Finite element solution of Poisson Equation over Polygonal Domains using a novel auto mesh generation technique and an explicit integration scheme for linear convex quadrilaterals of cubic order Serendipity and Lagrange families

H. T. Rathod, Md. Shafiquil Islam, Bharath Rathod, K. Sugantha Devi

Abstract:
This paper presents an explicit integration scheme to compute the stiffness matrix of twelve node and sixteen node linear convex quadrilateral finite elements of Serendipity and Lagrange families using an explicit integration scheme and discretisation of polygonal domain by such finite elements using 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 matrices and the matrix of integrals with integrands as rational functions with polynomial numerator and the linear denominator \((4+\xi+\eta)\) in the bivariate \((\xi, \eta)\) over a 2-square \((-1\leq \xi, \eta \leq 1)\) with the nodes on the boundary and in the interior of this simple domain. The finite elements up to cubic order have nodes only on the boundary for Serendipity family and the finite elements with boundary as well as some interior nodes belong to the Lagrange family. The first order element is the bilinear convex quadrilateral finite element which is an exception and it belongs to both the families. We have for the present, the cubic order finite elements which have 12 boundary nodes at the nodal coordinates \((-1,-1),(1,1),(-1,1),(1,-1),(-1/3,-1/3),(1/3,1),(1,1/3),(-1/3,1/3),(-1,1/3),(1,-1/3),(-1/3,1/3),(1/3,1/3))\) and the four interior nodal coordinates at the points \((-1/3,-1/3),(1/3,-1/3),(1/3,1/3),(-1/3,1/3))\) in the local parametric space \((\xi, \eta)\). In this paper, we have computed the integrals of local derivative products with linear denominator \((4+\xi+\eta)\) in exact forms using the symbolic mathematics capabilities of MATLAB. The proposed explicit finite element integration scheme can be then 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 12-node and 16-node linear (straight edge) convex quadrilaterals for a polygonal domain \(\Omega \subset \mathbb{R}^2\) which provides the nodal coordinates and the element connectivity. We have used the explicit integration scheme and this novel auto mesh generation technique to solve the Poisson equation \(-\nabla^2 u = f\) where \(u\) is an unknown physical variable and \(f\) is a known smooth function in \(\Omega \subset \mathbb{R}^2\) with Dirichlet boundary conditions over the convex polygonal domain.

Key words: Explicit Integration, Finite Element Method, cubic order 2-D finite elements of Serendipity and Lagrange families, 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 problems 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 to 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 ≤ $\xi$ $\eta$ ≤ 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. We can always devise finite elements with higher order interpolation or shape functions by placing more nodes over these quadrilateral regions.

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 finite 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 us 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 for which properties must be determined from an analysis of the physical problem (e.g. as a result of experiments). These studies on a particular problem allow us to construct the 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 discretisation 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, MATHEMATICA 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 value problems governed by partial differential equations[27-33].

This paper presents an explicit integration scheme to compute the stiffness matrices of linear convex quadrilateral elements belonging to cubic order twelve nodal Serendipity and sixteen nodal Lagrange family using the symbolic mathematics and discretisation of polygonal domain by such finite elements using 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 $\xi$ and $\eta$ over a 2-square ($-1$ ≤ $\xi$ $\eta$ ≤ 1) with nodes at the boundary points $(-1,1),(-1,-1),(1,1),(-1,-1),(-1/3,1),(1/3,1),(-1/3,-1),(1/3,-1),(1/3,1/3),(1/3,-1/3),(-1,1/3),(-1,-1/3)$ and the

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482 Page 23330
four interior nodal coordinates at the points \((-1/3,-1/3),(1/3,-1/3),(1/3,1/3),(-1/3,1/3)\) in the local parametric space \((\xi, \eta)\). 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 12-node and 16-node linear convex quadrilaterals for a polygonal domain \(\Omega \subset \mathbb{R}^2\) 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 \(-\nabla^2 u = f\), where \(u\) is a unknown physical variable and \(f\) is a known smooth function in \(\Omega \subset \mathbb{R}^2\) with given Dirichlet boundary conditions over convex polygonal domains. We need a small amount of numerical integration to complete the solution of the Poisson boundary value problem when \(f\) is a known smooth function other than a constant.

2. POISSON EQUATION
2.1 Statement of the Problem
The Poisson equation
\[-\nabla^2 u = f\] is the simplest and most famous elliptic partial differential equations. The source (or load) function is given on some two or three dimensional domain \(\Omega \subset \mathbb{R}^2\) or \(\mathbb{R}^3\). A solution \(u\) satisfying (1.1) will also satisfy boundary conditions on the boundary \(\partial \Omega\) of \(\Omega\); for example
\[\alpha u + \beta \frac{\partial u}{\partial n} = g\] on \(\partial \Omega\)

where \(\partial u / \partial n\) denotes directional derivative in the direction normal to the boundary \(\partial \Omega\) (conveniently pointing outwards) and \(\alpha\) and \(\beta\) 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 \(\beta\) 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 \(\alpha\) 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 \(\alpha\) and \(\beta\) are both nonzero) hold on remainder \(\partial \Omega \setminus \partial \Omega_D\). 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 \(\Omega\) using Gauss’s theorem gives
\[-\int_\Omega \frac{\partial u}{\partial n} = -\int_\Omega \nabla^2 u = \int_\Omega f\] 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:
\[\int_{\partial \Omega_D} g + \int_\Omega f = 0\] (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 continuous second derivatives in \(\Omega\) (i.e. \(u\) is in \(C^2(\Omega)\)) and is continuous up to the boundary i.e. \(u\) is in \(C^0(\bar{\Omega})\). In case of nonsmooth domains or discontinuous source functions, the function \(u\) satisfying eqns (1) and (2) may not be smooth (or regular) enough to be regarded as classical solution. For problems which arise from perfectly reasonable mathematical models an alternative description of the boundary value problem is required. Since this alternative description is less restrictive in terms of admissible data it is called weak formulation.

To derive a weak formulation of a Poisson problem, we require that for an appropriate set of test functions \(v\),
\[\int_\Omega (\nabla^2 u + f) v = 0\] (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
\[-\int_\Omega v \nabla^2 u = \int_\Omega \nabla v \cdot \nabla u - \int_\Omega \nabla \cdot (v \nabla u)\]
\[= \int_\Omega \nabla u \cdot \nabla v + \int_\partial \Omega v \frac{\partial u}{\partial n},\] so that
\[\int_\Omega \nabla u \cdot \nabla v = \int_\Omega v f + \int_\partial \Omega v \frac{\partial u}{\partial n}\] (6a)
The point here is that the problem posed by eqn (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 straightforward. Substituting from eqn (2) into eqn (6) gives us the following formulation: find \(u\) defined on \(\Omega\) such that...
\[ \int_{\Omega} \nabla u \cdot \nabla v = \int_{\Omega} vf + \int_{\partial\Omega} vg \]  
\[ \text{for all suitable test functions } v. \]

### 2.3 Finite Elements for Poisson’s Equation with Dirichlet conditions: Implementation and Review

#### Of Theory

##### 2.3.1 Weak Form

Given Poisson Equation:

\[ -\Delta u(x) = f(x) \text{ for all } x \in \Omega \]

\[ u = g(x) \text{ on } \partial\Omega \]

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 \(\Omega\) and performing integration by parts or by application of Divergence(GREEN) theorem. The result is

\[ \int_{\Omega} \nabla u \cdot \nabla v \, dx = \int_{\Omega} vf \, dx \]

\[ u = g(x) \text{ on } \partial\Omega \]

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

\[ \int_{\Omega} \nabla u_h \cdot \nabla v \, dx = \int_{\Omega} vf \, dx \quad \text{for all } v \in V_h \]

\[ u_h \text{ is called the finite element solution and functions in } V_h \text{ are called finite elements.} \]

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 \(\{\phi_j\}_{j=1}^N\) then we can write \(u_h = \sum_{j=1}^N \alpha_j \phi_j\). Substituting this in eqn(8) and choosing \(v\) to be a basis function \(\phi_i\) gives the following set of equations

\[ \sum_{j=1}^N \alpha_j \int_{\Omega} \nabla \phi_i \cdot \nabla \phi_j \, dx = \int_{\Omega} f \phi_i \, dx \quad i = 1, 2, 3, \ldots, N \]

This is really a linear system of the form

\[ K u = f \]

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

\[ K_{ij} = \int_{\Omega} \nabla \phi_i \cdot \nabla \phi_j \, dx \]

\[ f_i = \int_{\Omega} f \phi_i \, dx \]

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

In general, \(\Omega = \sum_{e=1}^{N_e} \Omega_e\), where \(N_e\) is the number of elements discretised in the domain \(\Omega\). 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 chosen 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 \(\Omega\)?
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 method is suited for solving the linear system?

In this paper, we present a 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 systems 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)

\[ K_{ij} = \int_{\Omega} \nabla \phi_i \cdot \nabla \phi_j \, dx \]

\[ \text{for all } i, j = 1, 2, 3, \ldots, N \]
The most obvious way to assemble the stiffness matrix is to compute the integrals $K_{ij}$ for the nodal pairs $i$ and $j$; this is a node oriented computation and we need to know the common support of basis functions $\varphi_i$ and $\varphi_j$. 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

$$K_{ij} = \sum_{e=1}^{N_e} \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$$

(12a)

where

$$K_{ij}^e = \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$$

(12b)

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

Notice that the integrals

$$K_{ij}^e = \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$$

look like the entries $K_{ij} = \int_{\Omega} \varphi_i \cdot \varphi_j \, d\mathbf{x} = \sum_{e=1}^{N_e} \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$ except the domain of integration is an element $\Omega^e$. So, we only need to save all entries of $K_{ij}^e = [K_{ij}^e]$ which corresponds to nodes on $\Omega^e$. Then if $\Omega^e$ has $d$ nodes, we can think of $K_{ij}^e$ as a $d \times d$ matrix. In view of the above, the procedure for computing the stiffness matrix is done on an element by element basis.

We must also compute the integrals

$$f_i = \int_{\Omega} \varphi_i \, d\mathbf{x} = \sum_{e=1}^{N_e} f_i^e$$

(12c)

where

$$f_i^e = \int_{\Omega^e} \varphi_i \, d\mathbf{x}$$

(12d)

Now further assume that on an element $\Omega^e$, $u_h = u^e = \sum_{j=1}^{d} u_j^e \varphi_j$.

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

$$\sum_{e=1}^{N_e} f_i^e = \int_{\Omega^e} \varphi_i \, d\mathbf{x}$$

(12e)

Where

$$u^e = (u_1^e, u_2^e, u_3^e, \ldots, u_d^e)^T \quad f^e = (f_1^e, f_2^e, f_3^e, \ldots, f_d^e)^T$$

(12f)

d refers to number of nodes per element, $N_e$ refers to the total number of elements in the domain $\Omega$.

### 2.3.5 Computing the Integrals $K_{ij}^e$ and $f_i^e$

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

(1) For each element $\Omega^e$, compute its local stiffness matrix $K_{ij}^e$. This requires computing the integrals $K_{ij}^e = \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$ which we compute by transforming to a reference element. In two dimensions $\Omega^e$ is an arbitrary linear triangle and each triangle will be further discretised into 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 element: the standard triangle (a right isosceles triangle) and further. Each triangle will be transformed to the corresponding reference element: 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 $K_{ij}^e = \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x}$ is given by

$$K_{ij}^e = \int_{\Omega^e} \varphi_i \cdot \varphi_j \, d\mathbf{x} = \int_{\Omega^e} \left( \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} + \frac{\partial \varphi_i}{\partial y} \frac{\partial \varphi_j}{\partial y} \right) \, d\mathbf{x}$$

(12g)

Where $S_{ij}^e = \int_{\Omega^e} \left( \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial y} + \frac{\partial \varphi_i}{\partial y} \frac{\partial \varphi_j}{\partial x} \right) \, d\mathbf{x}$ and $E = 3e + n, e = 1, 2, \ldots, N_e$ and $n = 0, 1, 2$

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 $\Omega^e$ is an arbitrary linear triangle) and bilinear transformations (when $\Omega^e$ is an arbitrary linear convex quadrilateral).

$$f_i^e = \int_{\Omega^e} \varphi_i \, d\mathbf{x}$$

can be computed in a straightforward manner if $f$ is a simple function otherwise we have to apply numerical integration.
For each element $\Omega^e$, first compute the local stiffness matrices $S^e = \begin{bmatrix} S_{ij}^e \end{bmatrix}$ 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 $\Omega^e$ which are discretised in the domain $\Omega$. In fact we have $\Omega = \sum_{e=1}^{N_e} \Omega^e = \sum_{e=1}^{N_e} \sum_{n=0}^{E} Q^e E = 3e + n - 2$.

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 local (Cartesian) coordinate system $(u, v)$ as in Fig 1a, which mapped into a 2-square in the local (natural) parametric coordinate $(\xi, \eta)$ as in Fig 1b.

\[ (u) = \sum_{k=1}^{4} \begin{pmatrix} u_k \\ v_k \end{pmatrix} M_k(\xi, \eta) \]

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 $(\xi, \eta)$ and they are given by

\[ M_k(\xi, \eta) = \frac{1}{4} (1 + \xi_k)(1 + \eta_k), \quad k = 1, 2, 3, 4 \]

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

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 $\phi, \psi$, etc governing the physical problem as

\[ (\phi, \psi) = \sum_{k=1}^{4} \begin{pmatrix} \phi_k \\ \psi_k \end{pmatrix} N_k(\xi, \eta) \]
Where $\phi_k, \psi_k$ 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[1].

2.4.3 Subparametric Transformation:

For the subparametric transformation over the nde – noded element we define the field variables $\phi, \psi$ (say) governing the physical problem as

$$
\begin{bmatrix}
\phi \\
\psi
\end{bmatrix} = \sum_{k=1}^{nde} \begin{bmatrix}
\phi_k^e \\
\psi_k^e
\end{bmatrix} N_k^e(\xi, \eta)
$$

(16)

Where $\phi_k, \psi_k$ 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 = 12, a 12 noded 2 square of cubic Serendipity type and nde=16, a 16-node 2-square of cubic Lagrange type.

3. Cubic order Linear Convex Quadrilateral Elements:

In this section, we give a brief description of the 12-node cubic Serendipity quadrilateral element under subparametric transformation as shown in Fig 1c, Fig 1d; and the 16-node cubic Lagrange quadrilateral element under subparametric transformation as shown in Fig 1e, Fig 1f.
We use the transform of Eqns.(13-14) to define the element geometry i.e.

\[
\begin{pmatrix}
  u(x_i) \\
  v(x_i)
\end{pmatrix} = \begin{pmatrix}
  u \\
  v
\end{pmatrix} = \sum_{k=1}^{4} \begin{pmatrix}
  u_k \\
  v_k
\end{pmatrix} M_k(\xi, \eta)
\]

Where

\[
M_k(\xi, \eta) = \frac{1}{4} (1 + \xi_k)(1 + \eta_k), \quad (k = 1, 2, 3, 4)
\]

(14a)

With \((u_k, v_k)\) are the vertices of the linear convex quadrilateral in global \((u, v)\) space.

\[
\left\{ \begin{array}{l}
(\xi_k, \eta_k), k = 1, 2, 3, 4 = \{(-1, -1), (1, -1), (1, 1), (-1, 1)\}
\end{array} \right.
\]

(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 \(\{(\xi_k, \eta_k), k = 1(1)12\}\) and \(\{(u_k, v_k) = (u(\xi_k, \eta_k), v(\xi_k, \eta_k)), k = 1(1)12\}\) for cubic Serendipity element.

Using the transformation of Eqns.(13-14) and from Fig 1e, Fig 1f; we see that there is a one to one correspondence between \(\{(\xi_k, \eta_k), k = 1(1)16\}\) and \(\{(u_k, v_k) = (u(\xi_k, \eta_k), v(\xi_k, \eta_k)), k = 1(1)16\}\) for cubic Lagrange element.

Where, the coordinates of the boundary nodes are:

\[
\{(\xi_k, \eta_k), k = 1(1)12\} = \{(-1, -1), (1, -1), (1, 1), (-1, 1), (-1/3, -1), (1/3, -1), (1, -1/3), (1, 1/3), (1/3, 1), (-1/3, 1), (-1, 1/3), (-1, -1/3)\}
\]

(14c)

and coordinates of the interior nodes are:

\[
\{(\xi_k, \eta_k), k = 13, 14, 15, 16\} = \{(-1/3, -1/3), (1/3, -1/3), (1/3, 1/3), (-1/3, 1/3)\}
\]

(14d)
\[
\begin{align*}
(u_5, v_5) &= (2u_1 + u_2)/3, (2v_1 + v_2)/3, \\
(u_9, v_9) &= (2u_1 + u_2)/3, (2v_1 + v_2)/3, \\
(u_7, v_7) &= ((2u_2 + u_3)/3, (2v_2 + v_3)/3, \\
(u_{10}, v_{10}) &= (u_2 + 2u_3)/3, (v_2 + 2v_3)/3, \\
(u_0, v_0) &= ([2u_3 + u_4]/3, (2v_3 + v_4)/3, \\
(u_{10}, v_{10}) &= (u_2 + 2u_3)/3, (v_2 + 2v_3)/3, \\
(u_{11}, v_{11}) &= (u_1 + u_2)/3, (v_1 + v_2)/3, \\
(u_{12}, v_{12}) &= (u_1 + u_2)/3, (v_1 + v_2)/3, \\
(u_{13}, v_{13}) &= ((4u_1 + 2u_2 + u_3 + u_4)/9, (4v_1 + 2v_2 + v_3 + v_4)/9, \\
(u_{14}, v_{14}) &= ((2u_1 + 4u_2 + 2u_3 + u_4)/9, (2v_1 + 4v_2 + 2v_3 + v_4)/9, \\
(u_{15}, v_{15}) &= ((u_1 + 2u_2 + 4u_3 + 2u_4)/9, (v_1 + 2v_2 + 4v_3 + 2v_4)/9, \\
(u_{16}, v_{16}) &= ((2u_1 + u_2 + 2u_3 + 4u_4)/9, (2v_1 + v_2 + 2v_3 + 4v_4)/9, \\
\end{align*}
\]

We then define the variation of physical variables \( \phi^e, \psi^e \) (say) over 12-node cubic Serendipity element of Fig 1c, 1d and 16-node cubic Lagrange element of Fig 1e, 1f by Eqn. (16) with \( \text{nde} = 12,16 \)

\[
(\phi^e, \psi^e) = \sum_{k=1}^{\text{nde}} N_k^e (\xi, \eta) \left( \phi_k^e, \psi_k^e \right) 
\]

Where \( \phi_k^e, \psi_k^e \) are the nodal values at node \( k \)

The shape functions \( N_k^e \) of the 12-node cubic Serendipity element shown in Fig 1c, Fig 1d are given by

\[
\begin{align*}
N_1^e (\xi, \eta) &= (1-\xi)(1-\eta)(-10+9(\xi^2+\eta^2))/32; \\
N_2^e (\xi, \eta) &= (1+\xi)(1-\eta)(-10+9(\xi^2+\eta^2))/32; \\
N_3^e (\xi, \eta) &= (1+\xi)(1+\eta)(-10+9(\xi^2+\eta^2))/32; \\
N_4^e (\xi, \eta) &= (1-\xi)(1+\eta)(-10+9(\xi^2+\eta^2))/32; \\
N_5^e (\xi, \eta) &= (9/32)(1-\xi)(1-\eta)(1-3\xi); \\
N_6^e (\xi, \eta) &= (9/32)(1-\xi)(1-\eta)(1+3\xi); \\
N_7^e (\xi, \eta) &= (9/32)(1+\xi)(1+\eta)(1-3\eta); \\
N_8^e (\xi, \eta) &= (9/32)(1+\xi)(1+\eta)(1+3\eta); \\
N_9^e (\xi, \eta) &= (9/32)(1-\xi)(1-\eta)(1-3\xi); \\
N_{10}^e (\xi, \eta) &= (9/32)(1+\xi)(1-\eta)(1-3\xi); \\
N_{11}^e (\xi, \eta) &= (9/32)(1+\xi)(1-\eta)(1+3\eta); \\
N_{12}^e (\xi, \eta) &= (9/32)(1+\xi)(1+\eta)(1-3\xi); \\
N_{13}^e (\xi, \eta) &= (9/32)(1+\xi)(1+\eta)(1+3\eta); \\
\end{align*}
\]

and we may check that

\[
\begin{align*}
N_k^e (\xi, \eta) &= 0, \text{when } j \neq k \\
\end{align*}
\]

The shape functions \( N_k^e \) of the 16-node cubic Serendipity element shown in Fig 1e, Fig 1f are given by

\[
\begin{align*}
N_1^e (\xi, \eta) &= -(9/16\xi^3+9/16\xi^2+1/16\xi+1/16)(-9/16\eta^3+9/16\eta^2+1/16\eta+1/16); \\
N_2^e (\xi, \eta) &= (9/16\xi^3+9/16\xi^2-1/16\xi+1/16)(9/16\eta^3+9/16\eta^2+1/16\eta+1/16); \\
N_3^e (\xi, \eta) &= (9/16\xi^2+9/16\xi^2-1/16\xi+1/16)(9/16\eta^2+9/16\eta+1/16\eta+1/16); \\
N_4^e (\xi, \eta) &= (9/16\xi+9/16\xi^2+1/16\xi+1/16)(9/16\eta^2+9/16\eta+1/16\eta+1/16); \\
N_5^e (\xi, \eta) &= -(9/16\xi^2+9/16\xi^2-1/16\xi+1/16)(9/16\eta^2+9/16\eta+1/16\eta+1/16); \\
N_6^e (\xi, \eta) &= -(9/16\xi+9/16\xi^2+1/16\xi+1/16)(9/16\eta^2+9/16\eta+1/16\eta+1/16); \\
N_7^e (\xi, \eta) &= 9/16(1-\xi^2)(1-\eta^2)(1-3\xi); \\
N_8^e (\xi, \eta) &= 9/16(1+\xi^2)(1+\eta^2)(1+3\xi); \\
N_9^e (\xi, \eta) &= 9/16(1-\xi^2)(1-\eta^2)(1-3\eta); \\
N_{10}^e (\xi, \eta) &= 9/16(1+\xi^2)(1+\eta^2)(1+3\eta); \\
N_{11}^e (\xi, \eta) &= 9/16(1-3\xi)(1-3\eta)(1-3\xi); \\
N_{12}^e (\xi, \eta) &= 9/16(1+3\xi)(1+3\eta)(1+3\xi); \\
N_{13}^e (\xi, \eta) &= 9/16(1-3\eta)(1+3\xi)(1+3\eta); \\
N_{14}^e (\xi, \eta) &= 9/16(1+3\eta)(1-3\xi)(1+3\eta); \\
N_{15}^e (\xi, \eta) &= 9/16(1+3\xi)(1+3\eta)(1-3\xi); \\
N_{16}^e (\xi, \eta) &= 9/16(1+3\eta)(1-3\xi)(1+3\eta); \\
\end{align*}
\]
H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-2348

\[
N_{e}^{v}(\xi, \eta) = (9/16\xi^{3} + 9/16\xi^{2} - 1/16\xi - 1/16\xi) (27/16\eta^{3} - 9/16\eta^{2} - 27/16\eta + 9/16);
N_{e}^{x}(\xi, \eta) = (9/16\eta^{3} + 9/16\eta^{2} - 1/16\eta - 1/16\eta) (27/16\xi^{3} - 9/16\xi^{2} + 27/16\xi + 9/16);
N_{e}^{z}(\xi, \eta) = (-27/16\xi^{3} + 9/16\xi^{2} + 27/16\xi + 9/16)(9/16\eta^{3} + 9/16\eta^{2} - 1/16\eta - 1/16);
N_{e}^{t}(\eta, \xi) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(9/16\eta^{3} + 9/16\eta^{2} - 1/16\eta - 1/16);
N_{e}^{e}(\xi, \eta) = (9/16\xi^{3} + 9/16\xi^{2} - 1/16\xi - 1/16\eta)(-27/16\eta^{3} + 9/16\eta^{2} + 27/16\eta + 9/16);
N_{e}^{f}(\xi, \eta) = (9/16\eta^{3} + 9/16\eta^{2} - 1/16\eta - 1/16\xi)(-27/16\xi^{3} + 9/16\xi^{2} + 27/16\xi + 9/16);
N_{e}^{g}(\xi, \eta) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(27/16\eta^{3} - 9/16\eta^{2} + 27/16\eta + 9/16);
N_{e}^{h}(\xi, \eta) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(27/16\eta^{3} - 9/16\eta^{2} + 27/16\eta + 9/16);
N_{e}^{i}(\xi, \eta) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(27/16\eta^{3} - 9/16\eta^{2} + 27/16\eta + 9/16);
N_{e}^{j}(\xi, \eta) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(27/16\eta^{3} - 9/16\eta^{2} + 27/16\eta + 9/16);
N_{e}^{k}(\xi, \eta) = (27/16\xi^{3} + 9/16\xi^{2} - 27/16\xi + 9/16)(27/16\eta^{3} - 9/16\eta^{2} + 27/16\eta + 9/16);
\]

and we may check that

\[N_{k}^{v}(\xi, \eta) = 1, N_{k}^{x}(\xi, \eta) = 0, \text{ when } j \neq k, k = 1(1)16\]

\(\xi_{k}, \eta_{k}\), \(k = 1, 12\) \(= \{(-1,-1),(1,-1),(1,1),(-1,1),(-1/3,-1),(1/3,-1),(1,-1/3),(1/3,1),(-1/3,1),(-1,-1/3)\}\) and \(\eta_{k}\), \(k = 1, 12\) \(= \{-1/3,-1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3\}\)

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 \((\xi, \eta)\) as in Fig 1b, d.

From the Eq.(1) and Eq.(2), we have

\[
\frac{\partial u}{\partial \xi} = \sum_{k=1}^{4} u_{k} \frac{\partial M_{k}}{\partial \xi} = \frac{1}{4} \left[ (-u_{1} + u_{2} + u_{3} - u_{4}) + (u_{1} - u_{2} + u_{3} - u_{4}) \xi \right] \quad (18a)
\]
\[
\frac{\partial u}{\partial \eta} = \sum_{k=1}^{4} u_{k} \frac{\partial M_{k}}{\partial \eta} = \frac{1}{4} \left[ (-u_{1} - u_{2} + u_{3} + u_{4}) + (u_{1} - u_{2} + u_{3} + u_{4}) \eta \right] \quad (18b)
\]
\[
\frac{\partial v}{\partial \xi} = \frac{1}{4} \left[ (-v_{1} + v_{2} + v_{3} - v_{4}) + (v_{1} - v_{2} + v_{3} - v_{4}) \xi \right] \quad (18c)
\]
\[
\frac{\partial v}{\partial \eta} = \frac{1}{4} \left[ (-v_{1} - v_{2} + v_{3} + v_{4}) + (v_{1} - v_{2} + v_{3} + v_{4}) \eta \right] \quad (18d)
\]

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

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

Where

\[
\alpha = \frac{1}{8} \left[ (u_{4} - u_{2})(v_{1} - v_{3}) + (u_{3} - u_{1})(v_{4} - v_{2}) \right]
\]
\[
\beta = \frac{1}{8} \left[ (u_{4} - u_{3})(v_{2} - v_{4}) + (u_{1} - u_{2})(v_{4} - v_{3}) \right]
\]
\[
\gamma = \frac{1}{8} \left[ (u_{4} - u_{1})(v_{2} - v_{3}) + (u_{3} - u_{2})(v_{4} - v_{1}) \right] \quad (19b)
\]

4.2 Global Derivatives:

If \(N_{f}^{i}\) 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 \([1, 2, 3]\)
Where $\frac{\partial y_i}{\partial x_i}$ and $\frac{\partial y_j}{\partial x_j}$ 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 = 12,16 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 discretisation of these triangles further into linear convex quadrilaterals. This is stated in the following Lemma [26].

We first consider an arbitrary triangle $\Delta PQR$ in the Cartesian space $(x, y)$ with vertices $P(x_p, y_p), Q(x_q, y_q)$, and $R(x_r, y_r)$. Let $Z((x_p + x_q + x_r)/3, (y_p + y_q + y_r)/3)$ be its centroid and also let $S$, $T$, $U$ be the midpoints of sides $PQ$, $QR$, and $RP$ respectively. Now by joining the centroid $Z$ to the midpoints $S$, $T$, $U$ by straight lines, we divided the triangle $\Delta PQR$ into three special quadrilaterals $Q_1$, $Q_2$, and $Q_3$ (say) which are spanned by vertices $<Z, U, P, S>$, $<Z, S, Q, T>$, and $<Z, T, R, U>$ respectively. This is shown in Fig. 2.0 a

We next consider the standard triangle $\Delta ABC$ in the Cartesian space $(u, v)$ with vertices, centroid and midpoints $A(1, 0)$, $B(0, 1)$, $C(0, 0)$, $G(\frac{1}{3}, \frac{1}{3})$, $D(\frac{1}{2}, \frac{1}{2})$, $E(0, \frac{1}{2})$ and $F(\frac{1}{2}, 0)$. We now divide the $\Delta ABC$ into three special quadrilaterals $\bar{Q}_1$, $\bar{Q}_2$ and $\bar{Q}_3$ (say) which are spanned by vertices $<G, E, C, F>$, $<G, F, A, D>$ and $<G, D, B, E>$ respectively. This is shown in Fig. 2.0 b

We apply linear transformations to map an arbitrary triangle into a triangle of our choice. In this section, we use the well known transformation which maps an arbitrary triangle into a standard triangle (a right isosceles triangle). We assume the special discretization scheme of the previous section for the following developments

5.1 Lemma 1. There exists a unique linear transformation to map the special quadrilaterals $Q_i$ in to $\bar{Q}_i$ $(i = 1,2,3)$ satisfying the conditions

(i) $\sum_{i=1}^{3} Q_i = \Delta PQR$, the arbitrary triangle in the $(x, y)$ space.

(ii) $\sum_{i=1}^{3} \bar{Q}_i = \Delta ABC$, the standard triangle (right isosceles triangle) in the $(u, v)$ space

then the Jacobian $J^e$ for each element $Q_e$, $(e=1,2,3)$ is given by

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

Where $\Delta pqr$ is the area of the triangle $\Delta PQR$

$$2\Delta pqr = \begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix} = \left[ (x_p - x_r)(y_q - y_r) - (x_q - x_r)(y_p - y_r) \right] \quad \text{-----------------(21b)}$$
Proof: We shall now refer to Fig 2.0 a, b for the following developments and consider the following linear transformation between (x, y) and (u, v) spaces.

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix} = \begin{pmatrix}
  x_p \\
  y_q
\end{pmatrix} + \begin{pmatrix}
  x_q \\
  y_r
\end{pmatrix} u + \begin{pmatrix}
  x_r \\
  y_p
\end{pmatrix} v , w = 1 - u - v .
\]

.................................(22)

We can verify that the above transformation of eqn. Above maps the arbitrary triangle \( \Delta PQR \) into the standard triangle \( \Delta ABC \). The points P, Q, R, S, T, U and Z are respectively mapped into the points A, B, C, D, E, F and G respectively. The quadrilaterals \( Q_i \) are mapped into quadrilaterals \( \tilde{Q}_i \). This proves the existence of the required transformation.

5.2 Lemma 2. There exists three linear transformations to map the quadrilaterals \( Q_i (i = 1,2,3) \) of \( \Delta PQR \) into a unique triangle \( \tilde{Q} = Q_i \) (say) of the standard triangle \( \Delta ABC \) which satisfy the following conditions

(i) \( \Sigma_{i=1}^3 Q_i = \Delta PQR \), the arbitrary triangle in the (x, y) space.

(ii) \( \Sigma_{i=1}^3 \tilde{Q}_i = \Delta ABC \), the standard triangle (right isosceles) in the (u, v) space

Proof: We again refer to Fig 1a, 1b and consider the following linear transformations between (x, y) and (u, v) spaces.

\[
\begin{pmatrix}
  x^{(1)} \\
  y^{(1)}
\end{pmatrix} = \begin{pmatrix}
  x_p \\
  y_q
\end{pmatrix} w + \begin{pmatrix}
  x_q \\
  y_r
\end{pmatrix} u + \begin{pmatrix}
  x_r \\
  y_p
\end{pmatrix} v , w = 1 - u - v ,
\]

.................................(23)

\[
\begin{pmatrix}
  x^{(2)} \\
  y^{(2)}
\end{pmatrix} = \begin{pmatrix}
  x_p \\
  y_q
\end{pmatrix} w + \begin{pmatrix}
  x_q \\
  y_r
\end{pmatrix} u + \begin{pmatrix}
  x_r \\
  y_p
\end{pmatrix} v , w = 1 - u - v ,
\]

.................................(24)

\[
\begin{pmatrix}
  x^{(3)} \\
  y^{(3)}
\end{pmatrix} = \begin{pmatrix}
  x_p \\
  y_q
\end{pmatrix} w + \begin{pmatrix}
  x_q \\
  y_r
\end{pmatrix} u + \begin{pmatrix}
  x_r \\
  y_p
\end{pmatrix} v , w = 1 - u - v ,
\]

.................................(25)

It is quite clear that each of the above transformations map the arbitrary triangle \( \Delta PQR \) into \( \Delta ABC \). We may further note the following
(i) The transformation of eqn.(2) maps the vertices P, Q, R in (x, y) space into vertices C(0, 0), A(1, 0), B(0, 1) in (u, v) space.

(ii) The transformation of eqn.(3) maps the vertices Q, R, P in (x, y) space into vertices C(0, 0), A(1, 0), B(0, 1) in (u, v) space.

(iii) The transformation of eqn.(4) maps the vertices R, P, Q in (x, y) space into vertices C(0, 0), A(1, 0), B(0, 1) in (u, v) space.

We can now verify that the transformation of eqn.(2) maps the quadrilateral Q₁ spanning the vertices < Z, U, P, S > in (x, y) space into the quadrilateral ̂Q₁ spanning the vertices < G, E, C, F > in the (u, v) space. In a similar manner, we find that using the transformation of eqn.(3), the quadrilateral Q₂ spanned by vertices < Z, S, Q, T > in (x, y) space is mapped into the quadrilateral ̂Q₂ spanned by the vertices < G, E, C, F > in the (u, v) space. Finally on using the linear transformation of eqn.(4), the quadrilateral Q₃ spanned by vertices < Z, T, R, U > in (x, y) space is mapped into the quadrilateral ̂Q₃ spanning the vertices < G, E, C, F > in the (u, v) space. This completes the proof of Lemma 2.

We may note here that the linear transformations in eqn.(23) and in eqn.(22) are identical. We wish to say in advance that the application of the above lemmas will be of immense help in the development of this paper.

5.3 Lemma 3. Let ∆ABC be an arbitrary triangle with the vertices A(1, 0), B(0, 1) and C(0, 0) and let D(½, ½), E(0, ½) and F(½, 0) be midpoints of sides AB, BC and CA respectively and also let G(1/3, 1/3) be its centroid. Then the Jacobian of the three special quadrilaterals ̂Q₁ (e = 1,2,3) viz. < G, E, C, F > , <G, F, A, D> and <G, D, B, E> have the same expression given by

$$\hat{f} = \hat{f}^e = \frac{1}{96}(4 + \xi + \eta), \quad (e = 1,2,3)$$

**Proof:** We can immediately verify that eqn.(10a) is true by substituting the nodal values of ̂Q₁ in eqn.(9a-b). The general result for a special quadrilateral ̂Qₖ (e = 1,2,3) follows by direct substitution of geometric coordinates of the vertices in eqns.(9a - b) or by chain rule of partial differentiation and use of eqn.(1):

$$J = J^e = \frac{\partial(x,y)}{\partial(\xi,\eta)} \frac{\partial(x,y)}{\partial(\xi,\eta)} = \frac{1}{96}(4 + \xi + \eta) = \frac{δ_{pqr}}{48}(4 + \xi + \eta)$$

We have shown in the foregoing Lemma that an arbitrary linear triangle can be discretised into three linear convex quadrilaterals. Further, each of these quadrilaterals in xy plane can be mapped into a unique linear convex quadrilateral spanned by the vertices G(1/3, 1/3), E(0, ½), C(0, 0), F(½, 0) using a proper linear transformation as given Eqn.(23) – (25).

We note that the quadrilaterals can be converted into 12-node cubic Serendipity and 16-node cubic Lagrange element by placing additional nodes on the boundary and interior of the quadrilaterals Q₁ (i = 1,2,3) in (x,y) space , ̂Q₁ (i = 1,2,3) in (u,v) space and the 2-square in (ξ, η) space.

6. Integration over a Triangular Region :

6.1 Composite Integration

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

$$\Pi_{\Delta PQR} = \int_{\Delta PQR} \phi(x,y) \, dxdy = \sum_{e=1}^{3} \int_{\Delta e} \phi(x,y) \, dxdy \quad \text{------------------ (26)}$$

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

$$= (2\Delta_{pqr}) \int_{\hat{Q}} \{ \sum_{e=1}^{3} \phi(x^{(e)}(u,v), y^{(e)}(u,v)) \} \, dudv \quad \text{----------------- (27)}$$
Where \((x^{(e)}(u, v), y^{(e)}(u, v))\), \(e = 1, 2, 3\) are the linear transformations of Eqs.(23)–(25) and \(\mathbf{Q}\) is the linear convex 4-node quadrilateral GECF spanning the vertices \(G(1/3, 1/3), E(0, 1/2), C(0, 0), F(1/2, 0)\) and \(\Delta_{PQR}\) is the area of triangle \(\Delta_{PQR}\). Now, we further use the bilinear transformation of Eqs.(1)–(2) in Eqn.(15) and obtain.

\[
\mathbf{II}_{\Delta_{PQR}} = (2 \Delta_{PQR}) \int_{-1}^{1} \int_{-1}^{1} \left[ \sum_{e=1}^{3} \phi \left( x^{(e)}(u, v), y^{(e)}(u, v) \right) \frac{\partial u}{\partial \xi} \right] \mathrm{d}x \mathrm{d}\eta \tag{28}
\]

In Eqn.(16) we have used the bilinear transformation given in Eqs.(13)–(14)

\[
u = v(\xi, \eta) = \frac{1}{3} M_2(\xi, \eta) + \frac{1}{2} M_4(\xi, \eta)
\]

\[
u = v(\xi, \eta) = \frac{1}{3} M_2(\xi, \eta) + \frac{1}{2} M_4(\xi, \eta)
\]

to map the arbitrary linear convex 4 noded quadrilateral into a \(2 \times 2\) square in \((\xi, \eta)\) – plane. Thus on using Eqn.(29), the integral of Eqn.(28) simplifies to the following.

\[
\mathbf{II}_{\Delta_{PQR}} = (2 \Delta_{PQR}) \int_{-1}^{1} \int_{-1}^{1} \left[ \sum_{e=1}^{3} \left( \frac{4+\xi+\eta}{96} \right) \phi \left( x^{(e)}(u, v), y^{(e)}(u, v) \right) \right] \mathrm{d}x \mathrm{d}\eta \tag{30}
\]

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

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

\[
\mathbf{II}_{\Delta_{PQR}} = 2\Delta_{PQR} \sum_{i=1}^{N} \sum_{j=1}^{N} \left( \frac{W_i^{(N)} W_j^{(N)} (4+\xi_i+\eta_j)}{96} \right) \sum_{e=1}^{3} \phi \left( x^{(e)}(u_{ij}, v_{ij}), y^{(e)}(u_{ij}, v_{ij}) \right)
\]

\[
\tag{31}
\]

Where from Eqn.(29), we write

\[
u_{ij}^{(N)} = \nu(\xi_i, \eta_j)
\]

\[
u_{ij}^{(N)} = \nu(\xi_i, \eta_j)
\]

and \((W_i^{(N)}, \xi_i^{(N)})\), \((W_j^{(N)}, \eta_j^{(N)})\) are the weight coefficients and sampling points along \(\xi, \eta\) 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 \(N_{i}^{(e)}(i=1(1)9)\) denotes the basis functions for node \(i\) of a linear convex 9- node linear convex quadrilateral element \(e\), then by use of chain rule of partial differentiation

\[
\frac{\partial N_{i}^{(e)}}{\partial x} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix} N_{i}^{(e)}
\]

\[
\tag{33}
\]

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

We must now use the earlier transformations.

\[
(x, y) = (x_{yp} + \frac{y_{yp}}{y_{yp}}) u + (x_{yp} - x_{yp}) v
\]

\[
\text{for } Q_1 \text{ in } \Delta_{PQR} \tag{23}
\]
\[
\begin{align*}
(x^2_y) &= (x_q + \frac{x_q - x_p}{y_q - y_p}) u + \frac{x_q - x_p}{y_q - y_p} v \quad \text{for } Q_2 \text{ in } \Delta PQR \hspace{1cm} (24) \\
(y^2_y) &= (y_r + \frac{x_p - x_q}{y_r - y_p}) u + \frac{x_p - x_q}{y_r - y_p} v \quad \text{for } Q_3 \text{ in } \Delta PQR \hspace{1cm} (25)
\end{align*}
\]

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

\[
\begin{align*}
\begin{pmatrix}
x \\
y
\end{pmatrix} &= \begin{pmatrix}
x_c + \frac{x_c - x_a}{y_c - y_a} u + \frac{x_c - x_a}{y_c - y_a} v \\
y_c + \frac{x_c - x_a}{y_c - y_a} u + \frac{x_c - x_a}{y_c - y_a} v
\end{pmatrix} \hspace{1cm} (34)
\end{align*}
\]

which can map an arbitrary triangle \( \Delta ABC \), \( A(x_a, y_a), B(x_b, y_b), C(x_c, y_c) \) in \( xy \)-plane into a right isosceles triangle in the \( uv \)-plane.

Hence, we have from Eqn.(34)

\[
\begin{align*}
\begin{pmatrix}
u \\
v
\end{pmatrix} &= \begin{pmatrix}
(x_a - x_c) & (x_b - x_c) \\
y_a - y_c & y_b - y_c
\end{pmatrix}^{-1} \begin{pmatrix}
x - x_c \\
y - y_c
\end{pmatrix} \hspace{1cm} (35)
\end{align*}
\]

This gives

\[
\begin{align*}
u &= (\alpha_a + \beta_a x + \gamma_a y)/(2 \Delta_{abc}) \\
v &= (\alpha_b + \beta_b x + \gamma_b y)/(2 \Delta_{abc}) \hspace{1cm} (36)
\end{align*}
\]

where

\[
\begin{align*}
\alpha_a &= (x_a - x_c, x_a, y_a) \\
\alpha_b &= (y_b - y_c) \\
\beta_a &= (y_b - y_c) \\
\beta_b &= (x_a - x_c) \\
\gamma_a &= (x_b - x_c) \\
\gamma_b &= (y_a - y_c) \hspace{1cm} (37a)
\end{align*}
\]

and

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

\[
= (y_b \beta_a - y_a \beta_b) \hspace{1cm} (37b)
\]

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

\[
\begin{align*}
\begin{pmatrix}
\frac{\partial \Phi}{\partial x} \\
\frac{\partial \Phi}{\partial y}
\end{pmatrix} &= \begin{pmatrix}
\beta_a^* \\
\gamma_a^*
\end{pmatrix} \begin{pmatrix}
\frac{\partial \Phi}{\partial u} \\
\frac{\partial \Phi}{\partial v}
\end{pmatrix} \hspace{1cm} (38a)
\end{align*}
\]

where

\[
\begin{align*}
\beta_a^* &= \frac{\beta_a}{(2 \Delta_{abc})} \hspace{1cm} , \hspace{1cm} \beta_b^* = \frac{\beta_b}{(2 \Delta_{abc})} \\
\gamma_a^* &= \frac{\gamma_a}{(2 \Delta_{abc})} \hspace{1cm} , \hspace{1cm} \gamma_b^* = \frac{\gamma_b}{(2 \Delta_{abc})} \hspace{1cm} (38b)
\end{align*}
\]

Letting,

\[
D_{x,y}^{L,e} = \begin{pmatrix}
\frac{\partial \Phi}{\partial u} \\
\frac{\partial \Phi}{\partial v}
\end{pmatrix}, \quad P = \begin{pmatrix}
\beta_a^* \\
\gamma_a^*
\end{pmatrix}, \quad D_{u,v}^{L,e} = \begin{pmatrix}
\frac{\partial \Phi}{\partial u} \\
\frac{\partial \Phi}{\partial v}
\end{pmatrix} \hspace{1cm} (39)
\]

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

\[
D_{x,y}^{L,e} = P D_{u,v}^{L,e} \hspace{1cm} (40)
\]

Hence from Eqn.(39) and Eqn.(40)
\[ G_{i,j} = \begin{pmatrix} \frac{\partial N_i^e}{\partial x} & \frac{\partial N_i^e}{\partial y} \\ \frac{\partial N_j^e}{\partial x} & \frac{\partial N_j^e}{\partial y} \end{pmatrix} (D^e_{i,j})^T \]

\[ G_{i,j} = \begin{pmatrix} \frac{\partial N_i^e}{\partial x} & \frac{\partial N_i^e}{\partial y} \\ \frac{\partial N_j^e}{\partial x} & \frac{\partial N_j^e}{\partial y} \end{pmatrix} (D^e_{i,j})^T \]

\[ G_{i,j} = \begin{pmatrix} \frac{\partial N_i^e}{\partial u} & \frac{\partial N_i^e}{\partial v} \\ \frac{\partial N_j^e}{\partial u} & \frac{\partial N_j^e}{\partial v} \end{pmatrix} (D^e_{i,j})^T \]

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

\[ G_{i,j} = (P D^e_{u,v}) (D^e_{i,j})^T \]

\[ = P (D^e_{u,v}) (D^e_{i,j})^T \]

\[ = P G_{i,j}^e P^T \]

We now define the submatrices of global derivative integrals in (x,y) and (u,v) space associated with the nodes i and j \((i,j = 1, 2, 3, 4, 5, 6, 7, 8, 9)\) as

\[ S_{i,j} = \int_{Q_e} G_{i,j}^e \ dx \ dy \]

\[ K_{i,j} = \int_{\bar{Q}} G_{i,j}^e \ du \ dv \]

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

\[ S_{i,j} = \int_{Q_e} G_{i,j}^e \ dx \ dy = \begin{pmatrix} \int_{Q_e} \frac{\partial N_i^e}{\partial x} \frac{\partial N_j^e}{\partial x} \ dx \ dy \\ \int_{Q_e} \frac{\partial N_i^e}{\partial y} \frac{\partial N_j^e}{\partial y} \ dx \ dy \end{pmatrix} = \begin{pmatrix} S_{i,j}^{e1-1,2j-1} \\ S_{i,j}^{e2,2j-1} \end{pmatrix} \]

and in similar manner

\[ K_{i,j} = \int_{\bar{Q}} G_{i,j}^e \ du \ dv = \begin{pmatrix} \int_{\bar{Q}} \frac{\partial N_i^e}{\partial u} \frac{\partial N_j^e}{\partial u} \ du \ dv \\ \int_{\bar{Q}} \frac{\partial N_i^e}{\partial v} \frac{\partial N_j^e}{\partial v} \ du \ dv \end{pmatrix} = \begin{pmatrix} K_{i,j}^{e1-1,2j-1} \\ K_{i,j}^{e2,2j-1} \end{pmatrix} \]

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

\[ S_{i,j}^e = \int_{Q_e} (P G_{i,j}^e P^T) \frac{\partial (x,y)}{\partial (u,v)} \ du \ dv \]
\[
\begin{align*}
&= 2\Delta_{abc} \int_Q \left( P \, G_{u,v}^{i,j,e} \right) \, du \, dv \quad = 2\Delta_{abc} \int_Q \left( \int_{u,v} \frac{\partial (u,v)}{\partial (\xi,\eta)} \right) \, du \, dv \quad \forall (i,j) = 1, 2, 3, 4, 5, 6, 7, 8, 9) \quad \text{(46)}
\end{align*}
\]

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 \(Q\) is the 8- node linear convex quadrilateral in \((u, v)\) space spanned by the vertices \((1/3, 1/3), (1/6, 5/12), (0, 1/2), (0, 1/4), (0, 0), (1/4, 0), (1/2, 0), (5/12, 1/6)\) and \((5/24, 5/24)\) in uv- plane hence from Eqn.(45)

\[
K_{ij}^{i,j,e} = \int_Q G_{u,v}^{i,j,e} \, du \, dv \quad \text{(47)}
\]

\[
\int \int \quad \text{(48)}
\]

We now refer to section 6.1 of this paper, in this section, we have derived the necessary relations to integrate Eqn.(47). As in Eqns.(27)-(28), we use the transformation of Eqn.(29) to map the 9- node quadrilateral \(Q\) to the 9- node 2-square \(-1 \leq \xi, \eta \leq 1\) Using Eqn.(29) in Eqn.(48), we obtain

\[
K_{ij}^{i,j,e} = \int_{-1}^{1} \int_{-1}^{1} G_{u,v}^{i,j,e} \left( \frac{\partial (u,v)}{\partial (\xi,\eta)} \right) \, d\xi \, d\eta \quad \text{(49)}
\]

Thus, we have from Eqn.(46)

\[
S_{ij}^{i,j,e} = \left( 2\Delta_{abc} \right) \quad \text{P} \left( K_{ij}^{i,j,e} \right) \quad \text{P}^T \quad \text{(50)}
\]

Where \(K_{ij}^{i,j,e}\) is given in Eqn.(49).

In Eqn.(50), \(2\Delta_{abc} = 2 \times \text{area of the triangle spanning vertices } A(x_a,y_a), B(x_b,y_b), C(x_c,y_c) \) is a scalar.

The matrices \(P, P^T\) depend purely on the node coordinates \((x_a,y_a), (x_b,y_b), (x_c,y_c)\) the matrix \(K_{ij}^{i,j,e}\) can be explicitly computed by the relations obtained in section 2 – 6. We find that \(K_{ij}^{i,j,e}\) 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 explicitly 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 \((2xn_{de}) \times (2xn_{de}), n_{de} = 8 = \) 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 \(Q\) 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)

\[
K_{ij}^{f} = \int_{Q_e} \nabla \varphi_i \cdot \nabla \varphi_j \, dx = \int_{Q_e} \left( \frac{\partial \varphi_i}{\partial x} + \frac{\partial \varphi_i}{\partial y} \right) \, dx dy \quad 
\]

\[
\text{......................................(51a)}
\]

from the above derivations, we can rewrite \(K_{ij}^{f}\) in the notations of this sections by taking \(\varphi_i = N_i \) and \(\varphi_j = N_j \) and \(\Omega^e = Q_e \) so that

\[
K_{ij}^{f} = \int_{Q_e} \nabla N_i \cdot \nabla N_j \, dx = \int_{Q_e} \left( \frac{\partial N_i}{\partial x} + \frac{\partial N_i}{\partial y} \right) \, dx dy = S_{ij}^{2i-1,2j-1} + S_{ij}^{2i,2j} \quad 
\]

\[
\text{......................................(51b)}
\]
6.3 Computation of $K_{ij}^e$

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 several 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 $F_{pq}^{ij} = \frac{\partial N_i \partial N_j}{\partial x \partial y} F_{pq} = \int_{Q_e} F_{pq}^{ij} dp dq$, then we have from eqns(44-45):

\[
S_{ij,e} = \int_{Q_e} G_{x,y}^{ij,e} dx dy = \begin{pmatrix}
\int_{Q_e} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx dy & \int_{Q_e} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} dx dy \\
\int_{Q_e} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} dx dy & \int_{Q_e} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} dx dy 
\end{pmatrix}
\]

\[
= \begin{pmatrix}
S_{2i-1,2j-1}^e & S_{2i,2j}^e \\
S_{2i-1,2j-1}^e & S_{2i,2j}^e 
\end{pmatrix}
\]

\[
= \begin{pmatrix}
I_{xx}^{ij} & I_{xy}^{ij} \\
I_{yx}^{ij} & I_{yy}^{ij}
\end{pmatrix}
\]

\[\text{(say)}\] \hspace{1cm} (52a)

\[
K_{ij,e} = \int_{Q} G_{u,x}^{ij,e} du dv = \begin{pmatrix}
\int_{Q} \frac{\partial N_i}{\partial u} \frac{\partial N_j}{\partial u} du dv & \int_{Q} \frac{\partial N_i}{\partial v} \frac{\partial N_j}{\partial v} du dv \\
\int_{Q} \frac{\partial N_i}{\partial v} \frac{\partial N_j}{\partial u} du dv & \int_{Q} \frac{\partial N_i}{\partial u} \frac{\partial N_j}{\partial v} du dv 
\end{pmatrix}
\]

\[
= \begin{pmatrix}
K_{2i-1,2j-1}^e & K_{2i,2j}^e \\
K_{2i-1,2j-1}^e & K_{2i,2j}^e 
\end{pmatrix}
\]

\[
= \begin{pmatrix}
I_{uu}^{ij} & I_{uv}^{ij} \\
I_{vu}^{ij} & I_{vv}^{ij}
\end{pmatrix}
\]

\[\text{(say)}\] \hspace{1cm} (52b)

Let $P_1 = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$, $P_2 = \begin{pmatrix} P_{11} & P_{21} \\ P_{12} & P_{22} \end{pmatrix}$

\[\text{(53)}\]

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

\[
S_{ij,e} = \int_{Q_e} G_{x,y}^{ij,e} dx dy = 2\Delta_{abc} P \left( \int_{Q} G_{u,x}^{ij,e} du dv \right) P^T
\]

\[= 2\Delta_{abc} \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} I_{uu}^{ij} & I_{uv}^{ij} \\ I_{vu}^{ij} & I_{vv}^{ij} \end{pmatrix} \begin{pmatrix} P_{11} & P_{21} \\ P_{12} & P_{22} \end{pmatrix}
\]

\[= 2\Delta_{abc} \left( \begin{pmatrix} \{ P_{11}(P_{11}I_{uu}^{ij} + P_{12}I_{uv}^{ij}) + P_{12}(P_{11}I_{uv}^{ij} + P_{12}I_{vv}^{ij}) \} & \{ P_{11}(P_{21}I_{uu}^{ij} + P_{22}I_{uv}^{ij}) + P_{12}(P_{21}I_{uv}^{ij} + P_{22}I_{vv}^{ij}) \} \\
\{ P_{21}(P_{11}I_{uu}^{ij} + P_{12}I_{uv}^{ij}) + P_{22}(P_{11}I_{uv}^{ij} + P_{12}I_{vv}^{ij}) \} & \{ P_{21}(P_{21}I_{uu}^{ij} + P_{22}I_{uv}^{ij}) + P_{22}(P_{21}I_{uv}^{ij} + P_{22}I_{vv}^{ij}) \} \end{pmatrix} \right)
\]

\[\text{............................................................................................}(54)\]

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

\[
\text{trace} \ (S_{ij,e}) = \text{trace}(\int_{Q_e} G_{x,y}^{ij,e} dx dy) = (S_{2i-1,2j-1}^e + S_{2i,2j}^e) = K_{ij}^e = \int_{Q_e} \nabla N_i \cdot \nabla N_j dx = \int_{Q_e} (\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y}) dx dy
\]

\[= (P_{11}^2 + P_{21}^2)(I_{uu}^{ij} + (P_{11} + P_{21}) (I_{uv}^{ij} + I_{vu}^{ij} + (P_{12}^2 + P_{22}^2) I_{vv}^{ij}) \]

\[\text{............................................................................................}(55)\]

We can obtain the above integral $\int_{Q_e} (\frac{\partial N_i}{\partial x} + \frac{\partial N_i}{\partial y}) dx dy$ by use of matrix operations which does not need the computation matrix triple product. This procedure is presented below
From eqn (44b) and eqn(45), let us do the following:

\[
(p^T p)_s \begin{pmatrix}
I_{u,h}^{i,j} & I_{v,h}^{i,j} \\
I_{u,v}^{i,j} & I_{v,v}^{i,j}
\end{pmatrix} = \begin{pmatrix}
(P_{11}^2 + P_{21}^2)I_{u,h}^{i,j} & (P_{11} P_{12} + P_{21} P_{22})I_{v,h}^{i,j} \\
(P_{11} P_{12} + P_{22} P_{21})I_{u,v}^{i,j} & (P_{12}^2 + P_{22}^2)I_{v,v}^{i,j}
\end{pmatrix}
\]

\[
(A)(56)
\]

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

\[
\int_0^1 \frac{\partial N_i}{\partial x} dx + \int_0^1 \frac{\partial N_i}{\partial y} dy \ dx dy = sum(sum((p^T p)_s \begin{pmatrix}
I_{u,h}^{i,j} & I_{v,h}^{i,j} \\
I_{u,v}^{i,j} & I_{v,v}^{i,j}
\end{pmatrix}))
\]

\[
(A)(57)
\]

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 $\int_{\Omega^v} f \varphi_i \ dx dy$

We shall now propose numerical integration for the complicated integrands in the force vector integrals over the domain $\Omega^v$ which is an arbitrary linear triangle and $\varphi(x,y) = f \varphi_i$. We also refer to the section 2 for the theory necessary to derive the composite numerical integration formula.

We shall now establish a composite integration formula for an arbitrary linear triangular region $\Delta PQR$ shown in Fig 2a or Fig 3a. We have for an arbitrary smooth function $\varphi(x,y)$

\[
\Pi_{\Delta PQR} = \int_{\Delta PQR} \varphi(x,y) \ dx dy = \sum_{s=1}^{3} \int_{Q_s} \varphi(x,y) \ dx dy
\]

\[
(A)(58)
\]

\[
= \int_{Q} \Sigma_{s=1}^{3} [\varphi(x^{(e)}(u,v), y^{(e)}(u,v))] \frac{\partial [x^{(e)}(u,v), y^{(e)}(u,v)]}{\partial (u,v)} \ dudv
\]

\[
= (2 \Delta_{pq} r) \int_{Q} \{ \Sigma_{s=1}^{3} [\varphi(x^{(e)}(u,v), y^{(e)}(u,v))] \} \ dudv
\]

\[
(A)(59)
\]

Where $(x^{(e)}(u,v), y^{(e)}(u,v))$, $e = 1, 2, 3$ are the transformations of Eqs.(8)--(10) and $Q$ is the quadrilateral in uv-plane spanned by vertices $G(1/3, 1/3)$, $E(0, 1/2)$, $C(0, 0)$ and $F(1/2,0)$, and $\Delta_{pq}$ is the area of triangle $\Delta PQR$. Now using the transformations defined in Eqs.(1)--(2) we obtain

\[
\Pi_{\Delta PQR} = \int_{Q} \{ \Sigma_{s=1}^{3} [\varphi(x^{(e)}(u,v), y^{(e)}(u,v))] \} \ d\xi \ d\eta
\]

\[
(A)(60)
\]

In Eq.(14) we have used the transformation

\[
u(\xi, \eta) = \frac{1}{3} N_1(\xi, \eta) + 2 \frac{1}{3} N_2(\xi, \eta)
\]

\[
(A)(61)
\]

to map the quadrilateral $Q$ into a 2–square in $\xi \eta$–plane.

We can now obtain from Eqs.(13)--(14)

\[
\Pi_{\Delta PQR} = \int_{-1}^{1} \int_{-1}^{1} \{ \Sigma_{s=1}^{3} \left( \frac{4+h^2}{h^2} \right) [\varphi(x^{(e)}(u,v), y^{(e)}(u,v)) \} \ d\xi \ d\eta
\]

\[
(A)(62)
\]

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

Using Numerical Integration :

\[
\Pi_{\Delta PQR} = 2 \Delta_{pq} \Sigma_{s=1}^{3} \Sigma_{i=1}^{N} \left( \frac{W_i^{(N)} W_j^{(N)} (4+4h^2) \eta(N)}{96} \right) \Sigma_{s=1}^{3} \varphi(x^{(e)}(u_{i,j}, v^{(e)}_{i,j}), y^{(e)}(u_{i,j}, v^{(e)}_{i,j}))
\]

\[
(A)(63)
\]

Where,
\( u_{ij}^{(N)} = u(\xi_i^{(N)}, \eta_j^{(N)}) \) and \( v_{ij}^{(N)} = v(\xi_i^{(N)}, \eta_j^{(N)}) \)

\[ (64) \]

and \((W_1^{(N)}, \xi_1^{(N)}), (W_2^{(N)}, \xi_2^{(N)})\) 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 \( \int_{\Omega} f \varphi_i \, dxdy \) when the integrand \( f \varphi_i \) is complicated

### 7 A NEW APPROACH TO MESH GENERATION

The first step in implementing finite element method is to generate a mesh. In a recent work, the author and his co-workers have proposed a new approach to mesh generation which can discretize 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 contains 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 does not refer to how the shapes of different elements relate to each other. So some elements can be large while others might be very small. There is a second type of requirement on the shape of elements. This requirement says that the 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 papers[-], the explicit finite element integration scheme is presented by using the isoparametric transformation over the 4 node, 8 node and 9 node linear convex quadrilateral elements which is applied to torsion of a square shaft, on considering symmetry of the problem domain, mesh generation for 1/8 of the cross section which is a triangle was discretized into an all quadrilateral mesh. In this paper we consider applications to Poisson boundary values for nonconstant functions over 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 differential equations(PDE). In the last few decades, 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 one 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 barycentre of the triangular element. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain, thus fully discretizes the given convex polygonal domain into all quadrilaterals, thus propagating 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 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 \(\Delta/n^2\) where \(\Delta\) is the area of a linear arbitrary triangle with vertices \((x_i, y_i), \ i = 1, 2, 3\) in the Cartesian space.
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 \( v = \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n} \) bounded on the right by the line \( u + v = 1 \). Thus the entire triangle is covered by \((n+1)(n+2)/2\) nodes. This is shown in the \( rr \) matrix of size \((n+1) \times (n+1)\), only nonzero entries of this matrix refer to the nodes of the triangles.

\[
\begin{bmatrix}
1, & 4, & 5, \ldots, & (n+1), \ldots, & (n+2) \\
3n, & (3n+1), \ldots, & (3n+n), \ldots, & (3n+(n+2)) \\
3n-1, & 3n+(n-1), \ldots, & 3n+(n-2), \ldots, & (3n+(n+3)) \\
3n-(n-3), \ldots, & (3n+(n+1)), \ldots, & 3n+(n+2), \ldots, & (3n+(n+3)) \\
3n-(n-2), \ldots, & (3n+(n+1)), \ldots, & 3n+(n+2), \ldots, & (3n+(n+3)) \\
3 & 0 & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0
\end{bmatrix}
\]

\[(58)\]

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 \( l_{12}, \ l_{23}, \ \text{and} \ l_{31} \). By
dividing the sides $l_{12}$, $l_{23}$, $l_{31}$ into $n = 2m$ divisions (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

![Fig 7(a). Division of an arbitrary triangle into three quadrilaterals](image)

![Fig 7(b). Division of a standard triangle into three quadrilaterals](image)

Four Divisions of Each side of an Arbitrary Triangle

![Fig 8a. Division of an arbitrary triangle into 4 six node triangles](image)

![Fig 8b. Division of a standard triangle into 4 right isosceles triangle](image)

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 $rr(i; i + 2, j; j + 2)$ are nonzero then two six node triangles can be
formed. If $\mathbf{rr}_{(i+1, j+2)} = \mathbf{rr}_{(i+2, j+1)} = 0$ then one six node triangle can be formed. If the submatrices $\mathbf{rr}_{(i, i+2, j; j+2)}$ is a $(3 \times 3)$ zero matrix, we cannot form the six node triangles. We now explain the creation of the six node triangles using the $\mathbf{rr}$ matrix of eqn.(58). We can form six node triangles by using node points of three consecutive rows and columns of $\mathbf{rr}$ 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 $\mathbf{rr}$ sub matrix.

**Formation of six node triangles using sub matrix $\mathbf{rr}$**

![Figure 9](image)

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

$$(e_1) < \mathbf{rr}_{(i, j)}, \mathbf{rr}_{(i, l+2)}, \mathbf{rr}_{(i+2, j)}, \mathbf{rr}_{(i, j+1)}, \mathbf{rr}_{(i+1, j+1)}, \mathbf{rr}_{(i+1, j)} >$$

$$(e_2) < \mathbf{rr}_{(i+2, j+2)}, \mathbf{rr}_{(i+2, j)}, \mathbf{rr}_{(i, j+2)}, \mathbf{rr}_{(i+2, j+1)}, \mathbf{rr}_{(i+1, j+1)}, \mathbf{rr}_{i+1, j+2} >$$

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

If the elements of sub matrix $(\mathbf{rr}_{(k, l)}, k = i, i+1, i+2, l = j, j+1, j+2)$ 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

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

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

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

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

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

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

$Q_{3n_1} < c_2, \mathbf{rr}_{(i+1, j+1)}, \mathbf{rr}_{(i, j+2)}, \mathbf{rr}_{(i+1, j+2)} >$

--- (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 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).

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 of 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 12-node Serendipity and 16-node Lagrange special quadrilateral elements by adding two nodes at the trisectional points on each side and also at the interior of 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[27-32 ].We include here some meshes of all 12-node and 16-node special quadrilaterals at initial stagtes of mesh generation which is self explainatory.

Example 1: right isosceles triangle

\[
\begin{align*}
&x = (0; 1/2; 1/2) \\
y &= (0; 0; 1/2)
\end{align*}
\]

We use this mesh to solve torsion of a square cross section, due to symmetry considerations mesh generation over the above domain is sufficient. This is a case of Poisson equation with constant right hand side(= -2). Our main aim here is to compute torsional constant and Prandtl stress function values for the given domain.

![Initial mesh of right isosceles triangle(12-node quadrilaterals)](image-url)
**Example 2:** equilateral triangle, each side = $2\sqrt{3}$

$x = \left(-\sqrt{3}, \sqrt{3}, 0\right)$

$y = \left(-1, -1, 2\right)$

We use this mesh to solve torsion of an arbitrary triangular cross section. This is a case of Poisson equation with constant right hand side ($= -1$). Our aims are to compute torsional constant and contour lines of Prandtl stress function.
Fig. 12a Initial mesh of equilateral triangle, each side = 2*sqrt(3) (12-node quadrilaterals)

Example 3: a square domain with eight triangles (9-boundary nodes)
We use this mesh to solve torsion of a square cross section. We would like to draw contour lines of Prandtl stress function over the entire domain. This is a case of Poisson equation with constant right hand side (= -2).

Example 4: pentagonal domain with seven triangles (8-boundary nodes)
We use this mesh to solve Poisson equation with a nonconstant smooth function on right hand side, with a known analytical solution.

\[
x = \begin{bmatrix} 1/2; 1/2; 1; 1/2; 0; 0 \end{bmatrix} \%
\]
\[
y = \begin{bmatrix} 1/2; 0; 0; 1/2; 1; 1/2 \end{bmatrix} \%
\]

Fig 14a: Initial Mesh for a pentagonal domain seven triangles-12 noded quadrilaterals
Fig 14b: Initial Mesh for a pentagonal domain seven triangles-16 noded quadrilaterals

Example 5: a square domain with eight triangles (9-nodes)

\[ x = \left( 0; \ 1/2; 1/2; 0; \ 0; -1/2; -1/2; -1/2 \right) \]
\[ y = \left( 0; -1/2; -1/2; 1/2; 1/2; 1/2; 0; -1/2 \right) \]

We use this mesh to solve torsion of a square cross section. We would like to draw contour lines of Prandtl stress function over the entire domain. This is a case of Poisson equation with a constant right hand side (=-2)

Fig 15a: Initial Mesh for a unit square domain eight triangles-12 noded quadrilaterals

Fig 15b: Initial Mesh for a unit square domain with eight triangles-16 noded quadrilaterals

7.5 Application Examples
7.5.1 Mesh Generation Over a Standard Triangle & a Square cross sections

Examples 1 & 5

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:

\[-\Delta u = f, \quad x \in \Omega \subset \mathbb{R}^2 \]
\[u = g, \quad x \in \partial \Omega \]

(1)

Where \( \Omega \) is a triangular or polygonal domain and \( \Delta \) is the 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:

\[\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + 2G\theta = 0 \quad \text{in} \ R \]

(62a)

\[\phi = 0 \quad \text{on} \ \partial R, \text{the boundary of} \ R \]

(62b)

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

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

(62c)

We take \( R \) as the 9-node special quadrilateral meshes described in Examples 1 & 5 in previous section.

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 torsional constant which are expressed in terms of infinite series. This triangular domain is a right isosceles triangle and it was discretised by 8-noded special linear convex quadrilaterals of serendipity family. We have considered this problem again and illustrated the application of 9-node quadrilateral of Lagrange family.

7.5.2 Mesh Generation Over an Arbitrary Triangular Domain

Example 2

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:

\[\nabla^2 u = -1, \quad x \in \Omega \subset \mathbb{R}^2 \]

(9)

\[u = 0, \quad x \in \partial \Omega \]

(10)

Where \( \Omega \) is a regular triangular or polygonal domain and \( \nabla^2 \) is the standard Laplace operator

Example 2

In this example, we would like to consider the linear elastic torsion of an equilateral triangle which is inscribed in a circle of unit radius. In our recent we considered torsion of an equilateral triangular cross section and it was discretised by 8-noded special linear convex quadrilaterals of serendipity family. Now we would like to illustrate the St. Venant Torsion problem for an arbitrary triangular cross section by using 9-node special linear convex quadrilaterals of Lagrange family.
7.5.3 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 9 noded convex quadrilaterals of Lagrange family for the complex domains such as those in [27-33]. We use the theory and procedure developed in sections 7.2, 7.3 and 7.4 for this purpose.

Example 3(with nonconstant smooth function as right hand side of Poisson equation)

\[-\Delta u = 2\pi^2 \sin(\pi x) \sin(\pi y), (x, y) \in \Omega \subset \mathbb{R}^2\]

\[u(x, 0) = 0, \text{ on } y = 0, 0 \leq x \leq 1\]

\[u(x, 1) = 0, \text{ on } y = 1, 0 \leq x \leq 1,\]

\[u(1, y) = 0, \text{ on } x = 1, 0 \leq y \leq 1/2,\]

\[u(x, y) = \sin(\pi x) \sin(\pi y), \text{ on the line } x = 1 - 0.5t, y = 0.5 + 0.5t, 0 \leq t \leq 1\] .............................(62)

Where \(\Delta\) is a standard Laplace operator and \(\Omega\) 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 4(with nonconstant smooth function as right hand side of Poisson equation)

\[-\Delta u = 2\pi^2 \sin(\pi x) \sin(\pi y), (x, y) \in \mathbb{R}^2\]

\[u = 0, \text{ on the boundary } \partial \Omega\] .................................(63)

Where \(\Delta\) is a standard Laplace operator and \(\Omega\) is a square domain \([0, 1]^2\).

The following MATLAB codes are written for the purpose of developing automesh generation techniques and explicit integration methods for 12-node Serendipity and 16-node Lagrange family elements. They can be developed by referring our previous works [27-33]

[1]quadrilateral_mesh_over Arbitrary_triangle_q12automeshgen.m
[2]quadrilateral_mesh_over Arbitrary_triangle_q16automeshgen.m
[3]quadrilateral_mesh4MOINEX_q12.m
[4]quadrilateral_mesh4MOINEX_q16LG.m
[5]D2LaplaceEquationQ12Ex3automeshgenNewContour.m√
[6]D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour.m√
[7]D2LaplaceEquationQ16Ex3automeshgenNewContour.m√
[8]D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour.m√
[9]polygonal_domain_coordinates_3rd_orderLG.m
[10]polygonal_domain_coordinates_3rd_order.m
[11]coordinate_special_quadrilaterals_in_stdtriangle_3rd_orderLAGR.m
[12]coordinate_special_quadrilaterals_in_stdtriangle_3rd_order.m
[13]integral_valuesof_localderivative_products.m
[14]nodaladdresses_special_convex_quadrilaterals_trial_3rd_order.m
[15]generate_area_coordinate_over_the_standard_triangle.m
[16]nodaladdresses_special_convex_quadrilaterals_trial_3rd_orderLG.m
[17]D2PoissonEquationQ12MoinEx_MeshgridContourNew.m√
[18]D2PoissonEquationQ16MoinEx_MeshgridContourNew.m√
[19]newtonmethod4spquadrileteral.m
[20]parameqnspqd.m
[21]paramdetJspqd.m
[22]paraminvJspqd.m
[23]glsampleptsweights.m

In the above list serial no.s 5,6,7,8,17,18 are marked as √ and they call the program integral_valuesof_localderivative_products.m at serial no.13
The programs at serial no.s 19,20,21,22 are called in programs at serial no.s 17 18 to solve bilinear equations in two variates and they are required for interpolating physical solutions at mesh grid points to draw the contour lines.

We are also appending these MATLAB programs towards end of this paper.

### 8.0 CONCLUSIONS

This paper presents the explicit integration schemes for a unique(special) linear convex 12- node and 16-node quadrilaterals of Serendipity and Lagrange family elements 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 9- 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 discretise this standard right isosceles triangle in (u, v) space into three unique linear convex 9- node quadrilaterals. We have shown by proving a lemma that any unique linear convex 9-node quadrilateral in (x, y) space can be mapped into one of the unique 9-node quadrilaterals in (u, v) space. We have then mapped these linear convex 9- 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 9- node quadrilaterals in the global (x,y) space interior to the arbitrary triangle and the linear convex 9- 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 $S_{i,j}^{l,e}$ in global (x, y) space can be expressed as a matrix triple product $P \ast (K^{l,e}) \ast P^T \ast (2 * area of the arbitrary triangle in (x, y) space )$, in which $P$ is a geometric properties matrix and $K^{l,e}$ is the product of global derivative integrals in (u, v) space, $i,j = 1(1)12$ and $i,j = 1(1)16$. We have shown that the explicit integration of the global derivative products in (u, v) space over the unique 12- node and 16-node quadrilaterals 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, engineering, medical, business and social sciences. The well known examples are the Laplace and Poisson equations with suitable boundary conditions and the some examples of system of equations are the plane stress, plane strain 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 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 nine 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 with fast convergence and economy for the computational problems.

### REFERENCES:

1. Zienkiewicz O.C, Taylor R.L and Zhu J.Z. Finite Element Method, its basis and fundamentals, Elsievier, (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. 111-21(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)
We now propose to compute the following integrals (see eqns (52a-b)) and they will be listed in Tables 1a-1i

\[
\mathbf{K}^{ij,e} = \iint_{Q} G^{i,j,e}_{u,v} \, du \, dv = \left( \begin{array}{cc}
\iint_{Q} \frac{\partial^{2} y}{\partial u^{2}} \frac{\partial^{2} y}{\partial v^{2}} \, du \, dv \\
\iint_{Q} \frac{\partial^{2} x}{\partial u^{2}} \frac{\partial^{2} y}{\partial v^{2}} \, du \, dv \\
\iint_{Q} \frac{\partial^{2} x}{\partial u^{2}} \frac{\partial^{2} x}{\partial v^{2}} \, du \, dv \\
\iint_{Q} \frac{\partial^{2} y}{\partial u^{2}} \frac{\partial^{2} x}{\partial v^{2}} \, du \, dv
\end{array} \right) = \left( \begin{array}{c}
K_{2i-1,2j-1}^{e} \\
K_{2i,2j-1}^{e} \\
K_{2i-1,2j}^{e} \\
K_{2i,2j}^{e}
\end{array} \right) \quad (52a-b)
\]

We map \( Q \) into a 2-square \( \mathcal{Q} \) by using the bilinear transformation from \((u,v)\) to \((\xi,\eta)\).

This gives (see eqns (18-20))

\[
\frac{\partial y}{\partial \xi} = \frac{1}{\eta} \left( \begin{array}{c}
\frac{\partial y}{\partial u} \\
\frac{\partial y}{\partial v}
\end{array} \right) \quad \frac{\partial x}{\partial \eta} = \frac{1}{\xi} \left( \begin{array}{c}
\frac{\partial x}{\partial u} \\
\frac{\partial x}{\partial v}
\end{array} \right)
\]

Let us replace the Greek letters \( \xi, \eta \) by English letters \( r, s \) for computing the integrals by using MATLAB programming. With this assumption, we denote the entries of submatrix \( \mathbf{K}^{ij,e} = \int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{juvrs} , (i,j=1(1)9) \)

and we have from eqn (52a-b):

\[
\int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{juvrs} = \begin{pmatrix}
\int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{11}^{juvrs} & \int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{12}^{juvrs} \\
\int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{21}^{juvrs} & \int_{\mathcal{Q}} \mathbf{d} \mathbf{n}^{22}^{juvrs}
\end{pmatrix}
\]
\begin{equation}
K_{2|1-1|2j-1} = \iint_Q \frac{\alpha^T_n \alpha^T_k}{\alpha^T_n \alpha^T_k} \; dudv = \int \text{idnjdjuvrs}(1,1), \quad K_{2|1-1|2j} = \iint_Q \frac{\alpha^T_n \alpha^T_k}{\alpha^T_n \alpha^T_k} \; dudv = \int \text{idnjdjuvrs}(1,2),
\end{equation}

\begin{equation}
K_{2|1-2j-1} = \iint_Q \frac{\alpha^T_n \alpha^T_k}{\alpha^T_n \alpha^T_k} \; dudv = \int \text{idnjdjuvrs}(2,1), \quad K_{2|1-2j} = \iint_Q \frac{\alpha^T_n \alpha^T_k}{\alpha^T_n \alpha^T_k} \; dudv = \int \text{idnjdjuvrs}(2,2)
\end{equation}

Tables of integral values for cubic order Serendipity and Lagrange elements

\begin{equation}
\text{idnjdjuvrs} = \begin{bmatrix}
\text{idnjdjuvrs}(1,1) & \text{idnjdjuvrs}(1,2) \\
\text{idnjdjuvrs}(2,1) & \text{idnjdjuvrs}(2,2)
\end{bmatrix}
\end{equation}

(I) Cubic order Serendipity Elements

Table 1a

\begin{equation}
\text{idnjdjuvrs}, (i=1, j=1(12))
\end{equation}

| \text{idnjdjuvrs} |
|-------------------|
| (1.8785274341641471428122496546, 1.64496677534890315539341209663, 1.885274341641471428122496546) |
| (0.167064190759899129234164489, 0.491299057543551745397431605) |
| (0.2439263805808402079811424029, 0.254564190759899129234164489) |
| (0.06514372856901812926417615089, -0.08063499948054168581340003167, 0.06514372856901812926417615089) |
| (0.4941299057543551745397431605, 0.167064190759899129234164489) |
| (0.2439263805808402079811424029, 0.254564190759899129234164489, 0.06514372856901812926417615089) |
| (0.377654219331636862052401790571, -0.078857287231191145226589121413) |
| (0.22114271268088574773140878857, 0.04710246204597543446965088233) |
| (0.3123680236442004878716433939097, -0.622591925470292070597183972271, -0.3236808236442004878716433939097) |
| (0.22114271268088574773140878857, 0.04710246204597543446965088233) |
| (0.223345682608113424452672757929, 0.415458052185700603634670459957) |
| (0.415458052185700603634670459957, 0.32179593823083588462709750703) |
| (0.32179593823083588462709750703, 0.415458052185700603634670459957) |
| (0.223345682608113424452672757929, 0.415458052185700603634670459957, 0.32179593823083588462709750703) |

Table 1b

\begin{equation}
\text{idnjdjuvrs}, (i=2, j=1(12))
\end{equation}

| \text{idnjdjuvrs} |
|-------------------|
| (0.254564190759899129234164489, 0.491299057543551745397431605) |
| (0.92724744175233410562280795189, -0.1548134165381630251639339579) |
| (-0.1548134165381630251639339579, 0.794715939250703449364685749) |
| (0.512461142342538462495731239596, 0.121011592735232346794931827124) |
| (0.033511592735232346794931827124, 0.15660585205946358641731576026) |
| (0.197040555450002326957831462744, 0.16985399992712677743504195136) |
| (0.0607231647944518067281174443, 0.0676185160002392640247802966) |
| (1.895073448163238931908970695871, -0.32947059876161316868575194333) |
| (0.38302940123838683131424805667, -0.0610174037597342509867019791) |
| (0.223345682608113424452672757929, 0.415458052185700603634670459957) |
| (0.415458052185700603634670459957, 0.32179593823083588462709750703) |
| (0.32179593823083588462709750703, 0.415458052185700603634670459957) |
| (0.32179593823083588462709750703, 0.415458052185700603634670459957, 0.32179593823083588462709750703) |

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482

Page 23364
int Jdn2dn8uvrs = (0.6602878538507373347942390579, 0.132773165331780684038772441541)
    [ 0.432773165331780684038772441541, 0.36542103383905958928044692959]  
int Jdn2dn9uvrs = [-0.82013797775086387086811693089, 0.102283035271046354061620393186]  
    [ 0.102283035271046354061620393186, 0.03640146972555684648076766768]  
int Jdn2dn10uvrs = (0.144533454915567441376932857, -0.3273205972269859324846490286288)  
    [-0.3273205972269859324846490286288, -0.107276020518178565447216676707]  
int Jdn2dn11uvrs = [-0.07898220006398491703637636243, -0.21219256084093252971692223465371]  
    [-0.21219256084093252971692223465371, 0.062634785048803624971612552731]  
int Jdn2dn12uvrs = [-0.09666849492244393286444322229, -0.006161011926712676752388618229]  
    [-0.006161011926712676752388618229, -0.661910533221511128478785205137]  

Table 1c
int Jdnidnjuvrs,(i=3,j=1(12)

Table 1d
int Jdnidnjuvrs,(i=4,j=1(12)
\[-0.32947059876163166868575194333, -1.89507344816323898310987069886\]  
\[\text{intJdn4n1uvrs} = (0.06761851600002392642047802966, 0.06072316479445181067281174443, 0.36072316479445181067281174443, 0.737450284572716169349341246171)\]

| Table-1e |
|--------------------------|
| \text{intJdn}i,juvrs, (i,j)=(1,1) |
| \[
\text{intJdn5n1uvrs} = [-1.91709736541803286033938722609, -1.2133289182132767284207844866, -0.5073891821327684207844866, -0.489515939507034439646857493]  
\| [0.737450284572716169349341246171, 0.06761851600002392642047802966, 0.36072316479445181067281174443, 0.737450284572716169349341246171]  
\| [0.42220907054400735850896000532, 0.338079890317619378517715742657, 0.303879890317619378517715742657, 0.1290238905079389835126431443]  
\| [-0.661910533221541128478752051537, -0.00616101192671266752388618229, -0.00616101192671266752388618229, -0.00616101192671266752388618229]  
\| [4.2002742905147908121147647017, 1.3438679189778808452404427751, 0.831814504364695263991287102]  
\]  
| \text{Table-1f} |
| \text{intJdn}i,juvrs, (i,j)=(6,1) (1,1) |
| \[
\text{intJdn6n1uvrs} = [0.37765421933316862052501790571, 0.22114271276808854773140878557, 0.070872827391145226859121431, 0.0710422604259754343466508283]  
\| [-1.89507344816323898310987069857, 0.3830294012388368831342480567]  
\| [-0.3294705987616131868575194333, -0.0610174037597324509867071996]  
\| [0.789711352194183953055704690412, -0.271075497057497523507647905621, -0.16559436501592339972922160759]  
\| [0.062364758048880362497125527371, -0.212192568043925297169223465371, -0.078892200036949170637636243]  
\| [-2.9970249445101024283266761099, -1.14478719196702554874616021, -0.2653061806266064120644443]  
\]  

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482 Page 23366
### Table 1g

| i, j | 1(12) |
|------|-------|
|      |       |

### Table 1h

| i, j | 1(12) |
|------|-------|
|      |       |

### Table 1i

| i, j | 1(12) |
|------|-------|
|      |       |

### Table 1j

| i, j | 1(12) |
|------|-------|
|      |       |

---

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482
Table 1j

| i, j | i=j=1 (1/12) |
|------|---------------|

Table 1k

| i, j | i=j=1 (1/12) |
|------|---------------|

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 2329-2348 Page 23368
Table II

\[
\begin{array}{cccc}
\text{intJdn1dnuvs, i=12, j=1 (1/12)} & \\
\hline
\text{intJdn1dn1uvs} & \left[ -0.489515939250703449364685749, -0.50073891821327672848247048486 \right] & \\
\text{intJdn1dn2uvs} & \left[ -0.21323891821327672848247048486, -1.91709736541803286033938722609 \right] & \\
\text{intJdn1dn3uvs} & \left[ -0.0966864949224433928644322229, -0.066161011926712676752388618229 \right] & \\
\text{intJdn1dn4uvs} & \left[ 0.12092398059739983152641344, 0.33807989031769717847572030157 \right] & \\
\text{intJdn1dn5uvs} & \left[ 0.33807989031769717847572030157, -0.66191053321541128478752035173 \right] & \\
\text{intJdn1dn6uvs} & \left[ 0.220309553134273520347347647, 0.16013975665860229629946992 \right] & \\
\text{intJdn1dn7uvs} & \left[ 0.195931102328325175250822, 0.0938905178333991676309618371 \right] & \\
\text{intJdn1dn8uvs} & \left[ 0.221278776266355628659448848351, 0.46814346049334316998992911 \right] & \\
\text{intJdn1dn9uvs} & \left[ 0.26581082595366911475364275533, -0.81719220580616212919469695233 \right] & \\
\text{intJdn1dn10uvs} & \left[ -0.81719220580616212919469695233, -1.54043101424598444521561548333 \right] & \\
\text{intJdn1dn11uvs} & \left[ -0.068297309371635561683531307714, -0.1539996649137517843979403152 \right] & \\
\text{intJdn1dn12uvs} & \left[ -0.1539996649137517843979403152, -0.3686025665386866032326594038152 \right] & \\
\text{intJdn1dn13uvs} & \left[ -0.30285329909637720797735289779, -0.13926553656621939017193095331 \right] & \\
\text{intJdn1dn14uvs} & \left[ -0.13926553656621939017193095331, -0.229689408692304476227096154 \right] & \\
\text{intJdn1dn15uvs} & \left[ -0.26536108626066104206434444, -1.14748317191958784045204472751 \right] & \\
\text{intJdn1dn16uvs} & \left[ -0.1348971916702554874616021, -2.9970249445101242383266710989 \right] & \\
\end{array}
\]

(II) Cubic order Lagrange Elements
\[ -0.104911199388141007005795705, -0.09224518884085198620300047040 \]
\[ 0.03351827978973348546689505, 0.039281153486431255496910516 \]
\[ 0.038921153486431255496910516, 0.03187907114531758725070008895 \]
\[ 0.038917907114531758725070008895, 0.039281153486431255496910515 \]
\[ 0.033518279789733485466895045, 0.03187907114531758725070008895 \]
\[ -0.009224518884085198620300047035, -0.104911199388141007005795705 \]
\[ -0.094916761402855925624077739, -0.00863032749922837615196918435 \]
\[ 0.050281043946617803788049234, -0.045951687880613681966667929 \]
\[ 0.01454830121388613768196667829, -0.02106669083300601536386540822 \]

| Table 2b |
|------------------|
| intJdnidnjuvs, i=2, j=1 (1) |

\[ intJdn1dn8uvrs= \frac{-0.02688822664294743129188827125}{0.01050544795498581251541917445}, \frac{0.032212211804107707531538537271}{0.015304657482688411066233853634} \]
\[ intJdn2dn2uvrs= \frac{0.211971597991981069039609993}{0.0350281043946617803788049234}, \frac{0.050281043946617803788049234}{0.04154830121388613681966667929} \]
\[ intJdn3dn3uvrs= \frac{-0.0459516987868113623618303332171}{0.0052081043946617803788049234}, \frac{-0.0459516987868113623618303332171}{0.0052081043946617803788049234} \]
\[ intJdn4dn4uvrs= \frac{-0.00045967731769692422889922627}{0.0052081043946617803788049234}, \frac{-0.00045967731769692422889922627}{0.0052081043946617803788049234} \]
\[ intJdn5dn5uvrs= \frac{0.133156485729803324632359097}{0.14935188545747747523525263}, \frac{0.133156485729803324632359097}{0.14935188545747747523525263} \]
\[ intJdn6dn6uvrs= \frac{0.0080406033653616406467618071191}{0.028346873138031592670232799}, \frac{0.0080406033653616406467618071191}{0.028346873138031592670232799} \]
\[ intJdn7dn7uvrs= \frac{-0.4508478537067492781247859798}{0.0004598673716093138997422889922627}, \frac{-0.4508478537067492781247859798}{0.0004598673716093138997422889922627} \]
\[ intJdn8dn8uvrs= \frac{0.114379626667666666772048}{0.1706455126237189438437270694}, \frac{0.114379626667666666772048}{0.1706455126237189438437270694} \]
\[ intJdn9dn9uvrs= \frac{0.035328033479583720419581655}{0.057501500322407089943211709137}, \frac{0.035328033479583720419581655}{0.057501500322407089943211709137} \]
\[ intJdn10dn10uvrs= \frac{0.00104003817923470925719506}{0.02476171979987754376976145649}, \frac{0.00104003817923470925719506}{0.02476171979987754376976145649} \]

| Table 2c |
|------------------|
| intJdnidnjuvs, i=3, j=1 (1) |

\[ intJdn3dn1uvrs= \frac{-0.00563454417914164990072595}{0.045762237038458441917844749441}, \frac{-0.00563454417914164990072595}{0.045762237038458441917844749441} \]
\[ intJdn3dn2uvrs= \frac{0.0080332805059661177025893}{0.0086300080830678135931299152}, \frac{0.0080332805059661177025893}{0.0086300080830678135931299152} \]
H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482

Table 2-d

| i,j | 1,1 | 1,2 | 1,3 | 1,4 | 1,5 | 1,6 |
|-----|-----|-----|-----|-----|-----|-----|
| 1   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |
| 2   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |
| 3   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |
| 4   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |
| 5   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |
| 6   | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       | 0.93125256923490725346584258       |

Page 23371
**Table 2c**  
intJdnidnuvr, i=5, j=1 (1/16)

|                |                |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|----------------|
| intJdn5nuvr   | \(-0.41263305411747967464094697, -0.44355447401522018456977467\) |  | \[0.26894525987447691854302253, 0.157871624541338704794557829\] |  |  |
| intJdn6nuvr   | \(0.15331648572908233462395097, -0.150648114542252247644773701\) |  | \[0.14938185547774574233352263, -0.03560700314514977048729502\] |  |  |
| intJdn7nuvr   | \(-0.0312258974888804915879185904, 0.03066254874074171107697545881\) |  | \[0.03066254874074171107697545881, 0.01203062469906479395309084256\] |  |  |
| intJdn8nuvr   | \(-0.0207050570857146813452057, 0.0495487663646607134721033842\) |  | \[0.0495487663646607134721033842, 0.022034666228595693400301699\] |  |  |
| intJdn9nuvr   | \[0.31755878728317232766908314, 0.902875989441502362683040391\] |  | \[0.31755878728317232766908314, 0.902875989441502362683040391\] |  |  |
| intJdn10nuvr  | \(-0.78668808464461673459831513779, 0.27477449405670333406360202\] |  | \[-0.78668808464461673459831513779, 0.27477449405670333406360202\] |  |  |

---

**Table 2f**  
intJdnidnuvr, i=6, j=1 (1/16)

|                |                |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|----------------|
| intJdn6nuvr   | \[0.1172503945734017479967464094697, 0.157047902935332961077945381\] |  | \[-0.14259207947664967038925468, -0.070027089525016846364967113\] |  |  |
| intJdn7nuvr   | \[-0.48406393561440646761807191, 0.38995920386652252899208329149\] |  | \[-0.3225407916324774100791670851, 0.1378297018364707164848354907\] |  |  |
| intJdn8nuvr   | \[-0.09563153302204266898097543501, 0.0701114223525693909913070362\] |  | \[-0.09563153302204266898097543501, 0.0701114223525693909913070362\] |  |  |
| intJdn9nuvr   | \[-0.002353045794885865623325083, -0.0239609914551285818348361\] |  | \[-0.002353045794885865623325083, -0.0239609914551285818348361\] |  |  |
| intJdn10nuvr  | \[0.78668808464461673459831513779, -0.7377259045670334063602205\] |  | \[0.78668808464461673459831513779, -0.7377259045670334063602205\] |  |  |
| intJdn6nuvr   | \[0.02744744905670334063602205, 0.22206247089136121359932407811\] |  | \[0.02744744905670334063602205, 0.22206247089136121359932407811\] |  |  |
| intJdn7nuvr   | \[0.3773255904392666693639797954, -0.2220624808136121359932407811\] |  | \[0.3773255904392666693639797954, -0.2220624808136121359932407811\] |  |  |
| intJdn8nuvr   | \[0.0605393794449485389242123256697487, 0.27265979766225212356697487\] |  | \[0.0605393794449485389242123256697487, 0.27265979766225212356697487\] |  |  |
| intJdn9nuvr   | \[0.08921165458163810696876669, 0.4271888849479047070117399236\] |  | \[0.08921165458163810696876669, 0.4271888849479047070117399236\] |  |  |
| intJdn10nuvr  | \[0.4721888849799078070173992386, 0.08921165458163810696876669\] |  | \[0.4721888849799078070173992386, 0.08921165458163810696876669\] |  |  |
\[
\text{int Jdn9dn10uvrs} = \begin{bmatrix}
-0.222366721554648838029403792,
0.489907940658406069046527082,
0.222366721554648838029403792
\end{bmatrix}
\]
\[
\text{int Jdn9dn10uvrs} = \begin{bmatrix}
0.0530267352097846205377516688,
-0.204622656214462989777149553,
0.46925490922703366956712246298752
\end{bmatrix}
\]
\[
\text{int Jdn9dn10uvrs} = \begin{bmatrix}
-0.115436426092527888318297208026,
-0.0079362521672496957651556136,
0.123880420711948724330823519
\end{bmatrix}
\]
\[
\text{int Jdn9dn10uvrs} = \begin{bmatrix}
-0.0283674858742885011995552228,
-0.113465061045299667712243619,
-0.1349507714997270711139867165
\end{bmatrix}
\]
\[
\text{int Jdn9dn10uvrs} = \begin{bmatrix}
0.060332905916541134334696078,
0.331278731126519730506371948,
0.1788501510256261210302257368
\end{bmatrix}
\]
\[
\text{int Jdn8dn10uvrs} = \begin{bmatrix}
-0.051995759267779853026220703,
-0.79506545935637821598361452,
-0.34336222419197932020630238335
\end{bmatrix}
\]
\[
\text{int Jdn8dn15uvrs} = \begin{bmatrix}
-1.944186194720450713243325388,
-0.0266497211117548073900616686,
0.550843979769193341763278679
\end{bmatrix}
\]
\[
\text{int Jdn8dn16uvrs} = \begin{bmatrix}
0.53139140501145964059996936,
0.368439503272838247652795296,
-0.154508813774518558307843735
\end{bmatrix}
\]

| Table-2i |
|----------|
| \( i,j = 1 \) (1/16) |

| Table-2j |
|----------|
| \( i,j = 1 \) (1/16) |
| Issue 2l | 1 January 2018 |
|--------------------------|
| Page No. 23329 |

**Table 2-2**

| intJdn1u1vs, i=11,j=1 (1)16 |
|-----------------------------|
|--------------------------------|
| [0.106679112592585571455322301, 0.0060593371924777808811068095] |
| [0.165709205230530363910745318, 0.117250397174796746049671] |
| [0.0080493488723694790441230002, -0.2396091455128535818643861] |
| [0.2396091455128535818643861, 0.2353045794858365262350827] |
| [0.02291948532922911508780784477, -0.7071142235256930991307632] |
| [0.07071142235256930991307632, -0.095631533302240689097543501] |
| [0.137829701836470761488534907, -0.322504791632477417079168051] |
| [0.38995208367522389280232919, -0.484060336536150646671807119] |
| [0.26970845869880871379531078, -0.1996663903410992890964084] |
| [0.1956663903410992890964084, -0.0789735824975260868273981] |
| [0.02752791093307208248725575, 0.08926809881605201657837726] |
| [0.08926809881605201657837726, 0.02757791093307208248725575] |
| [0.01270592352690190434798767, 0.02481901837653599808136231974] |
| [0.024337706121653439544353543, -0.62901590438554706766726590196] |
| [0.62901590438554706766726590196, -0.857977568743733702181109962] |

**Table 2-1**

| intJdn1u2vs, i=12,j=1 (1)16 |
|-----------------------------|
|--------------------------------|
| [1.14198144936888465826082049, 0.6020653753184351590313159] |
| [0.6020653753184351590313159, 0.1322897028037908049925685] |
| [0.22202648081631539932407811, 0.27477744905607733340360622046] |
| [0.17725295943926666936977954, -0.786688084461467345983151779] |
| [0.50202184956697981678232119, 0.82742377912994363911304061] |
| [0.82742377912994363911304061, 0.10736670368679853419887039] |
| [0.189527808511794839868468484, -0.3273240382682450655130436] |
| [0.3273240382682450655130436, -0.0206290221988175390182549] |
| [0.44988120080951953153671278, 0.10610467066746834133287489] |
| [0.10610467066746834133287489, -0.21623144244466091271015883] |
| [1.50175278777672092105195536, -0.5105994609520005197845965] |
| [0.5105994609520005197845965, 0.60108535705643549568526] |

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482
\begin{table}
\centering
\begin{tabular}{cccc}
\hline
\textbf{Column 1} & \textbf{Column 2} & \textbf{Column 3} & \textbf{Column 4} \\
\hline
Value 1 & Value 2 & Value 3 & Value 4 \\
\hline
Value 5 & Value 6 & Value 7 & Value 8 \\
\end{tabular}
\caption{Table Caption}
\end{table}

\[
\text{intJdn13dn15uvrs} = \{-0.18002409287106955030651138, -0.949948521110901296761134222\} \\
\text{intJdn13dn16uvrs} = \{-0.809622238944438740999396603, -0.87612662642062091794899095697\}
\]

**Table-2n**

| i | j | k |
|---|---|---|
| 14 | 1 | 1 |

\[
\text{intJdn14dn1uvrs} = \{0.33685199022272278640944172519, 0.381671404353293171771605003\} \\
\text{intJdn14dn2uvrs} = \{-0.005654454471971466900072595, 0.4276223703845844197184744941\} \\
\text{intJdn14dn3uvrs} = \{0.136996426878579334021389201, 0.22271314634900971981666572636\} \\
\text{intJdn14dn4uvrs} = \{-0.009846448660450092579084616, 0.863000808036781359312991519\} \\
\text{intJdn14dn5uvrs} = \{-1.195573601957603150563316788, -0.890289639589649219996546217\} \\
\text{intJdn14dn6uvrs} = \{0.60105835750654359495658263, -0.510954690592000551975845965\} \\
\text{intJdn14dn7uvrs} = \{-2.24559114498242587485265165, 0.32502712079217193433836137\} \\
\text{intJdn14dn8uvrs} = \{-0.0519957592677798530262703, -0.795060549935637821598461452\} \\
\text{intJdn14dn9uvrs} = \{-0.15450881374185583078433735, 0.3684539072283824762579296\} \\
\text{intJdn14dn10uvrs} = \{0.045605512128407846407826384, -0.30423328239753321664616866\} \\
\text{intJdn14dn11uvrs} = \{-0.1895278058117439066468435, -0.3273243082666406555153036\} \\
\text{intJdn14dn12uvrs} = \{0.153040155562725165901986887, 0.09126610139249880492224\} \\
\text{intJdn14dn13uvrs} = \{-3.400472479844200126665206259, -0.876126624626291794899095967\} \\
\text{intJdn14dn14uvrs} = \{5.181761920141211756612538031, 1.77075955808797873857081430869\} \\
\text{intJdn14dn15uvrs} = \{-0.8059779198066649293192672, -0.637841602652039432570953931\} \\
\text{intJdn14dn16uvrs} = \{0.877518569223348982633756744, 1.28773972299040771792187534\} \\
\]

**Table-2o**

| i | j | k |
|---|---|---|
| 15 | 1 | 1 |

\[
\text{intJdn15dn1uvrs} = \{-0.122483212211684061698393545, -0.145263095437075394958553495\} \\
\text{intJdn15dn2uvrs} = \{0.00803323805596981177025893, -0.19421697367314354534357175\} \\
\text{intJdn15dn3uvrs} = \{-0.319448664201393584204597479, -0.53496055880256698795985017\} \\
\text{intJdn15dn4uvrs} = \{0.041681314367425998111696919, -0.19421697367371435453457175\} \\
\text{intJdn15dn5uvrs} = \{0.4948797291226663612952535, 0.37936047707194774719460232042\} \\
\text{intJdn15dn6uvrs} = \{-0.26123614244466091271015883, 0.106140670667416834133287489\} \\
\text{intJdn15dn7uvrs} = \{0.5524217314970878606391031, 0.683049144738999567208941635\} \\
\text{intJdn15dn8uvrs} = \{-0.94148169472045071324352388, -0.0266497211117584073090616868\} \\
\text{intJdn15dn9uvrs} = \{0.550843979761993341763278679, -0.026649721111758407309061688\} \\
\text{intJdn15dn10uvrs} = \{-0.293283236703239911368713211, 0.676384914473899657208941635\} \\
\]

Page 23377
intJdn16nuvrs = [0.338459011543472359096086, 0.38167140435239171971771605003, 0.33685199222722864094712752]
intJdn16nuvrs = [0.020826689443892215290932469, 0.0863008080367813593129915199, 0.0941185558099118441713021744, 0.2272134136490979198166572566, 0.136994628785793340213892091, 0.09494852111092196761134222, -0.0180024092871069550306511383]
intJdn16nuvrs = [4.394764323667124663601540761, 1.0238880527664474888171737195, 4.394764323667124663601540761, -2.767994568776444949650936329, -0.637841602652030943257059331, -0.80597791980666492939192672]

Table 2p
intJdnidnuvrs, i=16,j=1 (116)

| i  | j  | uvrs |
|----|----|------|
| 16 | 1  | 0.020826689443892215290932469 |
| 16 | 2  | 0.0863008080367813593129915199 |
| 16 | 3  | 0.0941185558099118441713021744 |
| 16 | 4  | 0.2272134136490979198166572566 |
| 16 | 5  | 0.136994628785793340213892091 |
| 16 | 6  | 0.09494852111092196761134222 |
| 16 | 7  | -0.0180024092871069550306511383 |
| 15 | 1  | 4.394764323667124663601540761 |
| 15 | 2  | 1.0238880527664474888171737195 |
| 15 | 3  | 4.394764323667124663601540761 |
| 15 | 4  | -2.767994568776444949650936329 |
| 15 | 5  | -0.637841602652030943257059331 |

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482


**SOME SAMPLE RESULTS TABLES& FIGURES**

**Table 2a**

| Mesh No | node | nel | nnel | Torisonal constant | maximum absolute error |
|---------|------|-----|------|--------------------|------------------------|
| 1)      | 19   | 3   | 9    | 0.140226296123952 | 0.0011496140422405     |
|         | 25   | 3   | 12   | 0.140100662876437 | 0.001814895145593      |
|         | 37   | 3   | 16   | 0.140564616238274 | 0.000361119966844      |
| 2)      | 61   | 12  | 9    | 0.140549995991093 | 0.0002838382097359      |
|         | 79   | 12  | 12   | 0.140527987183054 | 0.000386966456726       |
|         | 127  | 12  | 16   | 0.140576254134519 | 0.000089194283500       |
| 3)      | 127  | 27  | 9    | 0.140571151900467 | 0.000125431921097136    |
|         | 163  | 27  | 12   | 0.140563898437538 | 0.000188851358012       |
|         | 271  | 27  | 16   | 0.140576864992942 | 0.00004770762286        |
| 4)      | 217  | 48  | 9    | 0.140575044191592 | 7.22896125021954e-005   |
|         | 277  | 48  | 12   | 0.140571630576105 | 0.000115489036847       |
|         | 469  | 48  | 16   | 0.140576967538218 | 0.000020792323046       |
| 5)      | 331  | 75  | 9    | 0.140576171269977 | 4.498537060151e-005     |
|         | 421  | 75  | 12   | 0.140574220427089 | 0.000080483984142       |
|         | 721  | 75  | 16   | 0.140576995539002 | 0.000014277421181       |
| 6)      | 469  | 108 | 9    | 0.140576593774739 | 2.9854615531262e-005    |
|         | 595  | 108 | 12   | 0.140575338318066 | 0.000058263173270       |
|         | 1027 | 108 | 16   | 0.140577005593141 | 0.000011806028713       |
| 7)      | 631  | 147 | 9    | 0.140576781096277 | 2.23280855678916e-005   |
|         | 799  | 147 | 12   | 0.140575906857497 | 0.000045873905107       |
|         | 1387 | 147 | 16   | 0.140577009902267 | 0.000009955551927       |
| 8)      | 817  | 192 | 9    | 0.140576874567466 | 1.8379013205128e-005    |
|         | 1033 | 192 | 12   | 0.140576230947825 | 0.000037905251021       |
|         | 1801 | 192 | 16   | 0.140577011993390 | 0.000008119911696       |
| 9)      | 1027 | 243 | 9    | 0.140576925494304 | 1.85338405955379e-005   |
|         | 1297 | 243 | 12   | 0.140576431755842 | 0.000030811538513       |
|         | 2269 | 243 | 16   | 0.140577013106209 | 0.000006337564194       |
| 10)     | 1261 | 300 | 9    | 0.14057655193951  | 1.38273942157716e-005   |
|         | 1591 | 300 | 12   | 0.140576564296181 | 0.000024665504597       |
|         | 2791 | 300 | 16   | 0.140577013742108 | 0.000004703818292       |

PRESENT PAPER: FEM SOLUTION FOR 12 node and 16 node cubic elements

PREVIOUS PAPER: FEM SOLUTION FOR 9 node quadratic Lagrange elements

Torsion of a square cross section modeled as a right isosceles triangle: R, where R=\{(x,y)|0\leq x, y \leq \frac{1}{2}, 0 \leq x - y \leq 1/2\}

nnode=9, Nine node linear convex quadrilaterals of Lagrange family elements

nnode=12, Twelve node linear convex quadrilaterals of Serendipity family elements

nnode=16, Sixteen node linear convex quadrilaterals of Lagrange family elements

exact solution of torsional constant= 0.140577014955156

nnode=number of nodes in the triangular region R

nel=number of elements in the region R
Table-3a

TORSION OF AN EQUILATERAL TRIANGULAR CROSS SECTION

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482
exact solution of torisonal constant= 3.11769145362398  

PRESENT PAPER: FEM SOLUTION FOR 12 node and 16 node cubic elements

PREVIOUS PAPER: FEM SOLUTION FOR 9 node quadratic Lagrange element

Linear elastic torsion of an equilateral triangle T which is inscribed in a circle of unit radius.

Equilateral triangle T has coordinates for vertices as

\[ x = [-\sqrt{3}, \sqrt{3}, 0] \]
\[ y = [-1, -1, 2] \]

nnode=number of nodes in the triangular region T

nel=number of elements in the region T

| Mesh No | nnode | nel | nnel | Torisonal constant | maximum absolute error |
|---------|-------|-----|------|--------------------|------------------------|
| 1       | 19    | 3   | 9    | 3.10739407578924   | 0.00317246382279579    |
| 25      | 3     | 12  | 3    | 3.032110091189756  | 0.02914323981426       |
| 37      | 3     | 16  | 3    | 3.11769145362398   | 0.000000000000000000   |
| 2       | 61    | 12  | 9    | 3.11704786750931   | 0.000396557978849543   |
| 79      | 12    | 12  | 3.110644194853498  | 0.006021532419355      |
| 3       | 127   | 27  | 9    | 3.11756432550257   | 0.000117498660104104   |
| 163     | 27    | 12  | 3.115753983225545  | 0.002021154323719      |
| 4       | 217   | 48  | 9    | 3.11765122949181   | 4.95697472316092e-005   |
| 277     | 48    | 12  | 3.116839757430294  | 0.001306214893304      |
| 5       | 331   | 75  | 9    | 3.11767497781946   | 2.53797105838616e-005   |
| 421     | 75    | 12  | 3.117217781295634  | 0.000901007607025       |

Example: TORSION OF AN EQUILATERAL TRIANGULAR CROSS SECTION
Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements

(MESH HAS 37 NODES AND 3 ELEMENTS)

Contour level curves for exact solution:
Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements

SUPERPOSITION OF
FEM/EXACT SOLUTIONS

- (red) FEM
- (blue) EXACT

NODES=37
ELEMENTS=3

equilateral triangular cross section using 12-node cubic serendipity quadrilateral elements

MESH NO.=1
number of elements=3
number of nodes=25
Example 5: TORSION OF A SQUARE CROSS SECTION

Mesh of 600 twelve node quadrilateral elements & 3121 nodes
Contour level curves for FEM solution of Twelve Noded Special Quadrilateral Elements

(MESH HAS 3121 NODES AND 600 ELEMENTS)
Mesh of 16 sixteen node quadrilateral elements & 5507 nodes

Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements

(MESH HAS 5507 NODES AND 600 ELEMENTS)
Solution of Poisson Boundary Value Problems Over Polygonal Domains

Meshes and Contour Level Curves for a Pentagonal Domain with 12-noded Quadrilateral Elements

Mesh with 525 twelve noded quadrilateral elements & no. of nodes = 2731
Contour level curves for FEM solution of Twelve Noded Special Quadrilateral Elements

Contour level curves for exact solution: \( \sin(\pi x) \times \sin(\pi y) \)
MESHES AND CONTOUR LEVEL CURVES FOR A SQUARE DOMAIN WITH TWELVE NOODED QUADRILATERAL ELEMENTS

Mesh with 600 twelve noded quadrilateral elements & no. of nodes = 3121
MESHES AND CONTOUR LEVEL CURVES FOR A PENTAGONAL DOMAIN WITH 16-NODED QUADRILATERAL ELEMENTS
Mesh with 525 sixteen noded quadrilateral elements & no. of nodes = 4831
MESHES AND CONTOUR LEVEL CURVES FOR A SQUARE DOMAIN WITH SIXTEEN NOODED QUADRILATERAL ELEMENTS

Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements

(MESH HAS 5521 NODES AND 600 ELEMENTS)
MATLAB PROGRAMS FOR 12-NODE SERENDIPITY AND 16-NODE LAGRANGE ELEMENTS

LIST OF PROGRAMS
[1] quadrilateralmesh_over_arbitrarytriangle_q12automeshgen.m
[2] quadrilateralmesh_over_arbitrarytriangle_q16automeshgen.m
MATLAB CODES

function [x]=quadrilateralmesh_over_arbitrarytriangle_q12automeshgen(mmesh,nmesh,tri)
% quadrilateralmesh_over_arbitrarytriangle_q9automeshgen(1,1,4)
% quadrilateralmesh_over_arbitrarytriangle_q12automeshgen(1,1,4)
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)

end
for mesh=mmesh:nmesh
figure(mesh)

end
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]);
    case 4
      axis tight
      xmin=-2;xmax=2;ymin=-1;ymax=2;
      axis([xmin,xmax,ymin,ymax]);
  end
  figure(ndiv/2)
  plot(xvec,yvec,'r-');% plot element
  %plot(xvec,yvec);% plot element
  hold on;
  %place element number
  midx=mean(xvec(1,1:4));
  midy=mean(yvec(1,1:4));
  if mesh<=2
    text(midx,midy,['\textbf{num2str(i),\textbf{)'));
  end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  switch tri
  case 1
    xlabel('\textbf{axis}')
    ylabel('\textbf{axis}')
    st1='\textbf{one eigth (1/8)square cross section}';
st2= 'using ';
st3='\textbf{12-node cubic serendipity}';
st4='\textbf{quadriateral}';
st5='\textbf{elements}'
title([st1,st2,st3,st4,st5])
  case 2
    xlabel('\textbf{axis}')
    ylabel('\textbf{axis}')
    st1='\textbf{one eigth (1/8)square cross section}';
st2='\textbf{using }';
st3='\textbf{12-node cubic serendipity}';
st4='\textbf{quadriateral}';
st5='\textbf{elements}'
title([st1,st2,st3,st4,st5])
% Mesh Details
% Case 3
xlabel('bfx axis')
ylabel('bfy axis')
st1='bfequilateral triangular cross section';
st2='using';
st3='12-node cubic serendipity';
st4='quadrilateral';
st5='elements'
title([st1,st2,st3,st4,st5])

% Case 4
xlabel('bfx axis')
ylabel('bfy axis')
st1='bfequilateral triangular cross section';
st2='using';
st3='12-node cubic serendipity';
st4='quadriateral';
st5='elements'
title([st1,st2,st3,st4,st5])
end

% Put node numbers
for jj=1:nnode
if mesh<=2
%text(gcoord(jj,1),gcoord(jj,2),['bf.',num2str(jj)]);
else
%text(gcoord(jj,1),gcoord(jj,2),['bf ']);
end
end
hold on
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;
figure(mesh),scatter(x(1,1:nnel),y(1,1:nnel),20,'filled','b')
figure(mesh),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),'MarkerFaceColor','g')
end
end
figure(mesh),scatter(gcoord(:,1),gcoord(:,2),'MarkerFaceColor','g')
axis off
%
%***********************
%**********************

% Switch tri
switch tri
case 1
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 2
text(0.6,0.75,["bfMESH NO.='",num2str(mesh)])
text(0.6,0.70,["bfnumber of elements='",num2str(nel)])
text(0.6,0.65,["bfnumber of nodes='",num2str(nnode)])
case 3
text(0.6,0.75,["bfMESH NO.='",num2str(mesh)])
text(0.6,0.70,["bfnumber of elements='",num2str(nel)])
text(0.6,0.65,["bfnumber of nodes='",num2str(nnode)])
case 4
text(0.6,0.75,["bfMESH NO.='",num2str(mesh)])
text(0.6,0.70,["bfnumber of elements='",num2str(nel)])
text(0.6,0.65,["bfnumber of nodes='",num2str(nnode)])
end

% Put node numbers
for jj=1:nnode
if mesh<=2
%text(gcoord(jj,1),gcoord(jj,2),['bf.',num2str(jj)]);
else
%text(gcoord(jj,1),gcoord(jj,2),['bf ']);
end
end
hold on
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;
figure(mesh),scatter(x(1,1:nnel),y(1,1:nnel),20,'filled','b')
figure(mesh),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),'MarkerFaceColor','g')
end
end
figure(mesh),scatter(gcoord(:,1),gcoord(:,2),'MarkerFaceColor','g')
%axis off
%
%***********************

%**********************
% figure(mesh),scatter(x(1,1:nnel),y(1,1:nnel),15,'filled','g')

end for nmesh the number of meshes

[2] quadrilateralmesh_over_arbitrarytriangle_q16automeshgen.m

function=[quadrilateralmesh_over_arbitrarytriangle_q16automeshgen(mmesh,nmesh,tri)
% quadrilateralmesh_over_arbitrarytriangle_q9automeshgen(1,1,4)
% quadrilateralmesh_over_arbitrarytriangle_q16automeshgen(1,1,4)

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]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_order(ndiv)
% [coord,gcoord]=coordinate_risoscelestriangle00_h0_hh_2ndorder(ndiv);
% [coord,gcoord]=coordinate_arbitrarytriangle_3rdorderLAGR(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]);
case 4
axis tight
xmin=-2;xmax=2;ymin=-1;ymax=2;
axis([xmin,xmax,ymin,ymax]);
end
plot(xvec,yvec,’b-’); % plot element
hold on;
% place element number
midx=mean(xvec(1,1:4));
midy=mean(yvec(1,1:4));
if mesh<=2
    text(midx,midy,’\textbf{\textit{Mesh No.}} ’,num2str(i));
    end
end
% i loop
switch tri
    case 1
    xlabel(’\textbf{x axis}’);
    ylabel(’\textbf{y axis}’);
st1=’\textbf{one eighth (1/8)square cross section ’;
st2=’using ’;
st3=’16-node cubic Lagrange ’;
st4=’quadrilateral ’;
st5=’elements’;
title([st1,st2,st3,st4,st5])
    case 2
    xlabel(’\textbf{x axis}’);
    ylabel(’\textbf{y axis}’);
st1=’\textbf{one eighth (1/8)square cross section ’;
st2=’using ’;
st3=’16-node cubic Lagrange ’;
st4=’quadrilateral ’;
st5=’elements’;
title([st1,st2,st3,st4,st5])
    case 3
    xlabel(’\textbf{x axis}’);
    ylabel(’\textbf{y axis}’);
st1=’\textbf{equilateral triangular cross section ’;
st2=’using ’;
st3=’16-node cubic Lagrange ’;
st4=’quadrilateral ’;
st5=’elements’;
title([st1,st2,st3,st4,st5])
    case 4
    xlabel(’\textbf{x axis}’);
    ylabel(’\textbf{y axis}’);
st1=’\textbf{equilateral triangular cross section ’;
st2=’using ’;
st3=’16-node cubic Lagrange ’;
st4=’quadrilateral ’;
st5=’elements’;
title([st1,st2,st3,st4,st5])
end

switch tri
    case 1
text(0.6,0.8,’\textbf{Mesh No.}=’ ,num2str(mesh))
text(0.6,0.75,’\textbf{number of elements}=’ ,num2str(nel))
% put node numbers
for jj=1:nnode
if mesh<=2
    text(gcoord(jj,1),gcoord(jj,2),num2str(jj));
else
    text(gcoord(jj,1),gcoord(jj,2),num2str(jj));
end
end
hold on
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;
figure(mesh),scatter(x(1,1:nnel-4),y(1,1:nnel-4),20,'filled','r')
figure(mesh),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),20,'filled','b')
end
% for nmesh-the number of meshes
function []=quadrilateral_mesh4MOINEX_q12(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
clc
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_q16LG([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,1,1)
@quadrilateral_mesh4MOINEX_q12([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,1,1)
@quadrilateral_mesh4MOINEX_q12([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nne]=size(nodes);
disp([xlength,ylength,nnode,nne])
@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 tight
switch mesh
  case 1
  axis([0 xlength 0 ylength])
  case 2
  axis([0 xlength 0 ylength])
  case 3
  axis([0 xlength 0 ylength])
  case 4
  axis([-xlength/2 xlength/2 -ylength/2 ylength/2])
end
figure(ndiv/2)
plot(xvec,yvec,'r-'); % plot element
hold on;
% place element number
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
if ndiv<=2
  text(midx+.01,midy-.03,['
',num2str(i),']']);
end
end; % i loop
xlabel('x axis')
ylabel('y axis')
st1='Mesh of ';
st2=num2str(nel);
st3=' twelve node ';
st4=' quadrilateral ';
st5=' elements & '
st6=num2str(nnode);
st7=' nodes'
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),[num2str(jj)]);
  end
end; % i loop
figure(1),scatter(gcoord(:,1),gcoord(:,2),20,'filled','r')
figure(1),scatter(x(1,1:nnel-4),y(1,1:nnel-4),15,'filled','g')
figure(1),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),15,'filled','r')
end % i loop
end

% axis off
if ndiv>=4
% for jj=1:nnode
% text(gcoord(jj,1),gcoord(jj,2),'o');
%figure(ndiv/2),scatter(gcoord(:,1),gcoord(:,2),'MarkerFaceColor','g')
%figure(ndiv/2),scatter(gcoord(:,1),gcoord(:,2),15,'g')
if ndiv>=4
%figure(ndiv/2),scatter(gcoord(:,1),gcoord(:,2),15,'filled','g')
hold on
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
end
figure(ndiv/2),scatter(x(1,1:4),y(1,1:4),20,'filled','k')
hold on
figure(ndiv/2),scatter(x(1,5:6),y(1,5:6),20,'filled','b')
hold on
figure(ndiv/2),scatter(x(1,7:8),y(1,7:8),20,'filled','r')
hold on
figure(ndiv/2),scatter(x(1,9:10),y(1,9:10),20,'filled','b')
hold on
figure(ndiv/2),scatter(x(1,11:12),y(1,11:12),20,'filled','r')
hold on
%figure(ndiv/2),scatter(x(1,1:nnel-4),y(1,1:nnel-4),15,'filled','g')
%figure(ndiv/2),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),15,'filled','r')
end
end
4]quadrilateral_mesh4MOINEX_q16LG.m
function[]=quadrilateral_mesh4MOINEX_q16LG(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)
%clc
%cclf
%(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_q16LG([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4,1,1)
%quadrilateral_mesh4MOINEX_q16LG([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_q16LG([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_q16LG([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,1,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygona
d_domain_coordinates_3rd_orderLG(n1,n2,n3,nmax,numtri,ndiv,mesh)
[nel,nnel]=size(nodes);
disp([xlength,ylength,nnode,nel,nnel])
%plot the mesh for the generated data
%x and y coordinates
xcoord(:,1)=gcoord(:,1);
ycoord(:,1)=gcoord(:,2);
extract coordinates for each element
cfl
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
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 tight
switch mesh
 case 1
axis([0 xlength 0 ylength])
 case 2

axis([0 xlength 0 ylength])
  case 3
axis([0 xlength 0 ylength])
  case 4
axis([-xlength/2 xlength/2 -ylength/2 ylength/2])
end
figure(ndiv/2)
plot(xvec,yvec,'r'); % plot element
hold on;
% place element number
midx=mean(xvec(1,1:4))
midy=mean(yvec(1,1:4))
if ndiv<=2
text(midx+.01,midy-.03, ['[',num2str(i),']']);
end
end;
% i loop
xlabel('x axis')
ylabel('y axis')
st1='Mesh of ';
st2=num2str(nel);
st3=' sixteen node ';
st4=' quadrilateral ';
st5=' elements & ';
st6=num2str(nnode);
st7='nodes'
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),[num2str(jj)]);
end
hold on
% figure(1), scatter(gcoord(:,1),gcoord(:,2),15,'filled','g')

hold on
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
figure(1), scatter(x(1,1:nnel-4),y(1,1:nnel-4),20,'filled','b')
figure(1), scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),20,'filled','r')
end
% i loop
end
% axis off
if ndiv>=4
  for jj=1:nnode
    text(gcoord(jj,1),gcoord(jj,2),['o']);
  end
  end
  hold on
  figure(ndiv/2), scatter(gcoord(:,1),gcoord(:,2),'MarkerFaceColor','g')
  figure(ndiv/2), scatter(gcoord(:,1),gcoord(:,2),15,'g')
if ndiv>=4
  figure(ndiv/2), scatter(gcoord(:,1),gcoord(:,2),15,'filled','g')
hold on
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
figure(ndiv/2), scatter(x(1,1:nnel-4),y(1,1:nnel-4),20,'filled','b')
figure(ndiv/2),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),20,'filled','r')
end

[H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482]

[5] D2LaplaceEquationQ12Ex3automeshgenNewContour.m
function[]=D2LaplaceEquationQ12Ex3automeshgenNewContour(n1,n2,n3,numtri,ndiv,mesh)
%function[]=improvedLaplaceEquationQuad8twodimensionEx3_explicitvfnmesh(nel,nnode,nnel,ndof,quadtype,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)]))
%sysms ff ss f sk N NN table1 table2
%D2LaplaceEquationQ12Ex3automeshgenNewContour(1,2,3,1,2,1)
%D2LaplaceEquationQ12Ex3automeshgenNewContour(1,2,3,1,2,2)
%g

%**********************************************************************
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
    x=sym([-sqrt(3);sqrt(3); 0])
    y=sym([-1;-1; 2])
end

%disp([ui vi wi])
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_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));

[eln,nodetel,nnode]=nodaladdresses_special_convex_quadrilaterals_3rd_order(ndiv);
[nel,nnel]=size(nodes);
%disp([nel nnel]=size(nodes);
%disp([nel nnel]=size(nodes);
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));
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
  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
\begin{verbatim}
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
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])
%table2=[N xx yy];
%integral values of local derivative products
[intJdndn]=integral_valuesof_localderivative_products(nnel);
%________________________________________________________________

for iel=1:nel
    index=zeros(nnel*nndof,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 gma;btb gmb]/delabc;
    GT=[bta gma;btb gmb]/delabc;
    Q=GT*G;
    sk(1:12,1:12)=(zeros(12,12));
    for i=1:12
        for j=i:12
            sk(i,j)=(delabc*sum(sum(Q.*(intJdndn(2*i-1:2*i,2*j-1:2*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);
    f=[-7/360; -1/48; -1/45; -1/48; 23/960; 1/30; 7/240; 37/960; 37/960; 7/240; 1/30; 23/960]*(2*delabc);
\end{verbatim}
% 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
%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f;
if mesh==2
    phi=phi/2;
end
for I=1:nnode
    NN(I,1)=I;
    phi_xi(I,1)=phi(I,1)-xi(I,1);
end
MAXPHI_XI=max(abs(phi_xi));

% disp('__________________________________________________________________')
% disp('number of nodes,elements & nodes per element')
% [nnode nel nnel ndof]
% disp('element number nodal connectivity for quadrilateral element')
% table1
% disp('__________________________________________________________________')
% disp('element number coordinates of the triangle spanning the quadrilateral element')
% table2
% disp('__________________________________________________________________')
% disp('node number Prandtl Stress Values')
% disp('fem-computed values anlytical(theoretical)-values')
% disp([NN 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('torisonal constant-----------------------------------------------')
%disp('fem-computed     analytical(theoretical)-values')

disp([nnod nel nnel])
disp([T k1 MAXPHI_XI])
%
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
                XX=xx(iel,1:3);
                YY=yy(iel,1:3);
                xa=XX(1,1);
                xb=XX(1,2);
                xc=XX(1,3);
                ya=YY(1,1);
                yb=YY(1,2);
                yc=YY(1,3);
                aLPa=xb*yc-xc*yb;
                aLPb=xc*ya-xa*yc;
                bta=yb-yc;
                btb=yc-ya;
                gma=xc-xb;
                gmb=xa-xc;
                delabc=gmb*bta-gma*btb;
                %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);
                nd9=nodes(iel,9); nd10=nodes(iel,10);nd11=nodes(iel,11);nd12=nodes(iel,12);
                %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(1,1),u(2,1),u(3,1),u(4,1),v(1,1),v(2,1),v(3,1),v(4,1));
                if (in==1)
                    X=x(i,j);Y=y(i,j);
                    p=(aLPa+bta*X+gma*Y)/delabc;
                    q=(aLPb+btb*X+gmb*Y)/delabc;
                    t0=[0;0];
                    [t]=convexquadrilateral_coordinates(u,v,X,Y);
                    %[t]=solveconvexquadrilateral_coordinates(u,v,X,Y);
                    %[t]=convexquadrilateral_coordinatesnew(u,v,X,Y);
                    %coordinates of quadrilateral(u,v)
                    r=t(1,1);
                    s=t(2,1);
                    shn1=((1-r)*(1-s)*(-10+9*(r^2+s^2)))/32;
                    shn2=((1+r)*(1-s)*(-10+9*(r^2+s^2)))/32;
            end
        end
    end
end
\[
shn3 = ((1+r)*(1+s)*(-10+9*(r^2+s^2)))/32;
\]
\[
shn4 = ((1-r)*(1+s)*(-10+9*(r^2+s^2)))/32;
\]
\[
shn5 = (9/32)*(1-s)*(1-r^2)*(1-3*r);\
\]
\[
shn6 = (9/32)*(1-s)*(1-r^2)*(1+3*r);\
\]
\[
shn7 = (9/32)*(1+r)*(1-s^2)*(1+3*s);\
\]
\[
shn8 = (9/32)*(1+r)*(1-s^2)*(1+3*s);\
\]
\[
shn9 = (9/32)*(1+s)*(1-r^2)*(1-3*r);\
\]
\[
shn10 = (9/32)*(1+s)*(1-r^2)*(1-3*r);\
\]
\[
shn11 = (9/32)*(1-r)*(1-s^2)*(1+3*s);\
\]
\[
shn12 = (9/32)*(1-r)*(1-s^2)*(1-3*s);\
\]

\[
PHI(i,j) = shn1*phi(nd1,1)+shn2*phi(nd2,1)+shn3*phi(nd3,1)+shn4*phi(nd4,1)+shn5*phi(nd5,1)+shn6*phi(nd6,1)+shn7*phi(nd7,1)+shn8*phi(nd8,1)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1);\
\]
\[
z(i,j) = ((Y+1)*((sqrt(3))*X-Y+2)*((-sqrt(3))*X-Y+2))/12;\
\]

\[
% THE PROGRAM EXECUTION JUMPS TO HERE if (in=1)\
% for iel\n% for j\
% for i\
% for ii=1:31\n% for jj=1:31\n% xx=(x(ii,jj));yy=(y(ii,jj));\n% z(ii,jj)=((yy+1/2)*((sqrt(3))*xx-yy+1)*((-sqrt(3))*xx-yy+1))/6;\n% end %ii\n% end %jj
\]

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

switch mesh
   case 2
   clf,figure(1)
   clf,figure(2)
   clf,figure(3)
   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)&(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);
end
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='Nine 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])
figure(2)
hold on
[x,y]=meshgrid(0:1:10:0.1:1)
% y((y>1/2) & (y<=1) & (x>1/2) & (x<=1) & (x+y>3/2)) = NaN;
%[c,h]=contour(x,y,z)
contour(x,y,PHI,'r')
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square
title('contour level curves for exact solution: ')
hold off
figure(3)
hold on
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
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='Nine Noded ';
st4='Special Quadrilateral';
st5='Elements'
title([st1,st2,st3,st4,st5])
sst1=' NODES='
sst2=num2str(nnnode)
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]

format long
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('---------------------------------------------------------------------------------------------' )

[ 6]D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour.m
function []=D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour(n1,n2,n3,nmax,numtri,ndiv,mesh)
%ndiv=2,4,6,...
%polygonal_domain_coordinates([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,1,2,12)
%polygonal_domain_coordinates([1;1;1;1;1;1;2;3;4;5;6;7;8;9;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;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,1,2,12)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;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;2;8,9,2,9,1,2,2,2)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,4,4,2)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;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;2;9,1,2,2,1,2)
%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;2;8,1,2,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,4,4,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,6,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,16,8,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,25,10,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ8MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;2;8,9,2,9,1,2,2,2)
%D2LaplaceEquationQ9Ex3automeshgenNewPolygon([1;1;1;1;1;1;1;1;2;3;4;5;6;7;8;9;3;4;5;6;7;8;9,2,9,1,2,4,2)

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482  Page 23412
%%D2LaplaceEquationQ9Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],[9,1,2,4])
%%D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],[9,1,2,4])
%%D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],[9,1,2,4])
syms coord
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
%nnel=4;
%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
%>>LaplaceEquationQuad4twodimensionEx2_explicit(69,83,4,1)#
%>>LaplaceEquationQuad4twodimensionEx2_explicit_fnmesh(69,83,4,1)#
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel=3,nnode=7,nnel=4,ndof=1)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel,nnode,nnel=4,ndof=1,quadtype=0/3,mesh=1,2,3...)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel=12,nnode=19,nnel=4,ndof=1,quadtype=0/3,mesh=3)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel=27,nnode=37,nnel=4,ndof=1,quadtype=0/3,mesh=4)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel=48,nnode=61,nnel=4,ndof=1,quadtype=0/3,mesh=5)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(nel=75,nnode=91,nnel=4,ndof=1,quadtype=0/3,mesh=6)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(108,127,4,1,3,7)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(147,169,4,1,3,8)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(192,217,4,1,3,9)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmex(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_ritisosccelestriangle00_h0_hh(ndiv);
%[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
%
%bcdof=[2;5;3]
%boundary conditions-1
switch mesh

case 4
%boundary conditions-2
nnn=0;
for nn=1:nnode
xnn=coord(nn,1);ynn=gcoord(nn,2);
if (xnn==-1/2)&(ynn>=-1/2)&(ynn<=1/2)
 nnn=nnn+1;
 bcdof(nn,1)=nn;
bval(nn,1)=0;
end
end
%boundary conditions-2
for nn=1:nnode
xnn=gcoord(nn,1);ynn=coord(nn,2);
}
if (ynn==1/2)&((xnn>=-1/2)&(xnn<=1/2))
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==1/2)&((xnn>=-1/2)&(xnn<=1/2))
    nnn=nnn+1;
bcdof(nnn,1)=nn;
bval(nnn,1)=0;
  end
end

%%boundary conditions-4
for nn=1:nnode
  xnn=coord(nn,1);ynn=gcoord(nn,2);
  if (xnn==1/2)&((ynn>=-1/2)&(ynn<=1/2))
    nnn=nnn+1;
bcdof(nnn,1)=nn;
bval(nnn,1)=0;
  end
end

% end

bcdof

nnn=length(bcdof);

format long g
k1 =double(0.14057701495515551037840396020329);
x1=(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
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

% integral values of local derivative products
[intJdndn]=integral_valuesof_localderivative_products(nnel);

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);
aya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
ba=ya-yc;btb=yc-ya;
gma=xa-xb;gmb=xa-xc;
delabc=gma*bta-gmb*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;
sk(1:12,1:12)=(zeros(12,12));
for i=1:12
 for j=i:12
  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
f=[-7/360; -1/48; -1/45; -1/48; 23/960; 1/30; 7/240; 37/960; 57/960; 7/240; 1/30; 23/960]*2*delabc)
%f=[19/11520; 1/384; 41/11520; 1/384; 192/3840; 31/3840; 1/96; 1/96; 31/3840; 29/3840; 1/192; 21/1280; 3/128; 39/1280; 3/128]*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);
end
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);

[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(phi_xi));
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
disp('fem-computed values anlytical(theoretical)-values ')
disp([NN phi xi])
disp('number of nodes,elements & nodes per element')
[nnode nel nnel]
disp([nnode nel nnel ndof])

gcoord

%********************************************
disp([NN phi xi])
t=0;
for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
end
T=t;
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([T k1 MAXPHI_XI])
disp('----------------------------')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
%
[x,y]=meshgrid(-1/2:0.05:1/2,-1/2:0.05:1/2);
a0=8/pi^3;

for i=1:21
    for j=1:21
        for iel=1:nel
            XX=xx(iel,1:3);
            YY=yy(iel,1:3);
            %disp([X Y])
            xa=XX(1,1);
            xb=XX(1,2);
            xc=XX(1,3);
            ya=YY(1,1);
            yb=YY(1,2);
            yc=YY(1,3);
            aLPa=xb*yc-xc*yb;
            aLPb=xc*ya-xa*yc;
            bta=yb-yc+bth=yc-ya;
            gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;

% 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);
nd9=nodes(iel,9);nd10=nodes(iel,10);nd11=nodes(iel,11);nd12=nodes(iel,12);
% nd13=nodes(iel,13);nd14=nodes(iel,14);nd15=nodes(iel,15);nd16=nodes(iel,16);
%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);
p=(aLPa+bta*X+gma*Y)/delabc;
q=(aLPb+btb*X+gmb*Y)/delabc;
t0=[0.5;0.5];
%[t]=convexquadrilateral_coordinates(u,v,X,Y);
%[t]=solveconvexquadrilateral_coordinates(u,v,X,Y);
%[t]=convexquadrilateral_coordinatesnew(u,v,X,Y);
%[t]=convexquadrilateral_naturalcoordinates([1/3;0;0;1/2],[1/3;1/2;0;0],U,V)
[t,iter] = newtonmethod4spquadrilateralspquad(t0,p,q,'parameqnspqd','paramdetJspqd','paraminvJspqd')
r=t(1,1);
s=t(2,1);

shn1=((1-r)*(1-s)*(-10+9*(r^2+s^2)))/32;
shn2=((1+r)*(1-s)*(-10+9*(r^2+s^2)))/32;
shn3=((1+r)*(1+s)*(-10+9*(r^2+s^2)))/32;
shn4=((1-r)*(1+s)*(-10+9*(r^2+s^2)))/32;
shn5=(9/32)*((1-s)*(1-r^2)*(1-3*r));
shn6=(9/32)*((1-s)*(1+r^2)*(1+3*r));
shn7=(9/32)*((1+r)*(1-s^2)*(1-3*s));
shn8=(9/32)*((1+r)*(1-s^2)*(1+3*s));
shn9=(9/32)*((1+s)*(1-r^2)*(1+3*r));
shn10=(9/32)*((1+s)*(1+r^2)*(1-3*r));
shn11=(9/32)*((1-r)*(1-s^2)*(1+3*s));
shn12=(9/32)*((1-r)*(1-s^2)*(1-3*s));

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)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1);
%--------------------------------------------------------------------------------------------------------------------------------

break
end%if (in==1)
end%for iel
%THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
end%for j
end%for i

a0=8/pi^3;
for ii=1:21
for jj=1:21
xx=(x(ii,jj));yy=(y(ii,jj));rr=0;

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

\% z=sin(pi*x).*sin(pi*y);

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

case 4

    hold off
clf
figure(1)
[x,y]=meshgrid(-1/2:.05:1/2,-1/2:0.05:1/2)
%[c,h]=contour(x,y,PHI,40)
contour(x,y,PHI,40)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for '; st2='FEM solution of ';
st3='Twelve Noded ';
st4='Special Quadrilateral'; st5=' Elements'
title([st1 st2 st3 st4 st5])
sst1='(MESH HAS ' sst2=num2str(nnode) sst3=' NODES ' sst4=' AND ' sst5=num2str(nel) sst6=' ELEMENTS)'
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
hold off
figure(2)
%[x,y]=meshgrid(0:.1:1.0;0:.1:1)
[x,y]=meshgrid(-1/2:.05:1/2,-1/2:0.05:1/2)
%[c,h]=contour(x,y,z,40)
contour(x,y,z,40)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
title('contour level curves for exact solution: in a series')
mm=0;
for i=1:21
    for j=1:21
        mm=mm+1;
        femsln(mm,1)=PHI(i,j);
        exactsln(mm,1)=z(i,j);
    end
end
%++++++++++++++++
hold off
figure(3)
[x,y]=meshgrid(-1/2:.05:1/2,-1/2:0.05:1/2);
%[x,y]=meshgrid(-sqrt(2)/2:0.1*sqrt(2)/2:sqrt(2)/2,-sqrt(2)/2:0.1*sqrt(2)/2:sqrt(2)/2);
%[c,h]=contour(x,y,PHI,'r-')
contour(x,y,PHI,40,'r-')
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM(red)&exact(green) ';
st3='Twelve Noded ';
st4='Special Quadrilateral';
st5='Elements'
title([st1,st2,st3,st4,st5])
sst1=('MESH HAS ')
sst2='NUM2STR(nn)'+
sst3='NODES' 
sst4='AND ' 
sst5='NUM2STR(nel) '
sst6='ELEMENTS')
text(-1/2,-1/2,[sst1 st2 st3 st4 st5 st6])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,PHI,'g-')
contour(x,y,PHI,40,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square

end
%switch mesh
%[fem soln exact soln]

disp('number of nodes,elements & nodes per element')
[nnode nel nel nel]
[1 phi(1,1) xi(1,1)]
disp('number of nodes,elements & nodes per element')
[nnode nel nel nel]
disp('fem-computed values anlytical(theoretical)-values ')
disp([NN phi xi])
disp('number of nodes,elements & nodes per element')
[nnode nel nel nel]

if mesh==4
%[phi xi]
for l=1:nnode
NN(1,1)=l;
phi_xi(l,1)=phi(l,1)-xi(l,1);
end
MAXPHI_XI=max(abs(double(phi_xi)));
format long e
t=0;
for iii=1:nnode
 t=t+phi(iii,1)*ff(iii,1);
end
T=t;
disp('number of nodes,elements & nodes per element')
disp([nnode nel nnel])
disp('torisonal constants(fem=phi&exact=ksi) error(max(abs(phi_xi))')
%disp('---------------------------------------------------------------------------------------------'%disp([nnode nel nnel])
disp('T k1 MAXPHI_XI')
disp('---------------------------------------------------------------------------------------------'%disp([nnode nel nnel])
for ncpt=1:nel
c1=nodes(ncpt,1);c2=nodes(ncpt,2);c3=nodes(ncpt,3);c4=nodes(ncpt,4);
elcentr(ncpt,1)=ncpt;
phicpt(ncpt,1)=(phi(c1,1)+phi(c2,1)+phi(c3,1)+phi(c4,1))/4;
xicpt(ncpt,1)=(ksi(c1,1)+ksi(c2,1)+ksi(c3,1)+ksi(c4,1))/4;
end
for I=1:nel
elnumm(I,1)=I;
end
disp('_____________________________________________________________________________________')
disp([elnumm elcentr phicpt xicpt])
disp('_____________________________________________________________________________________')
format compact
%===========================================================================================
\[
\text{ELM FEM SOLUTION EXACT SOLUTION ELM FEM SOLUTION EXACT SOLUTION ELCENTNODE FEM SOLUTION EXACT SOLUTION}
\]
for I=1:3:nel
  A=[elcentr(I) phicpt(I) xicpt(I)];B=[elcentr(I+1) phicpt(I+1) xicpt(I+1)];C=[elcentr(I+2) phicpt(I+2) xicpt(I+2)];
  fprintf('
%5d %18.14f %18.14f %5d %18.14f %18.14f %5d %18.14f %18.14f',elcentr(I), phicpt(I), xicpt(I), elcentr(I+1), phicpt(I+1), xicpt(I+1), elcentr(I+2), phicpt(I+2), xicpt(I+2));
  fprintf('
')
disp('_____________________________________________________________________________________')
end
fprintf('
')
function []=D2LaplaceEquationQ16Ex3automeshgenNewContour(n1,n2,n3,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)]))
%syms ff ss f sk N NN table1 table2
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,1)
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,2)
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,1)
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,2)
syms coord
dofof=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
        x=sym([-sqrt(3);sqrt(3); 0])
y=sym([-1; -1; 2])
end

syms ui vi xi yi
%[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order(ndiv);
%[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order_LAGR(ndiv)
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_order_LAGR(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);
%[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_2nd_order(ndiv);
[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_order(ndiv);
%*[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_order_LAGR(n)
%*[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_order_LAGR(n)
%******************************************************************************
%**sym coord
%**ndof=1;
%*[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_2nd_order(ndiv);
%*[coord,gcoord]=coordinate_risoscelestriangle_00_h0_hh_2ndorder_LAGR(ndiv);
%*[coord,gcoord]=coordinate_risoscelestriangle_00_h0_hh_2ndorder_LAGR(ndiv);
%*[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
%*[nel,nnel]=size(nodes);
%disp([nel nnode nel 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));

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

% table=[N nodes]
nel,nnode]=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(nnod,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;
bval(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;
bval(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;
bval(nnn,1)=0;
 end
end
% switch mesh
bcdof = 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
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
table2=[N xx yy];
syms r s
syms xa xb xc
syms ya yb yc
format long g
for i=1:nel
N(i,1)=i;
end
[intJdndn]=integral_valuesof_localderivative_products(nnel);
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;bth=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*bth;
G=[bta gma;bth gmb]/delabc;
GT=[bta gma;bth gmb]/delabc;
Q=GT*G;
sk(1:16,1:16)=(zeros(16,16));
for i=1:16
for j=i:16
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)
%f=[-7/360; -1/48; -1/45; -1/48; 23/960; 1/30; 7/240; 37/960; 37/960; 7/240; 1/30; 23/960]*(2*delabc)
f=[19/11520;1/384;41/11520;1/384;1/192;29/3840;31/3840;1/96;1/96;31/3840;29/3840;1/192;21/1280;3/128;39/1280;3/128]*2*
delabc;
edof=nnel*ndof;
k=0;
for i=1:mel
    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
%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f;
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(phi_xi));

%disp('__________________________________________________________________
%disp('number of nodes,elements & nodes per element')
%disp('[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])

%disp('switch mesh')
t=0;
for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
end
switch mesh
    case 1
        T=8*t;
    end
case 2
T=2*t;
end

%disp(' ---------------torisonal constant------------------------')
%disp(' fem-computed       anlytical(theoretical)-values ')
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
      XX=xx(iel,1:3);
      YY=yy(iel,1:3);
      xa=XX(1,1);
      xb=XX(1,2);
      xc=XX(1,3);
      ya=YY(1,1);
      yb=YY(1,2);
      yc=YY(1,3);
      aLPa=xb*yc-xc*yb;
      aLPb=xc*ya-xa*yc;
      bta=yb-yc;
      btb=yc-ya;
      gma=xc-xb;
      gmb=xa-xc;
      delabc=gmb*bta-gma*btb;
    end
    XX(i,j)=x(i,j);YY(i,j)=y(i,j);
    u(1,1)=gcoord(nd1,1);u(2,1)=gcoord(nd2,1);
    v(1,1)=gcoord(nd1,2);v(2,1)=gcoord(nd2,2);
    x(i,j)=x(i,j);y(i,j)=y(i,j);
  end
end

h1r=-9*(r+1/3)*(r-1/3)*(r-1)/16;
h2r=27*(r+1)*(r-1)/3*(r-1)/16;
h3r=-27*(r+1)*(r+1/3)*(r-1)/16;
h4r=9*(r+1)*(r+1/3)*(r-1)/16;

h1s=-9*(s+1/3)*(s-1/3)*(s-1)/16;
h2s=27*(s+1)*(s-1/3)*(s-1)/16;
h3s=-27*(s+1)*(s+1/3)*(s-1)/16;
h4s=9*(s+1)*(s+1/3)*(s-1)/16;

\% 
shn1=h1r*h1s;shn5=h2r*h1s;shn6=h3r*h1s;shn2=h4r*h1s;
shn12=h1r*h2s;shn13=h2r*h2s;shn14=h3r*h2s;shn7=h4r*h2s;
shn11=h1r*h3s;shn16=h2r*h3s;shn15=h3r*h3s;shn8=h4r*h3s;
shn4=h1r*h4s;shn10=h2r*h4s;shn9=h3r*h4s;shn3=h4r*h4s;

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)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1)+shn13*phi(nd13,1)+shn14*phi(nd14,1)+shn15*phi(nd15,1)+shn16*phi(nd16,1);

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
% 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 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 if (mesh==2)

switch mesh
  case 2
cif
figure(1)
x=[-sqrt(3):sqrt(3):0];
y=[  -1:  -1:2];
patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:0.1:10:0.0:1:1)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3);sqrt(3);-1:0.1);2;
%y=((y-3/2)/2);(y<1&x less than 1&x<1&x<y);NaN;
%y=((y-3/2)/2);(y<1&x less than 1&x>y);NaN;
%y=((y-3/2)/2);(y<1&x less than 1&x=y);NaN;
%c,h=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%c,label(c,h);
axis square
st1='Contour level curves for '; st2='FEM solution of '; st3='Sixteen Noded '; st4='Special Quadrilateral'; st5='Elements'
title([st1,st2,st3,st4,st5])
sst1=('MESH HAS ' num2str(nnode))
sst2=' NODES'
sst4='AND ' 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])
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')
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')
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];
p
% contour(x, y, z).
xlabel('X-axis'); ylabel('Y-axis');
axis square

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482
hold on

[x,y]=meshgrid(0:.1:1,0:0.1:1)
[c,h]=contour(x,y,z,'g-')
contour(x,y,z,40,'b-')
xlabel('X-axis');
ylabel('Y-axis');
clabel(c,h);
axis square

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

% [femsoln exactslofn]
format long
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('------------------------------------------------------------------------------------------')

[8]D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour.m

function[]=D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour(n1,n2,n3,nmax,numtri,ndiv,mesh)

syms coord
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_orderLG(n1,n2,n3,nmax,numtri,ndiv,mesh)
nnel=16;
ndof=1;
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
disp([nel nnode nnel ndof])
format long
g
for i=1:nel
    N(i,1)=i;
end
for i=1:nel
    NN(i,1)=i;
end
%
switch mesh
case 4
%boundary conditions-2
nnn=0;
for nn=1:nnode
  xnn=coord(nn,1);ynn=gcoord(nn,2);
  if (xnn==-1/2)&((ynn>=-1/2)&(ynn<=1/2))
    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==-1/2)&((xnn>=-1/2)&(xnn<=1/2))
    nnn=nnn+1;
    bcdof(nnn,1)=nn;
    bcval(nnn,1)=0;
  end
end
%boundary conditions-3
for nn=1:nnode
  xnn=coord(nn,1);ynn=gcoord(nn,2);
  if (ynn==1/2)&((xnn>=-1/2)&(xnn<=1/2))
    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==1/2)&((ynn>=-1/2)&(ynn<=1/2))
    nnn=nnn+1;
    bcdof(nnn,1)=nn;
    bcval(nnn,1)=0;
  end
end
%end
bcdof
mm=length(bcdof);

format long g
k1 =double(0.14057701495515551037840396020329);
x1=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
%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


\[ \text{table2} = \begin{bmatrix} N & xx & yy \end{bmatrix}; \]
\[ \text{disp([} xx \ yy \text{])} \]

\[ [\text{intJdndn}] = \text{integral_valuesof_localderivative_products(nnel)}; \]

\[
\text{for } iel = 1:nel \\
\text{index} = \text{zeros(nnel*ndof,1)}; \\
X = xx(iel,1:3); \\
Y = yy(iel,1:3); \\
\text{disp([} X \ Y \text{])} \\
x_a = X(1,1); \\
x_b = X(1,2); \\
x_c = X(1,3); \\
y_a = Y(1,1); \\
y_b = Y(1,2); \\
y_c = Y(1,3); \\
\text{bta} = y_b - y_c; \\
\text{btb} = y_c - y_a; \\
\text{gma} = x_c - x_b; \\
\text{gmb} = x_a - x_c; \\
\text{delabc} = \text{gmb} \times \text{bta} - \text{gma} \times \text{btb}; \\
G = [\text{bta} \ \text{btb}; \text{gma} \ \text{gmb}] / \text{delabc}; \\
\text{GT} = [\text{bta} \ \text{gma}; \text{btb} \ \text{gmb}] / \text{delabc}; \\
Q = \text{GT} \times \text{G}; \\
\text{sk(1:16,1:16) = zeros(16,16)}; \\
\text{for } i = 1:16 \\
\text{for } j = i:16 \\
\text{sk(i,j) = (delabc * sum(sum(Q.*(intJdndn(2*i-1:2*i,2*j-1:2*j))))));} \\
\text{sk(j,i) = sk(i,j);} \\
\text{end} \\
\text{end} \\
\]

\[ f = \begin{bmatrix} 5/144; 1/24; 7/144; 1/24 \end{bmatrix} \times (2 \times \text{delabc}) \]
\[ f = \begin{bmatrix} -5/432; -1/72; -5/432; -1/72; 11/216; 13/216; 13/216; 11/216 \end{bmatrix} \times (2 \times \text{delabc}) \]
\[ f = \begin{bmatrix} -7/360; -1/48; -1/48; -1/48; 23/960; 7/240; 37/960; 37/960; 7/240; 1/30; 23/960 \end{bmatrix} \times (2 \times \text{delabc}) \]
\[ f = \begin{bmatrix} 19/11520; 1/384; 41/11520; 1/384; 1/192; 29/3840; 31/3840; 1/96; 1/96; 31/3840; 29/3840; 1/192; 21/1280; 3/128; 39/1280; 3/128 \end{bmatrix} \times (2 \times \text{delabc}) \]

\[ \text{edof} = \text{nnel*ndof}; \]
\[
\text{for } i = 1:nnel \\
\text{nd(i,1) = nodes(iel,i);} \\
\text{start} = (\text{nd(i,1)-1} \times \text{ndof}); \\
\text{for } j = 1:\text{ndof} \\
\text{k = k+1;} \\
\text{index(k,1) = start+j;} \\
\text{end} \\
\text{end} \\
\]

\[ \text{ff(ii,1) = ff(ii,1) + f(i,1);} \\
\text{for } j = 1:\text{edof} \\
\text{jj = index(j,1);} \\
\text{ss(ii,jj) = ss(ii,jj) + sk(i,j);} \\
\text{end} \\
\text{end} \\
\text{end} \]
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,1:nnode)=zeros(1,nnode);
    ss(1:nnode,kk)=zeros(nnode,1);
end
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,kk)=1;
end
phi=ss*ff;
phi=double(phi);
[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(phi_xi));

t=t+phi(iii,1)*ff(iii,1);
T=t;

%disp('number of nodes,elements & nodes per element')
[nnod nel nnel ndof]
%disp([NN phi xi])
%disp('number of nodes,elements & nodes per element')
[nnod nel nnel ndof]

%**************************************
[x,y]=meshgrid(-1/2:0.05:1/2,-1/2:0.05:1/2);
a0=8/pi^3;

%bcdf=[13;37;35;33;31;29;27;25;23;21;19;17;15];
YY=yy(iel,1:3);
%disp([X Y])
xa=XX(1,1);
xb=XX(1,2);
xc=XX(1,3);
ya=YY(1,1);
yb=YY(1,2);
yc=YY(1,3);
aLPa=xb*yc-xc*yb;
aL Pb=xc*ya-xa*yc;
bta=yb-yc;btb=yc-ya;
gm a=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;

% 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);
nd9=nodes(iel,9);nd10=nodes(iel,10);nd11=nodes(iel,11);nd12=nodes(iel,12);
nd13=nodes(iel,13);nd14=nodes(iel,14);nd15=nodes(iel,15);nd16=nodes(iel,16);
% 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);
p=(aLPa+bta*X+gma*Y)/delabc;
q=(aLPb+btb*X+gmb*Y)/delabc;
t0=[0.5;0.5];
[t,iter] = ewtonmethod4spquadrileteral(t0,p,q,'parameqnspqd','paramdetJspqd','paraminvJspqd')
\quad r=t(1,1);
\quad s=t(2,1);

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

h1r=-9*(r+1/3)*(r-1/3)*(r-1)/16;
h2r=27*(r+1)*(r-1/3)*(r-1)/16;
h3r=-27*(r+1)*(r+1/3)*(r-1)/16;
h4r=9*(r+1)*(r+1/3)*(r-1)/16;
h1s=-9*(s+1/3)*(s-1/3)*(s-1)/16;
h2s=27*(s+1)*(s-1/3)*(s-1)/16;
h3s=-27*(s+1)*(s+1/3)*(s-1)/16;
h4s=9*(s+1)*(s+1/3)*(s-1)/16;
% shn1=h1r*h1s;shn5=h2r*h1s;shn6=h3r*h1s;shn2=h4r*h1s;
shn12=h1r*h2s;shn13=h2r*h2s;shn14=h3r*h2s;shn7=h4r*h2s;
shn11=h1r*h3s;shn16=h2r*h3s;shn15=h3r*h3s;shn8=h4r*h3s;
shn4=h1r*h4s;shn10=h2r*h4s;shn9=h3r*h4s;shn3=h4r*h4s;
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)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1)+shn13*phi(nd13,1)+shn14*phi(nd14,1)+shn15*phi(nd15,1)+shn16*phi(nd16,1);
break
end if (in==1)
end for iel
end for j
end for i
a0=8/pi^3;
for ii=1:21
\quad for jj=1:21
\[
xx = (x(ii,jj));
yy = (y(ii,jj));
rr = (0);
\]

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

\[
% z = \sin(pi*x) .* \sin(pi*y);
\]

for \( i = 1:21 \)
for \( j = 1:21 \)
if \( \text{abs}(PHI(i,j)) <= 1e-5 \)
\[
PHI(i,j) = 0;
\]
end
if \( \text{abs}(z(i,j)) <= 1e-5 \)
\[
z(i,j) = 0;
\]
end
end
end

\text{switch mesh}

\text{case 4}

\text{hold off}
\text{clf}
\text{figure(1)}
\text{[x,y]=meshgrid(-1/2:.05:1/2,-1/2:0.05:1/2)}
\[
%[c,h]=contour(x,y,PHI)
\]
\text{contour(x,y,PHI,40)}
\text{xlabel('X-axis');}
\text{ylabel('Y-axis');}
\[
%clabel(c,h);
\]
\text{axis square}
\text{st1='Contour level curves for '};
\text{st2='FEM solution of '};
\text{st3='Sixteen Noded '};
\text{st4='Special Quadrilateral';}
\text{st5=' Elements'}
\text{title([st1 st2 st3 st4 st5])}
\text{sst1='(MESH HAS '}
\text{sst2=num2str(nnode)}
\text{sst3=' NODES'}
\text{sst4=' AND '}
\text{sst5=num2str(nel)}
\text{sst6=' ELEMENTS')}
\text{text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])}
\text{hold on}
\text{figure(2)}
\[
%[x,y]=meshgrid(0.1:1:0.1:0.1)
\]
\text{[x,y]=meshgrid(-1/2:.05:1/2,-1/2:0.05:1/2)}
\[
%[c,h]=contour(x,y,z)
\]
\text{contour(x,y,z,40)}
\text{xlabel('X-axis');}
\text{ylabel('Y-axis');}
\[
%clabel(c,h);
\]
\text{axis square}
\text{title('contour level curves for exact solution: in a series')}
\text{mm=0;}
\text{for \( i = 1:21 \)}
\text{for \( j = 1:21 \)}
\text{mm=mm+1;}
\text{femso(n(mm,1))=PHI(i,j);}
exactsoln(mm,1)=z(i,j);
end
end

hold on

figure(3)
[x,y]=meshgrid(-1/2:.05:1/2,-1/2:.05:1/2)
%[x,y]=meshgrid(-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2,-sqrt(2)/2:(0.1)*sqrt(2)/2:sqrt(2)/2);
%[c,h]=contour(x,y PHI,'r-')
contour(x,y,PHI,40,'r-')
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM(red)&exact(green) ';
st3='Sixteen Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1 st2 st3 st4 st5])
sst1='(MESH HAS ';
sst2=num2str(nnode)
sst3=' NODES' 
sst4=' AND ' 
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1/2,-1/2,[sst1 st2 st3 st4 st5 st6])
hold on

%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,40,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square

end%switch mesh

 pueblo

end

if mesh==4
%[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)));
t=0;
for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
end
T=t;
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('number of nodes, elements & nodes per element')
    %disp([nnode nel nnel])
disp([T k1 MAXPHI_XI])
end
disp('_____________________________________________________________________________________')
disp(['number of nodes,elements & nodes per element'])
[nnode nel nnel ndof] for ncp=t=1:nel
c1=nodes(ncpt,13);c2=nodes(ncpt,14);c3=nodes(ncpt,15);c4=nodes(ncpt,16);
elcentr(ncpt,1)=ncpt;
phicpt(ncpt,1)=(phi(c1,1)+phi(c2,1)+phi(c3,1)+phi(c4,1))/4;
xicpt(ncpt,1)=(xi(c1,1)+xi(c2,1)+xi(c3,1)+xi(c4,1))/4;
end for I=1:nel
elnumm(I,1)=I;
end
disp('_____________________________________________________________________________________')
disp('serial no        center point                fem-computed values            analytical(theoretical)-values')
disp([elnumm  elcentr phicpt xicpt])
disp('_____________________________________________________________________________________')
format compact
%========================================================================================================
%========================================================================================================
%========================================================================================================
%---------------------------------------------------------------------
disp('__________________________________________________________________')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof] for ncp=t=1:nel
c1=nodes(ncpt,13);c2=nodes(ncpt,14);c3=nodes(ncpt,15);c4=nodes(ncpt,16);
elcentr(ncpt,1)=ncpt;
phicpt(ncpt,1)=(phi(c1,1)+phi(c2,1)+phi(c3,1)+phi(c4,1))/4;
xicpt(ncpt,1)=(xi(c1,1)+xi(c2,1)+xi(c3,1)+xi(c4,1))/4;
end for I=1:nel
elnumm(I,1)=I;
end
disp('__________________________________________________________________________')
disp('NODE FEM SOLUTION EXACT SOLUTION NODE FEM SOLUTION EXACT SOLUTION NODE FEM SOLUTION EXACT SOLUTION')
disp('----------------------------------------------------------------------------------')
for I=1:3:nel
    % A=[elcentr(I) phicpt(I) xicpt(I)];B=[elcentr(I+1) phicpt(I+1) xicpt(I+1)];C=[elcentr(I+2) phicpt(I+2) xicpt(I+2)];
    %disp([elcentr(I) phicpt(I) xicpt(I) elcentr(I+1) phicpt(I+1) xicpt(I+1)])
    %disp([elcentr(I+1) phicpt(I+1) xicpt(I+1) elcentr(I+2) phicpt(I+2) xicpt(I+2)])
    %disp([A B C])
    fprintf('%5d %18.14f %18.14f %5d %18.14f %18.14f %5d %18.14f %18.14f elcentr(I), phicpt(I), xicpt(I), elcentr(I+1), phicpt(I+1), xicpt(I+1), elcentr(I+2), phicpt(I+2), xicpt(I+2));
end
fprintf('
')
disp('----------------------------------------------------------------------------------')

[9] polygonal_domain_coordinates_3rd_orderLG.m
function [coord, gcoord, nodes, nodetel, nnode, nel] = polygonal_domain_coordinates_3rd_orderLG(n1, n2, n3, nmax, numtri, n, mesh)
    %n1=node number at (0,0) for a chosen triangle
    %n2=node number at (1,0) for a chosen triangle
    %n3=node number at (0,1) for a chosen 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 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],1,5,1,2)
%[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],5,1,2,3)
%[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;1],
%[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([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)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order([1:1:1:1],[2;3;4;5],[3;4;5;2],5,1,2,3
%syms U V W xi yi
%switch mesh
%case 1 %domain with seven triangles(8-nodes)
x=sym([1/2;1/2;1:1:1/2;0;0;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;1:1:1/2;0;0;0;0;0]);
%FOR UNIT SQUARE
%y=sym([1/2;0;0;1/2;1;1;1/2;0]);
%FOR UNIT SQUARE
%case 3 %square domain with four triangles(5-nodes)
x=sym([1/2;0;1;1;0]);
%y=sym([1/2;0;0;1;1]);
%case 4 %square domain with eight triangles(9-nodes)
% 1 2 3 4 5 6 7 8 9
%x=sym([0;0;1/2;1/2;0;1/2;1/2;0;1/2]);
%FOR UNIT SQUARE
%y=sym([0;0;1/2;1/2;0;1/2;1/2;0;1/2]);
%FOR UNIT SQUARE
%end
%[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_order(n1,n2,n3,nmax,numtri,n);
[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(nnel)])

x1(1:nnode,1)=zeros(nnode,1);
y1(1:nnode,1)=zeros(nnode,1);
nitri=nmax-1;
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)
\[ y_1 = y(n_1(\text{itri}, 1), 1) \]
\[ y_2 = y(n_2(\text{itri}, 1), 1) \]
\[ y_3 = y(n_3(\text{itri}, 1), 1) \]
\[ \text{rrr}(\cdot, \cdot, \cdot, \text{itri}) \]
\[ U' \]
\[ V' \]
\[ W' \]
\[ kk = 0; \]
\[ \text{for } ii = 1:n+1 \]
\[ \quad \text{for } jj = 1:(n+1)-(ii-1) \]
\[ \quad \quad \text{kk} = \text{kk} + 1; \]
\[ \quad \quad \text{mm} = \text{rrr}(ii, jj, \text{itri}); \]
\[ \quad \quad uu = U(kk, 1); vv = V(kk, 1); ww = W(kk, 1); \]
\[ \quad \quad xi(mm, 1) = x_1*ww + x_2*uu + x_3*vv; \]
\[ \quad \quad yi(mm, 1) = y_1*ww + y_2*uu + y_3*vv; \]
\[ \quad \text{end} \% \text{for } jj \]
\[ \text{end} \% \text{for } ii \]
\[ [xi \ yi] \]
\[ \% \text{add coordinates of centroid} \]
\[ \text{ne} = (n/2)^2; \]
\[ \% \text{stdnode}=kk; \]
\[ \text{for } iii = 1+(\text{itri}-1)*\text{ne}:\text{ne}^*\text{itri} \]
\[ \quad \% \text{kk} = \text{kk} + 1; \]
\[ \quad \text{node1} = \text{eln}(iii, 1) \]
\[ \quad \text{node2} = \text{eln}(iii, 2) \]
\[ \quad \text{node3} = \text{eln}(iii, 3) \]
\[ \quad \text{mm} = \text{eln}(iii, 7) \]
\[ \quad xi(mm, 1) = (xi(node1, 1) + xi(node2, 1) + xi(node3, 1))/3; \]
\[ \quad yi(mm, 1) = (yi(node1, 1) + yi(node2, 1) + yi(node3, 1))/3; \]
\[ \text{end} \% \text{for } iii \]
\[ [xi \ yi] \]
\[ \% \text{for itri}=1:\text{nitri} \]
\[ \text{for } mmm = 1:\text{nel} \]
\[ \quad mmm1 = \text{nodes}(mmm, 1) \]
\[ \quad mmm2 = \text{nodes}(mmm, 2) \]
\[ \quad mmm3 = \text{nodes}(mmm, 3) \]
\[ \quad mmm4 = \text{nodes}(mmm, 4) \]
\[ \quad mmm5 = \text{nodes}(mmm, 5) \]
\[ \quad mmm6 = \text{nodes}(mmm, 6) \]
\[ \quad mmm7 = \text{nodes}(mmm, 7) \]
\[ \quad mmm8 = \text{nodes}(mmm, 8) \]
\[ \quad mmm9 = \text{nodes}(mmm, 9) \]
\[ \quad mmm10 = \text{nodes}(mmm, 10) \]
\[ \quad mmm11 = \text{nodes}(mmm, 11) \]
\[ \quad mmm12 = \text{nodes}(mmm, 12) \]
\[ \% \]
\[ \quad mmm13 = \text{nodes}(mmm, 13) \]
\[ \quad mmm14 = \text{nodes}(mmm, 14) \]
\[ \quad mmm15 = \text{nodes}(mmm, 15) \]
\[ \quad mmm16 = \text{nodes}(mmm, 16) \]
\[ \% \]
\[ \quad xi1 = xi(mmm1, 1) \]
\[ \quad xi2 = xi(mmm2, 1) \]
\[ \quad xi3 = xi(mmm3, 1) \]
\[ \quad xi4 = xi(mmm4, 1) \]
\[ \% \]
\[ \quad yi1 = yi(mmm1, 1) \]
\[ \quad yi2 = yi(mmm2, 1) \]
\[ \quad yi3 = yi(mmm3, 1) \]
\[ \quad yi4 = yi(mmm4, 1) \]
\[ \% \]
\[ \quad xi(mmm5, 1) = (2*xi1 + xi2)/3; xi(mmm6, 1) = (xi1 + 2*xi2)/3; \]
\[ \quad xi(mmm7, 1) = (2*xi2 + xi3)/3; xi(mmm8, 1) = (xi2 + 2*xi3)/3; \]
\[ xi(mmm9,1) = \frac{2 \times xi3 + xi4}{3}; xi(mmm10,1) = \frac{xi3 + 2 \times xi4}{3}; \\
\]
\[ xi(mmm11,1) = \frac{2 \times xi4 + xi1}{3}; xi(mmm12,1) = \frac{xi4 + 2 \times xi1}{3}; \]
\[ yi(mmm5,1) = \frac{2 \times yi1 + yi2}{3}; yi(mmm6,1) = \frac{yi1 + 2 \times yi2}{3}; \\
\]
\[ yi(mmm7,1) = \frac{2 \times yi2 + yi3}{3}; yi(mmm8,1) = \frac{yi2 + 2 \times yi3}{3}; \]
\[ yi(mmm9,1) = \frac{2 \times yi3 + yi4}{3}; yi(mmm10,1) = \frac{yi3 + 2 \times yi4}{3}; \]
\[ yi(mmm11,1) = \frac{2 \times yi4 + yi1}{3}; yi(mmm12,1) = \frac{yi4 + 2 \times yi1}{3}; \]
\[ yi(mmm13,1) = \frac{4 \times yi1 + 2 \times yi2 + yi3 + 2 \times yi4}{9}; yi(mmm14,1) = \frac{2 \times yi1 + 4 \times yi2 + 2 \times yi3 + yi4}{9}; \]
\[ yi(mmm15,1) = \frac{yi1 + 2 \times yi2 + 4 \times yi3 + 2 \times yi4}{9}; yi(mmm16,1) = \frac{2 \times yi1 + yi2 + 2 \times yi3 + 4 \times yi4}{9}; \]

\end{verbatim}

\begin{verbatim}
end
end %for nel
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)

function [coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order(n1,n2,n3,nmax,numtri,n,mesh)
%n1=node number at(0,0) for a chosen triangle
%n2=node number at(1,0) for a chosen triangle
%n3=node number at(0,1) for a chosen 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
%PARVIZ MOIN EXAMPLE
syms U V W xi yi
syms x y

\[ switch mesh \]
\begin{verbatim}
case 1 %domain with seven triangles(8-nodes)
\end{verbatim}
\[ x=sym([1/2;1/2;1; 1;1/2;0; -1/2; -1/2]); \]
\[ y=sym([1/2; 0;0;1;1;1/2;1/2; 1/2]); \]
\[ end \]
\[ case 2 %square domain with eight triangles(9-nodes) \]
\[ x=sym([1/2; 1/2; 1; 1;1/2;0; -1/2; -1/2]); \]
\[ y=sym([1/2; 0;0;1;1;1/2;1/2; 1/2]); \]
\[ end \]
\[ case 3 %square domain with four triangles(5-nodes) \]
\[ x=sym([1/2;0;1;1;0]); \]
\[ y=sym([1/2;0;0;1;1]); \]
\[ end \]
\[ case 4 %square domain with eight triangles(9-nodes) \]
\[ x=sym([0; 0; 1/2;1/2;1/2; 0;-1/2;-1/2;-1/2]); \]
\[ y=sym([-1/2;1/2; 0;1/2;1/2; -1/2]); \]
\[ end \]
\end{verbatim}

\[ [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_order(n1,n2,n3,nmax,numtri,n); \]
\[ [U,V,W]=generate_area_coordinate_over_the_standard_triangle(n); \]

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

\% ss2='number of special convex quadrilaterals elements&nodes per element ='
[nelnnel]=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(nnel)])

xi(1:nnode,1)=zeros(nnode,1);yi(1:nnode,1)=zeros(nnode,1);
nitri=max1;
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)
\[
\begin{align*}
m_{9} &= \text{nodes}(m,9) \\
m_{10} &= \text{nodes}(m,10) \\
m_{11} &= \text{nodes}(m,11) \\
m_{12} &= \text{nodes}(m,12) \\
x_{1} &= x(m,1,1) \\
x_{2} &= x(m,2,1) \\
x_{3} &= x(m,3,1) \\
x_{4} &= x(m,4,1) \\
\% \\
y_{1} &= y(m,1,1) \\
y_{2} &= y(m,2,1) \\
y_{3} &= y(m,3,1) \\
y_{4} &= y(m,4,1) \\
\% \\
x(m,5,1) &= (2*x_{1}+x_{2})/3; \\
x(m,6,1) &= (x_{1}+2*x_{2})/3; \\
x(m,7,1) &= (2*x_{3}+x_{4})/3; \\
x(m,8,1) &= (x_{3}+2*x_{4})/3; \\
x(m,9,1) &= (2*x_{3}+x_{4})/3; \\
x(m,10,1) &= (x_{3}+2*x_{4})/3; \\
x(m,11,1) &= (2*x_{i}+x_{i+1})/3; \\
x(m,12,1) &= (x_{i}+2*x_{i+1})/3; \\
y(m,5,1) &= (2*y_{1}+y_{2})/3; \\
y(m,6,1) &= (y_{1}+2*y_{2})/3; \\
y(m,7,1) &= (2*y_{3}+y_{4})/3; \\
y(m,8,1) &= (y_{3}+2*y_{4})/3; \\
y(m,9,1) &= (2*y_{3}+y_{4})/3; \\
y(m,10,1) &= (y_{3}+2*y_{4})/3; \\
y(m,11,1) &= (2*y_{i}+y_{i+1})/3; \\
y(m,12,1) &= (y_{i}+2*y_{i+1})/3; \\
\end{align*}
\]

end

\% for nel
\% [x(18,1) y(18,1)]

N=(1:nnode)';
[N x y] %
coord(:,1)=(x(:,1));
coord(:,2)=(y(:,1));
gocoord(:,1)=double(x(:,1));
gocoord(:,2)=double(y(:,1));
%disp(gcoord)

function [ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_orderLAGR.m

% 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
kk=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
for iii=1:(n-2)
  for jjj=1:(n-1)-iii
    kkkkk=kkkk+1;
    ui(kkkkk,1)=sym(jjj/n);
    vi(kkkkk,1)=sym(iii/n);
    wi(kkkkk,1)=sym(1-ui(kkkkk,1)-vi(kkkkk,1));
  end
end
end

if (n-2)>0
  num=(1:6)';
else
  num=(1:kkkkk)';
end

%disp([ui'])
%disp([vi'])
%disp([wi'])
%length(ui)
%length(vi)
%length(wi)
%disp('first')
%disp([num ui vi wi])
[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_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('second')
%disp([num ui vi wi])
qqq=qq;
for ppp=1:3*nc
  qq1=nodes(ppp,1);
  qq2=nodes(ppp,2);
  qq3=nodes(ppp,3);
  qq4=nodes(ppp,4);
  %midside nodes-1.2
 qqq=nodes(ppp,5);qqqq=nodes(ppp,6);
  ui(qqqq,1)=(2*ui(qq1,1)+ui(qq2,1))/3;
  vi(qqqq,1)=(2*vi(qq1,1)+vi(qq2,1))/3;
  wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);
  ui(qqqq,1)=(ui(qq1,1)+2*ui(qq2,1))/3;
  vi(qqqq,1)=(vi(qq1,1)+2*vi(qq2,1))/3;
  wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);

%midside nodes-2.3
qqq=nodes(ppp,7);qqqq=nodes(ppp,8);
  ui(qqqq,1)=(2*ui(qq2,1)+ui(qq3,1))/3;
  vi(qqqq,1)=(2*vi(qq2,1)+vi(qq3,1))/3;
  wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);
ui(qqqq,1)=(ui(qq2,1)+2*ui(qq3,1))/3;
v(qqqq,1)=(vi(qq2,1)+2*vi(qq3,1))/3;
wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);

%midside nodes-3,4
qqq=nodes(ppp,9);qqqq=nodes(ppp,10);
ui(qqqq,1)=(2*ui(qq3,1)+ui(qq4,1))/3;
v(qqqq,1)=(2*vi(qq3,1)+vi(qq4,1))/3;
wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);
ui(qqqq,1)=(ui(qq3,1)+2*ui(qq4,1))/3;
v(qqqq,1)=(vi(qq3,1)+2*vi(qq4,1))/3;
wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);

%midside nodes-4,1
qqq=nodes(ppp,11);qqqq=nodes(ppp,12);
ui(qqq,1)=(2*ui(qq3,1)+ui(qq4,1))/3;
v(qqq,1)=(2*vi(qq3,1)+vi(qq4,1))/3;
wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
ui(qqqq,1)=(ui(qq4,1)+2*ui(qq1,1))/3;
v(qqqq,1)=(vi(qq4,1)+2*vi(qq1,1))/3;
wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);

%interior nodes
q1=nodes(ppp,13);
q2=nodes(ppp,14);
q3=nodes(ppp,15);
q4=nodes(ppp,16);

ui(q1,1)=(4*ui(qq1,1)+2*ui(qq2,1)+ui(qq3,1)+2*ui(qq4,1))/9;
v(q1,1)=(4*vi(qq1,1)+2*vi(qq2,1)+vi(qq3,1)+2*vi(qq4,1))/9;
wi(q1,1)=1-ui(q1,1)-vi(q1,1);

ui(q2,1)=(2*ui(qq1,1)+4*ui(qq2,1)+2*ui(qq3,1)+ui(qq4,1))/9;
v(q2,1)=(2*vi(qq1,1)+4*vi(qq2,1)+2*vi(qq3,1)+vi(qq4,1))/9;
wi(q2,1)=1-ui(q2,1)-vi(q2,1);

ui(q3,1)=(ui(qq1,1)+2*ui(qq2,1)+4*ui(qq3,1)+2*ui(qq4,1))/9;
v(q3,1)=(vi(qq1,1)+2*vi(qq2,1)+4*vi(qq3,1)+2*vi(qq4,1))/9;
wi(q3,1)=1-ui(q3,1)-vi(q3,1);

ui(q4,1)=(2*ui(qq1,1)+ui(qq2,1)+2*ui(qq3,1)+4*ui(qq4,1))/9;
v(q4,1)=(2*vi(qq1,1)+vi(qq2,1)+2*vi(qq3,1)+4*vi(qq4,1))/9;
wi(q4,1)=1-ui(q4,1)-vi(q4,1);
end

maxnode=max(max(nodes(1:3*nc,1:16)));
num=(1:maxnode)';
%disp('maximum value of node number=',num2str(maxnode))
%disp(' node    ui     vi     wi')
%disp([num ui vi wi])
12coordinate_special_quadrilaterals_in_stdtriangle_3rd_order.m
function [ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_order(n)

% if 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;
end
end
ui(kkk,1)=sym((n-ii)/n);
vi(kkk,1)=sym(1-(n-ii)/n);
wi(kkk,1)=0;
end;
nkkk=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
    kkkk=kkkk;
    for iiii=1:(n-2)
        for jjjj=1:(n-1)-iiii
            kkkk=kkkk+1;
            ui(kkkk,1)=sym(jjjj/n);
            vi(kkkk,1)=sym(iiii/n);
            wi(kkkk,1)=sym(1-ui(kkkk,1)-vi(kkkk,1));
        end
    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('first')
    %disp([num ui vi wi])
end
[end, nodetel, nodes, nnode]=nodaladdresses_special_convex_quadrilaterals_3rd_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('second')
%disp([num ui vi wi])
qqq=qq;
for ppp=1:3*nc
    q1=nodes(ppp,1);
    q2=nodes(ppp,2);
    q3=nodes(ppp,3);
    q4=nodes(ppp,4);
    %midside nodes -1.2
```matlab
%midside nodes-2,3
qq=qnodes(ppp,7);qq=qnodes(ppp,8);
vi(qqqq,1)=(vi(qq1,1)+2*vi(qq2,1))/3;
ui(qqqq,1)=(ui(qq1,1)+2*ui(qq2,1))/3;
vi(qqq,1)=(2*vi(qq1,1)+vi(qq2,1))/3;
ui(qqq,1)=(2*ui(qq1,1)+ui(qq2,1))/3;
wi(qqq,1)=1

%midside nodes-3,4
qq=qnodes(ppp,9);qq=qnodes(ppp,10);
vi(qqqq,1)=(vi(qq2,1)+2*vi(qq3,1))/3;
ui(qqqq,1)=(ui(qq2,1)+2*ui(qq3,1))/3;
vi(qqq,1)=(2*vi(qq2,1)+vi(qq3,1))/3;
ui(qqq,1)=(2*ui(qq2,1)+ui(qq3,1))/3;
wi(qqq,1)=1

%midside nodes-4,1
qq=qnodes(ppp,11);qq=qnodes(ppp,12);
vi(qqqq,1)=(vi(qq3,1)+2*vi(qq4,1))/3;
ui(qqqq,1)=(ui(qq3,1)+2*ui(qq4,1))/3;
vi(qqq,1)=(2*vi(qq3,1)+vi(qq4,1))/3;
ui(qqq,1)=(2*ui(qq3,1)+ui(qq4,1))/3;
wi(qqq,1)=1
end
maxnode=max(max(nodes(1:3*nc,1:12)));
num=(1:maxnode)'

%disp(' node    ui     vi     wi')
%disp(maxnode)
%disp(num ui vi wi)

[13] integral_valuesof_localderivative_products.m
function [intJdn]=integral_valuesof_localderivative_products(nnel)
switch nnel
    case 12
    %integral values of local derivative products
```

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482 Page 23444
intJdndn(21:22,1:32)=[intJdn11dn1uvrs intJdn11dn2uvrs intJdn11dn3uvrs intJdn11dn4uvrs intJdn11dn5uvrs intJdn11dn6uvrs intJdn11dn7uvrs intJdn11dn8uvrs intJdn11dn9uvrs intJdn11dn10uvrs intJdn11dn11uvrs intJdn11dn12uvrs intJdn11dn13uvrs intJdn11dn14uvrs intJdn11dn15uvrs intJdn11dn16uvrs];
intJdndn(23:24,1:32)=[intJdn12dn1uvrs intJdn12dn2uvrs intJdn12dn3uvrs intJdn12dn4uvrs intJdn12dn5uvrs intJdn12dn6uvrs intJdn12dn7uvrs intJdn12dn8uvrs intJdn12dn9uvrs intJdn12dn10uvrs intJdn12dn11uvrs intJdn12dn12uvrs intJdn12dn13uvrs intJdn12dn14uvrs intJdn12dn15uvrs intJdn12dn16uvrs];
intJdndn(25:26,1:32)=[intJdn13dn1uvrs intJdn13dn2uvrs intJdn13dn3uvrs intJdn13dn4uvrs intJdn13dn5uvrs intJdn13dn6uvrs intJdn13dn7uvrs intJdn13dn8uvrs intJdn13dn9uvrs intJdn13dn10uvrs intJdn13dn11uvrs intJdn13dn12uvrs intJdn13dn13uvrs intJdn13dn14uvrs intJdn13dn15uvrs intJdn13dn16uvrs];
intJdndn(27:28,1:32)=[intJdn14dn1uvrs intJdn14dn2uvrs intJdn14dn3uvrs intJdn14dn4uvrs intJdn14dn5uvrs intJdn14dn6uvrs intJdn14dn7uvrs intJdn14dn8uvrs intJdn14dn9uvrs intJdn14dn10uvrs intJdn14dn11uvrs intJdn14dn12uvrs intJdn14dn13uvrs intJdn14dn14uvrs intJdn14dn15uvrs intJdn14dn16uvrs];
intJdndn(29:30,1:32)=[intJdn15dn1uvrs intJdn15dn2uvrs intJdn15dn3uvrs intJdn15dn4uvrs intJdn15dn5uvrs intJdn15dn6uvrs intJdn15dn7uvrs intJdn15dn8uvrs intJdn15dn9uvrs intJdn15dn10uvrs intJdn15dn11uvrs intJdn15dn12uvrs intJdn15dn13uvrs intJdn15dn14uvrs intJdn15dn15uvrs intJdn15dn16uvrs];
intJdndn(31:32,1:32)=[intJdn16dn1uvrs intJdn16dn2uvrs intJdn16dn3uvrs intJdn16dn4uvrs intJdn16dn5uvrs intJdn16dn6uvrs intJdn16dn7uvrs intJdn16dn8uvrs intJdn16dn9uvrs intJdn16dn10uvrs intJdn16dn11uvrs intJdn16dn12uvrs intJdn16dn13uvrs intJdn16dn14uvrs intJdn16dn15uvrs intJdn16dn16uvrs];
intJdndn=double(intJdndn);
end

[14] nodaladdresses_special_convex_quadrilaterals_trial_3rd_order.m

function [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_order(n1,n2,n3,nmax,numtri,n)
% n1=node number at(0,0) for a chosen triangle
% n2=node number at(1,0) for a chosen triangle
% n3=node number at(0,1) for a chosen 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 points of the segment to the interior point(e.g the centroid) of the polygon
% PARVIZ MOIN EXAMPLE
% [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_order([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
% % sym
% % sym x
% if n even
% n1=nmmax-1;
% for itri=1:ntri
% elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
% elm(1,1)=n1(itri,1)
% elm(n1+1,1)=n2(itri,1)
% elm(n1+n+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)
% edge1n2(1:n+1,1)=elm(1:n+1,1)
% end
% if itri==1
% elm(1:n+1,1)=edge1n3(1:n+1,1,1);
% end
% if itri==1
% if itri==1
% lmax=nmax+3*(n-1);
% end
% if itri==1
% if (itri>1)&&(itri<ntri)
% lmax=nmax+2*(n-1);
% end
% if (itri>1)&&(itri<ntri)
% nmax=nmax;
% if itri==1

H. T. Rathod, IJECS Volume 7 Issue 1 January 2018 Page No. 23329-23482 Page 23458
\begin{verbatim}
mmax=max(max(edgen1n2(1:n+1,1)))
end
% if itri==1
disp('right edge nodes')
ni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
    hh=hh+1;
    ni=nni+(n-i);
    Elm(ni,1)=(mmax+1)+i;
    qq(hh,1)=(mmax+1)+i;
end
qq(n+1,1)=n3(itri,1);
edgen2n3(1:n+1,1,1)=qq;
if itri<nitri
disp('left edge nodes')
ni=1;gg=1;pp(1,1)=n1(itri,1);
for i=0:(n-2)
    gg=gg+1;
    ni=nni+(n-i)+1;
    Elm(ni,1)=lmax-i;
    pp(gg,1)=lmax-i;
end
pp(n+1,1)=n3(itri,1);
edgen1n3(1:n+1,1,1)=pp
end
% if itri<nitri
if itri==nitri
disp('left edge nodes')
ni=1;gg=1;pp(1,1)=n1(itri,1);
for i=0:(n-2)
    gg=gg+1;
    ni=nni+(n-i)+1;
    Elm(ni,1)=edgen1n2(gg,1);
end
% pp(n+1,1)=n3(itri,1);
% edgen1n3(1:n+1,1,1)=pp
end
% if itri==nitri
if itri==nitri
    lmax=max(max(edgen2n3(1:n+1,1,1)));
end
% if itri==nitri
disp('interior nodes')
nni=1;jj=0;
for i=0:(n-3)
ni=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=jj+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{verbatim}
%end
%[row_nodes]
rr=row_nodes;
rr
rr(:,itrri)=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+1,j);
eln(ne,3)=rr(i,j+1);
eln(ne,4)=rr(i+1,j+1);
eln(ne,5)=rr(i,j+2);
eln(ne,6)=rr(i+1,j+2);
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+1);
eln(ne,3)=rr(i,jj+2);
eln(ne,4)=rr(i+1,jj+2);
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+(itrri-1)*numtri:ne
nnd=nnd+1;
eln(kkk,7)=nnd;
end
end
if n==2
for kkk=itrri:ne
nnd=nnd+1;
eln(kkk,7)=nnd;
end
end
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd

end%itrri
%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
        trisect(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((trisect(mmm1,mmm2)==0)&(trisect(mmm2,mmm1)==0))
        nd=nd+1;
        trisect(mmm1,mmm2)=nd;
        nd=nd+1;
        trisect(mmm2,mmm1)=nd;
    end
    %midpoint side-2 of 4-node special quadrilateral
    if((trisect(mmm2,mmm3)==0)&(trisect(mmm3,mmm2)==0))
        nd=nd+1;
        trisect(mmm2,mmm3)=nd;
        nd=nd+1;
        trisect(mmm3,mmm2)=nd;
    end
    %midpoint side-3 of 4-node special quadrilateral
    if((trisect(mmm3,mmm4)==0)&(trisect(mmm4,mmm3)==0))
        nd=nd+1;
        trisect(mmm3,mmm4)=nd;
        nd=nd+1;
        trisect(mmm4,mmm3)=nd;
    end
    %midpoint side-4 of 4-node special quadrilateral
    if((trisect(mmm4,mmm1)==0)&(trisect(mmm1,mmm4)==0))
        nd=nd+1;
        trisect(mmm4,mmm1)=nd;
        nd=nd+1;
        trisect(mmm1,mmm4)=nd;
    end
end

nodes(mmm,5)=trisect(mmm1,mmm2);
nodes(mmm,6)=trisect(mmm2,mmm1);
% nodes(mmm,7)=trisect(mmm2,mmm3);
% nodes(mmm,8)=trisect(mmm3,mmm2);
% nodes(mmm,9)=trisect(mmm3,mmm4);
% nodes(mmm,10)=trisect(mmm4,mmm3);
% nodes(mmm,11)=trisect(mmm4,mmm1);
% nodes(mmm,12)=trisect(mmm1,mmm4);

end %for

%---------------------------------------------------------

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)])

[15]generate_area_coordinate_over_the_standard_triangle.m

function [U,V,W]=generate_area_coordinate_over_the_standard_triangle(n)
syms u v w
kk=0;
for j=1:n+1
    for i=1:(n+1)-(j-1)
        kk=kk+1;
        u(i,j)=(i-1)/n;
        v(i,j)=(j-1)/n;
        w(i,j)=1-u(i,j)-v(i,j);
        U(kk,1)=u(i,j);
        V(kk,1)=v(i,j);
        W(kk,1)=w(i,j);
    end
end

[16]nodaladdresses_special_convex_quadrilaterals_trial_3rd_orderLG.m

function [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_orderLG(n1,n2,n3,nmax,numtri,n)
% n1=node number at(0,0) for a chosen triangle
% n2=node number at(1,0) for a chosen triangle
% n3=node number at(0,1) for a chosen 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 points of the segment to the interior point(e.g:the centroid) of the polygon
% PARYIZ MOIN EXAMPLE
% [eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_orderLG([1;1;1;1],[2;3;4;5],[3;4;5;2],[5,1,2])
% symms 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
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)&(itri<nitri)
  lmax=nmax+2*(n-1);
end
if itri==nitri
  lmax=max(max(edgen1n2(1:n+1,1))
end
if 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;
end
%if itri<nitri
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);
%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]
nr=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
    elm(ne,1)=rr(i,j);
    elm(ne,2)=rr(i,j+2);
    elm(ne,3)=rr(i+2,j);
    elm(ne,4)=rr(i+1,j+1);
    elm(ne,5)=rr(i+1,j+1);
    elm(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
    elm(ne,1)=rr(i+2,jj+2);
    elm(ne,2)=rr(i+2,jj);
end
end
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
if(N-2)>0
end

%jj
end

%if(N
if(n>3)
for kkk=1+(itri-1)*numtri:ne
    nnd=max(max(eln));
    if(n>3)
        for kkk=itri: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
    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
        end
    end
end
%---------------------------------------------------------
for inum=1:nnd
    for jnum=1:nnd
        trisect(inum,jnum)=0;
    end
end
nd=nnd;
for mmm=1:mm
    mmn1=nodes(mmm,1);
    mmn2=nodes(mmm,2);
    mmn3=nodes(mmm,3);
    mmn4=nodes(mmm,4);
    %midpoint side-1 of 4-node special quadrilateral
    if((trisect(mmn1,mmn2)==0)&&(trisect(mmn2,mmn1)==0))
        nd=nd+1;
        trisect(mmn1,mmn2)=nd;
        nd=nd+1;
        trisect(mmn2,mmn1)=nd;
    end
    %-----------------------------------------------
if((trisect(mmm2,mmm3)==0)&(trisect(mmm3,mmm2)==0))
    nd=nd+1;
    trisect(mmm2,mmm3)=nd;
    nd=nd+1;
    trisect(mmm3,mmm2)=nd;
end

if((trisect(mmm3,mmm4)==0)&(trisect(mmm4,mmm3)==0))
    nd=nd+1;
    trisect(mmm3,mmm4)=nd;
    nd=nd+1;
    trisect(mmm4,mmm3)=nd;
end

if((trisect(mmm4,mmm1)==0)&(trisect(mmm1,mmm4)==0))
    nd=nd+1;
    trisect(mmm4,mmm1)=nd;
    nd=nd+1;
    trisect(mmm1,mmm4)=nd;
end

nodes(mmm,5)=trisect(mmm1,mmm2);
nodes(mmm,6)=trisect(mmm2,mmm1);

nodes(mmm,7)=trisect(mmm2,mmm3);
nodes(mmm,8)=trisect(mmm3,mmm2);

nodes(mmm,9)=trisect(mmm3,mmm4);
nodes(mmm,10)=trisect(mmm4,mmm3);

nodes(mmm,11)=trisect(mmm4,mmm1);
nodes(mmm,12)=trisect(mmm1,mmm4);

%----------------------------------------------

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)])
%nel=4;  
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order(n1,n2,n3,nmax,numtri,ndiv,mesh)  
nel=12;  
ndof=1;  
%nc=(ndiv/2)^2;  
%nnode=(ndiv+1)*(ndiv+2)/2+nc;  
%nel=3*nc;  
sdof=nnode*ndof;  
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));  
disp([nel nnode nnel ndof])  
format long g  
for i=1:nel  
N(i,1)=i;  
end  
for i=1:nel  
NN(i,1)=i;  
end  
switch mesh  
  case 1  
    %boundary conditions-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  
  end  
  case 2  
    %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  
  end  
  case 3  
    %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  
  end  
  case 4  
    %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  
  end  
  case 5  
    %boundary conditions-5  
    for nn=1:nnode  
      xnn=gcoord(nn,1);ynn=gcoord(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  
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=coord(nn,1);ynn=gcoord(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=gcoord(nn,1);ynn=coord(nn,2);
    if (xnn==1)&((ynn>=0)&(ynn<=1))
      nnn=nnn+1;
      bcdof(nnn,1)=nn;
      bcval(nnn,1)=0;
    end
  end

  bcdof
  nnn=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
    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])

  %integral values of local derivative products
\[ \text{intJdn} = \text{integral\_values\_of\_local\_derivative\_products(nel)}; \]
\[ \text{for}\ iel = 1:nel \]
\[ \text{index} = \text{zeros(nel*ndof,1)}; \]
\[ \begin{align*}
X &= xx(iel,1:3); \\
Y &= yy(iel,1:3); \\
\text{% disp([X Y])} \\
xe &= X(1,1); \\
xb &= X(1,2); \\
xc &= X(1,3); \\
ye &= Y(1,1); \\
yb &= Y(1,2); \\
cy &= Y(1,3); \\
\text{bta} &= yb - yc; \\
\text{btb} &= yc - ya; \\
gma &= xc - xb; \\
gmb &= xa - xc; \\
\text{delabc} &= gmb * bta - gma * btb; \\
G &= \text{[bta gma; btb gmb]}; \\
\text{GT} &= \text{[bta gma; btb gmb]}; \\
Q &= GT^*G; \\
\text{sk(1:12,1:12) = (zeros(12,12));} \\
\text{for i = 1:12} \\
\text{for j = i:12} \\
\text{sk(i,j) = (delabc * sum(sum(Q.*(intJdn(i-1:2*i,2*j-1:2*j))));} \\
\text{end} \\
\text{end} \\
\text{% f = [5/144;1/24;7/144;1/24]*(2*delabc);} \\
xe &= (xa + xb + xc) / 3; \\
x2 &= (xa + xc) / 2; \\
x3 &= xa; \\
x4 &= (xa + xb) / 2; \\
\text{% ye(1,1) = (ya + yb + yc) / 3; } \\
\text{ye(2,1) = (ya + yc) / 2; } \\
\text{ye(3,1) = ya; } \\
\text{ye(4,1) = (ya + yb) / 2;} \\
\text{% [sp, wt] = glsampleptsweights(ng);} \\
\text{x1 = xe(1,1): xe2 = xe(2,1): xe3 = xe(3,1): xe4 = xe(4,1); } \\
\text{ye1 = ye(1,1): ye2 = ye(2,1): ye3 = ye(3,1): ye4 = ye(4,1); } \\
\text{f(1:12,1) = zeros(12,1); } \\
\text{for i = 1:ng} \\
\text{si = sp(i,1): wi = wt(i,1); } \\
\text{for j = 1:ng} \\
\text{sj = sp(j,1): wj = wt(j,1);} \\
\text{n1ij} &= ((1-si) * (1-sj) * (-10 + 9 * (si^2 + sj^2))) / 32; \\
\text{n2ij} &= ((1+si) * (1-sj) * (-10 + 9 * (si^2 + sj^2))) / 32; \\
\text{n3ij} &= ((1+si) * (1+sj) * (-10 + 9 * (si^2 + sj^2))) / 32; \\
\text{n4ij} &= ((1-si) * (1+sj) * (-10 + 9 * (si^2 + sj^2))) / 32; \\
\text{n5ij} &= (9/32) * ((1-si) * (1-si^2) * (1-3*si)); \\
\text{n6ij} &= (9/32) * ((1-si) * (1-si^2) * (1+3*si)); \\
\text{n7ij} &= (9/32) * ((1+si) * (1-sj^2) * (1-3*sj)); \\
\text{n8ij} &= (9/32) * ((1+si) * (1-sj^2) * (1+3*sj)); \\
\text{n9ij} &= (9/32) * ((1+sj) * (1-si^2) * (1-3*si)); \\
\text{n10ij} &= (9/32) * ((1+sj) * (1-si^2) * (1+3*si)); \\
\text{n11ij} &= (9/32) * ((1-si) * (1-sj^2) * (1-3*sj)); \\
\text{n12ij} &= (9/32) * ((1-si) * (1-sj^2) * (1+3*sj)); \\
\text{\%} \\
\text{N1ij} &= (((1-si) * (1-sj))/4; } \\
\text{N2ij} &= (((1+si) * (1-sj))/4; } \\
\text{N3ij} &= (((1+si) * (1+sj))/4; } \\
\text{N4ij} &= (((1-si) * (1+sj))/4; } \\
\text{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;
f9i=n9ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f10i=n10ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f11i=n11ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f12i=n12ij*(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;
f(9,1)=f(9,1)+f9i*wi*wj;
f(10,1)=f(10,1)+f10i*wi*wj;
f(11,1)=f(11,1)+f11i*wi*wj;
f(12,1)=f(12,1)+f12i*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);
end
end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
solve the equations
phi=ssff;  
for I=1:nnode
   NN(I,1)=I;
end

disp(' number of nodes,elements & nodes per element')
[nnel nel nnel ndof]
disp(' fe-computed values anlytical(theoretical)-values')
disp([NN phi xi])
disp(' number of nodes,elements & nodes per element')
[nnel nel nnel ndof]

nodes
gcoord
[x,y]=meshgrid(0:0.05:1,0:0.05:1);

for i=1:21
   for j=1:21
      for iel=1:nel
         XX=xx(iel,1:3);
         YY=yy(iel,1:3);
         [X Y]=convexquadrilateral_coordinates(u,v,X,Y);
         p=(aLPa+bta*X+gma*Y)/delabc;
         q=(aLPb+btb*X+gmb*Y)/delabc;
         t0=[0.5;0.5];

         [t,iter] = newtonmethod4spquadrilateralspquadrilateral(
               t0,p,q,parameqnspqd,'paramdetJspqd','paraminvJspqd')
         r=t(1,1);
         r=s(2,1);
         shn1=((1-r)*(1-s)*(-10+9*(r^2+s^2)))/32;
         shn2=((1+r)*(1-s)*(-10+9*(r^2+s^2)))/32;
         shn3=((1+r)*(1+s)*(-10+9*(r^2+s^2)))/32;
         shn4=((1-r)*(1+s)*(-10+9*(r^2+s^2)))/32;
         shn5=((9/32)*((1-s)*(1-r^2)*(1-3*r)));
shn6=(9/32)*((1-s)*(1-r^2)*(1+3*r));
shn7=(9/32)*((1+r)*(1-s^2)*(1-3*s));
shn8=(9/32)*((1+r)*(1-s^2)*(1+3*r));
shn9=(9/32)*((1+s)*(1-r^2)*(1+3*r));
shn10=(9/32)*((1+s)*(1-r^2)*(1-3*r));
shn11=(9/32)*((1-r)*(1-s^2)*(1+3*s));
shn12=(9/32)*((1-r)*(1-s^2)*(1-3*s));

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)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1);

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

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

switch mesh
  case 1
    clc
    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:.05:1,0:0.05:1)
y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
    contour(x,y,PHI,40,'r-')
    %[c,h]=contour(x,y,PHI,'r:')
    xlabel('X-axis');
    ylabel('Y-axis');
    %clabel(c,h);
    axis square
    st1='Contour level curves for';
    st2='FEM solution of';
    st3=' Twelve Noded';
    st4=' Special Quadrilateral';
    st5=' Elements';
title([st1,st2,st3,st4,st5])
ss1=('MESH HAS ')
ss2=num2str(nnnode)
ss3=(' NODES ')
ss4=(' AND ')
ss5=num2str(nnode)
ss6=(' ELEMENTS ')
text(0.25,-0.8,[ss1 ss2 ss3 ss4 ss5 ss6])
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:.05:1,0:0.05:1)
y(y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2)=NaN;
contour(x,y,z,40,'g-')

%[c,h]=contour(x,y,z,'g-')
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
title('Contour level curves for exact solution: sin(pi*x)sin(pi*y)')
mm=0;
for i=1:21
    for j=1:21
        mm=mm+1;
        femsoln(mm,1)=PHI(i,j);
        exactsoln(mm,1)=z(i,j);
    end
end
case 2
clc
hold off
clf
figure(1)
x=[0.0 1.0 1.0 0.0];
y=[0.0 0.0 1.0 1.0];
patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.05:1,0:0.05:1)
contour(x,y,PHI,40,'r-')
%[c,h]=contour(x,y,PHI,'r:')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square
st1='Contour level curves for ';
st2='FEM and Exact solutions using ';
st3='Twelve Noded ';
st4='Special Quadrilateral';
st5='Elements on a SQUARE'
title([st1,st2,st3,st4,st5])
sst1=('MESH HAS ')
sst2=num2str(nnode)
sst3=' NODES ';
sst4=' AND ';
sst5=num2str(nel)
sst6=' ELEMENTS')
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])

figure(2)
x=[0.0 1.0 1.0 0.0];
y=[0.0 0.0 1.0 1.0];
patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.05:1,0:0.05:1)
contour(x,y,z,40,'g-')
%[c,h]=contour(x,y,z,'g-')
%xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
axis square
%text(6,.98,'...(red)FEM and--(green)EXACT')
%legend('FEM','EXACT')
title('Contour level curves for exact solution: sin(pi*x)sin(pi*y)')
mm=0;
for i=1:21
for j=1:21
    mm=mm+1;
    femsoln(mm,1)=PHI(i,j);
    exactsln(mm,1)=z(i,j);
end
end
end
end
%switch mesh
[femsoln exactsln]

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
[1 phi(1,1) xi(1,1)]
for I=1:nnode
    NN(I,1)=I;
    phi_xi(I,1)=phi(I,1)-xi(I,1);
end
MAXPHI_XI=max(abs(phi_xi))

[18]D2PoissonEquationQ16MoinEx_MeshgridContourNew.m
function[]=D2PoissonEquationQ16MoinEx_MeshgridContourNew(n1,n2,n3,nmax,numtri,ndiv,mesh)
%ndiv=2,4,6,8,......
% D2PoissonEquationQ16MoinEx_MeshgridContourNew([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])
% D2PoissonEquationQ16MoinEx_MeshgridContourNew([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])
clc
clf,figure(1)
clf,figure(2)
syms coord
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_orderLG(n1,n2,n3,nmax,numtri,ndiv,mesh)
nel=16;
nnode=1;
nnode=(ndiv+1)*(ndiv+2)/2+nc;
nel=3*nc;
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
disp([nel nnode nnel ndof])
format long g
for i=1:nel
    N(i,1)=i;
end
for i=1:nel
    NN(i,1)=i;
end
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
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
    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])
[intJdndn]=integral_valuesof_localderivative_products(nnel);

%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 gma;btb gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;
sk(1:16,1:16)=(zeros(16,16));
for i=1:16
    for j=i:16
        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;
xe1=x(1,1); xe2=x(2,1); xe3=x(3,1); xe4=x(4,1);
ye1=y(1,1); ye2=y(2,1); ye3=y(3,1); ye4=y(4,1);
f(1:16,1)=zeros(16,1);
for i=1:ng
    si=sp(i,1); wi=wt(i,1);
    h1i=-9*(si+1/3)*(si-1/3)*(si-1)/16;
    h2i=27*(si+1)*(si-1/3)*(si-1)/16;
    h3i=-27*(si+1)*(si+1/3)*(si-1)/16;
    h4i=9*(si+1)*(si+1/3)*(si-1/3)/16;
end
for j=1:ng
    sj=sp(j,1); wj=wt(j,1);
    h1j=-9*(sj+1/3)*(sj-1/3)*(sj-1)/16;
    h2j=27*(sj+1)*(sj-1/3)*(sj-1)/16;
    h3j=-27*(sj+1)*(sj+1/3)*(sj-1)/16;
    h4j=9*(sj+1)*(sj+1/3)*(sj-1/3)/16;
end

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;
f9i=n9ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f10i=n10ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f11i=n11ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f12i=n12ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f13i=n13ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f14i=n14ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f15i=n15ij*(2*pi^2)*sin(pi*xeij)*sin(pi*yeij)*(4+si+sj)/96;
f16i=n16ij*(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;
f(9,1)=f(9,1)+f9i*wi*wj;
f(10,1)=f(10,1)+f10i*wi*wj;
f(11,1)=f(11,1)+f11i*wi*wj;
f(12,1)=f(12,1)+f12i*wi*wj;
f(13,1)=f(13,1)+f13i*wi*wj;
f(14,1)=f(14,1)+f14i*wi*wj;
f(15,1)=f(15,1)+f15i*wi*wj;
f(16,1)=f(16,1)+f16i*wi*wj;
\[ f = (\text{delabc}) f; \]

\[ \text{edof} = \text{nnel} \times \text{ndof}; \]

\[ k = 0; \]

\[ \text{for } i = 1: \text{nnel} \]
\[ \quad \text{nd}(i,1) = \text{nodes}(i,1); \]
\[ \quad \text{start} = (\text{nd}(i,1) - 1) \times \text{ndof}; \]
\[ \text{for } j = 1: \text{ndof} \]
\[ \quad k = k + 1; \]
\[ \quad \text{index}(k,1) = \text{start} + j; \]
\[ \text{end} \]
\[ \text{end} \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]

\[ \% \]
xa=XX(1,1); 
xb=XX(1,2); 
xc=XX(1,3); 

ya=YY(1,1); 
yb=YY(1,2); 
cy=YY(1,3); 

aLPa=xb*yc-xc*yb; 
aLPb=xc*ya-xa*yc; 

bta=yb-yc; btb=yc-ya; 
gma=xc-xb; gmb=xa-xc; 
delabc=gmb*bta-gma*btb; 

% 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); 
nd9=nodes(iel,9);nd10=nodes(iel,10);nd11=nodes(iel,11);nd12=nodes(iel,12); 
nd13=nodes(iel,13);nd14=nodes(iel,14);nd15=nodes(iel,15);nd16=nodes(iel,16); 

% 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); 
    p=(aLPa+bta*X+gma*Y)/delabc; 
    q=(aLPb+btb*X+gmb*Y)/delabc; 
    t0=[0.5;0.5]; 
    r=t(1,1); 
    s=t(2,1); 
    h1r=-9*(r+1/3)*(r-1/3)*(r-1)/16; 
    h2r=27*(r+1)*(r-1/3)*(r-1)/16; 
    h3r=-27*(r+1)*(r+1/3)*(r-1)/16; 
    h4r=9*(r+1)*(r+1/3)*(r-1)/16; 
    h1s=-9*(s+1/3)*(s-1/3)*(s-1)/16; 
    h2s=27*(s+1)*(s-1/3)*(s-1)/16; 
    h3s=-27*(s+1)*(s+1/3)*(s-1)/16; 
    h4s=9*(s+1)*(s+1/3)*(s-1)/16; 

    % shn1=h1r*h1s; shn5=h2r*h1s; shn6=h3r*h1s; shn2=h4r*h1s; 
    shn12=h1r*h2s; shn13=h2r*h2s; shn14=h3r*h2s; shn7=h4r*h2s; 
    shn11=h1r*h3s; shn16=h2r*h3s; shn15=h3r*h3s; shn8=h4r*h3s; 
    shn4=h1r*h4s; shn10=h2r*h4s; shn9=h3r*h4s; shn3=h4r*h4s; 
    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)+shn9*phi(nd9,1)+shn10*phi(nd10,1)+shn11*phi(nd11,1)+shn12*phi(nd12,1)+shn13*phi(nd13,1)+shn14*phi 
(i(nd14,1))+shn15*phi(nd15,1)+shn16*phi(nd16,1); 

    break 
end 
end 

THE PROGRAM EXECUTION JUMPS TO HERE if (in==1) 
end 
end 

z=sin(pi*x).*sin(pi*y); 

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

switch mesh
case 1
  clc
  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:.05:1,0:0.05:1)
  y((y>1/2)&&(y<=1)&&(x>1/2)&&(x<=1)&&(x+y>3/2))=NaN;
  contour(x,y,PHI,40,'r-')
  %[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=' Sixteen Noded ';
  st4='Special Quadrilateral';
  st5=' Elements'
  title([st1 st2 st3 st4 st5])
  st1='(MESH HAS '
  st2=num2str(nnode)
  st3=' NODES' 
  st4=' AND ' 
  st5=num2str(nel)
  st6=' ELEMENTS)' 
  text(0.25,-0.08,[st1 st2 st3 st4 st5 st6])
  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:.05:1,0:0.05:1)
  y((y>1/2)&&(y<=1)&&(x>1/2)&&(x<=1)&&(x+y>3/2))=NaN;
  contour(x,y,z,40,'g-')
  %[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)')
  mm=0;
  for i=1:21
    for j=1:21
      mm=mm+1;
      femsoln(mm,1)=PHI(i,j);
      exactsoln(mm,1)=z(i,j);
    end
  end
end
case 2
  clc
  clf
  figure(1)
  x=[0.0 1.0 1.0 0.0];
  y=[0.0 0.0 1.0 1.0];
  patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.05:1,0:0.05:1)
contour(x,y,PHI,40,'r-')
%[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='Sixteen Noded ';
st4='Special Quadrilateral';
st5=' Elements'
title([st1 st2 st3 st4 st5])
sst1=('MESH HAS ')
sst2=num2str(nnode)
sst3=(' NODES')
sst4=(' AND ')
sst5=num2str(nel)
sst6=(' ELEMENTS')
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])

figure(2)
x=[0.0 1.0 1.0 0.0];
y=[0.0 0.0 1.0 1.0];
patch(x,y,'w')
hold on
[x,y]=meshgrid(0:.05:1,0:0.05:1)
contour(x,y,z,40,'g-')
%[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)')
mm=0;
for i=1:21
  for j=1:21
    mm=mm+1;
    femsoln(mm,1)=PHI(i,j);
    exactsoln(mm,1)=z(i,j);
  end
  end
end

switch mesh
  [femsoln exactsoltl]
  for I=1:nnode
    NN(I,1)=I;
    phi_xi(I,1)=phi(I,1)-xi(I,1);
  end
MAXPHI_XI=max(abs(phi_xi))
[NN phi]
  for I=1:nel
    NNN(I,1)=I;
  end
  [NNN nodes]
disp('number of nodes,elements & nodes per element')
  [nnode nel nel nel dof]
  [1 phi(1,1) xi(1,1)]

function [t,iter] = newtonmethod4spquadrileteral(t0,p,q,f,detJ,invJ)

  % Newton-Raphson method applied to a
  % system of linear equations f(x) = 0,
  % given the jacobian function J, with
% J = \frac{\Delta f_1, \Delta f_2, \ldots, \Delta f_n}{\Delta x_1, \Delta x_2, \ldots, \Delta x_n}
% x = [x_1, x_2, \ldots, x_n], f = [f_1, f_2, \ldots, f_n]
% x_0 is an initial guess of the solution

format short

N = 100; % define max. number of iterations
epsilon = 1e-10; % define tolerance
maxval = 10000.0; % define value for divergence
tt = (0); % load initial guess
while (N>0)
detJJ = feval(detJ, tt);
if abs(detJJ)<epsilon
    error(\'newtonm - Jacobian is singular - try new x0\');
    abort;
end;

tn = tt - feval(invJ, tt)*feval(f, tt)-[p; q];
if abs(feval(f, tn)-[p; q])<epsilon
    return;
end;

if abs(feval(f, tt)-[p; q])>maxval
    iter = 100-N;
    disp('[iterations = ',num2str(iter),']');
    error('Solution diverges');
    abort;
end;
N = N - 1;
tt = tn;
end;%while
error('No convergence after 100 iterations.); abort;

% end function

[20]parameqnspqd.m
function[f]=parameqnspqd(tt)
f = [-((tt(1,1) - 1)*(tt(2,1) + 5))/24; -((tt(1,1) + 5)*(tt(2,1) - 1))/24];

[21]paramdetJspqd.m
function[detJ]=paramdetJspqd(tt)
detJ =tt(1,1)/96 + tt(2,1)/96 + 1/24;

[22]paraminvJspqd.m
function[invJ]=paraminvJspqd(tt)
invJ =[-((tt(1,1)/24 + 5/24)/(tt(2,1)/96 + tt(1,1)/96 + 1/24); (tt(2,1)/24 - 1/24)/(tt(2,1)/96 + tt(1,1)/96 + 1/24);...
     (tt(1,1)/24 - 1/24)/(tt(2,1)/96 + tt(1,1)/96 + 1/24), -(tt(2,1)/24 + 5/24)/(tt(2,1)/96 + tt(1,1)/96 + 1/24)];

[23]glsampleptsweights.m: this program is regarding Gauss Legendre sampling points and weight coefficients and it is available in references[27-33]