www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 1 January 2018, Page No. 23329-23482
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i1.01

Finit 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[a*], Md.Shafiqul. Islam[b], Bharath Rathod[c], K. Sugantha Devi[d]

[a]Department of Mathematics, Central College Campus, Bangalore University, Bangalore -560001, Karnataka State, India.
[b]Department of Applied Mathematics, University of Dhaka,Dhaka-1000, Bangladesh
[c]Xavier Institute of Management and Entrepreneurship, Hosur Road, Electronic City Phase II, Bangalore, Karnataka 560034, Karnataka State, India.
[d]Department of Mathematics, Dr. T. Thimmaiah Institute of Technology, Oorgam Post, Kolar Gold Field, Kolar District, Karnataka state, Pin- 563120, India.

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 bivariates ξ and η 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 havee 12 boundary nodes at the nodal coordinates {(-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)} and the four interoior 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 (ξ,η). 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 $\Omega \subset \mathcal{R}^2$ 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 $\Omega \subset \mathcal{R}^2$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 problem in science, engineering and emerging technologies such as nanotechnology, biotechnology, aerospace, chemical. etc would be incapable of solution [1,2,3]. In FEM, various integrals are to be determined numerically in the evaluation of stiffness matrix, mass matrix, body force vector, etc. The algebraic integration needed to derive explicit finite element relations for second

order continuum mechanics problems generally defies our analytic skill and in most cases, it appears to be a prohibitive task. Hence, from a practical point of view, numerical integration scheme is not only necessary but very important as well. Among various numerical integration schemes, Gauss Legendre quadrature, which can evaluate exactly the $(2n-1)^{th}$ degree polynomial with 'n' Gaussian integration points, is mostly used in view of the accuracy and efficiency of calculation. However, the integrands of global derivative products in stiffness matrix computations of practical applications are not always simple polynomials but rational expressions which the Gaussian quadrature cannot evaluate exactly [7-15]. The integration points have to be increased in order 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 ($-1\leq \xi,\eta \leq 1$) in the natural space (ξ,η) to obtain the same expression of the Jacobian, namely $c(4+\xi+\eta)$ where c is some appropriate constant which depends on the geometric data for the triangle. 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 fnite elements to standard computational procedures [2]. A further historical development and present day concepts of finite element analysis are widely described in references [1, 3]. In this paper the well-known Laplace and Poisson equations will be examined by means of the finite element method applied to an appropriate 'mesh'. The class of physical situations in which we meet these equations is really broad. Let's recall such problems like heat conduction, seepage through porous media, irrotational flow of ideal fluids, distribution of electrical or magnetic potential, torsion of prismatic shafts, lubrication of pad bearings and others [4]. Therefore, in physics and engineering arises a need of some computational methods that allow 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 discrtisation is carried in two steps, We first discretise the linear polygon into a set of linear triangles in the Cartesian space (x,y) and these linear triangles are then mapped into a standard triangle in a local space (u,v). We further discretise the standard triangles into three linear quadrilaterals by joining the centroid to the midpoints of triangles in (u,v) space which are finally mapped into 2-square in the local (ξ,η) space. We then establish a derivative product relation between the linear convex quadrilaterals in the Cartesian space, (x,y) which are interior to an arbitrary triangle and the linear quadrilaterals in the local space (u,v) interior to the standard triangle. In this procedure, all computations in the local space (u,v) for product of global derivative integrals are free from geometric properties and hence they are pure numbers. We then propose a numerical scheme to integrate the products of global derivatives. We have shown that the matrix product of global derivative integrals is expressible as matrix triple product comprising of geometric properties matrices and the product of local derivative integrals matrix. We have obtained explicit integration of the product of local derivatives which is now possible by use of symbolic integration commands available in leading mathematical softwares MATLAB, MAPLE, MATHEMATIKA etc. In this paper, we have used the MATLAB symbolic mathematics to compute the integrals of the products of local derivatives in (u, v) space .The proposed explicit integration scheme is shown as a useful technique in the formation of element stiffness matrices for second order boundary 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 node Serendipity and sixteen node Lagrange families 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 ξ and η over a 2-square ($-1\leq \xi,\eta \leq 1$) with nodes at the boundary points {(-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)} and the

four interoior 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 (ξ, η). In this paper, we have computed these integrals in exact forms using the symbolic mathematics capabilities of MATLAB. The proposed explicit finite element integration scheme can be applied to solve boundary value problems in continuum mechanics over convex polygonal domains. We have also developed a novel auto mesh generation technique of all 12- node and 16-node linear convex quadrilaterals for a polygonal domain $\Omega \subset \mathcal{R}^2$ 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 $\Omega \subset \mathcal{R}^2$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

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

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

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

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

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

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

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

2.2 Weak Formulation of the Poisson Boundary Value Problem

A sufficiently smooth function u satisfying both eqns(1) and (2) is known as classical solution to the Poisson boundary value problem. For a Dirichlet problem, u is a classical solution only if it has continuousecond derivatives in Ω (i.e. u is $C^2(\Omega)$ and is continuous up to the boundary i.e.u is in $C^0(\bar{\Omega})$). 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,

$$\int_{\Omega} (\nabla^2 u + f)\, v = 0 \quad \text{.....(5)}$$

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

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

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

so that

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

The point here is that the problem posed byeqn(6) may have a solution u called a weak solution,that is not smooth enough to be a classical solution. If a classical solution does exist then eqn(6)is equivalent to eqns (1) and (2) and the weak solution is classical. The case of Neumann problem $(\alpha = 0\ , \beta = 1)$ in eqn(2) is particularly straight forward. Substituting from eqn(2) into eqn(6)gives us the following formulation: find u defined on Ω such that

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

for all suitable test functions v .

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

2.3.1 Weak Form

Given Poisson Equation:

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

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

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

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

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

For all test functions v .

2.3.2 Finite Elements

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

$\int_\Omega \nabla u_h \cdot \nabla v \, dx = \int_\Omega vf \, dx$ for all $v \in V_h$ .....(8)

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

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

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

$\sum_{j=1}^{N} \alpha_j \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, dx = \int_\Omega f\varphi_i \, dx$,i=1,2,3,...., N .....(9)

This is really a linear system of the form

$Ku = f$ .....(10)

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

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

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

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

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

2.3.3 Overview on the implementation of Finite Element Method

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

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

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

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

2.3.4 Assembling the Stiffness Matrix

In order to assemble the stiffness matrix,we need to compute integrals of the form(see eqn(11) in section (2.3.2)

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

The most obvious way to assemble the stiffness matrix is to compute the integrals $K_{i,j}$ for the nodal pairs i and j ; this is a node oriented computation and we need to know the common support of basis functions $\varphi_i \; and \; \varphi_j$.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

$K_{i,j} = \sum_{\Omega^e \; \varepsilon \Omega_h^e} \quad K_{i,j}^e \quad =$

.....(12a)

where

$K_{i,j}^e \qquad = \int_{\Omega^e} \nabla\varphi_i \, . \nabla\varphi_j \, dx$

.....(12b)

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

Notice that the integrals

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

We must also compute the integrals

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

where

$f_i^e \qquad = \qquad \int_{\Omega^e} f\varphi_i d\,\mathrm{x}$

.....(12d)

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

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

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

Where

$\mathrm{u}^e = (\mathrm{u}_1^e, \mathrm{u}_2^e, \mathrm{u}_3^e, \ldots\ldots\ldots\ldots \mathrm{u}_d^e)^T$, $f^e = (f_1^e, f_2^e, f_3^e, \ldots\ldots\ldots\ldots f_d^e)^T$ .....(12f)

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

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

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

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

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

.....(12g)

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

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

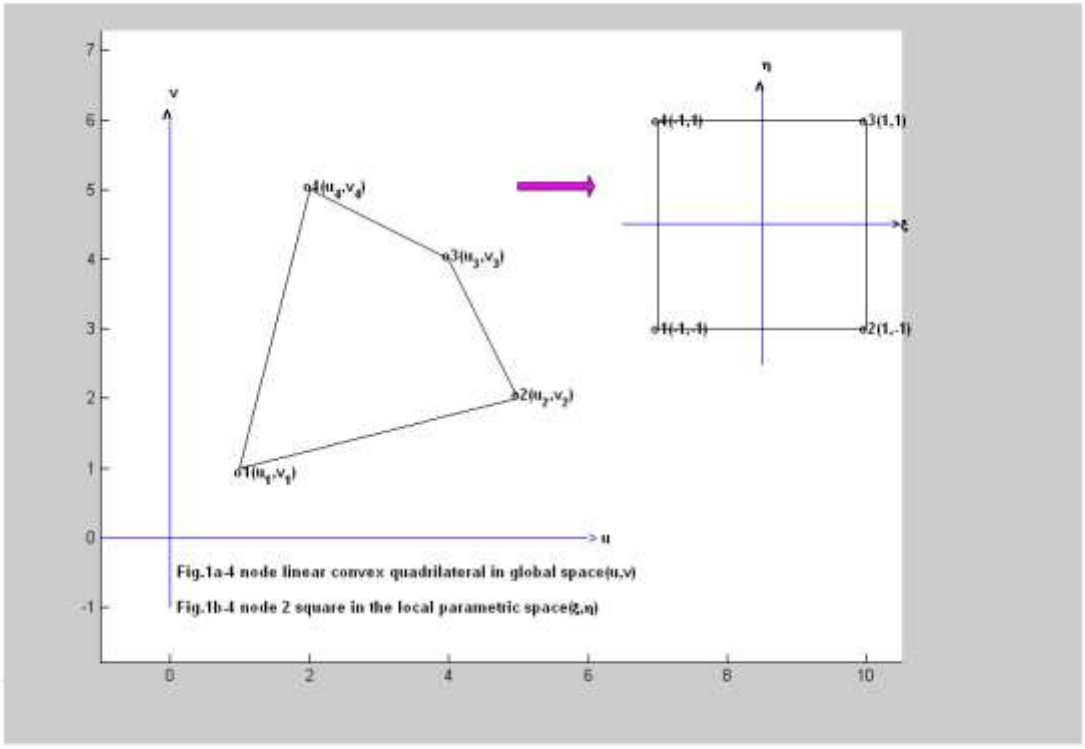
$f_i^e = \int_{\Omega^e} f\varphi_i$ dxdy can be computed in a straight forward manner if f is a simple function otherwise we have to apply numerical integration

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

2.4 Finite Element Types

2.4.1 Linear Convex Quadrilateral Elements :

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



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

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

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

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

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

2.4.2 Isoparametric Transformation :

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

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

Where ϕ_k , ψ_k refer to unknowns at node k and the shape functions $N_k{}^e = M_k$, and M_k are defined as in Eqn.(14a-b)

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

2.4.3 Subparmetric Transformation :

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

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

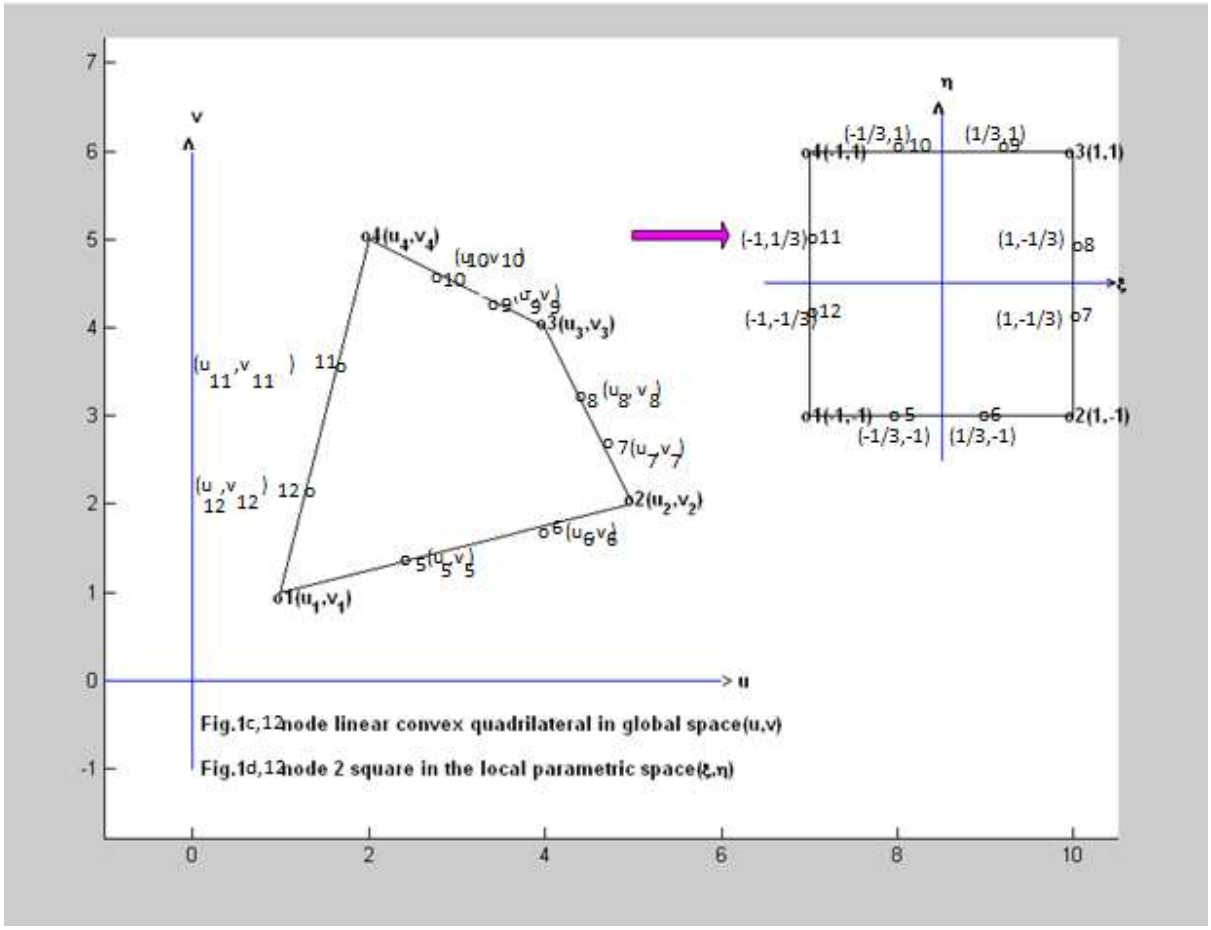
Where ϕ_k , ψ_k refer to unknowns at node k and nde >4

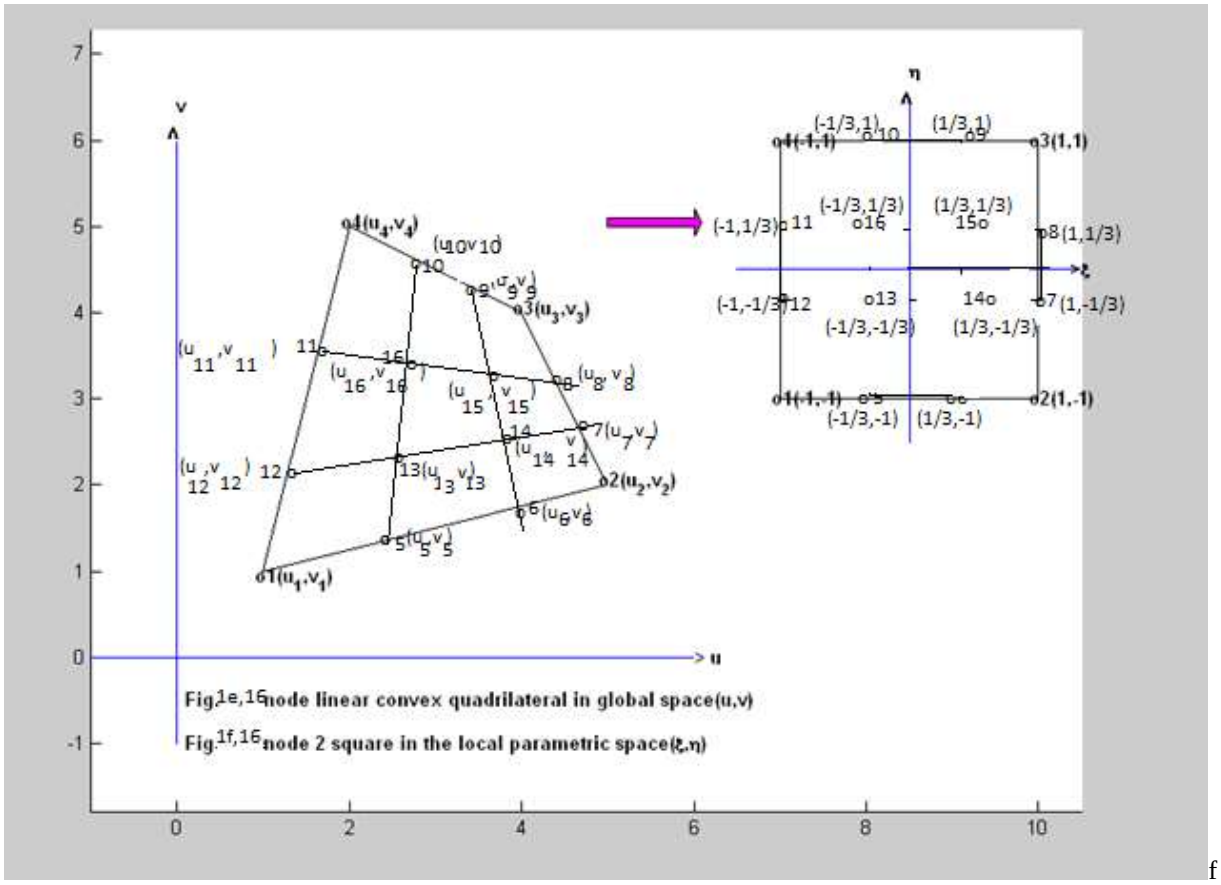
In our recent paper, the explicit finite element integration scheme is presented by using the isoparametric transformation over the 4 node linear convex quadrilateral element for which we set nde=4

In the present paper, we consider the subparametric transformation for a linear convex quadrilateral element for which nde = 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.





f

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

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

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

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

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

Using the transformation of Eqns.(13-14) and from Fig 1c , Fig 1d ; we see that there is a one to one correspondence between $((\xi_k,\eta_k),\ k = 1(1)12)$ 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)

where

$(u_5, v_5) = ((2u_1 + u_2)/3 , (2v_1 + v_2)/3)$,

$(u_6, v_6) = ((u_1 + 2u_2)/3 , (v_1 + 2v_2)/3)$,

$(u_7, v_7) = ((2u_2 + u_3)/3 , (2v_2 + v_3)/3)$,

$(u_8, v_8) = ((u_2 + 2u_3)/3 , (v_2 + 2v_3)/3)$,

$(u_9, v_9) = ((2u_3 + u_4)/3 , (2v_3 + v_4)/3)$,

$(u_{10}, v_{10}) =((u_3 + 2u_4)/3 , (v_3 + 2v_4)/3)$,

$(u_{11}, v_{11}) = ((u_1 + 2u_4)/3 , (v_1 + 2v_4)/3)$

$(u_{12}, v_{12}) = ((2u_1 + u_4)/3 , (2v_1 + v_4)/3)$,

$(u_{13}, v_{13}) == ((4u_1 + 2u_2 + u_3 + 2u_4)/9 , (4v_1 + 2v_2 + v_3 + 2v_4)/9)$,

$(u_{14}, v_{14}) = = ((2u_1 + 4u_2 + 2u_3 + u_4)/9 , (2v_1 + 4v_2 + 2v_3 + v_4)/9)$,

$(u_{15}, v_{15}) = = ((u_1 + 2u_2 + 4u_3 + 2u_4)/9 , (v_1 + 2v_2 + 4v_3 + 2v_4)/9)$,

$(u_{16}, v_{16}) == ((2u_1 + u_2 + 2u_3 + 4u_4)/9 , (2v_1 + v_2 + 2v_3 + 4v_4)/9)$, ---------- (15)

We then define the variation of physical variables ϕ^e, ψ^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 nde = 12,16

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

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

The shape functions $N_i{}^e$ of the 12- node cubic Serendipity element shown in Fig 1c , Fig 1d are given by

$N_1{}^e(\xi,\eta)=((1-\xi)(1-\eta)(-10+9(\xi^2+\eta^2)))/32;$
$N_2{}^e(\xi,\eta)=((1+\xi)(1-\eta)(-10+9(\xi^2+\eta^2)))/32;$
$N_3{}^e(\xi,\eta)=((1+\xi)(1+\eta)(-10+9(\xi^2+\eta^2)))/32;$
$N_4{}^e(\xi,\eta)=((1-\xi)(1+\eta)(-10+9(\xi^2+\eta^2)))/32;$
$N_5{}^e(\xi,\eta)=(9/32)((1-\eta)(1-\xi^2)(1-3\xi));$
$N_6{}^e(\xi,\eta)=(9/32)((1-\eta)(1-\xi^2)(1+3\xi));$
$N_7{}^e(\xi,\eta)=(9/32)((1+\xi)(1-\eta^2)(1-3\eta));$
$N_8{}^e(\xi,\eta)=(9/32)((1+\xi)(1-\eta^2)(1+3\eta));$
$N_9{}^e(\xi,\eta)=(9/32)((1+\eta)(1-\xi^2)(1+3\xi));$
$N_{10}{}^e(\xi,\eta)=(9/32)((1+\eta)(1-\xi^2)(1-3\xi));$
$N_{11}{}^e(\xi,\eta)=(9/32)((1-\xi)(1-\eta^2)(1+3\eta));$
$N_{12}{}^e(\xi,\eta)=(9/32)((1-\xi)(1-\eta^2)(1-3\eta));$ ----------(17a)

and we may check that

$N_k{}^e(\xi_k,\eta_k)=1$, $N_k{}^e(\xi_j,\eta_j)=0$,when $j \neq k$

$((\xi_k,\eta_k),\ k = 1(1)12) = \{(-1,-1),(1,-1),(1,1),(-1,1),(-1/3,-1),(1/3,-1),(1,-1/3),(1,1/3),(1/3,1),(-1/3,1),(-1,1/3),(-1,-1/3)\}$ ----------(17b)

The shape functions $N_i{}^e$ of the 16- node cubic Serendipity element shown in Fig 1e, Fig 1f are given by

$N_1{}^e(\xi,\eta)=(-9/16\xi^3+9/16\xi^2+1/16\xi-1/16)(-9/16\eta^3+9/16\eta^2+1/16\eta-1/16);$
$N_2{}^e(\xi,\eta)=(9/16\xi^3+9/16\xi^2-1/16\xi-1/16)(-9/16\eta^3+9/16\eta^2+1/16\eta-1/16);$
$N_3{}^e(\xi,\eta)=(9/16\xi^3+9/16\xi^2-1/16\xi-1/16)(9/16\eta^3+9/16\eta^2-1/16\eta-1/16);$
$N_4{}^e(\xi,\eta)=(-9/16\xi^3+9/16\xi^2+1/16\xi-1/16)(9/16\eta^3+9/16\eta^2-1/16\eta-1/16);$
$N_5{}^e(\xi,\eta)=(27/16\xi^3-9/16\xi^2-27/16\xi+9/16)(-9/16\eta^3+9/16\eta^2+1/16\eta-1/16);$
$N_6{}^e(\xi,\eta)=(-27/16\xi^3-9/16\xi^2+27/16\xi+9/16)(-9/16\eta^3+9/16\eta^2+1/16\eta-1/16);$

$N_7^e(\xi,\eta)=(9/16\xi^3+9/16\xi^2-1/16\xi-1/16)(27/16\eta^3-9/16\eta^2-27/16\eta+9/16);$
$N_8^e(\xi,\eta)=(9/16\xi^3+9/16\xi^2-1/16\xi-1/16)(-27/16\eta^3-9/16\eta^2+27/16\eta+9/16);$
$N_9^e(\xi,\eta)=(-27/16\xi^3-9/16\xi^2+27/16\xi+9/16)(9/16\eta^3+9/16\eta^2-1/16\eta-1/16);$
$N_{10}^e(\xi,\eta)=(27/16\xi^3-9/16\xi^2-27/16\xi+9/16)(9/16\eta^3+9/16\eta^2-1/16\eta-1/16);$
$N_{11}^e(\xi,\eta)=(-9/16\xi^3+9/16\xi^2+1/16\xi-1/16)(-27/16\eta^3-9/16\eta^2+27/16\eta+9/16);$
$N_{12}^e(\xi,\eta)=(-9/16\xi^3+9/16\xi^2+1/16\xi-1/16)(27/16\eta^3-9/16\eta^2-27/16\eta+9/16);$
$N_{13}^e(\xi,\eta)=(27/16\xi^3-9/16\xi^2-27/16\xi+9/16)(27/16\eta^3-9/16\eta^2-27/16\eta+9/16);$
$N_{14}^e(\xi,\eta)=(-27/16\xi^3-9/16\xi^2+27/16\xi+9/16)(27/16\eta^3-9/16\eta^2-27/16\eta+9/16);$
$N_{15}^e(\xi,\eta)=(-27/16\xi^3-9/16\xi^2+27/16\xi+9/16)(-27/16\eta^3-9/16\eta^2+27/16\eta+9/16);$
$N_{16}^e(\xi,\eta)=(27/16\xi^3-9/16\xi^2-27/16\xi+9/16)(-27/16\eta^3-9/16\eta^2+27/16\eta+9/16);$
and we may check that

$N_k^e(\xi_k,\eta_k)=1$, $N_k^e(\xi_j,\eta_j)=0$,when $j\neq k$,k=1(1)16

$((\xi_k,\eta_k),\ k=1(1)12)=\{(-1,-1),(1,-1),(1,1),(-1,1),(-1/3,-1),(1/3,-1),(1,-1/3),(1,1/3),(1/3,1),(-1/3,1),(-1,1/3),(-1,-1/3)\}$ and ----------(17b)

$((\xi_k,\eta_k),\ k=13,14,15,16)=\{(-1/3,-1/3),(1/3,-1/3),(1/3,1/3),(-1/3,1/3)\}$ --(17c)

4. Explicit Form of the Jacobian and Global Derivatives :

4.1 Jacobian

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

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

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

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

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

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

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

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

Where

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

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

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

4.2 Global Derivatives:

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

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

Where $\frac{\partial u}{\partial \xi}, \frac{\partial u}{\partial \eta}, \frac{\partial v}{\partial \xi}$ and $\frac{\partial v}{\partial \eta}$ are defined as in Eqs.(18a)–(18d) while J is defined in Eq.(19a-b) , ($i, j = 1,2,3, - - - - -$, nde) , nde = the number of nodes per element. We may recall that the explicit integration for linear convex quadrilateral with nde = 4 is already presented by the authors in their recent paper [18]. We take nde = 12,16for the present study.

5. Discretisation of an Arbitrary Triangle :

A linear convex polygon in the physical plane (x, y) can be always discretised into a finite number of linear triangles. However, we would like to study a particular discretization of these triangles further into linear convex quadrilaterals. This is stated in the following Lemma [26].

We first consider an arbitrary triangle ΔPQR 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 ΔPQR 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 ΔABC 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(½, ½), E(0, ½) and F(½, 0). We now divide the ΔABC into three special quadrilaterals $\widehat{Q_1}$, $\widehat{Q_2}$ and $\widehat{Q_3}$ (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 $\widehat{Q_i}$ $(i = 1,2,3)$ satisfying the conditions

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

(ii) $\sum_{i=1}^{3} \widehat{Q_i} = \Delta ABC$, 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

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

Where Δ pqr is the area of the triangle Δ PQR

$$2\Delta\, pqr = \begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix} = \left[(x_p - x_r)(y_q - y_r) - (x_q - x_r)(y_p - y_r) \right] \quad \text{---------- (21b)}$$

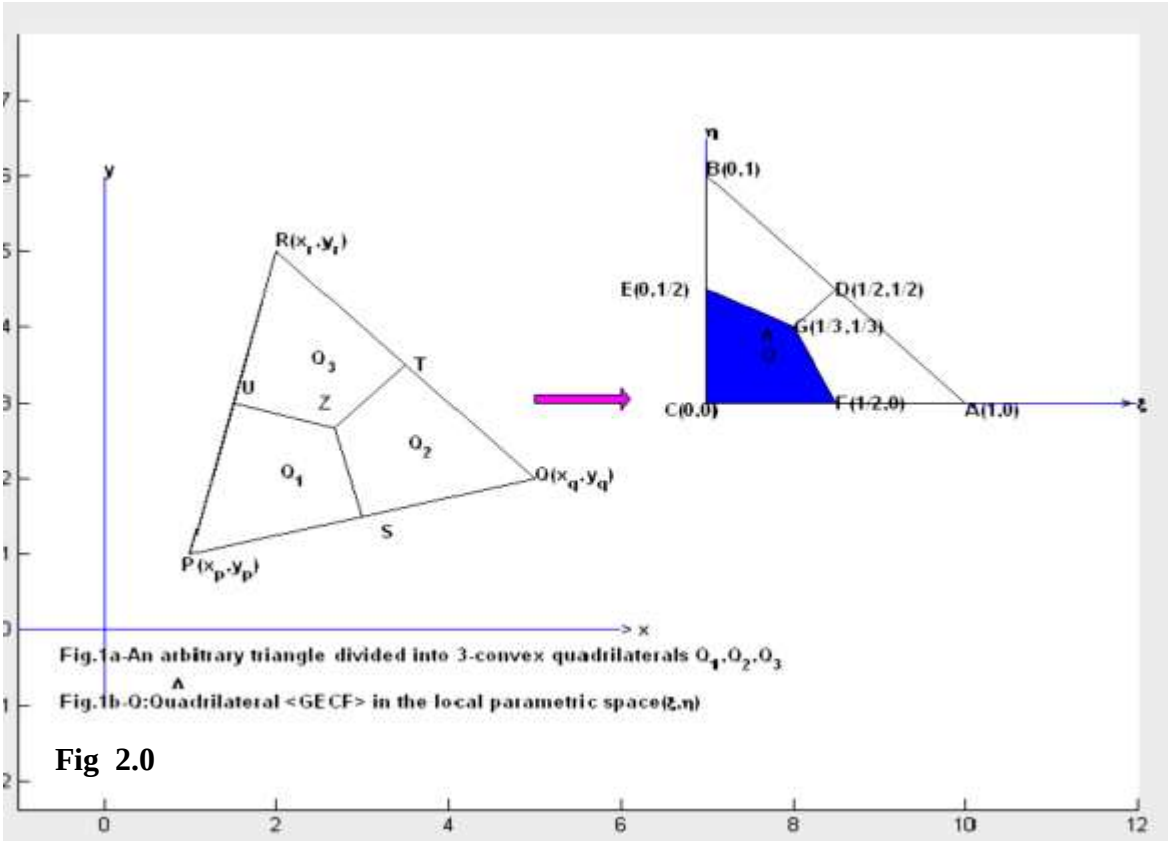


Fig 2.0

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.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} w + \begin{pmatrix} x_q \\ y_q \end{pmatrix} u + \begin{pmatrix} x_r \\ y_r \end{pmatrix} v \, , \; w = 1 - u - v, \qquad \text{.....(22)}$$

We can verify that the above transformation of eqn. Above maps the arbitrary triangle ΔPQR into the standard triangle ΔABC . 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 $\widehat{Q_i}$. 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 ΔPQR into a unique triangle $\hat{Q} = Q_i$ (say) of the standard triangle ΔABC which satisfy the following conditions

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

(ii) $\sum_{i=1}^{3} \widehat{Q_i} \; = \Delta ABC$, 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.

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

$$\begin{pmatrix} x^{(2)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} x_q \\ y_q \end{pmatrix} w + \begin{pmatrix} x_r \\ y_r \end{pmatrix} u + \begin{pmatrix} x_p \\ y_p \end{pmatrix} v \, , \; w = 1 - u - v, \qquad (24)$$

$$\begin{pmatrix} x^{(3)} \\ y^{(3)} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} w + \begin{pmatrix} x_p \\ y_p \end{pmatrix} u + \begin{pmatrix} x_q \\ y_q \end{pmatrix} v \, , \; w = 1 - u - v, \qquad (25)$$

It is quite clear that each of the above transformations map the arbitrary triangle ΔPQR into ΔABC. 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 Q_1 spanning the vertices < Z, U, P, S > in (x, y) space into the quadrilateral $\hat{Q} = Q_1$ 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 Q_2 spanned by vertices < Z, S, Q, T > in (x, y) space is mapped into the quadrilateral $\hat{Q} = \widehat{Q_1}$ spanned by the vertices < G, E, C, F > in the (u, v) space. Finally on using the linear transformation of eqn.(4), the quadrilateral Q_3 spanned by vertices < Z, T, R, U > in (x, y) space is mapped into the quadrilateral $\hat{Q} = Q_1$ 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 $(x^{(1)},\ y^{(1)})^T$ in eqn.(23) and $(x, y)^T$ 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 ΔABC 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 $\widehat{Q_e}$ $(e = 1,2,3)$ viz. < G, E, C, F> , <G, F, A, D> and <G, D, B, E> have the same expression given by

$$\hat{J} = \frac{\partial(u,v)}{\partial(\xi,\eta)} = \hat{J}^{e} = \frac{1}{96}(4 + \xi + \eta), \qquad (e = 1,2,3) \tag{21a}$$

Proof: We can immediately verify that eqn.(10a) is true by substituting the nodal values of $\widehat{Q_e}$ in eqn.(9a-b). The general result for a special quadrilateral Q_e $(e = 1,2,3)$ 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):

$$J = J^e = \frac{\partial(x,y)}{\partial(\xi,\eta)} = \frac{\partial(x,y)}{\partial(u,v)}\,\frac{\partial(u,v)}{\partial(\xi,\eta)} = (2\Delta_{pqr})\frac{1}{96}(4 + \xi + \eta) = \frac{\Delta_{pqr}}{48}(4 + \xi + \eta) \tag{21b}$$

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(1/2, 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 Q_i $(i = 1,2,3)$ in (x,y) space , $\widehat{Q_i}$ $(i = 1.2.3)$ 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 Δ PQR shown in Fig 2a or Fig 3a. Let ϕ(x, y) be an arbitrary and smooth function defined over the region Δ PQR . We now consider

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

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

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

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

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

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

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

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

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

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

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

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

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

---------- (31)

Where from Eqn.(29), we write

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

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

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

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

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

6.2 Global Derivative Integrals :

If $N_i^{(e)}$ (i =1(1)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

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

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

We must now use the earlier transformations.

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

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

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

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

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

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

Hence, we have from Eqn.(34)

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

This gives

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

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

where

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

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

$$\gamma_a = (x_c - x_b)\ , \qquad \gamma_b = (x_a - x_c)\ , \qquad \text{----------(37a)}$$

and

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

$$= (\gamma_b \beta_a - \gamma_a \beta_b) \qquad \text{---------- (37b)}$$

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

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

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

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

Letting,

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

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

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

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

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

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

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

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

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

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

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

$$= P\, G_{u,v}^{i,j,e}\, P^T \qquad \text{---------- (41c)}$$

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

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

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

where, we have already defined the 8- node linear convex quadrilaterals Q_e (e=1,2,3) in (x,y) space and $\widehat{Q}$ in (u,v) space in Fig 3a- 3b. From Eqns.(41)-(43) , we obtain the following relations connecting the submatrices $S^{i,j,e}$ and $K^{i,j,e}$

We now obtain the submatrices $S^{i,j,e}$ and $K^{i,j,e}$ in an explicit form from Eqs.(41a)- (41b)

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

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

and in similar manner

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

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

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

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

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

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

$$= 2\Delta_{abc}\, P\,(K^{i,j,e})\, P^T \quad , (i,j = 1,2,3,4,5,6,7,8,9) \qquad \text{----------(46)}$$

We can thus obtain the global derivative integrals in the physical space or Cartesian space (x,y) by using the matrix triple product established in Eqn.(46).

We note that $\widehat{Q}$ is the 8- node linear convex quadrilateral in (u, v) space spanned by the vertices (1/3, 1/3), (1/6, 5/12), (0, ½), (0, ¼), (0, 0), (1/4, 0), (1/2, 0) , (5/12, 1/6) and (5/24,5/24) in uv- plane hence from Eqn.(45)

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

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

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

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

Thus, we have from Eq.(46)

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

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

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

The matrices P, P^T depend purely on the nodel coordinates (x_a ,y_a) ,(x_b ,y_b) , (x_c , y_c) the matrix $K^{i,j,e}$ can be explicity computed by the relations obtained in section 2 – 6. We find that $K^{i,j,e}$ is a (2X2) matrix of integrals whose integrands are rational functions with polynomial numerator and the linear denominator $(4 + \xi + \eta)$. Hence these integrals can be explicity computed. The explicit values of these integrals are expressible in terms of logarithmic constants. We have used symbolic mathematics software of MATLAB to compute the explicit values and their conversion to any number of digits can be obtained by using variable precision arithmetic (vpa) command. The matrix K^e as noted in Eqn.(45) is of order $(2xn_{de})$ x $(2xn_{de})$, n_{de} = 8 = for 8-node convex quadrilateral element.

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

We may note that In order to compute the local/element stiffness matrices for the Poisson Boundary Value problem, we need to compute the integrals Eqns(12a-b)

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

,,(51a)

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

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

.....(51b)

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

The explicit integration scheme explained above compute four derivative product integrals as given in eqn(44) and they are necessary to compute the stiffness matrix entries of plane stress/plane strain problems in elasticity and sevral other applications in continuum mechanics.But this computation requires matrix triple product as given in eqn (50).Since,we only need the sum of two of these integrals viz : $S^{e}_{2i-1,2j-1} + S^{e}_{2i,2j}$ .We now present an efficient method to compute this sum by using matrix product.

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

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

$$= \begin{pmatrix} \mathbf{S^e_{2i-1,2j-1}} & \mathbf{S^e_{2i-1,2j}} \\ \mathbf{S^e_{2i,2j-1}} & \mathbf{S^e_{2i,2j}} \end{pmatrix} \quad \textbf{(say)}$$

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

.....(52a)

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

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

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

.....(52b)

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

.....(53)

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

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

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

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

.....(54)

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

$$\textbf{trace}\ (\mathbf{S^{i,j,e}}) = \textbf{trace}\left(\iint_{\mathbf{Q_e}} \mathbf{G^{i,j,e}_{x,y}}\ \mathbf{dxdy}\right) = (\mathbf{S^e_{2i-1,2j-1}} + \mathbf{S^e_{2i,2j}}) = K_{i,j}^{e} = \int_{Q_e} \nabla \mathbf{N}_i \,.\, \nabla N_j\, dx = \int_{Q_e} \left\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\right\} dxdy$$

$$= (P_{11}^2 + P_{21}^2) I_{u,u}^{i,j} + (P_{11}\,P_{12} + P_{21}\,P_{22})\left(I_{u,v}^{i,j} + I_{v,u}^{i,j}\right) + (P_{12}^2 + P_{22}^2) I_{v,v}^{i,j} \qquad \textbf{.....(55)}$$

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

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

$$(P^T\,P)\,.*\begin{pmatrix} I_{u,u}^{i,j} & I_{u,v}^{i,j} \\ I_{v,u}^{i,j} & I_{v,v}^{i,j} \end{pmatrix} = \begin{bmatrix} (P_{11}^2+P_{21}^2)\,I_{u,u}^{i,j} & (\,P_{11}\,P_{12}+P_{21}\,P_{22})I_{u,v}^{i,j} \\ (\,P_{11}\,P_{12}+P_{22}\,P_{21})I_{v,u}^{i,j} & (P_{12}^2+P_{22}^2)I_{v,v}^{i,j} \end{bmatrix} \quad \text{.....(56)}$$

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

$$\int_{Q_e}\{\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\}\,dxdy = sum(sum\left(\;(P^T\,P).*\begin{pmatrix} I_{u,u}^{i,j} & I_{u,v}^{i,j} \\ I_{v,u}^{i,j} & I_{v,v}^{i,j} \end{pmatrix}\;\right)) \quad \text{.....(57)}$$

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

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

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

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

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

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

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

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

-------- (59)

Where $(x^{(e)}(u,v),\;y^{(e)}(u,v)), e = 1,2,3)$ are the transformations of Eqs.(8)–(10) and $\hat{Q}$ is the quadrilateral in uv- plane spanned by vertices G(1/3 ,1/3) , E(0, ½) , C(0, 0) and F(1/2,0) , and Δ_{pqr} is the area of triangle Δ PQR, Now using the transformations defined in Eqs.(1)–(2) we obtain

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

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

In Eq.(14) we have used the transformation

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

$$v(\xi,\;\eta) = \frac{1}{3}N_1(\xi,\;\eta)+\frac{1}{2}N_2(\xi,\;\eta) \quad \text{--------}$$

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

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

We can now obtain from Eqs.(13)–(14)

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

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

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

Using Numerical Integration ;

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

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

Where,

$u_{i,j}^{(N)} = u(\xi_i^{(N)}, \eta_j^{(N)})$ and $v_{i,j}^{(N)} = v(\xi_i^{(N)}, \eta_j^{(N)})$ --------- ---------- (64)

and $(W_i^{(N)}, \xi_i^{(N)})$, $(W_j^{(N)}, \xi_j^{(N)})$ are the weight coefficients and sampling points of N^{th} order Gauss Legendre Quadrature rules.

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

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

7 A NEW APPROACH TO MESH GENERATION

The first step in implimenting finite element method isto generate a mesh.In a recent work the author and his co-workers have proposed a new approach to mesh generation which can discretise a convex polygon into an all quadrilateral mesh.This will be presented next.This new approach to mesh generation meets the necessary requirements of regularity on the shape of elements.There are two types of them which usually suffice in finite element computations.The first is called shape regularity. It says that the ratio of the diameter of the element to the radius of the inner circle must be less than some constant. For triangles,the diameter of the triangle is related to the smallest circle which contain the triangle.The inner circle refers to the largest circle which fits inside the triangle. Shape regularity focuses on the shape of individual triangles and doesnot refer to how the shapes of different elements relate to each other. So some elements can be large wshile others might be very small. There is a second type of requirement on the shape of elements.This requirement says that ratio of the maximum diameter of elements to the radius of the inner circle of an element must be less than some constant .If a mesh satisfies this requirement,it is called quasiuniform.This requirement is more important when we perform refinements.We must note that a mesh generation gives us the nodes on a particular element as well as the coordinates of the nodes.We now give an account of this novel mesh generation technique with an aim to use it further in the solution of Poisson problem. Stated in eqn(7a-b).

In our recent 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 torison 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 discritised 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 derivative equations(PDE).In the last few decade,one of the most relevant techniques to solve is the Finite Element Method(FEM).It is well known that a good quality mesh is required in order to obtain an accurate solution.Hence the construction of a mesh is on e of the most important steps.

In the next few sections , we present a novel mesh generation scheme of all quadrilateral elements for convex polygonal domains. This scheme converts the elements in background triangular mesh into quadrilaterals through the operation of splitting. We first decompose the convex polygon into simple subregions in the shape of triangles. These simple subregions are then triangulated to generate a fine mesh of triangles. We propose then an automatic triangular to quadrilateral conversion scheme in which each isolated triangle is split into three quadrilaterals according to the usual scheme, adding three vertices in the middle of edges and a vertex at the barrycentre of the triangular element. Further, to preserve the mesh conformity a similar procedure is also applied to every triangle of the domain and this fully discretizes the given convex polygonal domain into all quadrilaterals, thus propogating uniform refinement. In section 4.2, we present a scheme to discretize the arbitrary and standard triangles into a fine mesh of six node triangular elements. In section 4.3, we explain the procedure to split these triangles into quadrilaterals. In section 4.4,we have presented a method of piecing together of all triangular subregions and eventually creating a all quadrilateral mesh for the given convex polygonal domain. In section 4.5,we present several examples to illustrate the simplicity and efficiency of the proposed mesh generation method for standard and arbitrary triangles,rectangles and convex polygonal domains.

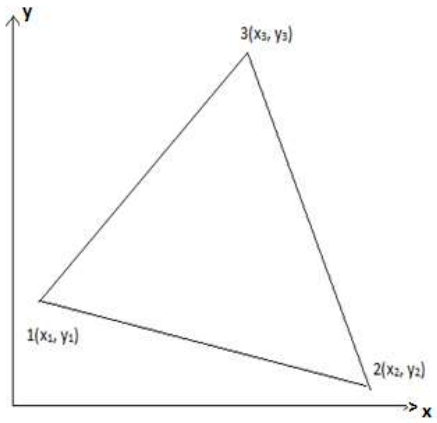
7.2 Division of an Arbitrary Triangle

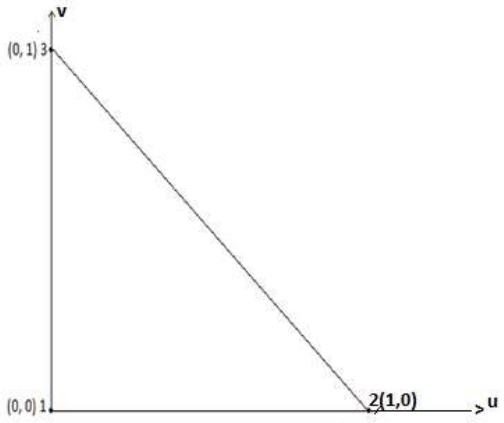
We can map an arbitrary triangle with vertices ((x_i, y_i), $i = 1, 2, 3$) into a right isosceles triangle in the (u, v) space as shown in Fig. 4a, b. The necessary transformation is given by the equations.

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

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

The mapping of eqn.(1) describes a unique relation between the coordinate systems. This is illustrated by using the area coordinates and division of each side into three equal parts in Fig. 5a Fig. 5b. It is clear that all the coordinates of this division can be determined by knowing the coordinates ((x_i, y_i), $i = 1, 2, 3$) of the vertices for the arbitrary triangle. In general , it is well known that by making 'n' equal divisions on all sides and the concept of area coordinates, we can divide an arbitrary triangle into n^2 smaller triangles having the same area which equals Δ/n^2 where Δ is the area of a linear arbitrary triangle with vertices ((x_i, y_i), $i = 1, 2, 3$) in the Cartesian space.

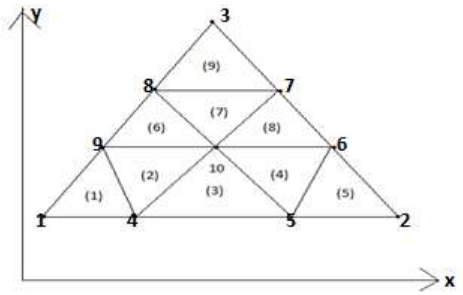


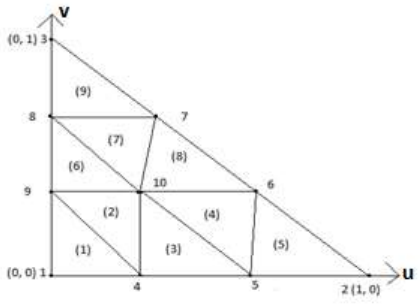


4.a **4 b**

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

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

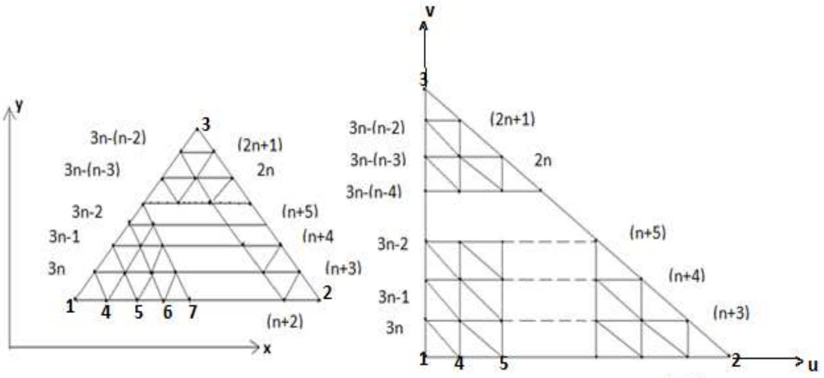




5a 5b

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

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



6a

6b

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

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

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

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

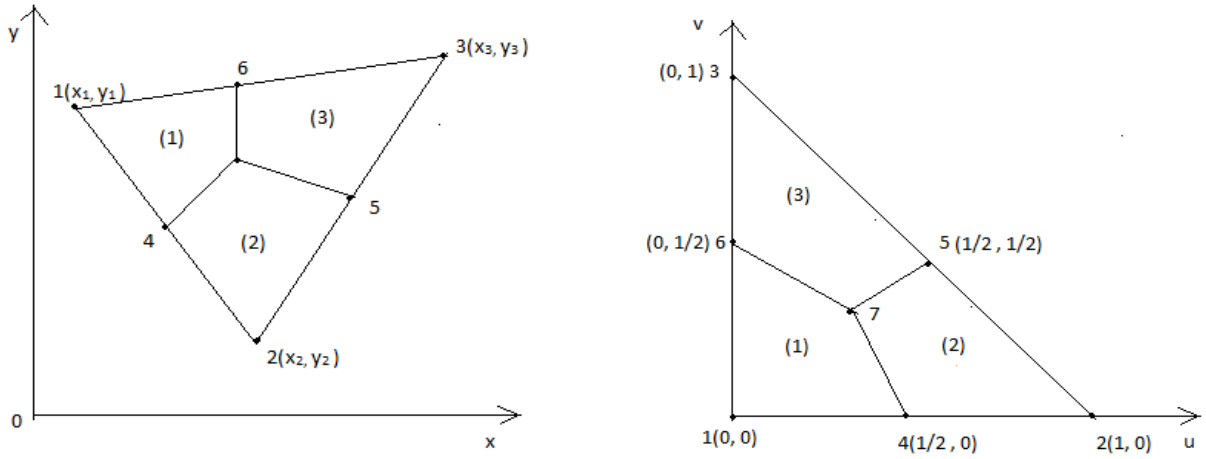
...(58)

7.3. Quadrangulation of an Arbitrary Triangle

We now consider the quadrangulation of an arbitrary triangle. We first divide the arbitrary triangle into a number of equal size six node triangles. Let us define l_{ij} as the line joining the points (x_i, y_i) and (x_j, y_j) in the Cartesian space (x, y). Then the arbitrary triangle with vertices at $((x_i, y_i), i = 1, 2, 3)$ is bounded by three lines l_{12}, l_{23}, and l_{31} . By

dividing the sides l_{12}, l_{23}, l_{31} into $n = 2m$ divisions (m, an integer) creates m^2 six node triangular divisions. Then by joining the centroid of these six node triangles to the midpoints of their sides, we obtain three quadrilaterals for each of these triangle. We have illustrated this process for the two and four divisions of l_{12}, l_{23}, and l_{31} sides of the arbitrary and standard triangles in Figs. 4 and 5

Two Divisions of Each side of an Arbitrary Triangle

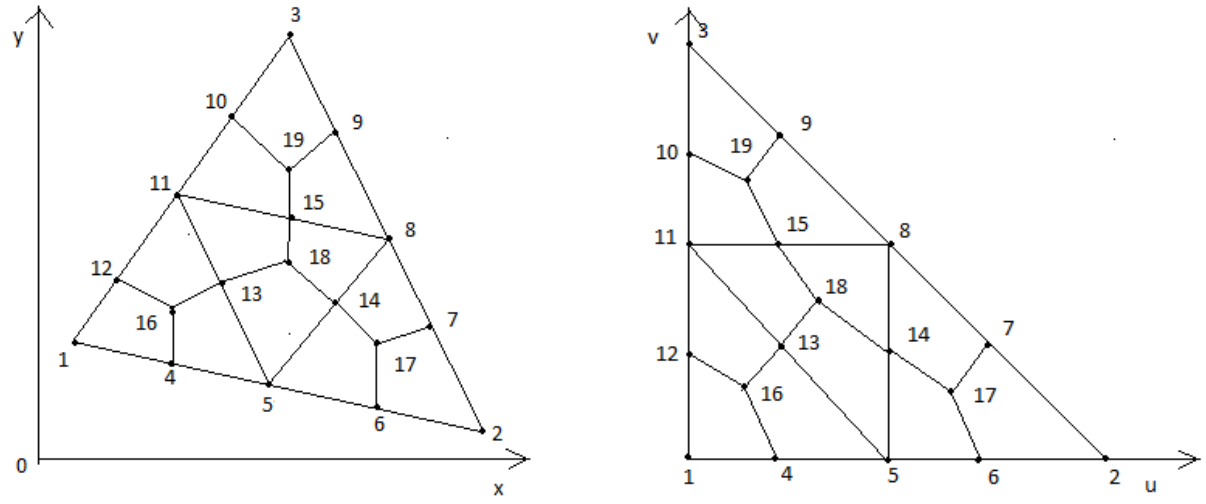


7(a) **7(b)**

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

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

Four Divisions of Each side of an Arbitrary Triangle



8a **8b**

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

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

In general, we note that to divide an arbitrary triangle into equal size six node triangle, we must divide each side of the triangle into an even number of divisions and locate points in the interior of triangle at equal spacing. We also do similar divisions and locations of interior points for the standard triangle. Thus n (even) divisions creates (n/2)^2 six node triangles in both the spaces. If the entries of the sub matrix $\underline{rr}\,(i\,;\,i+2,\ j\,;\,j+2)$ are nonzero then two six node triangles can be

formed. If $\underline{rr}\,(i+1,\ j+2) = \underline{rr}\ (i+2,\ j+1;\ j+2) = 0$ then one six node triangle can be formed. If the sub matrices $\underline{rr}\,(i\,;\ i+2,\ j\,;\ j+2)$ is a (3×3) zero matrix , we cannot form the six node triangles. We now explain the creation of the six node triangles using the $\underline{rr}$ matrix of eqn.( ). We can form six node triangles by using node points of three consecutive rows and columns of $\underline{rr}$ matrix. This procedure is depicted in Fig. 9 for three consecutive rows $i\,, i+1,$ $i+2$ and three consecutive columns $j\,,\ j+1,\ j+2$ of the $\underline{rr}$ sub matrix

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

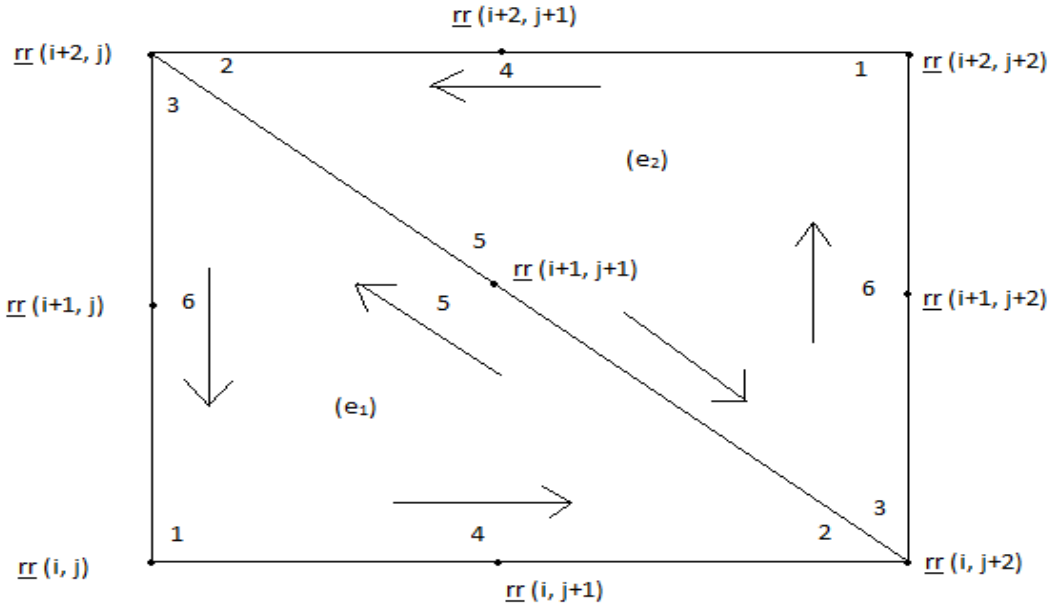


Fig. 9 Six node triangle formation for non zero sub matrix $\underline{rr}$

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

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

$(e_2) <\underline{rr}\ (i+2, j+2), \underline{rr}\,(i+2, j), \underline{rr}\,(i, j+2), \underline{rr}\,(i+2, j+1), \underline{rr}\,(i+1, j+1), \underline{rr}\ i+1, j+2)>$
.....(59)

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

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

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

$Q_{3n_1} < c_1\,, \underline{rr}\ (i+1,\ j+1), \underline{rr}\,(i+2,\ j)\,, \underline{rr}\,(i+1, j)>$
.....(60)

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

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

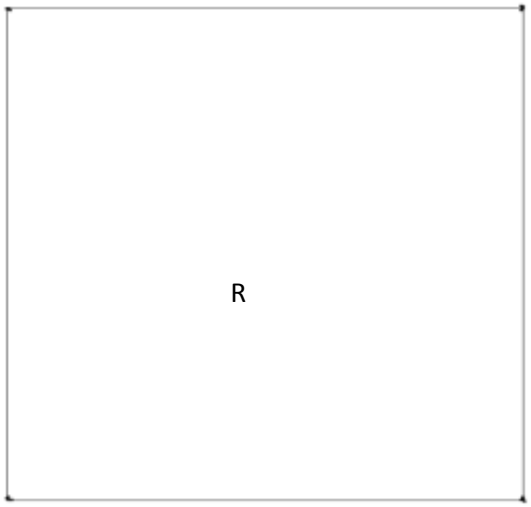
$Q_{3n_2-1} < c_2\,, \underline{rr}\ (i+2,\ j+1), \underline{rr}\,(i+2,\ j)\,, \underline{rr}\,(i+1,\ j+1)>$

$Q_{3n_1} < c_2\,, \underline{rr}\ (i+1,\ j+1), \underline{rr}\,(i,\ j+2)\,, \underline{rr}\,(i+1, j+2)>$ ----------
---- (61)

7.4 Quadrangulation of the Polygonal Domain

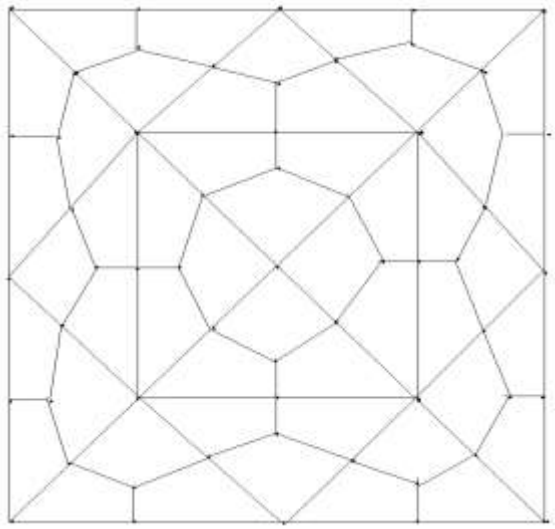
We can generate polygonal meshes by piecing together triangular with straight sides. Subsection (called LOOPs). The user specifies the shape of these LooPs by designating six coordinates of each LOOP

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



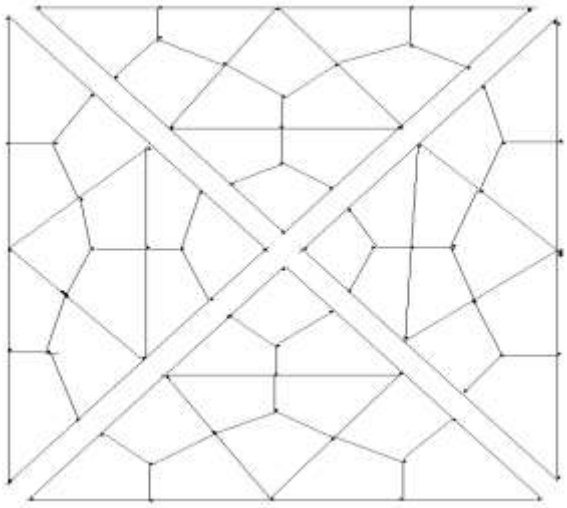
10a

(i)Fig.10a: Region R to be analyzed



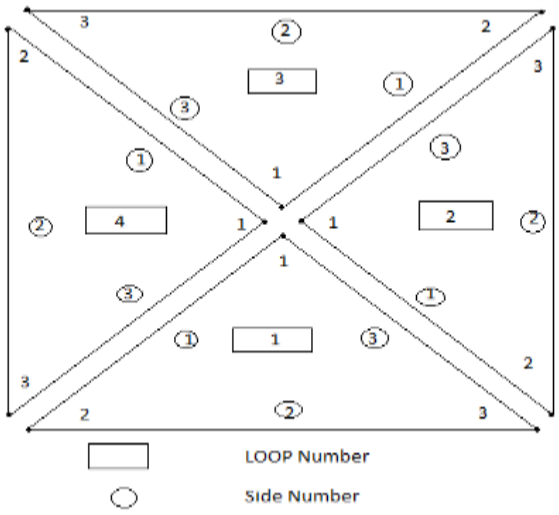
10b

(ii) Fig.10b: Example of completed mesh



10c

(iii)Fig.10c:Exploded view showing four loops



10d

(iv)Fig.10d:Example of a loop and side numbering scheme

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

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

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

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

The auto mesh generation technique discritises a polygonal domain into all four node special quadrilateral elements.We can convert these into 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
x=([0;1/2;1/2])
y=([0 0;1/2])
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.

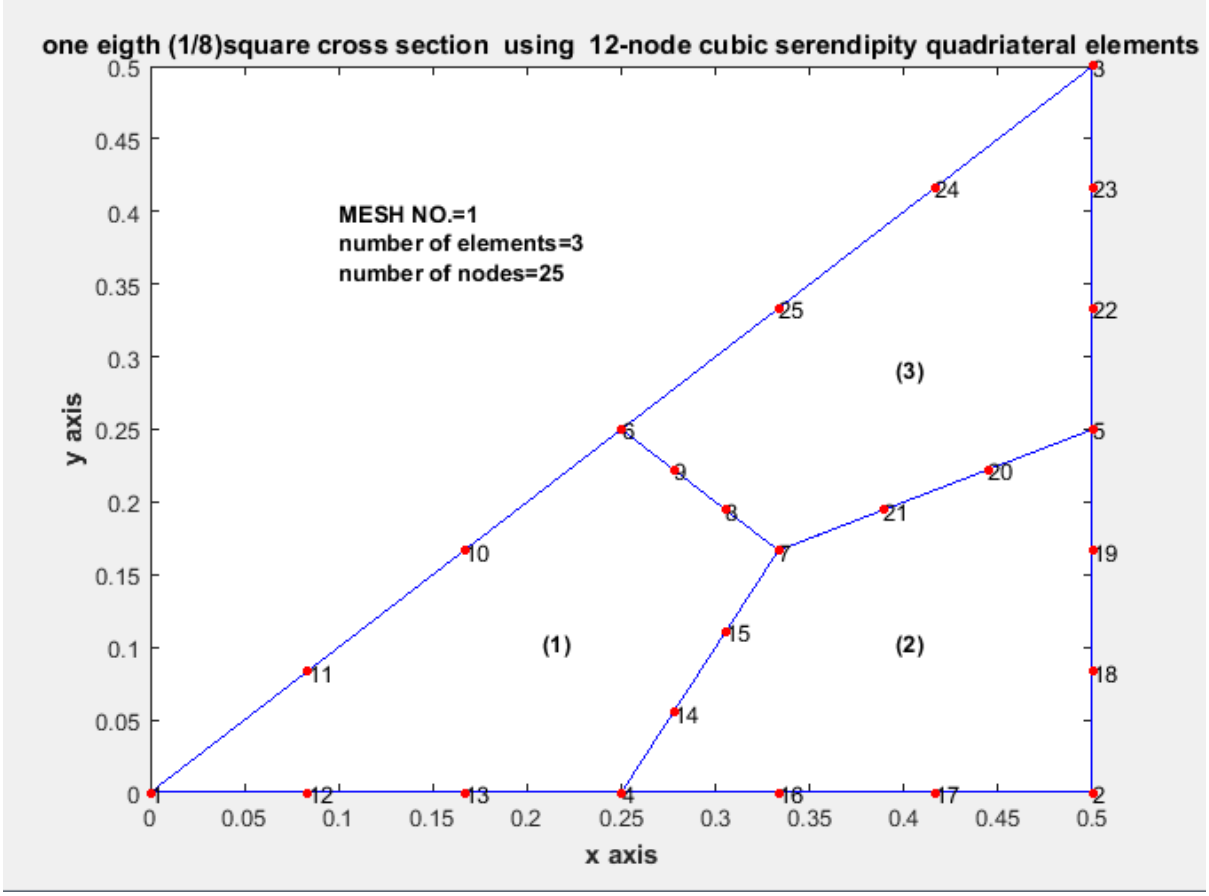


Fig.11a Initial mesh of right isosceles triangle(12-node quadrilaterals)

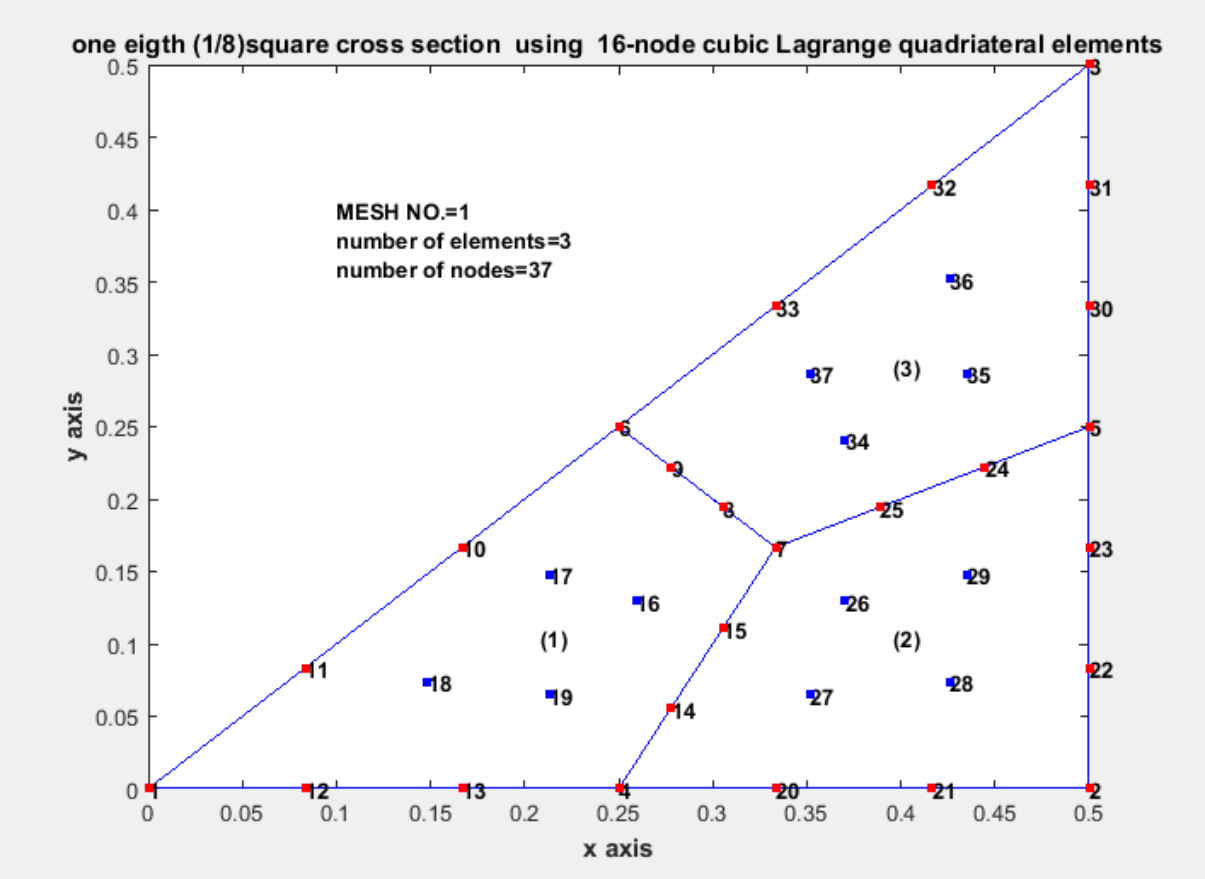


Fig.11b Initial mesh of right isosceles triangle(16-node quadrilaterals)

Example 2: equilateral triangle,each side=2*sqrt(3)
x=([$-\sqrt{3}$;$\sqrt{3}$; 0])
y=([-1;-1; 2])
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

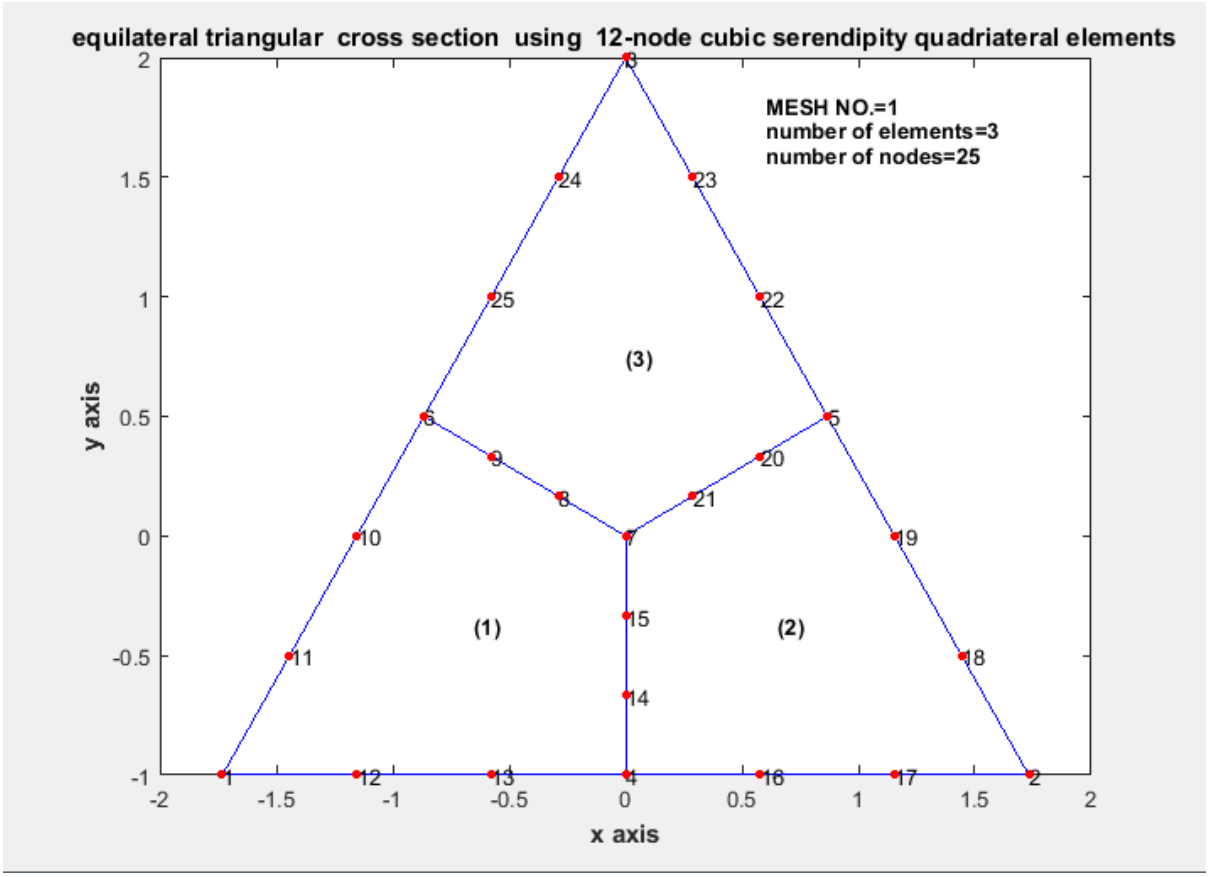


Fig.12a Initial mesh of equilateral triangle,each side=2*sqrt(3) (12-node quadrilaterals)

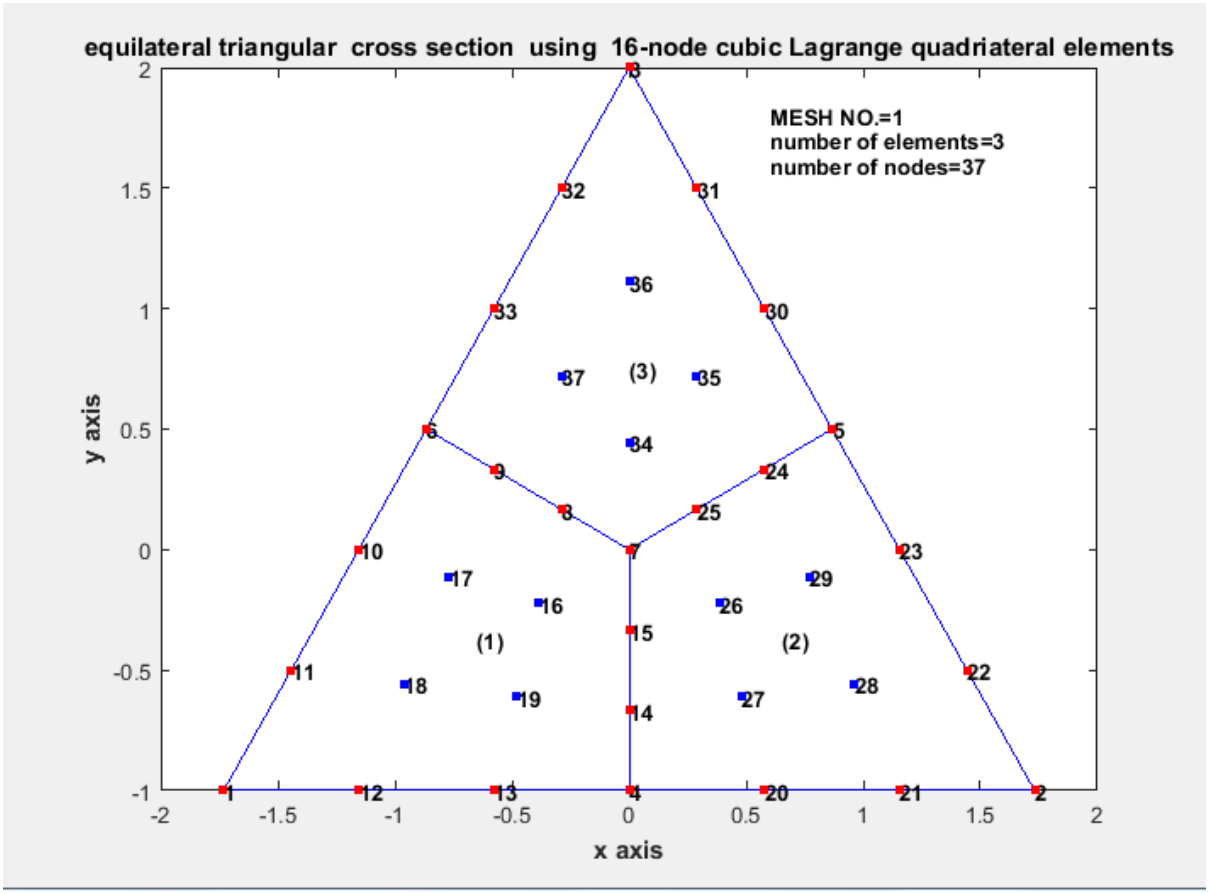


Fig.12b Initial mesh of equilateral triangle,each side=2*sqrt(3) (16-node quadrilaterals)

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

x=([1/2;1/2;1; 1; 1;1/2;0; 0;0])%FOR UNIT SQUARE
y=([1/2; 0;0;1/2; 1; 1;1;1/2;0])%FOR UNIT SQUARE
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)

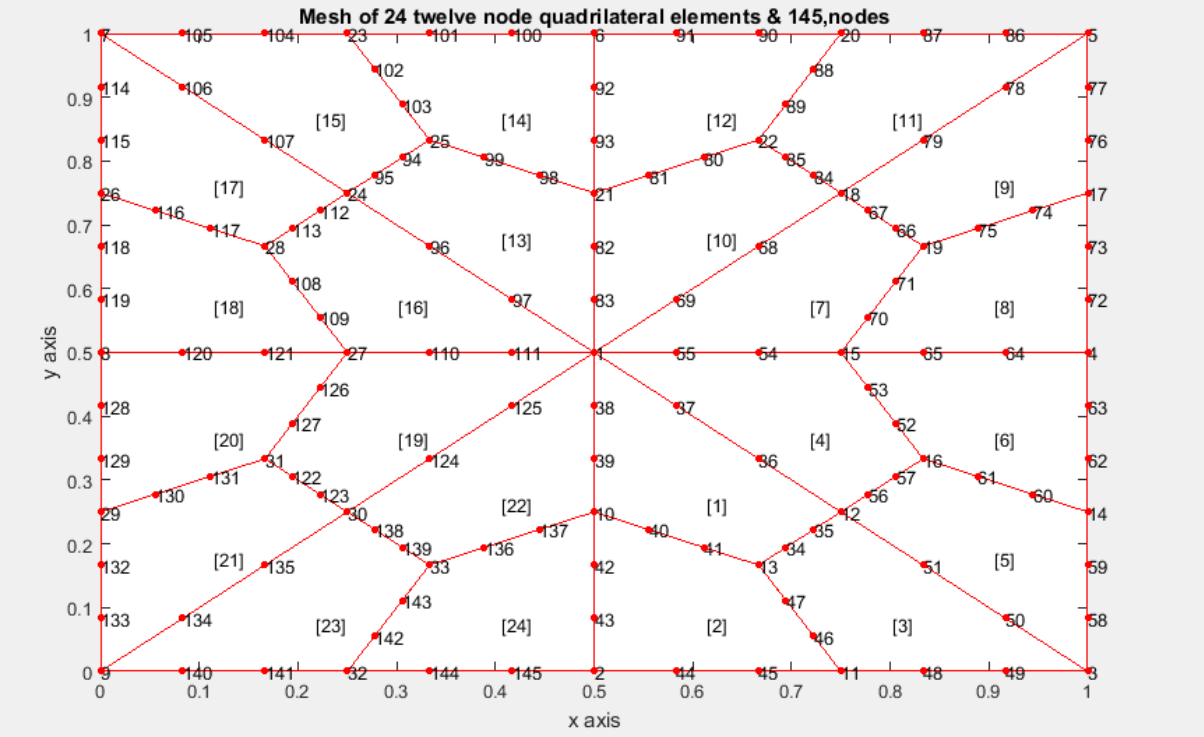


Fig.13a Initial Mesh for a square domain with eight triangles-12 noded quadrilaterals

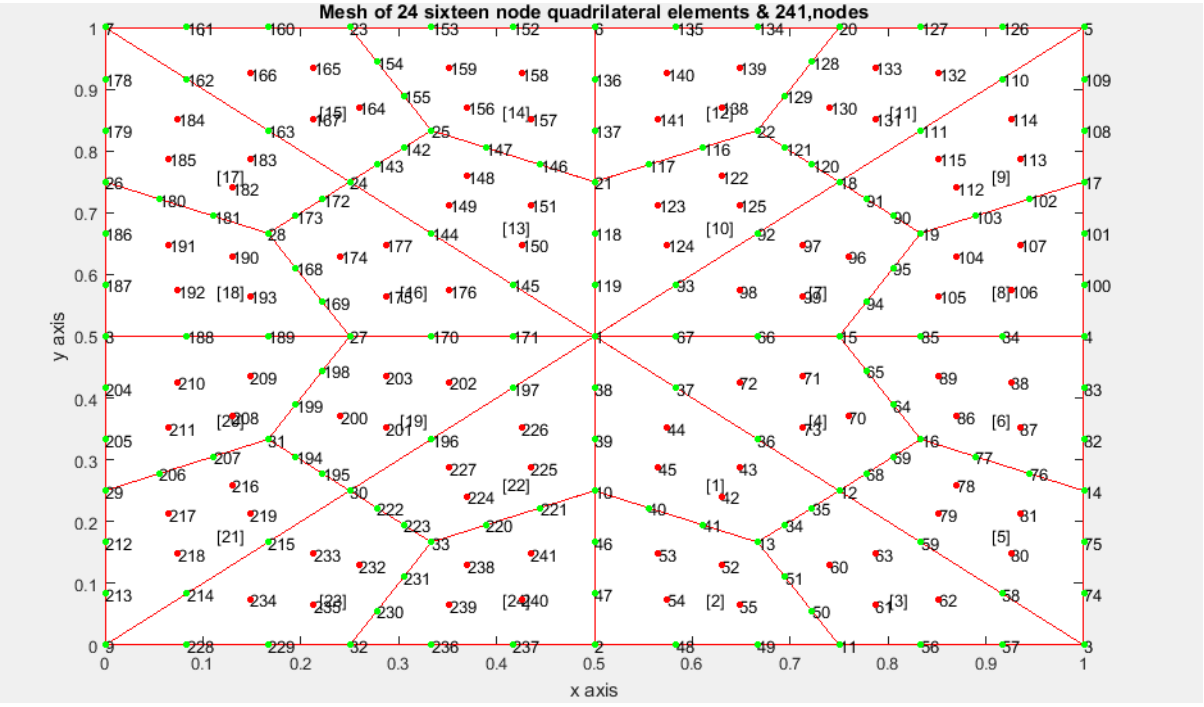


Fig.13b Initial Mesh for a square domain with eight triangles-16 noded quadrilaterals

Fig.13b Initial Mesh for a square domain with eight triangles(9-boundary nodes)

Example 4:pentagonal domain with seven triangles(8-boundary nodes)

x=([1/2;1/2;1; 1;1/2;0; 0;0])%for MOIN EXAMPLE
y=([1/2; 0;0;1/2; 1;1;1/2;0])%for MOIN EXAMPLE
We use this mesh to solve Poisson equation with a nonconstant smooth function on right hand side,with a known analytical solution

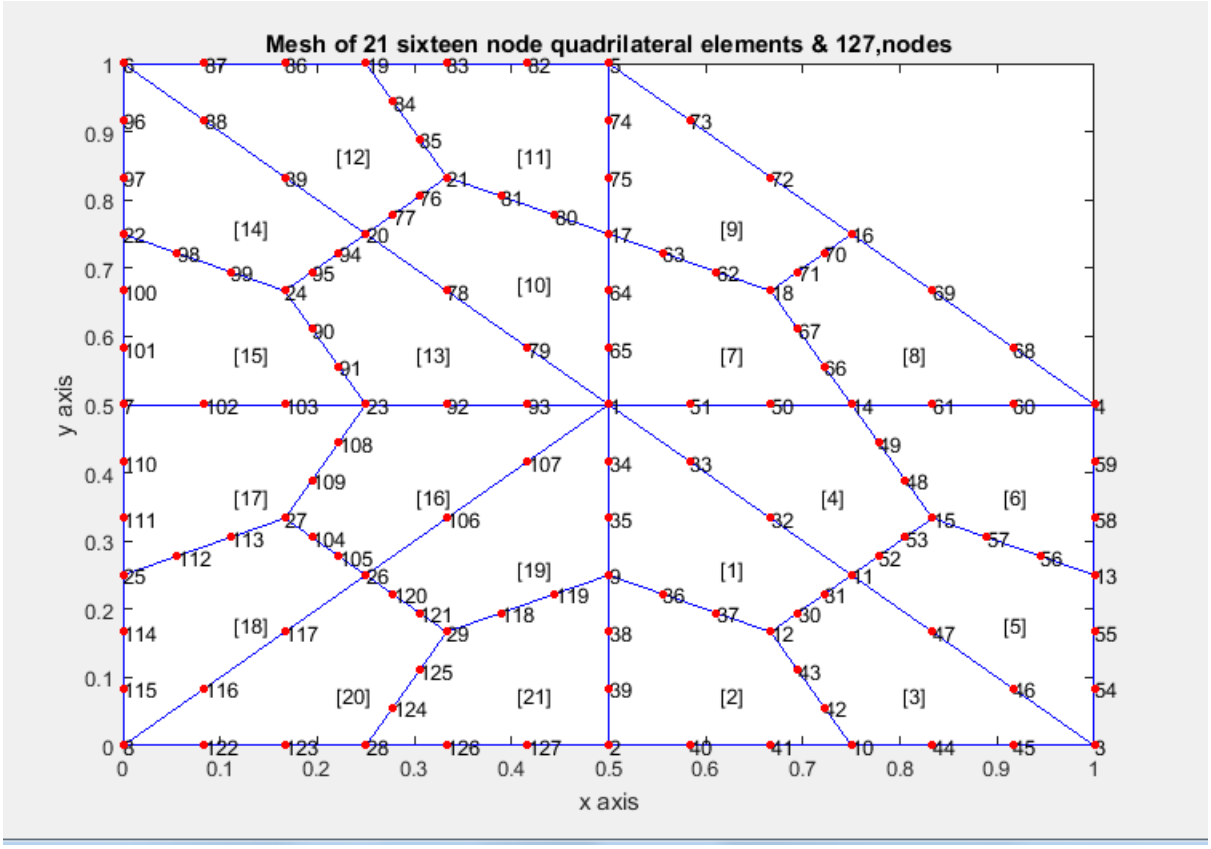


Fig 14a:Initial Mesh for a pentagonal domain seven triangles-12 noded quadrilaterals

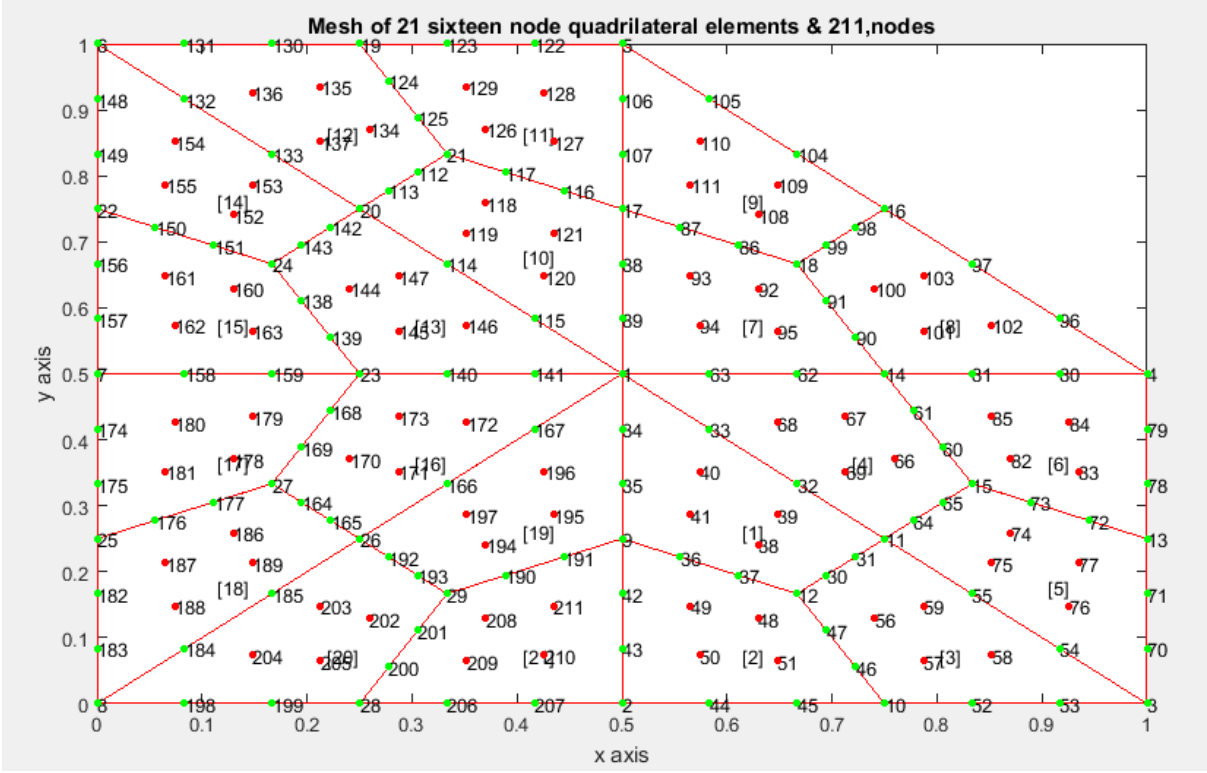


Fig 14b:Initial Mesh for a pentagonal domain seven triangles-16 noded quadrilaterals

Example 5 :a square domain with eight triangles(9-nodes)
x=([0; 0; 1/2;1/2;1/2; 0;-1/2;-1/2;-1/2])
y=([0;-1/2;-1/2; 0;1/2;1/2; 1/2; 0;-1/2])
We use this mesh to solve torsion of a square cross section.We would like to draw countour lines of Prandtl stress function over the entire domain.This is a case of Poisson equation with a constant right hand side(=-2)

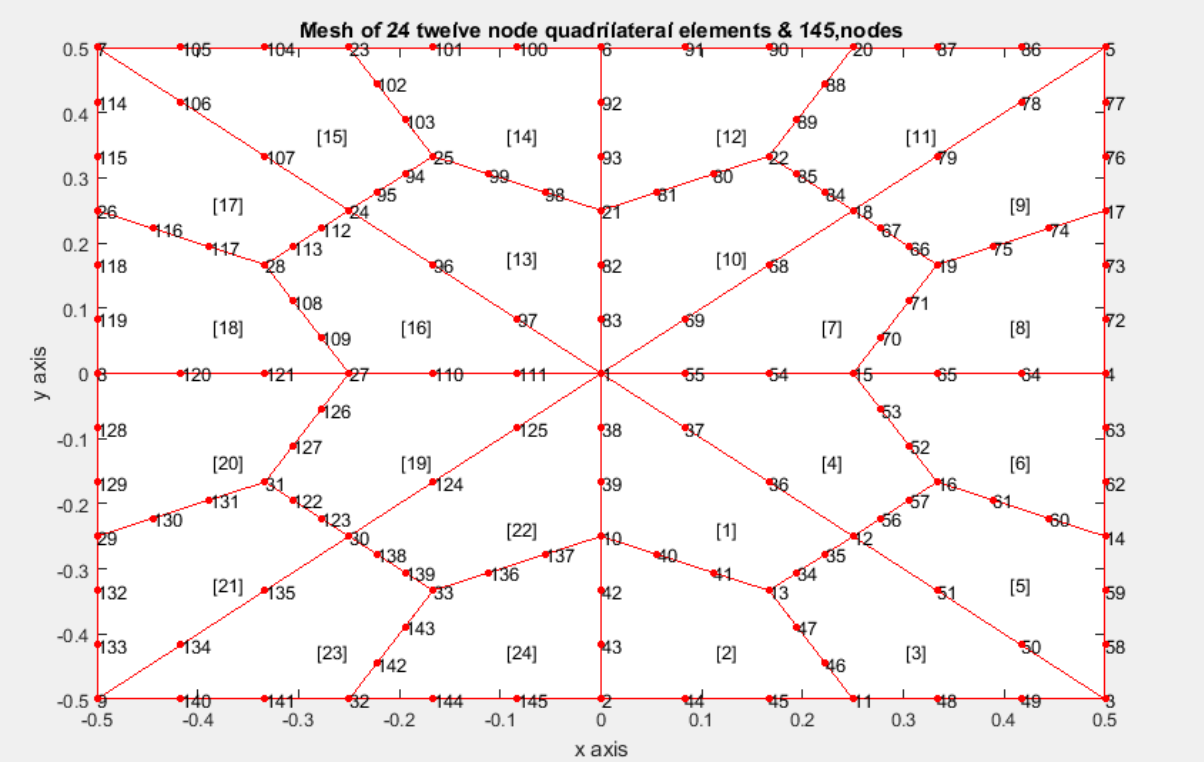


Fig 15a:Initial Mesh for a unit square domain eight triangles-12 noded quadrilaterals

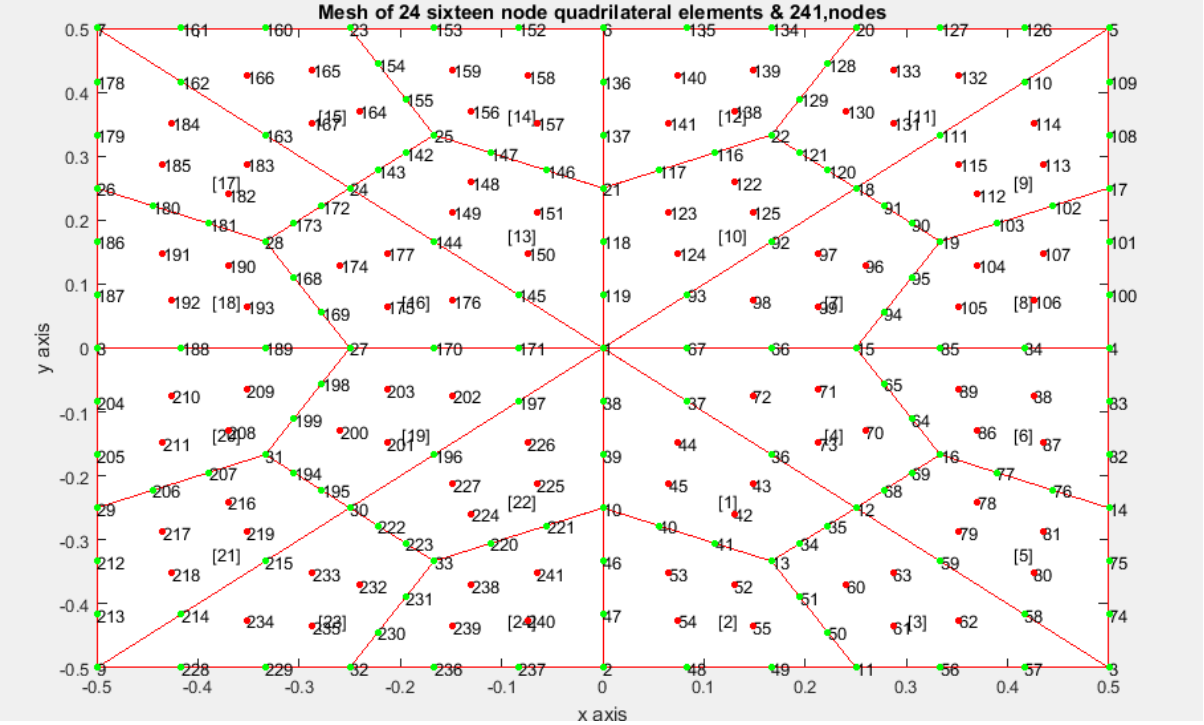


Fig 15b: Initial Mesh for a unit square domain with eight triangles-16 noded quadrilaterals

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:

$-\Delta \boldsymbol{u} = \boldsymbol{f},\ \boldsymbol{x}\epsilon\boldsymbol{\Omega} \subset \boldsymbol{\mathcal{R}^2}$ **.....(1)**

$\boldsymbol{u} = \boldsymbol{g},\ \boldsymbol{x}\epsilon\boldsymbol{\partial\Omega}$

.....(2)

Where Ω is a triangular or polygonal domain and Δ is the standard standard Laplace operator

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

$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + 2G\theta = 0$ in R ---------- (62a)

$\phi = 0$ on ∂R , the boundary of R ---------- (62b)

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

$.t_c = 2\iint_R \phi(x,y)dx\,dy$ ----------(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 torisonal constant which are expessed in terms of infinite series.This triangular domain is a right isosceles triangle and it was discretised by 8-noded special linear convex quadrilaterals 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

Exmple 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:

$\boldsymbol{\nabla^2 u} = \boldsymbol{-1},\ \boldsymbol{x}\epsilon\boldsymbol{\Omega} \subset \boldsymbol{\mathcal{R}^2}$

.....(9)

$\boldsymbol{u} = \boldsymbol{0},\ \boldsymbol{x}\epsilon\boldsymbol{\partial\Omega}$

.....(10)

Where Ω is a regular triangular or polygonal domain and ∇^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 t 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)

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

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

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

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

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

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

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

Example 4(with nonconstant smooth function as right hand side of Poisson equation)

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

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

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

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]quadrilateralmesh_over_arbitrarytriangle_q12automeshgen.m
[2]quadrilateralmesh_over_arbitrarytriangle_q16automeshgen.m
[3]quadrilateral_mesh4MOINEX_q12.m
[4]quadrilateral_mesh4MOINEX_q16LG.m
[5]D2LaplaceEquationQ12Ex3automeshgenNewContour.m$\surd^*$
[6]D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour.m$\surd^*$
[7]D2LaplaceEquationQ16Ex3automeshgenNewContour.m$\surd^*$
[8]D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour.m$\surd^*$
[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$\surd^*$
[18]D2PoissonEquationQ16MoinEx_MeshgridContourNew.m$\surd^*$
[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 $\surd*$ 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 discritise 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 $S^{i,j,e}$ in global (x, y) space can be expressed as a matrix triple product $P*(K^{i,j,e})*P^{T}*(2*\text{area of the arbitrary triangle in (x, y) space})$, in which P is a geometric properties matrix and $K^{i,j,e}$ is the product of global derivative integrals in (u, v) space, $(i,j = 1(1)12).\text{and}\ (i,j = 1(1)16)$. 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, Elservier, (2005)
2. Bathe K.J Finite Element Procedures, Prentice Hall, Englewood Cliffs, N J (1996)
3. Reddy J.N Finite Element Method, Third Edition, Tata Mc Graw-Hill (2005)
4. Burden R.L and Faires J.D Numerical Analysis, 9th Edition, Brooks/Cole, Cengage Learning (2011)
5. Stroud A.H and Secrest D , Gaussian quadrature formulas, Prentice Hall,Englewood Cliffs, N J, (1966)
6. Stoer J and Bulirsch R, Introduction to Numerical Analysis, Springer-Verlag, New York (1980)
7. Chung T.J Finite Element Analysis in Fluid Dynamics, pp. 191-199, Mc Graw Hill, Scarborough, C A , (1978)
8. Rathod H.T, Some analytical integration formulae for four node isoparametric element, Computer and structures 30(5), pp.1101-1109, (1988)
9. Babu D.K and Pinder G.F, Analytical integration formulae for linear isoparametric finite elements, Int. J. Numer. Methods Eng 20, pp.1153-1166
10. Mizukami A, Some integration formulas for four node isoparametric element Computer Methods in Applied Mechanics and Engineering. 59 pp. 111-21(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)

15. Yagawa G, Ye G.W and Yoshimura S, A numerical integration scheme for finite element method based on symbolic manipulation, International Journal for Numerical Methods in Engineering. 29, pp.1539-1549(1990)
16. Rathod H.T and Shafiqul Islam Md, Some pre-computed numeric arrays for linear convex quadrilateral finite elements, Finite Elements in Analysis and Design 38, pp. 113-136 (2001)
17. Hanselman D and Littlefield B, Mastering MATLAB 7 , Prentice Hall, Happer Saddle River, N J . (2005)
18. Hunt B.H, Lipsman R.L and Rosenberg J.M, A Guide to MATLAB for beginners and experienced users, Cambridge University Press (2005)
19. Char B, Geddes K, Gonnet G, Leong B, Monagan M and Watt S, First Leaves; A tutorial Introduction to Maple ∨ , New York : Springer–Verlag (1992)
20. Eugene D, Mathematica , Schaums Outlines Theory and Problems, Tata Mc Graw Hill (2001)
21. Ruskeepaa H, Mathematica Navigator, Academic Press (2009)
22. Timoshenko S.P and Goodier J.N, Theory of Elasticity, 3rd Edition, Tata Mc graw Hill Edition (2010)
23. Budynas R.G, Applied Strength and Applied Stress Analysis, Second Edition Tata Mc Graw Hill Edition (2011)
24. Roark R.J, Formulas for stress and strain, Mc Graw Hill, New York (1965)
25. Nguyen S.H , An accurate finite element formulation for linear elasticcalculations, Computers and Structures. 42, pp.707-711 (1992)
26. Rathod H.T, Venkatesh.B, Shivaram. K.T,Mamatha.T.M, Numerical Integration over polygonal domains using convex quadrangulation and Gauss Legendre Quadrature Rules, International Journal of Engineering and Computer Science, Vol. 2,issue 8,pp2576-2610(2013)
27. Rathod H.T, Rathod Bharath, Shivaram.K.T,Sugantha Devi.K, A new approach to automatic generation of all quadrilateral mesh for finite analysis, International Journal of Engineering and Computer Science, Vol. 2,issue 12,pp3488-3530(2013)
28. Rathod.H.T, Bharath Rathod, Shivaram K.T , H. Y. Shrivalli , Tara Rathod , K. Sugantha Devi,An explicit finite element integration scheme using automatic mesh generation technique for linear convex quadrilaterals over plane regions , international Journal of Engineering and Computer Science, Vol. 3,issue 4,pp5400-5435 (2014)
29. Rathod.H.T, Bharath Rathod, K.T.Shivaram,Sugantha Devi.K,Tara Rathod , An explicit finite element integration scheme for linear eight node convex quadrilaterals using automatic mesh generation technique over plane regions**,** international Journal of Engineering and Computer Science, Vol. 3,issue 4,pp5657-5713 (2014)
30. Rathod.H.T, Sugantha Devi.K ,Finite element solution of Poisson equation over polygonal domains using an explicit integration scheme and a novel auto mesh generation technique,,International Journal of Engineering and Computer Science, ISSN:2319-7242,Volume(5),issue 8,August(2016), pp. 17397-17481
31. Rathod.H.T, Sugantha Devi.K,Nagabhushana.C.S , Chudamani.H.M ,Finite element analysis of linear elastic torsion for regular polygons, International Journal of Engineering and Computer Science, ISSN:2319-7242,Volume(5),issue 10,October(2016), pp. 18413-1842
32. Rathod.H.T, Bharath Rathod, Sugantha Devi.K,Hariprasad.A.S, Finite element solution of Poisson Equation over Polygonal Domains using a novel auto mesh generation technique and an explicit integration scheme for eight node linear convex quadrilaterals, International Journal of Engineering and Computer Science, ISSN:2319-7242, Volume (6) Issue 10 October 2017, pp 22632-22787
33. H. T. Rathod,Md.Shafiqul. H. Y. Shrivall , Bharath Ratho K. Sugantha Devi, Finite element solution of Poisson Equation over Polygonal Domains using a novel auto mesh generation technique and an explicit integration scheme for nine node linear convex quadrilateral of Lagrange family**,** International Journal of Engineering and Computer Science, ISSN:2319-7242, Volume 6 Issue 11 November 2017, Page No. 22869-23058

APPENDIX-1

We now propose to compute the following integrals(see eqns(52a-b)) and they will be listed in Tables 1a-1i

$$\mathbf{K^{i,j,e}} = \iint_{\hat{Q}} \mathbf{G^{i,j,e}_{u,v}}\ \mathbf{du\ dv} = \begin{pmatrix} \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial u}\,\mathbf{dudv} & \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial v}\,\mathbf{dudv} \\ \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial u}\,\mathbf{dudv} & \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial v}\,\mathbf{dudv} \end{pmatrix} = \begin{pmatrix} \mathbf{K^e_{2i-1,2j-1}} & \mathbf{K^e_{2i-1,2j}} \\ \mathbf{K^e_{2i,2j-1}} & \mathbf{K^e_{2i,2j}} \end{pmatrix} \textbf{(say)} \quad \text{----------}$$

(52a-b)

We map $\hat{Q}$ into a 2-square We map $\hat{Q}$ into a $2-$ square, $-1 \le \xi, \eta \le 1$

in the natural parametric space (ξ, η) ***by using the bilinear transformation from*** (u, v) ***to*** (ξ, η)

This gives(see eqns(18-20))

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

Let us replace the Greek letters $\boldsymbol{\xi, \eta}$ by English letters **r,s** for computing the integrals by using MATLAB programming.With this assumption,we denote the entries of submatrix $\mathbf{K^{i,j,e}}$ **= intJdnidnjuvrs ,(i,j=1(1)9)**

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

$$\textbf{intJdnidnjuvrs} = \begin{pmatrix} [\mathbf{intJdnidnjuvrs}(1,1) & \mathbf{intJdnidnjuvrs}(1,2)] \\ [\mathbf{intJdnidnjuvrs}(2,1) & \mathbf{intJdnidnjuvrs}(2,2)] \end{pmatrix}$$

$K^e_{2i-1,2j-1} = \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial u}\,dudv = intJdnidnjuvrs(1,1)$, $K^e_{2i-1,2j} = \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial u}\frac{\partial N_j^e}{\partial v}\,dudv = intJdnidnjuvrs(1,2)$,

$K^e_{2i,2j-1} = \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial u}\,dudv = intJdnidnjuvrs(2,1)$, $K^e_{2i,2j} = \iint_{\hat{Q}} \frac{\partial N_i^e}{\partial v}\frac{\partial N_j^e}{\partial v}\,dudv = intJdnidnjuvrs(2,2)$

Tables of integral values for cubic order Serendipity and Lagrange elements

$$intJdnidnjuvrs = \begin{pmatrix} [intJdnidnjuvrs(1,1) & intJdnidnjuvrs(1,2)] \\ [intJdnidnjuvrs(2,1) & intJdnidnjuvrs(2,2)] \end{pmatrix}$$

(I)Cubic order Serendipity Elements

Table-1a

intJdnidnjuvrs,(i=1,j=1(1)12)

```
==========================================================================================
intJdn1dn1uvrs =([  1.878527434164174128122249654 6, 1.64496677534890315539341209663]
                 [ 1.64496677534890315539341209663,  1.878527434164174128122249654 6])
------------------------------------------------------------------------------------------------------------------------ -----
intJdn1dn2uvrs =([ 0.24392368058084020 79811424029, 0.25456419075959898129234164489];
                 [ 0.16706419075959898129234164489, 0.49412990575435517453974316 05])
------------------------------------------------------------------------------------------------------------------------ ----------
intJdn1dn3uvrs =([ 0.06514372856901812 9264176158009, -0.08063499940854168581340003167]
                 [ -0.08063499940854168581340003167, 0.06514372856901812 9264176158009])
------------------------------------------------------------------------------------------------------------------------ ----------
intJdn1dn4uvrs =([ 0.49412990575435517453974316 05, 0.16706419075959898129234164489]
                 [ 0.25456419075959898129234164489, 0.24392368058084020 79811424029])
------------------------------------------------------------------------------------------------------------------------ -------
intJdn1dn5uvrs =([ -1.91709736541803286033938722609, -0.50073891821327672848240784486]
                 [ -1.21323891821327672848240784486,   -0.48951593925070344936468574 9])
------------------------------------------------------------------------------------------------------------------------ --------
intJdn1dn6uvrs =([ 0.37765421933316862052540179057 1, -0.07885728723191145226859121413]
                 [  0.22114271276808854773140878587,   0.04710246204597543446965088 23])
------------------------------------------------------------------------------------------------------------------------ ----------
intJdn1dn7uvrs =([  -0.31236802364420048787164339097, -0.62259192547029207059718397227 1]
                 [ -0.62259192547029207059718397227 1,  -0.93264116705106716805029551279])
------------------------------------------------------------------------------------------------------------------------ ----------
intJdn1dn8uvrs =([  0.22334568260811342445267275929, 0.41545805218570060363476054995 7]
                 [ 0.41545805218570060363476054995 7,   0.32179538230835884627097507 03])
------------------------------------------------------------------------------------------------------------------------ ------------
intJdn1dn9uvrs =([   0.32179538230835884627097507 03, 0.41545805218570060363476054995 7]
                 [ 0.41545805218570060363476054995 7,  0.22334568260811342445267275929])
intJdn1dn10uvrs =([  -0.93264116705106716805029551279, -0.62259192547029207059718397227 1]
                  [ -0.62259192547029207059718397227 1,  -0.31236802364420048787164339097])
intJdn1dn11uvrs =([   0.04710246204597543446965088 23,  0.22114271276808854773140878587]
                   [ -0.07885728723191145226859121413, 0.37765421933316862052540179057 1])
intJdn1dn12uvrs =([   -0.48951593925070344936468574 9, -1.21323891821327672848240784486]
                   [ -0.50073891821327672848240784486, -1.91709736541803286033938722609])
------------------------------------------------------------------------------------------------------------------------ ----------
```

Table-1b

intJdnidnjuvrs,(i=2,j=1(1)12)

```
==========================================================================================
intJdn2dn1uvrs =([ 0.24392368058084020 79811424029, 0.16706419075959898129234164489]
                 [ 0.25456419075959898129234164489, 0.49412990575435517453974316 05])
intJdn2dn2uvrs =([ 0.92724744175233410562208079518 9, -0.15481341653816302516393935759]
                 [ -0.15481341653816302516393935759,   0.79471591647811679972930811 28])
intJdn2dn3uvrs =([ 0.51246142342538346249573123959 6, 0.12101159273523234674949318271 24]
                 [ 0.03351159273523234674949318271 24, 0.15660685205946358641731537602 6])
intJdn2dn4uvrs =([  0.16985339992712677743504195136, 0.17904505545000232695783146274 4]
                 [ 0.17904505545000232695783146274 4,  0.16985339992712677743504195136])
intJdn2dn5uvrs =([ 0.73745028457271616934934124617 1, 0.36072316479445118067281174443]
                 [  0.06072316479445118067281174443,  0.06761851600023926420478029 66]
intJdn2dn6uvrs =([ -1.89507344816323898310987069857 1, -0.32947059876161316868575194333]
                 [   0.38302940123838683131424805667,  -0.06101740375973425098670719 96])
intJdn2dn7uvrs =([ 0.08938175960791376107244162247 1,  -0.03294201189536998617617117 68]
                 [  -0.74544201189536998617617117 68, -1.32051059158026743632811545266])
```

intJdn2dn8uvrs =([0.066028785385703773347942390579, 0.132773165331780684038772441541]
[0.432773165331780684038772441541, 0.364521033839059598529804469259])
intJdn2dn9uvrs =([-0.820137977578506638706811693089, 0.102283035327104635406160239 3186]
[0.102283035327104635406160239 3186, 0.036404166972556546408672676 68])
intJdn2dn10uvrs =([0.144533345449155674413769328 57, -0.327320597226985932846449602 8286]
[-0.327320597226985932846449602 8286, -0.102776020518178556447216867 07])
intJdn2dn11uvrs =([-0.078982200036984917036376362 43, -0.212192568049325297169223465371]
[-0.212192568049325297169223465371, 0.062364758048803624976125527371])
intJdn2dn12uvrs =([-0.096686494922443392864432222 9, -0.006161011926712676752388618229]
[-0.006161011926712676752388618229, -0.661910533221541128478752051 37])
==========

Table-1c

intJdnidnjuvrs,(i=3,j=1(1)12)

==========-
intJdn3dn1uvrs =([0.065143728569018129264176158009, -0.080634999408541685813400031669]
[-0.080634999408541685813400031669, 0.065143728569018129264176158009])
intJdn3dn2uvrs =([0.512461423425383462495731239596, 0.033511592735232346794931827124]
([0.121011592735232346794931827124, 0.156606852059463586417315376026])
intJdn3dn3uvrs =([1.054266807002561599746314483 5803, 0.675139101205863811454268346 79]
([0.675139101205863811454268346 79, 1.054266807002561599746314483 5803])
intJdn3dn4uvrs =([0.156606852059463586417315376026, 0.121011592735232346794931827124]
([0.033511592735232346794931827124, 0.512461423425383462495731239596])
intJdn3dn5uvrs =([0.422209070544079358508960005 32, 0.338079890317619378517715472657]
([0.338079890317619378517715472657, 0.120923980597039893815264134 4])
intJdn3dn6uvrs =([-0.978113521941839530557046049 12, -0.271075459705479523507647905621]
[-0.271075459705479523507647905621, -0.165594360519239972922162075 9])
intJdn3dn7uvrs =([-0.097451565507866331982099391909, -0.002538103955586380338692247539]
[-0.302538103955586380338692247539, 0.463447558533213614696680089166])
intJdn3dn8uvrs =([0.099543897060373053107624738884, -0.645229970290446884286741304267]
[0.067270029709553115713258695733, -1.653443869822186862590758708 1])
intJdn3dn9uvrs =([-1.653443869822186862590758708 1, 0.067270029709553115713258695733]
[-0.645229970290446884286741304267, 0.099543897060373053107624738884])
intJdn3dn10uvrs =([0.463447558533213614696680089166, -0.302538103955586380338692247539]
[-0.002538103955586380338692247539, -0.097451565507866331982099391909])
intJdn3dn11uvrs =([-0.165594360519239972922162075896, -0.271075459705479523507647905621]
[-0.271075459705479523507647905621, -0.978113521941839530557046049 12])
intJdn3dn12uvrs =([0.120923980597039893815264134 4, 0.338079890317619378517715472657]
[0.338079890317619378517715472657, 0.422209070544079358508960005 32])
==========

Table-1d

intJdnidnjuvrs,(i=4,j=1(1)12)

---
intJdn4dn1uvrs =([0.494129905754355174539743160 5, 0.254564190759598912923416448 9]
[0.167064190759598912923416448 9, 0.243923680580840207981142402 9])
intJdn4dn2uvrs =([0.169853399927126777435041951 36, 0.179045055450002326957831462744]
[0.179045055450002326957831462744, 0.169853399927126777435041951 36])
intJdn4dn3uvrs =([0.156606852059463586417315376026, 0.033511592735232346794931827124]
[0.121011592735232346794931827124, 0.512461423425383462495731239596])
intJdn4dn4uvrs =([0.794715916478116799729308 1128, -0.154813416538163025163939357 6]
[-0.154813416538163025163939357 6, 0.927247441752334105622080795189])
intJdn4dn5uvrs =([-0.661910533221541128478752051 37, -0.006161011926712676752388618229]
[-0.006161011926712676752388618229, -0.096686494922443392864432222 9])
intJdn4dn6uvrs =([0.062364758048803624976125527371, -0.212192568049325297169223465371]
[-0.212192568049325297169223465371, -0.078982200036984917036376362 43])
intJdn4dn7uvrs =([-0.102776020518178556447216867 1, -0.327320597226985932846449602 8286]
[-0.327320597226985932846449602 8286, 0.144533345449155674413769328 57])
intJdn4dn8uvrs =([0.036404166972556546408672676 68, 0.102283035327104635406160239 3186])
[0.102283035327104635406160239 3186, -0.820137977578506638706811693089])
intJdn4dn9uvrs =([0.364521033839059598529804469259, 0.432773165331780684038772441541]
[0.132773165331780684038772441541, 0.066028785385703773347942390579])
intJdn4dn10uvrs =([-1.320510591580267436328115455 66, -0.745442011895369986176171176 8]
[-0.032942011895369986176171176 8, 0.089381759607913761072441622471])
intJdn4dn11uvrs =([-0.061017403759734250986707199 6, 0.383029401238386831314248056 67]

[-0.32947059876161316868575194333, -1.8950734481632389831098706986])
intJdn4dn12uvrs =([0.067618516000239264204780296 6, 0.060723164794451180672811744 43]
[0.360723164794451180672811744 43, 0.737450284572716169349341246171])

Table-1e
intJdnidnjuvrs,(i=5,j=1(1)12)

---

intJdn5dn1uvrs =([-1.917097365418032860339387226 09, -1.213238918213276728482407844 86]
[-0.500738918213276728482407844 86, -0.489515939250703449364685749])
intJdn5dn2uvrs =([0.737450284572716169349341246171, 0.060723164794451180672811744 43]
[0.360723164794451180672811744 43, 0.067618516000239264204780296 6])
intJdn5dn3uvrs =([0.422209070544079358508960005 32, 0.338079890317619378517715472657]
[0.338079890317619378517715472657, 0.120923980597039893815264134 4])
intJdn5dn4uvrs =([-0.661910533221541128478752051 37, -0.006161011926712676752388618229]
[-0.006161011926712676752388618229, -0.096686494922443392864432222 9])
intJdn5dn5uvrs =([4.200274290514790812114736470 17, 1.343867918947780845204424775 1]
[1.343867918947780845204424775 1, 0.831814504364695263991287102])
intJdn5dn6uvrs =([-2.997024944510102423832667109 89, -0.134987191916702554874616021]
[-1.147487191916702554874616021, -0.265306180626066014206434443])
intJdn5dn7uvrs =([-0.229689408692304476622709615 4, -0.139265536665219390197130935 31]
[-0.139265536665219390197130935 31, -0.032853529098637720977358297 79])
intJdn5dn8uvrs =([-0.036080256653866032265394083 51, -0.153999664491317548397943015 2]
[-0.153999664491317548397943015 2, -0.068297320390716358168351330 8])
intJdn5dn9uvrs =([-1.540413010424596444521561548 33, -0.817192205806162219949466952 33]
[-0.817192205806162219949466952 33, -0.265810825953669114753564275 3])
intJdn5dn10uvrs =([1.779490640629861611176434525 13, 0.468143460469334316998092911]
[0.468143460469334316998092911, -0.221728776266355628659448483 51])
intJdn5dn11uvrs =([0.022488277344760679680561940 46, 0.093890517833399167630961583 71]
[0.093890517833399167630961583 71, 0.199539110232382521752505822])
intJdn5dn12uvrs =([0.220302955314234735230437446 7, 0.160139576656806226299468991 9]
[0.160139576656806226299468991 9, 0.220302955314234735230437446 7])

Table-1f
intJdnidnjuvrs, i=6,j=1 (1)12)

---

intJdn6dn1uvrs =([0.377654219333168620525401790571, 0.221142712768088547731408785 87]
[-0.078857287231911452268591214 13, 0.047102462045975434469650882 3])
intJdn6dn2uvrs =([-1.895073448163238983109870698571, 0.383029401238386831314248056 67]
[-0.329470598761613168685751943 33, -0.061017403759734250986707199 6])
intJdn6dn3uvrs =([-0.978113521941839530557046049 12, -0.271075459705479523507647905621]
[-0.271075459705479523507647905621, -0.165594360519239972922162075 9])
intJdn6dn4uvrs =([0.062364758048803624976125527371, -0.212192568049325297169223465371]
[-0.212192568049325297169223465371, -0.078982200036984917036376362 43])
intJdn6dn5uvrs =([-2.997024944510102423832667109 89, -1.147487191916702554874616021]
[-0.134987191916702554874616021, -0.265306180626066014206434443])
intJdn6dn6uvrs =([4.542423957138916899011049125 84, 0.609286969852031114672828247 7]
[0.609286969852031114672828247 7, 0.443331612234228602914621619])
intJdn6dn7uvrs =([-0.218387508330363023666003821 97, -0.101460477169102579795843934 3]
[-0.101460477169102579795843934 3, 0.078281283861351532608066273 53])
intJdn6dn8uvrs =([-0.042271611042919621472681592 04, -0.170055359248205194945025268 5]
[-0.170055359248205194945025268 5, 0.066296433804103064946378510343])
intJdn6dn9uvrs =([1.985855365274808473372126605957, 0.174957084563132440132039516 83]
[0.174957084563132440132039516 83, -0.298822005552976959188963685 7])
intJdn6dn10uvrs =([-1.071217433095080466489312963 5, 0.293784086697631753006223399 94]
[0.293784086697631753006223399 94, 0.177971024149118890230991178 9])
intJdn6dn11uvrs =([0.034251057055463909490373363 41, 0.126180283136145295804647002 74]
[0.126180283136145295804647002 74, 0.034251057055463909490373363 41])
intJdn6dn12uvrs =([0.199539110232382521752505822, 0.093890517833399167630961583 71]
[0.093890517833399167630961583 71, 0.022488277344760679680561940 46])

Table-1g
intJdnidnjuvrs, i=7,j=1 (1)12)

---

intJdn7dn1uvrs =([-0.312368023644200487871643390 97, -0.622591925470292070597183972271]
[-0.622591925470292070597183972271, -0.932641167051067168050295512 79])
intJdn7dn2uvrs =([0.089381759607913761072441622471, -0.745442011895369986176171176 8]
[-0.032942011895369986176171176 8, -1.320510591580267436328115452 66])
intJdn7dn3uvrs =([-0.097451565507866331982099391909, -0.302538103955586380338692247539]
[-0.002538103955586380338692247539, 0.463447558533213614696680089166])
intJdn7dn4uvrs =([-0.102776020518178556446721686 71, -0.327320597226985932846449602828 6]
[-0.327320597226985932846449602828 6, 0.144533345449155674413769328 57])
intJdn7dn5uvrs =([-0.229689408692304476622709615 4, -0.139265536665219390197130935 31]
[-0.139265536665219390197130935 31, -0.032853529098637720977358297 79])
intJdn7dn6uvrs =([-0.218387508330363023666003821 97, -0.101460477169102579795843934 3]
[-0.101460477169102579795843934 3, 0.078281283861351532608066273 53])
intJdn7dn7uvrs =([0.943002697766520312484096704 9, 1.054490113764698836662802085186]
[1.054490113764698836662802085186, 3.175952316952668112535895550 38])
intJdn7dn8uvrs =([-0.234114965649640845821295009 14, 0.022996512009548291644560497057]
[-0.989503487990451708355439502943, -2.315691612905610821903565142 73])
intJdn7dn9uvrs =([0.109635501293948718617151410 79, 0.193694588216914851097944520 82]
[0.193694588216914851097944520 82, -0.065316097486994600347934059 82])
intJdn7dn10uvrs =([0.096525285791407668665735662 68, 0.205509891224428290541848454 97]
[0.205509891224428290541848454 97, 0.096525285791407668665735662 68])
intJdn7dn11uvrs =([0.177971024149118890230991178 9, 0.293784086697631753006223399 94]
[0.293784086697631753006223399 94, -1.071217433095080466489312963 5])
intJdn7dn12uvrs =([-0.221728776266355628659448483 51, 0.468143460469334316998092911]
[0.468143460469334316998092911, 1.779490640629861611176434525 13])

Table-1h
intJdnidnjuvrs, i=8,j=1 (1)12)

---

intJdn8dn1uvrs =([0.223345682608113424452672759 29, 0.415458052185700603634760549957]
[0.415458052185700603634760549957, 0.321795382308358846270975070 3])
intJdn8dn2uvrs =([0.066028785385703773347942390579, 0.432773165331780684038772441541]
[0.132773165331780684038772441541, 0.364521033839059598529804469259])
intJdn8dn3uvrs =([0.099543897060373053107624738884, 0.067270029709553115713258695733]
[-0.645229970290446884286741304267, -1.653443869822186862590758708 1])
intJdn8dn4uvrs =([0.036404166972556546408672676 68, 0.102283035327104635406160239318 6]
[0.102283035327104635406160239318 6, -0.820137977578506638706811693089])
intJdn8dn5uvrs =([-0.036080256653866032265394083 51, -0.153999664491317548397943015 2]
[-0.153999664491317548397943015 2, -0.068297320390716358168351330 8])
intJdn8dn6uvrs =([-0.042271611042919621472681592 04, -0.170055359248205194945025268 5]
[-0.170055359248205194945025268 5, 0.066296433804103064946378510343])
intJdn8dn7uvrs =([-0.234114965649640845821295009 14, -0.989503487990451708355439502943]
[0.022996512009548291644560497057, -2.315691612905610821903565142 73])
intJdn8dn8uvrs =([0.573992876539626714370549258 66, 0.510536435455242956618237924447]
[0.510536435455242956618237924447, 3.606779720827644761992241475244])
intJdn8dn9uvrs =([-0.056899646226306337837629118 9, 0.233778326746707385006700850 18]
[0.233778326746707385006700850 18, -0.056899646226306337837629118 9])
intJdn8dn10uvrs =([-0.065316097486994600347934059 82, 0.193694588216914851097944520 82]
[0.193694588216914851097944520 82, 0.109635501293948718617151410 79])
intJdn8dn11uvrs =([-0.298822005552976959188963685 7, 0.174957084563132440132039516 83]
[0.174957084563132440132039516 83, 1.985855365274808473372126605957])
intJdn8dn12uvrs =([-0.265810825953669114753564275 33, -0.817192205806162219949466952 33]
[-0.817192205806162219949466952 33, -1.540413010424596444521561548 33])

Table-1i
intJdnidnjuvrs, i=9,j=1 (1)12)

---

intJdn9dn1uvrs =([0.321795382308358846270975070 3, 0.415458052185700603634760549957]

[0.41545805218570060363476054995 7, 0.22334568260811342445267275929])
intJdn9dn2uvrs =([-0.82013797757850663870681169308 9, 0.10228303532710463540616023931 86]
[0.10228303532710463540616023931 86, 0.03640416697255654640867267668])
intJdn9dn3uvrs =([-1.65344386982218686259075870 81, -0.64522997029044688428674130426 7]
[0.06727002970955311571325869573 3, 0.09954389706037305310762473888 4])
intJdn9dn4uvrs =([0.36452103383905959852980446925 9, 0.13277316533178068403877244154 1]
[0.43277316533178068403877244154 1, 0.06602878538570377334794239057 9])
intJdn9dn5uvrs =([-1.54041301042459644452156154833, -0.81719220580616221994946695233]
[-0.81719220580616221994946695233, -0.26581082595366911475356427 53])
intJdn9dn6uvrs =([1.98585536527480847337212660595 7, 0.17495708456313244013203951683]
[0.17495708456313244013203951683, -0.29882200555297695918896368 57])
intJdn9dn7uvrs =([0.10963550129394871861715141079, 0.19369458821691485109794452082]
[0.19369458821691485109794452082, -0.06531609748699460034793405982])
intJdn9dn8uvrs =([-0.05689964622630633783762911 89, 0.23377832674670738500670085018]
[0.23377832674670738500670085018, -0.05689964622630633783762911 89])
intJdn9dn9uvrs =([3.60677972082764476199224147524 4, 0.51053643545524295661823792444 7]
[0.51053643545524295661823792444 7, 0.57399287653962671437054925866])
intJdn9dn10uvrs =([-2.31569161290561082190356514272 7, 0.02299651200954829164456049705 7]
[-0.98950348799045170835543950294 3, -0.23411496564964084582129500914])
intJdn9dn11uvrs =([0.06629643380410306494637851034 3, -0.17005535924820519494502526 85]
[-0.17005535924820519494502526 85, -0.04227161104291962147268159204])
intJdn9dn12uvrs =([-0.06829732039071635816835133077 14, -0.15399966449131754839794301 52]
[-0.15399966449131754839794301 52, -0.03608025665386603226539408351])

Table-1j
intJdnidnjuvrs, i=10,j=1 (1)12)

---

intJdn10dn1uvrs =([-0.93264116705106716805029551279, -0.62259192547029207059718397227 1]
[-0.62259192547029207059718397227 1, -0.31236802364420048787164339097])
intJdn10dn2uvrs =([0.14453334544915567441376932857, -0.32732059722698593284644960282 86]
[-0.32732059722698593284644960282 86, -0.10277602051817855644721686707])
intJdn10dn3uvrs =([0.46344755853321361469668008916 6, -0.00253810395558638033869224753 9]
[-0.00253810395558638033869224753 9, -0.09745156550786633198209939190 9])
intJdn10dn4uvrs =([-1.32051059158026743632811545266, -0.03294201189536998617617117 68]
[-0.74544201189536998617617117 68, 0.08938175960791376107244162247 1])
intJdn10dn5uvrs =([1.77949064062986161117643452513, 0.46814346046933431699809291 1]
[0.46814346046933431699809291 1, -0.22172877626635562865944848351])
intJdn10dn6uvrs =([-1.07121743309508046648931296 35, 0.29378408669763175300622339994]
[0.29378408669763175300622339994, 0.17797102414911889023099117 89])
intJdn10dn7uvrs =([0.09652528579140766866573566268, 0.20550989122442829054184845497]
[0.20550989122442829054184845497, 0.09652528579140766866573566268])
intJdn10dn8uvrs =([-0.06531609748699460034793405982, 0.19369458821691485109794452082]
[0.19369458821691485109794452082, 0.10963550129394871861715141079])
intJdn10dn9uvrs =([-2.31569161290561082190356514272 7, -0.98950348799045170835543950294 3]
[0.02299651200954829164456049705 7, -0.23411496564964084582129500914])
intJdn10dn10uvrs =([3.17595231695266811253589555038, 1.05449011376469883666280208518 6]
[1.05449011376469883666280208518 6, 0.94300269776652031248409670487])
intJdn10dn11uvrs =([0.07828128386135153260806627353, -0.10146047716910257979584393 43]
[-0.10146047716910257979584393 43, -0.21838750833036302366600382197])
intJdn10dn12uvrs =([-0.03285352909863772097735829779, -0.13926553666521939019713093531]
[-0.13926553666521939019713093531, -0.22968940869230447662270961 54])

Table-1k
intJdnidnjuvrs, i=11,j=1 (1)12)

---

intJdn11dn1uvrs =([0.04710246204597543446965088 23, -0.07885728723191145226859121413]
[0.22114271276808854773140878587, 0.37765421933316862052540179057 1])
intJdn11dn2uvrs =([-0.07898220003698491703637636243, -0.21219256804932529716922346537 1]
[-0.21219256804932529716922346537 1, 0.06236475804880362497612552737 1])
intJdn11dn3uvrs =([-0.16559436051923997292216207589 6, -0.27107545970547952350764790562 1]
[-0.27107545970547952350764790562 1, -0.97811352194183953055704604912])
intJdn11dn4uvrs =([-0.06101740375973425098670719 96, -0.32947059876161316868575194333]
[0.38302940123838683131424805667, -1.89507344816323898310987069 86])

intJdn11dn5uvrs =([0.022488277344760679680561940 46, 0.093890517833399167630961583 71]
[0.093890517833399167630961583 71, 0.199539110232382521752505822])
intJdn11dn6uvrs =([0.034251057055463909490373363 41, 0.126180283136145295804647002 74]
[0.126180283136145295804647002 74, 0.034251057055463909490373363 41])
intJdn11dn7uvrs =([0.177971024149118890230991178 9, 0.293784086697631753006223399 94]
[0.293784086697631753006223399 94, -1.071217433095080466489312963 5])
intJdn11dn8uvrs =([-0.298822005552976959188963685 7, 0.174957084563132440132039516 83]
[0.174957084563132440132039516 83, 1.985855365274808473372126605957])
intJdn11dn9uvrs =([0.066296433804103064946378510343, -0.170055359248205194945025 2685]
[-0.170055359248205194945025 2685, -0.042271611042919621472681592 04])
intJdn11dn10uvrs =([0.078281283861351532608066273 53, -0.101460477169102579795843934 3]
[-0.101460477169102579795843934 3, -0.218387508330363023666003821 97])
intJdn11dn11uvrs =([0.443331612234228602914621619, 0.609286969852031114672828247 7]
[0.609286969852031114672828247 7, 4.542423957138916899011049125 84])
intJdn11dn12uvrs =([-0.265306180626066014206434443, -0.134987191916702554874616021]
[-1.147487191916702554874616021, -2.997024944510102423832667109 89])

Table-1l
intJdnidnjuvrs, i=12,j=1(1)12)

---

intJdn12dn1uvrs =([-0.489515939250703449364685749, -0.500738918213276728482407844 86]
[-1.213238918213276728482407844 86, -1.917097365418032860339387226 09])
intJdn12dn2uvrs =([-0.096686494922443392864432222 9, -0.006161011926712676767523886182 29]
[-0.006161011926712676767523886182 29, -0.661910533221541128478752051 37])
intJdn12dn3uvrs =([0.120923980597039893815264134 4, 0.338079890317619378517715472657]
[0.338079890317619378517715472657, 0.422209070544079358508960005 32])
intJdn12dn4uvrs =([0.067618516000239264204780296 6, 0.360723164794451180672811744 43]
[0.060723164794451180672811744 43, 0.737450284572716169349341246171])
intJdn12dn5uvrs =([0.220302955314234735230437446 7, 0.160139576656806229629946899 2]
[0.160139576656806229629946899 2, 0.220302955314234735230437446 7])
intJdn12dn6uvrs =([0.199539110232382521752505822, 0.093890517833399167630961583 71]
[0.093890517833399167630961583 71, 0.022488277344760679680561940 46])
intJdn12dn7uvrs =([-0.221728776266355628659448483 51, 0.468143460469334316998092911]
[0.468143460469334316998092911, 1.779490640629861611176434525 13])
intJdn12dn8uvrs =([-0.265810825953669911475356427533, -0.817192205806162219949466952 33]
[-0.817192205806162219949466952 33, -1.540413010424596444521561548 33])
intJdn12dn9uvrs =([-0.068297320390716358168351330 7714, -0.153999664491317548397943015 2]
[-0.153999664491317548397943015 2, -0.036080256653866032265394083 51])
intJdn12dn10uvrs =([-0.032853529098637720977358297 79, -0.139265536665219390197130935 31]
[-0.139265536665219390197130935 31, -0.229689408692304476622709615 4])
intJdn12dn11uvrs =([-0.265306180626066014206434443, -1.147487191916702554874616021]
[-0.134987191916702554874616021, -2.997024944510102423832667109 89])
intJdn12dn12uvrs =([0.831814504364695263991287102, 1.343867918947780845204424775 1]
[1.343867918947780845204424775 1, 4.200274290514790812114736470 17])

(II)Cubic order Lagrange Elements

Table-2a
intJdnidnjuvrs, i=1,j=1 (1)16)

---

intJdn1dn1uvrs =([0.701681093145454720870380087, 0.675893220631106832241358344]
[0.675893220631106832241358344, 0.701681093145454720870380087])
intJdn1dn2uvrs =([-0.026882286462947431291882712 5, 0.051054447959458125511491745]
[-0.036445552040541874488508255 5, 0.032212211804107075133583727])
intJdn1dn3uvrs =([-0.008630327499228376151969184 35, -0.010496176140285592562407773 9]
[-0.010496176140285592562407773 9, -0.008630327499228376151969184 35])
intJdn1dn4uvrs =([0.032212211804107075133583727, -0.036445552040541874488508255 5]
[0.051054447959458125511491745, -0.026882286462947431291882712 5])
intJdn1dn5uvrs =([-0.412633054117600230716206898, 0.268945525984776981543022533]
[-0.443554474015223018456977467, 0.157871624541338704794557829])
intJdn1dn6uvrs =([0.117250394537401747967460496 7, -0.142592079476496670389254682]
[0.157407920523503329610745318, -0.070027089525016846364967113])
intJdn1dn7uvrs =([-0.093144679402720430660566560 5, -0.104911199388184100700597507 5]

[-0.10491119938818410070059750 75, -0.092245188408519862030004704])
intJdn1dn8uvrs =([0.033518279789733485446869505, 0.039281153486431255496910516]
[0.039281153486431255496910516, 0.031879071145317587250700 8895])
intJdn1dn9uvrs =([0.031879071145317587250700 8895, 0.039281153486431255496910515 5]
[0.039281153486431255496910515 5, 0.033518279789733485446869504 5])
intJdn1dn10uvrs =([-0.092245188408519862030004703 5, -0.10491119938818410070059750 75]
[-0.10491119938818410070059750 75, -0.093144679402720430660566560 5])
intJdn1dn11uvrs =([-0.070027089525016846364967113, 0.157407920523503329610745318]
[-0.142592079476496670389254682, 0.117250394537401747967460496 7])
intJdn1dn12uvrs =([0.157871624541338704794557829, -0.443554474015223018456977467]
[0.268945525984776981543022533, -0.412633054117600230716206898])
intJdn1dn13uvrs =([-0.923677839101831305550677102, -1.007032454890500463075642 3]
[-1.007032454890500463075642 3, -0.923677839101831305550677102])
intJdn1dn14uvrs =([0.336851990222722864094172519, 0.381671404352391717716050003]
[0.381671404352391717716050003, 0.338459011543472358906908599])
intJdn1dn15uvrs =([-0.122483212211684061698359345, -0.145263095437075394958553495]
[-0.145263095437075394958553495, -0.122483212211684061698359345])
intJdn1dn16uvrs =([0.338459011543472358906908599, 0.381671404352391717716050003]
[0.381671404352391717716050003, 0.336851990222722864094172 52])

Table-2b
intJdnidnjuvrs, i=2,j=1 (1)16)

---

intJdn2dn1uvrs =([-0.026882286462947431291882712 5, -0.036445552040541874488508255 5]
[0.051054447959458125511491744 5, 0.032212211804107075133583727])
intJdn2dn2uvrs =([0.211971597991981069039609973, -0.153046574826884110662338533 64]
[-0.153046574826884110662338533 64, 0.194935467116933386760696773 78])
intJdn2dn3uvrs =([0.050281043496617803780849234, 0.041548301213886137681966678 29]
[-0.045951698786113862318033321 71, -0.021066690833006013563865640 82])
intJdn2dn4uvrs =([-0.000459867371696942288992267 5, 0.006932774692844569729564652 2]
[0.006932774692844569729564652 2, -0.000459867371696942288992267 5])
intJdn2dn5uvrs =([0.135316485729802334623950907, 0.149351885457747745233522 63]
[-0.150648114542252254766477370 1, -0.035607003145149770487295207])
intJdn2dn6uvrs =([-0.484060336356140646761807119, -0.322540791632477410079167085 1]
[0.389959208367522589920832914 9, 0.137829701836470761484853490 7])
intJdn2dn7uvrs =([0.283468713180315926702327 99, 0.261652146923052718725140201 5]
[-0.450847853076947281274859798, -0.264647289956764019523548 11])
intJdn2dn8uvrs =([-0.114137962266763606747720 48, -0.129354487376281056516272933]
[0.170645512623718943483727067 04, 0.080327166131836665212584867 1])
intJdn2dn9uvrs =([-0.035328033479583724019581655, 0.057501002240708994321170913 7]
[0.057501002240708994321170913 7, -0.010765639631107266918194210 9])
intJdn2dn10uvrs =([0.005100403817923470925719506, -0.024761717999877543769671456 9])
[-0.024761717999877543769671456 9, 0.000327370706036359563493129])
intJdn2dn11uvrs =([-0.008049498872369479004123, -0.023960991455128535818634836 1]
[-0.023960991455128535818634836 1, 0.002353045794858365263235082 7])
intJdn2dn12uvrs =([0.022034662628595693400301698 6, 0.049548766326640713476103384 2]
[0.049548766326640713476103384 2, -0.020270550705857146813452057])
intJdn2dn13uvrs =([-0.062460304818179354667931657, -0.196130238975238548948111488]
[-0.196130238975238548948111488, 0.018066447765168896456321824])
intJdn2dn14uvrs =([-0.005654544717914146990072595, 0.427622370384584419718474494 1]
[0.427622370384584419718474494 1, -0.153931015713211339133137848 6])
intJdn2dn15uvrs =([0.008033238055969811770258 93, -0.194216973763714354534537517 5]
[-0.194216973763714354534537517 5, 0.041681314367425998111696911 9])
intJdn2dn16uvrs =([0.020826689444389221529093246 9, 0.086300080830678135931299152]
[0.086300080830678135931299152, -0.000984668166045009257980461 6])

Table-2c
intJdnidnjuvrs, i=3,j=1 (1)16)

---

intJdn3dn1uvrs =([-0.008630327499228376151969184 35, -0.010496176140285592562407773 9]
[-0.010496176140285592562407773 9, -0.008630327499228376151969184 35])
intJdn3dn2uvrs =([0.050281043496617803780849234, -0.045951698786113862318033321 71]
[0.041548301213886137681966678 29, -0.021066690833006013563865640 82])

intJdn3dn3uvrs =([0.29345645824851617725556814 91, 0.27260539568220496940415201629]
[0.27260539568220496940415201629, 0.29345645824851617725556814 91])
intJdn3dn4uvrs =([-0.02106669083300601356386564082, 0.04154830121388613768196667829]
[-0.04595169878611386231803332171, 0.05028104349661780378084923 4])
intJdn3dn5uvrs =([0.03122589745888049158791859 04, 0.03066254874047117069795458 81]
[0.03066254874047117069795458 81, 0.01203062469906479395309804256])
intJdn3dn6uvrs =([-0.09563153300224068809754350 1, -0.07011142235256930991307362]
[-0.07011142235256930991307362, -0.02290194853292015105870834477])
intJdn3dn7uvrs =([-0.06683028025765907274929363 9, 0.15714640889191176538987417817]
[-0.14285359110808823461012582183, 0.09583107259151565959073070 62])
intJdn3dn8uvrs =([0.21711976690295516640568278 5, -0.36432561947182595155951418839]
[0.34817438052817404844048581161, -0.35254272491668073283298402])
intJdn3dn9uvrs =([-0.35254272491668073283298402, 0.34817438052817404844048581161]
[-0.36432561947182595155951411884, 0.21711976690295516640568278 5])
intJdn3dn10uvrs =([0.09583107259151565959073070 62, -0.14285359110808823461012582183]
[0.15714640889191176538987417817, -0.06683028025765907274929363 9])
intJdn3dn11uvrs =([-0.02290194853292015105870834477, -0.07011142235256930991307362]
[-0.07011142235256930991307362, -0.09563153300224068809754350 1])
intJdn3dn12uvrs =([0.01203062469906479395309804256, 0.03066254874047117069795458 81]
[0.03066254874047117069795458 81, 0.03122589745888049158791859 04])
intJdn3dn13uvrs =([-0.04401071284291265210755706 44, -0.09641597529566077118991423 76]
[-0.09641597529566077118991423 76, -0.04401071284291265210755706 44])
intJdn3dn14uvrs =([0.13699946287857933402138920 1, 0.22721341364900971981665726 36]
[0.22721341364900971981665726 36, 0.09411855580991184417130217 44])
intJdn3dn15uvrs =([-0.31944866420139358420459747 9, -0.53496050558802566987955980517]
[-0.53496050558802566987955980517, -0.31944866420139358420459747 9])
intJdn3dn16uvrs =([0.09411855580991184417130217 44, 0.22721341364900971981665726 36]
[0.22721341364900971981665726 36, 0.13699946287857933402138920 1])

Table-2d

intJdnidnjuvrs, i=4,j=1 (1)16)

---

intJdn4dn1uvrs =([0.03221221180410707513358372 7, 0.05105444795945812551149174 5]
[-0.03644555204054187448850825 55, -0.02688228646294743129188271 25])
intJdn4dn2uvrs =([-0.00045986737169694228899226 75, 0.00693277469284456972956465 22]
[0.00693277469284456972956465 22, -0.00045986737169694228899226 75])
intJdn4dn3uvrs =([-0.02106669083300601356386564082, -0.04595169878611386231803332171]
[0.04154830121388613768196667829, 0.05028104349661780378084923 4])
intJdn4dn4uvrs =([0.19493546711693338676069677 378, -0.15304657482688411066233853364]
[-0.15304657482688411066233853364, 0.21197159799198106903960997 3])
intJdn4dn5uvrs =([-0.02027055070585714681345205 7, 0.04954876632664071347610338 42]
[0.04954876632664071347610338 42, 0.02203466262859569340030169 9])
intJdn4dn6uvrs =([0.00235304579485836526323508 3, -0.02396099145512853581863483 6]
[-0.02396099145512853581863483 6, -0.00804949887236947900412300 02])
intJdn4dn7uvrs =([0.00032737070603635956349312 8961, -0.02476171799987754376967145 69]
[-0.02476171799987754376967145 69, 0.00510040381792347092571950 6])
intJdn4dn8uvrs =([-0.01076563963110726691819421 09, 0.05750100224070899432117091 37]
[0.05750100224070899432117091 37, -0.03532803347958372401958165 5])
intJdn4dn9uvrs =([0.08032716613183666521258486 71, 0.17064551262371894348372706 704]
[-0.12935448737628105651627293 296, -0.11413796226676360674772048])
intJdn4dn10uvrs =([-0.26464728995676401952354811, -0.45084785307694728127485979 85]
[0.26165214692305271872514020 15, 0.28346871318031592670232799])
intJdn4dn11uvrs =([0.13782970183647076148485349 07, 0.38995920836752258992083291 49]
[-0.32254079163247741007916708 51, -0.48406033635614064676180711 9])
intJdn4dn12uvrs =([-0.03560700314514977048729520 7, -0.15064811454225225476647737 01]
[0.14935188545774774523352262 99, 0.13531648572980233462395090 7])
intJdn4dn13uvrs =([0.01806644776516889645632182 4, -0.19613023897523854894811148 8]
[-0.19613023897523854894811148 8, -0.06246030481817935466793166])
intJdn4dn14uvrs =([-0.00098466816604509257980461 6, 0.08630008083067813593129915 2]
[0.08630008083067813593129915 2, 0.02082668944438922152909324 69])
intJdn4dn15uvrs =([0.04168131436742599811169691 19, -0.19421697376371435453453751 75]
[-0.19421697376371435453453751 75, 0.00803323805596981177025893])
intJdn4dn16uvrs =([-0.15393101571321133913313786 86, 0.42762237038458441971847449 41]
[0.42762237038458441971847449 41, -0.00565454471791414699007259 5])

Table-2e

intJdnidnjuvrs, i=5,j=1 (1)16)

```
intJdn5dn1uvrs =(  [ -0.41263305411760023071620689 8, -0.44355447401522301845697746 7]
                   [  0.26894552598477698154302253 3,  0.15787162454133870479455782 9])
intJdn5dn2uvrs =( [ 0.13531648572980233462395090 7, -0.15064811454225225476647737 01]
                  [  0.14935188545774774523352263,  -0.03560700314514977048729520 7])
intJdn5dn3uvrs =( [ 0.03122589745888049158791859 04,  0.03066254874047117069795458 81]
                  [ 0.03066254874047117069795458 81, 0.01203062469906479395309804256])
intJdn5dn4uvrs =(  [ -0.02027055070585714681345205 7, 0.04954876632664071347610338 42]
                   [ 0.04954876632664071347610338 42,  0.02203466262859569340030169 9])
intJdn5dn5uvrs =( [  1.31758678728731273266908314, 0.90287598944150236268304039 1]
                  [ 0.90287598944150236268304039 1,  1.49530824854081722232161367])
intJdn5dn6uvrs =( [ -0.78668680846441673459831513779,   0.27477449056073334063602205]
                  [   -0.73772550943926665936397795 4, -0.22202648081362135993240781 1])
intJdn5dn7uvrs =( [ 0.30653972944448350892451249 69, 0.27260597976622541223567498 7]
                  [  0.27260597976622541223567498 7, 0.11801080546717621414491773 4])
intJdn5dn8uvrs =(  [  -0.11236027725385717296152000 2, -0.10857831941052489303240522 64]
                   [ -0.10857831941052489303240522 64, -0.04240317317256157255425408 41])
intJdn5dn9uvrs =( [  -0.13495077149972707113986761 5, -0.11346506104520996771622436 19]
                  [ -0.11346506104520996771622436 19, -0.02836748587428850119955522 38])
intJdn5dn10uvrs =( [ 0.10667911259258557145532230 09,   0.00605933719247780881106809 5]
                   [  0.00605933719247780881106809 5, -0.13068498019167439266960372 94])
intJdn5dn11uvrs =(  [ -0.02697804586988087137951307 8, -0.19696639034109092890969408 4]
                    [ -0.19696639034109092890969408 4, -0.07889735824975260866827398 1])
intJdn5dn12uvrs =(  [  0.08921165458163810696876696, 0.42718888479907807017399238 6]
                    [ 0.42718888479907807017399238 6,  0.08921165458163810696876696])
intJdn5dn13uvrs =( [ 0.47555890783193200318874647 8, -0.53084335598705132620498792]
                   [ -0.53084335598705132620498792, -1.83117956250893266072281398])
intJdn5dn14uvrs =( [  -1.19557360195760315006531368, -0.89028963958964921999654621 7]
                   [ -0.89028963958964921999654621 7,  -0.14623767990238166834414548])
intJdn5dn15uvrs =(  [  0.44987769212666366162955235, 0.37936074707194774196023204 2]
                    [ 0.37936074707194774196023204 2, 0.08690454777248010390310669 2])
intJdn5dn16uvrs =(  [ -0.22254315718435603337366471 88,   0.09126861103192498840922 47]
                    [     0.09126861103192498840922 47, 0.53403155562725169509198688 7])
```

Table-2f

intJdnidnjuvrs, i=6,j=1 (1)16)

```
intJdn6dn1uvrs =(  [ 0.11725039453740174796746049 67,  0.15740792052350332961074531 8]
                   [ -0.14259207947649667038925468 2, -0.07002708952501684636496711 3])
intJdn6dn2uvrs =( [  -0.48406033635614064676180711 9, 0.38995920836752258992083291 49]
                  [ -0.32254079163247741007916708 51, 0.13782970183647076148485349 07])
intJdn6dn3uvrs =( [ -0.09563153300224068809754350 1,   -0.07011142235256930991307362]
                  [  -0.07011142235256930991307362, -0.02290194853292015105870834477])
intJdn6dn4uvrs =( [   0.00235304579485836526323508 3, -0.02396099145512853581863483 61]
                  [ -0.02396099145512853581863483 61, -0.00804949887236947900412300 02])
intJdn6dn5uvrs =(  [ -0.78668680846441673459831513779, -0.73772550943926665936397795 4]
                   [     0.27477449056073334063602205, -0.22202648081362135993240781 1])
intJdn6dn6uvrs =(  [ 1.33228970820397080499256458 9, 0.60206537351843515950311351 9]
                   [ 0.60206537351843515950311351 9, 1.14198144936988465286082004 9])
intJdn6dn7uvrs =(  [ -0.85797756874373307218110996 2,  -0.62901590438554706762765091 7]
                   [ -0.62901590438554706762765091 7, -0.25433776012165343954453545 34])
intJdn6dn8uvrs =( [  0.31571130247318704127722018 9, 0.24819901873655990881362319 74]
                  [ 0.24819901873655990881362319 74, 0.08373299434033970386656689 96])
intJdn6dn9uvrs =(  [   0.12380804271194872433082513 9,  -0.00793625216724699576515561 36]
                   [ -0.00793625216724699576515561 36, -0.11543642609252788831829720852])
intJdn6dn10uvrs =(  [ -0.04261918023706552353743972 94, 0.08321216960744057825494587 91]
                    [  0.08321216960744057825494587 91, 0.04988270099028496702346497 81])
intJdn6dn11uvrs =( [ 0.02752779109330720824872557 5, 0.08926809881605201657838772 6]
                   [ 0.08926809881605201657838772 6, 0.02752779109330720824872557 5])
intJdn6dn12uvrs =( [ -0.07889735824975260866827398 1, -0.19696639034109092890969408 4]
                   [ -0.19696639034109092890969408 4, -0.02697804586988087137951307 8])
```

intJdn6dn13uvrs =([0.10736670736867985341988703 9, 0.82742377912994363911301406 1]
[0.82742377912994363911301406 1, 0.52020218495669789167823211 9])
intJdn6dn14uvrs =([0.60105835750654359495658263, -0.51059946095920005197845965]
[-0.51059946095920005197845965, -1.50175278775672029105195536])
intJdn6dn15uvrs =([-0.26123164244466091271015883, 0.10610467066741683413328748 9]
[0.10610467066741683413328748 9, 0.44988102080951953135671271 8])
intJdn6dn16uvrs =([-0.02026092219188715390185249, -0.32732430826682450655130346]
[-0.32732430826682450655130346, -0.18952780581179438986486843])

Table-2g

intJdnidnjuvrs, i=7,j=1 (1)16)

---

intJdn7dn1uvrs =([-0.09314467940272043066056656 05, -0.10491119938818410070059750 75]
[-0.10491119938818410070059750 75, -0.09224518840851986203000470 4])
intJdn7dn2uvrs =([0.28346871318031592670232799, -0.45084785307694728127485979 8]
[0.26165214692305271872514020 15, -0.26464728995676401952354811])
intJdn7dn3uvrs =([-0.06683028025765907274929363 9, -0.14285359110808823461012582183]
[0.15714640889191176538987417817, 0.09583107259151565959073070 62])
intJdn7dn4uvrs =([0.00032737070603635956349312896 1, -0.02476171799987754376967145 69]
[-0.02476171799987754376967145 69, 0.00510040381792347092571950 6])
intJdn7dn5uvrs =([0.30653972944448350892451249 69, 0.27260597976622541223567498 7]
[0.27260597976622541223567498 7, 0.11801080546717621414491773 4])
intJdn7dn6uvrs =([-0.85797756874373307218110996 2, -0.62901590438554706762765091 69]
[-0.62901590438554706762765091 69, -0.25433776012165343954453545 34])
intJdn7dn7uvrs =([2.00108965805772110239126432 7, 0.27309567677916265610654186068]
[0.27309567677916265610654186068, 0.66650411320415920954619503 8])
intJdn7dn8uvrs =([-0.26463170281510494973966064 8, 0.39916403134969030939385162 2]
[-0.61333596865030969060614837761, -0.48812034084686916950769540 6])
intJdn7dn9uvrs =([0.04692540922720336695671244, -0.20462265621446298937714495 53]
[-0.20462265621446298937714495 53, 0.05302673520798462053775166 8])
intJdn7dn10uvrs =([-0.00545524177917866825797114 94, 0.08777306247789606020319170 01]
[0.08777306247789606020319170 01, -0.00545524177917866825797114 94])
intJdn7dn11uvrs =([0.04988270099028496702346497 81, 0.08321216960744057825494587 91]
[0.08321216960744057825494587 91, -0.04261918023706552353743972 94])
intJdn7dn12uvrs =([-0.13068498019167439266960372 94, 0.00605933719247780881106809 5]
[0.00605933719247780881106809 5, 0.10667911259258557145532230 09])
intJdn7dn13uvrs =([0.62179054482977490239946348, 0.03044832084463045004466 47]
[0.03044832084463045004466 47, -0.11315194309163614035589551 58])
intJdn7dn14uvrs =([-2.24559114498245358748525615, 0.03250271207921719343383613 7]
[0.03250271207921719343383613 7, 0.46364242643523354151733994 35])
intJdn7dn15uvrs =([0.55242173149670587806391031, 0.67638491447389996572089416 35]
[0.67638491447389996572089416 35, -0.29328323700329931136871321 1])
intJdn7dn16uvrs =([-0.19813025976000183828168734 1, -0.30423328239753321684461868 6]
[-0.30423328239753321684461868 6, 0.04506551212840784640782638 4])

Table-2h

intJdnidnjuvrs, i=8,j=1 (1)16)

---

intJdn8dn1uvrs =([0.03351827978973348544686950 5, 0.03928115348643125549691051 6]
[0.03928115348643125549691051 6, 0.03187907114531758725070088 95])
intJdn8dn2uvrs =([-0.11413796226676360674772048, 0.17064551262371894348372706704]
[-0.12935448737628105651627293 3, 0.08032716613183666521258486 71])
intJdn8dn3uvrs =([0.21711976690295516640568278 5, 0.34817438052817404844048581161]
[-0.36432561947182595155951418839, -0.35254272491668073283298402])
intJdn8dn4uvrs =([-0.01076563963110726691819421 09, 0.05750100224070899432117091 37]
[0.05750100224070899432117091 37, -0.03532803347958372401958165 5])
intJdn8dn5uvrs =([-0.11236027725385717296152000 2, -0.10857831941052489303240522 64]
[-0.10857831941052489303240522 64, -0.04240317317256157255425408 41])
intJdn8dn6uvrs =([0.31571130247318704127722018 9, 0.24819901873655990881362319 74]
[0.24819901873655990881362319 74, 0.08373299434033970386656689 96])
intJdn8dn7uvrs =([-0.26463170281510494973966064 8, -0.61333596865030969060614837761]
[0.39916403134969030939385162 2, -0.48812034084686916950769540 6])
intJdn8dn8uvrs =([1.65315162092583559267291948 5, 0.15317854106260736184624261268]

[0.153178541062607361846242612 68, 0.767115078598661079333576 13])
intJdn8dn9uvrs =([-0.22236672155468883029403792, 0.484990794065840660904685270 82]
[0.484990794065840660904685270 82, -0.22236672155468883029403792])
intJdn8dn10uvrs =([0.05302673520798462053775166 8, -0.204622656214462989377144955 3]
[-0.204622656214462989377144955 3, 0.046925409227203366956712 44])
intJdn8dn11uvrs =([-0.115436426092527888318297208 52, -0.007936252167246995765155613 6]
[-0.007936252167246995765155613 6, 0.123808042711948724330825139])
intJdn8dn12uvrs =([-0.028367485874288501199555223 8, -0.113465061045209967716224361 9]
[-0.113465061045209967716224361 9, -0.134950771499727071139867615])
intJdn8dn13uvrs =([0.060329059165411543334696078, 0.331278731126193705006371948]
[0.331278731126193705006371948, 0.178851051525261210302257368])
intJdn8dn14uvrs =([-0.051995759267777985302627 03, -0.795506545993563782159836415 2]
[-0.795506545993563782159836415 2, -0.433262214197932020360238335])
intJdn8dn15uvrs =([-1.944186194720450713243523 88, -0.026649721111754807309061686 8]
[-0.026649721111754807309061686 8, 0.550843979761993341763278679])
intJdn8dn16uvrs =([0.531391405011459465049996936, 0.036845390722838247652759296]
[0.036845390722838247652759296, -0.154508813774518558307843375])

Table-2i

intJdnidnjuvrs, i=9,j=1 (1)16)

---

intJdn9dn1uvrs =([0.031879071145317587250700889 5, 0.039281153486431255496910516]
[0.039281153486431255496910516, 0.033518279789733485446869504 5])
intJdn9dn2uvrs =([-0.035328033479583724019581655, 0.057501002240708994321170913 7]
[0.057501002240708994321170913 7, -0.010765639631107266918194210 9])
intJdn9dn3uvrs =([-0.352542724916680732832984 02, -0.364325619471825951559514188 4]
[0.348174380528174048440485811 61, 0.217119766902955166405682785])
intJdn9dn4uvrs =([0.080327166131836665212584867 1, -0.129354487376281056516272932 96]
[0.170645512623718943483727067 04, -0.114137962266763606747720 48])
intJdn9dn5uvrs =([-0.134950771499727071139867615, -0.113465061045209967716224362]
[-0.113465061045209967716224362, -0.028367485874288501199555223 8])
intJdn9dn6uvrs =([0.123808042711948724330825139, -0.007936252167246995765155613 6]
[-0.007936252167246995765155613 6, -0.115436426092527888318297208 52])
intJdn9dn7uvrs =([0.046925409227203366956712 44, -0.204622656214462989377144955 3]
[-0.204622656214462989377144955 3, 0.053026735207984620537751668])
intJdn9dn8uvrs =([-0.222366721554688830294037 92, 0.484990794065840660904685270 82]
[0.484990794065840660904685270 82, -0.222366721554688830294037 92])
intJdn9dn9uvrs =([0.767115078598661079333576 13, 0.153178541062607361846242612 68]
[0.153178541062607361846242612 68, 1.653151620925835592672919485])
intJdn9dn10uvrs =([-0.488120340846869169507695406, 0.399164031349690309393851622 39]
[-0.613335968650309690606148377 61, -0.264631702815104949739660648])
intJdn9dn11uvrs =([0.083732994340339703866566899 6, 0.248199018736559908813623197 4]
[0.248199018736559908813623197 4, 0.315711302473187041277220189])
intJdn9dn12uvrs =([-0.042403173172561572554254084 1, -0.108578319410524893032405226 4]
[-0.108578319410524893032405226 4, -0.112360277253857172961520002])
intJdn9dn13uvrs =([0.178851051525261210302257368, 0.331278731126193705006371948]
[0.331278731126193705006371948, 0.060329059165411543334696078])
intJdn9dn14uvrs =([-0.154508813774518558307843375, 0.036845390722838247652759296]
[0.036845390722838247652759296, 0.531391405011459465049996936])
intJdn9dn15uvrs =([0.550843979761993341763278679, -0.026649721111754807309061686 8]
[-0.026649721111754807309061686 8, -1.944186194720450713243523 88])
intJdn9dn16uvrs =([-0.433262214197932020360238335, -0.795506545993563782159836415 2]
[-0.795506545993563782159836415 2, -0.051995759267777985302627 03])

Table-2j

intJdnidnjuvrs, i=10,j=1 (1)16)

---

intJdn10dn1uvrs =([-0.092245188408519862030004704, -0.104911199388184100700597508]
[-0.104911199388184100700597508, -0.093144679402720430660566560 5])
intJdn10dn2uvrs =([0.005100403817923470925719506, -0.024761717999877543769671456 9]
[-0.024761717999877543769671456 9, 0.000327370706036359563493128961])
intJdn10dn3uvrs =([0.095831072591515659590730706 2, 0.157146408891911765389874178 17]
[-0.142853591108088234610125821 83, -0.066830280257659072749293639])
intJdn10dn4uvrs =([-0.264647289956764019523548 11, 0.261652146923052718725140201 5]
[-0.450847853076947281274859798 5, 0.283468713180315926702327 99])

intJdn10dn5uvrs =([0.10667911259258557145532301, 0.00605933719247780881106809 5]
[0.00605933719247780881106809 5, -0.13068498019167439266960372 94])
intJdn10dn6uvrs =([-0.04261918023706552353743972 94, 0.08321216960744057825494587 91]
[0.08321216960744057825494587 91, 0.04988270099028496702346497 81])
intJdn10dn7uvrs =([-0.00545524177917866825797114 94, 0.08777306247789606020319170 01]
[0.08777306247789606020319170 01, -0.00545524177917866825797114 94])
intJdn10dn8uvrs =([0.05302673520798462053775166 8, -0.20462265621446298937714495 53]
[-0.20462265621446298937714495 53, 0.04692540922720336695671244])
intJdn10dn9uvrs =([-0.48812034084686916950769540 6, -0.61333596865030969060614837761]
[0.39916403134969030939385162239, -0.26463170281510494973966064 8])
intJdn10dn10uvrs =([0.66650411320415920954619503 76, 0.27309567677916265610654186068]
[0.27309567677916265610654186068, 2.00108965805772110239126432 7])
intJdn10dn11uvrs =([-0.25433776012165343954453545 34, -0.62901590438554706762765091 69]
[-0.62901590438554706762765091 69, -0.85797756874373307218110996 2])
intJdn10dn12uvrs =([0.11801080546717621414491773 4, 0.27260597976622541223567498 7]
[0.27260597976622541223567498 7, 0.30653972944448350892451249 69])
intJdn10dn13uvrs =([-0.11315194309163614035589551 58, 0.03044832084463045004466 47]
[0.03044832084463045004466 47, 0.62179054482977490239946348])
intJdn10dn14uvrs =([0.04506551212840784640782638 4, -0.30423328239753321684461868 6]
[-0.30423328239753321684461868 6, -0.19813025976000183828168734 1])
intJdn10dn15uvrs =([-0.29328323700329931136871321 1, 0.67638491447389969657208941635]
[0.67638491447389969657208941635, 0.55242173149670587806391031])
intJdn10dn16uvrs =([0.46364242643523354151733994 35, 0.03250271207921719343383613 7]
[0.03250271207921719343383613 7, -2.24559114498245358748525615])

Table-2k
intJdnidnjuvrs, i=11,j=1 (1)16)

---

intJdn11dn1uvrs =([-0.07002708952501684636496711 3, -0.14259207947649667038925468 2]
[0.15740792052350332961074531 8, 0.11725039453740174797460496 7])
intJdn11dn2uvrs =([-0.00804949887236947900412300 02, -0.02396099145512853581863483 61]
[-0.02396099145512853581863483 61, 0.00235304579485836526323508 27])
intJdn11dn3uvrs =([-0.02290194853292015105870834477, -0.07011142235256930991307362]
[-0.07011142235256930991307362, -0.09563153300224068809754350 1])
intJdn11dn4uvrs =([0.13782970183647076148485349 07, -0.32254079163247741007916708 51]
[0.38995920836752258992083291 49, -0.48406033635614064676180711 9])
intJdn11dn5uvrs =([-0.02697804586988087137951307 8, -0.19696639034109092890969408 4]
[-0.19696639034109092890969408 4, -0.07889735824975260866827398 1])
intJdn11dn6uvrs =([0.02752779109330720824872557 5, 0.08926809881605201657838772 6]
[0.08926809881605201657838772 6, 0.02752779109330720824872557 5])
intJdn11dn7uvrs =([0.04988270099028496702346497 81, 0.08321216960744057825494587 91]
[0.08321216960744057825494587 91, -0.04261918023706552353743972 94])
intJdn11dn8uvrs =([-0.11543642609252788831829720 852, -0.00793625216724699576515561 36]
[-0.00793625216724699576515561 36, 0.12380804271194872433082513 9])
intJdn11dn9uvrs =([0.08373299434033970386656689 96, 0.24819901873655990881362319 74]
[0.24819901873655990881362319 74, 0.31571130247318704127722018 9])
intJdn11dn10uvrs =([-0.25433776012165343954453545 34, -0.62901590438554706762765091 69]
[-0.62901590438554706762765091 69, -0.85797756874373307218110996 2])

intJdn11dn11uvrs =([1.14198144936988465286082004 9, 0.60206537351843515950311351 9]
[0.60206537351843515950311351 9, 1.33228970820397080499256458 9])
intJdn11dn12uvrs =([-0.22202648081362135993240781 1, 0.27477449056073334063602204 6]
[-0.73772550943926665936397795 4, -0.78668680846441673459831513779])
intJdn11dn13uvrs =([0.52020218495669789167823211 9, 0.82742377912994363911301406 1]
[0.82742377912994363911301406 1, 0.10736670736867985341988703 9])
intJdn11dn14uvrs =([-0.18952780581179438986486843, -0.32732430826682450655130346]
[-0.32732430826682450655130346, -0.02026092219188715390185249])
intJdn11dn15uvrs =([0.44988102080951953135671271 8, 0.10610467066741683413328748 9]
[0.10610467066741683413328748 9, -0.26123164244466091271015883])
intJdn11dn16uvrs =([-1.50175278775672029105195536, -0.51059946095920005197845965]
[-0.51059946095920005197845965, 0.60105835750654359495658263])

Table-2l
intJdnidnjuvrs, i=12,j=1 (1)16)

---

intJdn12dn1uvrs =([0.157871624541338704794557829, 0.26894552598477698154302253 3]
[-0.44355447401522301845697746 7, -0.41263305411760023071620689 8])
intJdn12dn2uvrs =([0.022034662628595693400301698 6, 0.04954876632664071347610338 42]
[0.04954876632664071347610338 42, -0.02027055070585714681345205 7])
intJdn12dn3uvrs =([0.012030624699064793953098043, 0.03066254874047117069795458 81]
[0.03066254874047117069795458 81, 0.03122589745888049158791859 04])
intJdn12dn4uvrs =([-0.035607003145149770487295207, 0.14935188545774774523352262 99]
[-0.15064811454225225476647737, 0.13531648572980233462395090 7])
intJdn12dn5uvrs =([0.089211654581638106968766 96, 0.42718888479907807017399238 6]
[0.42718888479907807017399238 6, 0.08921165458163810696876696])
intJdn12dn6uvrs =([-0.078897358249752608668273981, -0.19696639034109092890969408 4]
[-0.19696639034109092890969408 4, -0.02697804586988087137951307 8])
intJdn12dn7uvrs =([-0.130684980191674392669603729 4, 0.00605933719247780881106809 5]
[0.00605933719247780881106809 5, 0.10667911259258557145532230 09])
intJdn12dn8uvrs =([-0.028367485874288501199552238, -0.11346506104520996771622436 19]
[-0.11346506104520996771622436 19, -0.13495077149972707113986761 5])
intJdn12dn9uvrs =([-0.042403173172561572554254084 1, -0.10857831941052489303240522 64]
[-0.10857831941052489303240522 64, -0.11236027725385717296152000 2])
intJdn12dn10uvrs =([0.11801080546717621414491773 4, 0.27260597976622541223567498 7]
[0.27260597976622541223567498 7, 0.30653972944448350892451249 69])
intJdn12dn11uvrs =([-0.22202648081362135993240781 1, -0.73772550943926665936397795 4]
[0.27477449056073334063602204 6, -0.78668680846441673459831513 779])
intJdn12dn12uvrs =([1.49530824854081722232161367 5, 0.90287598944150236268304039 1]
[0.90287598944150236268304039 1, 1.31758678728731273266908313 7])
intJdn12dn13uvrs =([-1.83117956250893266072281398 3, -0.53084335598705132620498791 7]
[-0.53084335598705132620498791 7, 0.47555890783193200318874648])
intJdn12dn14uvrs =([0.53403155562725169509198688 7, 0.09126861103192498840922 47]
[0.09126861103192498840922 47, -0.22254315718435603337366471 88])
intJdn12dn15uvrs =([0.08690454777248010390310669 2, 0.37936074707194774196023204 2]
[0.37936074707194774196023204 2, 0.44987769212666366162955235])
intJdn12dn16uvrs =([-0.14623767990238166834414548, -0.89028963958964921999654621 7]
[-0.89028963958964921999654621 7, -1.19557360195760315006531367 88])

Table-2m
intJdnidnjuvrs, i=13,j=1 (1)16)

---

intJdn13dn1uvrs =([-0.92367783910183130555067 71, -1.00703245489050046307564 23])
[-1.00703245489050046307564 23, -0.92367783910183130555067 71])
intJdn13dn2uvrs =([-0.06246030481817935466793166, -0.19613023897523854894811148 8]
[-0.19613023897523854894811148 8, 0.01806644776516889645632182 4])
intJdn13dn3uvrs =([-0.04401071284291265210757706 44, -0.09641597529566077118991423 76]
[-0.09641597529566077118991423 76, -0.04401071284291265210757706 44])
intJdn13dn4uvrs =([0.01806644776516889645632182 4, -0.19613023897523854894811148 8]
[-0.19613023897523854894811148 8, -0.06246030481817935466793165 7])
intJdn13dn5uvrs =([0.47555890783193200318874648, -0.53084335598705132620498792]
[-0.53084335598705132620498792, -1.83117956250893266072281398])
intJdn13dn6uvrs =([0.10736670736867985341988703 9, 0.82742377912994363911301406 1]
[0.82742377912994363911301406 1, 0.52020218495669789167823211 9])
intJdn13dn7uvrs =([0.62179054482977490239946348, 0.03044832084463045004466 47]
[0.03044832084463045004466 47, -0.11315194309163614035589551 58])
intJdn13dn8uvrs =([0.06032905916541154333469607 8, 0.33127873112619370500637194 8]
[0.33127873112619370500637194 8, 0.17885105152526121030225736 8])
intJdn13dn9uvrs =([0.17885105152526121030225736 8, 0.33127873112619370500637194 8]
[0.33127873112619370500637194 8, 0.06032905916541154333469607 8])

intJdn13dn10uvrs =([-0.11315194309163614035589551 58, 0.03044832084463045004466 47]
[0.03044832084463045004466 47, 0.62179054482977490239946348])
intJdn13dn11uvrs =([0.52020218495669789167823211 9, 0.82742377912994363911301406 1]
[0.82742377912994363911301406 1, 0.10736670736867985341988703 9])
intJdn13dn12uvrs =([-1.83117956250893266072281398 3, -0.53084335598705132620498791 7]
[-0.53084335598705132620498791 7, 0.47555890783193200318874648])
intJdn13dn13uvrs =([5.22041258699831676842108804 5, 2.88129572754069028280297951]
[2.88129572754069028280297951, 5.22041258699831676842108804 5])
intJdn13dn14uvrs =([-3.40047247984420012666520625 9, -0.87612662426029179489909569 7]
[-0.87612662426029179489909569 7, -0.80962223894644387409993966 03])

intJdn13dn15uvrs =([-0.018002409287106955030651138, -0.949948521110901296761134222]
[-0.949948521110901296761134222, -0.018002409287106955030651138])
intJdn13dn16uvrs =([-0.809622238946443874099396603, -0.876126624260291794899095697]
[-0.876126624260291794899095697, -3.400472479844200126665206259])

Table-2n

intJdnidnjuvrs, i=14,j=1 (1)16)

---

intJdn14dn1uvrs =([0.33685199022272286409417 2519, 0.381671404352391717716050003]
[0.381671404352391717716050003, 0.33845901154347235890690 86])
intJdn14dn2uvrs =([-0.005654544717914146990072595, 0.427622370384584419718474 4941]
[0.427622370384584419718474 4941, -0.153931015713211339133137 8486])
intJdn14dn3uvrs =([0.136999462878579334021389201, 0.227213413649009719816657 2636]
[0.227213413649009719816657 2636, 0.094118555809911844171302174])
intJdn14dn4uvrs =([-0.000984668166045009257980 4616, 0.086300080830678135931299 1519]
[0.086300080830678135931299 1519, 0.020826689444389221529093 2469])
intJdn14dn5uvrs =([-1.195573601957603150065313 6788, -0.890289639589649219996546217]
[-0.890289639589649219996546217, -0.146237679902381668344145 48])
intJdn14dn6uvrs =([0.601058357506543594956582 63, -0.510599460959200051978459 65]
[-0.510599460959200051978459 65, -1.501752787756720291051955 36])
intJdn14dn7uvrs =([-2.245591144982453587485256 15, 0.032502712079217193433836137]
[0.032502712079217193433836137, 0.463642426435233541517339 9435])
intJdn14dn8uvrs =([-0.051995759267777985302627 03, -0.795506545993563782159836 4152]
[-0.795506545993563782159836 4152, -0.433262214197932020360238335])
intJdn14dn9uvrs =([-0.154508813774518558307843375, 0.036845390722838247652759296]
[0.036845390722838247652759296, 0.531391405011459465049996936])
intJdn14dn10uvrs =([0.045065512128407846407826384, -0.304233282397533216844618686]
[-0.304233282397533216844618686, -0.198130259760001838281687341])
intJdn14dn11uvrs =([-0.189527805811794389864868 43, -0.327324308266824506551303 46]
[-0.327324308266824506551303 46, -0.020260922191887153901852 49])
intJdn14dn12uvrs =([0.534031555627251695091986887, 0.091268611031924988409224 7]
[0.091268611031924988409224 7, -0.222543157184356033373664 7188])
intJdn14dn13uvrs =([-3.400472479844200126665206259, -0.876126624260291794899095697]
[-0.876126624260291794899095697, -0.809622238946443874099939 6603])
intJdn14dn14uvrs =([5.518761290142117566125380 31, 1.770759558087978375081430869]
[1.770759558087978375081430869, 3.929578185962763299703317274])
intJdn14dn15uvrs =([-0.805977919806664929391926 72, -0.637841602652030943257059331]
[-0.637841602652030943257059331, -2.769794568377644494965093629])
intJdn14dn16uvrs =([0.877518569823348982633756744, 1.287737922980470717927187534]
[1.287737922980470717927187534, 0.877518569823348982633756744])

Table-2o

intJdnidnjuvrs, i=15,j=1 (1)16)

---

intJdn15dn1uvrs =([-0.122483212211684061698359345, -0.145263095437075394958553495])
[-0.145263095437075394958553495, -0.122483212211684061698359345])
intJdn15dn2uvrs =([0.008033238055969811770259 3, -0.194216973763714354534537 5175]
[-0.194216973763714354534537 5175, 0.041681314367425998111696912])
intJdn15dn3uvrs =([-0.319448664201393584204597479, -0.534960505588025669879559 80517]
[-0.534960505588025669879559 80517, -0.319448664201393584204597479])
intJdn15dn4uvrs =([0.041681314367425998111696919, -0.194216973763714354534537 5175]
[-0.194216973763714354534537 5175, 0.008033238055969811770258 93])
intJdn15dn5uvrs =([0.449877692126663661629552 35, 0.379360747071947741960232042]
[0.379360747071947741960232042, 0.086904547772480103903106692])
intJdn15dn6uvrs =([-0.261231642444660912710158 83, 0.106104670667416834133287489]
[0.106104670667416834133287489, 0.449881020809519531356712 72])
intJdn15dn7uvrs =([0.552421731496705878063910 31, 0.676384914473899965720894163 5]
[0.676384914473899965720894163 5, -0.293283237003299311368713211])
intJdn15dn8uvrs =([-1.944186194720450713243523 88, -0.026649721111754807309061686 8]
[-0.026649721111754807309061686 8, 0.550843979761993341763278679])

intJdn15dn9uvrs =([0.550843979761993341763278679, -0.026649721111754807309061686 8]
[-0.026649721111754807309061686 8, -1.944186194720450713243523 88])
intJdn15dn10uvrs =([-0.293283237003299311368713211, 0.676384914473899965720894163 5]
[0.676384914473899965720894163 5, 0.552421731496705878063910 31])

intJdn15dn11uvrs =([0.44988102080951953135671271 8, 0.10610467066741683413328748 9]
[0.10610467066741683413328748 9, -0.26123164244466091271015883])
intJdn15dn12uvrs =([0.08690454777248010390310669 2, 0.37936074707194774196023204 2]
[0.37936074707194774196023204 2, 0.44987769212666366162955235])
intJdn15dn13uvrs =([-0.01800240928710695503065113 8, -0.94994852111090129676113422 2]
[-0.94994852111090129676113422 2, -0.01800240928710695503065113 8])
intJdn15dn14uvrs =([-0.80597791980666492939192672, -0.63784160265203094325705933 1]
[-0.63784160265203094325705933 1, -2.76979456837764449496509362 9])
intJdn15dn15uvrs =([4.39476432366214663601450761, 1.02388805276447348817173719 5]
[1.02388805276447348817173719 5, 4.39476432366214663601450761])
intJdn15dn16uvrs =([-2.76979456837764449496509362 9, -0.63784160265203094325705933 1]
[-0.63784160265203094325705933 1, -0.80597791980666492939192672])

Table-2p
intJdnidnjuvrs, i=16,j=1 (1)16)

---

intJdn16dn1uvrs =([0.33845901154347235890690 86, 0.38167140435239171771605000 3]
[0.38167140435239171771605000 3, 0.33685199022272286409417252])
intJdn16dn2uvrs =([0.02082668944438922152909324 69, 0.08630008083067813593129915 19]
[0.08630008083067813593129915 19, -0.00098466816604500925798046 16])
intJdn16dn3uvrs =([0.09411855580991184417130217 44, 0.22721341364900971981665726 36]
[0.22721341364900971981665726 36, 0.13699946287857933402138920 1])
intJdn16dn4uvrs =([-0.15393101571321133913313784 86, 0.42762237038458441971847449 41]
[0.42762237038458441971847449 41, -0.00565454471791414699007259 5])
intJdn16dn5uvrs =([-0.22254315718435603337366471 88, 0.09126861103192498840922 47]
[0.09126861103192498840922 47, 0.53403155562725169509198688 7])
intJdn16dn6uvrs =([-0.02026092219188715390185249, -0.32732430826682450655130346]
[-0.32732430826682450655130346, -0.18952780581179438986486843])
intJdn16dn7uvrs =([-0.19813025976000183828168734 1, -0.30423328239753321684461868 6]
[-0.30423328239753321684461868 6, 0.04506551212840784640782638 4])
intJdn16dn8uvrs =([0.53139140501145946504999693 6, 0.03684539072283824765275929 6]
[0.03684539072283824765275929 6, -0.15450881377451855830784337 5])
intJdn16dn9uvrs =([-0.43326221419793202036023833 5, -0.79550654599356378215983641 52]
[-0.79550654599356378215983641 52, -0.05199575926777798530262703])
intJdn16dn10uvrs =([0.46364242643523354154173399 435, 0.03250271207921719343383613 7]
[0.03250271207921719343383613 7, -2.24559114498245358748525615])
intJdn16dn11uvrs =([-1.50175278775672029105195536, -0.51059946095920005197845965]
[-0.51059946095920005197845965, 0.60105835750654359495658263])
intJdn16dn12uvrs =([-0.14623767990238166834414548, -0.89028963958964921999654621 7]
[-0.89028963958964921999654621 7, -1.19557360195760315006531367 88])
intJdn16dn13uvrs =([-0.80962223894644387409993966 03, -0.87612662426029179489909569 7]
[-0.87612662426029179489909569 7, -3.40047247984420012666520625 9])
intJdn16dn14uvrs =([0.87751856982334898263375674 4, 1.28773792298047071792718753 4]
[1.28773792298047071792718753 4, 0.87751856982334898263375674 4])
intJdn16dn15uvrs =([-2.76979456837764449496509362 9, -0.63784160265203094325705933 1]
[-0.63784160265203094325705933 1, -0.80597791980666492939192672])
intJdn16dn16uvrs =([3.92957818596276329970331727 4, 1.77075955808797837508143086 9]
[1.77075955808797837508143086 9, 5.51876129014211756612538031])

SOME SAMPLE RESULTS TABLES& FIGURES)

Table 2a

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

PREVIOUS PAPER: FEM SOLUTION FOR 9 node quadratic Lagrange element)

Torsion of a square cross section modeled as a right isoscles triangle:R

,where R={ x,y)|0≤ $x, y \leq \frac{1}{2}$,0 ≤ $x - y) \leq 1/2\}$

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

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

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

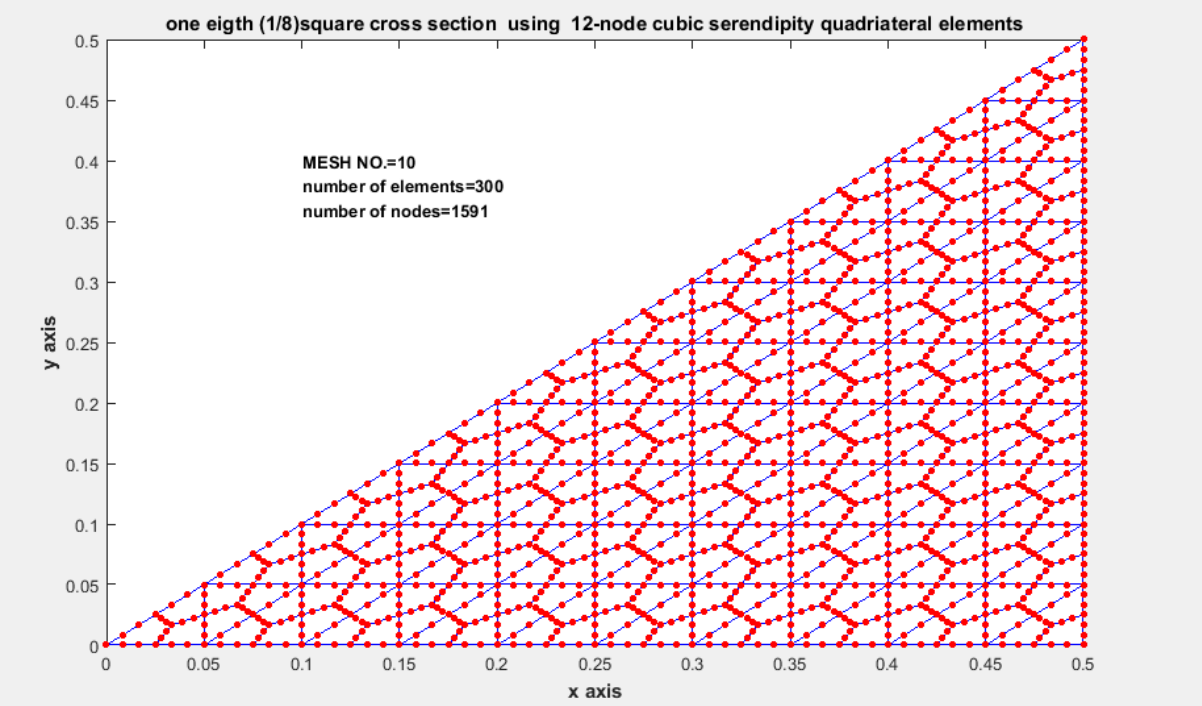
exact solution of torisonal constant= 0.140577014955156)

nnode=number of nodes in the triangular region R

nel=number of elements in the region R

FEM solution for Prandtl Stress function

Mesh No	nnode	nel	nnel	Torisonal constant	maximum absolute error
1)	19	3	9	0.140226269123952	0.0011496140422405
	25	3	12	0.140100662876437	0.001814895145593
	37	3	16	0.140564616238274	0.000361119966844
2)	61	12	9	0.140549995991093	0.000283838209277359
	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	0.000040770762286
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	2.98546615531262e-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	1.8370913205128e-005
	1033	192	12	0.140576230947825	0.000037905251021
	1801	192	16	0.140577011993390	0.000008119911696
9)	1027	243	9	0 .140576925494304	1.58338405955379e-005
	1297	243	12	0.140576431755842	0.000030811538513
	2269	243	16	0.140577013106209	0.000006337564194
10)	1261	300	9	0.140576955193951	1.38273942157716e-005
	1591	300	12	0.140576564296181	0.000024665504597
	2791	300	16	0.140577013742108	0.000004703818292



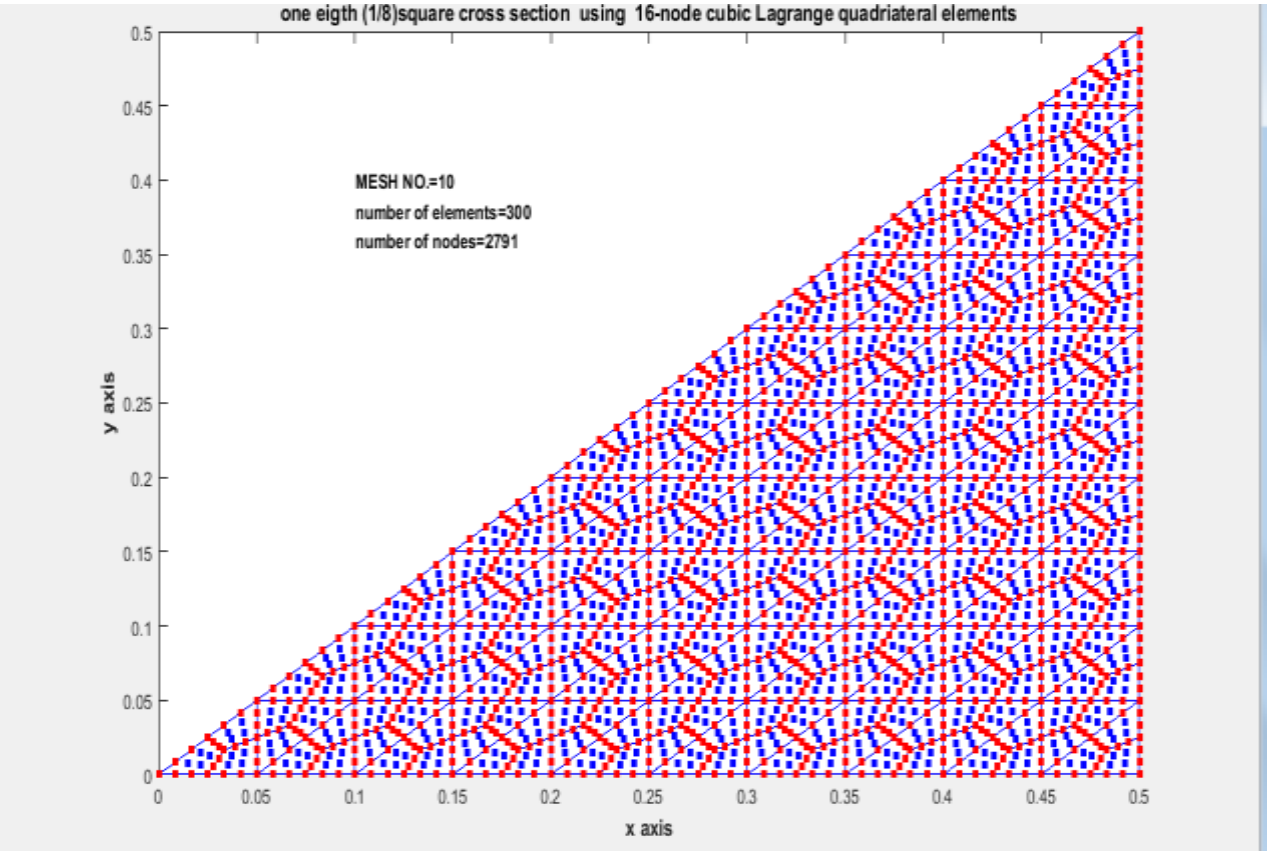


Table-3a

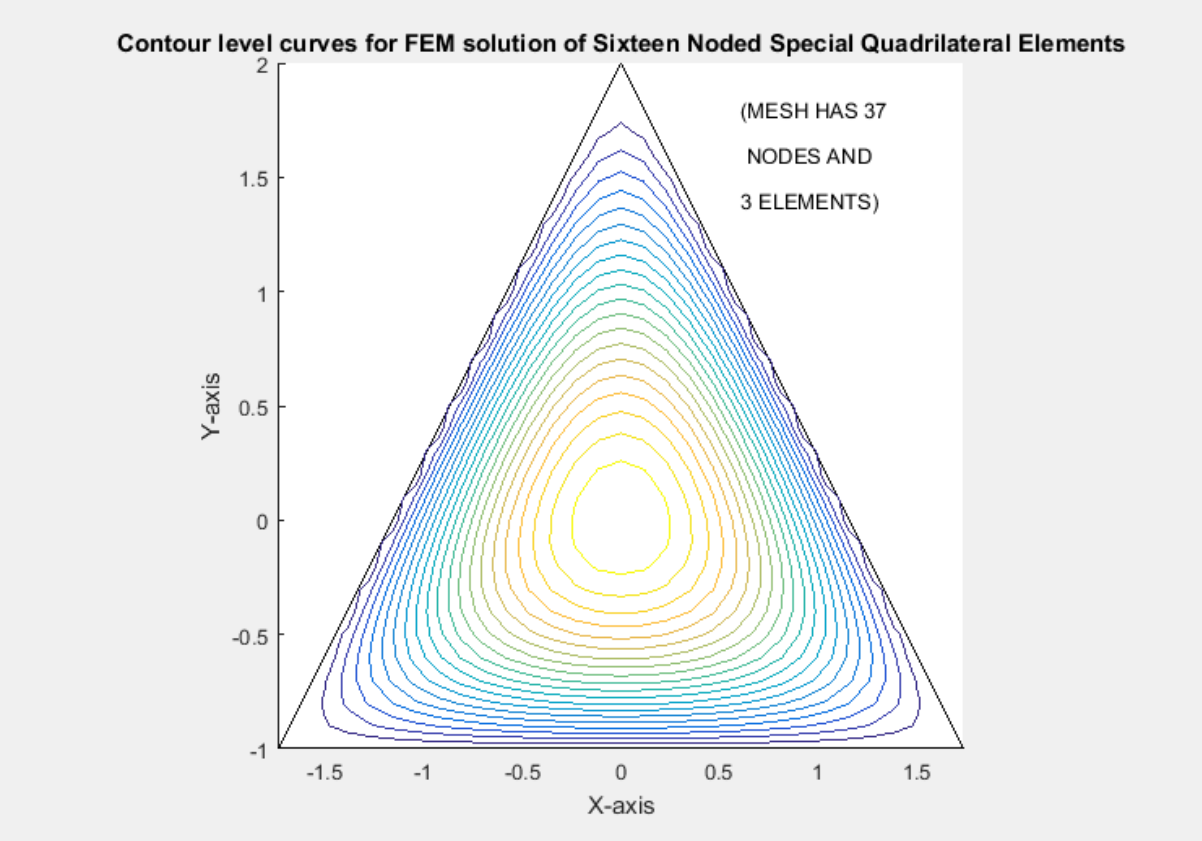
TORSION OF AN EQUILATERAL TRIANGULAR CROSS SECTION

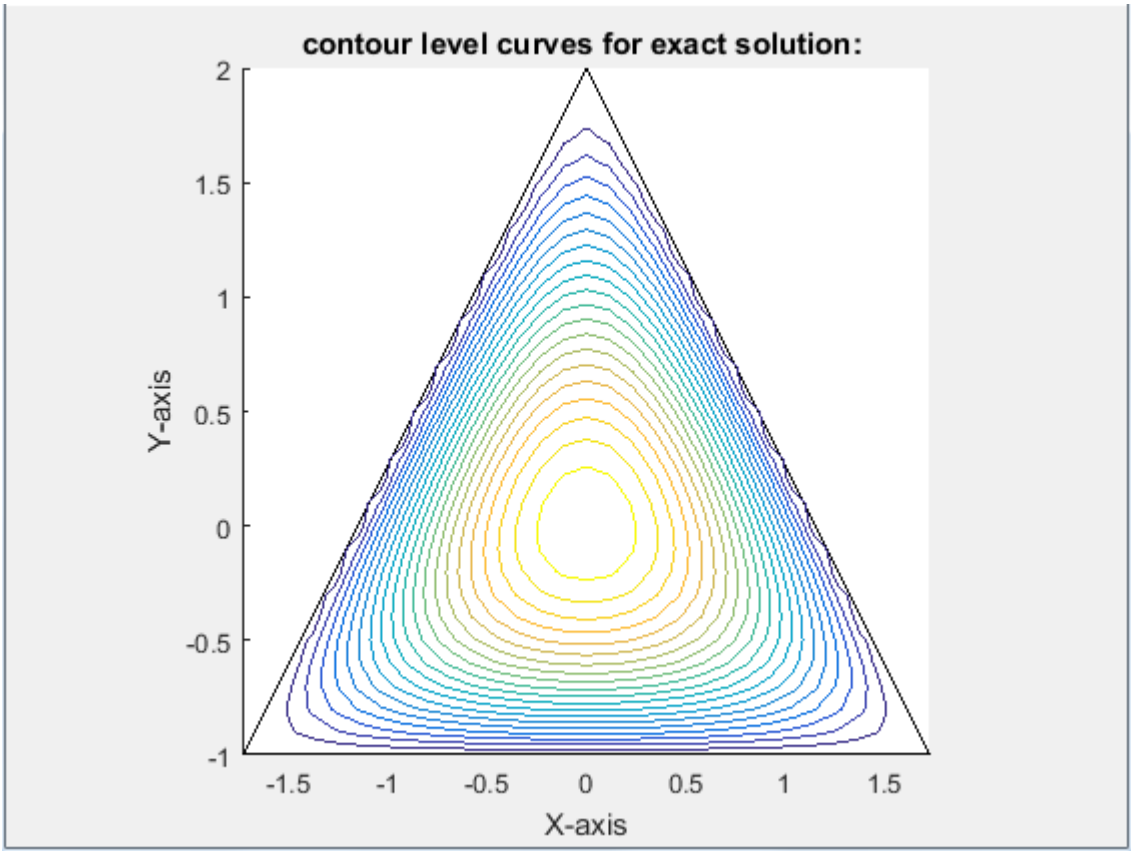
exact solution of torisonal constant= 3.11769145362398)
PRESENT PAPER: FEM SOLUTION FOR 12 node and 16 node cubic elements)
PREVIOUS PAPER: FEM SOLUTION FOR 9 node quadratic Lagrange element)
Linear elastic torsion of an equilateral triangle T which is inscribed in a circle of unit radius.
Equilateral 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

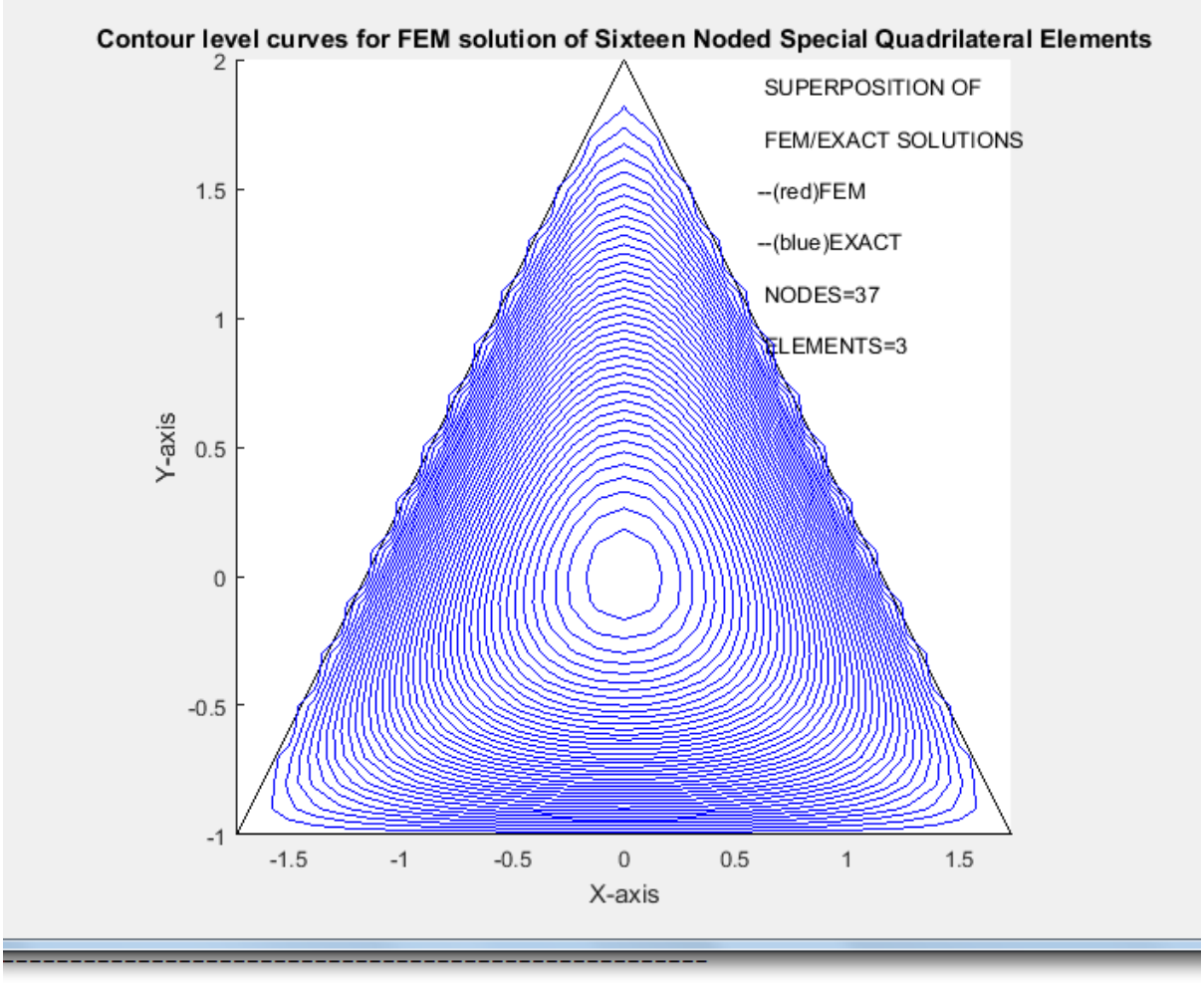
FEM solution for of Prandtl Stress function

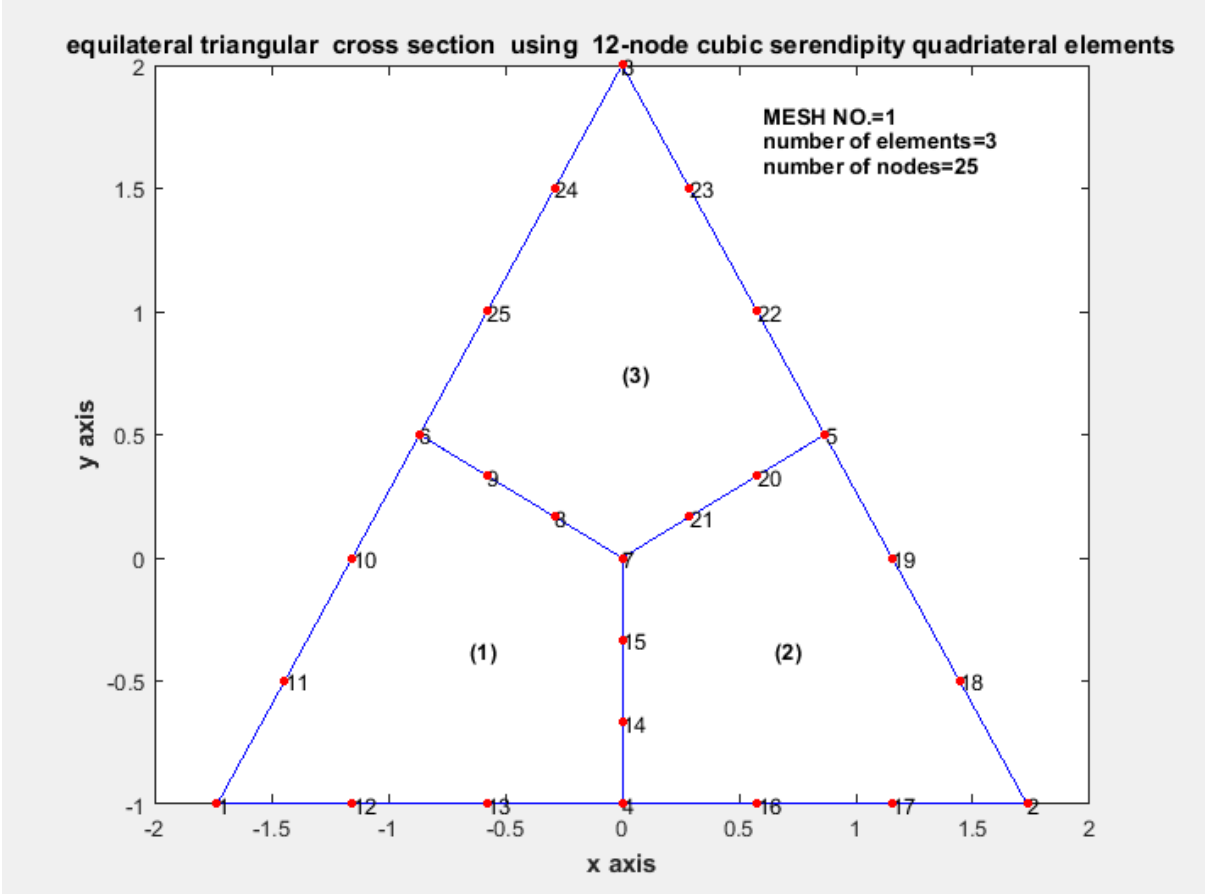
Mesh No	nnode	nel	nnel	Torisonal constant	maximum absolute error
1	1	9	3	9	3.10739407578924	0.00317246382279579
	25	3	12	3.032110091189756	0.02914323981426
	37	3	16	3.117691453623980	0.000000000000000
2	61	12	9	3.11704786750931	0.000396557977849543
	79	12	12	3.110644194853498	0.006021532419355
3	127	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

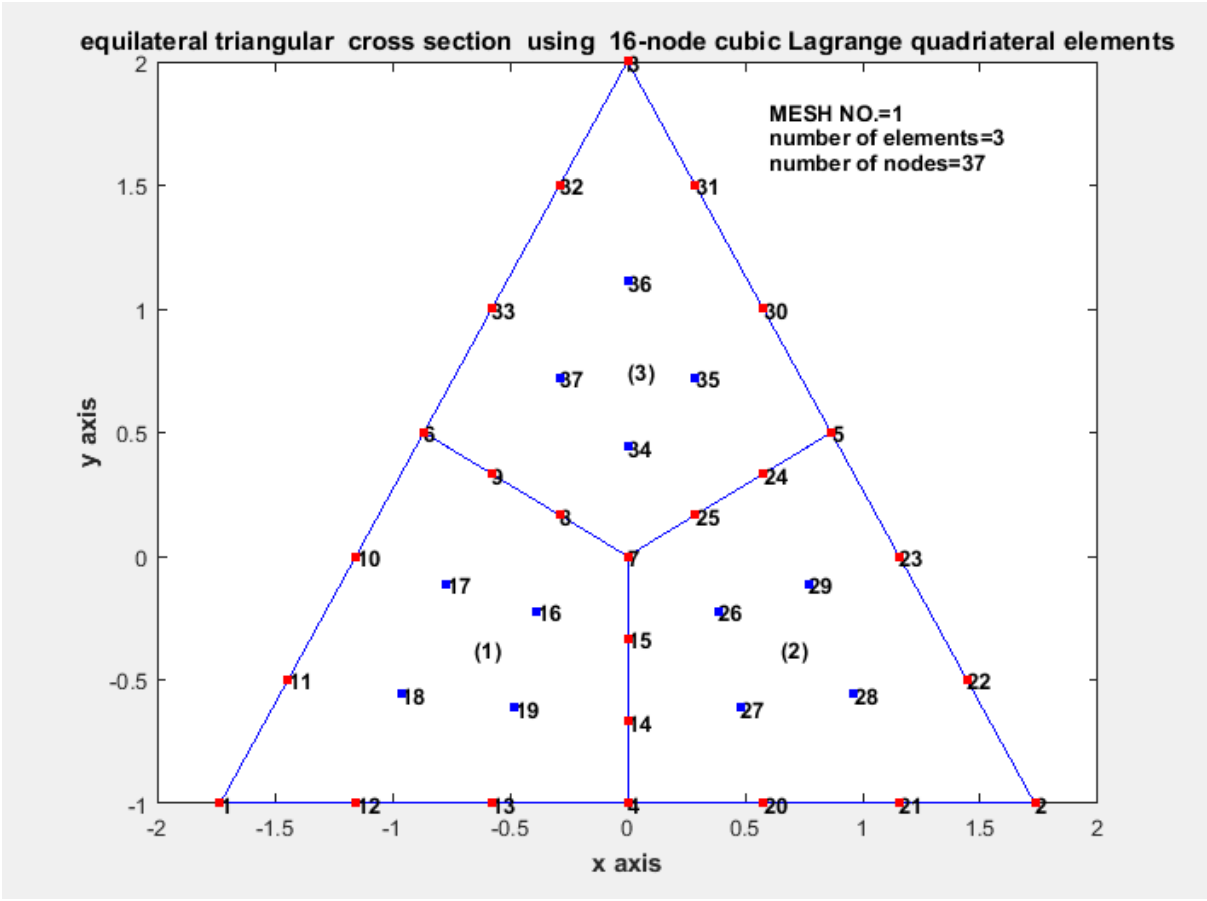
Example:TORSION OF AN EQUILATERAL TRIANGULAR CROSS SECTION

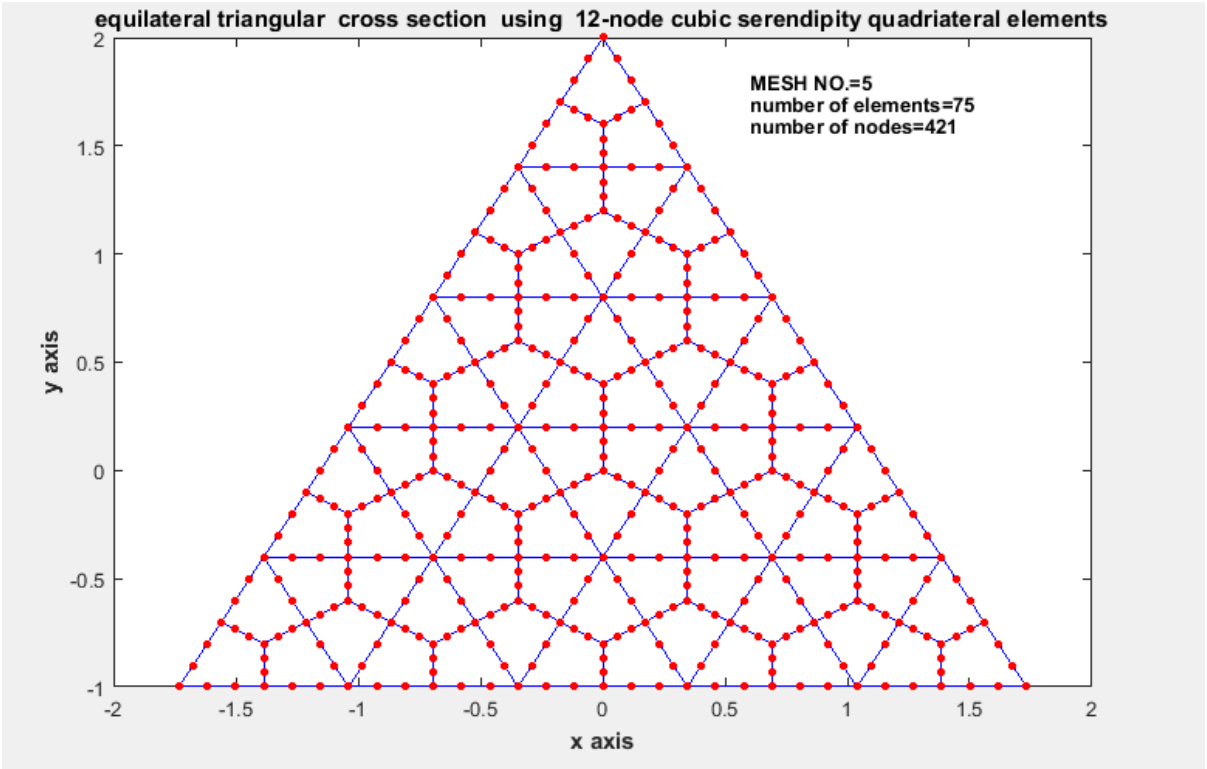

Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements
(MESH HAS 37
NODES AND
3 ELEMENTS)
Y-axis
X-axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

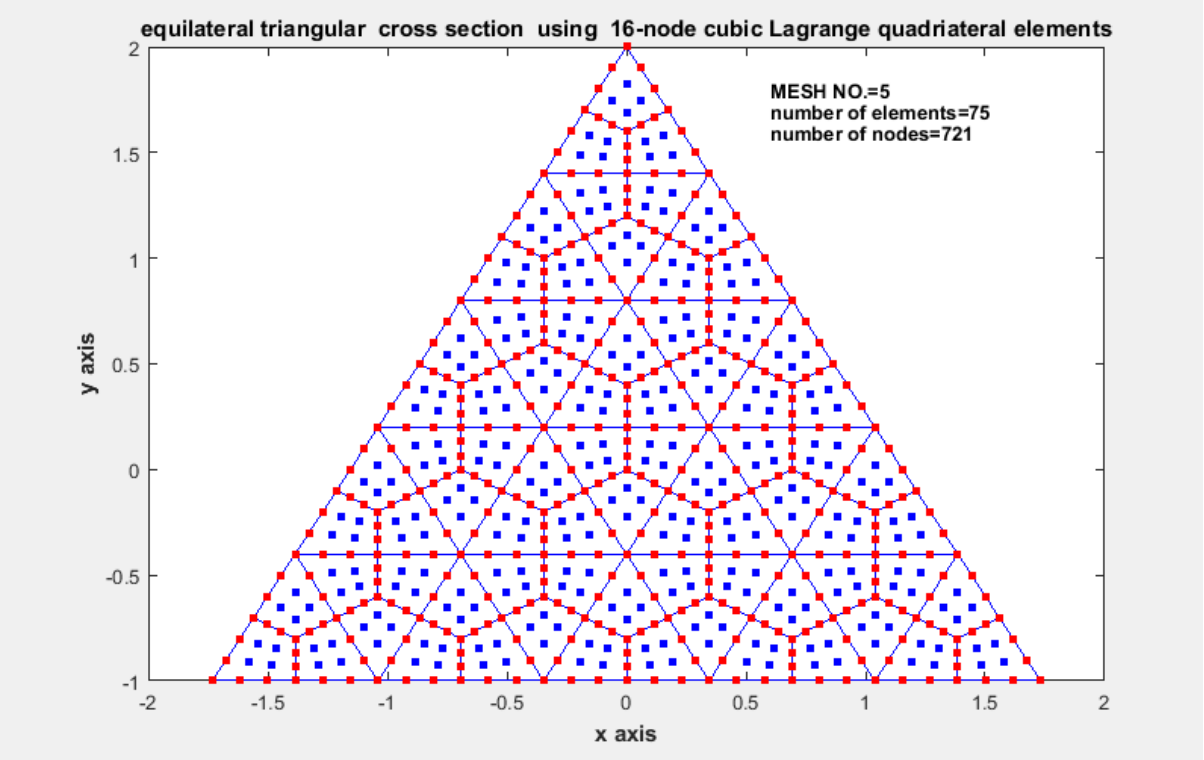

contour level curves for exact solution:
Y-axis
X-axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5


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
Y-axis
X-axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

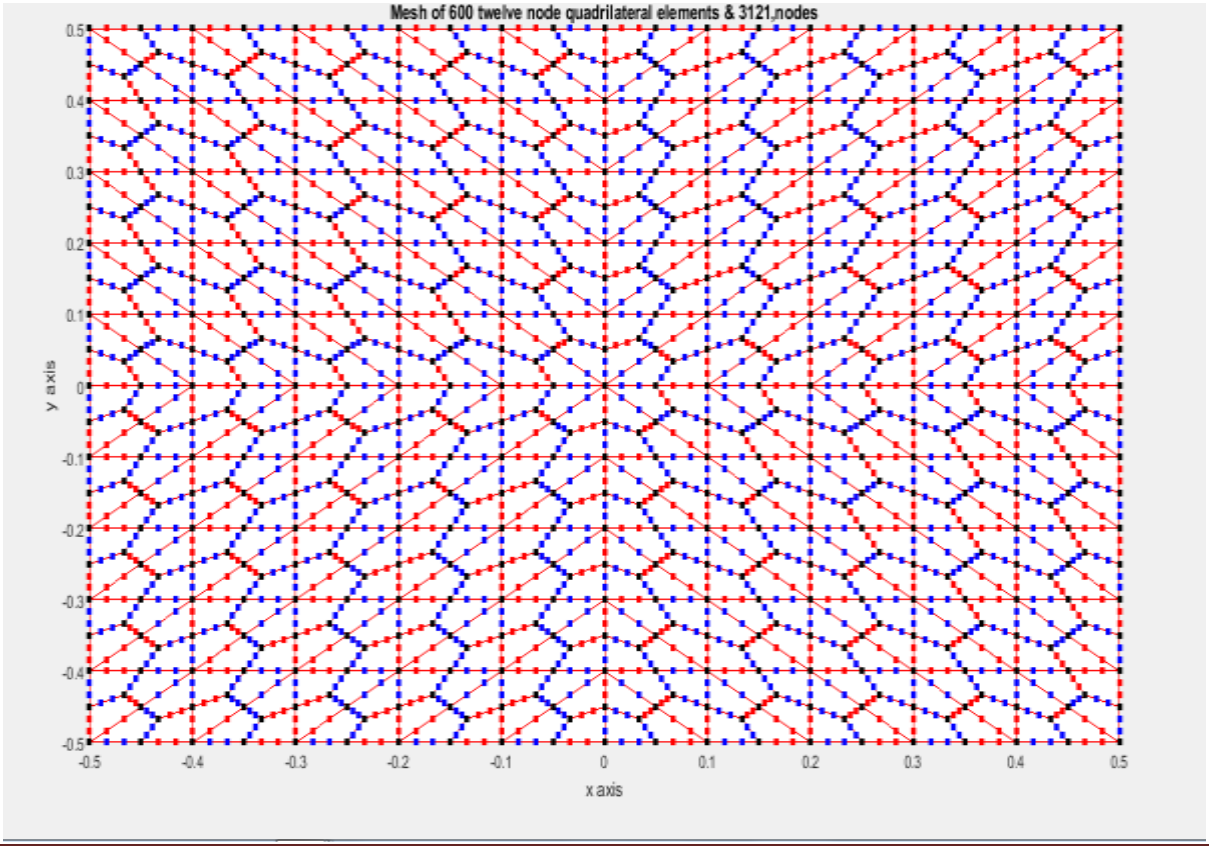

equilateral triangular cross section using 12-node cubic serendipity quadriateral elements
MESH NO.=1
number of elements=3
number of nodes=25
(1)
(2)
(3)
y axis
x axis

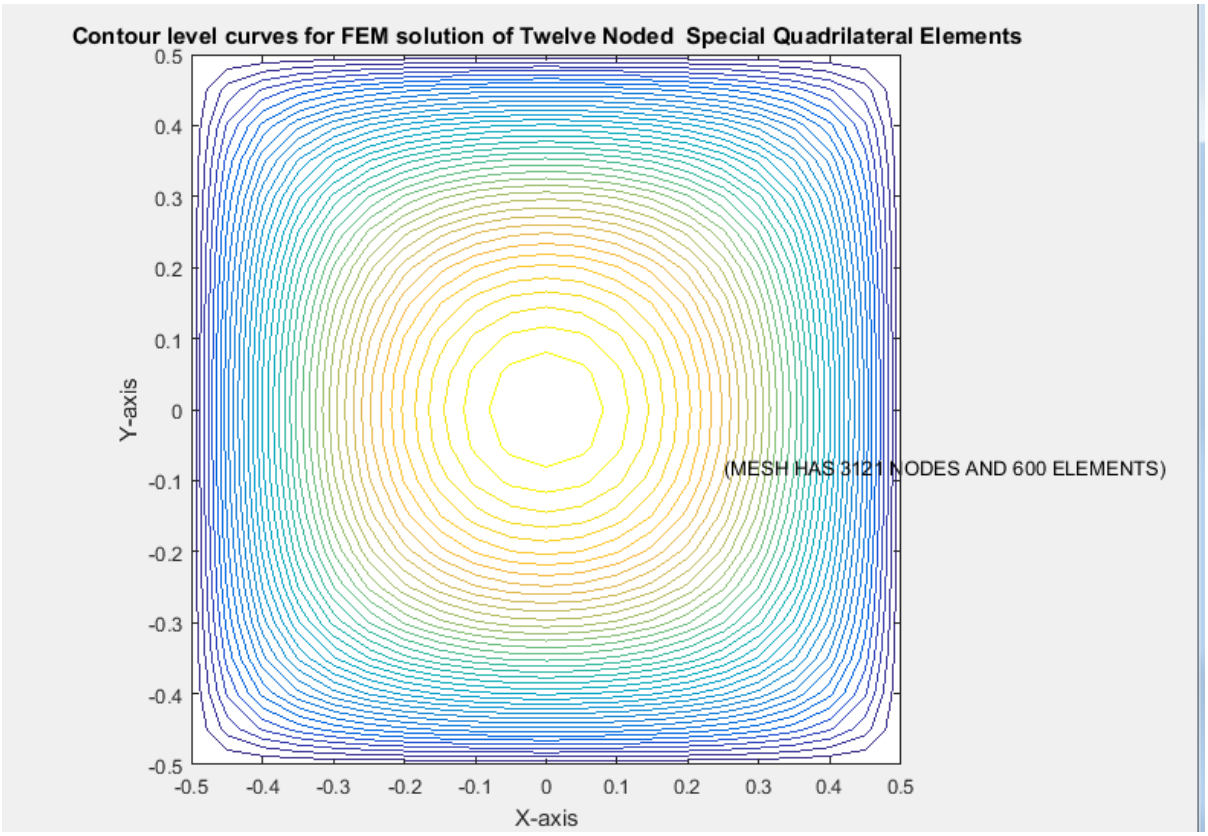

equilateral triangular cross section using 16-node cubic Lagrange quadriateral elements
MESH NO.=1
number of elements=3
number of nodes=37
y axis
x axis
(1)
(2)
(3)

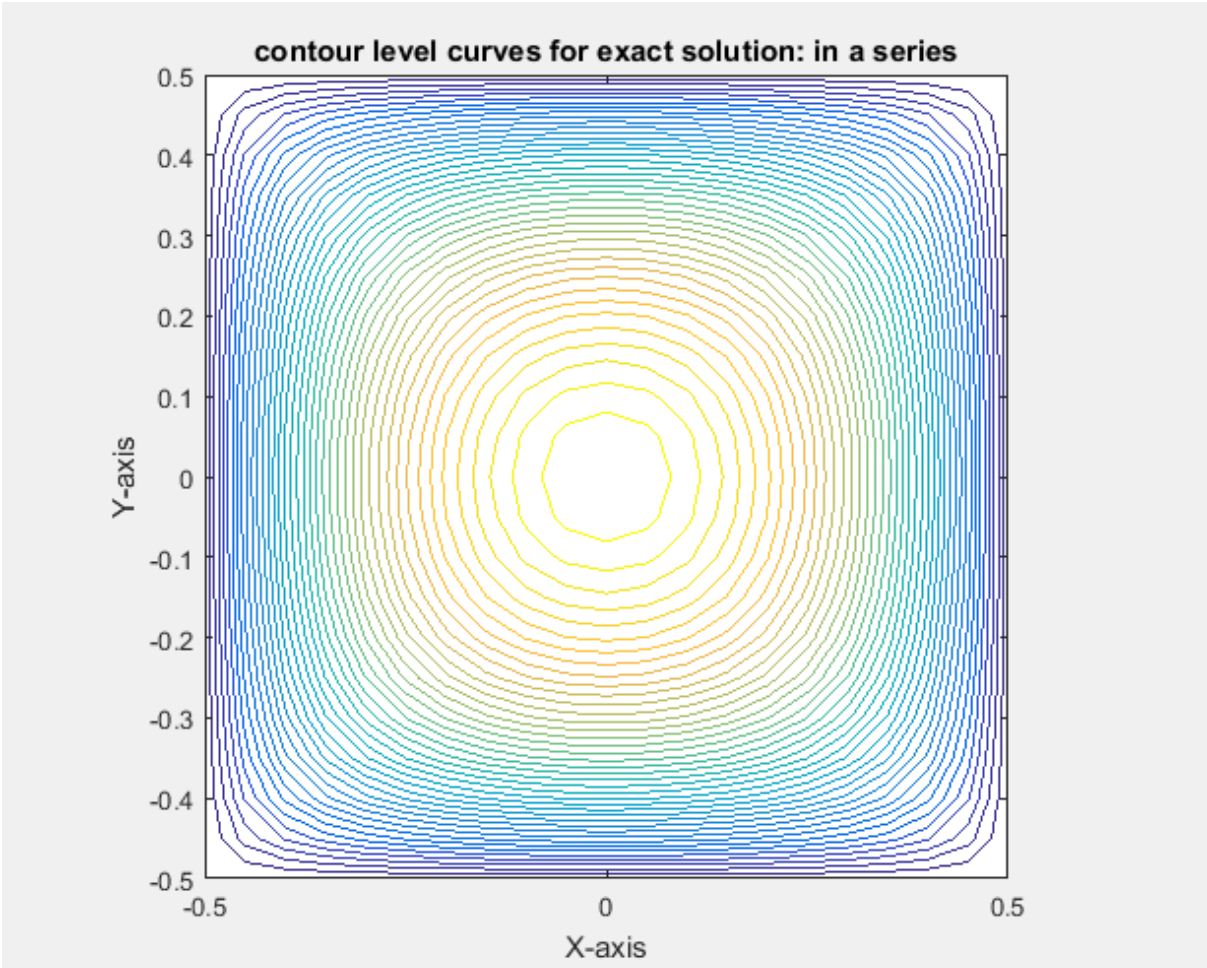

equilateral triangular cross section using 12-node cubic serendipity quadriateral elements
MESH NO.=5
number of elements=75
number of nodes=421
y axis
x axis

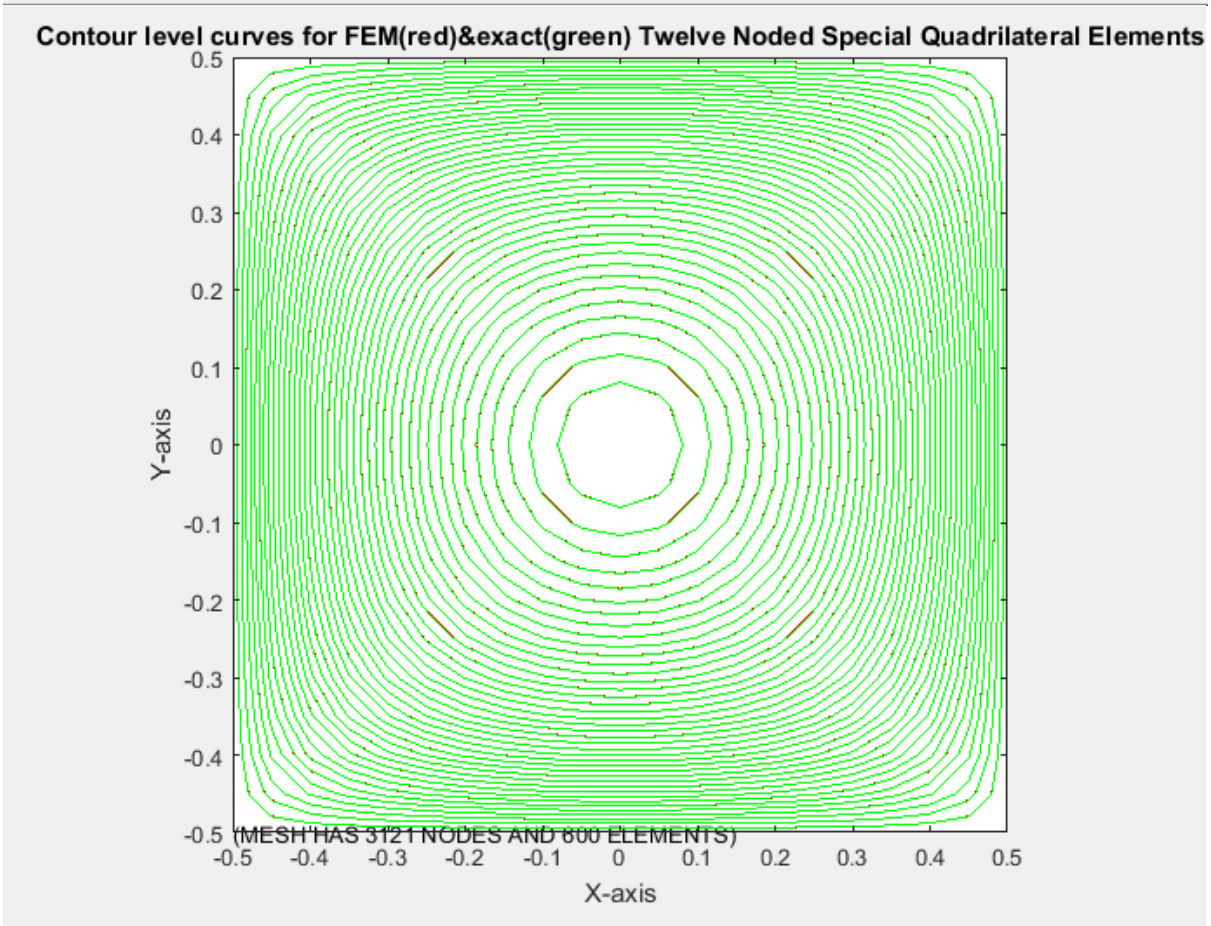


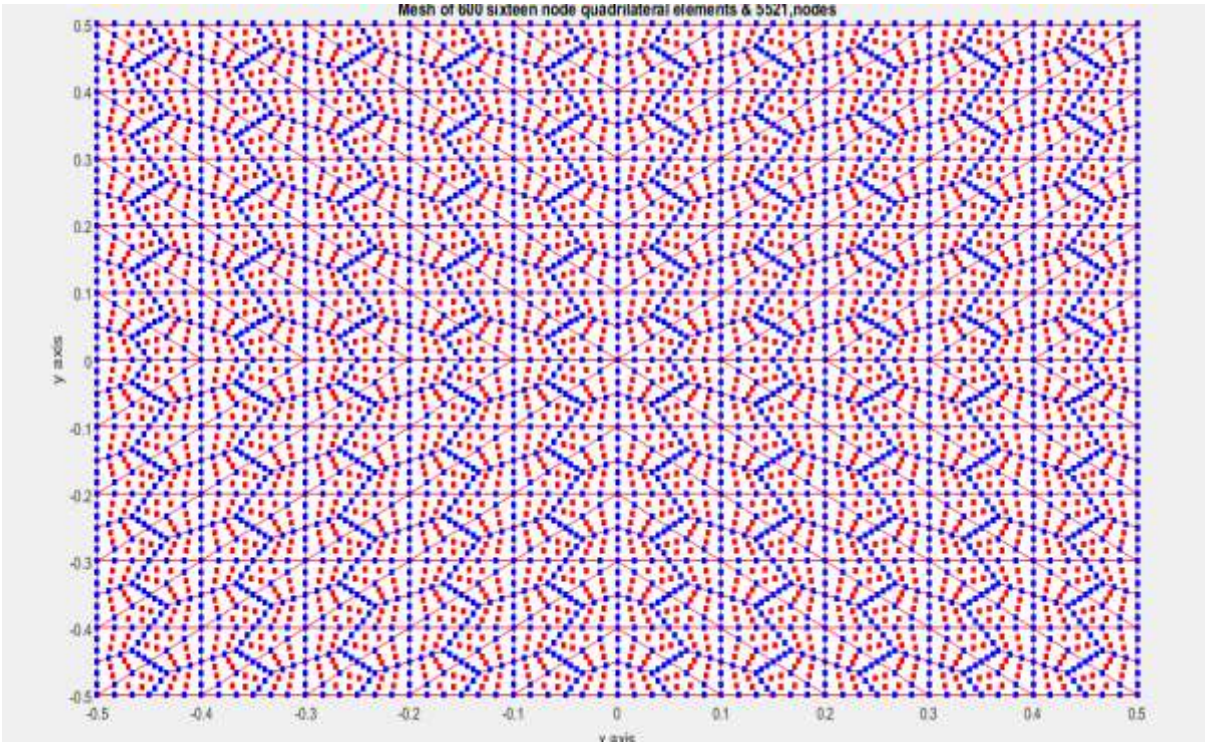
Example5:TORSION OF A SQUARE CROSS SECTION



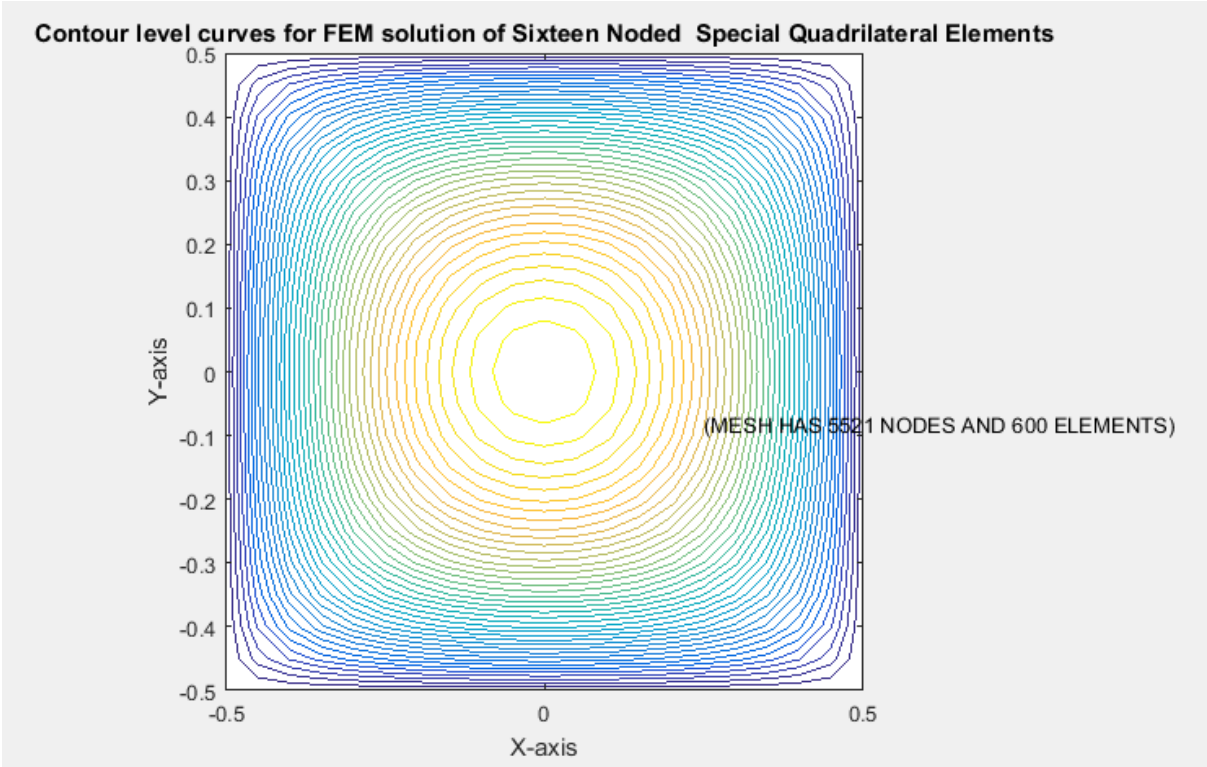

Contour level curves for FEM solution of Twelve Noded Special Quadrilateral Elements
(MESH HAS 3121 NODES AND 600 ELEMENTS)
Y-axis
X-axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5

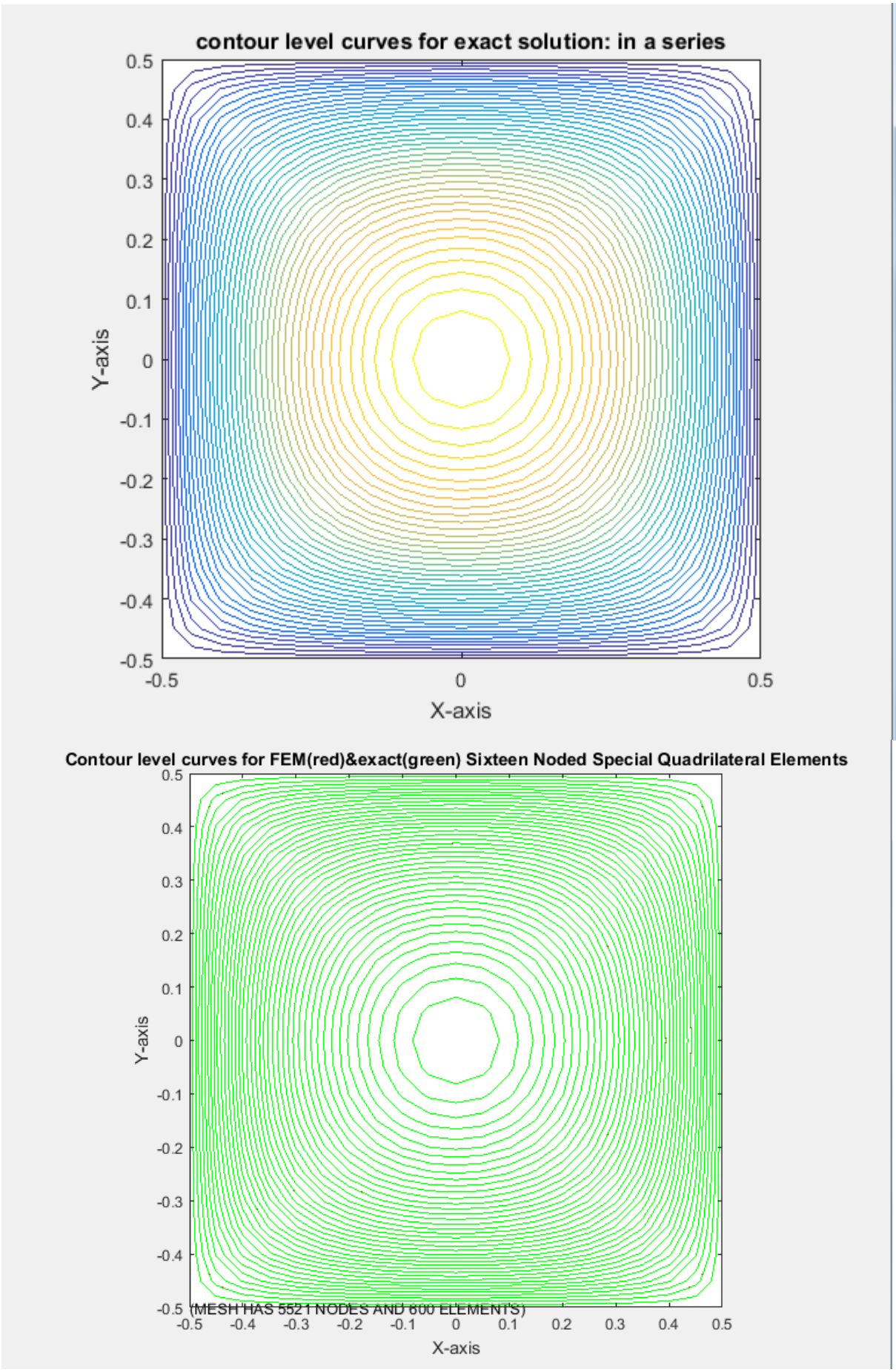

contour level curves for exact solution: in a series
Y-axis
X-axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.5
0
0.5


Contour level curves for FEM(red)&exact(green) Twelve Noded Special Quadrilateral Elements
(MESH HAS 3121 NODES AND 600 ELEMENTS)
Y-axis
X-axis
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5


Mesh of 600 sixteen node quadrilateral elements & 5521,nodes
y axis
x axis

X-AXIS

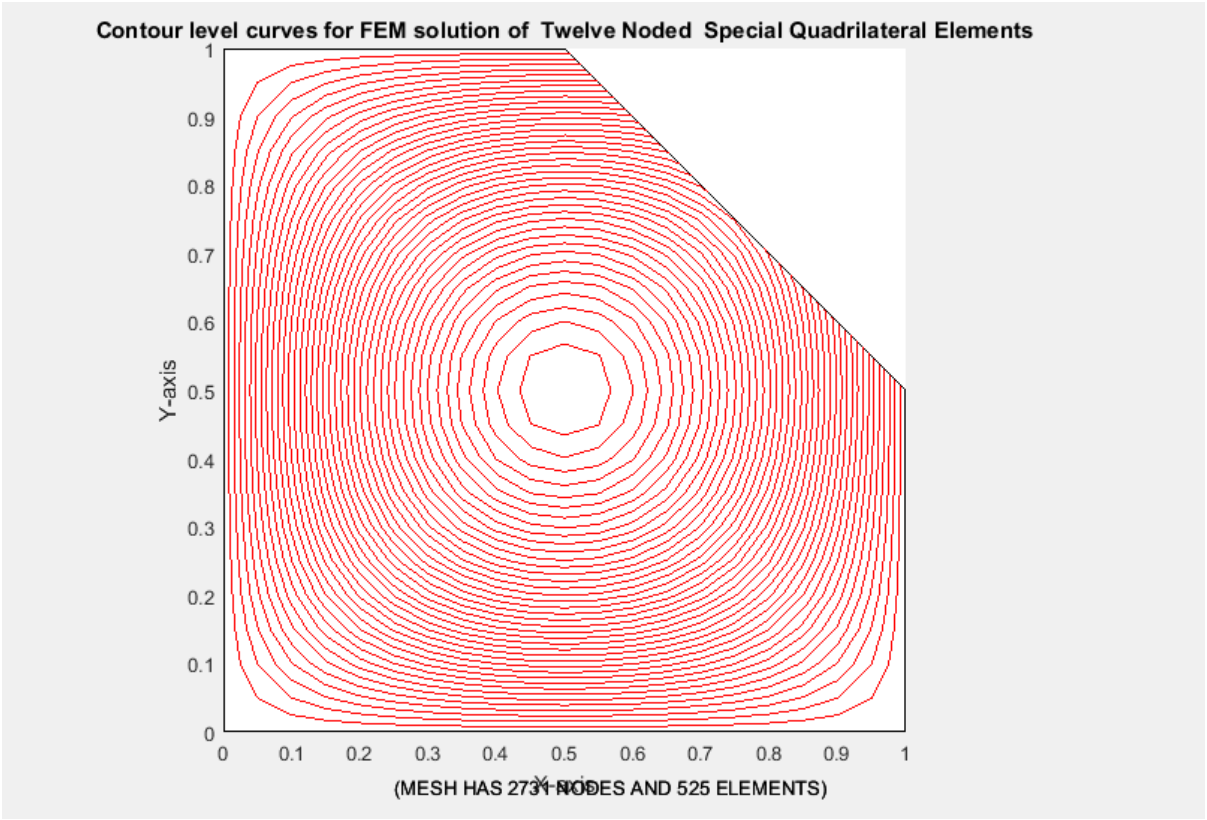

Contour level curves for FEM solution of Sixteen Noded Special Quadrilateral Elements
Y-axis
X-axis
(MESH HAS 5521 NODES AND 600 ELEMENTS)

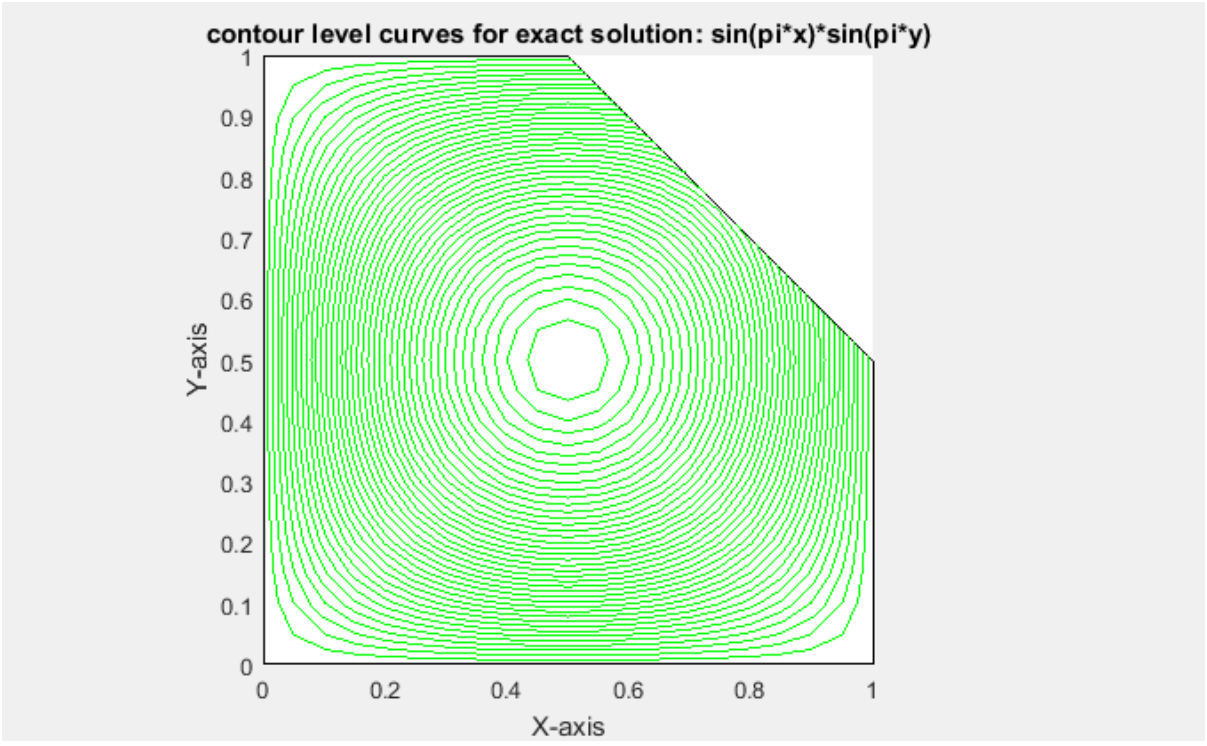


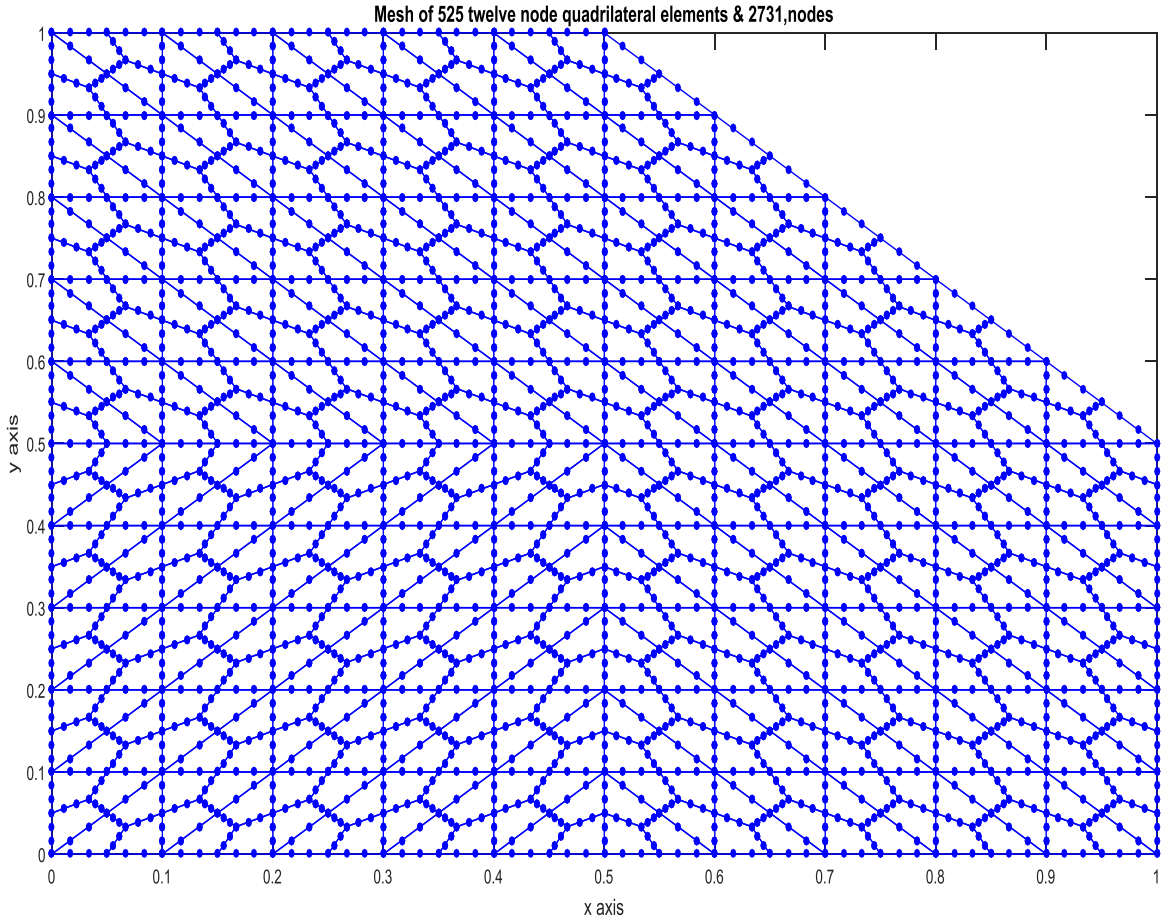
Solution of Poisson Boundary Value Problems Over Polygonal Domains

MESHES AND CONTOUR LEVEL CURVES FOR A PENTAGONAL DOMAIN WITH 12-NO0DED QUADRILATERAL ELEMENTS

Mesh with 525 twelve noded quadrilateral elements & no. of nodes = 2731

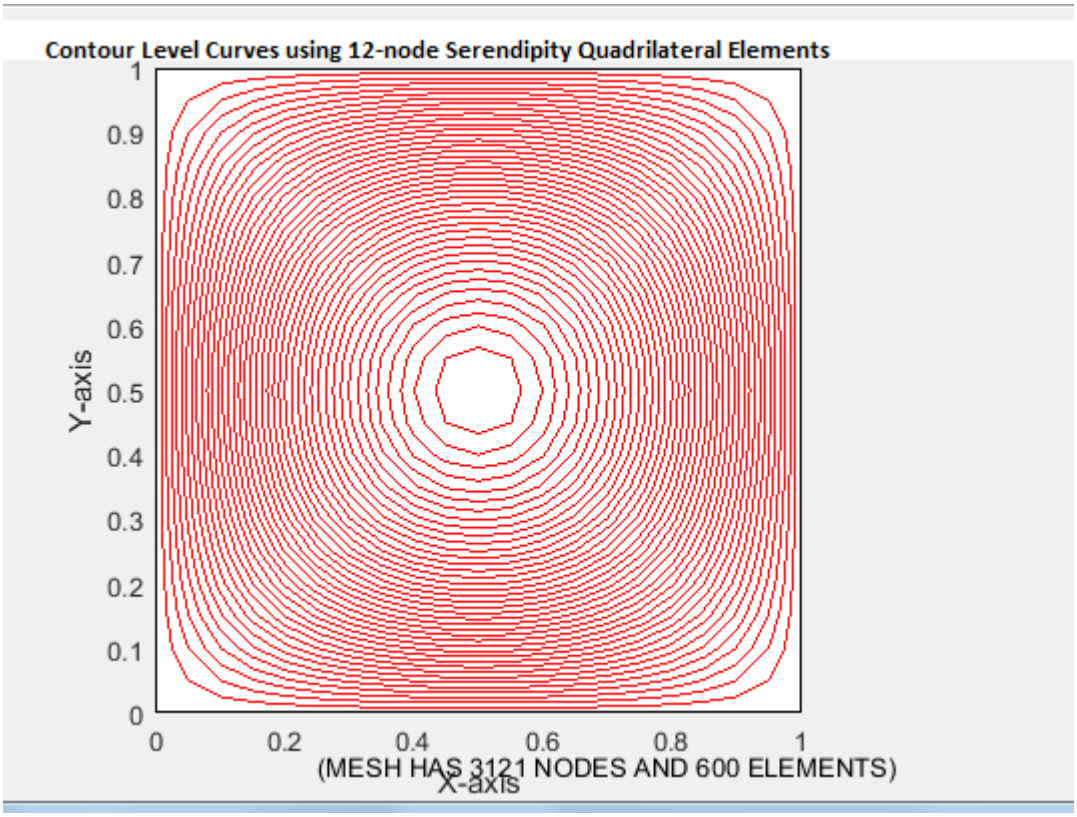

Contour level curves for FEM solution of Twelve Noded Special Quadrilateral Elements
Y-axis
X-axis
(MESH HAS 2731 NODES AND 525 ELEMENTS)

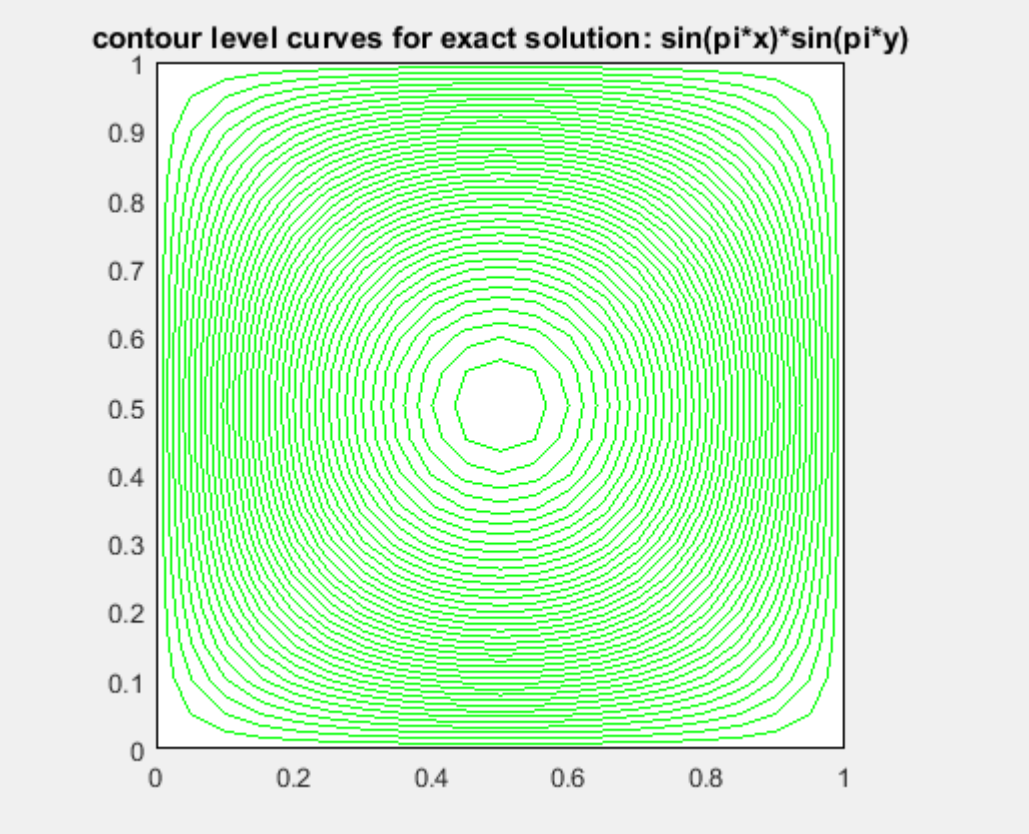

contour level curves for exact solution: sin(pi*x)*sin(pi*y)
Y-axis
X-axis

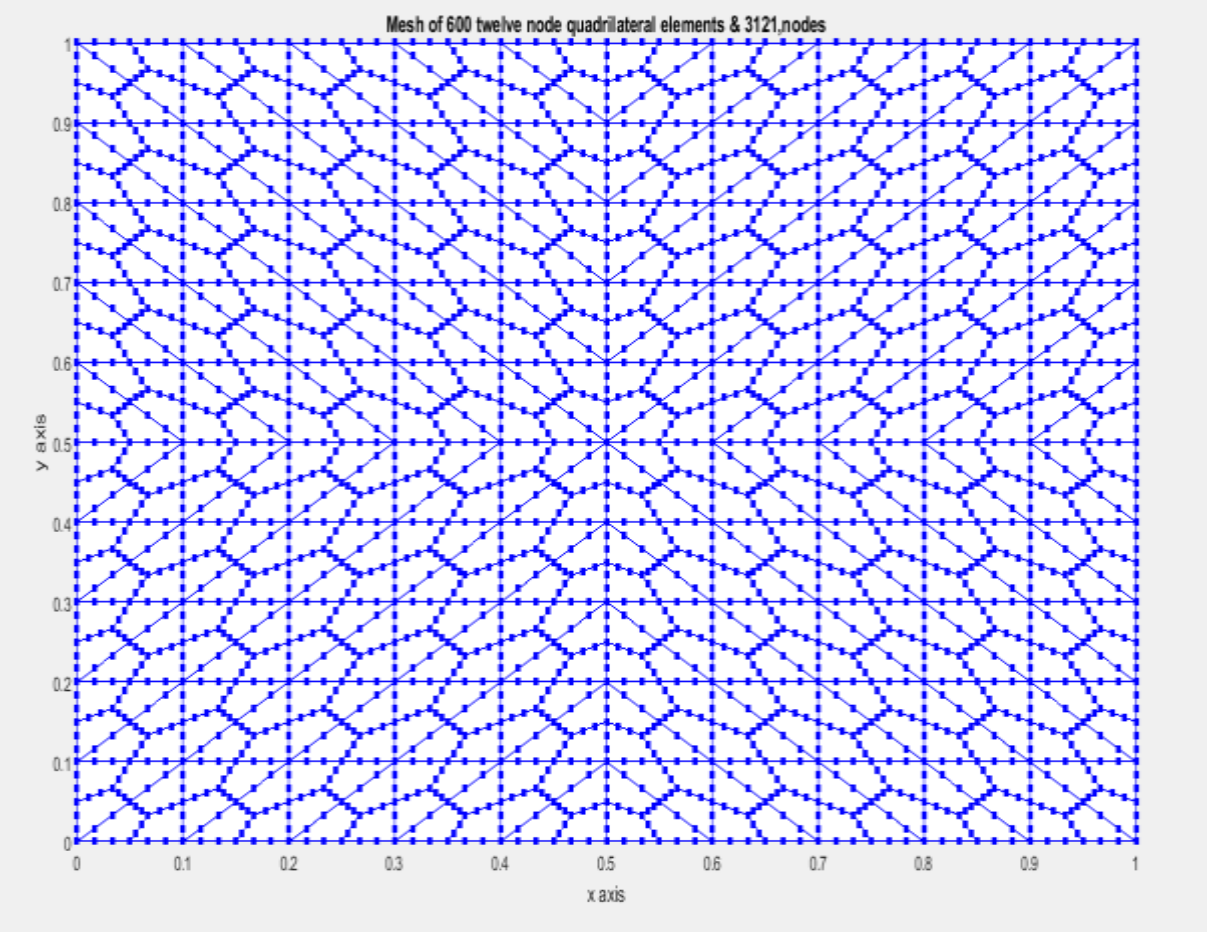


MESHES AND CONTOUR LEVEL CURVES FOR A SQUARE DOMAIN WITH TWELVE NO0DED QUADRILATERAL ELEMENTS

Mesh with 600 twelve noded quadrilateral elements & no. of nodes = 3121

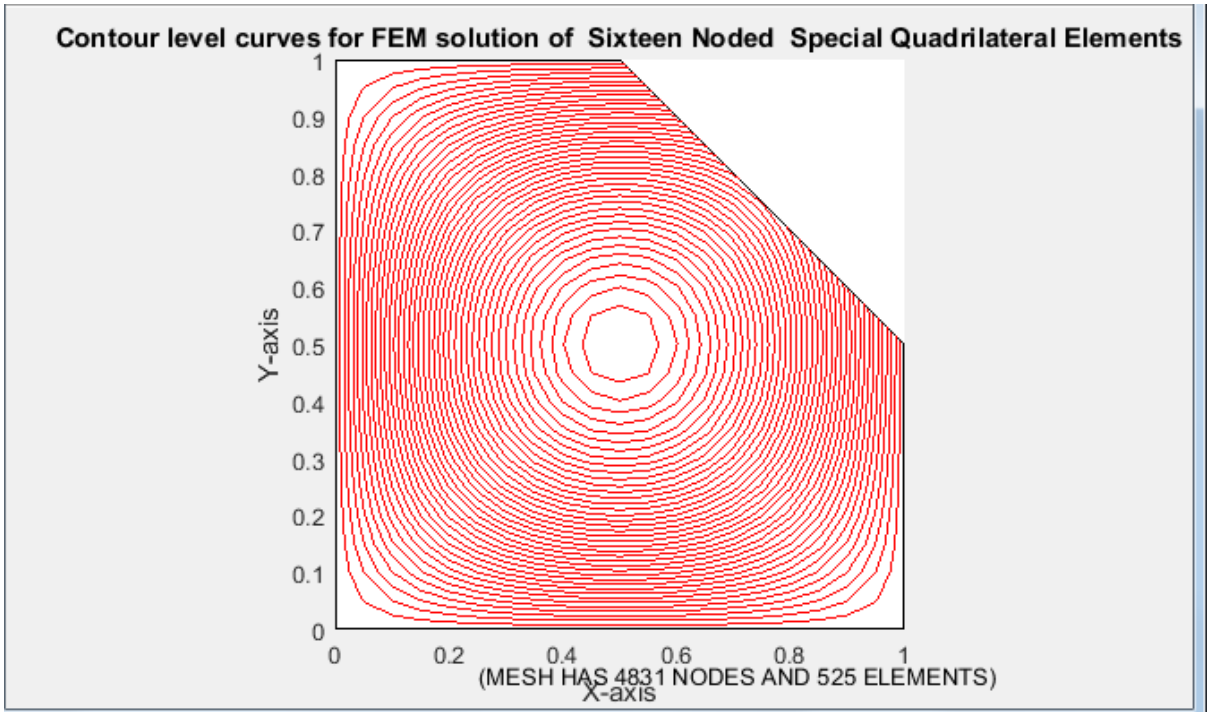


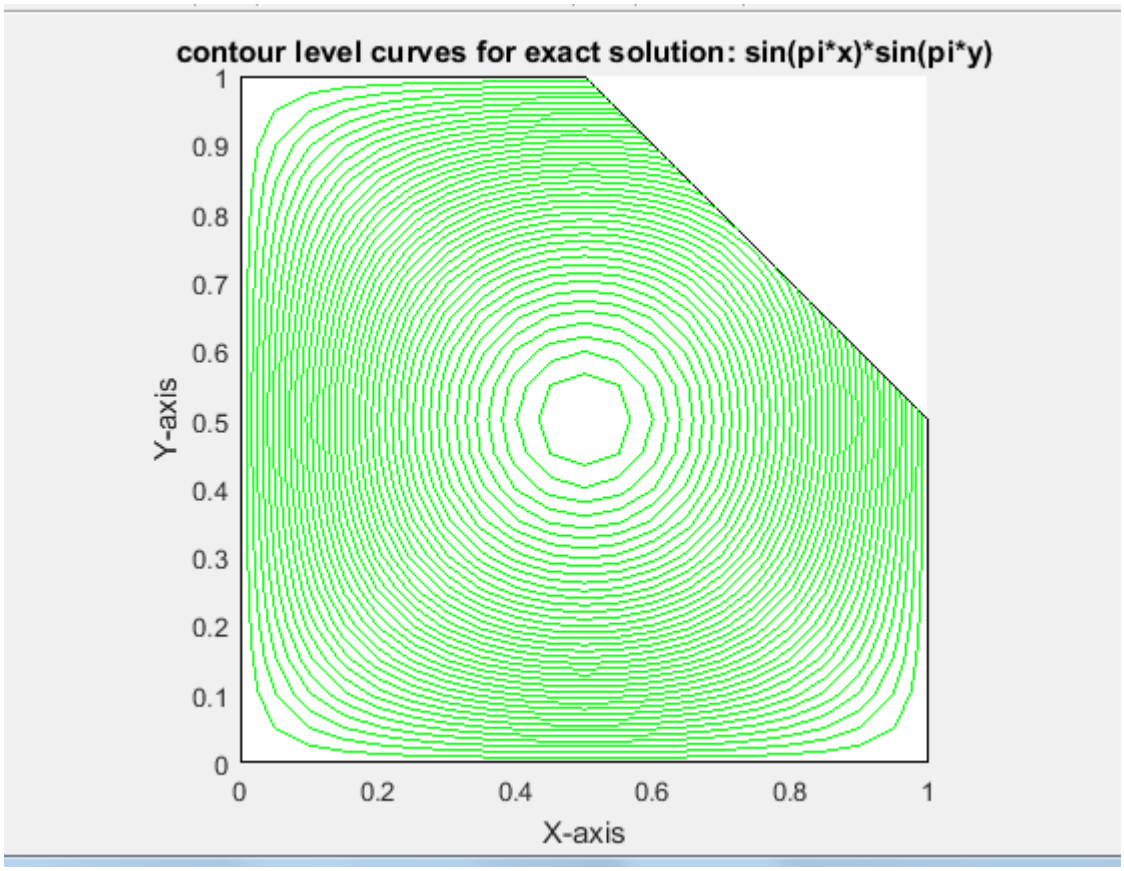


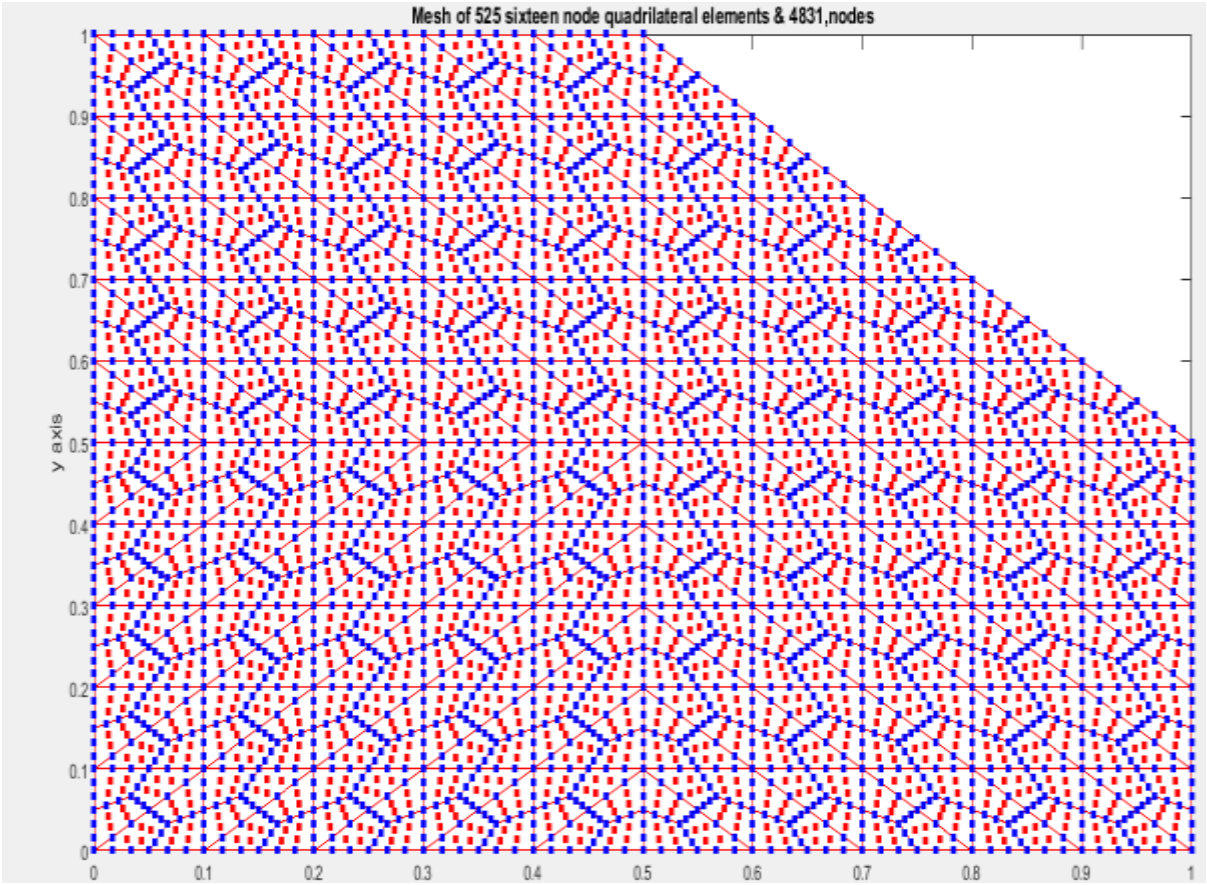


MESHES AND CONTOUR LEVEL CURVES FOR A PENTAGONAL DOMAIN WITH 16-NO0DED QUADRILATERAL ELEMENTS

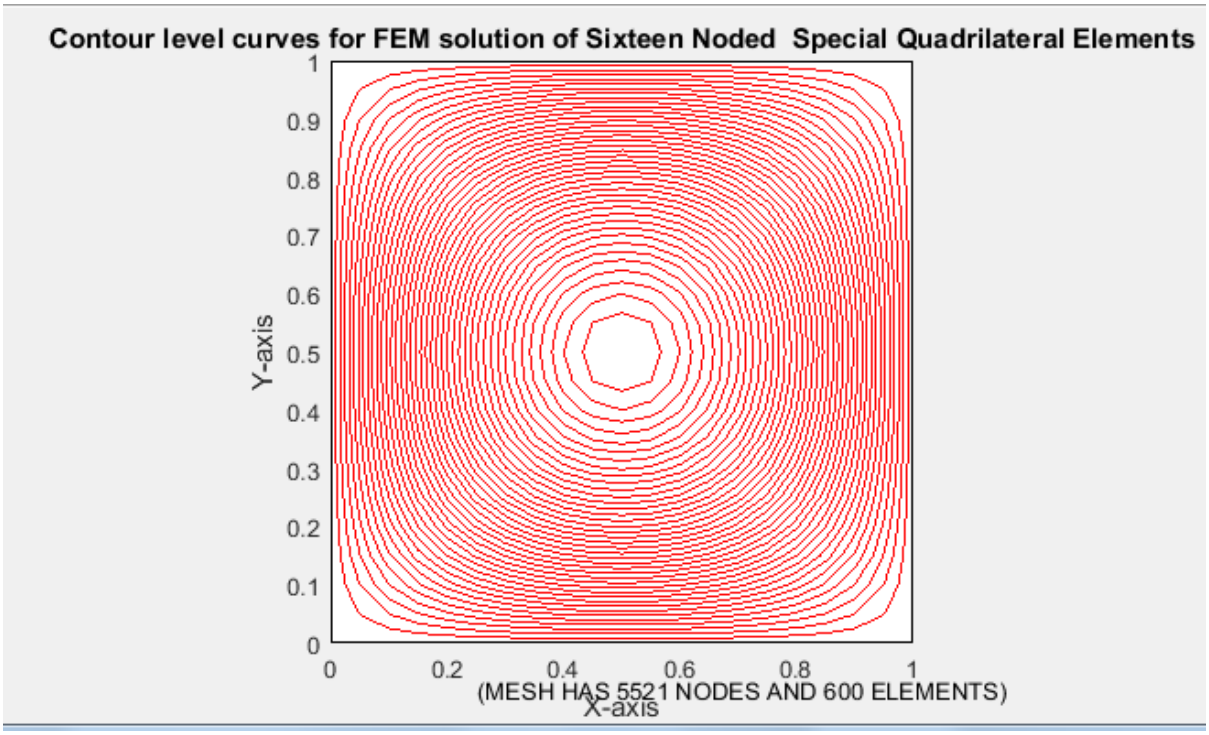
Mesh with 525 sixteen noded quadrilateral elements & no. of nodes = 4831

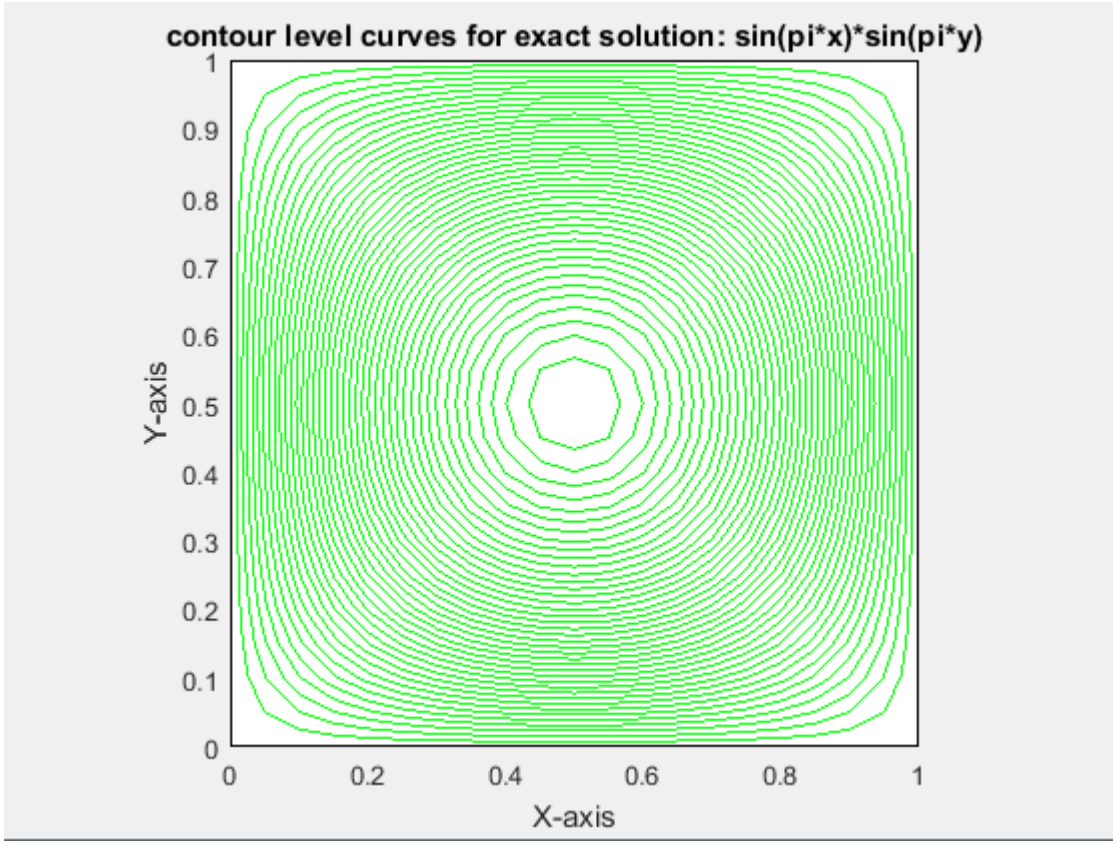


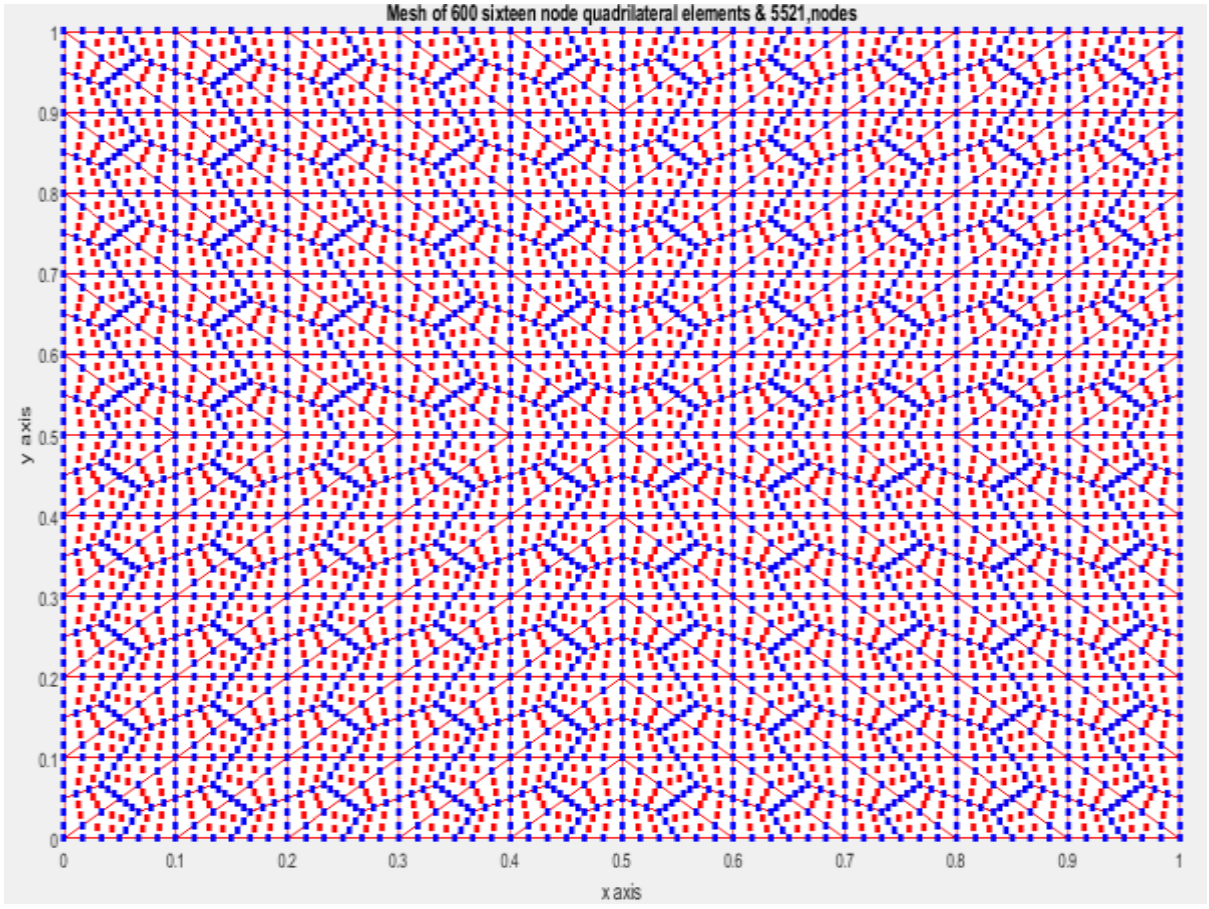




MESHES AND CONTOUR LEVEL CURVES FOR A SQUARE DOMAIN WITH SIXTEEN NO0DED QUADRILATERAL ELEMENTS







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

[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

MATLAB CODES

[1]quadrilateralmesh_over_arbitrarytriangle_q12automeshgen.m

```
function[]=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)
   ndiv=2*mesh;
 %[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_order(ndiv)
 [eln,nodetel,nodes,nnode]=nodaladdresses_special_convex_quadrilaterals_3rd_order(ndiv);
%[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh_2ndorder(ndiv);
%[coord,gcoord]=coordinate_arbitrarytriangle_3rdorderLAGR(xx,yy,ndiv)
[coord,gcoord]=coordinate_arbitrarytriangle_3rdorder(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
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,['\bf(',num2str(i),'\bf)']);
end
%************************************
%*******************************
end;%i loop
switch tri
   case 1
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfone eigth (1/8)square cross section ';
st2=' using  ';
st3='12-node cubic serendipity ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
   case 2
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfone eigth (1/8)square cross section ';
st2=' using  ';
st3='12-node cubic serendipity ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
```

```
   case 3
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])
   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
%**************************************
%**********************************
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(0.1,0.4,['\bfMESH NO.=',num2str(mesh)])
text(0.1,0.38,['\bfnumber of elements=',num2str(nel)])
text(0.1,0.36,['\bfnumber of nodes=',num2str(nnode)])
case 3
 text(0.65,0.8,['\bfMESH NO.=',num2str(mesh)])
text(0.65,0.75,['\bfnumber of elements=',num2str(nel)])
text(0.65,0.70,['\bfnumber of nodes=',num2str(nnode)])
case 4
 text(0.6,1.8,['\bfMESH NO.=',num2str(mesh)])
text(0.6,1.7,['\bfnumber of elements=',num2str(nel)])
text(0.6,1.6,['\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)]);
text(gcoord(jj,1),gcoord(jj,2),[num2str(jj)]);
else
%text(gcoord(jj,1),gcoord(jj,2),['\bf.']);
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;%j loop
figure(mesh),scatter(x(1,1:nnel),y(1,1:nnel),20,'filled','b')
%figure(mesh),scatter(x(1,1:nnel),y(1,1:nnel),'MarkerFaceColor','g')
%figure(mesh),scatter(x(1,nnel-3:nnel),y(1,nnel-3:nnel),'MarkerFaceColor','r')
end%i loop
%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_rtisoscelestriangle00_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,['\bf(',num2str(i),'\bf)']);
end
end;%i loop
switch tri
   case 1
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfone eigth (1/8)square cross section ';
st2=' using  ';
st3='16-node cubic Lagrange ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
   case 2
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfone eigth (1/8)square cross section ';
st2=' using  ';
st3='16-node cubic Lagrange ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
   case 3
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfequilateral triangular  cross section ';
st2=' using  ';
st3='16-node cubic Lagrange ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
   case 4
xlabel('\bfx axis')
ylabel('\bfy axis')
st1='\bfequilateral triangular  cross section ';
st2=' using  ';
st3='16-node cubic Lagrange ';
st4='quadriateral';
st5=' elements'
title([st1,st2,st3,st4,st5])
end


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(0.1,0.4,['\bfMESH NO.=',num2str(mesh)])
text(0.1,0.38,['\bfnumber of elements=',num2str(nel)])
text(0.1,0.36,['\bfnumber of nodes=',num2str(nnode)])
case 3
 text(0.65,0.8,['\bfMESH NO.=',num2str(mesh)])
text(0.65,0.75,['\bfnumber of elements=',num2str(nel)])
text(0.65,0.70,['\bfnumber of nodes=',num2str(nnode)])
case 4
 text(0.6,1.8,['\bfMESH NO.=',num2str(mesh)])
text(0.6,1.7,['\bfnumber of elements=',num2str(nel)])
text(0.6,1.6,['\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;%j loop
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%i loop
%figure(mesh),scatter(gcoord(:,1),gcoord(:,2),'MarkerFaceColor','g')
%axis off
end%for nmesh-the number of meshes
```

[3]quadrilateral_mesh4MOINEX_q12.m

```
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],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_q12([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%quadrilateral_mesh4MOINEX_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,nnel]=size(nodes);
disp([xlength,ylength,nnode,nel,nnel])
%gcoord(i,j),where i->node no. and j->x or y
%_____

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

```
clf
for i=1:nel
for j=1:nnel
x(1,j)=xcoord(nodes(i,j),1);
y(1,j)=ycoord(nodes(i,j),1);
end;%j loop
xvec(1,1:5)=[x(1,1),x(1,2),x(1,3),x(1,4),x(1,1)];
yvec(1,1:5)=[y(1,1),y(1,2),y(1,3),y(1,4),y(1,1)];
axis 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
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),y(1,1:nnel),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']);
```

```
%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(1,1:4),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%i loop
end
```

[4]quadrilateral_mesh4MOINEX_q16LG.m

```
function[]=quadrilateral_mesh4MOINEX_q16LG(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;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],[3;4;5;6;7;8;2],8,1,2,1,1,1)
%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],9,25,10,2,1,1)
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_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
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=' 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%i loop
end
```

[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)]))
%syms ff ss f sk N NN table1 table2
%D2LaplaceEquationQ12Ex3automeshgenNewContour(1,2,3,1,2,1)
%D2LaplaceEquationQ12Ex3automeshgenNewContour(1,2,3,1,2,2)
%
%**************************************************************************
syms coord
syms x y
ndof=1;


switch mesh
    case 1
    x=sym([0;1/2;1/2])
    y=sym([0;0;1/2])
    case 2  %isoscles triangle(torsion of an equilateral triangle,each side=2*sqrt(3))
 x=sym([-sqrt(3);sqrt(3); 0])
 y=sym([     -1;      -1; 2])
end
syms ui vi wi xi yi
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_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,nodes,nnode]=nodaladdresses_special_convex_quadrilaterals_3rd_order(ndiv);
 [nel,nnel]=size(nodes);
%disp([nel nnode nnel ndof])
format long g
for i=1:nel
N(i,1)=i;
end
for i=1:nel
NN(i,1)=i;
end



sdof=nnode*ndof;
```

```
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));
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);
%*************************************************************8

%***********************************
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(0.140577014955155551037840396020329);
 xi=(zeros(nnode,1));
a0=8/pi^3;
for m=1:nnode
   gx=(gcoord(m,1));gy=(gcoord(m,2));rr=(0);
for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*gy)/cosh(n*pi/2)))*cos(n*pi*gx)/n^3;
end
xi(m,1)=(a0*rr);
end
 mm=length(bcdof);

   case 2%torsion of an equilateral triangle

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

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

end%switch



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*ndof,1);

X=xx(iel,1:3);
Y=yy(iel,1:3);
%disp([X Y])
xa=X(1,1);
xb=X(1,2);
xc=X(1,3);
ya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
bta=yb-yc;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;

sk(1:12,1:12)=(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; 37/960;  7/240;   1/30; 23/960]*(2*delabc);
```

```
%_____________________________________
 edof=nnel*ndof;
 k=0;
 for i=1:nnel
     nd(i,1)=nodes(iel,i);
     start=(nd(i,1)-1)*ndof;
     for j=1:ndof
         k=k+1;
         index(k,1)=start+j;
     end
 end
 %-----------------------------------------------------------------------
for i=1:edof
    ii=index(i,1);
    ff(ii,1)=ff(ii,1)+f(i,1);
 for j=1:edof
     jj=index(j,1);
     ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
 %-----------------------------------------------------------------------
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,1:nnode)=zeros(1,nnode);
    ss(1:nnode,kk)=zeros(nnode,1);
    ff(kk,1)=0;
end
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,kk)=1;
end
phi=ss\ff;
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                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);
%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;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);
          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);
%[t]=convexquadrilateral_naturalcoordinates([1/3;0;0;1/2],[1/3;1/2;0;0],U,V)
[t,iter] = newtonmethod4spquadrileteral(t0,p,q,'parameqnspqd','paramdetJspqd','paraminvJspqd');

         % X=x(i,j);Y=y(i,j);
          %[t]=convexquadrilateral_coordinates(u,v,X,Y);
          r=t(1,1);
          s=t(2,1);
shn1=((1-r)*(1-s)*(-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);
          z(i,j)=((Y+1)*((sqrt(3))*X-Y+2)*(-(sqrt(3))*X-Y+2))/12;;
           break
         end%if (in==1)
       end%for iel
  %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
   end%for j
end%for i
 % z=sin(pi*x).*sin(pi*y);
  %z=(zeros(31,31));

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

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

   end
end


end%(mesh==2)

switch mesh
   case 2
  clf,figure(1)
   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>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
%[c,h]=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
```

```
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='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(0.6,1.8,[sst1 sst2])
text(0.6,1.6,[sst3 sst4])
text(0.6,1.4,[sst5 sst6])
%text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
%
figure(2)
 %x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
 x=[-sqrt(3);sqrt(3);0];
 y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%[c,h]=contour(x,y,z)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);

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

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

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

```
sst4=num2str(nel)
text(0.6,1.1,[sst1 sst2])
text(0.6,.9,[sst3 sst4])

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

 mm=0;
 for i=1:31
  for j=1:31
     mm=mm+1;
     femsoln(mm,1)=PHI(i,j);
     exactsoln(mm,1)=z(i,j);
  end
end
end
% [femsoln exactsoln]
 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=2,4,6,8,.....
%polygonal_domain_coordinates([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2)
%polygonal_domain_coordinates([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4)
%D2LaplaceEquationQ4MoinExautomeshgen(n1,n2,n3,nmax,numtri,ndiv)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,4,4,2)
%quadrilateral_mesh4MOINEX_q4(n1,n2,n3,nmax,numtri,ndiv,mesh,xlength,ylength)([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6
;7;8;9;2],9,1,2,2,1,1)
%D2POISSONEQUATION_NODALINTERPOLATION_VALUES(n1,n2,n3,nmax,numtri,ndiv,mesh)([1;1;1;1;1;1;1;1],[2;3;4;5
;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%D2LaplaceEquationQ4MoinExautomeshgen([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,100,20,2)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,4,4,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,9,6,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,16,8,1)
%D2PoissonEquationQ4MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,25,10,1)
%D2PoissonEquationQ8MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%D2PoissonEquationQ8MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1,1,10)
%D2PoissonEquationQ9MoinEx_MeshgridContour([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,2,1,10)
%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)
```

```
%D2LaplaceEquationQ9Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4)
%D2LaplaceEquationQ16Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4)
%D2LaplaceEquationQ12Ex3automeshgenNewPolygonContour([1;1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8;9],[3;4;5;6;7;8;9;2],9,1,2,4)
syms coord
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
%nnel=4;
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order(n1,n2,n3,nmax,numtri,ndiv,mesh)
nnel=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));

%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
%>>improvedLaplace EquationQuad4twodimensionEx2_explicit(69,83,4,1)#
%>>improvedLaplaceEquationQuad4twodimensionEx2_explicitfnmesh(69,83,4,1)#
%improvedLaplaceEquationQuad4twodimensionEx2_explicitvfnmesh(72,87,4,1)#new
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=3,nnode=7,nnel=4,ndof=1)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel,nnode,nnel=4,ndof=1,quadtype=0/3,mesh=1,2,3...)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=12,nnode=19,nnel=4,ndof=1,quadtype=0/3,mesh=3)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=27,nnode=37,nnel=4,ndof=1,quadtype=0/3,mesh=4)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=48,nnode=61,nnel=4,ndof=1,quadtype=0/3,mesh=5)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(nel=75,nnode=91,nnel=4,ndof=1,quadtype=0/3,mesh=6)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(108,127,4,1,3,7)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(147,169,4,1,3,8)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(192,217,4,1,3,9)
%improvedLaplaceEquationQuad4twodimensionEx3_explicitmesh(243,271,4,1,3,10)
disp([nel nnode nnel ndof])
format long g
for i=1:nel
N(i,1)=i;
end
for i=1:nel
NN(i,1)=i;
end
   %[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh(ndiv);
   %[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
   %
   %bcdof=[2;5;3]
  %boundary conditions-1
 switch mesh


case 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=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-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);
 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

  end %switch
 for L=1:nel
  for M=1:3
    LM=nodetel(L,M);
    xx(L,M)=gcoord(LM,1);
    yy(L,M)=gcoord(LM,2);
   end
 end
%_____________________________________________________________
%ng=10

table2=[N xx yy];
%disp([xx yy])
%_____________________________________________________________________________________
___
%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);
ya=Y(1,1);
yb=Y(1,2);
yc=Y(1,3);
bta=yb-yc;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;

sk(1:12,1:12)=(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; 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:nnel
     nd(i,1)=nodes(iel,i);
     start=(nd(i,1)-1)*ndof;
     for j=1:ndof
         k=k+1;
         index(k,1)=start+j;
     end
 end
 %-----------------------------------------------------------------------
for i=1:edof
    ii=index(i,1);
    ff(ii,1)=ff(ii,1)+f(i,1);
 for j=1:edof
     jj=index(j,1);
     ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
 %-----------------------------------------------------------------------
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,1:nnode)=zeros(1,nnode);
    ss(1:nnode,kk)=zeros(nnode,1);
    ff(kk,1)=0;
end
for ii=1:mm
    kk=bcdof(ii,1);
    ss(kk,kk)=1;
end
phi=ss\ff;
```

```
phi=double(phi);

[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')
[nnode nel nnel ndof]
disp('_____')
disp('                fem-computed values          anlytical(theoretical)-values          ')

disp([NN phi xi])
disp('_____')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
nodes
gcoord
%************************************************
disp([NN phi xi])
t=0;
 for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
 end
 T=t;
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('-------------------------------------------------------------------------------------------')
%end
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;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];
         %[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] = newtonmethod4spquadrileteral(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;
     femsoln(mm,1)=PHI(i,j);
     exactsoln(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(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(-1/2,-1/2,[sst1 sst2 sst3 sst4 sst5 sst6])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,40,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square


end%switch mesh
%[femsoln exactsoln]

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

disp([NN phi xi])
disp('_____________________________________________________________________________________')

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

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)));
format long e
t=0;
 for iii=1:nnode
    t=t+phi(iii,1)*ff(iii,1);
 end
 T=t;
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('-------------------------------------------------------------------------------------------')
end
disp('_____________________________________________________________________________________________')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
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)=(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           anlytical(theoretical)-values          ')

disp([elnumm  elcentr phicpt xicpt])
disp('_____________________________________________________________________________________________')

format compact
%=============================================================================================
======
disp('_____________________________________________________________')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
disp('_____________________________________________________________________________________________
_____________________________________')
disp('     ELM FEM SOLUTION EXACT SOLUTION  ELM FEM SOLUTION EXACT SOLUTION ELCENTNODE 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+2) phicpt(I+2) xicpt(I+2)])
    %disp([A B C])
    fprintf('\n%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('\n')
disp('-------------------------------------------------------------------------------------------------------------------------------------------- --')
```

[7]D2LaplaceEquationQ16Ex3automeshgenNewContour.m

```
function[]=D2LaplaceEquationQ16Ex3automeshgenNewContour(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)]))
%syms ff ss f sk N NN table1 table2
%D2LaplaceEquationQ16Ex3automeshgenNew(1,2,3,1,2,1)
%D2LaplaceEquationQ16Ex3automeshgenNew(1,2,3,1,2,2)
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,1)
%D2LaplaceEquationQ16Ex3automeshgenNewContour(1,2,3,1,2,2)

syms coord
ndof=1;
%*************************************************************************8
%*********************************************************
```

```
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  % an equilateral triangle,each side=2*sqrt(3)
 x=sym([-sqrt(3);sqrt(3); 0])
 y=sym([     -1;      -1; 2])


end
syms ui vi wi xi yi
%[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_order(ndiv);
%[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_2nd_orderLAGR(ndiv)
[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_orderLAGR(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_orderLAGR(n)
%[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_3rd_order(n)
%*************************************************************
%syms coord
%ndof=1;

%[eln,nodetel,nodes,nnode]=nodaladdresses4Lagrangespecial_convex_quadrilaterals_2nd_order(ndiv);
%[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh_2ndorderLAGR(ndiv);

   %[coord,gcoord]=coordinate_rtisoscelestriangle00_h0_hh(ndiv);
   %[nodetel,nodes]=nodaladdresses4special_convex_quadrilaterals(ndiv)
[nel,nnel]=size(nodes);
%disp([nel nnode nnel ndof])
format long g
for i=1:nel
N(i,1)=i;
end
for i=1:nel
NN(i,1)=i;
end
```

```
sdof=nnode*ndof;
ff=(zeros(sdof,1));ss=(zeros(sdof,sdof));

format long g
for i=1:nel
N(i,1)=i;
end
%radius of the hole=1.25cm
%input data for nodal coordinate values
%gcoord(i,j),where i->node no. and j->x or y

%
table1=[N nodes]
[nel,nnel]=size(nodes);
%************************************
switch mesh
   case 1
   nnn=0;
   for nn=1:nnode
      if gcoord(nn,1)==(1/2)
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
      end
   end
   format long g
k1 =double(0.140577014955155510378403960203 29);
 xi=(zeros(nnode,1));
a0=8/pi^3;
for m=1:nnode
   gx=(gcoord(m,1));gy=(gcoord(m,2));rr=(0);
for n=1:2:99
rr=rr+(-1)^((n-1)/2)*(1-(cosh(n*pi*gy)/cosh(n*pi/2)))*cos(n*pi*gx)/n^3;
end
xi(m,1)=(a0*rr);
end
 mm=length(bcdof);

   case 2%torsion of an equilateral triangle

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

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

end%switch
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];
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;btb=yc-ya;
gma=xc-xb;gmb=xa-xc;
delabc=gmb*bta-gma*btb;
G=[bta btb;gma gmb]/delabc;
GT=[bta gma;btb gmb]/delabc;
Q=GT*G;

sk(1:16,1:16)=(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:nnel
    nd(i,1)=nodes(iel,i);
    start=(nd(i,1)-1)*ndof;
    for j=1:ndof
       k=k+1;
       index(k,1)=start+j;
    end
 end
 %-------------------------------------------------------------------------
for i=1:edof
   ii=index(i,1);
   ff(ii,1)=ff(ii,1)+f(i,1);
 for j=1:edof
    jj=index(j,1);
    ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
 %-------------------------------------------------------------------------
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
for ii=1:mm
   kk=bcdof(ii,1);
   ss(kk,1:nnode)=zeros(1,nnode);
   ss(1:nnode,kk)=zeros(nnode,1);
   ff(kk,1)=0;
end
for ii=1:mm
   kk=bcdof(ii,1);
   ss(kk,kk)=1;
end
phi=ss\ff;
phi=double(phi);
if mesh==2
   phi=phi/2;
end
[phi xi]
for I=1:nnode
NN(I,1)=I;
phi_xi(I,1)=phi(I,1)-xi(I,1);
end
MAXPHI_XI=max(abs(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                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);
%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;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);
          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);
%[t]=convexquadrilateral_naturalcoordinates([1/3;0;0;1/2],[1/3;1/2;0;0],U,V)
[t,iter] = newtonmethod4spquadrileteral(t0,p,q,'parameqnspqd','paramdetJspqd','paraminvJspqd');

         % X=x(i,j);Y=y(i,j);
          %[t]=convexquadrilateral_coordinates(u,v,X,Y);
          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-1/3)/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-1/3)/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*ph
i(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
  %  xx=(x(ii,jj));yy=(y(ii,jj));
%z(ii,jj)=((yy+1/2)*((sqrt(3))*xx-yy+1)*(-(sqrt(3))*xx-yy+1))/6;;
%end %ii
%end%jj

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

   end
end


end%(mesh==2)

switch mesh
   case 2
  clf
figure(1)
 x=[-sqrt(3);sqrt(3);0];
 y=[     -1;     -1;2];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
%[c,h]=contour(x,y,PHI)
contour(x,y,PHI,20)
xlabel('X-axis');
ylabel('Y-axis');
%clabel(c,h);
```

```
axis square
st1='Contour level curves for ';
st2='FEM solution of ';
st3='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(0.6,1.8,[sst1 sst2])
text(0.6,1.6,[sst3 sst4])
text(0.6,1.4,[sst5 sst6])
%text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
%
figure(2)
 %x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
 x=[-sqrt(3);sqrt(3);0];
 y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%[c,h]=contour(x,y,z)
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);

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

figure(3)
% x=[0.0 1.0 1.0 0.5 0.0];
 %y=[0.0 0.0 0.5 1.0 1.0];
x=[-sqrt(3);sqrt(3);0];
y=[      -1;     -1;2];
 patch(x,y,'w')
hold on
[x,y]=meshgrid(-sqrt(3):(1/15)*sqrt(3):sqrt(3),-1:(0.1):2);
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%y((y>1/2)&(y<=1)&(x>1/2)&(x<=1)&(x+y>3/2))=NaN;
%%y((y>-1/2)&(y<=1)&(x>0)&(x<=(sqrt(3)/2))&((-sqrt(3)*x-y+1)<0))=NaN;
%%y((y>-1/2)&(y<=1)&(x>(-sqrt(3)/2))&(x<=0)&((sqrt(3)*x-y+1)<0))=NaN;
contour(x,y,PHI,40,'r-')

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=' NODES='
sst2=num2str(nnode)
sst3=' ELEMENTS='
```

```
sst4=num2str(nel)
text(0.6,1.1,[sst1 sst2])
text(0.6,.9,[sst3 sst4])

hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,40,'b-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square
text(0.6,1.9,'{ SUPERPOSITION OF }')
text(0.6,1.7,'{ FEM/EXACT SOLUTIONS}')
text(0.6,1.5,'--(red)FEM ')
text(0.6,1.3,'--(blue)EXACT')

 mm=0;
 for i=1:31
  for j=1:31
     mm=mm+1;
     femsoln(mm,1)=PHI(i,j);
     exactsoln(mm,1)=z(i,j);
  end
end
end
% [femsoln exactsoln]
 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)
%ndiv=2,4,6,8,.....
%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)
%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],9,25,10,4)


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
    %[coord,gcoord]=coordinate_rtisoscelestriangle00_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(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=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-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);
 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

  end %switch
 for L=1:nel
  for M=1:3
    LM=nodetel(L,M);
    xx(L,M)=gcoord(LM,1);
    yy(L,M)=gcoord(LM,2);
  end
 end
```

```
%_____________________________________________________________
%ng=10

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

```
 %-----------------------------------------------------------------
%bcdof=[13;37;35;33;31;29;27;25;23;21;19;17;15];
for ii=1:mm
   kk=bcdof(ii,1);
   ss(kk,1:nnode)=zeros(1,nnode);
   ss(1:nnode,kk)=zeros(nnode,1);
   ff(kk,1)=0;
end
for ii=1:mm
   kk=bcdof(ii,1);
   ss(kk,kk)=1;
end
phi=ss\ff;
phi=double(phi);

[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')
[nnode nel nnel ndof]
disp('_____________________________________________________________________')
disp('                fem-computed values          anlytical(theoretical)-values          ')

disp([NN phi xi])
disp('_____________________________________________________________________')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
nodes
gcoord
disp([NN phi xi])
t=0;
 for iii=1:nnode
   t=t+phi(iii,1)*ff(iii,1);
 end
 T=t;
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('---------------------------------------------------------------------------------------')
%end
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;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];
 [t,iter] = ewtonmethod4spquadrileteral(t0,p,q,'parameqnspqd','paramdetJspqd','paraminvJspqd')
         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-1/3)/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-1/3)/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*ph
i(nd14,1)+shn15*phi(nd15,1)+shn16*phi(nd16,1);
           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)
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='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])
hold on
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)
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;
     femsoln(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,-1/2:0.05:1/2)
%[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='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 sst2 sst3 sst4 sst5 sst6])
hold on
%[x,y]=meshgrid(0:.1:1,0:0.1:1)
%[c,h]=contour(x,y,z,'g-')
contour(x,y,z,40,'g-')
%xlabel('X-axis');
%ylabel('Y-axis');
%clabel(c,h);
axis square


end%switch mesh
%[femsoln exactsoln]

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

disp([NN phi xi])
disp('_____')

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

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('-------------------------------------------------------------------------------------------')
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('-------------------------------------------------------------------------------------------')
end
disp('_____________________________________________________________________________________________')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
for ncpt=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           anlytical(theoretical)-values          ')

disp([elnumm  elcentr phicpt xicpt])
disp('_____________________________________________________________________________________________')

format compact
%=============================================================================================
======
disp('_____________________________________________________________________')
disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
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+2) phicpt(I+2) xicpt(I+2)])
    %disp([A B C])
    fprintf('\n%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('\n')
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 choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
```

```
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial(n1=1,n2=2,n3=3,nmax=3,n=2,4,6,...)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,4,4)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,9,6)
%[eln,spqd]=nodaladdresses_special_convex_quadrilaterals_trial([1;1;1;1],[2;3;4;5],[3;4;5;2],5,16,8)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([1;1;1;1],[2;3;4;5],[3;4;5;2],5,1,2,3)
%[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],[2;3;4;5;6;7;8],[3;4;5;6;7;8;
2],8,1,2,1)
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_2nd_order([1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;
2],8,4,4,1)
%[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
syms x y
switch mesh
case 1%domain with seven triangles(8-nodes)
x=sym([1/2;1/2;1;  1;1/2;0;  0;0])%for MOIN EXAMPLE
y=sym([1/2;  0;0;1/2;  1;1;1/2;0])%for MOIN EXAMPLE
case 2%square domain with eight triangles(9-nodes)
x=sym([1/2;1/2;1;  1;  1;1/2;0;  0;0])%FOR UNIT SQUARE
y=sym([1/2;  0;0;1/2;  1;  1;1;1/2;0])%FOR UNIT SQUARE

case 3%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;1/2;  0;-1/2;-1/2;-1/2])%FOR UNIT SQUARE
y=sym([0;-1/2;-1/2;  0;1/2;1/2; 1/2;   0;-1/2])%FOR UNIT SQUARE


end
[eln,spqd,rrr,nodes,nodetel]=nodaladdresses_special_convex_quadrilaterals_trial_3rd_orderLG(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(nel)])

xi(1:nnode,1)=zeros(nnode,1);yi(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)
%
```

```
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)
  mmm9=nodes(mmm,9)
  mmm10=nodes(mmm,10)
  mmm11=nodes(mmm,11)
  mmm12=nodes(mmm,12)
  %
  mmm13=nodes(mmm,13)
  mmm14=nodes(mmm,14)
  mmm15=nodes(mmm,15)
  mmm16=nodes(mmm,16)
  %
  xi1=xi(mmm1,1)
  xi2=xi(mmm2,1)
  xi3=xi(mmm3,1)
  xi4=xi(mmm4,1)
  %
  yi1=yi(mmm1,1)
  yi2=yi(mmm2,1)
  yi3=yi(mmm3,1)
  yi4=yi(mmm4,1)
  %
 xi(mmm5,1)=(2*xi1+xi2)/3;xi(mmm6,1)=(xi1+2*xi2)/3;
 xi(mmm7,1)=(2*xi2+xi3)/3;xi(mmm8,1)=(xi2+2*xi3)/3;
```

```
 xi(mmm9,1)=(2*xi3+xi4)/3;xi(mmm10,1)=(xi3+2*xi4)/3;
 xi(mmm11,1)=(2*xi4+xi1)/3;xi(mmm12,1)=(xi4+2*xi1)/3;
 xi(mmm13,1)=(4*xi1+2*xi2+xi3+2*xi4)/9;
 xi(mmm14,1)=(2*xi1+4*xi2+2*xi3+xi4)/9;
 xi(mmm15,1)=(xi1+2*xi2+4*xi3+2*xi4)/9;
 xi(mmm16,1)=(2*xi1+xi2+2*xi3+4*xi4)/9;
 yi(mmm5,1)=(2*yi1+yi2)/3;yi(mmm6,1)=(yi1+2*yi2)/3;
 yi(mmm7,1)=(2*yi2+yi3)/3;yi(mmm8,1)=(yi2+2*yi3)/3;
 yi(mmm9,1)=(2*yi3+yi4)/3;yi(mmm10,1)=(yi3+2*yi4)/3;
 yi(mmm11,1)=(2*yi4+yi1)/3;yi(mmm12,1)=(yi4+2*yi1)/3;
 yi(mmm13,1)=(4*yi1+2*yi2+yi3+2*yi4)/9;
 yi(mmm14,1)=(2*yi1+4*yi2+2*yi3+yi4)/9;
 yi(mmm15,1)=(yi1+2*yi2+4*yi3+2*yi4)/9;
 yi(mmm16,1)=(2*yi1+yi2+2*yi3+4*yi4)/9;

end%for nel
%[xi(18,1) yi(18,1)]

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

[10]polygonal_domain_coordinates_3rd_order.m

```
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 choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%PARVIZ MOIN EXAMPLE
syms U V W xi yi
syms x y
switch mesh
case 1%domain with seven triangles(8-nodes)
x=sym([1/2;1/2;1;  1;1/2;0;  0;0])%for MOIN EXAMPLE
y=sym([1/2;  0;0;1/2;  1;1;1/2;0])%for MOIN EXAMPLE
case 2%square domain with eight triangles(9-nodes)
x=sym([1/2;1/2;1;  1;  1;1/2;0;  0;0])%FOR UNIT SQUARE
y=sym([1/2;  0;0;1/2;  1;  1;1;1/2;0])%FOR UNIT SQUARE

case 3%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;1/2;  0;-1/2;-1/2;-1/2])%FOR UNIT SQUARE
y=sym([0;-1/2;-1/2;  0;1/2;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(nel)])

xi(1:nnode,1)=zeros(nnode,1);yi(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)
%
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)
```

```
  mmm9=nodes(mmm,9)
  mmm10=nodes(mmm,10)
  mmm11=nodes(mmm,11)
  mmm12=nodes(mmm,12)
  xi1=xi(mmm1,1)
  xi2=xi(mmm2,1)
  xi3=xi(mmm3,1)
  xi4=xi(mmm4,1)
  %
  yi1=yi(mmm1,1)
  yi2=yi(mmm2,1)
  yi3=yi(mmm3,1)
  yi4=yi(mmm4,1)
  %
 xi(mmm5,1)=(2*xi1+xi2)/3;xi(mmm6,1)=(xi1+2*xi2)/3;
 xi(mmm7,1)=(2*xi2+xi3)/3;xi(mmm8,1)=(xi2+2*xi3)/3;
 xi(mmm9,1)=(2*xi3+xi4)/3;xi(mmm10,1)=(xi3+2*xi4)/3;
 xi(mmm11,1)=(2*xi4+xi1)/3;xi(mmm12,1)=(xi4+2*xi1)/3;
 yi(mmm5,1)=(2*yi1+yi2)/3;yi(mmm6,1)=(yi1+2*yi2)/3;
 yi(mmm7,1)=(2*yi2+yi3)/3;yi(mmm8,1)=(yi2+2*yi3)/3;
 yi(mmm9,1)=(2*yi3+yi4)/3;yi(mmm10,1)=(yi3+2*yi4)/3;
 yi(mmm11,1)=(2*yi4+yi1)/3;yi(mmm12,1)=(yi4+2*yi1)/3;

end%for nel
%[xi(18,1) yi(18,1)]

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

[11]coordinate_special_quadrilaterals_in_stdtriangle_3rd_orderLAGR.m

```
 function[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_orderLAGR(n)
%n must be even:n=2,4,6,.....
syms ui vi wi
ui(1:3,1)=[0;1;0];
vi(1:3,1)=[0;0;1];
wi(1:3,1)=[1;0;0];
if (n-1)>0
 kk=3;
 for i=1:n-1
    kk=kk+1;
    ui(kk,1)=sym(i/n);
    vi(kk,1)=sym(0);
    wi(kk,1)=sym(1-ui(kk,1)-vi(kk,1));
 end
 kkk=kk;
 for ii=1:n-1
    kkk=kkk+1;
    ui(kkk,1)=sym((n-ii)/n);
    vi(kkk,1)=sym(1-(n-ii)/n);
    wi(kkk,1)=0;
 end;
 kkkk=kkk;
  for iii=1:n-1
    kkkk=kkkk+1;
    ui(kkkk,1)=0;
    vi(kkkk,1)=sym(1-iii/n);
    wi(kkkk,1)=sym(iii/n);
  end
end%if (n-1)>0
if (n-2)>0
```

```
  kkkkk=kkkk;
  for iiii=1:(n-2)
    for jjjj=1:(n-1)-iiii
        kkkkk=kkkkk+1;
        ui(kkkkk,1)=sym(jjjj/n);
        vi(kkkkk,1)=sym(iiii/n);
        wi(kkkkk,1)=sym(1-ui(kkkkk,1)-vi(kkkkk,1));
     end
  end
end%if (n-2)>0
if n==2

    num=(1:6)';
else
  num=(1:kkkkk)';

end
%disp([ui'])
%disp([vi'])
%disp([wi'])
%length(ui)
%length(vi)
%length(wi)
%disp('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(qqq,1)=(2*ui(qq1,1)+ui(qq2,1))/3;
 vi(qqq,1)=(2*vi(qq1,1)+vi(qq2,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,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(qqq,1)=(2*ui(qq2,1)+ui(qq3,1))/3;
 vi(qqq,1)=(2*vi(qq2,1)+vi(qq3,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
```

```
 ui(qqqq,1)=(ui(qq2,1)+2*ui(qq3,1))/3;
 vi(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(qqq,1)=(2*ui(qq3,1)+ui(qq4,1))/3;
 vi(qqq,1)=(2*vi(qq3,1)+vi(qq4,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 ui(qqqq,1)=(ui(qq3,1)+2*ui(qq4,1))/3;
 vi(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(qq4,1)+ui(qq1,1))/3;
 vi(qqq,1)=(2*vi(qq4,1)+vi(qq1,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 ui(qqqq,1)=(ui(qq4,1)+2*ui(qq1,1))/3;
 vi(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;
 vi(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;
 vi(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;
 vi(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;
 vi(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])
[12]coordinate_special_quadrilaterals_in_stdtriangle_3rd_order.m
  function[ui,vi,wi]=coordinate_special_quadrilaterals_in_stdtriangle_3rd_order(n)
%n must be even:n=2,4,6,.....
syms ui vi wi
ui(1:3,1)=[0;1;0];
vi(1:3,1)=[0;0;1];
wi(1:3,1)=[1;0;0];
if (n-1)>0
 kk=3;
 for i=1:n-1
    kk=kk+1;
    ui(kk,1)=sym(i/n);
    vi(kk,1)=sym(0);
    wi(kk,1)=sym(1-ui(kk,1)-vi(kk,1));
 end
 kkk=kk;
 for ii=1:n-1
    kkk=kkk+1;
```

```
    ui(kkk,1)=sym((n-ii)/n);
    vi(kkk,1)=sym(1-(n-ii)/n);
    wi(kkk,1)=0;
 end;
 kkkk=kkk;
  for iii=1:n-1
    kkkk=kkkk+1;
    ui(kkkk,1)=0;
    vi(kkkk,1)=sym(1-iii/n);
    wi(kkkk,1)=sym(iii/n);
  end
end%if (n-1)>0
if (n-2)>0
  kkkkk=kkkk;
  for iiii=1:(n-2)
    for jjjj=1:(n-1)-iiii
        kkkkk=kkkkk+1;
        ui(kkkkk,1)=sym(jjjj/n);
        vi(kkkkk,1)=sym(iiii/n);
        wi(kkkkk,1)=sym(1-ui(kkkkk,1)-vi(kkkkk,1));
     end
  end
end%if (n-2)>0
if n==2

    num=(1:6)';
else
  num=(1:kkkkk)';

end
%disp([ui'])
%disp([vi'])
%disp([wi'])
%length(ui)
%length(vi)
%length(wi)
%disp('first')
%disp([num ui vi wi])
[eln,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
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(qqq,1)=(2*ui(qq1,1)+ui(qq2,1))/3;
 vi(qqq,1)=(2*vi(qq1,1)+vi(qq2,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,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(qqq,1)=(2*ui(qq2,1)+ui(qq3,1))/3;
 vi(qqq,1)=(2*vi(qq2,1)+vi(qq3,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 ui(qqqq,1)=(ui(qq2,1)+2*ui(qq3,1))/3;
 vi(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(qqq,1)=(2*ui(qq3,1)+ui(qq4,1))/3;
 vi(qqq,1)=(2*vi(qq3,1)+vi(qq4,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 ui(qqqq,1)=(ui(qq3,1)+2*ui(qq4,1))/3;
 vi(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(qq4,1)+ui(qq1,1))/3;
 vi(qqq,1)=(2*vi(qq4,1)+vi(qq1,1))/3;
 wi(qqq,1)=1-ui(qqq,1)-vi(qqq,1);
 ui(qqqq,1)=(ui(qq4,1)+2*ui(qq1,1))/3;
 vi(qqqq,1)=(vi(qq4,1)+2*vi(qq1,1))/3;
 wi(qqqq,1)=1-ui(qqqq,1)-vi(qqqq,1);
end
maxnode=max(max(nodes(1:3*nc,1:12)));
 num=(1:maxnode)';
 %disp(['maximum value of node number=',num2str(maxnode)])
%disp(' node    ui     vi     wi')
%disp([num ui vi wi])
```

[13]integral_valuesof_localderivative_products.m

```
function[intJdndn]=integral_valuesof_localderivative_products(nnel)
switch nnel
   case 12
      %integral values of local derivative products

intJdn1dn1uvrs =[vpa(sym('  1.878527434164174128122249654 6')) vpa(sym(' 1.64496677534890315539341209663'));...
            vpa(sym(' 1.64496677534890315539341209663')) vpa(sym('  1.878527434164174128122249654 6'))];
intJdn1dn2uvrs =[vpa(sym(' .24392368058084020798114240290')) vpa(sym(' .25456419075959891292341644890'));...
            vpa(sym(' .16706419075959891292341644890')) vpa(sym('  .49412990575435517453974316 05'))];
intJdn1dn3uvrs =[vpa(sym(' .65143728569018129264176158009e-1')) vpa(sym(' -.80634999408541685813400031 67e-1'));...
            vpa(sym(' -.80634999408541685813400031 67e-1')) vpa(sym(' .65143728569018129264176158009e-1'))];
intJdn1dn4uvrs =[vpa(sym('  .49412990575435517453974316 05')) vpa(sym(' .16706419075959891292341644890'));...
            vpa(sym(' .25456419075959891292341644890')) vpa(sym(' .24392368058084020798114240290'))];
intJdn1dn5uvrs =[vpa(sym(' -1.9170973654180328603393872 2609')) vpa(sym('  -.50073891821327672848240784486'));...
            vpa(sym(' -1.2132389182132767284824078 4486')) vpa(sym('    -.48951593925070344936468 5749'))];
intJdn1dn6uvrs =[vpa(sym('   .37765421933316862052540179 0571')) vpa(sym(' -.78857287231911452268591214 13e-1'));...
            vpa(sym('    .22114271276808854773140878587')) vpa(sym('   .47102462045975434469650 8823e-1'))];
intJdn1dn7uvrs =[vpa(sym('  -.31236802364420048787164339097')) vpa(sym(' -.62259192547029207059718397 2271'));...
            vpa(sym(' -.62259192547029207059718397 2271')) vpa(sym('  -.93264116705106716805029551279'))];
intJdn1dn8uvrs =[vpa(sym('  .22334568260811342445267275929')) vpa(sym(' .41545805218570060363476054 9957'));...
            vpa(sym(' .41545805218570060363476054 9957')) vpa(sym(' .32179538230835884627097507 0300'))];
intJdn1dn9uvrs =[vpa(sym(' .32179538230835884627097507 0300')) vpa(sym(' .41545805218570060363476054 9957'));...
            vpa(sym(' .41545805218570060363476054 9957')) vpa(sym('  .22334568260811342445267275929'))];
intJdn1dn10uvrs =[vpa(sym('  -.93264116705106716805029551279')) vpa(sym(' -.62259192547029207059718397 2271'));...
             vpa(sym(' -.62259192547029207059718397 2271')) vpa(sym('  -.31236802364420048787164339097'))];
intJdn1dn11uvrs =[vpa(sym('   .47102462045975434469650 8823e-1')) vpa(sym('    .22114271276808854773140878587'));...
```

```
            vpa(sym(' -.78857287231911452268591214l3e-1')) vpa(sym('   .377654219333168620525401790571'))];
intJdn1dn12uvrs =[vpa(sym('    -.48951593925070344936468574l9')) vpa(sym(' -1.213238918213276728482407844l86'));...
            vpa(sym('  -.50073891821327672848240784486')) vpa(sym(' -1.917097365418032860339387226l9'))];
intJdn2dn1uvrs =[vpa(sym(' .243923680580840207981142402l0')) vpa(sym(' .167064190759598912923416448l0'));...
            vpa(sym(' .254564190759598912923416448l0')) vpa(sym('  .494129905754355174539743160l'))];
intJdn2dn2uvrs =[vpa(sym(' .927247441752334105622080795189')) vpa(sym(' -.154813416538163025163939357l9'));...
            vpa(sym(' -.15481341653816302516393935759')) vpa(sym('   .794715916478116799729308112l8'))];
intJdn2dn3uvrs =[vpa(sym('   .5124614234253834624957312395l6')) vpa(sym('   .12101159273523234679493182712l'));...
            vpa(sym(' .335115927352323467949318271l4e-1')) vpa(sym('   .156606852059463586417315376026'))];
intJdn2dn4uvrs =[vpa(sym('  .16985339992712677743504195136')) vpa(sym(' .179045055450002326957831462744'));...
            vpa(sym(' .17904505545000232695783146274l')) vpa(sym('  .16985339992712677743504195136'))];
intJdn2dn5uvrs =[vpa(sym('  .737450284572716169349341246171')) vpa(sym('   .360723164794451180672811744l3'));...
            vpa(sym(' .607231647944511806728117444l3e-1')) vpa(sym('  .67618516000239264204780296l6e-1'))];
intJdn2dn6uvrs =[vpa(sym(' -1.895073448163238983109870698571')) vpa(sym('   -.32947059876161316868575194333'));...
            vpa(sym('    .383029401238386831314248056l7')) vpa(sym('  -.610174037597342509867071996e-1'))];
intJdn2dn7uvrs =[vpa(sym(' .893817596079137610724416224l1e-1')) vpa(sym(' -.32942011895369986176171176l0e-1'));...
            vpa(sym('   -.7454420118953699861761711768l')) vpa(sym('  -1.320510591580267436328115452l6'))];
intJdn2dn8uvrs =[vpa(sym(' .660287853857037733479423905l9e-1')) vpa(sym('   .132773165331780684038772441l41'));...
            vpa(sym('   .432773165331780684038772441541')) vpa(sym('   .364521033839059598529804469259'))];
intJdn2dn9uvrs =[vpa(sym(' -.820137977578506638706811693089')) vpa(sym(' .102283035327104635406160239318l'));...
            vpa(sym(' .102283035327104635406160239318l')) vpa(sym(' .364041669725565464086726766l8e-1'))];
intJdn2dn10uvrs =[vpa(sym('    .144533345449155674413769328l7')) vpa(sym(' -.32732059722698593284644960282l6'));...
            vpa(sym(' -.32732059722698593284644960282l6')) vpa(sym('   -.10277602051817855644721686707'))];
intJdn2dn11uvrs =[vpa(sym(' -.7898220003698491703637636243e-1')) vpa(sym('  -.21219256804932529716922346537l'));...
            vpa(sym('  -.21219256804932529716922346537l')) vpa(sym(' .623647580488036249761255273l1e-1'))];
intJdn2dn12uvrs =[vpa(sym('  -.96686494922443392864432222l9e-1')) vpa(sym(' -.61610119267126767523886182l9e-2'));...
            vpa(sym(' -.61610119267126767523886182l9e-2')) vpa(sym('   -.66191053322154112847875205l37'))];
intJdn3dn1uvrs =[vpa(sym('  .651437285690181292641761580l9e-1')) vpa(sym(' -.80634999408541685813400031669e-1'));...
            vpa(sym(' -.8063499940854168581340003166l9e-1')) vpa(sym('  .651437285690181292641761580l9e-1'))];
intJdn3dn2uvrs =[vpa(sym('   .5124614234253834624957312395l6')) vpa(sym(' .335115927352323467949318271l4e-1'));...
            vpa(sym('   .12101159273523234679493182712l')) vpa(sym('   .156606852059463586417315376026'))];
intJdn3dn3uvrs =[vpa(sym(' 1.0542668070025615997463144835803')) vpa(sym('   .675139101205863811454268346790'));...
            vpa(sym('   .67513910120586381145426834679l')) vpa(sym(' 1.0542668070025615997463144835803'))];
intJdn3dn4uvrs =[vpa(sym('   .156606852059463586417315376026')) vpa(sym('   .12101159273523234679493182712l'));...
            vpa(sym(' .335115927352323467949318271l4e-1')) vpa(sym('   .5124614234253834624957312395l6'))];
intJdn3dn5uvrs =[vpa(sym('  .422209070544079358508960005l2')) vpa(sym(' .338079890317619378517715472657'));...
            vpa(sym(' .33807989031761937851771547265l')) vpa(sym('  .120923980597039893815264134l0'))];
intJdn3dn6uvrs =[vpa(sym('  -.97811352194183953055704604912')) vpa(sym(' -.27107545970547952350764790562l'));...
            vpa(sym(' -.27107545970547952350764790562l')) vpa(sym('  -.16559436051923997292216207590'))];
intJdn3dn7uvrs =[vpa(sym(' -.97451565507866331982099391909e-1')) vpa(sym('  -.25381039555863803386922475l9e-2'));...
            vpa(sym('   -.30253810395558638033869224753l9')) vpa(sym('    .463447558533213614696680089166'))];
intJdn3dn8uvrs =[vpa(sym(' .995438970603730531076247388l4e-1')) vpa(sym('  -.64522997029044688428674130426l7'));...
            vpa(sym(' .67270029709553115713258695733e-1')) vpa(sym(' -1.65344386982218686259075870810l'))];
intJdn3dn9uvrs =[vpa(sym(' -1.65344386982218686259075870810l')) vpa(sym(' .67270029709553115713258695733e-1'));...
            vpa(sym('  -.64522997029044688428674130426l7')) vpa(sym(' .995438970603730531076247388l4e-1'))];
intJdn3dn10uvrs =[vpa(sym('    .463447558533213614696680089166')) vpa(sym('   -.30253810395558638033869224753l9'));...
            vpa(sym('  -.25381039555863803386922475l9e-2')) vpa(sym(' -.97451565507866331982099391909e-1'))];
intJdn3dn11uvrs =[vpa(sym(' -.16559436051923997292216207589')) vpa(sym(' -.27107545970547952350764790562l'));...
            vpa(sym(' -.27107545970547952350764790562l')) vpa(sym('  -.97811352194183953055704604912'))];
intJdn3dn12uvrs =[vpa(sym('  .120923980597039893815264134l0')) vpa(sym(' .338079890317619378517715472657'));...
            vpa(sym(' .33807989031761937851771547265l')) vpa(sym('  .422209070544079358508960005l2'))];
intJdn4dn1uvrs =[vpa(sym('  .494129905754355174539743160l')) vpa(sym(' .254564190759598912923416448l0'));...
            vpa(sym(' .167064190759598912923416448l0')) vpa(sym(' .243923680580840207981142402l0'))];
intJdn4dn2uvrs =[vpa(sym('  .16985339992712677743504195136')) vpa(sym(' .179045055450002326957831462744'));...
            vpa(sym(' .17904505545000232695783146274l')) vpa(sym('  .16985339992712677743504195136'))];
intJdn4dn3uvrs =[vpa(sym('   .156606852059463586417315376026')) vpa(sym(' .335115927352323467949318271l4e-1'));...
            vpa(sym('   .12101159273523234679493182712l')) vpa(sym('   .5124614234253834624957312395l6'))];
intJdn4dn4uvrs =[vpa(sym('   .794715916478116799729308112l8')) vpa(sym('  -.15481341653816302516393935l6'));...
            vpa(sym('  -.15481341653816302516393935l6')) vpa(sym(' .927247441752334105622080795189'))];
intJdn4dn5uvrs =[vpa(sym('   -.66191053322154112847875205l37')) vpa(sym(' -.61610119267126767523886182l9e-2'));...
            vpa(sym(' -.61610119267126767523886182l9e-2')) vpa(sym('  -.96686494922443392864432222l9e-1'))];
intJdn4dn6uvrs =[vpa(sym(' .623647580488036249761255273l1e-1')) vpa(sym('  -.21219256804932529716922346537l'));...
            vpa(sym('  -.21219256804932529716922346537l')) vpa(sym(' -.78982200036984917036376362l3e-1'))];
intJdn4dn7uvrs =[vpa(sym('    -.10277602051817855644721686707')) vpa(sym(' -.32732059722698593284644960282l6'));...
            vpa(sym(' -.32732059722698593284644960282l6')) vpa(sym('    .144533345449155674413769328l7'))];
```

```
intJdn4dn8uvrs =[vpa(sym(' .364041669725565464086726766 8e-1')) vpa(sym(' .10228303532710463546 4061602393186'));...
        vpa(sym(' .10228303532710463546 4061602393186')) vpa(sym(' -.82013797757850663870 6811693089'))];
intJdn4dn9uvrs =[vpa(sym('  .364521033839059598529804469259')) vpa(sym('  .43277316533178068403 8772441541'));...
        vpa(sym('  .13277316533178068403 8772441541')) vpa(sym(' .66028785385703773347942390579e-1'))];
intJdn4dn10uvrs =[vpa(sym('  -1.32051059158026743632811545266')) vpa(sym('  -.74544201189536998617617117680'));...
        vpa(sym(' -.32942011895369986176171176 80e-1')) vpa(sym(' .89381759607913761072441622471e-1'))];
intJdn4dn11uvrs =[vpa(sym(' -.61017403759734250986707199 6e-1')) vpa(sym('  .38302940123838683131424805667'));...
        vpa(sym('  -.32947059876161316868575194333')) vpa(sym('  -1.89507344816323898310987069 86'))];
intJdn4dn12uvrs =[vpa(sym('  .67618516000239264204780296 6e-1')) vpa(sym(' .60723164794451180672811744 43e-1'));...
        vpa(sym('  .36072316479445118067281174443')) vpa(sym('  .73745028457271616934934124 6171'))];
intJdn5dn1uvrs =[vpa(sym(' -1.91709736541803286033938722609')) vpa(sym(' -1.21323891821327672848240784486'));...
        vpa(sym('  -.50073891821327672848240784486')) vpa(sym('  -.48951593925070344936468 5749'))];
intJdn5dn2uvrs =[vpa(sym('  .73745028457271616934934124 6171')) vpa(sym(' .60723164794451180672811744 43e-1'));...
        vpa(sym('  .36072316479445118067281174443')) vpa(sym('  .67618516000239264204780296 6e-1'))];
intJdn5dn3uvrs =[vpa(sym('  .42220907054407935850896000532')) vpa(sym(' .33807989031761937851771547 2657'));...
        vpa(sym(' .33807989031761937851771547 2657')) vpa(sym('  .12092398059703989381526413440'))];
intJdn5dn4uvrs =[vpa(sym('  -.66191053322154112847875205137')) vpa(sym(' -.61610119267126767523886182 29e-2'));...
        vpa(sym(' -.61610119267126767523886182 29e-2')) vpa(sym('  -.96686494922443392864432 2229e-1'))];
intJdn5dn5uvrs =[vpa(sym(' 4.20027429051479081211473647017')) vpa(sym('  1.34386791894778084520442477 51'));...
        vpa(sym('  1.34386791894778084520442477 51')) vpa(sym('  .83181450436469526399128 7102'))];
intJdn5dn6uvrs =[vpa(sym(' -2.99702494451010242383266710989')) vpa(sym('  -.13498719191670255487461602 10'));...
        vpa(sym('  -1.14748719191670255487461602 10')) vpa(sym('  -.26530618062606601420643444 3'))];
intJdn5dn7uvrs =[vpa(sym('  -.22968940869230447662270961540')) vpa(sym('  -.13926553666521939019713093531'));...
        vpa(sym('  -.13926553666521939019713093531')) vpa(sym(' -.32853529098637720977358297 79e-1'))];
intJdn5dn8uvrs =[vpa(sym(' -.36080256653866032265394083 51e-1')) vpa(sym('  -.15399966449131754839794301520'));...
        vpa(sym('  -.15399966449131754839794301520')) vpa(sym('  -.68297320390716358168351330 8e-1'))];
intJdn5dn9uvrs =[vpa(sym(' -1.54041301042459644452156154833')) vpa(sym('  -.81719220580616221994946695233'));...
        vpa(sym('  -.81719220580616221994946695233')) vpa(sym('  -.26581082595366911475356427 53'))];
intJdn5dn10uvrs =[vpa(sym(' 1.77949064062986161117643452513')) vpa(sym('  .46814346046933431699809291100'));...
        vpa(sym('  .46814346046933431699809291100')) vpa(sym(' -.22172877626635562865944848351'))];
intJdn5dn11uvrs =[vpa(sym(' .22488277344760679680561940 46e-1')) vpa(sym(' .93890517833399167630961583 71e-1'));...
        vpa(sym(' .93890517833399167630961583 71e-1')) vpa(sym('  .19953911023238252175250582200'))];
intJdn5dn12uvrs =[vpa(sym('  .22030295531423473523043744 67')) vpa(sym(' .16013957665680622962994689919'));...
        vpa(sym(' .16013957665680622962994689919')) vpa(sym('  .22030295531423473523043744 67'))];
intJdn6dn1uvrs =[vpa(sym('  .37765421933316862052540179 0571')) vpa(sym('  .22114271276808854773140878587'));...
        vpa(sym(' -.78857287231911452268591214 13e-1')) vpa(sym('  .47102462045975434469650882 3e-1'))];
intJdn6dn2uvrs =[vpa(sym(' -1.89507344816323898310987069 8571')) vpa(sym('  .38302940123838683131424805667'));...
        vpa(sym('  -.32947059876161316868575194333')) vpa(sym('  -.61017403759734250986707199 6e-1'))];
intJdn6dn3uvrs =[vpa(sym('  -.97811352194183953055704604912')) vpa(sym(' -.27107545970547952350764790 5621'));...
        vpa(sym(' -.27107545970547952350764790 5621')) vpa(sym('  -.16559436051923997292216207590'))];
intJdn6dn4uvrs =[vpa(sym(' .62364758048803624976125527371e-1')) vpa(sym('  -.21219256804932529716922346 5371'));...
        vpa(sym('  -.21219256804932529716922346 5371')) vpa(sym(' -.78982200036984917036376362 43e-1'))];
intJdn6dn5uvrs =[vpa(sym(' -2.99702494451010242383266710989')) vpa(sym('  -1.14748719191670255487461602 10'));...
        vpa(sym('  -.13498719191670255487461602 10')) vpa(sym('  -.26530618062606601420643444 3'))];
intJdn6dn6uvrs =[vpa(sym(' 4.54242395713891689901104912584')) vpa(sym('  .60928696985203111467282824 77'));...
        vpa(sym('  .60928696985203111467282824 77')) vpa(sym('  .44333161223422860291462161 9'))];
intJdn6dn7uvrs =[vpa(sym('  -.21838750833036302366600382197')) vpa(sym('  -.10146047716910257979584393430'));...
        vpa(sym('  -.10146047716910257979584393430')) vpa(sym(' .78281283861351532608066273 53e-1'))];
intJdn6dn8uvrs =[vpa(sym(' -.42271611042919621472681592 04e-1')) vpa(sym('  -.17005535924820519494502526850'));...
        vpa(sym('  -.17005535924820519494502526850')) vpa(sym(' .66296433804103064946378510343e-1'))];
intJdn6dn9uvrs =[vpa(sym(' 1.98585536527480847337212660 5957')) vpa(sym('  .17495708456313244013203951683'));...
        vpa(sym('  .17495708456313244013203951683')) vpa(sym('  -.29882200555297695918896368570'))];
intJdn6dn10uvrs =[vpa(sym(' -1.07121743309508046648931296 3500')) vpa(sym('  .29378408669763175300622339994'));...
        vpa(sym('  .29378408669763175300622339994')) vpa(sym('  .17797102414911889023099117 89'))];
intJdn6dn11uvrs =[vpa(sym(' .34251057055463909490373363 41e-1')) vpa(sym('  .12618028313614529580464700274'));...
        vpa(sym('  .12618028313614529580464700274')) vpa(sym(' .34251057055463909490373363 41e-1'))];
intJdn6dn12uvrs =[vpa(sym('  .19953911023238252175250582200')) vpa(sym(' .93890517833399167630961583 71e-1'));...
        vpa(sym(' .93890517833399167630961583 71e-1')) vpa(sym(' .22488277344760679680561940 46e-1'))];
intJdn7dn1uvrs =[vpa(sym('  -.31236802364420048787164339097')) vpa(sym(' -.62259192547029207059718397 2271'));...
        vpa(sym(' -.62259192547029207059718397 2271')) vpa(sym('  -.93264116705106716805029551279'))];
intJdn7dn2uvrs =[vpa(sym(' .89381759607913761072441622471e-1')) vpa(sym('  -.74544201189536998617617117680'));...
        vpa(sym(' -.32942011895369986176171176 80e-1')) vpa(sym('  -1.32051059158026743632811545266'))];
intJdn7dn3uvrs =[vpa(sym(' -.97451565507866331982099391909e-1')) vpa(sym('  -.30253810395558638033869224 7539'));...
        vpa(sym('  -.25381039555863803386922475 39e-2')) vpa(sym('  .46344755853321361469668008 9166'))];
intJdn7dn4uvrs =[vpa(sym('  -.10277602051817855644721686 71')) vpa(sym(' -.32732059722698593284644960 28286'));...
```

```
            vpa(sym(' -.32732059722698593284644960286')) vpa(sym('    .144533345449155674413769328 57'))];
intJdn7dn5uvrs =[vpa(sym('   -.22968940869230447662270961540')) vpa(sym('   -.139265536665219390197130935 31'));...
            vpa(sym('   -.13926553666521939019713093531')) vpa(sym(' -.32853529098637720977358297 79e-1'))];
intJdn7dn6uvrs =[vpa(sym('  -.21838750833036302366600382197')) vpa(sym('  -.10146047716910257979584393430'));...
            vpa(sym('  -.10146047716910257979584393430')) vpa(sym(' .78281283861351532608066273 53e-1'))];
intJdn7dn7uvrs =[vpa(sym('    .94300269776652031248409670 49')) vpa(sym(' 1.05449011376469883666280208 5186'));...
            vpa(sym(' 1.05449011376469883666280208 5186')) vpa(sym('  3.17595231695266811253589555038'))];
intJdn7dn8uvrs =[vpa(sym('   -.23411496564964084582129500914')) vpa(sym(' .22996512009548291644560497057e-1'));...
            vpa(sym('  -.98950348799045170835543950 2943')) vpa(sym('  -2.31569161290561082190356514273'))];
intJdn7dn9uvrs =[vpa(sym('    .10963550129394871861715141079')) vpa(sym('    .19369458821691485109794452082'));...
            vpa(sym('    .19369458821691485109794452082')) vpa(sym(' -.65316097486994600347934059 82e-1'))];
intJdn7dn10uvrs =[vpa(sym(' .96525285791407668666573566268e-1')) vpa(sym('   .20550989122442829054184845497'));...
            vpa(sym('   .20550989122442829054184845497')) vpa(sym(' .96525285791407668666573566268e-1'))];
intJdn7dn11uvrs =[vpa(sym('  .17797102414911889023099117890')) vpa(sym('  .29378408669763175300622339994'));...
            vpa(sym('  .29378408669763175300622339994')) vpa(sym(' -1.07121743309508046648931296 35'))];
intJdn7dn12uvrs =[vpa(sym(' -.22172877626635562865944848351')) vpa(sym('  .46814346046933431699809291100'));...
            vpa(sym('  .46814346046933431699809291100')) vpa(sym(' 1.77949064062986161117643452513'))];
 intJdn8dn1uvrs =[vpa(sym('  .22334568260811342445267275929')) vpa(sym(' .41545805218570060363476054995 7'));...
            vpa(sym(' .41545805218570060363476054995 7')) vpa(sym(' .32179538230835884627097507030 0'))];
intJdn8dn2uvrs =[vpa(sym(' .66028785385703773347942390579e-1')) vpa(sym('   .43277316533178068403877244154 1'));...
            vpa(sym('   .13277316533178068403877244154 1')) vpa(sym('   .36452103383905959852980446925 9'))];
intJdn8dn3uvrs =[vpa(sym(' .99543897060373053107624738884e-1')) vpa(sym(' .67270029709553115713258695733e-1'));...
            vpa(sym('  -.64522997029044688428674130426 7')) vpa(sym(' -1.65344386982218686259075870810 0'))];
intJdn8dn4uvrs =[vpa(sym(' .36404166972556546408672676 68e-1')) vpa(sym(' .10228303532710463540616023931 86'));...
            vpa(sym(' .10228303532710463540616023931 86')) vpa(sym(' -.82013797757850663870681169308 9'))];
intJdn8dn5uvrs =[vpa(sym(' -.36080256653866032265394083 51e-1')) vpa(sym('   -.15399966449131754839794301520'));...
            vpa(sym('   -.15399966449131754839794301520')) vpa(sym('  -.68297320390716358168351330 8e-1'))];
intJdn8dn6uvrs =[vpa(sym(' -.42271611042919621472681592 04e-1')) vpa(sym('   -.17005535924820519494502526850'));...
            vpa(sym('   -.17005535924820519494502526850')) vpa(sym(' .66296433804103064946378510343e-1'))];
intJdn8dn7uvrs =[vpa(sym('   -.23411496564964084582129500914')) vpa(sym('  -.98950348799045170835543950 2943'));...
            vpa(sym(' .22996512009548291644560497057e-1')) vpa(sym('  -2.31569161290561082190356514273'))];
intJdn8dn8uvrs =[vpa(sym('   .57399287653962671437054925866')) vpa(sym('  .51053643545524295661823792444 7'));...
            vpa(sym('  .51053643545524295661823792444 7')) vpa(sym(' 3.60677972082764476199224147524 4'))];
intJdn8dn9uvrs =[vpa(sym(' -.56899646226306337837629118 90e-1')) vpa(sym('    .23377832674670738500670085018'));...
            vpa(sym('    .23377832674670738500670085018')) vpa(sym(' -.56899646226306337837629118 90e-1'))];
intJdn8dn10uvrs =[vpa(sym(' -.65316097486994600347934059 82e-1')) vpa(sym('    .19369458821691485109794452082'));...
             vpa(sym('    .19369458821691485109794452082')) vpa(sym('    .10963550129394871861715141079'))];
intJdn8dn11uvrs =[vpa(sym('  -.29882200555297695918896368570')) vpa(sym('   .17495708456313244013203951683'));...
             vpa(sym('   .17495708456313244013203951683')) vpa(sym(' 1.98585536527480847337212660 5957'))];
intJdn8dn12uvrs =[vpa(sym('  -.26581082595366911475356427533')) vpa(sym('  -.81719220580616221994946695233'));...
             vpa(sym('  -.81719220580616221994946695233')) vpa(sym(' -1.54041301042459644452156154833'))];
 intJdn9dn1uvrs =[vpa(sym(' .32179538230835884627097507030 0')) vpa(sym(' .41545805218570060363476054995 7'));...
             vpa(sym(' .41545805218570060363476054995 7')) vpa(sym('  .22334568260811342445267275929'))];
intJdn9dn2uvrs =[vpa(sym(' -.82013797757850663870681169308 9')) vpa(sym(' .10228303532710463540616023931 86'));...
             vpa(sym(' .10228303532710463540616023931 86')) vpa(sym(' .36404166972556546408672676 68e-1'))];
intJdn9dn3uvrs =[vpa(sym(' -1.65344386982218686259075870810 0')) vpa(sym('  -.64522997029044688428674130426 7'));...
             vpa(sym(' .67270029709553115713258695733e-1')) vpa(sym(' .99543897060373053107624738884e-1'))];
intJdn9dn4uvrs =[vpa(sym('   .36452103383905959852980446925 9')) vpa(sym('   .13277316533178068403877244154 1'));...
             vpa(sym('   .43277316533178068403877244154 1')) vpa(sym(' .66028785385703773347942390579e-1'))];
intJdn9dn5uvrs =[vpa(sym(' -1.54041301042459644452156154833')) vpa(sym('  -.81719220580616221994946695233'));...
             vpa(sym('  -.81719220580616221994946695233')) vpa(sym('   -.26581082595366911475356427753'))];
intJdn9dn6uvrs =[vpa(sym(' 1.98585536527480847337212660 5957')) vpa(sym('   .17495708456313244013203951683'));...
             vpa(sym('   .17495708456313244013203951683')) vpa(sym('  -.29882200555297695918896368570'))];
intJdn9dn7uvrs =[vpa(sym('    .10963550129394871861715141079')) vpa(sym('    .19369458821691485109794452082'));...
             vpa(sym('    .19369458821691485109794452082')) vpa(sym(' -.65316097486994600347934059 82e-1'))];
intJdn9dn8uvrs =[vpa(sym(' -.56899646226306337837629118 90e-1')) vpa(sym('    .23377832674670738500670085018'));...
             vpa(sym('    .23377832674670738500670085018')) vpa(sym(' -.56899646226306337837629118 90e-1'))];
intJdn9dn9uvrs =[vpa(sym(' 3.60677972082764476199224147524 4')) vpa(sym('  .51053643545524295661823792444 7'));...
             vpa(sym('  .51053643545524295661823792444 7')) vpa(sym('   .57399287653962671437054925866'))];
intJdn9dn10uvrs =[vpa(sym(' -2.31569161290561082190356514272 7')) vpa(sym(' .22996512009548291644560497057e-1'));...
             vpa(sym('  -.98950348799045170835543950 2943')) vpa(sym('   -.23411496564964084582129500914'))];
intJdn9dn11uvrs =[vpa(sym(' .66296433804103064946378510343e-1')) vpa(sym('   -.17005535924820519494502526850'));...
             vpa(sym('   -.17005535924820519494502526850')) vpa(sym(' -.42271611042919621472681592 04e-1'))];
intJdn9dn12uvrs =[vpa(sym(' -.68297320390716358168351330 7714e-1')) vpa(sym('     -.15399966449131754839794301520'));...
             vpa(sym('     -.15399966449131754839794301520')) vpa(sym('   -.36080256653866032265394083 51e-1'))];
```

```
intJdn10dn1uvrs =[vpa(sym('  -.93264116705106716805029551279')) vpa(sym(' -.62259192547029207059718397227l'));...
        vpa(sym(' -.62259192547029207059718397227l')) vpa(sym('  -.31236802364420048787164339097'))];
intJdn10dn2uvrs =[vpa(sym('    .144533345449155674413769328571')) vpa(sym(' -.32732059722698593284644960282861'));...
        vpa(sym(' -.3273205972269859328464496028286')) vpa(sym('   -.10277602051817855644721686707'))];
intJdn10dn3uvrs =[vpa(sym('    .463447558533213614696680089166')) vpa(sym('  -.25381039555863803386922475391e-2'));...
        vpa(sym('   -.30253810395558638038692247539')) vpa(sym(' -.97451565507866331982099391909e-1'))];
intJdn10dn4uvrs =[vpa(sym('  -1.32051059158026743632811545266')) vpa(sym(' -.32942011895369986176171176801e-1'));...
        vpa(sym('   -.74544201189536998617617117680')) vpa(sym(' .893817596079137610724416224l1e-1'))];
intJdn10dn5uvrs =[vpa(sym(' 1.77949064062986161117643452513')) vpa(sym('  .468143460469334316998092911l0'));...
        vpa(sym('  .46814346046933431699809291100')) vpa(sym(' -.22172877626635562865944848351'))];
intJdn10dn6uvrs =[vpa(sym(' -1.07121743309508046648931296350l')) vpa(sym('    .2937840866976317530062233999l'));...
        vpa(sym('    .293784086697631753006223399941')) vpa(sym('     .17797102414911889023099117891'))];
intJdn10dn7uvrs =[vpa(sym(' .96525285791407668665735662681e-1')) vpa(sym('   .2055098912244282905418484549l'));...
        vpa(sym('   .20550989122442829054184845497')) vpa(sym(' .96525285791407668665735662681e-1'))];
intJdn10dn8uvrs =[vpa(sym(' -.65316097486994600347934059821e-1')) vpa(sym('    .1936945882169148510979445208l'));...
        vpa(sym('    .193694588216914851097944520821')) vpa(sym('    .10963550129394871861715141079'))];
intJdn10dn9uvrs =[vpa(sym(' -2.31569161290561082190356514272l')) vpa(sym('  -.9895034879904517080355439502943'));...
        vpa(sym(' .22996512009548291644560497057e-1')) vpa(sym('   -.23411496564964084582129500914'))];
intJdn10dn10uvrs =[vpa(sym('  3.17595231695266811253589555038')) vpa(sym(' 1.05449011376469883666280208518l'));...
         vpa(sym(' 1.0544901137646988366628020851l6')) vpa(sym('   .943002697766520312484096704l7'))];
intJdn10dn11uvrs =[vpa(sym(' .78281283861351532608066273l3e-1')) vpa(sym('  -.10146047716910257979584393430'));...
         vpa(sym('  -.10146047716910257979584393430')) vpa(sym('  -.21838750833036302366600382197'))];
intJdn10dn12uvrs =[vpa(sym(' -.32853529098637720977357298l9e-1')) vpa(sym('   -.13926553666521939019713093531'));...
         vpa(sym('   -.13926553666521939019713093531')) vpa(sym('   -.22968940869230447662270961540'))];
intJdn11dn1uvrs =[vpa(sym('   .47102462045975434469650882l3e-1')) vpa(sym(' -.78857287231911452268591214l3e-1'));...
        vpa(sym('    .22114271276808854773140878587')) vpa(sym('   .37765421933316862052540179057l'))];
intJdn11dn2uvrs =[vpa(sym(' -.78982200036984917036376362l3e-1')) vpa(sym('  -.21219256804932529716922346537l'));...
        vpa(sym('  -.21219256804932529716922346537l')) vpa(sym(' .623647580488036249761255273l1e-1'))];
intJdn11dn3uvrs =[vpa(sym(' -.16559436051923997292216207589l')) vpa(sym(' -.27107545970547952350764790562l'));...
        vpa(sym(' -.27107545970547952350764790562l')) vpa(sym('  -.97811352194183953055046049l2'))];
intJdn11dn4uvrs =[vpa(sym(' -.61017403759734250986707199l6e-1')) vpa(sym('  -.32947059876161316868575194333'));...
        vpa(sym('   .38302940123838683131424805667')) vpa(sym('  -1.89507344816323898310987069l6'))];
intJdn11dn5uvrs =[vpa(sym(' .22488277344760679680561940l6e-1')) vpa(sym(' .93890517833399167630961583l1e-1'));...
        vpa(sym(' .93890517833399167630961583l1e-1')) vpa(sym('   .19953911023238252175250582200'))];
intJdn11dn6uvrs =[vpa(sym(' .34251057055463909490373363l1e-1')) vpa(sym('   .12618028313614529580464700274'));...
        vpa(sym('   .12618028313614529580464700274')) vpa(sym(' .34251057055463909490373363l1e-1'))];
intJdn11dn7uvrs =[vpa(sym('  .17797102414911889023099117890')) vpa(sym('  .293784086697631753006223399941'));...
        vpa(sym('  .29378408669763175300622339994')) vpa(sym(' -1.07121743309508046648931296350'))];
intJdn11dn8uvrs =[vpa(sym('  -.29882200555297695918896368570')) vpa(sym('   .17495708456313244013203951683'));...
        vpa(sym('   .17495708456313244013203951683')) vpa(sym(' 1.98585536527480847337212660595l'))];
intJdn11dn9uvrs =[vpa(sym(' .66296433804103064946378510343e-1')) vpa(sym('   -.17005535924820519494502526850'));...
        vpa(sym('   -.17005535924820519494502526850')) vpa(sym(' -.42271611042919621472681592l4e-1'))];
intJdn11dn10uvrs =[vpa(sym(' .78281283861351532608066273l3e-1')) vpa(sym('  -.10146047716910257979584393430'));...
         vpa(sym('  -.10146047716910257979584393430')) vpa(sym('  -.21838750833036302366600382197'))];
intJdn11dn11uvrs =[vpa(sym('    .44333161223422860291462161l9')) vpa(sym('   .60928696985203111467282824l7'));...
         vpa(sym('   .60928696985203111467282824l7')) vpa(sym(' 4.54242395713891689901104912584'))];
intJdn11dn12uvrs =[vpa(sym('    -.26530618062606601420643444l3')) vpa(sym('   -.13498719191670255487461602l0'));...
         vpa(sym('  -1.14748719191670255487461602l0')) vpa(sym(' -2.99702494451010242383266710989'))];
 intJdn12dn1uvrs =[vpa(sym('    -.48951593925070344936468574l9')) vpa(sym('  -.50073891821327672848240784486'));...
        vpa(sym(' -1.21323891821327672848240784486')) vpa(sym(' -1.91709736541803286033938722609'))];
intJdn12dn2uvrs =[vpa(sym('  -.96686494922443392864432222l9e-1')) vpa(sym(' -.61610119267126767523886182l9e-2'));...
        vpa(sym(' -.61610119267126767523886182l9e-2')) vpa(sym('   -.66191053322154112847875205137'))];
intJdn12dn3uvrs =[vpa(sym('  .12092398059703989381526413440')) vpa(sym(' .33807989031761937851771547265l'));...
        vpa(sym(' .33807989031761937851771547265l')) vpa(sym('  .42220907054407935850896000532'))];
intJdn12dn4uvrs =[vpa(sym('  .67618516000239264204780296l6e-1')) vpa(sym('   .36072316479445118067281174443'));...
        vpa(sym(' .60723164794451180672811744l3e-1')) vpa(sym('  .73745028457271616934934124617l'))];
intJdn12dn5uvrs =[vpa(sym(' .22030295531423473523043744l7')) vpa(sym(' .16013957665680622962994689l2'));...
        vpa(sym(' .16013957665680622962994689l2')) vpa(sym(' .22030295531423473523043744l7'))];
intJdn12dn6uvrs =[vpa(sym('   .19953911023238252175250582200')) vpa(sym(' .93890517833399167630961583l1e-1'));...
        vpa(sym(' .93890517833399167630961583l1e-1')) vpa(sym(' .22488277344760679680561940l6e-1'))];
intJdn12dn7uvrs =[vpa(sym(' -.22172877626635562865944848351')) vpa(sym('  .46814346046933431699809291100'));...
        vpa(sym('  .46814346046933431699809291100')) vpa(sym(' 1.77949064062986161117643452513'))];
intJdn12dn8uvrs =[vpa(sym('  -.26581082595366911475356427533')) vpa(sym('  -.81719220580616221994946695233'));...
        vpa(sym('  -.81719220580616221994946695233')) vpa(sym(' -1.54041301042459644452156154833'))];
intJdn12dn9uvrs =[vpa(sym(' -.68297320390716358168351330776l4e-1')) vpa(sym('     -.153999664491317548397943015l0'));...
```

```
            vpa(sym('    -.153999664491317548397943015 20')) vpa(sym('   -.360802566538660322653940835 1e-1'))];
intJdn12dn10uvrs =[vpa(sym(' -.3285352909863772097735829779e-1')) vpa(sym('   -.139265536665219390197130935 31'));...
            vpa(sym('   -.139265536665219390197130935 31')) vpa(sym('   -.229689408692304476622709615 40'))];
intJdn12dn11uvrs =[vpa(sym('    -.265306180626066014206434443')) vpa(sym('  -1.14748719191670255487461602 10'));...
            vpa(sym('   -.134987191916702554874616021 0')) vpa(sym(' -2.997024944510102423832667109 89'))];
intJdn12dn12uvrs =[vpa(sym('    .831814504364695263991287102')) vpa(sym('  1.343867918947780845204424775 1'));...
            vpa(sym('  1.343867918947780845204424775 1')) vpa(sym(' 4.200274290514790812114736470 17'))];
%disp('---------------------------- integrals of products of global derivatives----------------------------------------------')

intJdndn=[intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs intJdn1dn5uvrs intJdn1dn6uvrs intJdn1dn7uvrs
intJdn1dn8uvrs intJdn1dn9uvrs intJdn1dn10uvrs intJdn1dn11uvrs intJdn1dn12uvrs;...
    intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs intJdn2dn5uvrs intJdn2dn6uvrs intJdn2dn7uvrs
intJdn2dn8uvrs intJdn2dn9uvrs intJdn2dn10uvrs intJdn2dn11uvrs intJdn2dn12uvrs;...
    intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs intJdn3dn5uvrs intJdn3dn6uvrs intJdn3dn7uvrs
intJdn3dn8uvrs intJdn3dn9uvrs intJdn3dn10uvrs intJdn3dn11uvrs intJdn3dn12uvrs;...
    intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs intJdn4dn5uvrs intJdn4dn6uvrs intJdn4dn7uvrs
intJdn4dn8uvrs intJdn4dn9uvrs intJdn4dn10uvrs intJdn4dn11uvrs intJdn4dn12uvrs;...
    intJdn5dn1uvrs intJdn5dn2uvrs intJdn5dn3uvrs intJdn5dn4uvrs intJdn5dn5uvrs intJdn5dn6uvrs intJdn5dn7uvrs
intJdn5dn8uvrs intJdn5dn9uvrs intJdn5dn10uvrs intJdn5dn11uvrs intJdn5dn12uvrs;...
    intJdn6dn1uvrs intJdn6dn2uvrs intJdn6dn3uvrs intJdn6dn4uvrs intJdn6dn5uvrs intJdn6dn6uvrs intJdn6dn7uvrs
intJdn6dn8uvrs intJdn6dn9uvrs intJdn6dn10uvrs intJdn6dn11uvrs intJdn6dn12uvrs;...
    intJdn7dn1uvrs intJdn7dn2uvrs intJdn7dn3uvrs intJdn7dn4uvrs intJdn7dn5uvrs intJdn7dn6uvrs intJdn7dn7uvrs
intJdn7dn8uvrs intJdn7dn9uvrs intJdn7dn10uvrs intJdn7dn11uvrs intJdn7dn12uvrs;...
    intJdn8dn1uvrs intJdn8dn2uvrs intJdn8dn3uvrs intJdn8dn4uvrs intJdn8dn5uvrs intJdn8dn6uvrs intJdn8dn7uvrs
intJdn8dn8uvrs intJdn8dn9uvrs intJdn8dn10uvrs intJdn8dn11uvrs intJdn8dn12uvrs;...
    intJdn9dn1uvrs intJdn9dn2uvrs intJdn9dn3uvrs intJdn9dn4uvrs intJdn9dn5uvrs intJdn9dn6uvrs intJdn9dn7uvrs
intJdn9dn8uvrs intJdn9dn9uvrs intJdn9dn10uvrs intJdn9dn11uvrs intJdn9dn12uvrs;...
    intJdn10dn1uvrs intJdn10dn2uvrs intJdn10dn3uvrs intJdn10dn4uvrs intJdn10dn5uvrs intJdn10dn6uvrs intJdn10dn7uvrs
intJdn10dn8uvrs intJdn10dn9uvrs intJdn10dn10uvrs intJdn10dn11uvrs intJdn10dn12uvrs;...
    intJdn11dn1uvrs intJdn11dn2uvrs intJdn11dn3uvrs intJdn11dn4uvrs intJdn11dn5uvrs intJdn11dn6uvrs intJdn11dn7uvrs
intJdn11dn8uvrs intJdn11dn9uvrs intJdn11dn10uvrs intJdn11dn11uvrs intJdn11dn12uvrs;...
    intJdn12dn1uvrs intJdn12dn2uvrs intJdn12dn3uvrs intJdn12dn4uvrs intJdn12dn5uvrs intJdn12dn6uvrs intJdn12dn7uvrs
intJdn12dn8uvrs intJdn12dn9uvrs intJdn12dn10uvrs intJdn12dn11uvrs intJdn12dn12uvrs];
intJdndn=double(intJdndn);
  case 16
intJdn1dn1uvrs =[vpa(sym(' .7016810931454547208703800 87')) vpa(sym(' .6758932206311068322413583 44'));...
vpa(sym(' .6758932206311068322413583 44')) vpa(sym(' .7016810931454547208703800 87'))];
intJdn1dn2uvrs =[vpa(sym(' -.2688228646294743129188271 25e-1')) vpa(sym('  .5105444795945812551149174 45e-1'));...
vpa(sym(' -.3644555204054187448850825 55e-1')) vpa(sym('  .3221221180410707513358372 7e-1'))];
intJdn1dn3uvrs =[vpa(sym(' -.8630327499228376151969184 35e-2')) vpa(sym(' -.1049617614028559256240777 39e-1'));...
vpa(sym(' -.1049617614028559256240777 39e-1')) vpa(sym(' -.8630327499228376151969184 35e-2'))];
intJdn1dn4uvrs =[vpa(sym('  .3221221180410707513358372 7e-1')) vpa(sym(' -.3644555204054187448850825 55e-1'));...
vpa(sym('  .5105444795945812551149174 5e-1')) vpa(sym(' -.2688228646294743129188271 25e-1'))];
intJdn1dn5uvrs =[vpa(sym(' -.4126330541176002307162068 98')) vpa(sym('  .2689455259847769815430225 33'));...
vpa(sym(' -.4435544740152230184569774 67')) vpa(sym('  .1578716245413387047945578 29'))];
intJdn1dn6uvrs =[vpa(sym('  .1172503945374017479674604 967')) vpa(sym('  -.1425920794764966703892546 82'));...
vpa(sym('   .1574079205235033296107453 18')) vpa(sym(' -.7002708952501684636496711 3e-1'))];
intJdn1dn7uvrs =[vpa(sym(' -.9314467940272043066056656 05e-1')) vpa(sym('  -.1049111993881841007005975 075'));...
vpa(sym('  -.1049111993881841007005975 075')) vpa(sym('  -.9224518840851986203000470 4e-1'))];
intJdn1dn8uvrs =[vpa(sym('  .3351827978973348544686950 5e-1')) vpa(sym('  .3928115348643125549691051 6e-1'));...
vpa(sym('  .3928115348643125549691051 6e-1')) vpa(sym(' .3187907114531758725070089 5e-1'))];
intJdn1dn9uvrs =[vpa(sym(' .3187907114531758725070089 5e-1')) vpa(sym(' .3928115348643125549691051 55e-1'));...
vpa(sym(' .3928115348643125549691051 55e-1')) vpa(sym(' .3351827978973348544686950 45e-1'))];
intJdn1dn10uvrs =[vpa(sym(' -.9224518840851986203000470 35e-1')) vpa(sym('  -.1049111993881841007005975 075'));...
vpa(sym('  -.1049111993881841007005975 075')) vpa(sym(' -.9314467940272043066056656 05e-1'))];
intJdn1dn11uvrs =[vpa(sym(' -.7002708952501684636496711 3e-1')) vpa(sym('   .1574079205235033296107453 18'));...
vpa(sym('  -.1425920794764966703892546 82')) vpa(sym('  .1172503945374017479674604 967'))];
intJdn1dn12uvrs =[vpa(sym('  .1578716245413387047945578 29')) vpa(sym(' -.4435544740152230184569774 67'));...
vpa(sym('  .2689455259847769815430225 33')) vpa(sym(' -.4126330541176002307162068 98'))];
intJdn1dn13uvrs =[vpa(sym(' -.9236778391018313055506771 02')) vpa(sym(' -1.0070324548905004630756423 0'));...
vpa(sym(' -1.0070324548905004630756423 0')) vpa(sym(' -.9236778391018313055506771 02'))];
intJdn1dn14uvrs =[vpa(sym(' .3368519902227228640941725 19')) vpa(sym(' .3816714043523917177160500 03'));...
vpa(sym(' .3816714043523917177160500 03')) vpa(sym(' .3384590115434723589069085 99'))];
intJdn1dn15uvrs =[vpa(sym(' -.1224832122116840616983593 45')) vpa(sym(' -.1452630954370753949585534 95'));...
vpa(sym(' -.1452630954370753949585534 95')) vpa(sym(' -.1224832122116840616983593 45'))];
```

```
intJdn1dn16uvrs =[vpa(sym(' .338459011543472358906908599')) vpa(sym(' .381671404352391717716050003'));...
vpa(sym(' .381671404352391717716050003')) vpa(sym(' .336851990222722864094172 52'))];
intJdn2dn1uvrs =[vpa(sym(' -.26882286462947431291882 7125e-1')) vpa(sym(' -.364455520405418744885082555e-1'));...
vpa(sym(' .5105444795945812551149174 45e-1')) vpa(sym(' .32212211804107075133583727e-1'))];
intJdn2dn2uvrs =[vpa(sym(' .211971597991981069039609973')) vpa(sym(' -.153046574826884110662338533 64'));...
vpa(sym(' -.153046574826884110662338533 64')) vpa(sym(' .194935467116933386760696773 78'))];
intJdn2dn3uvrs =[vpa(sym(' .502810434966178037808492340e-1')) vpa(sym(' .415483012138861376819666 7829e-1'));...
vpa(sym(' -.459516987861138623180333217 1e-1')) vpa(sym(' -.210666908330060135638656 4082e-1'))];
intJdn2dn4uvrs =[vpa(sym(' -.459867371696942288992267 5e-3')) vpa(sym(' .693277469284456972956465 22e-2'));...
vpa(sym(' .693277469284456972956465 22e-2')) vpa(sym(' -.459867371696942288992267 5e-3'))];
intJdn2dn5uvrs =[vpa(sym(' .135316485729802334623950907')) vpa(sym(' .149351885457747745233522630'));...
vpa(sym(' -.150648114542252254766477 3701')) vpa(sym(' -.356070031451497704872952070e-1'))];
intJdn2dn6uvrs =[vpa(sym(' -.484060336356140646761807119')) vpa(sym(' -.322540791632477410079167 0851'));...
vpa(sym(' .389959208367522589920832 9149')) vpa(sym(' .137829701836470761484853 4907'))];
intJdn2dn7uvrs =[vpa(sym(' .283468713180315926702327990')) vpa(sym(' .261652146923052718725140 2015'));...
vpa(sym(' -.450847853076947281274859 798')) vpa(sym(' -.264647289956764019523548 1100'))];
intJdn2dn8uvrs =[vpa(sym(' -.114137962266763606747720 48')) vpa(sym(' -.129354487376281056516272933'));...
vpa(sym(' .170645512623718943483727067 04')) vpa(sym(' .803271661318366652125848 6710e-1'))];
intJdn2dn9uvrs =[vpa(sym(' -.353280334795837240195816 55e-1')) vpa(sym(' .575010022407089943211709137e-1'));...
vpa(sym(' .575010022407089943211709137e-1')) vpa(sym(' -.107656396311072669181942 1090e-1'))];
intJdn2dn10uvrs =[vpa(sym(' .510040381792347092571950 60e-2')) vpa(sym(' -.247617179998775437696714569e-1'));...
vpa(sym(' -.247617179998775437696714569e-1')) vpa(sym(' .327370706036359563493129e-3'))];
intJdn2dn11uvrs =[vpa(sym(' -.804949887236947900412300 0e-2')) vpa(sym(' -.239609914551285358186348361e-1'));...
vpa(sym(' -.239609914551285358186348361e-1')) vpa(sym(' .235304579485836526323508 27e-2'))];
intJdn2dn12uvrs =[vpa(sym(' .220346626285956934003016986e-1')) vpa(sym(' .495487663266407134761033842e-1'));...
vpa(sym(' .495487663266407134761033842e-1')) vpa(sym(' -.202705507058571468134520570e-1'))];
intJdn2dn13uvrs =[vpa(sym(' -.624603048181793546679316570e-1')) vpa(sym(' -.196130238975238548948111488'));...
vpa(sym(' -.196130238975238548948111488')) vpa(sym(' .180664477651688964563218 24e-1'))];
intJdn2dn14uvrs =[vpa(sym(' -.565454471791414690900725 95e-2')) vpa(sym(' .427622370384584419718474 4941'));...
vpa(sym(' .427622370384584419718474 4941')) vpa(sym(' -.153931015713211339133137 8486'))];
intJdn2dn15uvrs =[vpa(sym(' .803323805596981177025893 0e-2')) vpa(sym(' -.194216973763714354534537 5175'));...
vpa(sym(' -.194216973763714354534537 5175')) vpa(sym(' .416813143674259981116969119e-1'))];
intJdn2dn16uvrs =[vpa(sym(' .208266894443892215290932469e-1')) vpa(sym(' .863000808306781359312991 52e-1'));...
vpa(sym(' .863000808306781359312991 52e-1')) vpa(sym(' -.984668166045009257980461 6e-3'))];
intJdn3dn1uvrs =[vpa(sym(' -.863032749922837615196918435e-2')) vpa(sym(' -.104961761402855925624077739e-1'));...
vpa(sym(' -.104961761402855925624077739e-1')) vpa(sym(' -.863032749922837615196918435e-2'))];
intJdn3dn2uvrs =[vpa(sym(' .502810434966178037808492340e-1')) vpa(sym(' -.459516987861138623180333217 1e-1'));...
vpa(sym(' .415483012138861376819666 7829e-1')) vpa(sym(' -.210666908330060135638656 4082e-1'))];
intJdn3dn3uvrs =[vpa(sym(' .293456458248516177255568 1491')) vpa(sym(' .272605395682204969404152016 29'));...
vpa(sym(' .272605395682204969404152016 29')) vpa(sym(' .293456458248516177255568 1491'))];
intJdn3dn4uvrs =[vpa(sym(' -.210666908330060135638656 4082e-1')) vpa(sym(' .415483012138861376819666 7829e-1'));...
vpa(sym(' -.459516987861138623180333217 1e-1')) vpa(sym(' .502810434966178037808492340e-1'))];
intJdn3dn5uvrs =[vpa(sym(' .312258974588804915879185904e-1')) vpa(sym(' .306625487404711706979545881e-1'));...
vpa(sym(' .306625487404711706979545881e-1')) vpa(sym(' .120306246990647939530980 4256e-1'))];

intJdn3dn6uvrs =[vpa(sym(' -.956315330022406880975435 01e-1')) vpa(sym(' -.701114223525693099130736200e-1'));...
vpa(sym(' -.701114223525693099130736200e-1')) vpa(sym(' -.229019485329201510587083 4477e-1'))];
intJdn3dn7uvrs =[vpa(sym(' -.668302802576590727492936 39e-1')) vpa(sym(' .157146408891911765389874178 17'));...
vpa(sym(' -.142853591108088234610125821 83')) vpa(sym(' .958310725915156595907307062e-1'))];
intJdn3dn8uvrs =[vpa(sym(' .217119766902955166405682785')) vpa(sym(' -.364325619471825951559514188 39'));...
vpa(sym(' .348174380528174048440485811 61')) vpa(sym(' -.352542724916680732832984020'))];
intJdn3dn9uvrs =[vpa(sym(' -.352542724916680732832984020')) vpa(sym(' .348174380528174048440485811 61'));...
vpa(sym(' -.364325619471825951559514188 4')) vpa(sym(' .217119766902955166405682785'))];
intJdn3dn10uvrs =[vpa(sym(' .958310725915156595907307062e-1')) vpa(sym(' -.142853591108088234610125821 83'));...
vpa(sym(' .157146408891911765389874178 17')) vpa(sym(' -.668302802576590727492936 39e-1'))];
intJdn3dn11uvrs =[vpa(sym(' -.229019485329201510587083 4477e-1')) vpa(sym(' -.701114223525693099130736200e-1'));...
vpa(sym(' -.701114223525693099130736200e-1')) vpa(sym(' -.956315330022406880975435 01e-1'))];
intJdn3dn12uvrs =[vpa(sym(' .120306246990647939530980 4256e-1')) vpa(sym(' .306625487404711706979545881e-1'));...
vpa(sym(' .306625487404711706979545881e-1')) vpa(sym(' .312258974588804915879185904e-1'))];
intJdn3dn13uvrs =[vpa(sym(' -.440107128429126521075770644e-1')) vpa(sym(' -.964159752956607711899142376e-1'));...
vpa(sym(' -.964159752956607711899142376e-1')) vpa(sym(' -.440107128429126521075770644e-1'))];
intJdn3dn14uvrs =[vpa(sym(' .136999462878579334021389201')) vpa(sym(' .227213413649009719816657263 6'));...
vpa(sym(' .227213413649009719816657263 6')) vpa(sym(' .941185558099118441713021744e-1'))];
intJdn3dn15uvrs =[vpa(sym(' -.319448664201393584204597479')) vpa(sym(' -.534960505588025669879559805 17'));...
vpa(sym(' -.534960505588025669879559805 17')) vpa(sym(' -.319448664201393584204597479'))];
```

intJdn3dn16uvrs =[vpa(sym(' .941185558099118441713021744e-1')) vpa(sym(' .227213413649009719816657263 6')));...
vpa(sym(' .227213413649009719816657263 6')) vpa(sym(' .136999462878579334021389201'))];
intJdn4dn1uvrs =[vpa(sym(' .32212211804107075133583727e-1')) vpa(sym(' .5105444795945812551149174 5e-1'));...
vpa(sym(' -.364455520405418744885082555e-1')) vpa(sym(' -.26882286462947431291882712 5e-1'))];
intJdn4dn2uvrs =[vpa(sym(' -.45986737169694228899226 75e-3')) vpa(sym(' .693277469284456972956465 22e-2'));...
vpa(sym(' .693277469284456972956465 22e-2')) vpa(sym(' -.45986737169694228899226 75e-3'))];
intJdn4dn3uvrs =[vpa(sym(' -.210666908330060135638656 4082e-1')) vpa(sym(' -.459516987861138623180333217 1e-1'));...
vpa(sym(' .415483012138861376819666782 9e-1')) vpa(sym(' .502810434966178037808492340e-1'))];
intJdn4dn4uvrs =[vpa(sym(' .19493546711693338676069677378')) vpa(sym(' -.153046574826884110662338533 64'));...
vpa(sym(' -.153046574826884110662338533 64')) vpa(sym(' .211971597991981069039609973'))];
intJdn4dn5uvrs =[vpa(sym(' -.20270550705857146813452057 0e-1')) vpa(sym(' .4954876632664071347610338 42e-1'));...
vpa(sym(' .4954876632664071347610338 42e-1')) vpa(sym(' .22034662628595693400301699e-1'))];
intJdn4dn6uvrs =[vpa(sym(' .23530457948583652632350 83e-2')) vpa(sym(' -.2396099145512853581863483 6e-1'));...
vpa(sym(' -.2396099145512853581863483 6e-1')) vpa(sym(' -.80494988723694790041230002e-2'))];
intJdn4dn7uvrs =[vpa(sym(' .32737070603635956349312896 1e-3')) vpa(sym(' -.2476171799987754376967145 69e-1'));...
vpa(sym(' -.2476171799987754376967145 69e-1')) vpa(sym(' .510040381792347092571950 60e-2'))];
intJdn4dn8uvrs =[vpa(sym(' -.1076563963110726691819421090e-1')) vpa(sym(' .5750100224070899432117091 37e-1'));...
vpa(sym(' .5750100224070899432117091 37e-1')) vpa(sym(' -.3532803347958372401958165 5e-1'))];
intJdn4dn9uvrs =[vpa(sym(' .80327166131836665212584867 10e-1')) vpa(sym(' .1706455126237189434837270 6704'));...
vpa(sym(' -.1293544873762810565162729 3296')) vpa(sym(' -.1141379622667636067477204 80'))];
intJdn4dn10uvrs =[vpa(sym(' -.26464728995676401952354811 00')) vpa(sym(' -.4508478530769472812748597985'));...
vpa(sym(' .26165214692305271872514020 15')) vpa(sym(' .28346871318031592670232799 0'))];
intJdn4dn11uvrs =[vpa(sym(' .13782970183647076148485349 07')) vpa(sym(' .3899592083675225899208329149'));...
vpa(sym(' -.3225407916324774100791670851')) vpa(sym(' -.4840603363561406467618071 19'))];
intJdn4dn12uvrs =[vpa(sym(' -.3560700314514977048729520 70e-1')) vpa(sym(' -.1506481145422522547664773701'));...
vpa(sym(' .14935188545774774523352262 99')) vpa(sym(' .1353164857298023346239509070'))];
intJdn4dn13uvrs =[vpa(sym(' .18066447765168896456321824e-1')) vpa(sym(' -.1961302389752385489481114 88'));...
vpa(sym(' -.196130238975238548948111488')) vpa(sym(' -.6246030481817935466793166e-1'))];
intJdn4dn14uvrs =[vpa(sym(' -.9846681660450092579804616e-3')) vpa(sym(' .8630008083067813593129915 2e-1'));...
vpa(sym(' .863000808306781359312991 52e-1')) vpa(sym(' .2082668944438922152909324 69e-1'))];
intJdn4dn15uvrs =[vpa(sym(' .41681314367425998111696911 9e-1')) vpa(sym(' -.1942169737637143545345375175'));...
vpa(sym(' -.194216973763714354534537 5175')) vpa(sym(' .8033238055969811770258930e-2'))];
intJdn4dn16uvrs =[vpa(sym(' -.1539310157132113391331378486')) vpa(sym(' .4276223703845844197184744941'));...
vpa(sym(' .427622370384584419718474 4941')) vpa(sym(' -.5654544717914146990072595e-2'))];
intJdn5dn1uvrs =[vpa(sym(' -.4126330541176002307162068 98')) vpa(sym(' -.4435544740152230184569774 67'));...
vpa(sym(' .2689455259847769815430225 33')) vpa(sym(' .1578716245413387047945578 29'))];
intJdn5dn2uvrs =[vpa(sym(' .1353164857298023346239509 07')) vpa(sym(' -.1506481145422522547664773701'));...
vpa(sym(' .149351885457747745233522630')) vpa(sym(' -.3560700314514977048729520 70e-1'))];
intJdn5dn3uvrs =[vpa(sym(' .31225897458880491587918590 4e-1')) vpa(sym(' .3066254874047117069795458 81e-1'));...
vpa(sym(' .306625487404711706979545881e-1')) vpa(sym(' .1203062469906479395309804256e-1'))];
intJdn5dn4uvrs =[vpa(sym(' -.20270550705857146813452057 0e-1')) vpa(sym(' .4954876632664071347610338 42e-1'));...
vpa(sym(' .4954876632664071347610338 42e-1')) vpa(sym(' .22034662628595693400301699e-1'))];
intJdn5dn5uvrs =[vpa(sym(' 1.3175867872873127326690 8314')) vpa(sym(' .9028759894415023626830403 91'));...
vpa(sym(' .9028759894415023626830403 91')) vpa(sym(' 1.4953082485408172223216136 7'))];
intJdn5dn6uvrs =[vpa(sym(' -.7866868084644167345983151 3779')) vpa(sym(' .2747744905607333406360220 5'));...
vpa(sym(' -.7377255094392666593639779 54')) vpa(sym(' -.2220264808136213599324078 11'))];
intJdn5dn7uvrs =[vpa(sym(' .3065397294444835089245124969')) vpa(sym(' .2726059797662254122356749 87'));...
vpa(sym(' .2726059797662254122356749 87')) vpa(sym(' .1180108054671762141449177 34'))];
intJdn5dn8uvrs =[vpa(sym(' -.1123602772538571729615200020')) vpa(sym(' -.1085783194105248930324052264'));...
vpa(sym(' -.1085783194105248930324052264')) vpa(sym(' -.4240317317256157255425408 41e-1'))];
intJdn5dn9uvrs =[vpa(sym(' -.1349507714997270711398676150')) vpa(sym(' -.1134650610452099677162243619'));...
vpa(sym(' -.1134650610452099677162243619')) vpa(sym(' -.2836748587428850119955522 38e-1'))];
intJdn5dn10uvrs =[vpa(sym(' .10667911259258557145532230 09')) vpa(sym(' .6059337192477808811068095e-2'));...
vpa(sym(' .6059337192477808811068095e-2')) vpa(sym(' -.1306849801916743926696037294'))];
intJdn5dn11uvrs =[vpa(sym(' -.2697804586988087137951307 8e-1')) vpa(sym(' -.1969663903410909289096940 84'));...
vpa(sym(' -.196966390341090928909694084')) vpa(sym(' -.7889735824975260866827398 1e-1'))];
intJdn5dn12uvrs =[vpa(sym(' .89211654581638106968766 96e-1')) vpa(sym(' .4271888847990780701739923 86'));...
vpa(sym(' .427188884799078070173992386')) vpa(sym(' .89211654581638106968766 96e-1'))];
intJdn5dn13uvrs =[vpa(sym(' .47555890783193200318874647 8')) vpa(sym(' -.5308433559870513262049879 2'));...
vpa(sym(' -.530843355987051326204987 92')) vpa(sym(' -1.8311795625089326607228139 8'))];

intJdn5dn14uvrs =[vpa(sym(' -1.19557360195760315006531368')) vpa(sym(' -.8902896395896492199965462 17'));...
vpa(sym(' -.890289639589649219996546217')) vpa(sym(' -.1462376799023816683441454 80'))];
intJdn5dn15uvrs =[vpa(sym(' .44987769212666366162955235 0')) vpa(sym(' .3793607470719477419602320 42'));...
vpa(sym(' .379360747071947741960232042')) vpa(sym(' .8690454777248010390310669 2e-1'))];

```
intJdn5dn16uvrs =[vpa(sym(' -.22254315718435603333736647188')) vpa(sym(' .91268611031924988409224700e-1'));...
vpa(sym(' .91268611031924988409224700e-1')) vpa(sym(' .53403155562725169509198688 7'))];
intJdn6dn1uvrs =[vpa(sym(' .11725039453740174796746049 67')) vpa(sym(' .15740792052350332961074531 8'));...
vpa(sym(' -.14259207947649667038925468 2')) vpa(sym(' -.70027089525016846364967113e-1'))];
intJdn6dn2uvrs =[vpa(sym(' -.48406033635614064676180711 9')) vpa(sym(' .38995920836752258992083291 49'));...
vpa(sym(' -.32254079163247741007916708 51')) vpa(sym(' .13782970183647076148485349 07'))];
intJdn6dn3uvrs =[vpa(sym(' -.95631533002240688097543501e-1')) vpa(sym(' -.70111422352569309913073620 0e-1'));...
vpa(sym(' -.70111422352569309913073620 0e-1')) vpa(sym(' -.22901948532920151058708344 77e-1'))];
intJdn6dn4uvrs =[vpa(sym(' .23530457948583652632350 83e-2')) vpa(sym(' -.23960991455128535818634836 1e-1'));...
vpa(sym(' -.23960991455128535818634836 1e-1')) vpa(sym(' -.80494988723694790041230002e-2'))];
intJdn6dn5uvrs =[vpa(sym(' -.78668680846441673459831513779')) vpa(sym(' -.73772550943926665936397795 4'));...
vpa(sym(' .27477449056073334063602205')) vpa(sym(' -.22202648081362135993240781 1'))];
intJdn6dn6uvrs =[vpa(sym(' 1.33228970820397080499256458 9')) vpa(sym(' .60206537351843515950311351 9'));...
vpa(sym(' .60206537351843515950311351 9')) vpa(sym(' 1.14198144936988465286082004 9'))];
intJdn6dn7uvrs =[vpa(sym(' -.85797756874373307218110996 2')) vpa(sym(' -.62901590438554706762765091 7'));...
vpa(sym(' -.62901590438554706762765091 7')) vpa(sym(' -.25433776012165343954453545 34'))];
intJdn6dn8uvrs =[vpa(sym(' .31571130247318704127722018 9')) vpa(sym(' .24819901873655990881362319 74'));...
vpa(sym(' .24819901873655990881362319 74')) vpa(sym(' .83732994340339703866566899 6e-1'))];
intJdn6dn9uvrs =[vpa(sym(' .12380804271194872433082513 9')) vpa(sym(' -.79362521672469957651556136e-2'));...
vpa(sym(' -.79362521672469957651556136e-2')) vpa(sym(' -.11543642609252788831829720852'))];
intJdn6dn10uvrs =[vpa(sym(' -.42619180237065523537439729 4e-1')) vpa(sym(' .83212169607440578254945879 1e-1'));...
vpa(sym(' .83212169607440578254945879 1e-1')) vpa(sym(' .49882700990284967023464978 1e-1'))];
intJdn6dn11uvrs =[vpa(sym(' .27527791093307208248725575e-1')) vpa(sym(' .89268098816052016578387726e-1'));...
vpa(sym(' .89268098816052016578387726e-1')) vpa(sym(' .27527791093307208248725575e-1'))];
intJdn6dn12uvrs =[vpa(sym(' -.78897358249752608668273981e-1')) vpa(sym(' -.19696639034109092890969408 4'));...
vpa(sym(' -.19696639034109092890969408 4')) vpa(sym(' -.26978045869880871379513078e-1'))];
intJdn6dn13uvrs =[vpa(sym(' .10736670736867985341988703 9')) vpa(sym(' .82742377912994363911301406 1'));...
vpa(sym(' .82742377912994363911301406 1')) vpa(sym(' .52020218495669789167823211 9'))];
intJdn6dn14uvrs =[vpa(sym(' .60105835750654359495658263')) vpa(sym(' -.51059946095920005197845965 0'));...
vpa(sym(' -.51059946095920005197845965 0')) vpa(sym(' -1.50175278775672029105195536 0'))];
intJdn6dn15uvrs =[vpa(sym(' -.26123164244466091271015883')) vpa(sym(' .10610467066741683413328748 9'));...
vpa(sym(' .10610467066741683413328748 9')) vpa(sym(' .44988102080951953135671271 8'))];
intJdn6dn16uvrs =[vpa(sym(' -.20260922191887153901852490e-1')) vpa(sym(' -.32732430826682450655130346 0'));...
vpa(sym(' -.32732430826682450655130346 0')) vpa(sym(' -.18952780581179438986486843 0'))];
intJdn7dn1uvrs =[vpa(sym(' -.93144679402720430660566560 5e-1')) vpa(sym(' -.10491119938818410070059750 75'));...
vpa(sym(' -.10491119938818410070059750 75')) vpa(sym(' -.92245188408519862030004704e-1'))];
intJdn7dn2uvrs =[vpa(sym(' .28346871318031592670232799 0')) vpa(sym(' -.45084785307694728127485979 8'));...
vpa(sym(' .26165214692305271872514020 15')) vpa(sym(' -.26464728995676401952354811 00'))];
intJdn7dn3uvrs =[vpa(sym(' -.66830280257659072749293639e-1')) vpa(sym(' -.14285359110808823461012582183'));...
vpa(sym(' .15714640889191176538987417817')) vpa(sym(' .95831072591515659590730706 2e-1'))];
intJdn7dn4uvrs =[vpa(sym(' .32737070603635956349312896 1e-3')) vpa(sym(' -.24761717999877543769671456 9e-1'));...
vpa(sym(' -.24761717999877543769671456 9e-1')) vpa(sym(' .51004038179234709257195060e-2'))];
intJdn7dn5uvrs =[vpa(sym(' .30653972944448350892451249 69')) vpa(sym(' .27260597976622541223567498 7'));...
vpa(sym(' .27260597976622541223567498 7')) vpa(sym(' .11801080546717621414491773 4'))];
intJdn7dn6uvrs =[vpa(sym(' -.85797756874373307218110996 2')) vpa(sym(' -.62901590438554706762765091 69'));...
vpa(sym(' -.62901590438554706762765091 69')) vpa(sym(' -.25433776012165343954453545 34'))];
intJdn7dn7uvrs =[vpa(sym(' 2.00108965805772110239126432 7')) vpa(sym(' .27309567677916265610654186068'));...
vpa(sym(' .27309567677916265610654186068')) vpa(sym(' .66650411320415920954619503 8'))];
intJdn7dn8uvrs =[vpa(sym(' -.26463170281510494973966064 8')) vpa(sym(' .39916403134969030939385162 2'));...
vpa(sym(' -.61333596865030969060614837761')) vpa(sym(' -.48812034084686916950769540 60'))];
intJdn7dn9uvrs =[vpa(sym(' .46925409227203366956712440e-1')) vpa(sym(' -.20462265621446298937714495 53'));...
vpa(sym(' -.20462265621446298937714495 53')) vpa(sym(' .53026735207984620537751668 0e-1'))];
intJdn7dn10uvrs =[vpa(sym(' -.54552417791786682579711494e-2')) vpa(sym(' .87773062477896060203191700 1e-1'));...
vpa(sym(' .87773062477896060203191700 1e-1')) vpa(sym(' -.54552417791786682579711494e-2'))];
intJdn7dn11uvrs =[vpa(sym(' .49882700990284967023464978 1e-1')) vpa(sym(' .83212169607440578254945879 1e-1'));...
vpa(sym(' .83212169607440578254945879 1e-1')) vpa(sym(' -.42619180237065523537439729 4e-1'))];
intJdn7dn12uvrs =[vpa(sym(' -.13068498019167439266960372 94')) vpa(sym(' .60593371924778088110680952 -2'));...
vpa(sym(' .60593371924778088110680952 -2')) vpa(sym(' .10667911259258557145532230 09'))];
intJdn7dn13uvrs =[vpa(sym(' .62179054482977490239946348 0')) vpa(sym(' .30448320844630450044664700e-1'));...
vpa(sym(' .30448320844630450044664700e-1')) vpa(sym(' -.11315194309163614035589551 58'))];
intJdn7dn14uvrs =[vpa(sym(' -2.24559114498245358748525615')) vpa(sym(' .32502712079217193433836137e-1'));...
vpa(sym(' .32502712079217193433836137e-1')) vpa(sym(' .46364242643523354151733994 35'))];
intJdn7dn15uvrs =[vpa(sym(' .55242173149670587806391031')) vpa(sym(' .67638491447389996572089416 35'));...
vpa(sym(' .67638491447389996572089416 35')) vpa(sym(' -.29328323700329931136871321 1'))];
intJdn7dn16uvrs =[vpa(sym(' -.19813025976000183828168734 1')) vpa(sym(' -.30423328239753321684461868 6'));...
```

```
vpa(sym('  -.30423328239753321684461868'))  vpa(sym(' .450655121284078464078263 84e-1'))];
intJdn8dn1uvrs =[vpa(sym('  .335182797897334854468695 05e-1'))  vpa(sym('  .392811534864312554969105 16e-1'));...
vpa(sym('  .392811534864312554969105 16e-1'))  vpa(sym(' .318790711453175872507008895e-1'))];
intJdn8dn2uvrs =[vpa(sym('    -.114137962266763606747720 48'))  vpa(sym('  .17064551262371894348372706704'));...
vpa(sym('   -.12935448737628105651627 2933'))  vpa(sym(' .803271661318366652125848671 0e-1'))];
intJdn8dn3uvrs =[vpa(sym('   .217119766902955166405682785'))  vpa(sym('  .348174380528174048440485811 61'));...
vpa(sym(' -.364325619471825951559514188 39'))  vpa(sym('  -.352542724916680732832984020'))];
intJdn8dn4uvrs =[vpa(sym(' -.107656396311072669181942109 0e-1'))  vpa(sym('  .575010022407089943211709137e-1'));...
vpa(sym('  .575010022407089943211709137e-1'))  vpa(sym('  -.353280334795837240195816 55e-1'))];
intJdn8dn5uvrs =[vpa(sym('  -.112360277253857172961520 0020'))  vpa(sym('  -.108578319410524893032405 2264'));...
vpa(sym('  -.108578319410524893032405 2264'))  vpa(sym(' -.424031731725615725542540841e-1'))];

intJdn8dn6uvrs =[vpa(sym('   .315711302473187041277220189'))  vpa(sym('  .248199018736559908813623 1974'));...
vpa(sym('  .248199018736559908813623 1974'))  vpa(sym(' .837329943403397038665668996e-1'))];
intJdn8dn7uvrs =[vpa(sym('  -.264631702815104949739660648'))  vpa(sym(' -.613335968650309690606148377 61'));...
vpa(sym('   .399164031349690309393851622'))  vpa(sym('  -.488120340846869169507695 4060'))];
intJdn8dn8uvrs =[vpa(sym('  1.653151620925835592672919485'))  vpa(sym(' .153178541062607361846242612 68'));...
vpa(sym(' .153178541062607361846242612 68'))  vpa(sym('  .767115078598661079333576130'))];
intJdn8dn9uvrs =[vpa(sym('  -.222366721554688830294037920'))  vpa(sym(' .484990794065840660904685270 82'));...
vpa(sym(' .484990794065840660904685270 82'))  vpa(sym('  -.222366721554688830294037920'))];
intJdn8dn10uvrs =[vpa(sym(' .530267352079846205377516680e-1'))  vpa(sym('  -.204622656214462989377144955 3'));...
vpa(sym('  -.204622656214462989377144955 3'))  vpa(sym('  .469254092272033669567124 40e-1'))];
intJdn8dn11uvrs =[vpa(sym(' -.115436426092527888318297208 52'))  vpa(sym(' -.793625216724699576515561 36e-2'));...
vpa(sym(' -.793625216724699576515561 36e-2'))  vpa(sym('   .123808042711948724330825139'))];
intJdn8dn12uvrs =[vpa(sym(' -.283674858742885011995552238e-1'))  vpa(sym('  -.113465061045209967716224 3619'));...
vpa(sym('  -.113465061045209967716224 3619'))  vpa(sym('  -.134950771499727071139867 6150'))];
intJdn8dn13uvrs =[vpa(sym(' .603290591654115433346960 78e-1'))  vpa(sym('  .331278731126193705006371948'));...
vpa(sym('  .331278731126193705006371948'))  vpa(sym('  .178851051525261210302257368'))];
intJdn8dn14uvrs =[vpa(sym(' -.519957592677779853026270 3e-1'))  vpa(sym(' -.795506545993563782159836 4152'));...
vpa(sym(' -.795506545993563782159836 4152'))  vpa(sym('  -.433262214197932020360238335'))];
intJdn8dn15uvrs =[vpa(sym('   -1.944186194720450713243523 88'))  vpa(sym(' -.266497211117548073090616868e-1'));...
vpa(sym(' -.266497211117548073090616868e-1'))  vpa(sym('    .550843979761993341763278679'))];
intJdn8dn16uvrs =[vpa(sym('   .531391405011459465049996936'))  vpa(sym(' .368453907228382476527592 96e-1'));...
vpa(sym(' .368453907228382476527592 96e-1'))  vpa(sym('  -.154508813774518558307843375'))];
intJdn9dn1uvrs =[vpa(sym(' .318790711453175872507008895e-1'))  vpa(sym('  .392811534864312554969105 16e-1'));...
vpa(sym('  .392811534864312554969105 16e-1'))  vpa(sym(' .335182797897334854468695 045e-1'))];
intJdn9dn2uvrs =[vpa(sym('  -.353280334795837240195816 55e-1'))  vpa(sym('  .575010022407089943211709137e-1'));...
vpa(sym('  .575010022407089943211709137e-1'))  vpa(sym(' -.107656396311072669181942109 0e-1'))];
intJdn9dn3uvrs =[vpa(sym('  -.352542724916680732832984020'))  vpa(sym(' -.364325619471825951559514188 4'));...
vpa(sym(' .348174380528174048440485811 61'))  vpa(sym('  .217119766902955166405682785'))];
intJdn9dn4uvrs =[vpa(sym(' .803271661318366652125848671 0e-1'))  vpa(sym('  -.129354487376281056516272933'));...
vpa(sym('   .170645512623718943483727067 04'))  vpa(sym('    -.114137962266763606747720480'))];
intJdn9dn5uvrs =[vpa(sym('  -.134950771499727071139867 6150'))  vpa(sym('   -.113465061045209967716224362'));...
vpa(sym('   -.113465061045209967716224362'))  vpa(sym(' -.283674858742885011995552238e-1'))];
intJdn9dn6uvrs =[vpa(sym('   .123808042711948724330825139'))  vpa(sym(' -.793625216724699576515561 36e-2'));...
vpa(sym(' -.793625216724699576515561 36e-2'))  vpa(sym(' -.115436426092527888318297208 52'))];
intJdn9dn7uvrs =[vpa(sym('  .469254092272033669567124 40e-1'))  vpa(sym('  -.204622656214462989377144955 3'));...
vpa(sym('  -.204622656214462989377144955 3'))  vpa(sym('  .530267352079846205377516680e-1'))];
intJdn9dn8uvrs =[vpa(sym('  -.222366721554688830294037920'))  vpa(sym(' .484990794065840660904685270 82'));...
vpa(sym(' .484990794065840660904685270 82'))  vpa(sym('  -.222366721554688830294037920'))];
intJdn9dn9uvrs =[vpa(sym('   .767115078598661079333576130'))  vpa(sym(' .153178541062607361846242612 68'));...
vpa(sym(' .153178541062607361846242612 68'))  vpa(sym('  1.653151620925835592672919485'))];
intJdn9dn10uvrs =[vpa(sym('  -.488120340846869169507695 4060'))  vpa(sym('  .399164031349690309393851622 39'));...
vpa(sym(' -.613335968650309690606148377 61'))  vpa(sym('  -.264631702815104949739660648'))];
intJdn9dn11uvrs =[vpa(sym(' .837329943403397038665668996e-1'))  vpa(sym('  .248199018736559908813623 1974'));...
vpa(sym('  .248199018736559908813623 1974'))  vpa(sym('   .315711302473187041277220189'))];
intJdn9dn12uvrs =[vpa(sym(' -.424031731725615725542540841e-1'))  vpa(sym('  -.108578319410524893032405 2264'));...
vpa(sym('  -.108578319410524893032405 2264'))  vpa(sym('  -.112360277253857172961520 0020'))];
intJdn9dn13uvrs =[vpa(sym('   .178851051525261210302257368'))  vpa(sym('  .331278731126193705006371948'));...
vpa(sym('  .331278731126193705006371948'))  vpa(sym(' .603290591654115433346960 78e-1'))];
intJdn9dn14uvrs =[vpa(sym('  -.154508813774518558307843375'))  vpa(sym(' .368453907228382476527592 96e-1'));...
vpa(sym(' .368453907228382476527592 96e-1'))  vpa(sym('   .531391405011459465049996936'))];
intJdn9dn15uvrs =[vpa(sym('    .550843979761993341763278679'))  vpa(sym(' -.266497211117548073090616868e-1'));...
vpa(sym(' -.266497211117548073090616868e-1'))  vpa(sym('    -1.944186194720450713243523 88'))];
intJdn9dn16uvrs =[vpa(sym('  -.433262214197932020360238335'))  vpa(sym(' -.795506545993563782159836 4152'));...
```

vpa(sym(' -.795506545993563782159836415 2')) vpa(sym(' -.5199575926777798530262703e-1'))];
intJdn10dn1uvrs =[vpa(sym(' -.92245188408519862030004704e-1')) vpa(sym(' -.1049111993881841007005975 08')));...
vpa(sym(' -.1049111993881841007005975 08')) vpa(sym(' -.9314467940272043066056656 05e-1'))];
intJdn10dn2uvrs =[vpa(sym(' .5100403817923470925719506 0e-2')) vpa(sym(' -.2476171799987754376967145 69e-1'));...
vpa(sym(' -.2476171799987754376967145 69e-1')) vpa(sym(' .3273707060363595634931289 61e-3'))];
intJdn10dn3uvrs =[vpa(sym(' .9583107259151565959073070 62e-1')) vpa(sym(' .1571464088919117653898741 7817'));...
vpa(sym(' -.1428535911080882346101258 2183')) vpa(sym(' -.6683028025765907274929363 9e-1'))];
intJdn10dn4uvrs =[vpa(sym(' -.2646472899567640195235481 100')) vpa(sym(' .2616521469230527187251402 015'));...
vpa(sym(' -.4508478530769472812748597 985')) vpa(sym(' .2834687131803159267023279 90'))];
intJdn10dn5uvrs =[vpa(sym(' .1066791125925855714553223 01')) vpa(sym(' .6059337192477808811068095 e-2'));...
vpa(sym(' .6059337192477808811068095 e-2')) vpa(sym(' -.1306849801916743926696037 294'))];
intJdn10dn6uvrs =[vpa(sym(' -.4261918023706552353743972 94e-1')) vpa(sym(' .8321216960744057825494587 91e-1'));...
vpa(sym(' .8321216960744057825494587 91e-1')) vpa(sym(' .4988270099028496702346497 81e-1'))];
intJdn10dn7uvrs =[vpa(sym(' -.5455241779178668257971149 4e-2')) vpa(sym(' .8777306247789606020319170 01e-1'));...
vpa(sym(' .8777306247789606020319170 01e-1')) vpa(sym(' -.5455241779178668257971149 4e-2'))];
intJdn10dn8uvrs =[vpa(sym(' .5302673520798462053775166 80e-1')) vpa(sym(' -.2046226562144629893771449 553'));...
vpa(sym(' -.2046226562144629893771449 553')) vpa(sym(' .4692540922720336695671244 0e-1'))];
intJdn10dn9uvrs =[vpa(sym(' -.4881203408468691695076954 060')) vpa(sym(' -.6133359686503096906061483 7761'));...
vpa(sym(' .3991640313496903093938516 2239')) vpa(sym(' -.2646317028151049497396606 48'))];
intJdn10dn10uvrs =[vpa(sym(' .6665041132041592095461950 376')) vpa(sym(' .2730956767791626561065418 6068'));...
vpa(sym(' .2730956767791626561065418 6068')) vpa(sym(' 2.0010896580577211023912643 27'))];
intJdn10dn11uvrs =[vpa(sym(' -.2543377601216534395445354 534')) vpa(sym(' -.6290159043855470676276509 169'));...
vpa(sym(' -.6290159043855470676276509 169')) vpa(sym(' -.8579775687437330721811099 62'))];
intJdn10dn12uvrs =[vpa(sym(' .1180108054671762141449177 34')) vpa(sym(' .2726059797662254122356749 87'));...
vpa(sym(' .2726059797662254122356749 87')) vpa(sym(' .3065397294444835089245124 969'))];
intJdn10dn13uvrs =[vpa(sym(' -.1131519430916361403558955 158')) vpa(sym(' .3044832084463045004466470 0e-1'));...
vpa(sym(' .3044832084463045004466470 0e-1')) vpa(sym(' .6217905448297749023994634 80'))];

intJdn10dn14uvrs =[vpa(sym(' .4506551212840784640782638 4e-1')) vpa(sym(' -.3042332823975332168446186 86'));...
vpa(sym(' -.3042332823975332168446186 86')) vpa(sym(' -.1981302597600018382816873 41'))];
intJdn10dn15uvrs =[vpa(sym(' -.2932832370032993113687132 11')) vpa(sym(' .6763849144738999657208941 635'));...
vpa(sym(' .6763849144738999657208941 635')) vpa(sym(' .5524217314967058780639103 1'))];
intJdn10dn16uvrs =[vpa(sym(' .4636424264352335415173399 435')) vpa(sym(' .3250271207921719343383613 7e-1'));...
vpa(sym(' .3250271207921719343383613 7e-1')) vpa(sym(' -2.2455911449824535874852561 5'))];
intJdn11dn1uvrs =[vpa(sym(' -.7002708952501684636496711 3e-1')) vpa(sym(' -.1425920794764966703892546 82'));...
vpa(sym(' .1574079205235033296107453 18')) vpa(sym(' .1172503945374017479674604 967'))];
intJdn11dn2uvrs =[vpa(sym(' -.8049498872369479004123000 2e-2')) vpa(sym(' -.2396099145512853581863483 61e-1'));...
vpa(sym(' -.2396099145512853581863483 61e-1')) vpa(sym(' .2353045794858365263235082 7e-2'))];
intJdn11dn3uvrs =[vpa(sym(' -.2290194853292015105870834 77e-1')) vpa(sym(' -.7011142235256930991307362 00e-1'));...
vpa(sym(' -.7011142235256930991307362 00e-1')) vpa(sym(' -.9563153300224068809754350 1e-1'))];
intJdn11dn4uvrs =[vpa(sym(' .1378297018364707614848534 907')) vpa(sym(' -.3225407916324774100791670 851'));...
vpa(sym(' .3899592083675225899208329 149')) vpa(sym(' -.4840603365614064676180711 9'))];
intJdn11dn5uvrs =[vpa(sym(' -.2697804586988087137951307 8e-1')) vpa(sym(' -.1969663903410909289096940 84'));...
vpa(sym(' -.1969663903410909289096940 84')) vpa(sym(' -.7889735824975260866827398 1e-1'))];
intJdn11dn6uvrs =[vpa(sym(' .2752779109330720824872557 5e-1')) vpa(sym(' .8926809881605201657838772 6e-1'));...
vpa(sym(' .8926809881605201657838772 6e-1')) vpa(sym(' .2752779109330720824872557 5e-1'))];
intJdn11dn7uvrs =[vpa(sym(' .4988270099028496702346497 81e-1')) vpa(sym(' .8321216960744057825494587 91e-1'));...
vpa(sym(' .8321216960744057825494587 91e-1')) vpa(sym(' -.4261918023706552353743972 94e-1'))];
intJdn11dn8uvrs =[vpa(sym(' -.1154364260925278883182972 0852')) vpa(sym(' -.7936252167246995765155613 6e-2'));...
vpa(sym(' -.7936252167246995765155613 6e-2')) vpa(sym(' .1238080427119487243308251 39'))];
intJdn11dn9uvrs =[vpa(sym(' .8373299434033970386656689 96e-1')) vpa(sym(' .2481990187365599088136231 974'));...
vpa(sym(' .2481990187365599088136231 974')) vpa(sym(' .3157113024731870412772201 89'))];
intJdn11dn10uvrs =[vpa(sym(' -.2543377601216534395445354 534')) vpa(sym(' -.6290159043855470676276509 169'));...
vpa(sym(' -.6290159043855470676276509 169')) vpa(sym(' -.8579775687437330721811099 62'))];
intJdn11dn11uvrs =[vpa(sym(' 1.1419814493698846528608200 49')) vpa(sym(' .6020653735184351595031135 19'));...
vpa(sym(' .6020653735184351595031135 19')) vpa(sym(' 1.3322897082039708049925645 89'))];
intJdn11dn12uvrs =[vpa(sym(' -.2220264808136213599324078 11')) vpa(sym(' .2747744905607333406360220 46'));...
vpa(sym(' -.7377255094392666593639779 54')) vpa(sym(' -.7866868084644167345983151 3779'))];
intJdn11dn13uvrs =[vpa(sym(' .5202021849566978916782321 19')) vpa(sym(' .8274237791299436391130140 61'));...
vpa(sym(' .8274237791299436391130140 61')) vpa(sym(' .1073667073686798534198870 39'))];
intJdn11dn14uvrs =[vpa(sym(' -.1895278058117943898648684 30')) vpa(sym(' -.3273243082668245065513034 60'));...
vpa(sym(' -.3273243082668245065513034 60')) vpa(sym(' -.2026092219188715390185249 0e-1'))];
intJdn11dn15uvrs =[vpa(sym(' .4498810208095195313567127 18')) vpa(sym(' .1061046706674168341332874 89'));...
vpa(sym(' .1061046706674168341332874 89')) vpa(sym(' -.2612316424446609127101588 3'))];
intJdn11dn16uvrs =[vpa(sym(' -1.5017527877567202910519553 60')) vpa(sym(' -.5105994609592000519784596 50'));...

```
vpa(sym(' -.5105994609592000051978459650')) vpa(sym('  .601058357506543594956582 63'))];
intJdn12dn1uvrs =[vpa(sym(' .157871624541338704794557829')) vpa(sym(' .26894552598477698154302 2533'));...
vpa(sym(' -.443554474015223018456977467')) vpa(sym(' -.41263305411760023071620 6898'))];
intJdn12dn2uvrs =[vpa(sym(' .220346626285956934003016986e-1')) vpa(sym(' .49548766326640713476103 3842e-1'));...
vpa(sym(' .495487663266407134761033842e-1')) vpa(sym(' -.20270550705857146813452057e-1'))];
intJdn12dn3uvrs =[vpa(sym(' .120306246990647939530980 43e-1')) vpa(sym(' .30662548740471170697954 5881e-1'));...
vpa(sym(' .306625487404711706979545881e-1')) vpa(sym(' .31225897458880491587918 5904e-1'))];
intJdn12dn4uvrs =[vpa(sym(' -.356070031451497704872952070e-1')) vpa(sym('  .14935188545774774523352 26299'));...
vpa(sym('  -.150648114542252254766477370')) vpa(sym('  .13531648572980233462395 09070'))];
intJdn12dn5uvrs =[vpa(sym(' .892116545816381069687669 6e-1')) vpa(sym(' .42718888479907807017399238 6'));...
vpa(sym(' .427188884799078070173992386')) vpa(sym(' .892116545816381069687669 6e-1'))];
intJdn12dn6uvrs =[vpa(sym(' -.788973582497526086682739 81e-1')) vpa(sym('  -.19696639034109092890969 4084'));...
vpa(sym('  -.196966390341090928909694084')) vpa(sym(' -.26978045869880871379513 078e-1'))];
intJdn12dn7uvrs =[vpa(sym(' -.130684980191674392669603 7294')) vpa(sym(' .60593371924778088110680 95e-2'));...
vpa(sym(' .605933719247780881106809 5e-2')) vpa(sym(' .10667911259258557145532 23009'))];
intJdn12dn8uvrs =[vpa(sym(' -.283674858742885011995552 38e-1')) vpa(sym('  -.11346506104520996771622 43619'));...
vpa(sym('  -.113465061045209967716224 3619')) vpa(sym('  -.13495077149972707113986 76150'))];
intJdn12dn9uvrs =[vpa(sym(' -.424031731725615725542540841e-1')) vpa(sym('  -.10857831941052489303240 52264'));...
vpa(sym('  -.108578319410524893032405 2264')) vpa(sym('  -.11236027725385717296152 00020'))];
intJdn12dn10uvrs =[vpa(sym(' .118010805467176214144917734')) vpa(sym(' .27260597976622541223567 4987'));...
vpa(sym(' .272605979766225412235674987')) vpa(sym(' .30653972944448350892451 24969'))];
intJdn12dn11uvrs =[vpa(sym('  -.22202648081362135993240 7811')) vpa(sym('  -.73772550943926665936397 7954'));...
vpa(sym('  .27477449056073334063602 2046')) vpa(sym(' -.78668680846441673459831 513779'))];
intJdn12dn12uvrs =[vpa(sym(' 1.49530824854081722232161 3675')) vpa(sym(' .90287598944150236268304 0391'));...
vpa(sym(' .902875989441502362683040391')) vpa(sym(' 1.31758678728731273266908 3137'))];
intJdn12dn13uvrs =[vpa(sym(' -1.83117956250893266072281 3983')) vpa(sym(' -.53084335598705132620498 7917'));...
vpa(sym(' -.530843355987051326204987917')) vpa(sym('  .47555890783193200318874 678'))];
intJdn12dn14uvrs =[vpa(sym('  .53403155562725169509198 6887')) vpa(sym(' .91268611031924988409224 700e-1'));...
vpa(sym(' .912686110319249884092247 00e-1')) vpa(sym(' -.22254315718435603337366 47188'))];
intJdn12dn15uvrs =[vpa(sym(' .869045477724801039031066 92e-1')) vpa(sym('  .37936074707194774196023 2042'));...
vpa(sym('  .379360747071947741960232042')) vpa(sym('  .44987769212666366162955 2350'))];
intJdn12dn16uvrs =[vpa(sym('  -.14623767990238166834414 5480')) vpa(sym('  -.89028963958964921999654 6217'));...
vpa(sym('  -.890289639589649219996546217')) vpa(sym(' -1.19557360195760315006531 36788'))];
intJdn13dn1uvrs =[vpa(sym('  -.92367783910183130555067 710')) vpa(sym(' -1.00703245489050046307564 230'));...
vpa(sym(' -1.007032454890500463075642 30')) vpa(sym('  -.92367783910183130555067 710'))];
intJdn13dn2uvrs =[vpa(sym(' -.624603048181793546679316 6e-1')) vpa(sym('  -.19613023897523854894811 1488'));...
vpa(sym('  -.196130238975238548948111488')) vpa(sym(' .18066447765168896456321 824e-1'))];
intJdn13dn3uvrs =[vpa(sym(' -.440107128429126521075770 644e-1')) vpa(sym(' -.96415975295660771189914 2376e-1'));...
vpa(sym(' -.964159752956607711899142376e-1')) vpa(sym(' -.44010712842912652107577 0644e-1'))];
intJdn13dn4uvrs =[vpa(sym('  .18066447765168896456321 824e-1')) vpa(sym('  -.19613023897523854894811 1488'));...
vpa(sym('  -.196130238975238548948111488')) vpa(sym(' -.62460304818179354667931 6570e-1'))];
intJdn13dn5uvrs =[vpa(sym('  .47555890783193200318874 648')) vpa(sym('  -.53084335598705132620498 792'));...
vpa(sym('  -.530843355987051326204987 92')) vpa(sym(' -1.83117956250893266072281 398'))];

intJdn13dn6uvrs =[vpa(sym(' .107366707368679853419887039')) vpa(sym(' .82742377912994363911301 4061'));...
vpa(sym(' .827423779129943639113014061')) vpa(sym(' .52020218495669789167823 2119'))];
intJdn13dn7uvrs =[vpa(sym('  .62179054482977490239946348')) vpa(sym(' .30448320844630450044664 700e-1'));...
vpa(sym(' .304483208446304500446647 00e-1')) vpa(sym(' -.11315194309163614035589 55158'))];
intJdn13dn8uvrs =[vpa(sym(' .603290591654115433346960 78e-1')) vpa(sym('  .33127873112619370500637 1948'));...
vpa(sym('  .331278731126193705006371948')) vpa(sym('  .17885105152526121030225 7368'))];
intJdn13dn9uvrs =[vpa(sym('  .17885105152526121030225 7368')) vpa(sym('  .33127873112619370500637 1948'));...
vpa(sym('  .331278731126193705006371948')) vpa(sym(' .60329059165411543334696 078e-1'))];
intJdn13dn10uvrs =[vpa(sym(' -.113151943091636140355895 5158')) vpa(sym(' .30448320844630450044664 700e-1'));...
vpa(sym(' .304483208446304500446647 00e-1')) vpa(sym('  .62179054482977490239946 3480'))];
intJdn13dn11uvrs =[vpa(sym(' .520202184956697891678232119')) vpa(sym(' .82742377912994363911301 4061'));...
vpa(sym(' .827423779129943639113014061')) vpa(sym(' .10736670736867985341988 7039'))];
intJdn13dn12uvrs =[vpa(sym(' -1.83117956250893266072281 3983')) vpa(sym(' -.53084335598705132620498 7917'));...
vpa(sym(' -.530843355987051326204987 917')) vpa(sym('  .47555890783193200318874 678'))];
intJdn13dn13uvrs =[vpa(sym(' 5.22041258699831676842108 8045')) vpa(sym('  2.88129572754069028280297 951'));...
vpa(sym('  2.881295727540690282802979 51')) vpa(sym(' 5.22041258699831676842108 8045'))];
intJdn13dn14uvrs =[vpa(sym(' -3.400472479844200126665206259')) vpa(sym('  -.87612662426029179489909 5697'));...
vpa(sym('  -.876126624260291794899095697')) vpa(sym(' -.80962223894644387409993 96603'))];
intJdn13dn15uvrs =[vpa(sym(' -.180024092871069550306511 38e-1')) vpa(sym('  -.94994852111090129676113 4222'));...
vpa(sym('  -.949948521110901296761134222')) vpa(sym(' -.18002409287106955030651 138e-1'))];
intJdn13dn16uvrs =[vpa(sym(' -.809622238946443874099939 6603')) vpa(sym('  -.87612662426029179489909 5697'));...
```

```
vpa(sym('  -.87612662426029179489909569 7')) vpa(sym(' -3.400472479844200126665206259'))];
intJdn14dn1uvrs =[vpa(sym(' .336851990222722864094172519')) vpa(sym(' .381671404352391717716050003'));...
vpa(sym(' .381671404352391717716050003')) vpa(sym('  .338459011543472358906908 60'))];
intJdn14dn2uvrs =[vpa(sym(' -.565454471791414699007259 5e-2')) vpa(sym('  .427622370384584419718474494 1'));...
vpa(sym('  .427622370384584419718474494 1')) vpa(sym(' -.153931015713211339133137848 6'))];
intJdn14dn3uvrs =[vpa(sym('  .136999462878579334021389201')) vpa(sym('  .227213413649009719816657263 6'));...
vpa(sym('  .227213413649009719816657263 6')) vpa(sym(' .941185558099118441713021 74e-1'))];
intJdn14dn4uvrs =[vpa(sym('  -.984668166045009257980461 6e-3')) vpa(sym(' .863000808306781359312991519e-1'));...
vpa(sym(' .863000808306781359312991519e-1')) vpa(sym(' .208266894443892215290932469e-1'))];
intJdn14dn5uvrs =[vpa(sym(' -1.195573601957603150065313 6788')) vpa(sym('  -.890289639589649219996546217'));...
vpa(sym('  -.890289639589649219996546217')) vpa(sym('  -.146237679902381668344145 48'))];
intJdn14dn6uvrs =[vpa(sym('  .601058357506543594956582 63')) vpa(sym('  -.510599460959200051978459650'));...
vpa(sym('  -.510599460959200051978459650')) vpa(sym(' -1.501752787756720291051955360'))];
intJdn14dn7uvrs =[vpa(sym('  -2.245591144982453587485256 15')) vpa(sym(' .325027120792171934338361 37e-1'));...
vpa(sym(' .325027120792171934338361 37e-1')) vpa(sym('  .463642426435233541517339943 5'))];
intJdn14dn8uvrs =[vpa(sym(' -.519957592677779853026270 3e-1')) vpa(sym(' -.795506545993563782159836415 2'));...
vpa(sym(' -.795506545993563782159836415 2')) vpa(sym('  -.433262214197932020360238335'))];
intJdn14dn9uvrs =[vpa(sym('  -.154508813774518558307843375')) vpa(sym(' .368453907228382476527592 96e-1'));...
vpa(sym(' .368453907228382476527592 96e-1')) vpa(sym('  .531391405011459465049996936'))];
intJdn14dn10uvrs =[vpa(sym(' .450655121284078464078263 84e-1')) vpa(sym('  -.304233282397533216844618686'));...
vpa(sym('  -.304233282397533216844618686')) vpa(sym('  -.198130259760001838281687341'))];
intJdn14dn11uvrs =[vpa(sym('  -.189527805811794389864868430')) vpa(sym('  -.327324308266824506551303460'));...
vpa(sym('  -.327324308266824506551303460')) vpa(sym(' -.202609221918871539018524 90e-1'))];
intJdn14dn12uvrs =[vpa(sym('  .534031555627251695091986887')) vpa(sym(' .912686110319249884092247 00e-1'));...
vpa(sym(' .912686110319249884092247 00e-1')) vpa(sym(' -.222543157184356033373664 7188'))];
intJdn14dn13uvrs =[vpa(sym(' -3.400472479844200126665206259')) vpa(sym('  -.876126624260291794899095697'));...
vpa(sym('  -.876126624260291794899095697')) vpa(sym(' -.809622238946443874099939 6603'))];
intJdn14dn14uvrs =[vpa(sym('  5.518761290142117566125380 31')) vpa(sym(' 1.770759558087978375081430869'));...
vpa(sym(' 1.770759558087978375081430869')) vpa(sym(' 3.929578185962763299703317274'))];
intJdn14dn15uvrs =[vpa(sym('  -.805977919806664929391926 72')) vpa(sym('  -.637841602652030943257059331'));...
vpa(sym('  -.637841602652030943257059331')) vpa(sym(' -2.769794568377644494965093629'))];
intJdn14dn16uvrs =[vpa(sym('  .877518569823348982633756744')) vpa(sym(' 1.287737922980470717927187534'));...
vpa(sym(' 1.287737922980470717927187534')) vpa(sym('  .877518569823348982633756744'))];
intJdn15dn1uvrs =[vpa(sym(' -.122483212211684061698359345')) vpa(sym(' -.145263095437075394958553495'));...
vpa(sym(' -.145263095437075394958553495')) vpa(sym(' -.122483212211684061698359345'))];
intJdn15dn2uvrs =[vpa(sym('  .803323805596981177025893 0e-2')) vpa(sym(' -.194216973763714354534537 5175'));...
vpa(sym(' -.194216973763714354534537 5175')) vpa(sym(' .416813143674259981116969 12e-1'))];
intJdn15dn3uvrs =[vpa(sym('  -.319448664201393584204597479')) vpa(sym(' -.534960505588025669879559805 17'));...
vpa(sym(' -.534960505588025669879559805 17')) vpa(sym('  -.319448664201393584204597479'))];
intJdn15dn4uvrs =[vpa(sym(' .416813143674259981116969 119e-1')) vpa(sym('  -.194216973763714354534537 5175'));...
vpa(sym('  -.194216973763714354534537 5175')) vpa(sym('  .803323805596981177025893 0e-2'))];
intJdn15dn5uvrs =[vpa(sym('  .449877692126663661629552350')) vpa(sym('  .379360747071947741960232042'));...
vpa(sym('  .379360747071947741960232042')) vpa(sym(' .869045477724801039031066 92e-1'))];
intJdn15dn6uvrs =[vpa(sym(' -.261231642444660912710158 83')) vpa(sym(' .106104670667416834133287489'));...
vpa(sym(' .106104670667416834133287489')) vpa(sym('  .449881020809519531356712 72'))];
intJdn15dn7uvrs =[vpa(sym('  .552421731496705878063910 31')) vpa(sym(' .676384914473899965720894 1635'));...
vpa(sym(' .676384914473899965720894 1635')) vpa(sym(' -.293283237003299311368713211'))];
intJdn15dn8uvrs =[vpa(sym('  -1.944186194720450713243523 88')) vpa(sym(' -.266497211117548073090616868e-1'));...
vpa(sym(' -.266497211117548073090616868e-1')) vpa(sym('  .550843979761993341763278679'))];
intJdn15dn9uvrs =[vpa(sym('  .550843979761993341763278679')) vpa(sym(' -.266497211117548073090616868e-1'));...
vpa(sym(' -.266497211117548073090616868e-1')) vpa(sym('  -1.944186194720450713243523 88'))];
intJdn15dn10uvrs =[vpa(sym(' -.293283237003299311368713211')) vpa(sym(' .676384914473899965720894 1635'));...
vpa(sym(' .676384914473899965720894 1635')) vpa(sym('  .552421731496705878063910 31'))];
intJdn15dn11uvrs =[vpa(sym(' .449881020809519531356712 718')) vpa(sym(' .106104670667416834133287489'));...
vpa(sym(' .106104670667416834133287489')) vpa(sym(' -.261231642444660912710158 83'))];
intJdn15dn12uvrs =[vpa(sym(' .869045477724801039031066 92e-1')) vpa(sym('  .379360747071947741960232042'));...
vpa(sym('  .379360747071947741960232042')) vpa(sym('  .449877692126663661629552350'))];
intJdn15dn13uvrs =[vpa(sym(' -.180024092871069550306511 38e-1')) vpa(sym('  -.949948521110901296761134222'));...
vpa(sym('  -.949948521110901296761134222')) vpa(sym(' -.180024092871069550306511 38e-1'))];

intJdn15dn14uvrs =[vpa(sym('  -.805977919806664929391926 72')) vpa(sym('  -.637841602652030943257059331'));...
vpa(sym('  -.637841602652030943257059331')) vpa(sym(' -2.769794568377644494965093629'))];
intJdn15dn15uvrs =[vpa(sym('  4.394764323662146636014507 61')) vpa(sym(' 1.023888052764473488171737195'));...
vpa(sym(' 1.023888052764473488171737195')) vpa(sym('  4.394764323662146636014507 61'))];
intJdn15dn16uvrs =[vpa(sym(' -2.769794568377644494965093629')) vpa(sym('  -.637841602652030943257059331'));...
```

```
vpa(sym('  -.63784160265203094325705933'))  vpa(sym('   -.80597791980666492939192672'))];
intJdn16dn1uvrs =[vpa(sym('  .33845901154347235890690860'))  vpa(sym(' .38167140435239171771605003'));...
vpa(sym(' .38167140435239171771605003'))  vpa(sym('  .33685199022272286409417252'))];
intJdn16dn2uvrs =[vpa(sym(' .20826689444389221529093246 9e-1'))  vpa(sym(' .86300080830678135931299151 9e-1'));...
vpa(sym(' .86300080830678135931299151 9e-1'))  vpa(sym('  -.98466816604500925798046 16e-3'))];
intJdn16dn3uvrs =[vpa(sym(' .94118555809911844171302174 4e-1'))  vpa(sym('   .22721341364900971981665726 36'));...
vpa(sym('   .22721341364900971981665726 36'))  vpa(sym('    .13699946287857933402138920 1'))];
intJdn16dn4uvrs =[vpa(sym(' -.15393101571321133913313784 86'))  vpa(sym('  .42762237038458441971847449 41'));...
vpa(sym('  .42762237038458441971847449 41'))  vpa(sym(' -.56545447179141469900725 95e-2'))];
intJdn16dn5uvrs =[vpa(sym(' -.22254315718435603337366471 88'))  vpa(sym(' .91268611031924988409224700e-1'));...
vpa(sym(' .91268611031924988409224700e-1'))  vpa(sym('   .53403155562725169509198688 7'))];
intJdn16dn6uvrs =[vpa(sym(' -.20260922191887153901852490e-1'))  vpa(sym('    -.32732430826682450655130346'));...
vpa(sym('    -.32732430826682450655130346'))  vpa(sym('   -.18952780581179438986486843 0'))];
intJdn16dn7uvrs =[vpa(sym('  -.19813025976000183828168734 1'))  vpa(sym('  -.30423328239753321684461868 6'));...
vpa(sym('  -.30423328239753321684461868 6'))  vpa(sym(' .45065512128407846407826384e-1'))];
intJdn16dn8uvrs =[vpa(sym('   .53139140501145946504999693 6'))  vpa(sym(' .36845390722838247652759296e-1'));
vpa(sym(' .36845390722838247652759296e-1'))  vpa(sym('  -.15450881377451855830784337 5'))];
intJdn16dn9uvrs =[vpa(sym('  -.43326221419793202036023833 5'))  vpa(sym(' -.79550654599356378215983641 52'));...
vpa(sym(' -.79550654599356378215983641 52'))  vpa(sym(' -.51995759267777985302627 03e-1'))];
intJdn16dn10uvrs =[vpa(sym('  .46364242643523354151733994 35'))  vpa(sym(' .32502712079217193433836137e-1'));...
vpa(sym(' .32502712079217193433836137e-1'))  vpa(sym('  -2.24559114498245358748525615'))];
intJdn16dn11uvrs =[vpa(sym(' -1.50175278775672029105195536 0'))  vpa(sym('  -.51059946095920005197845965 0'));...
vpa(sym('  -.51059946095920005197845965 0'))  vpa(sym('    .60105835750654359495658263'))];
intJdn16dn12uvrs =[vpa(sym('   -.14623767990238166834414548 0'))  vpa(sym('   -.89028963958964921999654621 7'));...
vpa(sym('   -.89028963958964921999654621 7'))  vpa(sym(' -1.19557360195760315006531367 88'))];
intJdn16dn13uvrs =[vpa(sym(' -.80962223894644387409993966 03'))  vpa(sym('  -.87612662426029179489909569 7'));...
vpa(sym('  -.87612662426029179489909569 7'))  vpa(sym(' -3.40047247984420012666520625 9'))];
intJdn16dn14uvrs =[vpa(sym('  .87751856982334898263375674 4'))  vpa(sym(' 1.28773792298047071792718753 4'));...
vpa(sym(' 1.28773792298047071792718753 4'))  vpa(sym('  .87751856982334898263375674 4'))];
intJdn16dn15uvrs =[vpa(sym(' -2.76979456837764449496509362 9'))  vpa(sym('  -.63784160265203094325705933 1'));...
vpa(sym('  -.63784160265203094325705933 1'))  vpa(sym('   -.80597791980666492939192672'))];
intJdn16dn16uvrs =[vpa(sym(' 3.92957818596276329970331727 4'))  vpa(sym(' 1.77075955808797837508143086 9'));...
vpa(sym(' 1.77075955808797837508143086 9'))  vpa(sym('  5.51876129014211756612538031'))];
%disp('---------------------------- integrals of products of global deravatives----------------------------------------------')

intJdndn(1:2,1:32)=[intJdn1dn1uvrs intJdn1dn2uvrs intJdn1dn3uvrs intJdn1dn4uvrs intJdn1dn5uvrs intJdn1dn6uvrs
intJdn1dn7uvrs intJdn1dn8uvrs intJdn1dn9uvrs intJdn1dn10uvrs intJdn1dn11uvrs intJdn1dn12uvrs ...
        intJdn1dn13uvrs intJdn1dn14uvrs intJdn1dn15uvrs intJdn1dn16uvrs];
intJdndn(3:4,1:32)=[intJdn2dn1uvrs intJdn2dn2uvrs intJdn2dn3uvrs intJdn2dn4uvrs intJdn2dn5uvrs intJdn2dn6uvrs
intJdn2dn7uvrs intJdn2dn8uvrs intJdn2dn9uvrs intJdn2dn10uvrs intJdn2dn11uvrs intJdn2dn12uvrs ...
        intJdn2dn13uvrs intJdn2dn14uvrs intJdn2dn15uvrs intJdn2dn16uvrs];
intJdndn(5:6,1:32)=[intJdn3dn1uvrs intJdn3dn2uvrs intJdn3dn3uvrs intJdn3dn4uvrs intJdn3dn5uvrs intJdn3dn6uvrs
intJdn3dn7uvrs intJdn3dn8uvrs intJdn3dn9uvrs intJdn3dn10uvrs intJdn3dn11uvrs intJdn3dn12uvrs ...
        intJdn3dn13uvrs intJdn3dn14uvrs intJdn3dn15uvrs intJdn3dn16uvrs];
intJdndn(7:8,1:32)=[intJdn4dn1uvrs intJdn4dn2uvrs intJdn4dn3uvrs intJdn4dn4uvrs intJdn4dn5uvrs intJdn4dn6uvrs
intJdn4dn7uvrs intJdn4dn8uvrs intJdn4dn9uvrs intJdn4dn10uvrs intJdn4dn11uvrs intJdn4dn12uvrs ...
        intJdn4dn13uvrs intJdn4dn14uvrs intJdn4dn15uvrs intJdn4dn16uvrs];
intJdndn(9:10,1:32)=[intJdn5dn1uvrs intJdn5dn2uvrs intJdn5dn3uvrs intJdn5dn4uvrs intJdn5dn5uvrs intJdn5dn6uvrs
intJdn5dn7uvrs intJdn5dn8uvrs intJdn5dn9uvrs intJdn5dn10uvrs intJdn5dn11uvrs intJdn5dn12uvrs ...
        intJdn5dn13uvrs intJdn5dn14uvrs intJdn5dn15uvrs intJdn5dn16uvrs];
intJdndn(11:12,1:32)=[intJdn6dn1uvrs intJdn6dn2uvrs intJdn6dn3uvrs intJdn6dn4uvrs intJdn6dn5uvrs intJdn6dn6uvrs
intJdn6dn7uvrs intJdn6dn8uvrs intJdn6dn9uvrs intJdn6dn10uvrs intJdn6dn11uvrs intJdn6dn12uvrs ...
        intJdn6dn13uvrs intJdn6dn14uvrs intJdn6dn15uvrs intJdn6dn16uvrs];
intJdndn(13:14,1:32)=[intJdn7dn1uvrs intJdn7dn2uvrs intJdn7dn3uvrs intJdn7dn4uvrs intJdn7dn5uvrs intJdn7dn6uvrs
intJdn7dn7uvrs intJdn7dn8uvrs intJdn7dn9uvrs intJdn7dn10uvrs intJdn7dn11uvrs intJdn7dn12uvrs ...
        intJdn7dn13uvrs intJdn7dn14uvrs intJdn7dn15uvrs intJdn7dn16uvrs];
intJdndn(15:16,1:32)=[intJdn8dn1uvrs intJdn8dn2uvrs intJdn8dn3uvrs intJdn8dn4uvrs intJdn8dn5uvrs intJdn8dn6uvrs
intJdn8dn7uvrs intJdn8dn8uvrs intJdn8dn9uvrs intJdn8dn10uvrs intJdn8dn11uvrs intJdn8dn12uvrs ...
        intJdn8dn13uvrs intJdn8dn14uvrs intJdn8dn15uvrs intJdn8dn16uvrs];
intJdndn(17:18,1:32)=[intJdn9dn1uvrs intJdn9dn2uvrs intJdn9dn3uvrs intJdn9dn4uvrs intJdn9dn5uvrs intJdn9dn6uvrs
intJdn9dn7uvrs intJdn9dn8uvrs intJdn9dn9uvrs intJdn9dn10uvrs intJdn9dn11uvrs intJdn9dn12uvrs ...
        intJdn9dn13uvrs intJdn9dn14uvrs intJdn9dn15uvrs intJdn9dn16uvrs];
intJdndn(19:20,1:32)=[intJdn10dn1uvrs intJdn10dn2uvrs intJdn10dn3uvrs intJdn10dn4uvrs intJdn10dn5uvrs intJdn10dn6uvrs
intJdn10dn7uvrs intJdn10dn8uvrs intJdn10dn9uvrs intJdn10dn10uvrs intJdn10dn11uvrs intJdn10dn12uvrs ...
        intJdn10dn13uvrs intJdn10dn14uvrs intJdn10dn15uvrs intJdn10dn16uvrs];
```

```
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 choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%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)
%syms mst_tri x
ne=0;
nitri=nmax-1;
for itri=1:nitri
   elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
if itri==1
kk=nmax;
for k=2:n
   kk=kk+1
   elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
   elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
   lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
```

```
    mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
    hh=hh+1;
    nni=nni+(n-i);
    elm(nni,1)=(mmax+1)+i;
    qq(hh,1)=(mmax+1)+i;

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

if itri<nitri
disp('left edge nodes')
nni=1;gg=1;pp(1,1)=n1(itri,1);
for i=0:(n-2)
    gg=gg+1;
    nni=nni+(n-i)+1;
    elm(nni,1)=lmax-i;
    pp(gg,1)=lmax-i;
end
pp(n+1,1)=n3(itri,1);
edgen1n3(1:n+1,itri)=pp
end%if itri<nitri
if itri==nitri
disp('left edge nodes')
nni=1;gg=1;
for i=0:(n-2)
    gg=gg+1;
    nni=nni+(n-i)+1;
    elm(nni,1)=edgen1n2(gg,1);
end
%pp(n+1,1)=n3(itri,1);
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri
disp('interior nodes')
nni=1;jj=0;
for i=0:(n-3)
    nni=nni+(n-i)+1;
    for j=1:(n-2-i)
        jj=jj+1;
        nnj=nni+j;
        elm(nnj,1)=lmax+jj;
        [nnj lmax+jj];
    end
end
%disp(elm);
%disp(length(elm));

jj=0;kk=0;
for j=0:n-1
    jj=j+1;
for k=1:(n+1)-j
    kk=kk+1;
    row_nodes(jj,k)=elm(kk,1);
end
end
row_nodes(n+1,1)=n3(itri,1);
%for jj=(n+1):-1:1
%    (row_nodes(jj,:));
```

```
%end
%[row_nodes]
rr=row_nodes;
rr
rrr(:,:,itri)=rr;
disp('element computations')
if rem(n,2)==0
N=n+1;

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

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

nnd=max(max(eln))
if (n>3)
for kkk=1+(itri-1)*numtri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
end
end
if n==2
 for kkk=itri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
 end
end
nmax=max(max(eln));
%nel=mm;
%
%ne
%spqd


end%itri
```

```
%to generate special quadrilaterals
mm=0;

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

for 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
  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 ui vi wi U V W
kk=0;
for j=1:n+1
   for i=1:(n+1)-(j-1)
      kk=kk+1;
      ui(i,j)=(i-1)/n;
      vi(i,j)=(j-1)/n;
      wi(i,j)=1-ui(i,j)-vi(i,j);
      U(kk,1)=ui(i,j);
      V(kk,1)=vi(i,j);
      W(kk,1)=wi(i,j);
   end
end
```

[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 choosen triangle
%n2=node number at(1,0)for  a choosen triangle
%n3=node number at(0,1)for  a choosen triangle
%eln=6-node triangles with centroid
%spqd=4-node special convex quadrilateral
%n must be even,i.e.n=2,4,6,.....i.e number of divisions
%nmax=one plus the number of segments of the polygon
%nmax=the number of segments of the polygon plus a node interior to the polygon
%numtri=number of T6 triangles in each segment i.e a triangle formed by
%joining the end poits of the segment to the interior point(e.g:the centroid) of the polygon
%PARVIZ 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)
%syms mst_tri x
ne=0;
nitri=nmax-1;
for itri=1:nitri
   elm(1:(n+1)*(n+2)/2,1)=zeros((n+1)*(n+2)/2,1)
```

```
elm(1,1)=n1(itri,1)
elm(n+1,1)=n2(itri,1)
elm((n+1)*(n+2)/2,1)=n3(itri,1)
disp('vertex nodes of the itri triangle')
[n1(itri,1) n2(itri,1) n3(itri,1)]
if itri==1
kk=nmax;
for k=2:n
   kk=kk+1
   elm(k,1)=kk
end
disp('base nodes=')
%elm(2:n)
edgen1n2(1:n+1,itri)=elm(1:n+1,1)
end%itri==1
if itri>1
   elm(1:n+1,1)=edgen1n3(1:n+1,itri-1);
end%if itri>1
if itri==1
   lmax=nmax+3*(n-1);
end%if itri==1
if (itri>1)&(itri<nitri)
   lmax=nmax+2*(n-1);
end% if (itri>1)&(itri<nitri)
mmax=nmax;
if itri==1
   mmax=max(max(edgen1n2(1:n+1,1)))
end%f itri==1
disp('right edge nodes')
nni=n+1;hh=1;qq(1,1)=n2(itri,1);
for i=0:(n-2)
   hh=hh+1;
   nni=nni+(n-i);
   elm(nni,1)=(mmax+1)+i;
   qq(hh,1)=(mmax+1)+i;

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

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


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

if itri==nitri
disp('left edge nodes')
nni=1;gg=1;
for i=0:(n-2)
   gg=gg+1;
   nni=nni+(n-i)+1;
   elm(nni,1)=edgen1n2(gg,1);
```

```
end
%pp(n+1,1)=n3(itri,1);
%edgen1n3(1:n+1,itri)=pp
end%if itri==nitri
if itri==nitri
lmax=max(max(edgen2n3(1:n+1,itri)));
end%if itri==nitri



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

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

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

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

end% if rem(n,2)==0
ne

nnd=max(max(eln))
if (n>3)
for kkk=1+(itri-1)*numtri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
end
end
if n==2
 for kkk=itri:ne
   nnd=nnd+1;
   eln(kkk,7)=nnd;
 end
end
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
   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);
   %
   nd=nd+1;nodes(mmm,13)=nd;
   nd=nd+1;nodes(mmm,14)=nd;
   nd=nd+1;nodes(mmm,15)=nd;
   nd=nd+1;nodes(mmm,16)=nd;


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)])
```

[17]D2PoissonEquationQ12MoinEx_MeshgridContourNew.m

```
function[]=D2PoissonEquationQ12MoinEx_MeshgridContourNew(n1,n2,n3,nmax,numtri,ndiv,mesh)
%ndiv=2,4,6,8,.....
%D2PoissonEquationQ12MoinEx_MeshgridContourNew([1;1;1;1;1;1;1;1],[2;3;4;5;6;7;8],[3;4;5;6;7;8;2],8,1,2,1)
%D2PoissonEquationQ12MoinEx_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
syms coord
%[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates(n1,n2,n3,nmax,numtri,ndiv,mesh)
```

```
%nnel=4;
[coord,gcoord,nodes,nodetel,nnode,nel]=polygonal_domain_coordinates_3rd_order(n1,n2,n3,nmax,numtri,ndiv,mesh)
nnel=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
   %boundary conditions-2
    for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (ynn==0)&((xnn>=0)&(xnn<=1))
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
   end
   %boundary conditions-3
   for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (ynn==1)&((xnn>=0)&(xnn<=1/2))
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
   end
  %boundary conditions-4
  for nn=1:nnode
     xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (xnn==1)&((ynn>=0)&(ynn<=1/2))
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
   end
 %boundary conditions-5
 for nn=1:nnode
    xnn=coord(nn,1);ynn=coord(nn,2);
     if ((xnn+ynn)==3/2)
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=double((sin(pi*xnn))*(sin(pi*ynn)))
      end
 end
```

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


end
   bcdof
   mm=length(bcdof);

format long g
%analytical solution

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

```
[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 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);

 xe(1,1)=(xa+xb+xc)/3;
 xe(2,1)=(xa+xc)/2;
 xe(3,1)=xa;
 xe(4,1)=(xa+xb)/2;
 %
 ye(1,1)=(ya+yb+yc)/3;
 ye(2,1)=(ya+yc)/2;
 ye(3,1)=ya;
 ye(4,1)=(ya+yb)/2;
 %
 [sp,wt]=glsampleptsweights(ng);
  xe1=xe(1,1);xe2=xe(2,1);xe3=xe(3,1);xe4=xe(4,1);
 ye1=ye(1,1);ye2=ye(2,1);ye3=ye(3,1);ye4=ye(4,1);
 f(1:12,1)=zeros(12,1)
 for i=1:ng
    si=sp(i,1);wi=wt(i,1);
    for j=1:ng
        sj=sp(j,1);wj=wt(j,1);
n1ij=((1-si)*(1-sj)*(-10+9*(si^2+sj^2)))/32;
n2ij=((1+si)*(1-sj)*(-10+9*(si^2+sj^2)))/32;
n3ij=((1+si)*(1+sj)*(-10+9*(si^2+sj^2)))/32;
n4ij=((1-si)*(1+sj)*(-10+9*(si^2+sj^2)))/32;
n5ij=(9/32)*((1-sj)*(1-si^2)*(1-3*si));
n6ij=(9/32)*((1-sj)*(1-si^2)*(1+3*si));
n7ij=(9/32)*((1+si)*(1-sj^2)*(1-3*sj));
n8ij=(9/32)*((1+si)*(1-sj^2)*(1+3*sj));
n9ij=(9/32)*((1+sj)*(1-si^2)*(1+3*si));
n10ij=(9/32)*((1+sj)*(1-si^2)*(1-3*si));
n11ij=(9/32)*((1-si)*(1-sj^2)*(1+3*sj));
n12ij=(9/32)*((1-si)*(1-sj^2)*(1-3*sj));

%
        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;
         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);
 for j=1:edof
     jj=index(j,1);
     ss(ii,jj)=ss(ii,jj)+sk(i,j);
 end
end
end%for iel
mm=length(bcdof);
sdof=size(ss);
%
for i=1:mm
c=bcdof(i,1);
for j=1:sdof
ss(c,j)=0;
end
%
ss(c,c)=1;
ff(c,1)=bcval(i,1);
end
%solve the equations
```

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

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

disp([NN phi xi])
disp('_____________________________________________________________________________')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
nodes
gcoord
[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);
%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;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,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] = newtonmethod4spquadrileteral(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
  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])
sst1='(MESH HAS '
sst2=num2str(nnode)
sst3=' NODES'
sst4=' AND '
sst5=num2str(nel)
sst6=' ELEMENTS)'
text(0.25,-.08,[sst1 sst2 sst3 sst4 sst5 sst6])
figure(2)
 x=[0.0 1.0 1.0 0.5 0.0];
 y=[0.0 0.0 0.5 1.0 1.0];
 patch(x,y,'w')
```

```
hold on
[x,y]=meshgrid(0:.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);
     exactsoln(mm,1)=z(i,j);
  end
end

end%switch mesh
[femsoln exactsoln]

 disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
[1 phi(1,1) xi(1,1)]
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;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;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)
nnel=16;
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
   nnn=0;
   for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (xnn==0)&((ynn>=0)&(ynn<=1))
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
   end
   %boundary conditions-2
    for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
      if (ynn==0)&((xnn>=0)&(xnn<=1))
         nnn=nnn+1;
         bcdof(nnn,1)=nn;
         bcval(nnn,1)=0;
      end
   end
```

```
    %boundary conditions-3
    for nn=1:nnode
        xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (ynn==1)&((xnn>=0)&(xnn<=1/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
   %boundary conditions-4
   for nn=1:nnode
      xnn=gcoord(nn,1);ynn=gcoord(nn,2);
        if (xnn==1)&((ynn>=0)&(ynn<=1/2))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
  %boundary conditions-5
  for nn=1:nnode
      xnn=coord(nn,1);ynn=coord(nn,2);
        if ((xnn+ynn)==3/2)
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=double((sin(pi*xnn))*(sin(pi*ynn)))
        end
  end
case 2
      nnn=0;
    for nn=1:nnode
        xnn=coord(nn,1);ynn=gcoord(nn,2);
        if (xnn==0)&((ynn>=0)&(ynn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-2
     for nn=1:nnode
        xnn=gcoord(nn,1);ynn=coord(nn,2);
        if (ynn==0)&((xnn>=0)&(xnn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
    %boundary conditions-3
    for nn=1:nnode
        xnn=gcoord(nn,1);ynn=coord(nn,2);
        if (ynn==1)&((xnn>=0)&(xnn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
   %boundary conditions-4
   for nn=1:nnode
      xnn=coord(nn,1);ynn=gcoord(nn,2);
        if (xnn==1)&((ynn>=0)&(ynn<=1))
            nnn=nnn+1;
            bcdof(nnn,1)=nn;
            bcval(nnn,1)=0;
        end
    end
```

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

format long g
%analytical solution

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



 for L=1:nel
  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 btb;gma 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=xe(1,1);xe2=xe(2,1);xe3=xe(3,1);xe4=xe(4,1);
ye1=ye(1,1);ye2=ye(2,1);ye3=ye(3,1);ye4=ye(4,1);
f(1:16,1)=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;

   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;
%
n1ij =h1i*h1j;n5ij =h2i*h1j;n6ij =h3i*h1j;n2ij =h4i*h1j;
n12ij=h1i*h2j;n13ij=h2i*h2j;n14ij=h3i*h2j;n7ij=h4i*h2j;
n11ij=h1i*h3j;n16ij=h2i*h3j;n15ij=h3i*h3j;n8ij=h4i*h3j;
n4ij=h1i*h4j;n10ij=h2i*h4j;n9ij=h3i*h4j;n3ij=h4i*h4j;

       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;
```

```
    end
 end
f=(delabc)*f;

edof=nnel*ndof;
k=0;
for i=1:nnel
   nd(i,1)=nodes(iel,i);
   start=(nd(i,1)-1)*ndof;
   for j=1:ndof
      k=k+1;
      index(k,1)=start+j;
   end
end
%-------------------------------------------------------------------------
for i=1:edof
   ii=index(i,1);
   ff(ii,1)=ff(ii,1)+f(i,1);
for j=1:edof
    jj=index(j,1);
    ss(ii,jj)=ss(ii,jj)+sk(i,j);
end
end
end%for iel
mm=length(bcdof);
sdof=size(ss);
%
for i=1:mm
c=bcdof(i,1);
for j=1:sdof
ss(c,j)=0;
end
%
ss(c,c)=1;
ff(c,1)=bcval(i,1);
end
%solve the equations

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

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

disp([NN phi xi])
disp('_____________________________________________________________________________')

disp('number of nodes,elements & nodes per element')
[nnode nel nnel ndof]
nodes
gcoord
[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);
%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;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-1/3)/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-1/3)/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*ph
i(nd14,1)+shn15*phi(nd15,1)+shn16*phi(nd16,1);
           break
         end%if (in==1)
       end%for iel
   %THE PROGRAM EXECUTION JUMPS TO HERE if (in==1)
    end%for j
end%for i
  z=sin(pi*x).*sin(pi*y);

for i=1:21
    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 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)
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 0.5 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
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)
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 exactsoln]
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 nnel ndof]
[1 phi(1,1) xi(1,1)]
```

[19]newtonmethod4spquadrileteral.m

```
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 = del(f1,f2,...,fn)/del(x1,x2,...,xn)
% x = [x1;x2;...;xn], f = [f1;f2;...;fn]
% x0 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 = t0; % 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
 ftn= feval(f,tn)-[p;q];
 t=tn;
 iter = 100-N;
 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]