.72631001724706
290	0.640583783242233	0.643582297554377
291	0.54251364498578	0.545007445768716
292	0.431067217149918	0.433012701892219
293	0.308989865793076	0.31035574820837
294	0.179285086767194	0.180056805991955
295	0.0451466782671521	0.0453242676377401
296	0.648769331976662	0.65176944487161
297	0.558656779341739	0.56118014566465
298	0.454767279687949	0.4567727288213
299	0.339659811908236	0.341118051452597
300	0.216166481077009	0.217063915551223
301	0.0873184253419778	0.0876649456551453
302	0.574936268536357	0.577531999897402
303	0.486945653440633	0.489073800366903
304	0.386939819217785	0.388572980728485
305	0.277379504052058	0.278504204704746

DOI: 10.18535/ijecs/v6i10.13

306 0.160956764557052 0.161577730858712 307
0.0405351964003543 0.0406726770331925
308 0.495130366085337 0.497260947684137 309
0.403079856789113 0.404745680624419
310 0.301076113717553 0.302264231633827 311
0.191625859630585 0.192340034102939
312 0.077412500020055 0.0776797865924606 313
0.419356460499053 0.421097534857171
314 0.333253623917117 0.334565303179429 315
0.238911982670303 0.239794963378827
316 0.138646836869418 0.139120075745983 317
0.0349206179357372 0.0350195901352117
318 0.341436624799222 0.342752450496663 319
0.255050980782898 0.255967663274761
320 0.16234603611647 0.162880102675064 321
0.0655912423031192 0.0657818933794382
322 0.271331783729484 0.272319517507514 323
0.194536213508076 0.195181174220666
324 0.112906299714679 0.113236822655327 325
0.0284417272330714 0.0285042047047456
326 0.202717768134339 0.2033683215379 327
0.129048327049509 0.12940952255126
328 0.0521459930975743 0.0522642316338267 329
0.145342001507621 0.145761376784013
330 0.0843668958826119 0.0845653031794291 331
0.0212574664184982 0.0212869511544169
332 0.0925487243774249 0.092752450496663 333
0.0374058986914838 0.0374596510503502
334 0.0537247702828451 0.0538115052831031 335
0.0135426733705527 0.013545542219326
336 0.0217285086193259 0.0217326895365599 337
0.00548104088740238 0.00547059707971809
547 0.988115294680126 0.993158937674856 548
0.940074062258249 0.944818029471471
549 0.84868659277686 0.852868157970561 550
0.72288353909615 0.72631001724706
551 0.574962372363255 0.577531999897402 552
0.419382530535584 0.421097534857172
553 0.271355237742225 0.272319517507513 554
0.145360487998849 0.145761376784013
555 0.0537363315350633 0.053811505283103 556
0.00548138336203361 0.00547059707971812
557 0.967851488243857 0.972789205831713 558
0.897265275884595 0.90176944487161
559 0.787312854827525 0.791153573830373 560
0.648744098964744 0.65176944487161
561 0.495103786592424 0.497260947684137 562
0.341411253483791 0.342752450496663
563 0.202696009183127 0.2033683215379 564
0.0925325478334607 0.092752450496663
565 0.0217195284356808 0.0217326895365599 566
0.959699512817682 0.96460205851448

567 0.889115551219371 0.893582297554377 568
0.779171210947253 0.782966426513139
569 0.640608843566406 0.643582297554377 570
0.486971908538221 0.489073800366903
571 0.333278411027673 0.334565303179429 572
0.194556952527875 0.195181174220666
573 0.0843812583382638 0.0845653031794291 574
0.0135433453558384 0.0135455422193261
575 0.923741635544387 0.928466175238718 576
0.832351987969175 0.836516303737808
577 0.706549115928692 0.709958163014308 578
0.558630441631063 0.561180145664649
579 0.403053354360953 0.404745680624419 580
0.255026763221411 0.255967663274761
581 0.12902851418015 0.12940952255126 582
0.0373929619908702 0.0374596510503503
583 0.9076275870862 0.912293475342785 584
0.816243083740822 0.820343603841875
585 0.69045032498364 0.693785463118375 586
0.542539529480189 0.545007445768716
587 0.386965478451488 0.388572980728485 588
0.238934711339352 0.239794963378828
589 0.112923566463732 0.113236822655327 590
0.0212594115463316 0.0212869511544171
591 0.856864680252519 0.861281226008774 592
0.746924524372397 0.750665354967537
593 0.608369521526102 0.611281226008774 594
0.454740410625833 0.456772728821301
595 0.301050213101246 0.302264231633827 596
0.162323370274809 0.162880102675064
597 0.0521299351236674 0.0522642316338269 598
0.833186924364233 0.837521199079693
599 0.723255724719739 0.726905328038456 600
0.584713433885384 0.587521199079693
601 0.431093216660732 0.433012701892219 602
0.277403814677302 0.278504204704746

603 0.138666800162288 0.139120075745983 604
0.0284450891846639 0.0285042047047458
605 0.768872036915708 0.772888674565986 606
0.643091843697678 0.646330533842485
607 0.49519619188016 0.497552516492828 608
0.33963298085304 0.341118051452596
609 0.191601059765426 0.192340034102939 610
0.0655726336307235 0.0657818933794384
611 0.738215877759054 0.742126371321759 612
0.612449630587231 0.615568230598259
613 0.464569818932347 0.466790213248601 614
0.309015255423265 0.31035574820837
615 0.160979049082545 0.161577730858712 616
0.0349253837052765 0.035019590135212

DOI: 10.18535/ijecs/v6i10.13

617 0.661936870335342 0.665465038884933 618
0.523417848722286 0.526080909926171
619 0.369820278982843 0.371572412738697 620
0.216140252739392 0.217063915551224
621 0.0773918801004648 0.0776797865924607 622
0.625060450744221 0.628457929325666
623 0.486562124055127 0.489073800366903 624
0.332983747159406 0.334565303179429
625 0.179309213578206 0.180056805991955 626
0.040541282512786 0.0406726770331929
627 0.538701547691154 0.541655445394692 628
0.390858695726015 0.392877428045034
629 0.235331612265815 0.236442963004804 630
0.0872963470177792 0.0876649456551455
631 0.49651791278512 0.499314767377287 632
0.34870536623425 0.350536750027629
633 0.193199255898102 0.194102284987398 634
0.0451539547900752 0.0453242676377405
635 0.402214489309541 0.404508497187474 636
0.24868988788863 0.25
637 0.0950392768281445 0.0954915028125265 638
0.355770874642917 0.357876818725374
639 0.202290364254583 0.2033683215379 640
0.0486475785598044 0.0488598243504268
641 0.255859367484794 0.257401207292766 642
0.100421537234303 0.100966742252535
643 0.206316809444234 0.207626755071376 644
0.0509289978755852 0.0511922900311454
645 0.103281721890494 0.10395584540888 646
0.0518995701631075 0.0522642316338272
856 0.988111952397117 0.993158937674856 857
0.95969310962914 0.96460205851448
858 0.907618249217404 0.912293475342785 859
0.833174876762066 0.837521199079693
860 0.738201418926487 0.742126371321759 861
0.625043940727524 0.628457929325666
862 0.496499762060656 0.499314767377287 863
0.355751523594173 0.357876818725374
864 0.206296519449986 0.207626755071376 865
0.0518974284783556 0.0522642316338272
866 0.967859476875114 0.972789205831714 867
0.923753282722182 0.928466175238718
868 0.856879681245573 0.861281226008774 869
0.76889002397233 0.772888674565986
870 0.661957407796391 0.665465038884934 871
0.538724138989473 0.541655445394692
872 0.402238582048674 0.404508497187474 873
0.255884317139606 0.257401207292766
874 0.103303146785265 0.10395584540888 875
0.940062060623072 0.944818029471471
876 0.88910003618025 0.893582297554377 877
0.81622443573615 0.820343603841875

878	0.723234413206174	0.726905328038456
879	0.612426198171094	0.615568230598259
880	0.486537168624708	0.489073800366903
881	0.348679516809252	0.350536750027629
882	0.202264174994631	0.2033683215379
883	0.050919774488151	0.0511922900311454
884	0.897280859664431	0.90176944487161
885	0.832370766797028	0.836516303737808
886	0.746946046194925	0.750665354967537
887	0.643115599277123	0.646330533842485
888	0.52344328032939	0.526080909926171
889	0.390885205766638	0.392877428045034
890	0.248716840359394	0.25
891	0.100444765598865	0.100966742252535
892	0.848667889315243	0.852868157970561
893	0.779149859343794	0.782966426513139
894	0.69042687441183	0.693785463118375
895	0.58468849724172	0.587521199079693
896	0.46454405515364	0.466790213248601
897	0.332957841517679	0.334565303179429
898	0.193173829502662	0.194102284987398
899	0.0486392724669062	0.0488598243504268
900	0.787334302303147	0.791153573830373
901	0.70657280304083	0.709958163014308
902	0.608394902680392	0.611281226008774
903	0.495222692133813	0.497552516492828
904	0.36984730603751	0.371572412738697
905	0.23535856097542	0.236442963004804
906	0.0950622380280072	0.0954915028125265
907	0.722859975706549	0.72631001724706
908	0.64058378324221	0.643582297554377
909	0.542513644985758	0.545007445768716
910	0.431067217149902	0.433012701892219
911	0.308989865793066	0.31035574820837
912	0.179285086767188	0.180056805991955
913	0.0451466782671507	0.0453242676377405
914	0.648769331976639	0.65176944487161
915	0.558656779341716	0.561180145664649
916	0.454767279687932	0.456772728821301
917	0.339659811908224	0.34111805145259
918	0.216166481077002	0.217063915551224
919	0.0873184253419751	0.0876649456551456
920	0.574936268536334	0.577531999897403
921	0.486945653440614	0.489073800366903
922	0.386939819217771	0.388572980728485
923	0.27737950405205	0.278504204704746
924	0.160956764557047	0.161577730858712
925	0.0405351964003534	0.0406726770331929
926	0.495130366085317	0.497260947684137
927	0.403079856789097	0.404745680624419
928	0.301076113717542	0.302264231633827
929	0.191625859630579	0.192340034102939

930 0.0774125000200533 0.0776797865924608 931
0.419356460499035 0.421097534857171
932 0.333253623917105 0.334565303179429 933
0.238911982670295 0.239794963378828
934 0.138646836869413 0.139120075745983 935
0.0349206179357364 0.0350195901352119
936 0.341436624799208 0.342752450496663 937
0.25505098078289 0.255967663274761
938 0.162346036116465 0.162880102675064 939
0.0655912423031179 0.0657818933794384
940 0.271331783729473 0.272319517507514 941
0.19453621350807 0.195181174220667
942 0.112906299714676 0.113236822655327 943
0.0284417272330708 0.0285042047047459
944 0.202717768134333 0.2033683215379 945
0.129048327049505 0.12940952255126
946 0.0521459930975728 0.0522642316338269 947
0.145342001507616 0.145761376784013
948 0.0843668958826092 0.0845653031794291 949
0.0212574664184977 0.0212869511544171
950 0.0925487243774218 0.0927524504966631 951
0.0374058986914827 0.0374596510503503
952 0.0537247702828434 0.0538115052831031 953
0.0135426733705524 0.0135455422193262
954 0.0217285086193253 0.0217326895365599 955
0.00548104088740222 0.00547059707971813
1165 0.988115294680124 0.993158937674856 1166
0.940074062258231 0.944818029471471
1167 0.848686592776835 0.852868157970561 1168
0.722883539096124 0.72631001724706
1169 0.574962372363234 0.577531999897403 1170
0.419382530535565 0.421097534857171
1171 0.271355237742213 0.272319517507514 1172
0.145360487998845 0.145761376784013
1173 0.0537363315350617 0.0538115052831031 1174
0.00548138336203343 0.00547059707971813
1175 0.967851488243846 0.972789205831714 1176
0.897265275884565 0.90176944487161
1177 0.787312854827497 0.791153573830373 1178
0.648744098964722 0.65176944487161
1179 0.495103786592405 0.497260947684137 1180
0.341411253483776 0.342752450496663
1181 0.20269600918312 0.2033683215379 1182
0.092532547833458 0.0927524504966631
1183 0.0217195284356799 0.0217326895365599 1184
0.95969951281767 0.96460205851448
1185 0.889115551219346 0.893582297554377 1186
0.77917121094723 0.782966426513139
1187 0.640608843566386 0.643582297554377 1188
0.486971908538202 0.489073800366903
1189 0.333278411027659 0.334565303179429 1190
0.194556952527868 0.195181174220667

1191	0.0843812583382613	0.0845653031794291
1192	0.013543345355838	0.0135455422193262
1193	0.923741635544364	0.928466175238718
1194	0.832351987969144	0.836516303737808
1195	0.706549115928673	0.709958163014308
1196	0.558630441631044	0.561180145664649
1197	0.403053354360936	0.404745680624419
1198	0.255026763221403	0.255967663274761
1199	0.129028514180146	0.12940952255126
1200	0.0373929619908692	0.0374596510503503
1201	0.907627587086182	0.912293475342785
1202	0.816243083740795	0.820343603841875
1203	0.690450324983622	0.693785463118375
1204	0.542539529480172	0.545007445768716
1205	0.386965478451473	0.388572980728485
1206	0.238934711339343	0.239794963378828
1207	0.112923566463727	0.113236822655327
1208	0.0212594115463308	0.0212869511544171
1209	0.8568646802525	0.861281226008774
1210	0.746924524372379	0.750665354967537
1211	0.608369521526085	0.611281226008774
1212	0.454740410625817	0.456772728821301
1213	0.301050213101234	0.302264231633827
1214	0.162323370274802	0.162880102675064
1215	0.0521299351236655	0.0522642316338269
1216	0.83318692436422	0.83752119907969
1217	0.72325572471972	0.726905328038456
1218	0.58471343388537	0.587521199079693
1219	0.431093216660719	0.433012701892219
1220	0.277403814677292	0.278504204704746
1221	0.138666800162283	0.139120075745983
1222	0.0284450891846629	0.0285042047047459
1223	0.768872036915695	0.772888674565986
1224	0.643091843697664	0.646330533842485
1225	0.495196191880148	0.497552516492828
1226	0.339632980853028	0.341118051452596
1227	0.191601059765418	0.192340034102939
1228	0.0655726336307214	0.0657818933794384
1229	0.738215877759044	0.742126371321759
1230	0.61244963058722	0.615568230598259
1231	0.464569818932337	0.466790213248601
1232	0.309015255423254	0.31035574820837
1233	0.160979049082538	0.161577730858712
1234	0.0349253837052752	0.0350195901352119
1235	0.661936870335336	0.665465038884934
1236	0.523417848722273	0.526080909926171
1237	0.369820278982833	0.371572412738697
1238	0.216140252739385	0.217063915551224
1239	0.0773918801004621	0.0776797865924608
1240	0.625060450744217	0.628457929325666
1241	0.486562124055121	0.489073800366903
1242	0.3329837471594	0.334565303179429

DOI: 10.18535/ijecs/v6i10.13

1243 0.1793092135782 0.180056805991955 1244
0.0405412825127846 0.0406726770331929
1245 0.538701547691147 0.541655445394692 1246
0.390858695726009 0.392877428045034
1247 0.235331612265809 0.236442963004804 1248
0.0872963470177766 0.0876649456551456
1249 0.496517912785114 0.499314767377287 1250
0.348705366234245 0.350536750027629
1251 0.193199255898098 0.194102284987398 1252
0.0451539547900739 0.0453242676377405
1253 0.402214489309537 0.404508497187474 1254
0.248689887888625 0.25
1255 0.0950392768281424 0.0954915028125265 1256
0.355770874642914 0.357876818725374
1257 0.202290364254579 0.2033683215379 1258
0.0486475785598033 0.0488598243504268
1259 0.255859367484791 0.257401207292766 1260
0.100421537234301 0.100966742252535
1261 0.206316809444232 0.207626755071376 1262
0.0509289978755843 0.0511922900311454
1263 0.103281721890493 0.10395584540888 1264
0.0518995701631073 0.0522642316338272
1474 0.988111952397116 0.993158937674856 1475
0.959693109629127 0.96460205851448
1476 0.907618249217389 0.912293475342785 1477
0.833174876762053 0.837521199079693
1478 0.738201418926481 0.742126371321759 1479
0.625043940727517 0.628457929325666
1480 0.496499762060653 0.499314767377287 1481
0.355751523594172 0.357876818725374
1482 0.206296519449983 0.207626755071376 1483
0.051897428478355 0.0522642316338272
1484 0.967859476875106 0.972789205831713 1485
0.923753282722173 0.928466175238718
1486 0.856879681245561 0.861281226008774 1487
0.768890023972325 0.772888674565986
1488 0.661957407796385 0.665465038884933 1489
0.538724138989469 0.541655445394692
1490 0.402238582048672 0.404508497187474 1491
0.255884317139604 0.257401207292766
1492 0.103303146785264 0.10395584540888 1493
0.940062060623064 0.944818029471471
1494 0.889100036180241 0.893582297554377 1495
0.816224435736146 0.820343603841875
1496 0.723234413206174 0.726905328038456 1497
0.612426198171095 0.615568230598259
1498 0.486537168624705 0.489073800366903 1499
0.348679516809252 0.350536750027629
1500 0.202264174994629 0.2033683215379 1501
0.0509197744881504 0.0511922900311454
1502 0.897280859664426 0.90176944487161 1503
0.832370766797026 0.836516303737808

DOI: 10.18535/ijecs/v6i10.13

1504	0.746946046194924	0.750665354967537
1505	0.643115599277124	0.646330533842485
1506	0.52344328032939	0.526080909926171
1507	0.390885205766636	0.392877428045034
1508	0.248716840359392	0.25
1509	0.100444765598864	0.100966742252535
1510	0.848667889315237	0.852868157970561
1511	0.779149859343795	0.782966426513139
1512	0.690426874411829	0.693785463118375
1513	0.58468849724172	0.587521199079693
1514	0.464544055153642	0.466790213248601
1515	0.332957841517677	0.334565303179429
1516	0.193173829502661	0.194102284987398
1517	0.0486392724669059	0.0488598243504268
1518	0.787334302303148	0.791153573830373
1519	0.706572803040829	0.709958163014308
1520	0.608394902680393	0.611281226008774
1521	0.49522269213381	0.497552516492828
1522	0.369847306037508	0.371572412738697
1523	0.23535856097542	0.236442963004804
1524	0.0950622380280069	0.0954915028125265
1525	0.722859975706552	0.72631001724706
1526	0.640583783242209	0.643582297554377
1527	0.542513644985761	0.545007445768716
1528	0.431067217149903	0.433012701892219
1529	0.308989865793066	0.31035574820837
1530	0.179285086767188	0.180056805991955
1531	0.045146678267151	0.0453242676377405
1532	0.648769331976642	0.65176944487161
1533	0.558656779341721	0.561180145664649
1534	0.454767279687933	0.456772728821301
1535	0.339659811908224	0.341118051452596
1536	0.216166481077002	0.217063915551224
1537	0.0873184253419753	0.0876649456551455
1538	0.574936268536335	0.577531999897402
1539	0.486945653440616	0.489073800366903
1540	0.386939819217771	0.388572980728485
1541	0.27737950405205	0.278504204704746
1542	0.160956764557048	0.161577730858712
1543	0.0405351964003535	0.0406726770331929
1544	0.495130366085319	0.497260947684137
1545	0.403079856789097	0.404745680624419
1546	0.301076113717541	0.302264231633827
1547	0.19162585963058	0.192340034102939
1548	0.0774125000200536	0.0776797865924607
1549	0.419356460499037	0.421097534857172
1550	0.333253623917105	0.334565303179429
1551	0.238911982670296	0.239794963378828
1552	0.138646836869414	0.139120075745983
1553	0.0349206179357365	0.035019590135212
1554	0.341436624799207	0.342752450496663
1555	0.255050980782891	0.255967663274761

DOI: 10.18535/ijecs/v6i10.13

1556 0.162346036116466 0.162880102675064 1557
0.0655912423031182 0.0657818933794384
1558 0.271331783729474 0.272319517507513 1559
0.194536213508071 0.195181174220666
1560 0.112906299714677 0.113236822655327 1561
0.028441727233071 0.0285042047047458
1562 0.202717768134334 0.2033683215379 1563
0.129048327049507 0.12940952255126
1564 0.0521459930975732 0.0522642316338269 1565
0.145342001507617 0.145761376784013
1566 0.0843668958826098 0.0845653031794291 1567
0.0212574664184977 0.0212869511544171
1568 0.0925487243774224 0.092752450496663 1569
0.0374058986914828 0.0374596510503503
1570 0.0537247702828437 0.053811505283103 1571
0.0135426733705524 0.0135455422193261
1572 0.0217285086193254 0.0217326895365599 1573
0.00548104088740221 .00547059707971812
1783 0.988115294680123 0.993158937674856 1784
0.940074062258228 0.944818029471471
1785 0.848686592776829 0.852868157970561 1786
0.722883539096119 0.72631001724706
1787 0.57496237236323 0.577531999897402 1788
0.419382530535566 0.421097534857171
1789 0.271355237742216 0.272319517507514 1790
0.145360487998845 0.145761376784013
1791 0.053736331535062 0.0538115052831031 1792
0.00548138336203345 0.00547059707971809
1793 0.96785148824385 0.972789205831714 1794
0.897265275884565 0.90176944487161
1795 0.78731285482749 0.791153573830373 1796
0.648744098964719 0.65176944487161
1797 0.495103786592405 0.497260947684137 1798
0.341411253483777 0.342752450496663
1799 0.202696009183122 0.2033683215379 1800
0.0925325478334584 0.092752450496663
1801 0.0217195284356801 0.0217326895365599 1802
0.959699512817674 0.96460205851448
1803 0.889115551219342 0.893582297554377 1804
0.779171210947213 0.782966426513139
1805 0.64060884356638 0.643582297554377 1806
0.4869719085382 0.489073800366903
1807 0.33327841102766 0.334565303179429 1808
0.194556952527868 0.195181174220666
1809 0.0843812583382617 0.0845653031794291 1810
0.013543345355838 0.013545542219326
1811 0.92374163554437 0.928466175238718 1812
0.832351987969141 0.836516303737808
1813 0.706549115928662 0.709958163014307 1814
0.558630441631042 0.56118014566465
1815 0.403053354360934 0.404745680624419 1816
0.255026763221403 0.255967663274761

1817 0.129028514180146 0.12940952255126 1818
0.0373929619908693 0.0374596510503502
1819 0.907627587086182 0.912293475342785 1820
0.816243083740791 0.820343603841875
1821 0.690450324983609 0.693785463118374 1822
0.542539529480168 0.545007445768716
1823 0.386965478451472 0.388572980728485 1824
0.238934711339343 0.239794963378827
1825 0.112923566463728 0.113236822655327 1826
0.0212594115463309 0.0212869511544169
1827 0.856864680252496 0.861281226008774 1828
0.746924524372366 0.750665354967537
1829 0.608369521526079 0.611281226008774 1830
0.454740410625816 0.4567727288213
1831 0.301050213101235 0.302264231633827 1832
0.162323370274804 0.162880102675064
1833 0.0521299351236659 0.0522642316338267 1834
0.833186924364215 0.837521199079693
1835 0.723255724719712 0.726905328038456 1836
0.584713433885364 0.587521199079693
1837 0.431093216660716 0.433012701892219 1838
0.277403814677293 0.278504204704746
1839 0.138666800162284 0.139120075745983 1840
0.028445089184663 0.0285042047047456
1841 0.768872036915686 0.772888674565986 1842
0.643091843697656 0.646330533842485
1843 0.495196191880145 0.497552516492827 1844
0.339632980853028 0.341118051452597
1845 0.19160105976542 0.192340034102939 1846
0.0655726336307218 0.0657818933794382
1847 0.738215877759038 0.742126371321759 1848
0.612449630587217 0.615568230598259
1849 0.464569818932333 0.466790213248601 1850
0.309015255423254 0.31035574820837
1851 0.160979049082539 0.161577730858712 1852
0.0349253837052753 0.0350195901352117
1853 0.661936870335331 0.665465038884934 1854
0.523417848722274 0.526080909926171
1855 0.369820278982832 0.371572412738697 1856
0.216140252739386 0.217063915551223
1857 0.0773918801004624 0.0776797865924606 1858
0.625060450744215 0.628457929325666
1859 0.486562124055123 0.489073800366903 1860
0.332983747159401 0.334565303179429
1861 0.1793092135782 0.180056805991955 1862
0.0405412825127846 0.0406726770331925
1863 0.538701547691151 0.541655445394692 1864
0.390858695726009 0.392877428045034
1865 0.23533161226581 0.236442963004803 1866
0.0872963470177766 0.0876649456551453
1867 0.496517912785123 0.499314767377287 1868
0.348705366234246 0.350536750027629

1869 0.193199255898098 0.194102284987398 1870
0.0451539547900739 0.0453242676377401
1871 0.402214489309539 0.404508497187474 1872
0.248689887888626 0.25
1873 0.0950392768281423 0.0954915028125263 1874
0.355770874642918 0.357876818725374
1875 0.20229036425458 0.2033683215379 1876
0.0486475785598032 0.0488598243504264
1877 0.255859367484795 0.257401207292766 1878
0.100421537234301 0.100966742252535
1879 0.206316809444234 0.207626755071376 1880
0.0509289978755844 0.0511922900311449
1881 0.103281721890494 0.10395584540888 1882
0.0518995701631077 0.0522642316338267
2092 0.988111952397121 0.993158937674856 2093
0.959693109629134 0.96460205851448
2094 0.907618249217393 0.912293475342785 2095
0.833174876762054 0.837521199079693
2096 0.73820141892648 0.742126371321759 2097
0.625043940727519 0.628457929325666
2098 0.496499762060662 0.499314767377287 2099
0.355751523594179 0.357876818725374
2100 0.206296519449987 0.207626755071376 2101
0.0518974284783561 0.0522642316338267
2102 0.967859476875112 0.972789205831713 2103
0.923753282722178 0.928466175238718
2104 0.856879681245566 0.861281226008774 2105
0.768890023972326 0.772888674565986
2106 0.661957407796388 0.665465038884934 2107
0.53872413898948 0.541655445394692
2108 0.402238582048681 0.404508497187474 2109
0.25588431713961 0.257401207292766
2110 0.103303146785267 0.10395584540888 2111
0.940062060623081 0.944818029471471
2112 0.889100036180253 0.893582297554377 2113
0.816224435736148 0.820343603841875
2114 0.723234413206173 0.726905328038456 2115
0.612426198171101 0.615568230598259
2116 0.486537168624717 0.489073800366903 2117
0.348679516809262 0.350536750027629
2118 0.202264174994636 0.2033683215379 2119
0.0509197744881521 0.0511922900311449
2120 0.897280859664436 0.90176944487161 2121
0.832370766797032 0.836516303737808
2122 0.746946046194924 0.750665354967537 2123
0.643115599277127 0.646330533842485
2124 0.523443280329401 0.526080909926171 2125
0.390885205766646 0.392877428045034
2126 0.248716840359401 0.25 2127
0.100444765598867 0.100966742252535
2128 0.848667889315251 0.852868157970561 2129
0.779149859343798 0.782966426513139

DOI: 10.18535/ijecs/v6i10.13

2130	0.690426874411831	0.693785463118374	2131	0.584688497241727	0.587521199079693
2132	0.464544055153651	0.466790213248601	2133	0.332957841517687	0.334565303179429
2134	0.193173829502668	0.194102284987398	2135	0.0486392724669074	0.0488598243504264
2136	0.787334302303156	0.791153573830373	2137	0.706572803040833	0.709958163014307
2138	0.608394902680397	0.611281226008774	2139	0.495222692133819	0.497552516492827
2140	0.369847306037517	0.371572412738697	2141	0.235358560975428	0.236442963004803
2142	0.0950622380280101	0.0954915028125263	2143	0.722859975706561	0.72631001724706
2144	0.640583783242216	0.643582297554376	2145	0.542513644985769	0.545007445768716
2146	0.431067217149914	0.433012701892219	2147	0.308989865793075	0.31035574820837
2148	0.179285086767194	0.180056805991955	2149	0.0451466782671521	0.0453242676377401
2150	0.648769331976649	0.65176944487161	2151	0.558656779341727	0.56118014566465
2152	0.454767279687941	0.4567727288213	2153	0.339659811908233	0.341118051452596
2154	0.216166481077008	0.217063915551223	2155	0.0873184253419778	0.0876649456551453
2156	0.574936268536345	0.577531999897402	2157	0.486945653440624	0.489073800366903
2158	0.386939819217778	0.388572980728485	2159	0.277379504052057	0.278504204704746
2160	0.160956764557051	0.161577730858712	2161	0.0405351964003543	0.0406726770331925
2162	0.495130366085325	0.497260947684137	2163	0.403079856789103	0.404745680624419
2164	0.301076113717548	0.302264231633827	2165	0.191625859630584	0.192340034102939
2166	0.0774125000200551	0.0776797865924605	2167	0.419356460499042	0.421097534857172
2168	0.333253623917109	0.334565303179429	2169	0.238911982670299	0.239794963378828
2170	0.138646836869416	0.139120075745983	2171	0.0349206179357369	0.0350195901352117
2172	0.341436624799213	0.342752450496663	2173	0.255050980782893	0.255967663274761
2174	0.162346036116468	0.162880102675064	2175	0.0655912423031187	0.0657818933794383
2176	0.271331783729475	0.272319517507513	2177	0.194536213508073	0.195181174220666
2178	0.112906299714678	0.113236822655327	2179	0.0284417272330712	0.0285042047047456
2180	0.202717768134335	0.2033683215379	2181	0.129048327049508	0.12940952255126

DOI: 10.18535/ijecs/v6i10.13

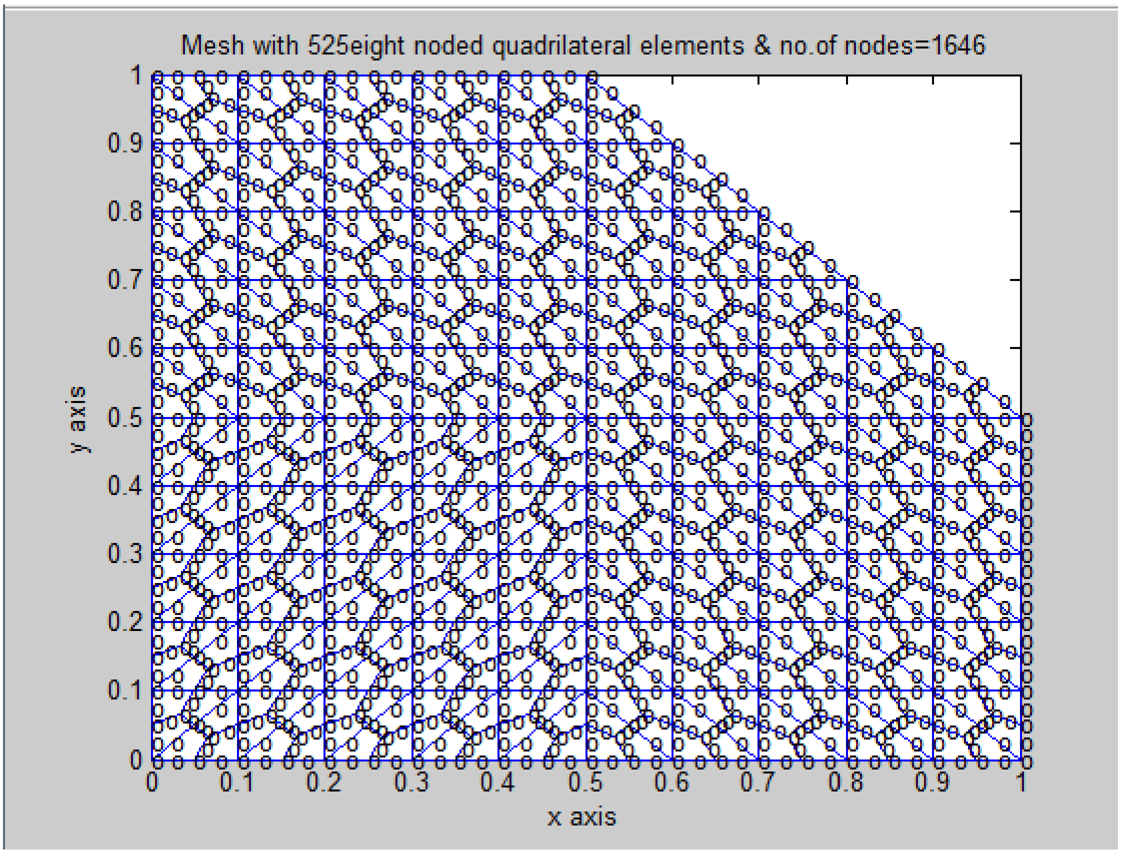
2182 0.0521459930975738 0.0522642316338267 2183
0.145342001507618 0.145761376784013
2184 0.0843668958826108 0.0845653031794291 2185
0.021257466418498 .0212869511544169
2186 0.0925487243774236 0.0927524504966629 2187
0.0374058986914834 0.0374596510503502
2188 0.0537247702828449 0.053811505283103 2189
0.0135426733705527 0.013545542219326
2190 0.0217285086193258 0.0217326895365599 2191
0.00548104088740231 .00547059707971807
2382 0.988115294680128 0.993158937674856 2383
0.940074062258251 0.944818029471471
2384 0.848686592776846 0.852868157970561 2385
0.722883539096136 0.72631001724706
2386 0.574962372363244 0.577531999897402 2387
0.419382530535573 0.421097534857172
2388 0.271355237742217 0.272319517507513 2389
0.145360487998847 0.145761376784013
2390 0.0537363315350629 0.053811505283103 2391
0.00548138336203353 0.00547059707971807
2392 0.967851488243867 0.972789205831713 2393
0.897265275884586 0.90176944487161
2394 0.787312854827511 0.791153573830373 2395
0.648744098964735 0.65176944487161
2396 0.495103786592414 0.497260947684137 2397
0.341411253483782 0.342752450496663
2398 0.202696009183122 0.2033683215379 2399
0.0925325478334594 0.0927524504966629
2400 0.0217195284356804 0.0217326895365599 2401
0.959699512817696 0.96460205851448
2402 0.889115551219364 0.893582297554377 2403
0.779171210947247 0.782966426513139
2404 0.640608843566401 0.643582297554376 2405
0.486971908538209 0.489073800366903
2406 0.333278411027663 0.334565303179429 2407
0.19455695252787 0.195181174220666
2408 0.0843812583382624 0.0845653031794291 2409
0.0135433453558382 0.013545542219326
2410 0.923741635544387 0.928466175238718 2411
0.832351987969173 0.836516303737808
2412 0.706549115928686 0.709958163014307 2413
0.558630441631058 0.56118014566465
2414 0.403053354360942 0.404745680624419 2415
0.255026763221404 0.255967663274761
2416 0.129028514180147 0.12940952255126 2417
0.0373929619908696 0.0374596510503502
2418 0.907627587086206 0.912293475342785 2419
0.816243083740823 0.820343603841875
2420 0.690450324983639 0.693785463118374 2421
0.542539529480184 0.545007445768716
2422 0.386965478451478 0.388572980728485 2423
0.238934711339345 0.239794963378828

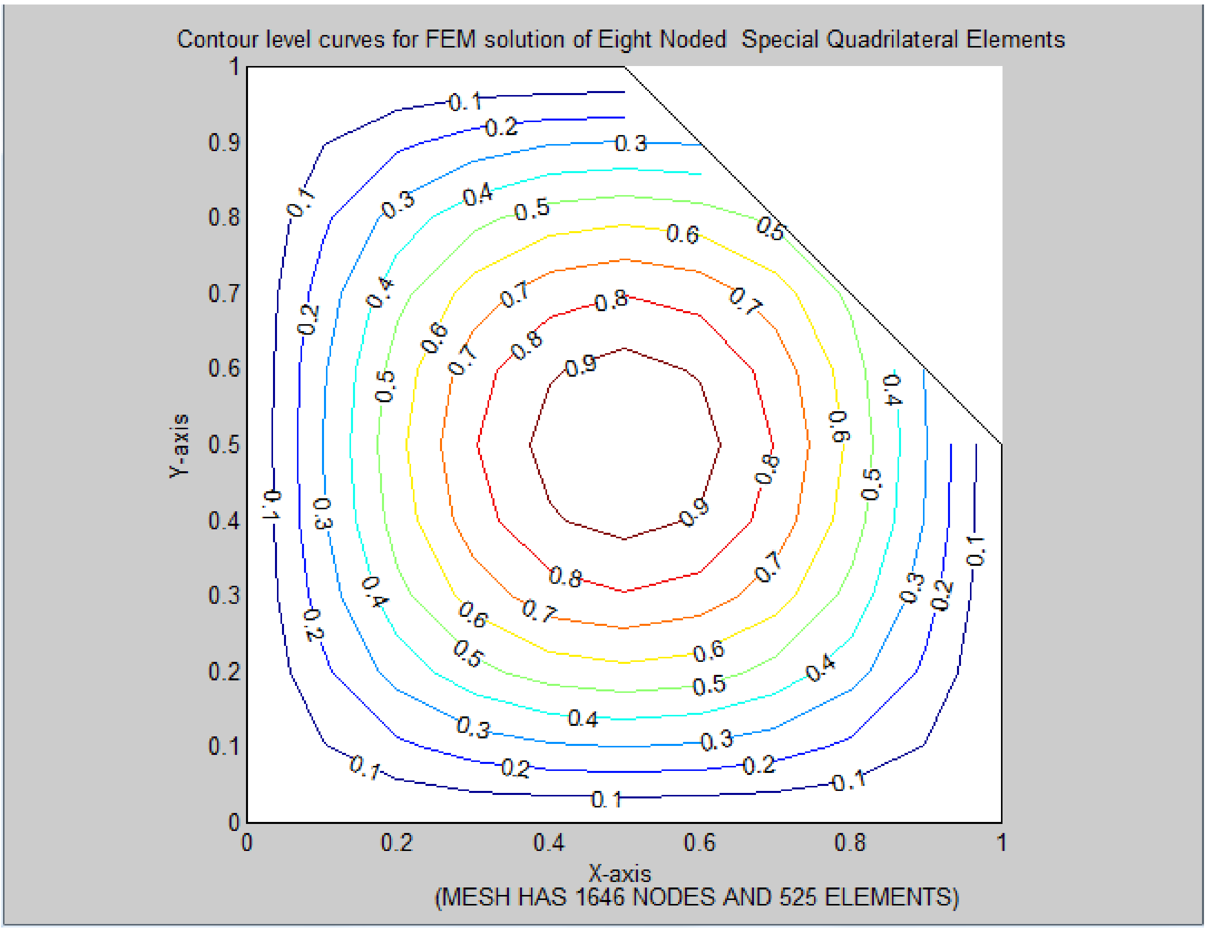
2424 0.112923566463729 0.113236822655327 2425
0.0212594115463312 0.0212869511544169
2426 0.856864680252524 0.861281226008774 2427
0.746924524372405 0.750665354967537
2428 0.608369521526105 0.611281226008774 2429
0.454740410625826 0.4567727288213
2430 0.301050213101238 0.302264231633827 2431
0.162323370274804 0.162880102675064
2432 0.0521299351236663 0.0522642316338267 2433
0.83318692436425 0.837521199079693
2434 0.723255724719751 0.726905328038456 2435
0.584713433885394 0.587521199079693
2436 0.431093216660728 0.433012701892219 2437
0.277403814677295 0.278504204704746
2438 0.138666800162284 0.139120075745983 2439
0.0284450891846633 0.0285042047047456
2440 0.76887203691572 0.772888674565986 2441
0.643091843697695 0.646330533842485
2442 0.495196191880164 0.497552516492827 2443
0.339632980853035 0.341118051452596
2444 0.191601059765421 0.192340034102939 2445
0.0655726336307221 0.0657818933794383
2446 0.738215877759074 0.742126371321759 2447
0.612449630587252 0.615568230598259
2448 0.464569818932353 0.466790213248601 2449
0.309015255423261 0.31035574820837
2450 0.160979049082542 0.161577730858712 2451
0.0349253837052756 0.0350195901352117
2452 0.661936870335364 0.665465038884934 2453
0.5234178487223 0.526080909926171
2454 0.369820278982844 0.371572412738697 2455
0.216140252739389 0.217063915551223
2456 0.0773918801004628 0.0776797865924605 2457
0.625060450744247 0.628457929325666
2458 0.486562124055148 0.489073800366903 2459
0.332983747159412 0.334565303179429
2460 0.179309213578204 0.180056805991955 2461
0.040541282512785 0.0406726770331925
2462 0.538701547691179 0.541655445394692 2463
0.390858695726028 0.392877428045034
2464 0.235331612265818 0.236442963004803 2465
0.0872963470177775 0.0876649456551453
2466 0.496517912785145 0.499314767377287 2467
0.348705366234264 0.350536750027629
2468 0.193199255898105 0.194102284987398 2469
0.045153954790075 0.0453242676377401
2470 0.40221448930956 0.404508497187474 2471
0.248689887888635 0.25
2472 0.0950392768281445 0.0954915028125263 2473
0.355770874642933 0.357876818725374
2474 0.202290364254588 0.2033683215379 2475
0.0486475785598049 0.0488598243504264

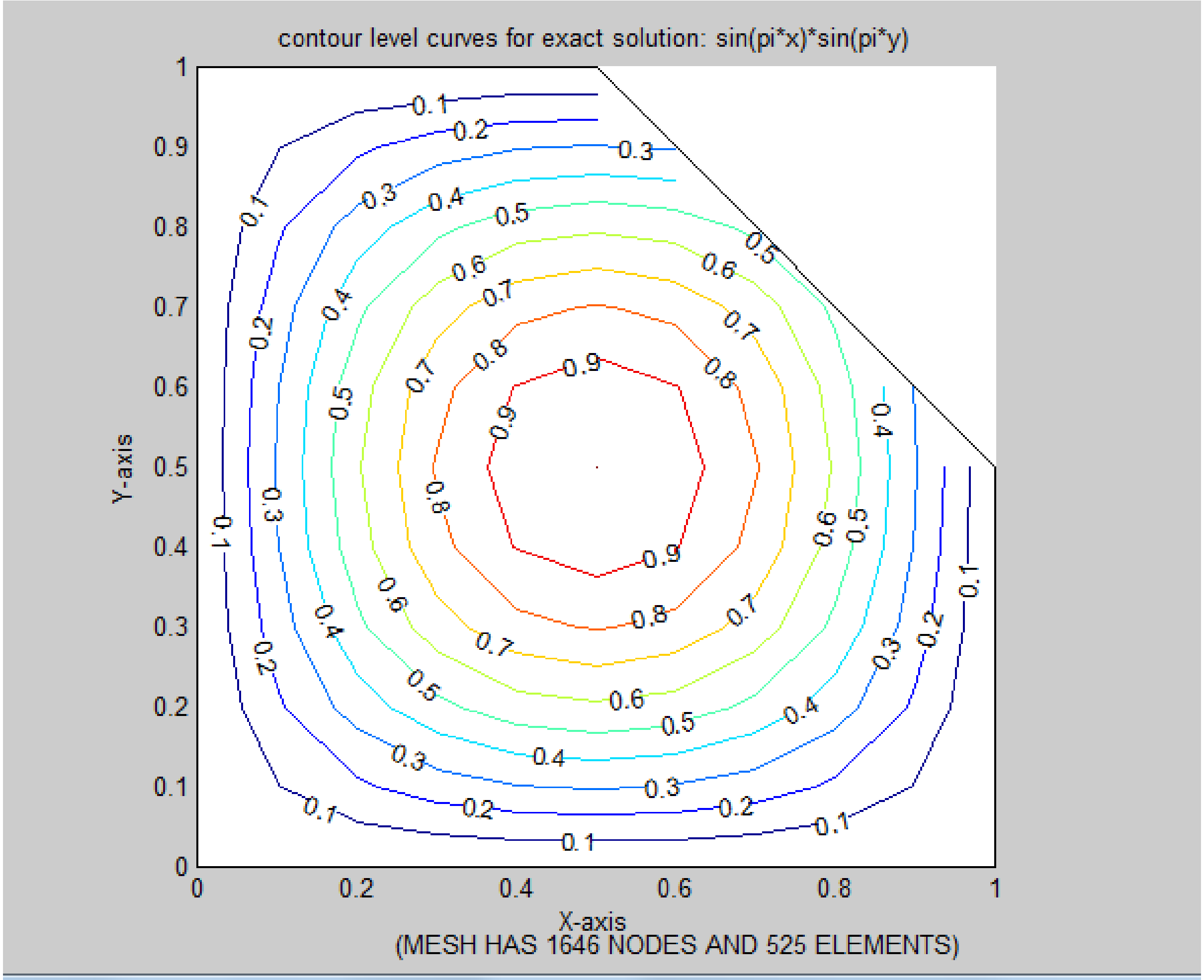
DOI: 10.18535/ijecs/v6i10.13

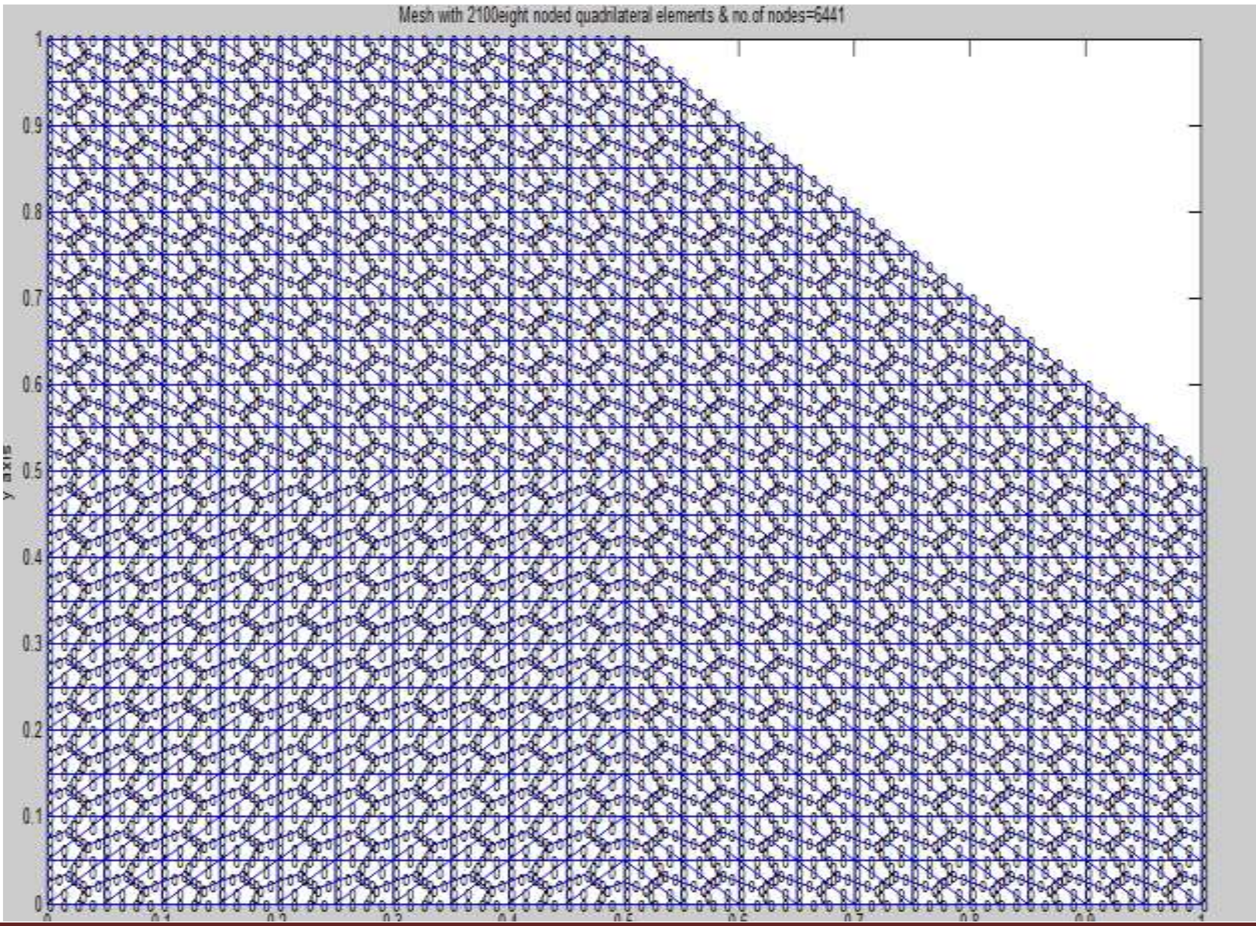
2476 0.255859367484803 0.257401207292766 2477
0.100421537234305 0.100966742252535
2478 0.206316809444241 0.207626755071376 2479
0.0509289978755865 0.0511922900311449
2480 0.103281721890497 0.10395584540888 2481
0.0518995701631094 0.0522642316338267

Meshes And Contour Level Curves For A Pentagonal Domain With 8-No0ded Quadrilateral Elements



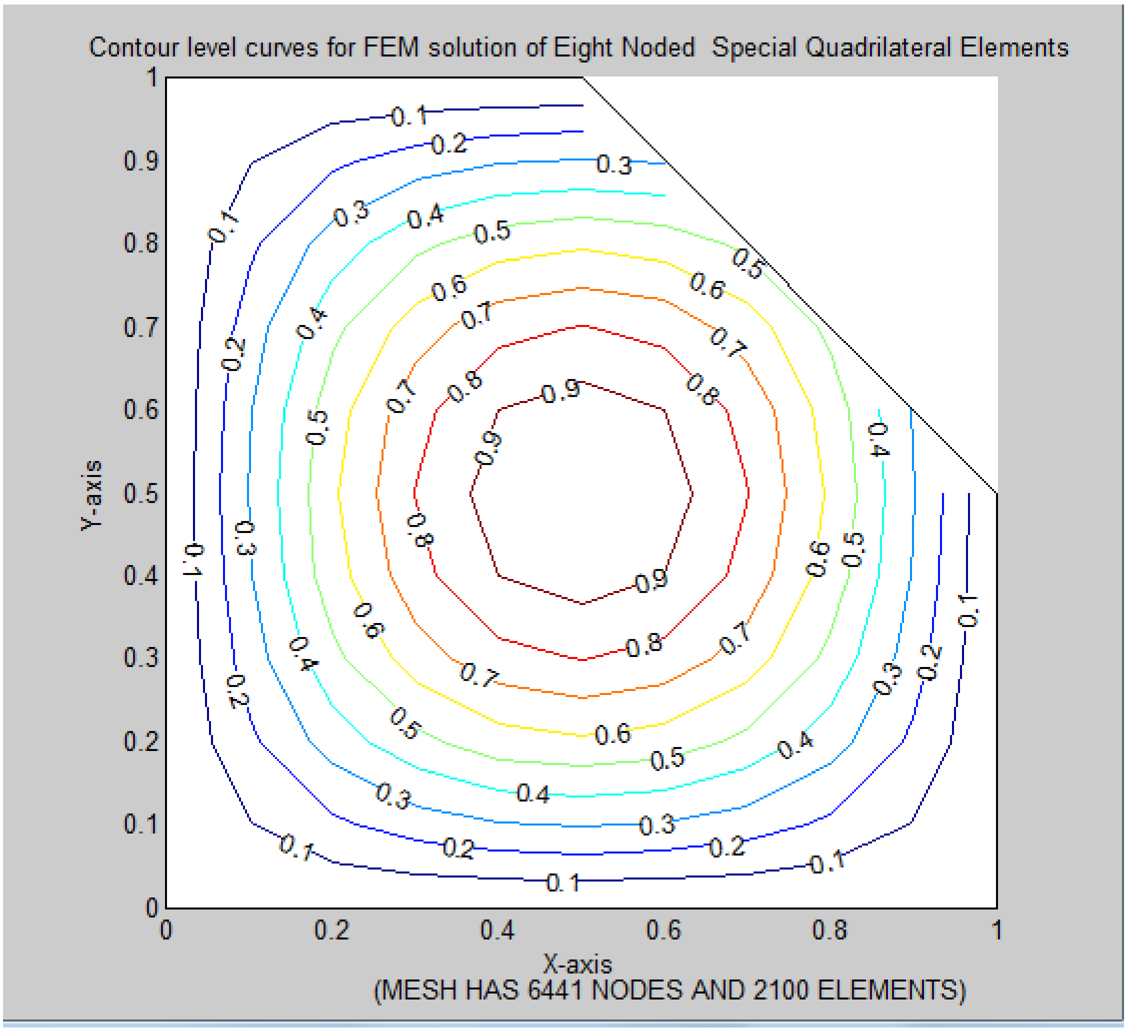

Contour level curves for FEM solution of Eight Noded Special Quadrilateral Elements
Y-axis
X-axis
(MESH HAS 1646 NODES AND 525 ELEMENTS)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


contour level curves for exact solution: sin(pi*x)*sin(pi*y)
Y-axis
X-axis
(MESH HAS 1646 NODES AND 525 ELEMENTS)

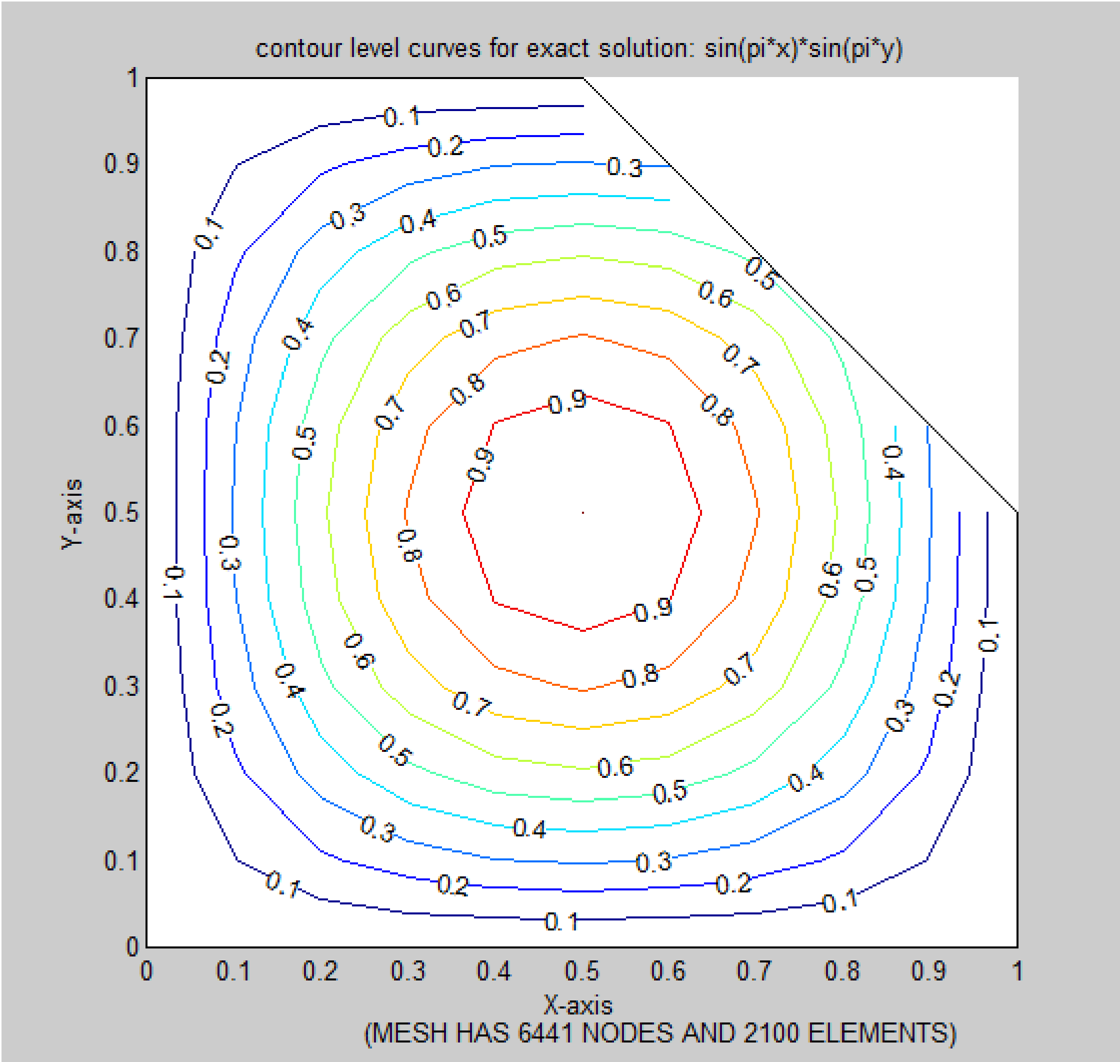

Mesh with 2100eight noded quadrilateral elements & no.of nodes=6441
y axis

DOI: 10.18535/ijecs/v6i10.13

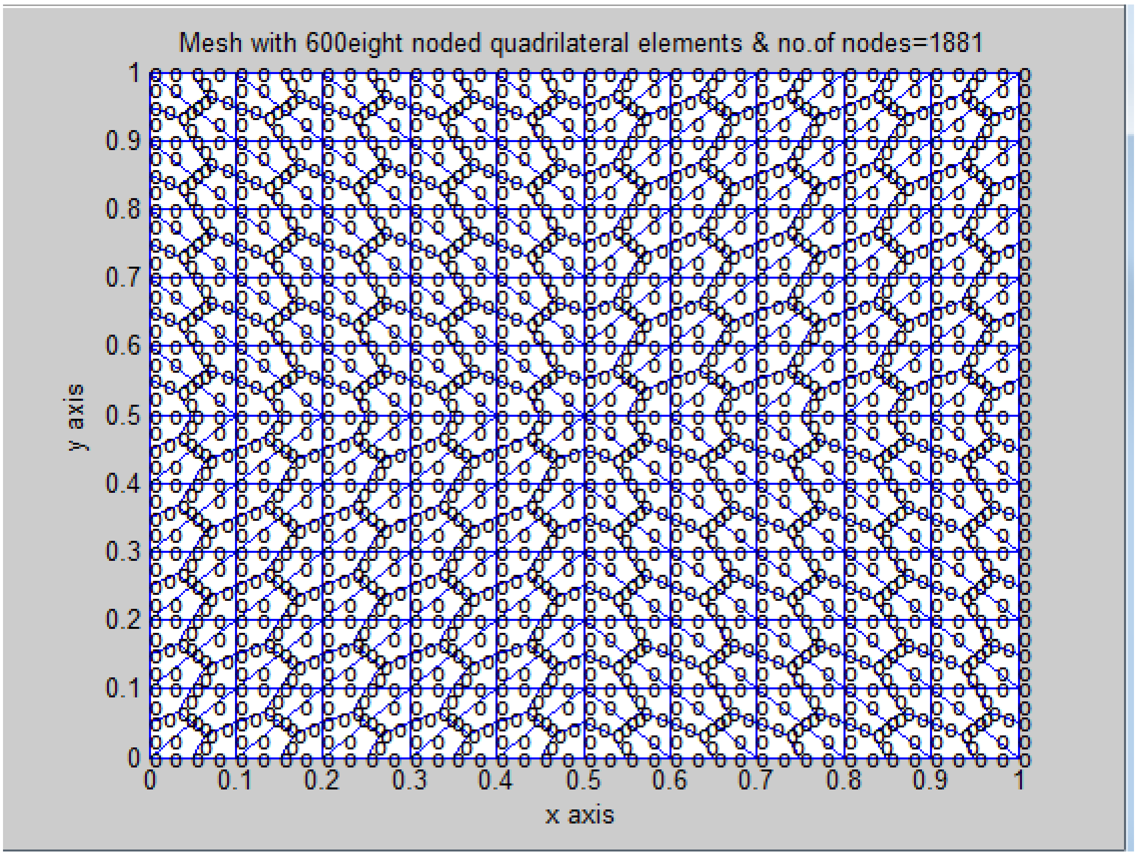
x-axis



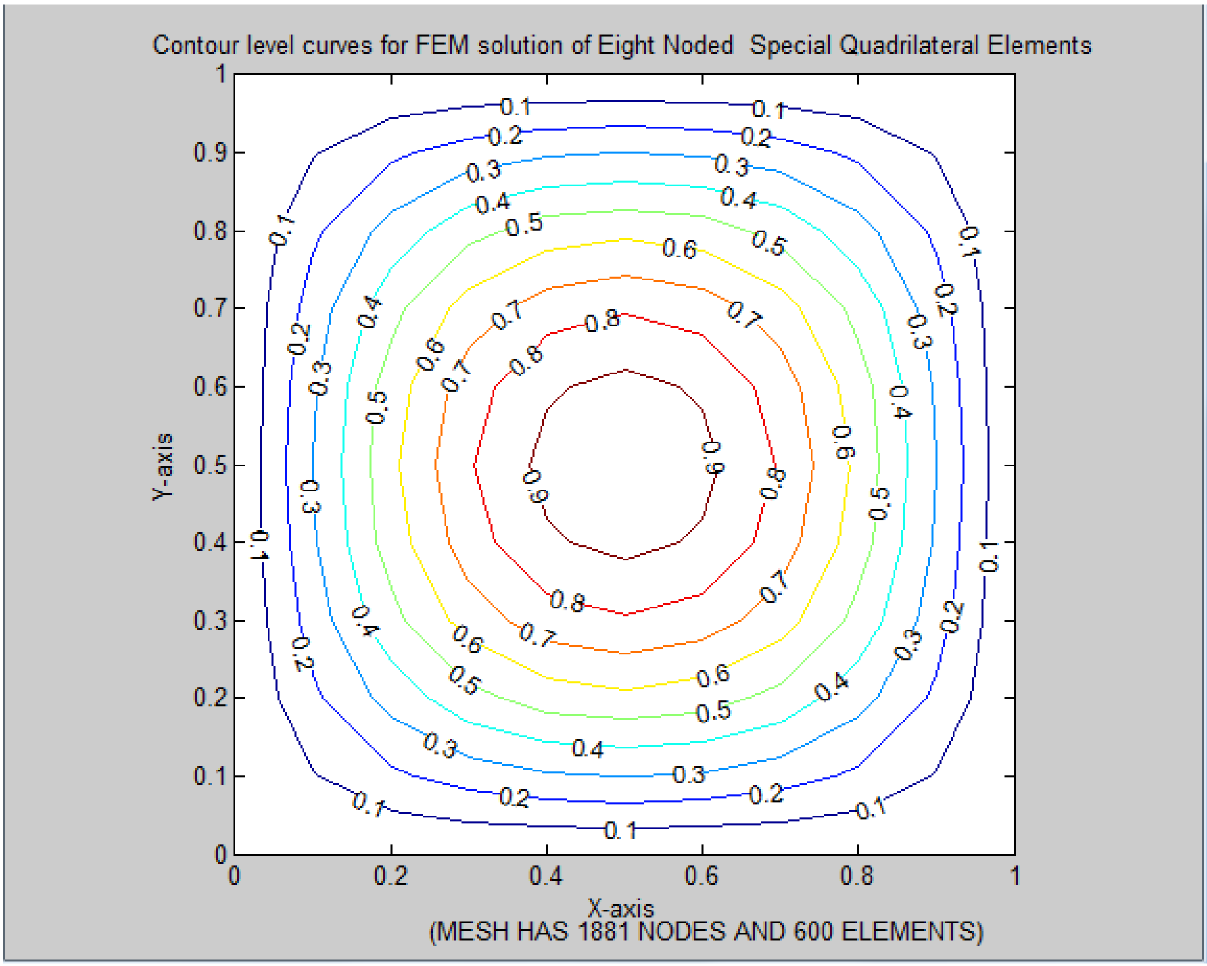
DOI: 10.18535/ijecs/v6i10.13

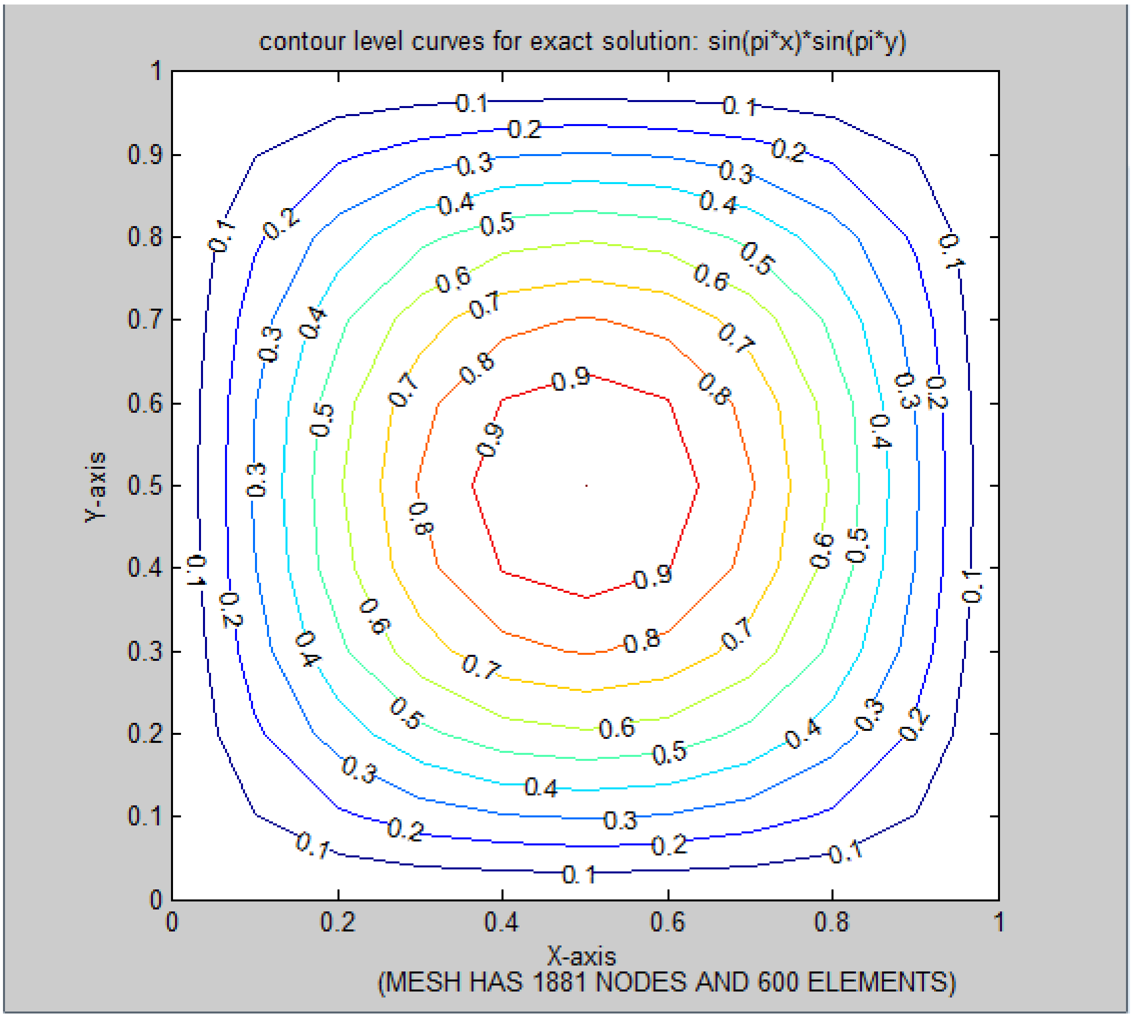


Meshes And Contour Level Curves For A Square Domain With 8-No0ded Quadrilateral Elements


Mesh with 600eight noded quadrilateral elements & no.of nodes=1881
y axis
x axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

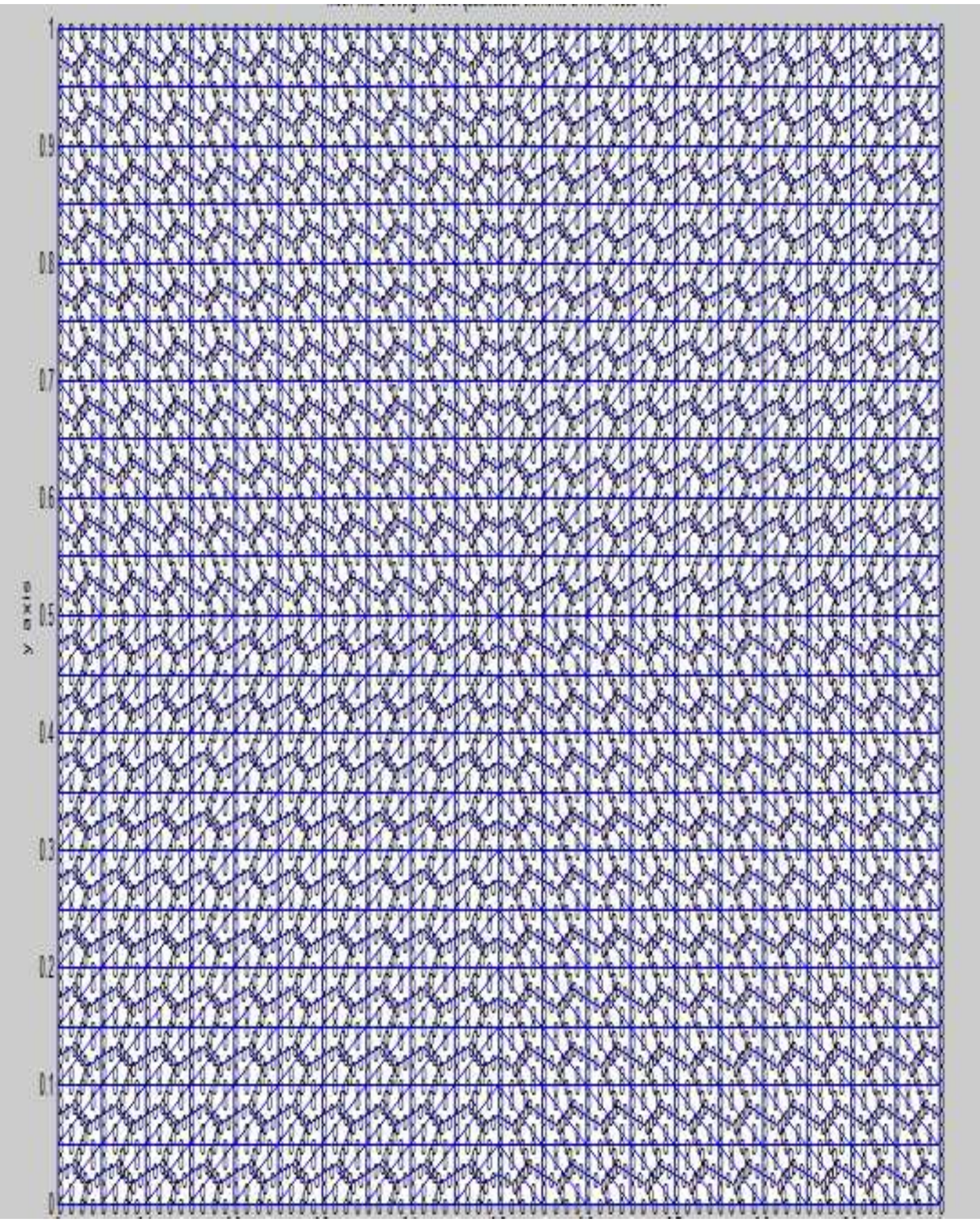
DOI: 10.18535/ijecs/v6i10.13





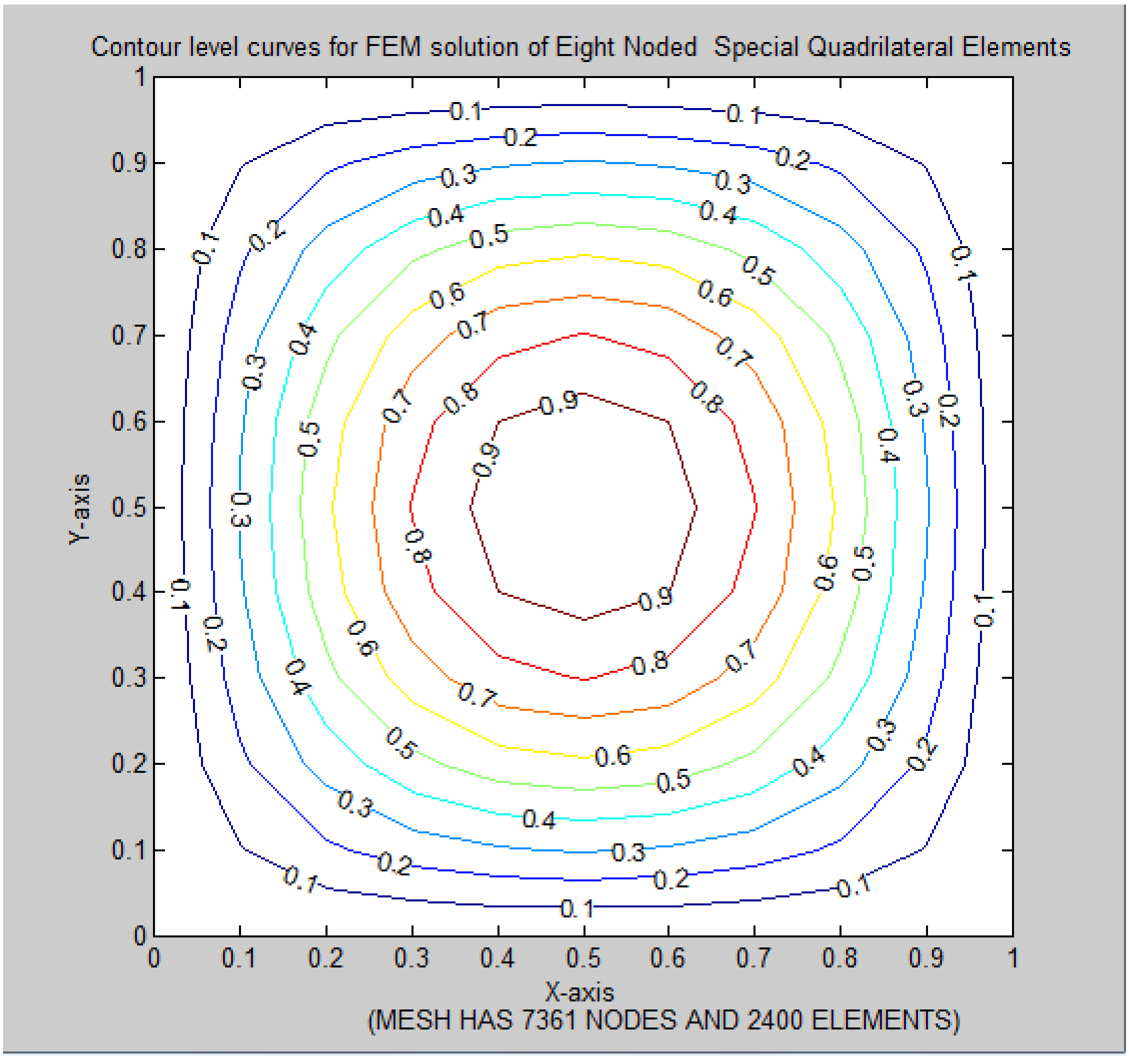
Mesh with 2400 eight noded quadrilateral elements & no. of nodes = 7361

DOI: 10.18535/ijecs/v6i10.13

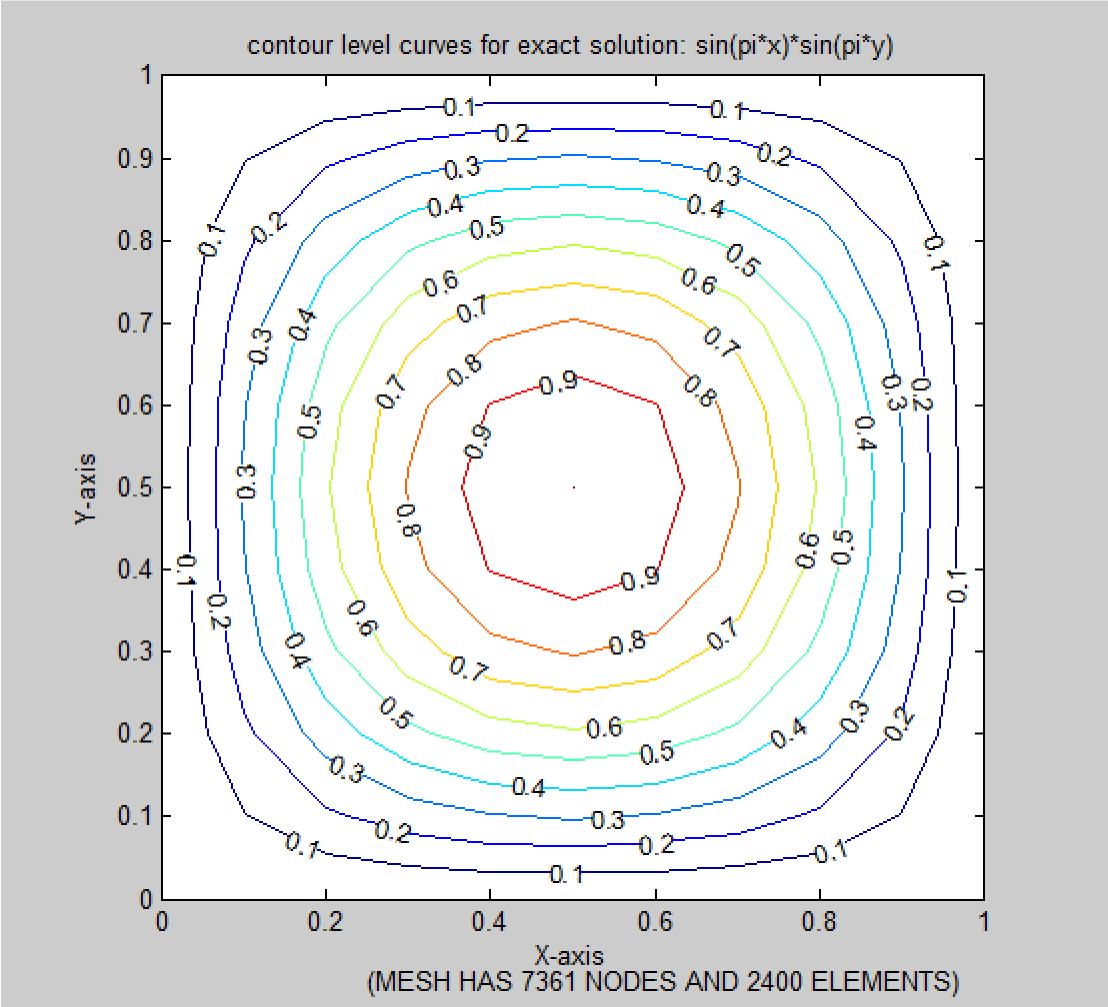


x-axis

DOI: 10.18535/ijecs/v6i10.13



DOI: 10.18535/ijecs/v6i10.13



(I)COMPUTER PROGRAMS:ARBITRARY TRIANGULAR D0MAINS
(1)*************

```
function[]=quadrilateralmesh_over_arbitrarytriangle_q8automeshgen(mmesh,nmesh,tri)
%quadrilateralmesh_over_arbitrarytriangle_q8automeshgen(mmesh,nmesh,tri)
%
clf
switch tri
case 1%standard triangle
xx=sym([0;1;0])
yy=sym([0;0;1])
case 2
 xx=sym([0;1/2;1/2])
 yy=sym([0;0;1/2])

case 3%equilateral triangle

xx=sym([0;1;1/2])
yy=sym([0;0;sqrt(3)/2])
   case 4%equilateral triangle
 xx=sym([-sqrt(3);sqrt(3); 0])
 yy=sym([     -1;      -1; 2])


end
```

```
for mesh=mmesh:nmesh
   figure(mesh)
   ndiv=2*mesh;
 [eln,nodetel,nodes,nnode]=nodaladdresses_special_convex_quadrilaterals_2nd_order(ndiv);

%[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh_2ndorder(ndiv);
[coord,gcoord]=coordinate_arbitrarytriangle_2ndorder(xx,yy,ndiv)

[nel,nnel]=size(nodes)

for i=1:nel
NN(i,1)=i;
end

table1=[NN nodes]

[nnode,dimension]=size(gcoord)
%plot the mesh for the generated data
%x and y coordinates
xcoord(1:nnode,1)=gcoord(1:nnode,1);
ycoord(1:nnode,1)=gcoord(1:nnode,2);
%extract coordinates for each element

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
switch tri

case 1
axis tight
xmin=0;xmax=1;ymin=0;ymax=1;
axis([xmin,xmax,ymin,ymax]);

case 2
axis tight
xmin=0;xmax=1/2;ymin=0;ymax=1/2;
axis([xmin,xmax,ymin,ymax]);

case 3
axis tight
xmin=0;xmax=1;ymin=0;ymax=1;
axis([xmin,xmax,ymin,ymax]);
end
plot(xvec,yvec);%plot element
hold on;
%place element number
midx=mean(xvec(1,1:4));
```

```
midy=mean(yvec(1,1:4));
if mesh<=5
text(midx,midy,['\bf(',num2str(i),'\bf)']);
end
end;%i loop
xlabel('\bfx axis')
ylabel('\bfy axis')

switch tri
case 1
st1='\bfstandard triangle  ';
st2=' using  ';
st3='8-node parabolic ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
text(1,1.5,['\bfMESH NO.=',num2str(mesh)])
text(1,1.7,['\bfnumber of elements=',num2str(nel)])
text(1,1.9,['\bfnumber of nodes=',num2str(nnode)])

case 2
st1='\bfone eigth (1/8)square cross section ';
st2=' using  ';
st3='8-node parabolic ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
text(0.1,0.4,['\bfMESH NO.=',num2str(mesh)])
text(0.1,0.38,['\bfnumber of elements=',num2str(nel)])
text(0.1,0.36,['\bfnumber of nodes=',num2str(nnode)])
case 3
st1='\bfequilateral triangle  ';
st2=' using  ';
st3='8-node parabolic ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
text(0.6,0.8,['\bfMESH NO.=',num2str(mesh)])
text(0.6,0.75,['\bfnumber of elements=',num2str(nel)])
text(0.6,0.70,['\bfnumber of nodes=',num2str(nnode)])

   case 4
st1='\bfequilateral triangle  ';
st2=' using  ';
st3='8-node parabolic ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
text(1,1.8,['\bfMESH NO.=',num2str(mesh)])
text(1,1.6,['\bfnumber of elements=',num2str(nel)])
text(1,1.4,['\bfnumber of nodes=',num2str(nnode)])
```

DOI: 10.18535/ijecs/v6i10.13

```
end

%put node numbers
for jj=1:nnode
if mesh<=5
text(gcoord(jj,1),gcoord(jj,2),['\bfo',num2str(jj)]);
else
text(gcoord(jj,1),gcoord(jj,2),['\bfo']);
end
end
hold on
%axis off
end%for nmesh-the number of meshes
```

(2)*************

```
function[eln,nodetel,nodes,nnode]=nodaladdresses_special_convex_quadrilaterals_2nd_order(n)
%division of a standard triangle(right isoscles triangle)
%into eight node special_convex_quadrilaterals
for nelm=1:3*(n/2)^2
   spqd(nelm,1:8)=0;
end
%disp('vertex nodes of triangle')
elm(1,1)=1;
elm(n+1,1)=2;
elm((n+1)*(n+2)/2,1)=3;
%disp('vertex nodes of triangle')
kk=3;
for k=2:n
   kk=kk+1;
   elm(k,1)=kk;
end
%disp('left edge nodes')
nni=1;
for i=0:(n-2)
   nni=nni+(n-i)+1;
   elm(nni,1)=3*n-i;
end
%disp('right edge nodes')
nni=n+1;
for i=0:(n-2)
   nni=nni+(n-i);
   elm(nni,1)=(n+3)+i;
end

%disp('interior nodes')
nni=1;jj=0;
for i=0:(n-3)
   nni=nni+(n-i)+1;
   for j=1:(n-2-i)
      jj=jj+1;
      nnj=nni+j;
```

```
        elm(nnj,1)=3*n+jj;
    end
end
%disp(elm)
%disp(length(elm))

jj=0;kk=0;
for j=0:n-1
    jj=j+1;
for k=1:(n+1)-j
    kk=kk+1;
    row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=3;
%for jj=(n+1):-1:1
 %   disp(row_nodes(jj,:))
 %end
[row_nodes]
rr=row_nodes;
rrr(:,:,1)=rr;
%rr
%disp('element computations')
if rem(n,2)==0
ne=0;N=n+1;

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

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

nnd=(n+1)*(n+2)/2;
for kkk=1:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
end
%for kk=1:ne
%[eln(kk,1:7)]
%end
%to generate special quadrilaterals
mm=0;
for iel=1:ne
   for jel=1:3
   mm=mm+1;
      switch jel
       case 1
        nodes(mm,1:4)=[eln(iel,7) eln(iel,6) eln(iel,1) eln(iel,4)];
        nodetel(mm,1:3)=[eln(iel,2) eln(iel,3) eln(iel,1)];
        case 2
        nodes(mm,1:4)=[eln(iel,7) eln(iel,4) eln(iel,2) eln(iel,5)];
        nodetel(mm,1:3)=[eln(iel,3) eln(iel,1) eln(iel,2)];
        case 3
        nodes(mm,1:4)=[eln(iel,7) eln(iel,5) eln(iel,3) eln(iel,6)];
        nodetel(mm,1:3)=[eln(iel,1) eln(iel,2) eln(iel,3)];
      end
    end
end


%for mmm=1:mm
   %spqd(:,1:4)
 %end
%mesh generation of eight node special quadrilaterals



for inum=1:nnd
   for jnum=1:nnd
      mdpt(inum,jnum)=0;
   end
end
nd=nnd;
for mmm=1:mm
   mmm1=nodes(mmm,1);
   mmm2=nodes(mmm,2);
   mmm3=nodes(mmm,3);
```

DOI: 10.18535/ijecs/v6i10.13

```
    mmm4=nodes(mmm,4);
%midpoint side-1 of 4-node special quadrilateral
if((mdpt(mmm1,mmm2)==0)&(mdpt(mmm2,mmm1)==0))
    nd=nd+1;
   mdpt(mmm1,mmm2)=nd;
   mdpt(mmm2,mmm1)=nd;
end
%midpoint side-2 of 4-node special quadrilateral
if((mdpt(mmm2,mmm3)==0)&(mdpt(mmm3,mmm2)==0))
    nd=nd+1;
   mdpt(mmm2,mmm3)=nd;
   mdpt(mmm3,mmm2)=nd;
end
%midpoint side-3 of 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
    nd=nd+1;
   mdpt(mmm3,mmm4)=nd;
   mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
    nd=nd+1;
   mdpt(mmm4,mmm1)=nd;
   mdpt(mmm1,mmm4)=nd;
end
   nodes(mmm,5)=mdpt(mmm1,mmm2);
   nodes(mmm,6)=mdpt(mmm2,mmm3);
   nodes(mmm,7)=mdpt(mmm3,mmm4);
   nodes(mmm,8)=mdpt(mmm4,mmm1);
end
nnode=nd;
nel=mm;
```

(3)***************

```
function[coord,gcoord]=coordinate_arbitrarytriangle_2ndorder(x,y,n)
syms ui vi wi xi yi
x1=x(1,1);x2=x(2,1);x3=x(3,1);
y1=y(1,1);y2=y(2,1);y3=y(3,1);
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order(n);
%disp([ui vi wi])
N=length(ui);
    NN=(1:N)';
for i=1:N
    xi(i,1)=x1*wi(i,1)+x2*ui(i,1)+x3*vi(i,1);
    yi(i,1)=y1*wi(i,1)+y2*ui(i,1)+y3*vi(i,1);
end
%disp('________________________________________________________')
%disp('NN   xi    yi')
%disp([NN xi yi])
%disp('________________________________________________________')
coord(:,1)=(xi(:,1));
coord(:,2)=(yi(:,1));
gcoord(:,1)=double(xi(:,1));
```

```
gcoord(:,2)=double(yi(:,1));
%disp(gcoord);
```

(4)**************

```
function[]=D2LaplaceEquationQ8Ex3automeshgenNew(n1,n2,n3,numtri,ndiv,mesh)
%function[]=improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel,nnode,nnel,ndof,quad
type,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)]))
%syms ff ss f sk N NN table1 table2
%D2LaplaceEquationQ8Ex3automeshgenNew(n1=1,n2=2,n3=3,numtri=1,ndiv=2,mesh=1)
%D2LaplaceEquationQ8Ex3automeshgenNew(1,2,3,1,2,1)
syms coord
syms x y
ndof=1;


switch mesh
   case 1
   x=sym([0;1/2;1/2])
   y=sym([0;0;1/2])
   case 2  %isoscles triangle(torsion of an equilateral triangle,each side=2*sqrt(3))
 x=sym([-sqrt(3);sqrt(3); 0])
 y=sym([    -1;      -1; 2])


end
syms ui vi wi xi yi
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order(ndiv);
%disp([ui vi wi])
N=length(ui);
   NN=(1:N)';
   x
   y
   x1=x(n1,1);x2=x(n2,1);x3=x(n3,1);y1=y(n1,1);y2=y(n2,1);y3=y(n3,1);
for i=1:N
   xxi(i,1)=x1+(x2-x1)*ui(i,1)+(x3-x1)*vi(i,1);
   yyi(i,1)=y1+(y2-y1)*ui(i,1)+(y3-y1)*vi(i,1);
end
%disp('_____________________________________________________')
%disp('NN   xi    yi')
%disp([NN xi yi])
%disp('_____________________________________________________')
coord(:,1)=(xxi(:,1));
coord(:,2)=(yyi(:,1));
gcoord(:,1)=double(xxi(:,1));
gcoord(:,2)=double(yyi(:,1));
%disp(gcoord);
[eln,nodetel,nodes,nnode]=nodaladdresses_special_convex_quadrilaterals_2nd_order(ndiv);
%[coord,gcoord]=coordinate4generaltriangle_2ndorder(x,y,n1,n2,n3,ndiv)
```

DOI: 10.18535/ijecs/v6i10.13

```
%[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh_2ndorder(ndiv);
   %[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh(ndiv);
   %[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
[nel,nnel]=size(nodes);
%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

sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
%syms r s
%syms xa xb xc
%syms ya yb yc

%nnode=17,nel=12,nnel=4,ndof=1
%>>LaplaceEquationQuad4twodimension(12,17,4,1)
%
%Ex1:nnode=41,nel=36,,nnel=4,nodf=1
%>>LaplaceEquationQuad4twodimensionEx1(36,41,4,1)
%>>improvedLaplaceEquationQuad4twodimensionEx1_explicit(36,41,4,1)
%Ex2:nnode=83,nel=69,,nnel=4,nodf=1
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicit(69,83,4,1)#
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicitfnmesh(69,83,4,1)#
%improvedLaplaceEquationQuad4twodimensionEx2_explicitvfnmesh(72,87,4,1)#new
%improvedLaplaceEquationQuad8twodimensionEx2_explicitvfnmesh(72,245,8,1)#new
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=3,nnode=16,nnel=8,ndof=1,quad
type=3,mesh=1)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=9,nnode=34,nnel=8,ndof=1,quad
type=3,mesh=2)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=12,nnode=49,nnel=8,ndof=1,qua
dtype=3,mesh=3)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=27,nnode=100,nnel=8,ndof=1,q
uadtype=3,mesh=4)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=48,nnode=169,nnel=8,ndof=1,q
uadtype=3,mesh=5)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=75,nnode=256,nnel=8,ndof=1,q
uadtype=3,mesh=6)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=108,nnode=361,nnel=8,ndof=1,
quadtype=3,mesh=7)
%improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel=147,nnode=484,nnel=8,ndof=1,
quadtype=3,mesh=8)
%disp([nel nnode nnel ndof quadtype mesh])
format long g
for i=1:nel
N(i,1)=i;
end
%radius of the hole=1.25cm
```

```
%input data for nodal coordinate values
%gcoord(i,j),where i->node no. and j->x or y

table1=[N nodes];
[nel,nnel]=size(nodes);
switch mesh
   case 1
   nnn=0;
   for nn=1:nnode
      if gcoord(nn,1)==(1/2)
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
      end
   end
   format long g
k1 =double(0.14057701495515551037840396020329);
 xi=(zeros(nnode,1));
a0=8/pi^3;
for m=1:nnode
   gx=(gcoord(m,1));gy=(gcoord(m,2));rr=(0);
for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*gy)/cosh(n*pi/2)))*cos(n*pi*gx)/n^3;
end
xi(m,1)=(a0*rr);
end
 mm=length(bcdof);

   case 2%torsion of an equilateral triangle

 nnn=0;
 %boundary conditions on side 1
  for nn=1:nnode
     xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if ((ynn+1)<1.e-5)
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
  end
%boundary conditions on side 2
  for nn=1:nnode
     xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if ((-(sqrt(3))*xnn-ynn+2)<1.e-5)
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
  end
%boundary conditions on side 3
  for nn=1:nnode
     xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (((sqrt(3))*xnn-ynn+2)<1.e-5)
```

```
          nnn=nnn+1
          bcdof(nnn,1)=nn;
          bcval(nnn,1)=0;
        end
  end
  bcdof
  bcval
  mm=length(bcdof);
 for m=1:nnode
    gx=(gcoord(m,1));gy=(gcoord(m,2));
    xi(m,1)=((gy+1)*((sqrt(3))*gx-gy+2)*(-(sqrt(3))*gx-gy+2))/12;
  end
xi=double(xi);
format long g
k1 =9*sqrt(3)/5;

end%switch


%___________________________________________________________________________
 % disp(gcoord)


   %________________________________________________________
 %quadtype=3:quadrilateral elements of special shape
 %quadtype=0:quadrilateral elements of arbitrary shape
 %for el=1:nel
  %   elmtype(el,1)=quadtype;%change 0 to 3 to take advantage of special shape
  %end

 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
%____________________________________________________________________

table2=[N xx yy];
%disp([xx yy])
intJdn1dn1uvrs =[vpa(sym(' 1.19732437518704939126225670841')),vpa(sym('
1.07234243081152493747384516139'));...
                 vpa(sym(' 1.07234243081152493747384516139')),vpa(sym('
1.19732437518704939126225670841'))];

intJdn1dn2uvrs =[vpa(sym('    .393282075247773712718727744686')),vpa(sym('
.665055030187779896040365029 27e-1'));...
                 vpa(sym('   .233172169685444656270703169594')),vpa(sym('
.309018301898183977243469181 43'))];
```

```
intJdn1dn3uvrs =[vpa(sym(' .39105823773230516010581465792')),vpa(sym('
.34520910581966151866889237468'));...
              vpa(sym(' .34520910581966151866889237468')),vpa(sym('
.39105823773230516010581465792'))];

intJdn1dn4uvrs =[vpa(sym('    .30901830189818397724346918143')),vpa(sym('
.23317216968544465627070316959 4'));...
              vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.39328207524777371271872744686'))];

intJdn1dn5uvrs =[vpa(sym(' -1.2060191445619956476192751 5849')),vpa(sym('  -
.19480639850558017529001262923'));...
              vpa(sym('  -.86147306517224684195667929589')),vpa(sym('  -
.27406792898205548885083220480'))];

intJdn1dn6uvrs =[vpa(sym('  -.19025279048691687518778377091')),vpa(sym('  -
.33047487282879104238539264173'));...
              vpa(sym('  -.33047487282879104238539264173')),vpa(sym(' -
.62034312603434422967237686041 0'))];

intJdn1dn7uvrs =[vpa(sym(' -.62034312603434422967237686041 0')),vpa(sym('  -
.33047487282879104238539264173'));...
              vpa(sym('  -.33047487282879104238539264173')),vpa(sym('  -
.19025279048691687518778377091'))];

intJdn1dn8uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.86147306517224684195667929589'));...
              vpa(sym('  -.19480639850558017529001262923')),vpa(sym(' -
1.2060191445619956476192751 5849'))];

intJdn2dn1uvrs =[vpa(sym('    .39328207524777371271872744686')),vpa(sym('
.23317216968544465627070316959 4'));...
              vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.30901830189818397724346918143'))];

intJdn2dn2uvrs =[vpa(sym('   .42652636618519994053203031884')),vpa(sym(' -
.26351356567528356682253632347 8'));...
              vpa(sym(' -.26351356567528356682253632347 8')),vpa(sym('
.45005030410782929514068096460'))];

intJdn2dn3uvrs =[vpa(sym('  .15950576453713864866357463072 58')),vpa(sym('
.15687259169421065039078881847e-1'));...
              vpa(sym('   .18235392583608773170574554851 3')),vpa(sym('
.31434135165461644796478483341 8'))];

intJdn2dn4uvrs =[vpa(sym('     .20218069152088195770651481813')),vpa(sym(' -
.97977735817538970518647437153e-1'));...
              vpa(sym(' -.97977735817538970518647437153e-1')),vpa(sym('
.20218069152088195770651481813'))];

intJdn2dn5uvrs =[vpa(sym('  -.92334437129978410949228855299')),vpa(sym('  -
.21347210674508644538200765866'));...
```

vpa(sym(' .453194559921580221284659008 01')),vpa(sym(' -.57472646168610406836969542 2e-2'))];

intJdn2dn6uvrs =[vpa(sym(' .21936235900108010796746055 6119')),vpa(sym(' .32421276181069833859048628 84652'));...
vpa(sym(' -.34245390485596832807618037 82014')),vpa(sym(' -.76880656144973102466708297 9907'))];

intJdn2dn7uvrs =[vpa(sym(' -.39068702588898312284200391 5751')),vpa(sym(' -.36332930895951737339235932 476e-1'));...
vpa(sym(' -.36332930895951737339235932 476e-1')),vpa(sym(' -.33760644045146244838888795 56e-1'))];

intJdn2dn8uvrs =[vpa(sym(' -.86825859303307135254015301 93e-1')),vpa(sym(' .38224148468296660162159011 855e-1'));...
vpa(sym(' .38224148468296660162159011 855e-1')),vpa(sym(' -.46727617906977336786578106 7869'))];

intJdn3dn1uvrs =[vpa(sym(' .39105823773230516010581465 792')),vpa(sym(' .34520910581966151866688923 7468'));...
vpa(sym(' .34520910581966151866688923 7468')),vpa(sym(' .39105823773230516010581465 792'))];

intJdn3dn2uvrs =[vpa(sym(' .15950576453713864866357463 07258')),vpa(sym(' .18235392583608773170574554 8513'));...
vpa(sym(' .15687259169421065039078881 847e-1')),vpa(sym(' .31434135165461644796478483 3418'))];

intJdn3dn3uvrs =[vpa(sym(' .66011274047258975940889336 4145')),vpa(sym(' .61928481956723524193595128 6773'));...
vpa(sym(' .61928481956723524193595128 6773')),vpa(sym(' .66011274047258975940889336 4145'))];

intJdn3dn4uvrs =[vpa(sym(' .31434135165461644796478483 3418')),vpa(sym(' .15687259169421065039078881 847e-1'));...
vpa(sym(' .18235392583608773170574554 8513')),vpa(sym(' .15950576453713864866357463 07258'))];

intJdn3dn5uvrs =[vpa(sym(' -.44732398802815826593641518 5565')),vpa(sym(' -.18278932817664103222138516 3017'));...
vpa(sym(' -.18278932817664103222138516 3017')),vpa(sym(' -.15179959092659718711045271 183'))];

intJdn3dn6uvrs =[vpa(sym(' .63171946139777238327583118 23e-2')),vpa(sym(' -.73181156035289507978678221 6220'));...
vpa(sym(' -.65144893686228413120115549 553e-1')),vpa(sym(' -.93221171005587228692895790 0634'))];

intJdn3dn7uvrs =[vpa(sym(' -.93221171005587228692895790 0634')),vpa(sym(' -.65144893686228413120115549 553e-1'));...
vpa(sym(' -.73181156035289507978678221 6220')),vpa(sym(' .63171946139777238327583118 23e-2'))];

DOI: 10.18535/ijecs/v6i10.13

```
intJdn3dn8uvrs =[vpa(sym('  -.15179959092659718711045271183')),vpa(sym(' -
.18278932817664103222138516301 7'));...
              vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -
.44732398802815826593641518556 5'))];

intJdn4dn1uvrs =[vpa(sym('    .30901830189818397724346918143')),vpa(sym('
.66505503018777989604036502927e-1'));...
              vpa(sym(' .23317216968544465627070316959 4')),   vpa(sym('
.39328207524777371271872744686'))];

intJdn4dn2uvrs =[vpa(sym('     .20218069152088195770651481813')),vpa(sym(' -
.97977735817538970518647437153e-1'));...
              vpa(sym(' -.97977735817538970518647437153e-1')),vpa(sym('
.20218069152088195770651481813'))];

intJdn4dn3uvrs =[vpa(sym('   .31434135165461644796478483341 8')),vpa(sym('
.18235392583608773170574554851 3'));...
              vpa(sym(' .15687259169421065039078881847e-1')),vpa(sym('
.15950576453713864866357463072 58'))];

intJdn4dn4uvrs =[vpa(sym('   .45005030410782929514068096460')),vpa(sym(' -
.26351356567528356682253632347 8'));...
              vpa(sym(' -.26351356567528356682253632347 8')),vpa(sym('
.42652636618519994053203031884'))];

intJdn4dn5uvrs =[vpa(sym('  -.46727617906977336786578106786 9')),vpa(sym('
.38224148468296660162159011855e-1'));...
              vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym(' -
.86825859303307135254015301 93e-1'))];

intJdn4dn6uvrs =[vpa(sym('  -.33760644045146244838888795 56e-1')),vpa(sym(' -
.36332930895951737339235932476e-1'));...
              vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym('   -
.39068702588898312284200391575 1'))];

intJdn4dn7uvrs =[vpa(sym('  -.76880656144973102466708297990 7')),vpa(sym(' -
.34245390485596832807618037820 14'));...
              vpa(sym('  .32421276181069833859048628846 52')),vpa(sym('
.21936235900108010796746055611 9'))];

intJdn4dn8uvrs =[vpa(sym(' -.57472646168610406836969542 2e-2')),vpa(sym('
.45319455992158022128465900801'));...
              vpa(sym('  -.21347210674508644538200765866')),vpa(sym('  -
.92334437129978410949228855299'))];

intJdn5dn1uvrs =[vpa(sym(' -1.2060191445619956476192751584 9')),vpa(sym('  -
.86147306517224684195667929589'));...
              vpa(sym('  -.19480639850558017529001262923')),vpa(sym('  -
.27406792898205548885083220480'))];
```

intJdn5dn2uvrs =[vpa(sym(' -.92334437129978410949228855299')),vpa(sym(' .45319455992158022128465900801'));...
vpa(sym(' -.21347210674508644538200765866')),vpa(sym(' -.57472646168610406836969542e-2'))];

intJdn5dn3uvrs =[vpa(sym(' -.44732398802815826593641518556 5')),vpa(sym(' -.18278932817664103222138516301 7'));...
vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -.15179959092659718711045271183'))];

intJdn5dn4uvrs =[vpa(sym(' -.46727617906977336786578106786 9')),vpa(sym(' .38224148468296660162159011855e-1'));...
vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym(' -.86825859303307135254015301 93e-1'))];

intJdn5dn5uvrs =[vpa(sym(' 2.1717715426240022624872702233 6')),vpa(sym(' .52992039898580294331102892478'));...
vpa(sym(' .52992039898580294331102892478')),vpa(sym(' .58827358447153923041425369 63'))];

intJdn5dn6uvrs =[vpa(sym(' -.41594373113635804762610497799 7')),vpa(sym(' -.44624674573735878584568347920 0'));...
vpa(sym(' -.44624674573735878584568347920 0')),vpa(sym(' .34312587200869545409370986785e-1'))];

intJdn5dn7uvrs =[vpa(sym(' .91157772830419213711619683500')),vpa(sym(' .94571793322616328048899852 21e-1'));...
vpa(sym(' .94571793322616328048899852 21e-1')),vpa(sym(' -.48070367101146296286102539491'))];

intJdn5dn8uvrs =[vpa(sym(' .37655814316787503893639788456 0')),vpa(sym(' .37459823838795050721700114125'));...
vpa(sym(' .37459823838795050721700114125')),vpa(sym(' .37655814316787503893639788456 0'))];

intJdn6dn1uvrs =[vpa(sym(' -.19025279048691687518778377091')),vpa(sym(' -.33047487282879104238539264173'));...
vpa(sym(' -.33047487282879104238539264173')),vpa(sym(' -.62034312603434422967237686041 0'))];

intJdn6dn2uvrs =[vpa(sym(' .21936235900108010796746055611 9')),vpa(sym(' -.34245390485596832807618037820 14'));...
vpa(sym(' .32421276181069833859048628846 52')),vpa(sym(' -.76880656144973102466708297990 7'))];

intJdn6dn3uvrs =[vpa(sym(' .63171946139777238327583118 23e-2')),vpa(sym(' -.65144893686228413120115549553e-1'));...
vpa(sym(' -.73181156035289507978678221622 0')),vpa(sym(' -.93221171005587228692895790063 4'))];

intJdn6dn4uvrs =[vpa(sym(' -.33760644045146244838888795 56e-1')),vpa(sym(' -.36332930895951737339235932476e-1'));...

vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym(' -.39068702588898312284200391575l'))];

intJdn6dn5uvrs =[vpa(sym(' -.415943731136358047626104977997')),vpa(sym(' -.446246745737358785845683479200'));...
vpa(sym(' -.446246745737358785845683479200')),vpa(sym(' .34312587200869545409370986785e-1'))];

intJdn6dn6uvrs =[vpa(sym(' .82865433521530100559964914224')),vpa(sym(' .472904354164178470376423509241'));...
vpa(sym(' .472904354164178470376423509241')),vpa(sym(' 1.699831160074343688470918905720'))];

intJdn6dn7uvrs =[vpa(sym(' .66326947849525293113934929200e-1')),vpa(sym(' .653177200517503508341284619711'));...
vpa(sym(' .653177200517503508341284619711')),vpa(sym(' .66326947849525293113934929200e-1'))];

intJdn6dn8uvrs =[vpa(sym(' -.48070367101146296286102539491')),vpa(sym(' .9457179332261632804889985221e-1'));...
vpa(sym(' .9457179332261632804889985221e-1')),vpa(sym(' .91157772830419213711619683500'))];

intJdn7dn1uvrs =[vpa(sym(' -.620343126034344229672376860410')),vpa(sym(' -.33047487282879104238539264173'));...
vpa(sym(' -.33047487282879104238539264173')),vpa(sym(' -.19025279048691687518778377091'))];

intJdn7dn2uvrs =[vpa(sym(' -.390687025888983122842003915751')),vpa(sym(' -.36332930895951737339235932476e-1'));...
vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym(' -.3376064404514624483888879556e-1'))];

intJdn7dn3uvrs =[vpa(sym(' -.932211710055872286928957900634')),vpa(sym(' -.731811560352895079786782216220'));...
vpa(sym(' -.65144893686228413120115549553e-1')),vpa(sym(' .6317194613977723832758311823e-2'))];

intJdn7dn4uvrs =[vpa(sym(' -.768806561449731024667082979907')),vpa(sym(' .3242127618106983385904862884652'));...
vpa(sym(' -.3424539048559683280761803782014')),vpa(sym(' .219362359001080107967460556119'))];

intJdn7dn5uvrs =[vpa(sym(' .91157772830419213711619683500')),vpa(sym(' .9457179332261632804889985221e-1'));...
vpa(sym(' .9457179332261632804889985221e-1')),vpa(sym(' -.48070367101146296286102539491'))];

intJdn7dn6uvrs =[vpa(sym(' .66326947849525293113934929200e-1')),vpa(sym(' .653177200517503508341284619711'));...
vpa(sym(' .653177200517503508341284619711')),vpa(sym(' .66326947849525293113934929200e-1'))];

```
intJdn7dn7uvrs =[vpa(sym(' 1.6998311600743436884709189057 20')),vpa(sym('
.47290435416417847037642350924 1'));...
            vpa(sym('  .47290435416417847037642350924 1')),vpa(sym('
.82865433521530100559964914224'))];

intJdn7dn8uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.44624674573735878584568347920 0'));...
            vpa(sym('  -.44624674573735878584568347920 0')),vpa(sym('  -
.41594373113635804762610497799 7'))];

intJdn8dn1uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.19480639850558017529001262923'));...
            vpa(sym('  -.86147306517224684195667929589')),vpa(sym(' -
1.2060191445619956476192751584 9'))];

intJdn8dn2uvrs =[vpa(sym(' -.86825859303307135254015301 93e-1')),vpa(sym('
.38224148468296660162159011855e-1'));...
            vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym('  -
.46727617906977336786578106786 9'))];

intJdn8dn3uvrs =[vpa(sym('  -.15179959092659718711045271183')),vpa(sym(' -
.18278932817664103222138516301 7'));...
            vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -
.44732398802815826593641518556 5'))];

intJdn8dn4uvrs =[vpa(sym(' -.57472646168610406836969542 2e-2')),vpa(sym('  -
.21347210674508644538200765866'));...
            vpa(sym('   .45319455992158022128465900801')),vpa(sym('  -
.92334437129978410949228855299'))];

intJdn8dn5uvrs =[vpa(sym(' .37655814316787503893639788456 0')),vpa(sym('
.37459823838795050721700114125'));...
            vpa(sym('  .37459823838795050721700114125')),vpa(sym('
.37655814316787503893639788456 0'))];

intJdn8dn6uvrs =[vpa(sym('  -.48070367101146296286102539491')),vpa(sym('
.94571793322616328048899852 21e-1'));...
            vpa(sym(' .94571793322616328048899852 21e-1')),vpa(sym('
.91157772830419213711619683500'))];

intJdn8dn7uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.44624674573735878584568347920 0'));...
            vpa(sym('  -.44624674573735878584568347920 0')),vpa(sym('  -
.41594373113635804762610497799 7'))];

intJdn8dn8uvrs =[vpa(sym('   .58827358447153923041425369 63')),vpa(sym('
.52992039898580294331102892478'));...
            vpa(sym('  .52992039898580294331102892478')),vpa(sym('
2.1717715426240022624872702233 6'))];

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

```
intJdndn=[intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs intJdn1dn5uvrs intJdn1dn6uvrs
intJdn1dn7uvrs intJdn1dn8uvrs;...
        intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs intJdn2dn5uvrs intJdn2dn6uvrs
intJdn2dn7uvrs intJdn2dn8uvrs;...
        intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs intJdn3dn5uvrs intJdn3dn6uvrs
intJdn3dn7uvrs intJdn3dn8uvrs;...
        intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs intJdn4dn5uvrs intJdn4dn6uvrs
intJdn4dn7uvrs intJdn4dn8uvrs;...
        intJdn5dn1uvrs intJdn5dn2uvrs intJdn5dn3uvrs intJdn5dn4uvrs intJdn5dn5uvrs intJdn5dn6uvrs
intJdn5dn7uvrs intJdn5dn8uvrs;...
        intJdn6dn1uvrs intJdn6dn2uvrs intJdn6dn3uvrs intJdn6dn4uvrs intJdn6dn5uvrs intJdn6dn6uvrs
intJdn6dn7uvrs intJdn6dn8uvrs;...
        intJdn7dn1uvrs intJdn7dn2uvrs intJdn7dn3uvrs intJdn7dn4uvrs intJdn7dn5uvrs intJdn7dn6uvrs
intJdn7dn7uvrs intJdn7dn8uvrs;...
        intJdn8dn1uvrs intJdn8dn2uvrs intJdn8dn3uvrs intJdn8dn4uvrs intJdn8dn5uvrs intJdn8dn6uvrs
intJdn8dn7uvrs intJdn8dn8uvrs];




intJdndn=double(intJdndn);



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(1:8,1:8)=(zeros(8,8));
for i=1:8
 for j=i:8
 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);
f=[ -7/432;  -1/72; -5/432;  -1/72; 11/216; 13/216; 13/216; 11/216]*(2*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;
phi=double(phi);
if mesh==2
    phi=phi/2;
end
[phi xi]
for I=1:nnode
NN(I,1)=I;
phi_xi(I,1)=phi(I,1)-xi(I,1);
end
MAXPHI_XI=max(abs(double(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('node number        Prandtl Stress Values')
%disp('                    fem-computed values           anlytical(theoretical)-values          ')

%disp([NN phi xi phi_xi])
t=0;
 for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
 end
switch mesh
    case 1
T=8*t;
    case 2
        T=2*t;
end

disp('--------------------------------------------------------------------------------------------')
disp('number of nodes,elements & nodes per element')
disp([nnode nel nnel ])
disp('torisonal constants(fem=phi&exact=xi)  error(max(abs(phi_xi))')
%disp('--------------------------------------------------------------------------------------------')
%disp([nnode nel nnel ])
disp([T k1 MAXPHI_XI ])
disp('--------------------------------------------------------------------------------------------')

if (mesh==2)

[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);
        nd5=nodes(iel,5);nd6=nodes(iel,6);nd7=nodes(iel,7);nd8=nodes(iel,8);
 %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);
```

DOI: 10.18535/ijecs/v6i10.13

```
shn1=((1-r)*(1-s)*(-1-r-s))/4;
           shn2=((1+r)*(1-s)*(-1+r-s))/4;
           shn3=((1+r)*(1+s)*(-1+r+s))/4;
           shn4=((1-r)*(1+s)*(-1-r+s))/4;
           shn5=((1-s)*(1-r^2))/2;
           shn6=((1+r)*(1-s^2))/2;
           shn7=((1+s)*(1-r^2))/2;
           shn8=((1-r)*(1-s^2))/2;
PHI(i,j)=shn1*phi(nd1,1)+shn2*phi(nd2,1)+shn3*phi(nd3,1)+shn4*phi(nd4,1)+shn5*phi(nd5,1)+shn6*phi(
nd6,1)+shn7*phi(nd7,1)+shn8*phi(nd8,1);

%            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==2)

switch mesh
    case 2
  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='Eight 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='Eight 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
end
 [femsoln exactsoln]
```

```
disp('-------------------------------------------------------------------------------------------')
disp('number of nodes,elements & nodes per element')
disp([nnode nel nnel ])
disp('torisonal constants(fem=phi&exact=xi)  error(max(abs(phi_xi))')
%disp('-------------------------------------------------------------------------------------------')
%disp([nnode nel nnel ])
disp([T k1 MAXPHI_XI ])
disp('-------------------------------------------------------------------------------------------')
```

(5)****************

```
function[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order(n)
%n must be even:n=2,4,6,......
syms ui vi wi
ui(1:3,1)=[0;1;0];
vi(1:3,1)=[0;0;1];
wi(1:3,1)=[1;0;0];
if (n-1)>0
 kk=3;
 for i=1:n-1
   kk=kk+1;
   ui(kk,1)=sym(i/n);
   vi(kk,1)=sym(0);
   wi(kk,1)=sym(1-ui(kk,1)-vi(kk,1));
 end
 kkk=kk;
 for ii=1:n-1
   kkk=kkk+1;
   ui(kkk,1)=sym((n-ii)/n);
   vi(kkk,1)=sym(1-(n-ii)/n);
   wi(kkk,1)=0;
 end;
 kkkk=kkk;
  for iii=1:n-1
   kkkk=kkkk+1;
   ui(kkkk,1)=0;
   vi(kkkk,1)=sym(1-iii/n);
   wi(kkkk,1)=sym(iii/n);
  end
end%if (n-1)>0
if (n-2)>0
  kkkkk=kkkk;
  for iiii=1:(n-2)
   for jjjj=1:(n-1)-iiii
        kkkkk=kkkkk+1;
        ui(kkkkk,1)=sym(jjjj/n);
        vi(kkkkk,1)=sym(iiii/n);
        wi(kkkkk,1)=sym(1-ui(kkkkk,1)-vi(kkkkk,1));
    end
  end
end%if (n-2)>0
if n==2

   num=(1:6)';
```

```
else
  num=(1:kkkkk)';

end
%disp([ui'])
%disp([vi'])
%disp([wi'])
%length(ui)
%length(vi)
%length(wi)

%disp([num ui vi wi])
[eln,nodetel,spqd,nnode]=nodaladdresses_special_convex_quadrilaterals_2nd_order(n)
qq=(n+1)*(n+2)/2;
    nc=(n/2)^2;
for pp=1:nc
 qq=qq+1;
 q1=eln(pp,1);
 q2=eln(pp,2);
 q3=eln(pp,3);
 ui(qq,1)=(ui(q1,1)+ui(q2,1)+ui(q3,1))/3;
 vi(qq,1)=(vi(q1,1)+vi(q2,1)+vi(q3,1))/3;
 wi(qq,1)=1-ui(qq,1)-vi(qq,1);
end
 %disp([ui vi wi])
%length(ui)
%length(vi)
%length(wi)

 num=(1:qq)';
%disp([num ui vi wi])
qqq=qq;
for ppp=1:3*nc
qq1=spqd(ppp,1);
qq2=spqd(ppp,2);
qq3=spqd(ppp,3);
qq4=spqd(ppp,4);
%midside nodes-1,2
 qqq=spqd(ppp,5);
 ui(qqq,1)=(ui(qq1,1)+ui(qq2,1))/2;
 vi(qqq,1)=(vi(qq1,1)+vi(qq2,1))/2;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
%midside nodes-2,3
 qqq=spqd(ppp,6);
 ui(qqq,1)=(ui(qq2,1)+ui(qq3,1))/2;
 vi(qqq,1)=(vi(qq2,1)+vi(qq3,1))/2;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 %midside nodes-3,4
 qqq=spqd(ppp,7);
 ui(qqq,1)=(ui(qq3,1)+ui(qq4,1))/2;
 vi(qqq,1)=(vi(qq3,1)+vi(qq4,1))/2;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
```

DOI: 10.18535/ijecs/v6i10.13

```
%midside nodes-4,1
 qqq=spqd(ppp,8);
 ui(qqq,1)=(ui(qq1,1)+ui(qq4,1))/2;
 vi(qqq,1)=(vi(qq1,1)+vi(qq4,1))/2;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
end
maxnode=max(max(spqd(1:3*nc,1:8)));
 num=(1:maxnode)';
 disp(['maximum value of node number=',num2str(maxnode)])
disp(' node    ui     vi     wi')
disp([num ui vi wi])
```

(II)COMPUTER PROGRAMS:LINEAR CONVEX POLYGONAL D0MAINS

(1)*************

```
 function[]=quadrilateral_mesh4MOINEX_q8(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;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)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh
)
%quadrilateral_mesh4MOINEX_q8([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_q8([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,3,1,1)
%quadrilateral_mesh4MOINEX_q8([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)
%
%[nel,nnel]=size(nodes);
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order(n1,n2,n3,nmax,numtri,
ndiv,mesh)
[nel,nnel]=size(nodes);
disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where i->node no. and j->x or y
%___________________________________________________________________

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

```
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
%axis equal
axis tight
if mesh==1
axis([0 xlength 0 ylength])
end
if mesh==2
axis([0 xlength 0 ylength])
end
if mesh==3
axis([-1/2 xlength-1/2 -1/2 ylength-1/2])
end
if mesh==4
axis([-1/2 xlength-1/2 -1/2 ylength-1/2])
end
plot(xvec,yvec);%plot element
hold on;
%place element number
if (ndiv==2)
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
text(midx,midy,['[',num2str(i),']']);
end
end;%i loop
xlabel('x axis')
ylabel('y axis')
st1='Mesh with ';
st2=num2str(nel);
st3='eight noded ';
st4='quadrilateral ';
st5='elements & no.of '
st6='nodes=';
st7=num2str(nnode);
title([st1,st2,st3,st4,st5,st6,st7])
%put node numbers
disp(nnode)
if (ndiv==2)
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o',num2str(jj)]);
end
end
if (ndiv>2)
for jj=1:nnode
text(gcoord(jj,1),gcoord(jj,2),['o']);
end
end
```

(2)****************

```
function[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order(n1,n2,n3,nmax,
numtri,n,mesh)
```

DOI: 10.18535/ijecs/v6i10.13

```
%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)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([1;1;1;1],[2;3;4;5],[3;
4;5;2],5,1,2,1)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([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)
%PARVIZ MOIN EXAMPLE
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([1;1;1;1;1;1;1],[2;3;4
;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,3)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([1;1;1;1;1;1;1],[2;3;4
;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,3)
syms U V W xi yi
syms x y
switch mesh
case 1%domain with seven triangles(8-nodes)
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%square domain with eight triangles(9-nodes)
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     9
  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%square domain with four triangles(5-nodes)
 x=sym([1/2;0;1;1;0])
 y=sym([1/2;0;0;1;1])
case 50%isoscles triangle(torsion of an equilateral triangle,each side=1 unit)
 x=sym([-1/2;1/2;       0])
 y=sym([   0;  0;sqrt(3)/2])

end
%
```

```
if nmax>3
[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_2nd_order(n1,n2,n3,nmax,
numtri,n);
end
%if nmax==3
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses4special_convex_quadrilaterals_2nd_order(n1,n2,n3,nmax,nu
mtri,n)
%end
[U,V,W]=generate_area_coordinate_over_the_standard_triangle(n);

ss1='number of 6-node triangles with centroid=';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
%
eln
%
ss2='number of special convex quadrilaterals elements&nodes per element =';
[nel,nnel]=size(spqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
%
spqd
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])

xi(1:nnode,1)=zeros(nnode,1);yi(1:nnode,1)=zeros(nnode,1);
if nmax>3
nitri=nmax-1;
end
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%for jj
end%for ii
[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 %for iii
 [xi yi]

end%for itri=1:nitri
for mmm=1:nel
    mmm1=nodes(mmm,1)
    mmm2=nodes(mmm,2)
    mmm3=nodes(mmm,3)
    mmm4=nodes(mmm,4)
   mmm5=nodes(mmm,5)
   mmm6=nodes(mmm,6)
   mmm7=nodes(mmm,7)
   mmm8=nodes(mmm,8)
   xi1=xi(mmm1,1)
   xi2=xi(mmm2,1)
   xi3=xi(mmm3,1)
   xi4=xi(mmm4,1)
   %(xi1+xi2)/2
   %
   yi1=yi(mmm1,1)
   yi2=yi(mmm2,1)
   yi3=yi(mmm3,1)
   yi4=yi(mmm4,1)
   %(yi1+yi2)/2
 xi(mmm5,1)=(xi1+xi2)/2;
 xi(mmm6,1)=(xi2+xi3)/2;
 xi(mmm7,1)=(xi3+xi4)/2;
 xi(mmm8,1)=(xi4+xi1)/2;
 yi(mmm5,1)=(yi1+yi2)/2;
 yi(mmm6,1)=(yi2+yi3)/2;
 yi(mmm7,1)=(yi3+yi4)/2;
 yi(mmm8,1)=(yi4+yi1)/2;

end%for nel
```

```
%[xi(18,1) yi(18,1)]

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)
```

(3)***************

```
function[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_2nd_order(n1,n2,
n3,nmax,numtri,n)
%n1=node number at(0,0)for  a choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.......i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5
;2],5,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5
;2],5,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5
;2],5,9,6)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5
;2],5,16,8)
%PARVIZ MOIN EXAMPLE
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1],[2;3;4;5;6
;7;8],[3;4;5;6;7;8;2],8,1,2)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1;1;1;1],[2;3;4;5;6
;7;8],[3;4;5;6;7;8;2],8,4,4)
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_2nd_order([1;1;1;1],[2;3
;4;5],[3;4;5;2],5,1,2)
%syms mst_tri x
ne=0;
nitri=nmax-1;
for itri=1:nitri
   elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
if itri==1
kk=nmax;
for k=2:n
```

```
    kk=kk+1
    elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
    elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
    lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
    lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
    mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
    hh=hh+1;
    nni=nni+(n-i);
    elm(nni,1)=(mmax+1)+i;
    qq(hh,1)=(mmax+1)+i;

end
qq(n+1,1)=n3(itri,1);
edgen2n3(1:n+1,itri)=qq;

if itri<nitri
disp('left edge nodes')
nni=1;gg=1;pp(1,1)=n1(itri,1);
for i=0:(n-2)
    gg=gg+1;
    nni=nni+(n-i)+1;
    elm(nni,1)=lmax-i;
    pp(gg,1)=lmax-i;
end
pp(n+1,1)=n3(itri,1);
edgen1n3(1:n+1,itri)=pp
end%if itri<nitri


%if itri==n
% elm(1:n+1,1)=edgen1n2(1:n+1,1)
%end

if itri==nitri
disp('left edge nodes')
```

```
nni=1;gg=1;
for i=0:(n-2)
   gg=gg+1;
   nni=nni+(n-i)+1;
   elm(nni,1)=edgen1n2(gg,1);
end
%pp(n+1,1)=n3(itri,1);
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri




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

jj=0;kk=0;
for j=0:n-1
   jj=j+1;
for k=1:(n+1)-j
   kk=kk+1;
   row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=n3(itri,1);
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
%end
%[row_nodes]
rr=row_nodes;
rr
rrr(:,:,itri)=rr;
disp('element computations')
if rem(n,2)==0
N=n+1;

for k=1:2:n-1
```

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

end% if rem(n,2)==0
ne
%for kk=1:ne
%[eln(kk,1:6)]
%end
%add node numbers for element centroids

nnd=max(max(eln))
if (n>3)
for kkk=1+(itri-1)*numtri:ne
    nnd=nnd+1;
    eln(kkk,7)=nnd;
end
end
if n==2
 for kkk=itri:ne
    nnd=nnd+1;
    eln(kkk,7)=nnd;
 end
end
%for kk=1:ne
%[eln(kk,1:7)]
%end
%to generate special quadrilaterals
%mm=0;
```

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


end%itri

%to generate special quadrilaterals
mm=0;

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

```
for inum=1:nnd
    for jnum=1:nnd
        mdpt(inum,jnum)=0;
    end
end
nd=nnd;
for mmm=1:mm
    mmm1=nodes(mmm,1);
    mmm2=nodes(mmm,2);
    mmm3=nodes(mmm,3);
    mmm4=nodes(mmm,4);
%midpoint side-1 of 4-node special quadrilateral
if((mdpt(mmm1,mmm2)==0)&(mdpt(mmm2,mmm1)==0))
    nd=nd+1;
   mdpt(mmm1,mmm2)=nd;
   mdpt(mmm2,mmm1)=nd;
end
%midpoint side-2 of 4-node special quadrilateral
if((mdpt(mmm2,mmm3)==0)&(mdpt(mmm3,mmm2)==0))
    nd=nd+1;
   mdpt(mmm2,mmm3)=nd;
   mdpt(mmm3,mmm2)=nd;
end
%midpoint side-3 of 4-node special quadrilateral
if((mdpt(mmm3,mmm4)==0)&(mdpt(mmm4,mmm3)==0))
    nd=nd+1;
   mdpt(mmm3,mmm4)=nd;
   mdpt(mmm4,mmm3)=nd;
end
%midpoint side-4 of 4-node special quadrilateral
if((mdpt(mmm4,mmm1)==0)&(mdpt(mmm1,mmm4)==0))
    nd=nd+1;
   mdpt(mmm4,mmm1)=nd;
   mdpt(mmm1,mmm4)=nd;
end
   nodes(mmm,5)=mdpt(mmm1,mmm2);
   nodes(mmm,6)=mdpt(mmm2,mmm3);
   nodes(mmm,7)=mdpt(mmm3,mmm4);
   nodes(mmm,8)=mdpt(mmm4,mmm1);
end
nnode=nd;
nel=mm;
spqd=nodes;
ss1='number of 6-node triangles with centroid=';
[p1,q1]=size(eln);
disp([ss1 num2str(p1)])
%
eln
%
ss2='number of special convex quadrilaterals elements&nodes per element =';
[nel,nnel]=size(spqd);
disp([ss2 num2str(nel) ','  num2str(nnel)])
```

```
%
nnode=max(max(spqd));
ss3='number of nodes of the triangular domain& number of special quadrilaterals=';
disp([ss3 num2str(nnode) ',' num2str(nel)])
```

(4)***************

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

(5)*****************

```
function[NNC,phic8,xic]=D2LaplaceEquationQ8MoinExautomeshgen(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)
%D2LaplaceEquationQ8MoinExautomeshgen([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)
%D2LaplaceEquationQ8MoinExautomeshgen([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)
%D2LaplaceEquationQ8MoinExautomeshgen([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)
syms coord
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh
)
%nnel=4;
%ndof=1;
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order(n1,n2,n3,nmax,numtri,
ndiv,mesh)
```

```
ntriel=0;
for triel=1:nel/3
   if ( (nodes(3*triel-2,1)==nodes(3*triel-1,1))&(nodes(3*triel-1,1)==nodes(3*triel,1)) )
   ntriel=ntriel+1;
   centnode(ntriel,1)=nodes(3*triel,1);
   end
end
centnode
nnel=8;
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));

%nnode=17,nel=12,nnel=4,ndof=1
%>>LaplaceEquationQuad4twodimension(12,17,4,1)
%
%Ex1:nnode=41,nel=36,,nnel=4,nodf=1
%>>LaplaceEquationQuad4twodimensionEx1(36,41,4,1)
%>>improvedLaplaceEquationQuad4twodimensionEx1_explicit(36,41,4,1)
%Ex2:nnode=83,nel=69,,nnel=4,nodf=1
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicit(69,83,4,1)#
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicitfnmesh(69,83,4,1)#
%improvedLaplaceEquationQuad4twodimensionEx2_explicitvfnmesh(72,87,4,1)#new
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=3,nnode=7,nnel=4,ndof=1)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel,nnode,nnel=4,ndof=1,quadtype=0/3,
mesh=1,2,3...)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=12,nnode=19,nnel=4,ndof=1,quadty
pe=0/3,mesh=3)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=27,nnode=37,nnel=4,ndof=1,quadty
pe=0/3,mesh=4)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=48,nnode=61,nnel=4,ndof=1,quadty
pe=0/3,mesh=5)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=75,nnode=91,nnel=4,ndof=1,quadty
pe=0/3,mesh=6)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(108,127,4,1,3,7)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(147,169,4,1,3,8)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(192,217,4,1,3,9)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(243,271,4,1,3,10)
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
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


end
    bcdof
    mm=length(bcdof);

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



 for L=1:nel
```

DOI: 10.18535/ijecs/v6i10.13

```
  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(' 1.19732437518704939126225670841')),vpa(sym('
1.07234243081152493747384516139'));...
            vpa(sym(' 1.07234243081152493747384516139')),vpa(sym('
1.19732437518704939126225670841'))];

intJdn1dn2uvrs =[vpa(sym('    .39328207524777371271872744686')),vpa(sym('
.66505503018777989604036502927e-1'));...
            vpa(sym('   .233172169685444656270703169594')),vpa(sym('
.30901830189818397724346918143'))];

intJdn1dn3uvrs =[vpa(sym(' .39105823773230516010581465792')),vpa(sym('
.34520910581966151866889237468'));...
            vpa(sym(' .34520910581966151866889237468')),vpa(sym('
.39105823773230516010581465792'))];

intJdn1dn4uvrs =[vpa(sym('    .30901830189818397724346918143')),vpa(sym('
.233172169685444656270703169594'));...
            vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.39328207524777371271872744686'))];

intJdn1dn5uvrs =[vpa(sym(' -1.20601914456199564761927515849')),vpa(sym('  -
.19480639850558017529001262923'));...
            vpa(sym('  -.86147306517224684195667929589')),vpa(sym('  -
.27406792898205548885083220480'))];

intJdn1dn6uvrs =[vpa(sym('  -.19025279048691687518778377091')),vpa(sym('  -
.33047487282879104238539264173'));...
            vpa(sym('  -.33047487282879104238539264173')),vpa(sym(' -
.620343126034344229672376860410'))];

intJdn1dn7uvrs =[vpa(sym(' -.620343126034344229672376860410')),vpa(sym('  -
.33047487282879104238539264173'));...
            vpa(sym('  -.33047487282879104238539264173')),vpa(sym('  -
.19025279048691687518778377091'))];

intJdn1dn8uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.86147306517224684195667929589'));...
            vpa(sym('  -.19480639850558017529001262923')),vpa(sym(' -
1.20601914456199564761927515849'))];
```

```
intJdn2dn1uvrs =[vpa(sym('    .39328207524777371271872744686')),vpa(sym('
.233172169685444656270703169594'));...
            vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.30901830189818397724346918143'))];

intJdn2dn2uvrs =[vpa(sym('   .42652636618519994053203031884')),vpa(sym(' -
.263513565675283566822536323478'));...
            vpa(sym(' -.263513565675283566822536323478')),vpa(sym('
.45005030410782929514068096460'))];

intJdn2dn3uvrs =[vpa(sym('  .1595057645371386486635746307258')),vpa(sym('
.15687259169421065039078881847e-1'));...
            vpa(sym('   .182353925836087731705745548513')),vpa(sym('
.314341351654616447964784833418'))];

intJdn2dn4uvrs =[vpa(sym('     .20218069152088195770651481813')),vpa(sym(' -
.97977735817538970518647437153e-1'));...
            vpa(sym(' -.97977735817538970518647437153e-1')),vpa(sym('
.20218069152088195770651481813'))];

intJdn2dn5uvrs =[vpa(sym('  -.92334437129978410949228855299')),vpa(sym('  -
.21347210674508644538200765866'));...
            vpa(sym('   .45319455992158022128465900801')),vpa(sym(' -
.574726461686104068369695422e-2'))];

intJdn2dn6uvrs =[vpa(sym('   .219362359001080107967460556119')),vpa(sym('
.3242127618106983385904862884652'));...
            vpa(sym(' -.3424539048559683280761803782014')),vpa(sym('  -
.768806561449731024667082979907'))];

intJdn2dn7uvrs =[vpa(sym('   -.390687025888983122842003915751')),vpa(sym(' -
.36332930895951737339235932476e-1'));...
            vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym('  -
.3376064404514624483888879556e-1'))];

intJdn2dn8uvrs =[vpa(sym(' -.8682585930330713525401530193e-1')),vpa(sym('
.38224148468296660162159011855e-1'));...
            vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym('  -
.467276179069773367865781067869'))];

intJdn3dn1uvrs =[vpa(sym(' .39105823773230516010581465792')),vpa(sym('
.34520910581966151866889237468'));...
            vpa(sym(' .34520910581966151866889237468')),vpa(sym('
.39105823773230516010581465792'))];

intJdn3dn2uvrs =[vpa(sym('  .1595057645371386486635746307258')),vpa(sym('
.182353925836087731705745548513'));...
            vpa(sym(' .15687259169421065039078881847e-1')),vpa(sym('
.314341351654616447964784833418'))];

intJdn3dn3uvrs =[vpa(sym(' .660112740472589759408893364145')),vpa(sym('
.619284819567235241935951286773'));...
```

vpa(sym(' .619284819567235241935951286773')),vpa(sym(' .660112740472589759408893364145'))];

intJdn3dn4uvrs =[vpa(sym(' .314341351654616447964784833418')),vpa(sym(' .156872591694210650390788818470e-1'));...
vpa(sym(' .182353925836087731705745548513')),vpa(sym(' .159505764537138648663574630725 8'))];

intJdn3dn5uvrs =[vpa(sym(' -.447323988028158265936415185565')),vpa(sym(' -.182789328176641032221385163017'));...
vpa(sym(' -.182789328176641032221385163017')),vpa(sym(' -.151799590926597187110452711 83'))];

intJdn3dn6uvrs =[vpa(sym(' .631719461397772383275831182 3e-2')),vpa(sym(' -.731811560352895079786782216220'));...
vpa(sym(' -.651448936862284131201155495 53e-1')),vpa(sym(' -.932211710055872286928957900634'))];

intJdn3dn7uvrs =[vpa(sym(' -.932211710055872286928957900634')),vpa(sym(' -.651448936862284131201155495 53e-1'));...
vpa(sym(' -.731811560352895079786782216220')),vpa(sym(' .631719461397772383275831182 3e-2'))];

intJdn3dn8uvrs =[vpa(sym(' -.151799590926597187110452711 83')),vpa(sym(' -.182789328176641032221385163017'));...
vpa(sym(' -.182789328176641032221385163017')),vpa(sym(' -.447323988028158265936415185565'))];

intJdn4dn1uvrs =[vpa(sym(' .309018301898183977243469181 43')),vpa(sym(' .665055030187779896040365029 27e-1'));...
vpa(sym(' .233172169685444656270703169594')), vpa(sym(' .393282075247773712718727446 86'))];

intJdn4dn2uvrs =[vpa(sym(' .202180691520881957706514818 13')),vpa(sym(' -.979777358175389705186474371 53e-1'));...
vpa(sym(' -.979777358175389705186474371 53e-1')),vpa(sym(' .202180691520881957706514818 13'))];

intJdn4dn3uvrs =[vpa(sym(' .314341351654616447964784833418')),vpa(sym(' .182353925836087731705745548513'));...
vpa(sym(' .156872591694210650390788818470e-1')),vpa(sym(' .159505764537138648663574630725 8'))];

intJdn4dn4uvrs =[vpa(sym(' .450050304107829295140680964 60')),vpa(sym(' -.263513565675283566822536323478'));...
vpa(sym(' -.263513565675283566822536323478')),vpa(sym(' .426526366185199940532030318 84'))];

intJdn4dn5uvrs =[vpa(sym(' -.467276179069773367865781067869')),vpa(sym(' .382241484682966601621590118 55e-1'));...
vpa(sym(' .382241484682966601621590118 55e-1')),vpa(sym(' -.868258593033071352540153019 3e-1'))];

intJdn4dn6uvrs =[vpa(sym(' -.33760644045146244838888795560e-1')),vpa(sym(' -.36332930895951737339235932476e-1'));...
vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym(' -.39068702588898312284200391575l'))];

intJdn4dn7uvrs =[vpa(sym(' -.76880656144973102466708297990 7')),vpa(sym(' -.34245390485596832807618037820 14'));...
vpa(sym(' .32421276181069833859048628846 52')),vpa(sym(' .21936235900108010796746055611 9'))];

intJdn4dn8uvrs =[vpa(sym(' -.57472646168610406836969542 2e-2')),vpa(sym(' .45319455992158022128465900801'));...
vpa(sym(' -.21347210674508644538200765866')),vpa(sym(' -.92334437129978410949228855299'))];

intJdn5dn1uvrs =[vpa(sym(' -1.2060191445619956476192751584 9')),vpa(sym(' -.86147306517224684195667929589'));...
vpa(sym(' -.19480639850558017529001262923')),vpa(sym(' -.27406792898205548885083220480'))];

intJdn5dn2uvrs =[vpa(sym(' -.92334437129978410949228855299')),vpa(sym(' .45319455992158022128465900801'));...
vpa(sym(' -.21347210674508644538200765866')),vpa(sym(' -.57472646168610406836969542 2e-2'))];

intJdn5dn3uvrs =[vpa(sym(' -.44732398802815826593641518556 5')),vpa(sym(' -.18278932817664103222138516301 7'));...
vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -.15179959092659718711045271183'))];

intJdn5dn4uvrs =[vpa(sym(' -.46727617906977336786578106786 9')),vpa(sym(' .38224148468296660162159011855e-1'));...
vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym(' -.86825859303307135254015301 93e-1'))];

intJdn5dn5uvrs =[vpa(sym(' 2.1717715426240022624872702233 6')),vpa(sym(' .52992039898580294331102892478'));...
vpa(sym(' .52992039898580294331102892478')),vpa(sym(' .58827358447153923041425369 63'))];

intJdn5dn6uvrs =[vpa(sym(' -.41594373113635804762610497799 7')),vpa(sym(' -.44624674573735878584568347920 0'));...
vpa(sym(' -.44624674573735878584568347920 0')),vpa(sym(' .34312587200869545409370986785e-1'))];

intJdn5dn7uvrs =[vpa(sym(' .91157772830419213711619683500')),vpa(sym(' .94571793322616328048899852 21e-1'));...
vpa(sym(' .94571793322616328048899852 21e-1')),vpa(sym(' -.48070367101146296286102539491'))];

```
intJdn5dn8uvrs =[vpa(sym(' .376558143167875038936397884560')),vpa(sym('
.374598238387950507217001141 25'));...
             vpa(sym('  .374598238387950507217001141 25')),vpa(sym('
.376558143167875038936397884560'))];

intJdn6dn1uvrs =[vpa(sym('  -.190252790486916875187783770 91')),vpa(sym('  -
.330474872828791042385392641 73'));...
             vpa(sym('  -.330474872828791042385392641 73')),vpa(sym(' -
.620343126034344229672376860410'))];

intJdn6dn2uvrs =[vpa(sym('   .219362359001080107967460556119')),vpa(sym(' -
.342453904855968328076180378201 4'));...
             vpa(sym('  .324212761810698338590486288465 2')),vpa(sym('  -
.768806561449731024667082979907'))];

intJdn6dn3uvrs =[vpa(sym('   .631719461397772383275831182 3e-2')),vpa(sym(' -
.651448936862284131201155495 53e-1'));...
             vpa(sym('   -.731811560352895079786782216220')),vpa(sym('   -
.932211710055872286928957900634'))];

intJdn6dn4uvrs =[vpa(sym('  -.337606440451462448388887955 6e-1')),vpa(sym(' -
.363329308959517373392359324 76e-1'));...
             vpa(sym(' -.363329308959517373392359324 76e-1')),vpa(sym('   -
.390687025888983122842003915751'))];

intJdn6dn5uvrs =[vpa(sym('  -.415943731136358047626104977997')),vpa(sym('  -
.446246745737358785845683479200'));...
             vpa(sym('  -.446246745737358785845683479200')),vpa(sym('
.343125872008695454093709867 85e-1'))];

intJdn6dn6uvrs =[vpa(sym('   .828654335215301005599649142 24')),vpa(sym('
.472904354164178470376423509241'));...
             vpa(sym('  .472904354164178470376423509241')),vpa(sym('
1.699831160074343688470918905720'))];

intJdn6dn7uvrs =[vpa(sym(' .663269478495252931139349292 00e-1')),vpa(sym('
.653177200517503508341284619711'));...
             vpa(sym('   .653177200517503508341284619711')),vpa(sym('
.663269478495252931139349292 00e-1'))];

intJdn6dn8uvrs =[vpa(sym('  -.480703671011462962861025394 91')),vpa(sym('
.945717933226163280488998522 1e-1'));...
             vpa(sym(' .945717933226163280488998522 1e-1')),vpa(sym('
.911577728304192137116196835 00'))];

intJdn7dn1uvrs =[vpa(sym(' -.620343126034344229672376860410')),vpa(sym('  -
.330474872828791042385392641 73'));...
             vpa(sym('  -.330474872828791042385392641 73')),vpa(sym('  -
.190252790486916875187783770 91'))];

intJdn7dn2uvrs =[vpa(sym('   -.390687025888983122842003915751')),vpa(sym(' -
.363329308959517373392359324 76e-1'));...
```

DOI: 10.18535/ijecs/v6i10.13

```
        vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym('  -
.33760644045146244838888879556e-1'))];

intJdn7dn3uvrs =[vpa(sym('   -.93221171005587228692895790063 4')),vpa(sym('   -
.73181156035289507978678221622 0'));...
        vpa(sym(' -.65144893686228413120115549553e-1')),vpa(sym('
.63171946139777238327583118 23e-2'))];

intJdn7dn4uvrs =[vpa(sym('  -.76880656144973102466708297990 7')),vpa(sym('
.32421276181069833859048628846 52'));...
        vpa(sym(' -.34245390485596832807618037820 14')),vpa(sym('
.21936235900108010796746055611 9'))];

intJdn7dn5uvrs =[vpa(sym('   .91157772830419213711619683500')),vpa(sym('
.94571793322616328048899852 21e-1'));...
        vpa(sym(' .94571793322616328048899852 21e-1')),vpa(sym('  -
.48070367101146296286102539491'))];

intJdn7dn6uvrs =[vpa(sym(' .66326947849525293113934929200e-1')),vpa(sym('
.65317720051750350834128461971 1'));...
        vpa(sym('   .65317720051750350834128461971 1')),vpa(sym('
.66326947849525293113934929200e-1'))];

intJdn7dn7uvrs =[vpa(sym(' 1.69983116007434368847091890572 0')),vpa(sym('
.47290435416417847037642350924 1'));...
        vpa(sym('  .47290435416417847037642350924 1')),vpa(sym('
.82865433521530100559964914224'))];

intJdn7dn8uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.44624674573735878584568347920 0'));...
        vpa(sym('  -.44624674573735878584568347920 0')),vpa(sym('  -
.41594373113635804762610497799 7'))];

intJdn8dn1uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.19480639850558017529001262923'));...
        vpa(sym('  -.86147306517224684195667929589')),vpa(sym(' -
1.20601914456199564761927515849'))];

intJdn8dn2uvrs =[vpa(sym(' -.86825859303307135254015301 93e-1')),vpa(sym('
.38224148468296660162159011855e-1'));...
        vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym('  -
.46727617906977336786578106786 9'))];

intJdn8dn3uvrs =[vpa(sym('  -.15179959092659718711045271183')),vpa(sym(' -
.18278932817664103222138516301 7'));...
        vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -
.44732398802815826593641518556 5'))];

intJdn8dn4uvrs =[vpa(sym(' -.57472646168610406836969542 2e-2')),vpa(sym('  -
.21347210674508644538200765866'));...
        vpa(sym('   .45319455992158022128465900801')),vpa(sym('  -
.92334437129978410949228855299'))];
```

DOI: 10.18535/ijecs/v6i10.13

```
intJdn8dn5uvrs =[vpa(sym(' .37655814316787503893639788456O')),vpa(sym('
.37459823838795050721700114125'));...
              vpa(sym('  .37459823838795050721700114125')),vpa(sym('
.37655814316787503893639788456O'))];

intJdn8dn6uvrs =[vpa(sym('  -.48070367101146296286102539491')),vpa(sym('
.9457179332261632804889985221e-1'));...
              vpa(sym(' .9457179332261632804889985221e-1')),vpa(sym('
.91157772830419213711619683500'))];

intJdn8dn7uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.446246745737358785845683479200'));...
              vpa(sym('  -.446246745737358785845683479200')),vpa(sym('  -
.415943731136358047626104977997'))];

intJdn8dn8uvrs =[vpa(sym('   .5882735844715392304142536963')),vpa(sym('
.52992039898580294331102892478'));...
              vpa(sym('  .52992039898580294331102892478')),vpa(sym('
2.17177154262400226248727022336'))];

%======================================================================
intJdndn=double([intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs intJdn1dn5uvrs
intJdn1dn6uvrs intJdn1dn7uvrs intJdn1dn8uvrs;...
        intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs intJdn2dn5uvrs intJdn2dn6uvrs
intJdn2dn7uvrs intJdn2dn8uvrs;...
        intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs intJdn3dn5uvrs intJdn3dn6uvrs
intJdn3dn7uvrs intJdn3dn8uvrs;...
        intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs intJdn4dn5uvrs intJdn4dn6uvrs
intJdn4dn7uvrs intJdn4dn8uvrs;...
        intJdn5dn1uvrs intJdn5dn2uvrs intJdn5dn3uvrs intJdn5dn4uvrs intJdn5dn5uvrs intJdn5dn6uvrs
intJdn5dn7uvrs intJdn5dn8uvrs;...
        intJdn6dn1uvrs intJdn6dn2uvrs intJdn6dn3uvrs intJdn6dn4uvrs intJdn6dn5uvrs intJdn6dn6uvrs
intJdn6dn7uvrs intJdn6dn8uvrs;...
        intJdn7dn1uvrs intJdn7dn2uvrs intJdn7dn3uvrs intJdn7dn4uvrs intJdn7dn5uvrs intJdn7dn6uvrs
intJdn7dn7uvrs intJdn7dn8uvrs;...
        intJdn8dn1uvrs intJdn8dn2uvrs intJdn8dn3uvrs intJdn8dn4uvrs intJdn8dn5uvrs intJdn8dn6uvrs
intJdn8dn7uvrs intJdn8dn8uvrs]);




%
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(1:8,1:8)=(zeros(8,8));
for i=1:8
 for j=i:8
 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(1:8,1)=zeros(8,1)
 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)*(-1-si-sj))/4;
         n2ij=((1+si)*(1-sj)*(-1+si-sj))/4;
         n3ij=((1+si)*(1+sj)*(-1+si+sj))/4;
         n4ij=((1-si)*(1+sj)*(-1-si+sj))/4;
         n5ij=((1-sj)*(1-si^2))/2;
         n6ij=((1+si)*(1-sj^2))/2;
         n7ij=((1+sj)*(1-si^2))/2;
         n8ij=((1-si)*(1-sj^2))/2;
```

```
        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;
       f5i=n5ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
       f6i=n6ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
       f7i=n7ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
       f8i=n8ij*(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;
         f(5,1)=f(5,1)+f5i*wi*wj;
         f(6,1)=f(6,1)+f6i*wi*wj;
         f(7,1)=f(7,1)+f7i*wi*wj;
         f(8,1)=f(8,1)+f8i*wi*wj;
      end
  end
 f=(delabc)*f;

%________________________________________
 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          analytical(theoretical)-values         ')

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

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
[1 phi(1,1) xi(1,1)]
centnode
for I=1:nel/3
II=centnode(I,1);
NNC(I,1)=II;
phic8(I,1)=phi(II,1);
xic(I,1)=xi(II,1);
end

disp('______________________________________________________________________________')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
disp('__________________________________________________________________________________________
____')
disp('                    fem-computed values          anlytical(theoretical)-values         ')

disp([NNC phic8 xic])
disp('__________________________________________________________________________________________
____')
```

```
disp('---------------------------------------------------------------------------------------------------------------------------------------------------------------')
disp('                    node number  fem-computed values     anlytical(theoretical)-values   node number  fem-computed values     anlytical(theoretical)-values')
disp('---------------------------------------------------------------------------------------------------------------------------------------------------------------')
for I=1:2:nel/3
   if((I+1)<=(nel/3))
   disp([NNC(I) phic8(I) xic(I) NNC(I+1) phic8(I+1) xic(I+1)])
   end
  if(I==nel/3)
   disp([NNC(I) phic8(I) xic(I) ])
   end
end
disp('---------------------------------------------------------------------------------------------------------------------------------------------------------------')

(6)******************
function [s,wwww]=glsampleptsweights(n)
switch n
   case 1
      s(1)=0;wwww(1)=2;
   case 2
      table=[  .5773502691896257645091487805019 5,  1.00000000000000000000000000000000
            -.57735026918962576450914878050195,  1.00000000000000000000000000000000];
s=table(:,1);wwww=table(:,2);
case 3
         table=[  .77459666924148337703585307995650,  .55555555555555555555555555555571
               -.77459666924148337703585307995650,  .55555555555555555555555555555571
                          0,                                        8/9];
s=table(:,1);wwww=table(:,2);
case 4
   table=[   .86113631159405257522394648889280,  .34785484513745385737306394922182
           -.86113631159405257522394648889280,  .34785484513745385737306394922182
            .33998104358485626480266575910326,  .65214515486254614262693605077797
           -.33998104358485626480266575910326,  .65214515486254614262693605077797];
       s=table(:,1);wwww=table(:,2);
    case 5
    table=[                           0,  .56888888888888888888888888888889
            .90617984593866399279762687829940,  .23692688505618908751426404072021
           -.90617984593866399279762687829940,  .23692688505618908751426404072021
            .53846931010568309103631442070020,  .47862867049936646804129151483563
           -.53846931010568309103631442070020,  .47862867049936646804129151483563];
       s=table(:,1);wwww=table(:,2);
    case 6
    table=[  .93246951420315202781230155449400,  .17132449237917034504029614217314
           -.93246951420315202781230155449400,  .17132449237917034504029614217314
            .23861918608319690863050172168070,  .46791393457269104738987034398952
           -.23861918608319690863050172168070,  .46791393457269104738987034398952
            .66120938646626451366139959501985,  .36076157304813860756983351383736
           -.66120938646626451366139959501985,  .36076157304813860756983351383736];
           s=table(:,1);wwww=table(:,2);
```

DOI: 10.18535/ijecs/v6i10.13

```
case 7
 table=[                              0,  .41795918367346938775510204081633
        .94910791234275852452618968404785,  .12948496616886969327061143267867
       -.94910791234275852452618968404785,  .12948496616886969327061143267867
        .40584515137739716690660641207694,  .38183005050511894495036977548908
       -.40584515137739716690660641207694,  .38183005050511894495036977548908
        .74153118559939443986386477328075,  .27970539148927666790146777142437
       -.74153118559939443986386477328075,  .27970539148927666790146777142437];
         s=table(:,1);wwww=table(:,2);
case 8
        table=[  .96028985649753623168356086856945,  .10122853629037625915253135431052
               -.96028985649753623168356086856945,  .10122853629037625915253135431052
                .79666647741362673959155393647585,  .22238103445337447054435599442789
               -.79666647741362673959155393647585,  .22238103445337447054435599442789
                .52553240991632898581773904918925,  .31370664587788728733796220198672
               -.52553240991632898581773904918925,  .31370664587788728733796220198672
                .18343464249564980493947614236015,  .36268378337836198296515044927715
               -.18343464249564980493947614236015,  .36268378337836198296515044927715];

                s=table(:,1);wwww=table(:,2);
case 9
    table=[                              0,    .33023935500125976316452506928697
            .96816023950762608983557620290365, .81274388361574411971892158114991e-1
           -.96816023950762608983557620290365, .81274388361574411971892158114991e-1
            .83603110732663579429942978806975,    .18064816069485740405847203124622
           -.83603110732663579429942978806975,    .18064816069485740405847203124622
            .61337143270059039730870203934145,    .26061069640293546231874286941878
           -.61337143270059039730870203934145,    .26061069640293546231874286941878
            .32425342340380892903853801464332,    .31234707704000284006863040658462
           -.32425342340380892903853801464332,    .31234707704000284006863040658462];

                  s=table(:,1);wwww=table(:,2);
  case 10
     table=[   -.14887433898163121088482600112972,    .29552422471475287017389299465132
            .14887433898163121088482600112972,    .29552422471475287017389299465132
           -.43339539412924719079926594316579,    .26926671930999635509122692156937
            .43339539412924719079926594316579,    .26926671930999635509122692156937
           -.67940956829902440623432736511485,    .21908636251598204399553493422951
            .67940956829902440623432736511485,    .21908636251598204399553493422951
           -.86506336668898451073209668842350,    .14945134915058059314577633965488
            .86506336668898451073209668842350,    .14945134915058059314577633965488
           -.97390652851717172007796401208445, .66671344308688137593568809896211e-1
            .97390652851717172007796401208445, .66671344308688137593568809896211e-1];

        s=table(:,1);wwww=table(:,2);
end
(7)**************************
function[]=D2PoissonEquationQ8MoinEx_MeshgridContour(n1,n2,n3,nmax,numtri,ndiv,mesh,fcn,ng)

%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)
%D2PoissonEquationQ8MoinEx_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)
%D2PoissonEquationQ8MoinEx_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)
%D2PoissonEquationQ8MoinEx_MeshgridContour([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,
10)
%D2PoissonEquationQ8MoinEx_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,
2,2,1,10)
%D2PoissonEquationQ8MoinEx_MeshgridContour(1,2,3,3,1,2,50,2,2)
syms coord
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh
)
%nnel=4;
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order(n1,n2,n3,nmax,numtri,
ndiv,mesh)
nnel=8;
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));

%nnode=17,nel=12,nnel=4,ndof=1
%>>LaplaceEquationQuad4twodimension(12,17,4,1)
%
%Ex1:nnode=41,nel=36,,nnel=4,nodf=1
%>>LaplaceEquationQuad4twodimensionEx1(36,41,4,1)
%>>improvedLaplaceEquationQuad4twodimensionEx1_explicit(36,41,4,1)
%Ex2:nnode=83,nel=69,,nnel=4,nodf=1
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicit(69,83,4,1)#
```

```
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicitfnmesh(69,83,4,1)#
%improvedLaplaceEquationQuad4twodimensionEx2_explicitvfnmesh(72,87,4,1)#new
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=3,nnode=7,nnel=4,ndof=1)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel,nnode,nnel=4,ndof=1,quadtype=0/3,
mesh=1,2,3...)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=12,nnode=19,nnel=4,ndof=1,quadty
pe=0/3,mesh=3)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=27,nnode=37,nnel=4,ndof=1,quadty
pe=0/3,mesh=4)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=48,nnode=61,nnel=4,ndof=1,quadty
pe=0/3,mesh=5)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=75,nnode=91,nnel=4,ndof=1,quadty
pe=0/3,mesh=6)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(108,127,4,1,3,7)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(147,169,4,1,3,8)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(192,217,4,1,3,9)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(243,271,4,1,3,10)
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
    bcdof
    mm=length(bcdof);

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
    bcdof
    mm=length(bcdof);

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 50%torsion of an equilateral triangle: example problem from rathod and shafiqul
     syms COORD
 nside=3;
  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);
     COORD(side,2)=coord(side,2);
 end
  COORD(nside+1,1)=coord(1,1);
  COORD(nside+1,2)=coord(1,2);
  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
  bcval
  mm=length(bcdof);
  for m=1:nnode
    x=(gcoord(m,1));y=(gcoord(m,2));
    xi(m,1)=(y*((sqrt(3))*x+y- sqrt(3)/2)*(-(sqrt(3))*x+y- sqrt(3)/2))/sqrt(3)/2;
  end

format long g
k1 =sqrt(3)/80;







   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
%___________________________________________________________________________
ng=10
 [sp,wt]=glsampleptsweights(ng)
table2=[N xx yy];
%disp([xx yy])
%_________________________________________________________________________________
```

DOI: 10.18535/ijecs/v6i10.13

```
intJdn1dn1uvrs =[vpa(sym(' 1.19732437518704939126225670841')),vpa(sym('
1.07234243081152493747384516139'));...
              vpa(sym(' 1.07234243081152493747384516139')),vpa(sym('
1.19732437518704939126225670841'))];

intJdn1dn2uvrs =[vpa(sym('    .39328207524777371271872744686')),vpa(sym('
.66505503018777989604036502927e-1'));...
              vpa(sym('   .233172169685444656270703169594')),vpa(sym('
.30901830189818397724346918143'))];

intJdn1dn3uvrs =[vpa(sym(' .39105823773230516010581465792')),vpa(sym('
.34520910581966151866889237468'));...
              vpa(sym(' .34520910581966151866889237468')),vpa(sym('
.39105823773230516010581465792'))];

intJdn1dn4uvrs =[vpa(sym('    .30901830189818397724346918143')),vpa(sym('
.233172169685444656270703169594'));...
              vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.39328207524777371271872744686'))];

intJdn1dn5uvrs =[vpa(sym(' -1.20601914456199564761927515849')),vpa(sym('  -
.19480639850558017529001262923'));...
              vpa(sym('  -.86147306517224684195667929589')),vpa(sym('  -
.27406792898205548885083220480'))];

intJdn1dn6uvrs =[vpa(sym('  -.19025279048691687518778377091')),vpa(sym('  -
.33047487282879104238539264173'));...
              vpa(sym('  -.33047487282879104238539264173')),vpa(sym(' -
.620343126034344229672376860410'))];

intJdn1dn7uvrs =[vpa(sym(' -.620343126034344229672376860410')),vpa(sym('  -
.33047487282879104238539264173'));...
              vpa(sym('  -.33047487282879104238539264173')),vpa(sym('  -
.19025279048691687518778377091'))];

intJdn1dn8uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.86147306517224684195667929589'));...
              vpa(sym('  -.19480639850558017529001262923')),vpa(sym(' -
1.20601914456199564761927515849'))];

intJdn2dn1uvrs =[vpa(sym('    .39328207524777371271872744686')),vpa(sym('
.233172169685444656270703169594'));...
              vpa(sym(' .66505503018777989604036502927e-1')),vpa(sym('
.30901830189818397724346918143'))];

intJdn2dn2uvrs =[vpa(sym('   .42652636618519994053203031884')),vpa(sym(' -
.263513565675283566822536323478'));...
              vpa(sym(' -.263513565675283566822536323478')),vpa(sym('
.45005030410782929514068096460'))];

intJdn2dn3uvrs =[vpa(sym('  .1595057645371386486635746307258')),vpa(sym('
.15687259169421065039078881847e-1'));...
```

```
            vpa(sym('  .18235392583608773170574554851 3')),vpa(sym('
.31434135165461644796478483341 8'))];

intJdn2dn4uvrs =[vpa(sym('    .20218069152088195770651481813')),vpa(sym(' -
.97977735817538970518647437153e-1'));...
            vpa(sym(' -.97977735817538970518647437153e-1')),vpa(sym('
.20218069152088195770651481813'))];

intJdn2dn5uvrs =[vpa(sym('  -.92334437129978410949228855299')),vpa(sym('  -
.21347210674508644538200765866'));...
            vpa(sym('  .45319455992158022128465900801')),vpa(sym(' -
.57472646168610406836969542 2e-2'))];

intJdn2dn6uvrs =[vpa(sym('  .21936235900108010796746055611 9')),vpa(sym('
.32421276181069833859048628846 52'));...
            vpa(sym(' -.34245390485596832807618037820 14')),vpa(sym('  -
.76880656144973102466708297990 7'))];

intJdn2dn7uvrs =[vpa(sym('  -.39068702588898312284200391575 1')),vpa(sym(' -
.36332930895951737339235932476e-1'));...
            vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym('  -
.33760644045146244838888795 56e-1'))];

intJdn2dn8uvrs =[vpa(sym(' -.86825859303307135254015301 93e-1')),vpa(sym('
.38224148468296660162159011855e-1'));...
            vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym('  -
.46727617906977336786578106786 9'))];

intJdn3dn1uvrs =[vpa(sym(' .39105823773230516010581465792')),vpa(sym('
.34520910581966151866889237468'));...
            vpa(sym(' .34520910581966151866889237468')),vpa(sym('
.39105823773230516010581465792'))];

intJdn3dn2uvrs =[vpa(sym('  .15950576453713864866357463072 58')),vpa(sym('
.18235392583608773170574554851 3'));...
            vpa(sym(' .15687259169421065039078881847e-1')),vpa(sym('
.31434135165461644796478483341 8'))];

intJdn3dn3uvrs =[vpa(sym(' .66011274047258975940889336414 5')),vpa(sym('
.61928481956723524193595128677 3'));...
            vpa(sym(' .61928481956723524193595128677 3')),vpa(sym('
.66011274047258975940889336414 5'))];

intJdn3dn4uvrs =[vpa(sym('  .31434135165461644796478483341 8')),vpa(sym('
.15687259169421065039078881847e-1'));...
            vpa(sym('  .18235392583608773170574554851 3')),vpa(sym('
.15950576453713864866357463072 58'))];

intJdn3dn5uvrs =[vpa(sym(' -.44732398802815826593641518556 5')),vpa(sym(' -
.18278932817664103222138516301 7'));...
            vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym('  -
.15179959092659718711045271183'))];
```

```
intJdn3dn6uvrs =[vpa(sym('   .63171946139777238327583118230e-2')),vpa(sym('   -
.7318115603528950797867822162200'));...
            vpa(sym(' -.65144893686228413120115549553e-1')),vpa(sym('   -
.9322117100558722869289579006340'))];

intJdn3dn7uvrs =[vpa(sym('   -.9322117100558722869289579006340')),vpa(sym(' -
.65144893686228413120115549553e-1'));...
            vpa(sym('   -.7318115603528950797867822162200')),vpa(sym('
.63171946139777238327583118230e-2'))];

intJdn3dn8uvrs =[vpa(sym('  -.1517995909265971871104527118300')),vpa(sym(' -
.1827893281766410322213851630170'));...
            vpa(sym(' -.1827893281766410322213851630170')),vpa(sym(' -
.4473239880281582659364151855650'))];

intJdn4dn1uvrs =[vpa(sym('    .3090183018981839772434691814300')),vpa(sym('
.66505503018777989604036502927e-1'));...
            vpa(sym(' .2331721696854446562707031695940')),   vpa(sym('
.3932820752477737127187274468600'))];

intJdn4dn2uvrs =[vpa(sym('     .2021806915208819577065148181300')),vpa(sym(' -
.97977735817538970518647437153e-1'));...
            vpa(sym(' -.97977735817538970518647437153e-1')),vpa(sym('
.2021806915208819577065148181300'))];

intJdn4dn3uvrs =[vpa(sym('   .3143413516546164479647848334180')),vpa(sym('
.1823539258360877317057455485130'));...
            vpa(sym(' .15687259169421065039078881847e-1')),vpa(sym('
.1595057645371386486635746307258'))];

intJdn4dn4uvrs =[vpa(sym('   .4500503041078292951406809646000')),vpa(sym(' -
.2635135656752835668225363234780'));...
            vpa(sym(' -.2635135656752835668225363234780')),vpa(sym('
.4265263661851999405320303188400'))];

intJdn4dn5uvrs =[vpa(sym('  -.4672761790697733678657810678690')),vpa(sym('
.38224148468296660162159011855e-1'));...
            vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym(' -
.86825859303307135254015301930e-1'))];

intJdn4dn6uvrs =[vpa(sym('  -.33760644045146244838888795560e-1')),vpa(sym(' -
.36332930895951737339235932476e-1'));...
            vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym('   -
.3906870258889831228420039157510'))];

intJdn4dn7uvrs =[vpa(sym('  -.7688065614497310246670829799070')),vpa(sym(' -
.3424539048559683280761803782014'));...
            vpa(sym('  .3242127618106983385904862884652')),vpa(sym('
.2193623590010801079674605561190'))];
```

intJdn4dn8uvrs =[vpa(sym(' -.57472646168610406836969542e-2')),vpa(sym(' .45319455992158022128465900801'));...
vpa(sym(' -.21347210674508644538200765866')),vpa(sym(' -.92334437129978410949228855299'))];

intJdn5dn1uvrs =[vpa(sym(' -1.2060191445619956476192751584 9')),vpa(sym(' -.86147306517224684195667929589'));...
vpa(sym(' -.19480639850558017529001262923')),vpa(sym(' -.27406792898205548885083220480'))];

intJdn5dn2uvrs =[vpa(sym(' -.92334437129978410949228855299')),vpa(sym(' .45319455992158022128465900801'));...
vpa(sym(' -.21347210674508644538200765866')),vpa(sym(' -.57472646168610406836969542e-2'))];

intJdn5dn3uvrs =[vpa(sym(' -.44732398802815826593641518556 5')),vpa(sym(' -.18278932817664103222138516301 7'));...
vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -.15179959092659718711045271183'))];

intJdn5dn4uvrs =[vpa(sym(' -.46727617906977336786578106786 9')),vpa(sym(' .38224148468296660162159011855e-1'));...
vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym(' -.86825859303307135254015301 93e-1'))];

intJdn5dn5uvrs =[vpa(sym(' 2.1717715426240022624872702233 6')),vpa(sym(' .52992039898580294331102892478'));...
vpa(sym(' .52992039898580294331102892478')),vpa(sym(' .58827358447153923041425369 63'))];

intJdn5dn6uvrs =[vpa(sym(' -.41594373113635804762610497799 7')),vpa(sym(' -.44624674573735878584568347920 0'));...
vpa(sym(' -.44624674573735878584568347920 0')),vpa(sym(' .34312587200869545409370986785e-1'))];

intJdn5dn7uvrs =[vpa(sym(' .91157772830419213711619683500')),vpa(sym(' .94571793322616328048899852 21e-1'));...
vpa(sym(' .94571793322616328048899852 21e-1')),vpa(sym(' -.48070367101146296286102539491'))];

intJdn5dn8uvrs =[vpa(sym(' .37655814316787503893639788456 0')),vpa(sym(' .37459823838795050721700114125'));...
vpa(sym(' .37459823838795050721700114125')),vpa(sym(' .37655814316787503893639788456 0'))];

intJdn6dn1uvrs =[vpa(sym(' -.19025279048691687518778377091')),vpa(sym(' -.33047487282879104238539264173'));...
vpa(sym(' -.33047487282879104238539264173')),vpa(sym(' -.62034312603434422967237686041 0'))];

intJdn6dn2uvrs =[vpa(sym(' .21936235900108010796746055611 9')),vpa(sym(' -.34245390485596832807618037820 14'));...

vpa(sym(' .32421276181069833859048628884652')),vpa(sym(' -.76880656144973102466708297907'))];

intJdn6dn3uvrs =[vpa(sym(' .63171946139777238327583118823e-2')),vpa(sym(' -.65144893686228413120115549553e-1'));...
vpa(sym(' -.73181156035289507978678216220')),vpa(sym(' -.93221171005587228692895790634'))];

intJdn6dn4uvrs =[vpa(sym(' -.33760644045146244838888795556e-1')),vpa(sym(' -.36332930895951737339235932476e-1'));...
vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym(' -.39068702588898312284200391575 1'))];

intJdn6dn5uvrs =[vpa(sym(' -.41594373113635804762610497797')),vpa(sym(' -.44624674573735878584568347920 0'));...
vpa(sym(' -.44624674573735878584568347920 0')),vpa(sym(' .34312587200869545409370986785e-1'))];

intJdn6dn6uvrs =[vpa(sym(' .82865433521530100559964914224')),vpa(sym(' .47290435416417847037642350924 1'));...
vpa(sym(' .47290435416417847037642350924 1')),vpa(sym(' 1.6998311600743436884709189057 20'))];

intJdn6dn7uvrs =[vpa(sym(' .66326947849525293113934929200e-1')),vpa(sym(' .65317720051750350834128461971 1'));...
vpa(sym(' .65317720051750350834128461971 1')),vpa(sym(' .66326947849525293113934929200e-1'))];

intJdn6dn8uvrs =[vpa(sym(' -.48070367101146296286102539491')),vpa(sym(' .94571793322616328048899852221e-1'));...
vpa(sym(' .94571793322616328048899852221e-1')),vpa(sym(' .91157772830419213711619683500'))];

intJdn7dn1uvrs =[vpa(sym(' -.62034312603434422967237686041 0')),vpa(sym(' -.33047487282879104238539264173'));...
vpa(sym(' -.33047487282879104238539264173')),vpa(sym(' -.19025279048691687518778377091'))];

intJdn7dn2uvrs =[vpa(sym(' -.39068702588898312284200391575 1')),vpa(sym(' -.36332930895951737339235932476e-1'));...
vpa(sym(' -.36332930895951737339235932476e-1')),vpa(sym(' -.33760644045146244838888795556e-1'))];

intJdn7dn3uvrs =[vpa(sym(' -.93221171005587228692895790634')),vpa(sym(' -.73181156035289507978678216220'));...
vpa(sym(' -.65144893686228413120115549553e-1')),vpa(sym(' .63171946139777238327583118823e-2'))];

intJdn7dn4uvrs =[vpa(sym(' -.76880656144973102466708297907')),vpa(sym(' .32421276181069833859048628884652'));...
vpa(sym(' -.34245390485596832807618037820 14')),vpa(sym(' .21936235900108010796746055611 9'))];

```
intJdn7dn5uvrs =[vpa(sym('   .91157772830419213711619683500')),vpa(sym('
.94571793322616328048899852221e-1'));...
              vpa(sym(' .94571793322616328048899852221e-1')),vpa(sym('  -
.48070367101146296286102539491'))];

intJdn7dn6uvrs =[vpa(sym(' .66326947849525293113934929200e-1')),vpa(sym('
.65317720051750350834128461971l'));...
              vpa(sym('   .65317720051750350834128461971l')),vpa(sym('
.66326947849525293113934929200e-1'))];

intJdn7dn7uvrs =[vpa(sym(' 1.6998311600743436884709189057 20')),vpa(sym('
.47290435416417847037642350924l'));...
              vpa(sym('  .47290435416417847037642350924l')),vpa(sym('
.82865433521530100559964914224'))];

intJdn7dn8uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.44624674573735878584568347920 0'));...
              vpa(sym('  -.44624674573735878584568347920 0')),vpa(sym('  -
.41594373113635804762610497799 7'))];

intJdn8dn1uvrs =[vpa(sym('  -.27406792898205548885083220480')),vpa(sym('  -
.19480639850558017529001262923'));...
              vpa(sym('  -.86147306517224684195667929589')),vpa(sym(' -
1.2060191445619956476192751584 9'))];

intJdn8dn2uvrs =[vpa(sym(' -.86825859303307135254015301 93e-1')),vpa(sym('
.38224148468296660162159011855e-1'));...
              vpa(sym(' .38224148468296660162159011855e-1')),vpa(sym('  -
.46727617906977336786578106786 9'))];

intJdn8dn3uvrs =[vpa(sym('  -.15179959092659718711045271183')),vpa(sym(' -
.18278932817664103222138516301 7'));...
              vpa(sym(' -.18278932817664103222138516301 7')),vpa(sym(' -
.44732398802815826593641518556 5'))];

intJdn8dn4uvrs =[vpa(sym(' -.57472646168610406836969542 2e-2')),vpa(sym('  -
.21347210674508644538200765866'));...
              vpa(sym('   .45319455992158022128465900801')),vpa(sym('  -
.92334437129978410949228855299'))];

intJdn8dn5uvrs =[vpa(sym(' .37655814316787503893639788456 0')),vpa(sym('
.37459823838795050721700114125'));...
              vpa(sym('  .37459823838795050721700114125')),vpa(sym('
.37655814316787503893639788456 0'))];

intJdn8dn6uvrs =[vpa(sym('  -.48070367101146296286102539491')),vpa(sym('
.94571793322616328048899852221e-1'));...
              vpa(sym(' .94571793322616328048899852221e-1')),vpa(sym('
.91157772830419213711619683500'))];
```

DOI: 10.18535/ijecs/v6i10.13

```
intJdn8dn7uvrs =[vpa(sym(' .34312587200869545409370986785e-1')),vpa(sym('  -
.44624674573735878584568347920')),...
              vpa(sym('  -.44624674573735878584568347920')),vpa(sym('  -
.41594373113635804762610497797'))];

intJdn8dn8uvrs =[vpa(sym('   .58827358447153923041425369 63')),vpa(sym('
.52992039898580294331102892478')),...
              vpa(sym('  .52992039898580294331102892478')),vpa(sym('
2.1717715426240022624872702233 6'))];

%============================================================================
intJdndn=double([intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs intJdn1dn5uvrs
intJdn1dn6uvrs intJdn1dn7uvrs intJdn1dn8uvrs;...
        intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs intJdn2dn5uvrs intJdn2dn6uvrs
intJdn2dn7uvrs intJdn2dn8uvrs;...
        intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs intJdn3dn5uvrs intJdn3dn6uvrs
intJdn3dn7uvrs intJdn3dn8uvrs;...
        intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs intJdn4dn5uvrs intJdn4dn6uvrs
intJdn4dn7uvrs intJdn4dn8uvrs;...
        intJdn5dn1uvrs intJdn5dn2uvrs intJdn5dn3uvrs intJdn5dn4uvrs intJdn5dn5uvrs intJdn5dn6uvrs
intJdn5dn7uvrs intJdn5dn8uvrs;...
        intJdn6dn1uvrs intJdn6dn2uvrs intJdn6dn3uvrs intJdn6dn4uvrs intJdn6dn5uvrs intJdn6dn6uvrs
intJdn6dn7uvrs intJdn6dn8uvrs;...
        intJdn7dn1uvrs intJdn7dn2uvrs intJdn7dn3uvrs intJdn7dn4uvrs intJdn7dn5uvrs intJdn7dn6uvrs
intJdn7dn7uvrs intJdn7dn8uvrs;...
        intJdn8dn1uvrs intJdn8dn2uvrs intJdn8dn3uvrs intJdn8dn4uvrs intJdn8dn5uvrs intJdn8dn6uvrs
intJdn8dn7uvrs intJdn8dn8uvrs]);

%________________________________________________________________________
%

%
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;
```

DOI: 10.18535/ijecs/v6i10.13

```
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;
sk(1:8,1:8)=(zeros(8,8));
for i=1:8
 for j=i:8
 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);
if (mesh==1)|(mesh==2)
 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(1:8,1)=zeros(8,1)
 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)*(-1-si-sj))/4;
         n2ij=((1+si)*(1-sj)*(-1+si-sj))/4;
         n3ij=((1+si)*(1+sj)*(-1+si+sj))/4;
         n4ij=((1-si)*(1+sj)*(-1-si+sj))/4;
         n5ij=((1-sj)*(1-si^2))/2;
         n6ij=((1+si)*(1-sj^2))/2;
         n7ij=((1+sj)*(1-si^2))/2;
         n8ij=((1-si)*(1-sj^2))/2;
         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;
       fcnxyij=fcnxy(fcn,xeij,yeij);
       f1i=n1ij*fcnxyij*(4+si+sj)/96;
       f2i=n2ij*fcnxyij*(4+si+sj)/96;
       f3i=n3ij*fcnxyij*(4+si+sj)/96;
       f4i=n4ij*fcnxyij*(4+si+sj)/96;
       f5i=n5ij*fcnxyij*(4+si+sj)/96;
       f6i=n6ij*fcnxyij*(4+si+sj)/96;
       f7i=n7ij*fcnxyij*(4+si+sj)/96;
       f8i=n8ij*fcnxyij*(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;
        f(5,1)=f(5,1)+f5i*wi*wj;
        f(6,1)=f(6,1)+f6i*wi*wj;
        f(7,1)=f(7,1)+f7i*wi*wj;
        f(8,1)=f(8,1)+f8i*wi*wj;
     end
  end
 f=(delabc)*f;
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
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);
      nd5=nodes(iel,5);nd6=nodes(iel,6);nd7=nodes(iel,7);nd8=nodes(iel,8);
 %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);
        shn1=((1-r)*(1-s)*(-1-r-s))/4;
        shn2=((1+r)*(1-s)*(-1+r-s))/4;
        shn3=((1+r)*(1+s)*(-1+r+s))/4;
        shn4=((1-r)*(1+s)*(-1-r+s))/4;
        shn5=((1-s)*(1-r^2))/2;
        shn6=((1+r)*(1-s^2))/2;
        shn7=((1+s)*(1-r^2))/2;
        shn8=((1-r)*(1-s^2))/2;
PHI(i,j)=shn1*phi(nd1,1)+shn2*phi(nd2,1)+shn3*phi(nd3,1)+shn4*phi(nd4,1)+shn5*phi(nd5,1)+shn6*phi(
nd6,1)+shn7*phi(nd7,1)+shn8*phi(nd8,1);
          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
switch mesh
case 1
    hold off
 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='Eight 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,-.1,[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)')
text(0.25,-.1,[sst1 sst2 sst3 sst4 sst5 sst6])
 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


case 2
   hold off
 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='Eight 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,-.1,[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
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
title('contour level curves for exact solution: sin(pi*x)*sin(pi*y)')
text(0.25,-.1,[sst1 sst2 sst3 sst4 sst5 sst6])
 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

end%switch mesh
[femsoln exactsoln]

 disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
[1 phi(1,1) xi(1,1)]
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]
```

DOI: 10.18535/ijecs/v6i10.13

```
%************************
function[fcn]=fcnxy(n,x,y)
switch n
case 1
   fcn=(2*pi^2)*sin(pi*x)*sin(pi*y);
case 2
   fcn=1;
 otherwise
   disp('something wrong')
end
```