Development of Eight Node Isoparametric Quadrilateral (QUAD8) Elements in Java

I.E Umeonyiagu[1], N.P Ogbonna[2]

[1]Department of Civil Engineering, Chukwuemeka Odumegwu Ojukwu University, Uli. Nigeria

[2]Department of Civil Engineering, Chukwuemeka Odumegwu Ojukwu University, Uli. Nigeria

Email: nopsoftinc@yahoo.com

ABSTRACT

This study developed of eight node Isoparametric Quadrilateral (QUAD8) elements using an object oriented approach and the Java programming language. A review of the work of Kansara (2004) and Nikishkov (2010) was conducted to develop a finite element program using Java programming language which addresses the deficiencies of four node Isoparametric (QUAD4) elements developed by Kansara (2004). Input to the program is done through a text file and the loads that can be applied to the structure include concentrated loads, and uniformly distributed surface loads. Various load combinations can also be used. The program computes displacements and stresses at each node of the finite element model. Several test examples were analyzed using the program and results were compared with those obtained from Kansara (2004), the commercial finite element analysis program SAP 2000 and LISA respectively. Results were compared at the points of maximum displacements and stresses. The average stress was taken in to consideration to calculate stresses at specific point. The results obtained from the analysis of the example problems were found to be very accurate when compared to those obtained from the Kansara (2004), SAP 2000 and LISA. The difference in displacements computed by the two programs was very negligible. The difference in stresses was also quite close.

Keywords *- Finite element model, Java programming language, Object oriented approach, Isoparametric Quadrilateral elements.*

List of Symbols *- N_i are Shape functions , β_i is displacement field, N_i is shape functions, $\varepsilon_x, \varepsilon_{Y,} \gamma_{xy}$ The element strain field, $\{p_\sigma\}$ is Equivalent stress vector, η_j are abscissas, w_i are weighting coefficient, L is Length, H is Depth, t is Thickness , E is Modulus of elasticity n is Poisson's ratio, $\sigma_x, \sigma_y, \sigma_x, \tau_z, \tau_{yz}, \tau_{zx}$ are the state of stress at any point $\epsilon_x, \epsilon_y, \epsilon_{z,} \gamma_{xy}, \gamma_{yz}, \gamma_{zx}$ are the components of the strain α is Coefficient of thermal expansion, T is Change in tempreture , ψ is the non-zero*

I. INTRODUCTION

Since the evolution of the term finite element by Clough in 1951, there have been significant developments in finite element method. A large number of different finite elements have been developed (Kansara, 2004).

McNeal (1978) developed the quadrilateral shell element QUAD4, by considering two inplane displacements that represent membrane properties and one out-of-plane displacement and two rotations, which represents the bending properties. McNeal (1978) included modifications in terms of a reduced order integration scheme for shear terms. He also included curvature and transverse shear flexibility to deal with the deficiency in the bending strain energy. The simplest method adopted to remove the rotational singularity is to add a fictitious rotational stiffness. However, Yang et al. (2000) suggested that, although the method solves the problem of singularity it creates a convergence problem that sometimes leads to poor results. Recent developments include using membrane elements with rotational degrees of freedom to develop an efficient flat shell element (Kansara, 2004). Several methods have been suggested by various authors for removing the singularity in the stiffness matrix based on variational principles such as those formulated by Gruttmann et al (1992). A degenerated shell element with drilling degrees of freedom was developed recently by Djermane *et al.* (2006) for application in linear and nonlinear analysis of thin shell structures for isotropic or anisotropic materials with using the assumed natural strains technique to alleviate locking phenomenon. Djermane et al. (2007) also extended the formulation by using the same techniques to study the dynamic responses of thick and thin nonlinear shells. Kanok–Kanukulchai (1979) devised utilization of the penalty

approach to remove the rotational singularity caused by the addition of a fictitious rotational stiffness. He introduced a constraint equation which "links" the drilling rotations in the fiber coordinate system to the in-plane twisting mode of the mid-surface.
Adam et al (2013) developed a bilinear degenerated four nodes shell elements with drilling degrees of freedom. They examined the element performance with respect to sensitivity to the value of penalty parameter and to evaluate the suitable value. Forte et al (1990) in one of the first publications on the object oriented approach to the finite element development, presented essential finite element classes such as elements, nodes, displacement and force boundary conditions, vectors and matrices. Several authors described detailed finite element architecture using the object oriented approach. Nikiskov (2010) developed procedures for programming finite element in Java. He was able to demonstrate on how to solve finite element problems for solid mechanics as well as Heat transfer problems.
Kansara (2004) developed membrane, plate and flat shell element in Java Programming language. He created a finite element analysis program using Java Programming Language to check the accuracy of the developed elements. Kansara (2004) while testing the output of his Java code with a commercial program SAP 2000 noted that for the test examples of a cantilever I-beam and a folded plate structure, the stresses from the program differed considerably from those obtained from SAP 2000. This was due to the fact that his Java program lacked the ability to solve flat shell problems with rotational degrees of freedom (also known as "drilling degree of freedom"). He advocated for a future research work that can handle most of the deficiencies inherent in his work. These were the inability to developing flat shell elements in which the membrane elements have rotational (drilling) degrees of freedom, the use of a band storage scheme for storing the structure stiffness matrix and band solvers in the program to solve large finite element analysis problems, absence of a graphical user interface in the program and, extending the application of the program to include other elements such as truss, and frame elements.
This work improved on the work of Kansara (2004) using the procedures prescribed by Nikishkov (2010). Eight node Isoparametric Elements (QUAD8) were developed and a finite element analysis program was also written with Java programming language to check the accuracy of the developed elements. Several test structures will be analyzed using the Java program and the results compared with those from other standard test validation examples.

II. LITERATURE REVIEW

A. *Quadratic Quadrilateral (QUAD8)*

The QUAD8 element is shown in Fig 2.1.
In terms of generalized coordinates β_i its displacement field is

$$\left.\begin{array}{c} u = \beta_1 + \beta_2 x + B_3 y + \beta_4 x^2 + \beta_5 xy + \beta_6 y^2 + \beta_7 x^2 y + \beta_8 xy^2 \\ v = \beta_9 + \beta_{10} x + B_{11} y + \beta_{12} x^2 + \beta_{13} xy + \beta_{14} y^2 + \beta_{15} x^2 y + \beta_{16} xy^2 \end{array}\right\} \quad (2.1)$$

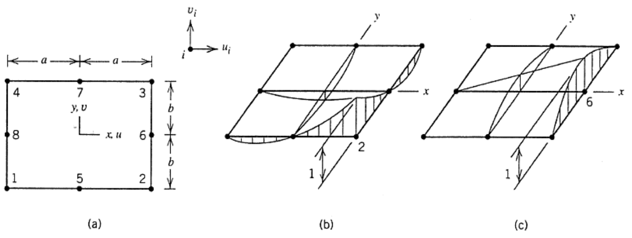


Figure 2.1: (a) A quadratic quadrilateral. (b) Shape function for N_2. (c) Shape function for N_6.

In more compact form, the displacement field in terms of shape functions N_i is

$$\left.\begin{array}{l} u = \sum N_i u_i \\ v = \sum N_i v_i \end{array}\right\} \quad (2.2)$$

Where index i runs from1 to 8, which explains the "8" in the name QUAD8. As examples, two of the eight shape functions are

$$\left.\begin{array}{c} N_2 = \frac{1}{4}(1+\xi)(1-\eta) - \frac{1}{4}(1-\xi^2)(1-\eta) - \frac{1}{4}(1+\xi)(1-\eta^2) \\ N_2 = \frac{1}{2}(1+\xi)(1-\eta^2) \end{array}\right\} \quad (2.3)$$

Where $\xi = x/a$ and $\eta = \frac{y}{b}$
By Looking at a typical edge, for example, the edge $x = a$, we see that the displacements are quadratic in y, which means that the edge deforms into a parabola when a single degree of freedom on that edge is nonzero.
The element strain field is

$$\left.\begin{array}{r} \varepsilon_x = \beta_2 + 2\beta_4 x + B_5 y + 2\beta_7 xy + \beta_8 x^2 \\ \varepsilon_y = \beta_{11} + \beta_{13} x + 2B_{14} y + \beta_{15} x^2 + 2\beta_{16} xy \\ \gamma_{xy} = (\beta_3 + \beta_{10}) + (\beta_5 + 2\beta_{12})x + (2\beta_6 + \beta_{13})y + \beta_7 x^2 + 2(2\beta_8 + \beta_{15})xy + \beta_{16} y^2 \end{array}\right\} \quad (2.4)$$

Each of the three strains contains all linear terms and some quadratic terms (e.g., there is no x^2 terms in the ε_x expression). The QUAD8 element can represent exactly all states of constant strain, and states of pure bending if it is rectangular. None rectangular shapes are allowed as well.

III. MATERIALS AND METHOD

A. Creation of Robot Finite Element Analysis Program (Rfea)

The required input to Rfea is in the form of a text file and the results from the program saved in an output file in text format. A simple GUI was used during visualization of finite element models and results. Rfea performs three tasks of the finite element analysis: preprocessing (finite element model generation), processing (problem solution); and postprocessing (results calculation and visualization). Rfea finite element system is organized into eight class packages. Using a Unified Modelling Language (UML) diagram, the various packages and their individual classes are presented as follows:

1) *Package fea*

This package shall contain the main classes which include FE, JFEM, JMGEN and JVIS.

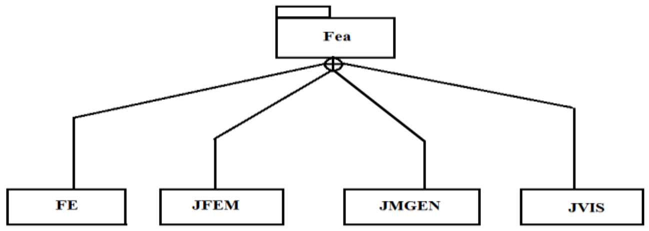


Figure 3.1: Package fea showing its member classes

2) *Package model*

This package shall contain finite element model and loading classes which are illustrated on the UML diagram in Fig 3.2.

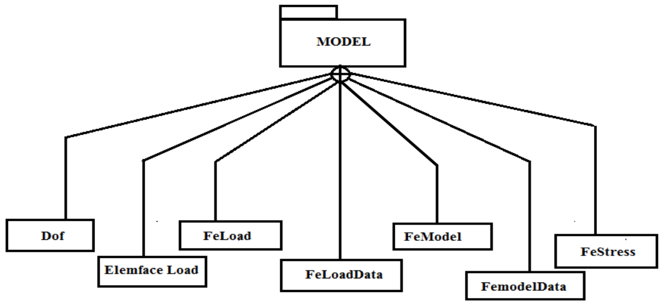


Figure 3.2: Package Model showing its member classes

3) *Package util*

Utility classes which include the FePrintWriter, FeScanner, GaussRule, UTIL

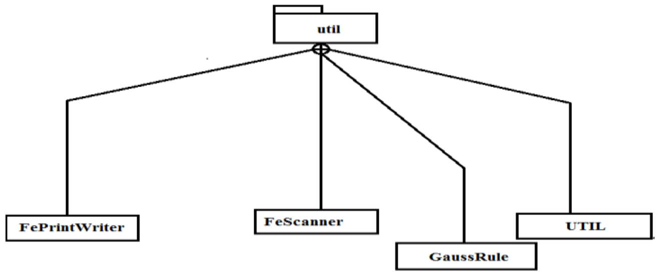


Figure 3.3: Package util with its member classes

4) *Package elem*

This package contains classes such ElementShellD, ShapeShellD.

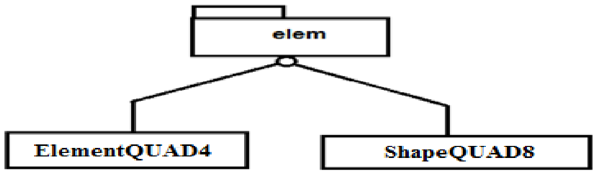


Figure 3.4: Package elem with its member classes

5) *Package material*

This package contains the constitutive relations for materials.

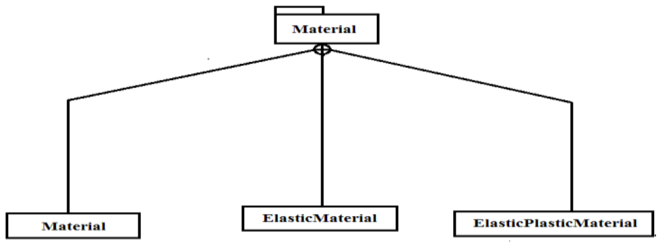


Figure 3.5: Package material with its member classes

6) *Package solver*

Classes for the Assembly and solution of global finite element equation systems:

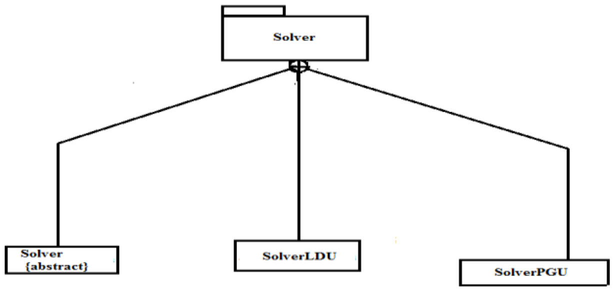


Figure 3.6: Package solver with its member classes

7) Package gener

The classes of this package will be used for generation of mesh and are represented with a UML diagram shown in figure 3.7.

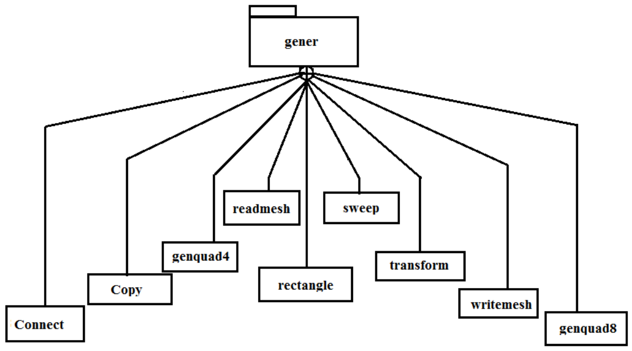


Figure 3.7: Package gener with its member classes

8) Package visual

Contains classes visualization of models and results as shown in Fig 3.7

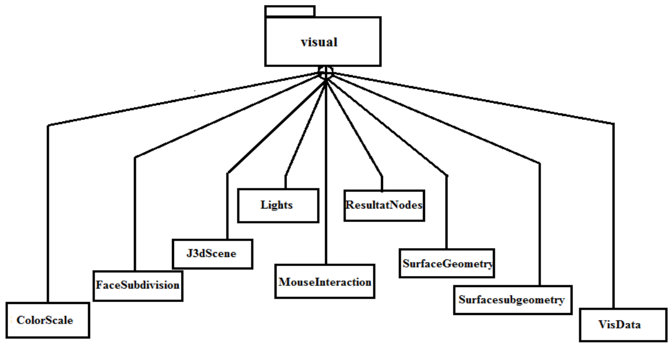


Figure 3.8 Package visual with its member classes

The new Java Programming classes developed to modify Kansara (2004) work using Nikishkov's approach will be explained in the successive sections of this thesis work

B. Development of Eight Node Isoparametric Quadrilateral Elements (QUAD8)

1) Formulation of Shape functions for interpolation of unknown fields and for description of element shape

Shape functions for the quadratic isoparametric element with eight nodes are given by:

$$\left.\begin{aligned} N_i &= \frac{1}{4}(1+\xi_0)(1+\eta_0) - \frac{1}{4}(1-\xi^2)(1+\eta_0) - \frac{1}{4}(1+\xi_0)(1-\eta^2), \quad i = 1,3,5,7, \\ N_i &= \frac{1}{2}(1-\xi^2)(1+\eta_0), \quad i = 2,6, \\ N_i &= \frac{1}{2}(1+\xi_0)(1+\eta^2), \quad i = 4,8, \end{aligned}\right\} \quad (3.1)$$

where $\xi_0 = \xi\,\xi_i,\ \eta_0 = \eta\eta_i$ *and* ξ_i, η_i are nodal values of local coordinates ξ, η .

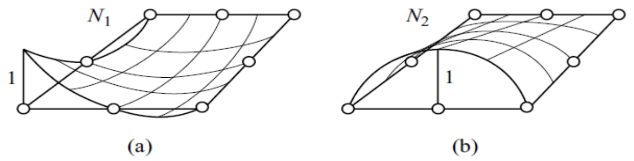


Figure 3.9: Shape functions of a corner node N_1 (a) and of a midside node N_2 (b) for a quadratic finite element (Nikishkov, 2010)

Performing element degeneration which makes it possible to create finite element meshes, requires modifying three shape functions for nodes located at a side opposite to the degenerated side. In the case of degeneration with coincident nodes 1, 2 and 3 the shape functions N_5, N_6 and N_7 are as follows:

$$\left.\begin{aligned} N'_5 &= N_5 + \Delta, \\ N'_6 &= N_6 + 2\Delta, \\ N'_7 &= N_7 + \Delta, \\ \Delta &= \frac{1}{8}(1-\xi^2)(1-\eta^2) \end{aligned}\right\} \quad (3.2)$$

2) Computation of strains $\{\boldsymbol{\varepsilon}\}$ at any point inside the element using displacement differentiation matrix $[\boldsymbol{B}(\boldsymbol{\xi},\boldsymbol{\eta})]$

Computing strains $\{\varepsilon\}$ at any point inside the element using a vector of nodal displacements $\{q\}$:

$$\{\varepsilon\} = [B]\{q\} \quad (3.3)$$

Matrix $[B]$ can be presented in a block form

$$[B] = [B1\ B2\ldots]. \quad (3.4)$$

Each block corresponds to displacements of one node. Next is the transformation of coordinates from ξ, η, to x, y using chain rule of partial differentiation:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{Bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \quad (3.5)$$

Where $[J]$ is the Jacobian matrix. The derivatives with respect to the global coordinates are computed with the use of the inverse of the Jacobian matrix:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \quad (3.6)$$

The components of the Jacobian matrix are calculated using derivatives of shape functions *Ni* with respect to the local coordinates ξ, η and global coordinates of element nodes x_i, y_i:

$$\left.\begin{aligned}\frac{\partial x}{\partial \xi} = \sum \frac{\partial N_i}{\partial \xi} x_i , \frac{\partial x}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} x_i \\ \frac{\partial y}{\partial \xi} = \sum \frac{\partial N_i}{\partial \xi} y_i , \frac{\partial y}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} y_i\end{aligned}\right\} \tag{3.7}$$

The determinant of the Jacobian matrix $|J|$ is used for the transformation of integrals from the global coordinate system to the local coordinate system. Assuming unit thickness in plane problems, it is possible to represent an elementary volume as:

$$dV = dxdy = |J| d\xi d\eta$$

3) *Formulating Element Matrices and Vectors*

Element matrices and vectors are calculated as follows:

$$Stiffness\ matrix\ [K] = \int_{-1}^{1} \int_{-1}^{1} [B]^T [E][B] t |J| d\xi d\eta \tag{3.8}$$

$[B]$ = the displacement differentiation matrix

Force vector (surface load) is given as

$$\{p\} = \int_{-1}^{1} [N]^T \{P^S\} t \frac{ds}{d\xi} d\xi, \tag{3.9}$$

Where is $\{P^S\}$ *is* the vector of surface forces
Equivalent stress vector (with negative sign) is

$$\{p_\sigma\} = -\int_{-1}^{1} \int_{-1}^{1} [B]^T \{\sigma\} t |J|\, d\xi d\eta \tag{3.10}$$

Integration of expressions for stiffness matrices and load vectors cannot be performed analytically for the general case of isoparametric elements. Instead, stiffness matrices and load vectors are evaluated numerically using Gauss quadrature over quadrilateral regions. The Gauss quadrature formula for the volume integral in the two-dimensional case is of the form (Nikishkov, 2010):

$$I = \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta) d\xi d\eta = \sum_{i=1}^{n} \sum_{j=1}^{n} f(\xi_i \eta_j)\, w_i w_j$$
$$= \sum_{k=1}^{N} f(\xi_k, \eta_k) W_k \tag{3.11}$$

where ξ_i, η_j are abscissas, w_i are weighting coefficients of the one-dimensional Gauss integration rule, $N = n \times n$ and W_k are products of pairs of one-dimensional integration weights.

4) *Calculation of Strains and Stresses*

Calculation of strains at any point of an element using Cauchy relations

$$\{\varepsilon\} = \{\varepsilon_x \quad \varepsilon_y \quad \gamma_{xy}\} = \left\{\frac{\partial u}{\partial x} \quad \frac{\partial v}{\partial y} \quad \frac{\partial v}{\partial x} \quad \frac{\partial u}{\partial y}\right\} \tag{3.12}$$

For quadratic elements with eight nodes, strains and stresses have the best precision at 2×2 integration points with local coordinates $\xi, \eta = \pm 1/\sqrt{3}$.

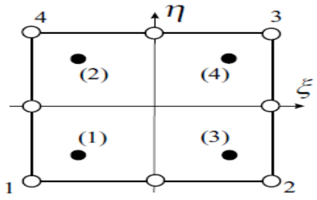


Figure 3.10: Numbering of integration points and vertices for the eight-node isoparametric element (Nikishkov 2010)

Consider a quadratic element in the local coordinate system ξ, η as shown in Figure 3.10 where integration points are numbered as (1)–(4). The order of integration points is determined using class GaussRule. Corner nodes are numbered 1–4 in anticlockwise order.

C. Implementation of Eight node isoparametric quadrilateral elements (QUAD8)

1) *Class ElementQuad8*

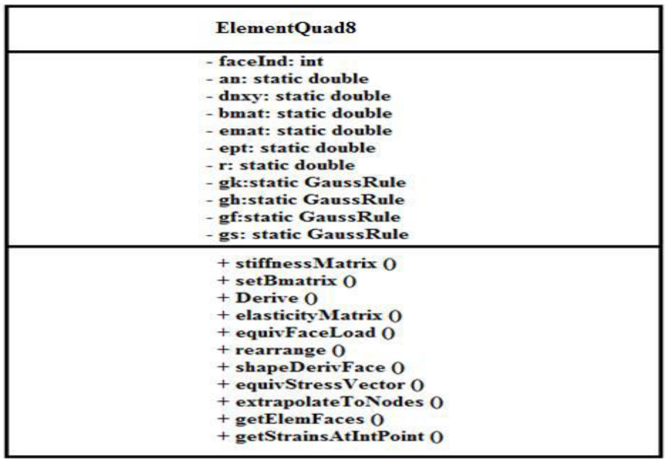
Class ElementQuad8 will extend class Element and to implement methods for computing element matrices and vectors. The following arrays are declared:
an – shape functions;
dnxy – derivatives of shape functions with respect to global coordinates *x*, *y* (first index is related to node number, second index to *x* and *y*);
bmat – displacement differentiation matrix, emat – elasticity matrix,
ept – vector of thermal strains.
GaussRule objects for integration of the element stiffness matrix, thermal vector, surface load, and equivalent stress vector will be created as follows:

```
private static GaussRule gk = new GaussRule(3,2);
private static GaussRule gh = new GaussRule(3,2);
private static GaussRule gf = new GaussRule(3,1);
private static GaussRule gs = new GaussRule(2,2);
```

Constructor ElementQuad8(){}calls the constructor of parent class Element and passes to it the element name quad8, the number of element nodes (8) and the number of points for storing stresses and strains. Method stiffnessMatrix() will perform the computation of the element stiffness matrix according to Equation (3.8) and the stiffness matrix kmat is set to zero as follows.
Integer variable ld represents a length of the strain or stress vector. For plane problems ld is equal to 3. Extraction of Material object mat will be done from the hash table materials using the material name. The elasticity matrix emat is set and then numerical integration of the element stiffness matrix will be performed in a loop with a parameter ip denoting integration point number and also performed inside a single loop.

```
// Gauss integration loop
for (int ip = 0; ip < gk.nIntPoints; ip++) {
```

The method setBmatrix () sets the displacement differentiation matrix bmat and returns a determinant of the Jacobian matrix det. Variable dv includes an integration weight and a circle length in the case of the axisymmetric problem and the loop parameter j starts from i, which is a parameter of the outer loop.
Other important methods and their description are given in Fig 3.11


ElementQuad8

- faceInd: int
- an: static double
- dnxy: static double
- bmat: static double
- emat: static double
- ept: static double
- r: static double
- gk:static GaussRule
- gh:static GaussRule
- gf:static GaussRule
- gs: static GaussRule

+ stiffnessMatrix ()
+ setBmatrix ()
+ Derive ()
+ elasticityMatrix ()
+ equivFaceLoad ()
+ rearrange ()
+ shapeDerivFace ()
+ equivStressVector ()
+ extrapolateToNodes ()
+ getElemFaces ()
+ getStrainsAtIntPoint ()

Figure 3.11: UML diagram of ElementQuad8 class with its attributes and operations

2) Class ShapeQuad8

Class ShapeQuad8 is placed in package elem. It is created to calculate the shape functions for 8 node quadratic isoparametric elements. Element nodes are numbered in an anticlockwise direction starting from any corner node. The local numbers of nodes are shown in Fig 3.12

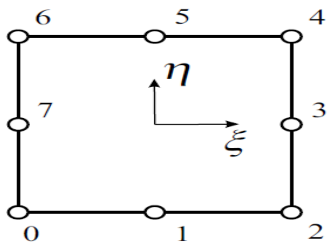


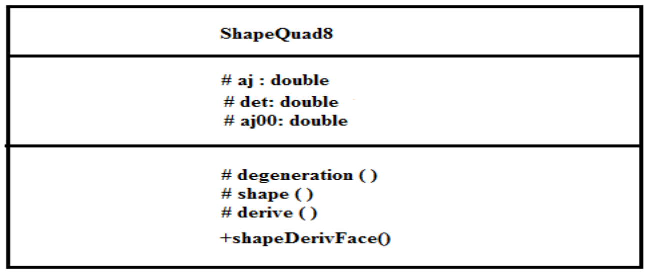
Figure 3.12: Local numbering of nodes for 8 nodes isoparametric element (Nikishkov, 2010)

The first step is to create a method responsible for discovering degenerate quadratic elements as shown in the code fragment bellow

```
static int degeneration(int[] ind) {
 int deg = 0;
 for (int i = 0; i < 7; i += 2) {
 if (ind[i] == ind[i + 1]) {
 deg = (i + 5) % 8;
 break;
}}
 return deg;}
```

Method degeneration checks if a corner connectivity number is equal to the next midside node. If so, then the method returns the local number of a midside node opposite to the degenerated side.
Method shape() computes element shape functions an for specified local coordinates xi (ξ) and et (η). Connectivity numbers ind are used as information on the existence of midside nodes.
Derivatives of shape functions with respect to global coordinates dnxy are given by method deriv. The method parameters are:

xi, et – local coordinates ξ and η ,
ind – element connectivities,
xy – array of nodal coordinates.


ShapeQuad8

aj : double
det: double
aj00: double

degeneration ()
shape ()
# derive ()
+shapeDerivFace()

Figure 3.13: UML diagram of ShapeQuad8 class with its attributes and operations

Calculation of the Jacobian matrix aj according to Equation (3.6) is done in as shown bellow

```
// Jacobian matrix
double[][] aj = new double[2][2];
for (int j = 0; j < 2; j++) {
for (int i = 0; i < 2; i++) {
aj[i][j] = 0.0;
for (int k = 0; k < 8; k++)
aj[i][j] += dnxe[k][j]*xy[k][i];
}}
```

The Jacobian matrix inverse is performed as follows

```
// Jacobian inverse
double aj00 = aj[1][1]/det;
aj[1][1] = aj[0][0]/det;
aj[0][0] = aj00;
aj[1][0] = -aj[1][0]/det;
aj[0][1] = -aj[0][1]/det;
```

Derivatives of shape functions with respect to global coordinates x, y are obtained by multiplication of the Jacobian matrix and derivatives with respect to local coordinates ξ , η

Method shapeDerivFace() will calculate three shape functions an and their derivatives dndxi with respect to the local coordinate ξ.

IV. NUMERICAL ANALYSIS

A. Verification of Eight Node Quadrilateral Plane Element (QUAD8)

1) Finite element Analysis of FE Model for a cantilever beam using three-Eight node quadrilateral plane elements

The finite element model of this cantilever beam is generated using three eight node quadrilateral plane elements as shown in figure 4.1

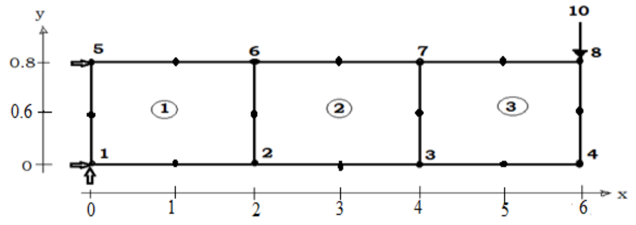


Figure 4.1: FE Model for a cantilever beam using three-eight node quadrilateral plane elements

Geometric Data: Length L = 6.0 m, Depth h = 0.8 m, Thickness t = 0.2 m.

Material Properties: Modulus of elasticity E = 30000 kN/m^2, Poisson's ratio η = 0.3

Boundary Conditions: Restraints are provided in the x and y directions at the left end of the cantilever.

Loading: A concentrated load of 10 kN is applied to the free end (node 8) of the cantilever.

Comparison of Results: Table 4.4 represents the results for displacements at nodes 8 and 3, and stresses at node 1 obtained from the program and LISA. The comparisons shown in the table suggest that the displacements and stresses obtained from the program are in good agreement with those obtained from LISA.

Table 4.1: Displacements and Stresses for a cantilever beam using three-Eight node quadrilateral plane elements

Location		**Program Rfea**	**Kansara (2004)**	**Difference (%)**
NODE 8	UX	0.021416	0.0215391	-0.57480389
	UY	-0.159369	-0.1572759	1.269417476
NODE 3	UX	-0.008919	-0.0088499	0.473867596
	UY	-0.068399	-0.068969	-0.83334552
NODE 1	S11	-149.06457	-146.96001	1.4117932
	S22	-46.61617	-44.041980	5.4308947
	S12	-144.65257	-141.11712	2.4440059

2) Finite element Analysis of a cantilever beam using Twenty- Eight node quadrilateral plane elements

The beam in Fig4.5 is modelled using 16 eight node quadrilateral plane elements.

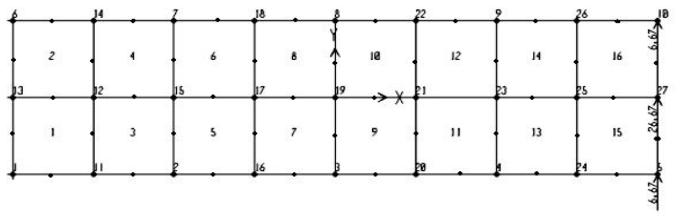


Figure 4.2 FE Model for a cantilever beam using Twenty- Eight node quadrilateral plane elements

Geometric Data: Length L = 48.0 m, Depth h = 12 m, Thickness t = 1 m

Material Properties: Modulus of elasticity E = 30000 kN/m^2., Poisson's ratio n = 0.3

Boundary Conditions: Restraints are provided at the left end (nodes 1, 13, and 6) of the cantilever.

Loading: A concentrated load $P_1 = 6.67$ kN is applied at node 10. A concentrated load $P_2 = 26.67$ kN is applied at node 27. A concentrated load $P_3 = 6.67$ kN is applied at node 5.

Table 4.2 Displacements and Stresses for a cantilever beam using Twenty- Eight node quadrilateral plane elements

Location		Program Rfea	LISA	Difference (%)
NODE 10	UX	-0.05184	-0.052061	-0.42631172
	UY	0.323328	0.322829	0.154332443
NODE 8	UX	-0.03654	-0.037349	-2.21401204
	UY	0.106799	0.106683	0.108615249
NODE 7	UX	0.0019524	0.0019387	0.701700
	UY	0.00078431	0.00078394	0.04717522
NODE 3	S11	-70.29888	-70.294707	0.005936083
	S22	-17.570795	-17.569609	0.006749837
	S12	-18.413922	18.413179	0.00403499

3) Finite Element Analysis of a Plate with a Semi-circular Hole using Fifty Two-Eight Nodes Quadrilateral plane Element

Fig 4.3 shows a plate with a semicircular hole. The plate is modeled using 52 eight node quadrilateral plane elements. The finite element model for the plate is shown in Fig 4.3.

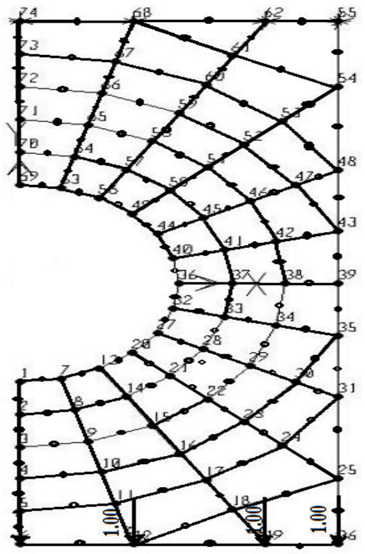


Figure 4.3: FE Model for a Plate with Semi Circular Hole.

Geometric Data: Length $L = 16.0$ m, Width $b = 16$ m, Thickness $t = 0.45$ m

Material Properties: Modulus of elasticity E = 30000 kN/m^2, Poisson's ratio $\eta = 0.3$

Boundary Conditions: Restraints are provided at top of the plate (i.e., nodes 55, 62, 68, and 74)

Loading: Concentrated loads of 1.0 kN are applied at each node at the bottom of the plate. i.e., nodes 6, 12, 19 and 26.

Comparison of Results: Table 4.3 represents the comparison of displacements and stresses at integration points obtained at various nodes the developed program Rfea and from LISA. The displacements at nodes 1 and 6 are tabulated. The differences in results obtained from the two programs are less than 5 %. The stresses at nodes 32 and 36 are obtained from the Rfea are compared to those from Kansara (2004). The differences in stresses are in the range 0.35% to 5%. The element stresses are calculated at the element edge using the derivatives of the displacements. The stresses calculated at the edge of one element may differ significantly from the stresses calculated at the edge of an adjacent element. The difference may be significant if the generated finite element mesh is coarse. This difference can be reduced using a finer mesh. Therefore, stresses averaged at any single point may be higher or lower depending on mesh pattern.

Table 4.3 Displacements and Stresses for a Plate with a Semi-circular Hole using Fifty Two-Eight Nodes Quadrilateral plane Element

Location		**Programe Rfea**	**Kansara (2004)**	**Difference (%)**
NODE 1	**UZ**	0.00716	0.007374	-3.0607966
	UZ	0.00144	0.001576	-9.67292972
NODE 6	**UX**	0.01050	0.010807	-2.9434178
	UY	0.00133	0.001353	-1.42428786
NODE 32	**S11**	2.13546	2.126713	0.40979384
	S22	10.73744	10.77271	-0.3284674
	S12	1.62556	1.65910	-2.0634183
NODE 36	**S11**	2.011274	1.918493	4.61304626
	S22	11.992304	12.2935	-2.5115190

V. CONCLUSIONS

This research presented the development of eight node Isoparametric Quadrilateral elements using the object oriented programming concept in Java as an alternative to the traditional procedural programming approach. A finite element analysis program was developed to verify the accuracy of the results. Some test example problems were analyzed using the developed program. The results from these analyses were compared with those obtained from Kansara (2004), the commercial finite element analysis program SAP 2000 and LISA respectively in order to verify the accuracy of the developed program. The difference in displacements computed by the two programs was very negligible. The difference in stresses was also quite close. However, stresses in a few cases differed by 5 to 9 %. The computed stresses were also in agreement for most cases.

REFERENCES

[1] Adam, F. M. and Mohamed, A. E. (2013), Finite Element Analysis of Shell structures, LAP LAMBERT Academic Publishing.

[2] B. Irons and S. Ahmad,(1980) Techniques of Finite Elements. Ellis Horwood, Chichester, UK.

[3] Djermane, M., Chelghoum, A., Amieur, B. and Labbaci, B. (2006), Linear and Nonlinear Thin Shell Analysis Using A Mixed Finite Element with Drilling Degrees of Freedom,International Journal of Applied Engineering Research, Volume 1 Number 2 pp. 217-236

[4] Fathelrahman. M. Adam, Abdelrahman. E. Mohamed, A. E. Hassaballa (2013) Degenerated Four Nodes Shell Element with Drilling Degree of Freedom,, IOSR Journal of Engineering (IOSRJEN).

[5] Gallagher R. H., (1975) Finite Element Analysis Fundamentals, Prentice-Hall.

[6] G.P. Nikishkov, (2010) Programming Finite Elements in Java™, Springer-Verlag London Limited.

[7] Kansara, K. (2004), Development of Membrane, Plate and Flat Shell Elements in Java, M.sc. Thesis, Faculty of the Virginia Polytechnic Institute & State University, 2004.

[8] McNeal R. H., (1978) A Simple Quadrilateral Shell Element, Computers and Structures, Vol. 8, pp. 175-183.