Fall 2015

Multiple Point Constraint (MPC)-Based Variable Node Super-Element

Mohamad Eftekharjoo
Old Dominion University, mefte001@odu.edu

Recommended Citation
Eftekharjoo, Mohamad. "Multiple Point Constraint (MPC)-Based Variable Node Super-Element" (2015). Master of Science (MS), Thesis, Mechanical & Aerospace Engineering, Old Dominion University, DOI: 10.25777/q2qf-rw19
https://digitalcommons.odu.edu/mae_etds/307

MULTIPLE POINT CONSTRAINT (MPC)-BASED VARIABLE NODE SUPER-ELEMENT

by

Mohamad Eftekharjoo
B.S July 2011, Azad University, South Tehran Branch

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

MECHANICAL ENGINEERING

OLD DOMINION UNIVERSITY
December 2015

Approved by:

Gene J.W. Hou (Director)

Duc T. Nguyen (Member)

Xiaoyo Zhang (Member)

ABSTRACT

MULTIPLE POINT CONSTRAINT (MPC)-BASED VARIABLE NODE SUPER-ELEMENT

Mohamad Eftekharjoo
Old Dominion University, 2015
Director: Dr. Gene J.W. Hou

The multiple point constraint (MPC)-based variable node element is introduced in this study to handle mismatched meshes between sub-domains in finite element analysis. The MPC-variable node element is a collection of a group of elements. The compatibility condition along the interface boundary is imposed along the edge of these elements through Lagrange multipliers. The Elimination method is then used to remove the effects of the dependent nodes in these elements to produce a single MPC-based variable node element. The derived variable node elements are applied to solve two plane strain problems in order to validate the accuracy of the proposed MPC-based variable node element.

AKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Gene J.W. Hou for the continuous support of my graduate study and research and for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me during the researching and writing of this thesis. I could not have imagined having a better advisor and mentor for my graduate study.

Besides my advisor, I would like to thank the rest of my thesis committee. Thanks go to Dr. Xiaoyo Zhang and Dr. Duc T. Nguyen for their encouragement, insightful comments, and hard questions.

Last but not least, I would like to thank my family, my parents Reza Eftekharjoo and Farzaneh Torabi and my sister Samira Eftekharjoo, for supporting me spiritually throughout my life. I would like to thank all my friends and my best friend Mahdi Daei for helping me get through the difficult times and for all the emotional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

The complexity of engineering applications has increased the level of difficulty for finite element modeling. The irregularities in the geometry and in the material distribution in these applications usually require multi-scaled and multi-phase modeling. It will therefore be convenient in these cases to divide the analysis domain into sub-domains and discretize the sub-domains independently based upon their specific features and requirements. The challenge is then laying on the afterward integration process for finite element analysis. This study will focus on one aspect of such a challenge: integration of two independently discretized finite element meshes together for a united finite element analysis.

When two independently meshed finite element models are assembled together, gaps and overlaps may be present between their meshed boundaries as shown in Figure 2-1. The boundary elements may be divided and the boundary nodes may have to be relocated so as to merge the meshed boundaries to become one that is close to the common interface boundary. This process may generate variable node elements adjacent to the interface boundary. These variable node elements play a transient role in connecting one finite element in the foreign domain to the one in the home domain. Notice that the order of the shape functions and the number of numbers in these elements can be different. Therefore the compatibility between the variable node elements and the connected elements becomes a concern for investigation.

In 2002. Park et al., introduced the displacement along the interface boundary as a new state variable. The compatibility condition between the boundary displacement of each sub-domain and the newly introduced interface displacement was treated as an equality constraint which was enforced by the method of Lagrange multipliers. The newly introduced interface displacement and two sets of Lagrange multipliers were considered as unknown and discretized in accordance with the standard finite element procedure. The final system matrix equation will include the nodal values of the Lagrange multipliers and the interface displacement. The proposed approach was validated by analyzing a solid with non-matching interfaces. The method can be simplified by imposing the compatibility condition directly between the boundary displacements of the connected sub-domains without introducing the interface displacement as unknown. Panteno and Averill (2002) followed the same procedure introduced by Park et al. (2002), but replaced the method of Lagrange multipliers by the penalty function method. As a result, their proposed approach maintained the positive definite of the system equation, which can be easily incorporated into the commercial FEA codes. Their paper suggested ways to select the value of the penalty coefficients. Aminpour et al. (2001) directly treated the compatibility condition between the involved boundary displacements as linear multipoint constraints (MPC) which can be directly incorporated into the existing commercially available FEA codes.

Kim (2002) designated a buffer zone that covers the last layer of the discretized elements on both sides of the interface boundary. The interface elements were then developed to solve the unknown displacement in the buffer zone. These interface elements are the moving least square (MLS)-based meshless elements. Pseudo nodes were added along the common interface boundary as well as the boundary connecting the interface elements and the discretized elements in the parental sub-domain. The values at the pseudo nodes placed along the boundaries between the interface elements and the discretized elements of the parental sub-domains were interpolated based upon shape functions used in the parental sub-domain elements. Thus, no new degrees of freedom associated with these pseudo nodes were added to the system equation. The same could not apply to the pseudo nodes added along the common interface boundary. Their associated degrees of freedom will be part of the unknown in the assembled finite element equation. The objective of such an arrangement is to ensure that the number of nodes along these boundaries is the same within an interface element. This led to a convenient way to describe the weighting functions as bilinear polynomials in the interface element. The compact support domain was tailored to fit into the domain of the interface element with zero values outside and along its boundaries.

Cho et al. (2005) later introduced the MLS-based *(n+4)*-variable node element as the interface element. The 4 refers to the 4 corner nodes of the element, while *n* are the additional nodes added along one edge of the element. For a two dimensional element, the weighting function defined at these *(n+4)*-nodes are the product of two quartic splines defined along each of the local coordinates. The value of the weight function is 1 at the associated node and non-zero along the edges that connect to the associated node. The resultant shape functions yield linear displacement along the edges divided by the *n* nodes. The same concept was extended to the more general *(k+l+m+n+4)*-variable node element where each edge is added with *k*, *l*, *m* and *n* nodes, respectively. This special MLS-based *(n+4)*-variable node element was later extended by Cho and Im (2006) to quadratic variable node elements in which the shape functions will produce quadratic displacement along every 3-node segment of an edge.

The MLS-based variable node elements mentioned above were derived by selecting more particles or nodal points where the weighting functions are defined than the number of base functions. As a result, the shape functions are rational in the element interior, which requires a high number of Guassian quadrature points, in order to ensure the quality of integration. Lim, Im and Cho (2007) and Lim and Im (2007) carefully selected the weighting function along its base of integration to produce a new class of MLS-based variable node elements in 2007. The number of nodal points in these new variable node elements is the same as that of the base functions. The resultant shape functions are polynomials which can be accurately integrated with 2×2 or 3×2 Gaussian quadrature points. They later extended the new variable node element to three dimensional applications [Lim, Im and Cho, 2007].

Lim et al. (2010) replaced the Gaussian quadrature integration used in their MLS-based variable node elements by the cell-smoothed integration technique. The goal here was to avoid numerical stability encountered by Gaussian quadrature integration and improve the quality of the solution. The revised variable node elements were successfully applied to solve two-dimensional multi-scale mechanics problems and three-dimensional elastic-plastic analysis [Lee, Son and Im, 2015].

In this study, the compatibility condition between the mismatched elements was treated as a linear multipoint constraint. This compatibility condition was enforced directly in the element level to generate a variable node element which can serve as a transition between mismatched elements. This study is different from the previous work done by Park et al. (2002), Panteno and Averill (2002) and Aminpour et al. (2001), in which such a compatibility condition was enforced on the system level.

The derivation of the constrained super-element are presented in Chapter 2. The demonstrative examples are presented in Chapter 2 to illustrate the use of the derived super-element as a variable node element to transmit the analysis domain from the fine mesh to the coarse mesh zone. Two plane strain problems, a cantilever beam and a plate with a hole, are used in Chapter 3 to validate the use of the variable node elements. The exact solutions of these two problems are compared to those of the finite element models with and without the use of the derived variable node elements. The concluding remarks are presented at the end of this report.

CHAPTER 2

BASIC DERIVATION AND THEORY

In this chapter, the Elimination methods will be introduced. This method is used to show how a model with different mesh resolutions can merge. The Elimination method derives new stiffness matrixes by eliminating unconnected and interior nodes in the transition zone. This method is derived in this chapter. Also, at the end of this chapter, a simple cantilever beam with a tip load will be analyzed to demonstrate the validity of the Elimination method.

2.1 Variable Node Super-element

A given structure, as shown in Figure 2-1' is discretized into a set of finite elements and nodes. A sub-set of the discretized finite elements and nodes is collected to form a super-element, as shown in Figure 2-2. A set of forces, $\boldsymbol{f}$ 'is applied to the nodes of this super-element and a set of interaction forces $\boldsymbol{R}$ 'are applied to the boundary nodes, imposed by the adjacent elements connected to this particular super-element. Furthermore, a group of the boundary nodes are subjected to a set of self-imposed linear multipoint constraints, $C\boldsymbol{q} = 0$, where $\boldsymbol{q}$ is the displacement vector collected from nodes associated with the super-element.

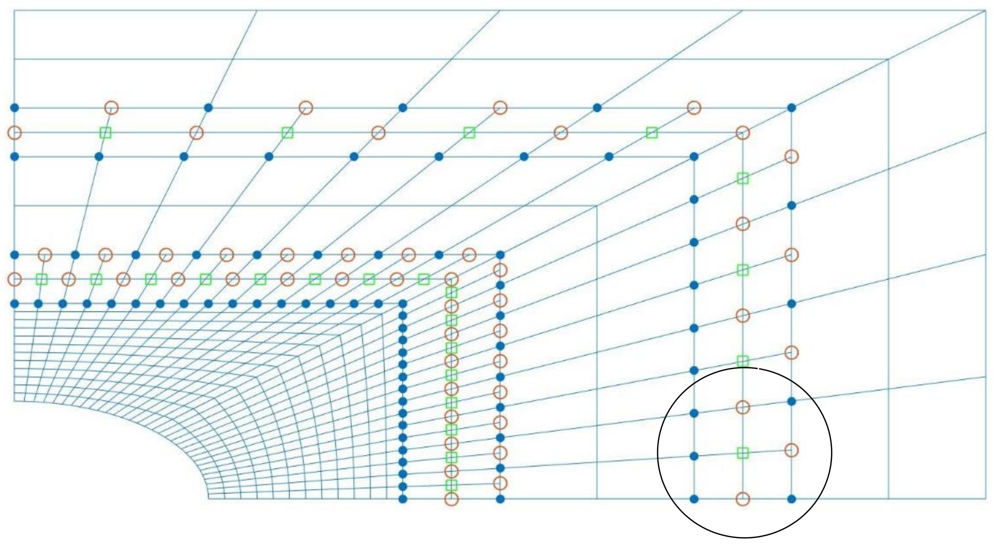

Figure 2-1. A discretized structure domain with mismatched elements

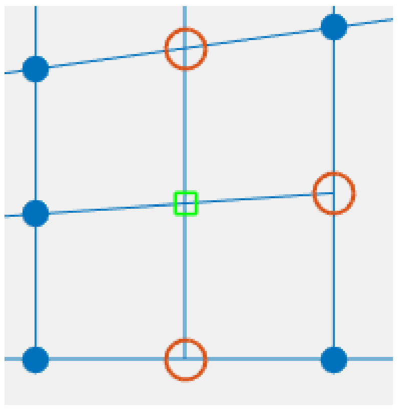

Figure 2-2. One variable node element from the model: (a) ●: the connected node, (b) □: the interior node, and (c) ○: the unconnected node

The solution of the finite element equation defined in this super-element can be casted as that of the following minimization problem Eq. (2.1):

$$\frac{1}{2}\boldsymbol{q}^T K\boldsymbol{q} - \boldsymbol{q}^T \boldsymbol{f} - \boldsymbol{q}^T \boldsymbol{R} + \boldsymbol{\lambda}^T C\boldsymbol{q} \tag{2.1}$$

where λ is the Lagrange multiplier. The necessary condition of minimization leads to Eq. (2.2):

$$K\boldsymbol{q} = \boldsymbol{f} + \boldsymbol{R} - C^T\boldsymbol{\lambda} \tag{2.2}$$

Next, a typical super-element divides the nodal degrees of freedom, $\boldsymbol{q}$ into the interior and the boundary nodes; the nodal degrees of freedom are divided into interior or boundary nodes, which are appropriately noted by the subscripts "*I*" and "*B*", respectively. Thus, one has $\boldsymbol{q}^{\boldsymbol{T}} = (\boldsymbol{q}_{\boldsymbol{I}} \quad \boldsymbol{q}_{\boldsymbol{B}})$.

Now, plugging in Eq. (2.2):

$$\begin{bmatrix} K_{II} & K_{IB} \\ K_{BI} & K_{BB} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q}_I \\ \boldsymbol{q}_B \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_I \\ \boldsymbol{f}_B \end{Bmatrix} + \begin{Bmatrix} 0 \\ R_B \end{Bmatrix} - \begin{bmatrix} 0 \\ C_B^T \end{bmatrix} \boldsymbol{\lambda}_B \tag{2.3}$$

Since the MPC constraints and the reaction force are all associated with the boundary nodes, they can be further specified as $C_B q_B = 0$ and $R_B = 0$, respectively. Static condensation can then be applied to Eq. (2.3) to eliminate C_I from the equation. Therefore, the equation for the boundary of degrees of freedom will be shown below:

$$\left[K_{BB} - K_{BI}K_{II}^{-1}K_{IB}\right]\boldsymbol{q}_{\boldsymbol{B}} = \boldsymbol{f}_B - K_{BI}K_{II}^{-1}\boldsymbol{f}_I + R_B - C_B^{\boldsymbol{T}}\boldsymbol{\lambda}_B \tag{2.4}$$

The boundary nodes are further divided into "connected" and "non-connected" categories based upon whether they are connected to the adjacent elements or not. If the boundary nodes are connected to the adjacent parental elements, then they are considered to be connected. While, unconnected nodes are the boundary nodes not connected to a node of parental elements on any side. The displacement of each of the unconnected nodes is a linear function of those of the connected nodes. Consequently, the interface reaction forces are applied only to those nodes which are considered connected.

Moreover, due to the fact that boundary nodes are divided into "connected" and "unconnected" nodes, the displacements can be realized in Eq. (2.5):

$$\boldsymbol{q}_B = \begin{Bmatrix} \boldsymbol{q}_U \\ \boldsymbol{q}_C \end{Bmatrix} = \begin{bmatrix} C_{UC} \\ I_C \end{bmatrix} \boldsymbol{q}_C \tag{2.5}$$

Therefore, Eq. (2.6) provides the relation between boundary node displacements.

$$q_u = C_{uc} q_c \tag{2.6}$$

Also, constraints are applied to the unconnected nodes that are simply part of the boundary nodes. So, in Eq. (2.7), the unconnected displacements are considered zeros.

$$C_B \boldsymbol{q}_B = 0 = \begin{bmatrix} I_U & -C_{UC} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q}_U \\ \boldsymbol{q}_C \end{Bmatrix} \tag{2.7}$$

It should be noted that by substituting the transpose coefficient matrix, it will end up as the following equation:

$$\begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} C_B^{\boldsymbol{T}} = \begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{bmatrix} I_U \\ -C_{UC}^T \end{bmatrix} = 0 \tag{2.8}$$

By comparing Eq. (2.4) with Eq. (2.8), it is clear that both equations can be combined and extended into the expression in Eq. (2.9)

$$\begin{aligned} &\left(\begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} \begin{bmatrix} K_{BB} - K_{BI} K_{II}^{-1} K_{IB} \end{bmatrix} \begin{bmatrix} -C_{UC} \\ I_C \end{bmatrix} \right) \boldsymbol{q}_{BC} \\ &= \begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_B - K_{BI} K_{II}^{-1} \boldsymbol{f}_I \end{bmatrix} + \begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} \boldsymbol{R}_B - \begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} C_B^{\boldsymbol{T}} \boldsymbol{\lambda}_B \; . \\ &= \begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_B - K_{BI} K_{II}^{-1} \boldsymbol{f}_I \end{bmatrix} + \begin{bmatrix} I_C & C_{UC}^T \end{bmatrix} \boldsymbol{R}_B \\ &= \boldsymbol{f}_C - K_{CI} K_{II}^{-1} \boldsymbol{f}_I + \boldsymbol{R}_C + C_{UC}^T \left(\boldsymbol{f}_U - K_{UI} K_{II}^{-1} \boldsymbol{f}_I + \boldsymbol{R}_U \right) \end{aligned} \tag{2.9}$$

Alternatively, it can be directly assumed that only the connected nodes of the variable-node element will be connected to the adjacent elements. Consequently, $\boldsymbol{R}_U = 0$ and Eq. (2.3) is repeated below as:

$$\begin{bmatrix} K_{II} & K_{IB} \\ K_{BI} & K_{BB} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q}_I \\ \boldsymbol{q}_B \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_I \\ \boldsymbol{f}_B \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{R_B} \end{Bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ C_B^T \end{bmatrix} \boldsymbol{\lambda}_B \; .$$

Eq. (2.10) is a further expanded based upon Eqs. (2.3) and (2.6):

$$\begin{bmatrix} K_{II} & K_{IU} & K_{IC} \\ K_{UI} & K_{UU} & K_{UC} \\ K_{CI} & K_{CU} & K_{CC} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q_I} \\ \boldsymbol{q_U} \\ \boldsymbol{q_C} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f_I} \\ \boldsymbol{f_U} \\ \boldsymbol{f_C} \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{R_U} \\ \boldsymbol{R_C} \end{Bmatrix} - \begin{Bmatrix} \boldsymbol{0} \\ I_U \\ \text{-}\, C_{UC}^T \end{Bmatrix} \boldsymbol{\lambda}_U \ . \qquad (2.10)$$

The last two rows of the above equation can be rearranged in terms of the boundary degrees of freedom by imposing the boundary condition in Eq. (2.11):

$$\begin{bmatrix} K_{UU} - K_{UI}K_{II}^{-1}K_{IU} & K_{UC} - K_{UI}K_{II}^{-1}K_{IC} \\ K_{CU} - K_{CI}K_{II}^{-1}K_{IU} & K_{CC} - K_{CI}K_{II}^{-1}K_{IC} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q}_U \\ \boldsymbol{q}_C \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_U - K_{UI}K_{II}^{-1}\boldsymbol{f_I} \\ \boldsymbol{f}_C - K_{CI}K_{II}^{-1}\boldsymbol{f_I} \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{R_U} \\ \boldsymbol{R_C} \end{Bmatrix} - \begin{bmatrix} I_U \\ \text{-}\, C_{UC}^T \end{bmatrix} \boldsymbol{\lambda}_U \ . \qquad (2.11)$$

Replacing $\boldsymbol{q}_U$ with $\boldsymbol{q}_C$ to Eq. (2.6) and pre-multiplying $\begin{bmatrix} I_U & C_{UC}^T \end{bmatrix}$ by Eq. (2.11), will be demonstrated in Eq. (2.12):

$$\begin{aligned} &\begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{bmatrix} K_{UU} - K_{UI}K_{II}^{-1}K_{IU} & K_{UC} - K_{UI}K_{II}^{-1}K_{IC} \\ K_{CU} - K_{CI}K_{II}^{-1}K_{IU} & K_{CC} - K_{CI}K_{II}^{-1}K_{IC} \end{bmatrix} \begin{bmatrix} C_{UC} \\ I_C \end{bmatrix} \boldsymbol{q}_C \\ &= \begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{Bmatrix} \boldsymbol{f}_U - K_{UI}K_{II}^{-1}\boldsymbol{f_I} \\ \boldsymbol{f}_C - K_{CI}K_{II}^{-1}\boldsymbol{f_I} \end{Bmatrix} + \begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{Bmatrix} \boldsymbol{R_U} \\ \boldsymbol{R_C} \end{Bmatrix} - \begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{bmatrix} I_U \\ \text{-}\, C_{UC}^T \end{bmatrix} \lambda_U \ . \qquad (2.12) \\ &= \begin{bmatrix} C_{UC}^T & I_C \end{bmatrix} \begin{Bmatrix} \boldsymbol{f}_U - K_{UI}K_{II}^{-1}\boldsymbol{f_I} \\ \boldsymbol{f}_C - K_{CI}K_{II}^{-1}\boldsymbol{f_I} \end{Bmatrix} + \boldsymbol{R_C} \end{aligned}$$

Note that $\boldsymbol{R}_U = 0$. Eq. (2.13) can be represented in terms of a new stiffness matrix for the variable node super-element to solve for $\boldsymbol{q}_C$ as:

$$K_{CC}^{*}\boldsymbol{q}_C = \boldsymbol{f}_C^{*} + \boldsymbol{R}_C \qquad (2.13)$$

where the stiffness matrix K_{CC}^{*} and the force vector $\boldsymbol{f}_C^{*}$ for the variable node super-element are given below:

$$
\begin{aligned}
K_{CC}^{*} &= \begin{bmatrix} C_{UC}^{T} & I_{C} \end{bmatrix} \begin{bmatrix} K_{UU} - K_{UI}K_{II}^{-1}K_{IU} & K_{UC} - K_{UI}K_{II}^{-1}K_{IC} \\ K_{CU} - K_{CI}K_{II}^{-1}K_{IU} & K_{CC} - K_{CI}K_{II}^{-1}K_{IC} \end{bmatrix} \begin{bmatrix} C_{UC} \\ I_{C} \end{bmatrix} \\
&= \begin{bmatrix} C_{UC}^{T}\left(K_{UU} - K_{UI}K_{II}^{-1}K_{IU}\right) + K_{CU} - K_{CI}K_{II}^{-1}K_{IU} & C_{UC}^{T}\left(K_{UC} - K_{UI}K_{II}^{-1}K_{IC}\right) + K_{CC} - K_{CI}K_{II}^{-1}K_{IC} \end{bmatrix} \begin{bmatrix} C_{UC} \\ I_{C} \end{bmatrix} \qquad (2.14) \\
&= C_{UC}^{T}\left(K_{UU} - K_{UI}K_{II}^{-1}K_{IU}\right)C_{UC} + \left(K_{CU} - K_{CI}K_{II}^{-1}K_{IU}\right)C_{UC} \\
&\quad + C_{UC}^{T}\left(K_{UC} - K_{UI}K_{II}^{-1}K_{IC}\right) + K_{CC} - K_{CI}K_{II}^{-1}K_{IC}
\end{aligned}
$$

$$
\boldsymbol{f}_{\boldsymbol{C}}^{*} = C_{UC}^{T}\left(\boldsymbol{f}_{U} - K_{UI}K_{II}^{-1}\boldsymbol{f}_{\boldsymbol{I}}\right) + \boldsymbol{f}_{C} - K_{CI}K_{II}^{-1}\boldsymbol{f}_{\boldsymbol{I}} \qquad (2.15)
$$

Because each variable node super-element contains one interior node, three unconnected nodes, and five connected nodes in the following derivation, the connected and unconnected displacements will be written as:

$$
\boldsymbol{q}_{U} = \begin{Bmatrix} q_2 \\ q_6 \\ q_8 \end{Bmatrix} \quad \text{and} \quad \boldsymbol{q}_{C} = \begin{Bmatrix} q_1 \\ q_3 \\ q_4 \\ q_7 \\ q_9 \end{Bmatrix} .
$$

Besides, C_{UC}^{T} is the coefficient matrix, which demonstrates the relationship between unconnected and connected variable node super-element nodal displacements. Since three unconnected and five connected nodes are placed in each element by considering two degrees of freedom at each node, a coefficient matrix will be constructed below as six by ten. Moreover, the coefficient matrix can be modified in a manner for more boundary and interior nodes. So, in this case, the average of two connected neighboring nodes is:

$$
C_{UC} = \begin{bmatrix}
0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5
\end{bmatrix} .
$$

Also, Eq. (2.6) can be readily expanded below as:

$$\begin{Bmatrix} u_2 \\ v_2 \\ u_6 \\ v_6 \\ u_8 \\ v_8 \end{Bmatrix} = \begin{bmatrix} 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \\ u_7 \\ v_7 \\ u_9 \\ v_9 \end{Bmatrix}.$$

Furthermore, from Eq. (2.10) the stiffness matrix of each interior node is derived.

$$\begin{aligned} K_{II}\boldsymbol{q}_I &= \boldsymbol{f}_I - K_{IU}\boldsymbol{q}_U - K_{IC}\boldsymbol{q}_C \\ &= \boldsymbol{f}_I - \left(K_{IU}C_{UC} - K_{IC}\right)\boldsymbol{q}_C \end{aligned} \tag{2.16}$$

The relative interior displacements vector is derived in Eq. (2.17):

$$\begin{aligned} \boldsymbol{q}_I &= K_{II}^{-1}\boldsymbol{f}_I - K_{II}^{-1}K_{IU}\boldsymbol{q}_U - K_{IC}\boldsymbol{q}_C \\ &= K_{II}^{-1}\boldsymbol{f}_I - K_{II}^{-1}\left(K_{IU}C_{UC} - K_{IC}\right)\boldsymbol{q}_C \end{aligned} . \tag{2.17}$$

Alternatively, the penalty method can be used to enforce the MPC constraints of Eq. (2.6) into the super-element. In this case, the minimization problem of Eq. (2.1) is reformulated with the introduction of the penalty coefficient, *α*, as:

$$\min \quad \frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{q}^T K\boldsymbol{q} - \boldsymbol{q}^T \boldsymbol{f} - \boldsymbol{q}^T \boldsymbol{R} + \alpha\boldsymbol{q}^T C^T C\boldsymbol{q} \ . \tag{2.18}$$

The necessary condition of minimization leads to Eq. (2.19)

$$K\boldsymbol{q} = \boldsymbol{f} + \boldsymbol{R} - \alpha C^T C\boldsymbol{q} \tag{2.19}$$

which can be explicitly spelled out with the help of Eq. (2.6),

$$\begin{bmatrix} K_{II} & K_{IU} & K_{IC} \\ K_{UI} & K_{UU}+\alpha I_U & K_{UC}-\alpha C_{UC} \\ K_{CI} & K_{CU}-\alpha C_{UC}^T & K_{CC}+\alpha C_{UC}^T C_{UC} \end{bmatrix} \begin{Bmatrix} \boldsymbol{q_I} \\ \boldsymbol{q_U} \\ \boldsymbol{q_C} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f_I} \\ \boldsymbol{f_U} \\ \boldsymbol{f_C} \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{R_C} \end{Bmatrix} \tag{2.20}$$

Static condensation can be applied here as well to eliminate the degrees of freedom, $\boldsymbol{q}_I$ and $\boldsymbol{q}_U$ to form a reduced matrix equation, similar to Eq. (2.13).

The MPC-penalty is given in Eq. (2.19) with the coefficient matrix consisting of $\beta_1,\ \beta_2,\ \beta_3$ and β_0. These are the known constants. The first and the third coefficients are 0.5; the second coefficient and the fourth coefficients are -1 and 0, respectively, since nodes within the super-element consists of four individual elements.

$$\beta_1 q_1 + \beta_2 q_2 + \beta_3 q_3 = \beta_0 \tag{2.21}$$

Such boundary conditions are referred to as “Multi Point Constraints” in the literature. The penalty approach will now be elicited in order to understand what happens when this type of boundary condition is applied. As is shown next in Eq. (2.22) the modified total potential-energy expression must be considered:

$$\pi_p = \frac{C}{2}\left(\beta_1 q_{p_1} + \beta_2 q_{p_2} + \beta_3 q_{p_3} - \beta_0\right)^2 \tag{2.22}$$

where C is a large number. Since C is a large number, π_p takes a minimum value only when Eq. (2.22) is set. In other words, Eq. (2.23) will further be expressed as:

$$\left(\frac{\partial \pi_p}{\partial \boldsymbol{q}}\right)^T = \begin{bmatrix} C\beta_1\beta_1 & C\beta_1\beta_2 & C\beta_1\beta_3 \\ C\beta_2\beta_1 & C\beta_2\beta_2 & C\beta_2\beta_3 \\ C\beta_3\beta_1 & C\beta_3\beta_2 & C\beta_3\beta_3 \end{bmatrix} \begin{Bmatrix} q_{p_1} \\ q_{p_2} \\ q_{p_3} \end{Bmatrix} - \begin{Bmatrix} C\beta_1\beta_0 \\ C\beta_2\beta_0 \\ C\beta_3\beta_0 \end{Bmatrix}. \tag{2.23}$$

In this case, since each unconnected node is placed between two connected nodes, the unconnected node is the average of two neighboring nodes within the variable node element. Based on Eq. (2.21), the following MPCs will be:

$$0.5u_1 - u_2 + 0.5u_3 = 0$$
$$0.5v_1 - v_2 + 0.5v_3 = 0$$
$$0.5u_3 - u_6 + 0.5u_9 = 0$$
$$0.5v_3 - v_6 + 0.5v_9 = 0$$
$$0.5u_7 - u_8 + 0.5u_9 = 0$$
$$0.5v_7 - v_8 + 0.5v_9 = 0$$

2.2 Demonstrative examples

The cantilever beam challenge will be exemplified to show how it will be solved in an effort to validate the displacements and Von-Mises stresses. The beam's length, depth, and thickness are 12, 2, and 0.1 meters. Additionally, it is also assumed that force is applied at the top end of the beam with +100 N in a standard coordinate system. The modulus of elasticity is 7×10^4 Pa as well.

Moreover, the "Plane Stress" condition is assumed, and the portrait of this example is shown in Figure 2-3 below.

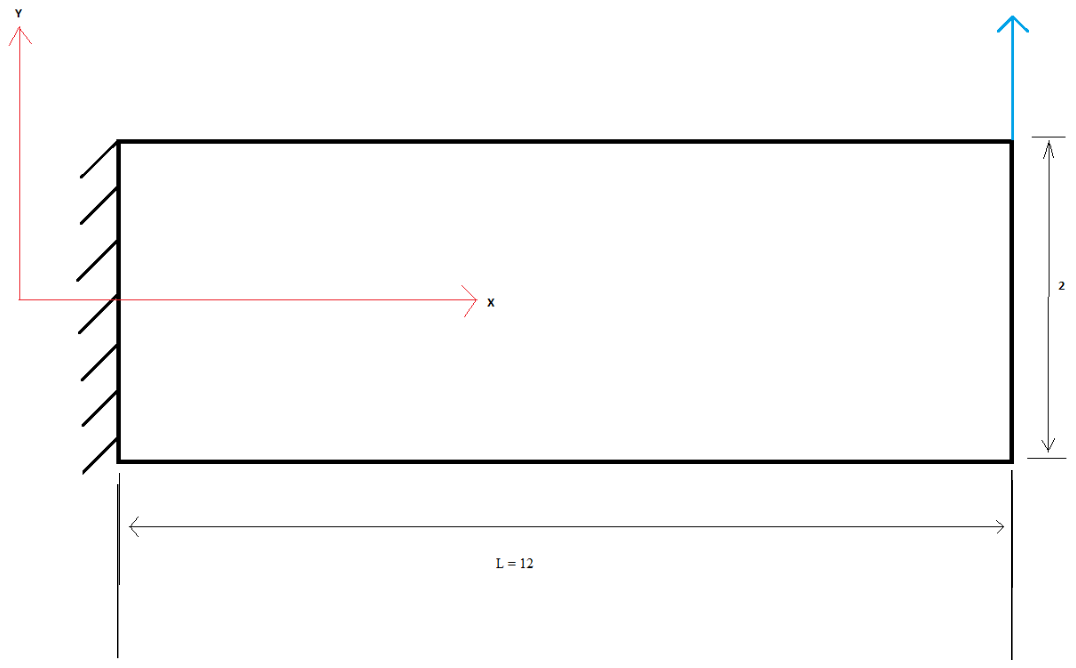

Figure 2-3. A simple cantilever beam with a tip load

2.2.1 Analytical solution

The cantilever beam is loaded with a point load P at the free end. The length and the depth of the beam are set to L and D, respectively. The Young's modulus is E and the Poisson's ratio ,v. The displacement fields, $[U_x(x,y), U_y(x,y)]$ of the plane stress problem are given by:

$$U_x(x,y) = -\frac{Py}{6EI}\left\{(6L-3x)x + (2+v)\times\left(y^2 - \frac{D^2}{4}\right)\right\} \tag{2.24}$$

$$U_y(x,y) = \frac{P}{6EI}\left\{3vy^2(L-x) + (4+5v)\frac{D^2x}{4} + (3L-x)x^2\right\}. \tag{2.25}$$

Since the aim here is to provide the displacements along the tip load (*x*=*l*), the simplified equations are displayed below:

$$U_x(y) = -\frac{Py}{6EI}\left\{(6L^2) + (2+v)\times\left(y^2 - \frac{D^2}{4}\right)\right\} \tag{2.26}$$

$$U_y(y) = \frac{P}{6EI}\left\{(4+5v)\frac{D^2L}{4} + 2L^3\right\}. \tag{2.27}$$

Eqs. (2.24-2.25) will provide the elongations and deflections of the two dimensional beams, respectively, in terms of plane stress. Besides, the neutral axis should be considered regarding the exact solution [Timoshenko, 1970].

Also, the following Eqs. (2.28-2.31) are given so that the stresses in a two dimensional cantilever beam can be computed as [Timoshenko, 1970]:

$$\sigma_{xx} = -\frac{P(L-x)y}{I} \tag{2.28}$$

$$\sigma_{yy} = 0 \tag{2.29}$$

$$\sigma_{xy} = \frac{P}{2I}\left(\frac{D^2}{4} - y^2\right). \tag{2.30}$$

Finally, the Von-Mises stress equation will be:

$$\sigma_v = \sqrt{\sigma_{xx}^2 - \sigma_{xx}\sigma_{yy} + \sigma_{yy}^2 + 3\sigma_{xy}^2}\,. \tag{2.31}$$

The results of the displacements are given in Table 2-1, according to the analytical solutions that resulted, in terms of plane stress:

Table 2-1. Results for analytical solution for the cantilever beam

X=L	Analytical-solutions			
X	Y	Ux-disp	Uy_disp	σ_v
0	-1	0	0.0424	18000
0	0	0	0	18000
0	1	0	0.0424	18000
12	-1	1.5429	12.585	0
12	0	0	12.585	0
12	1	-1.5429	12.585	0

The results above signify elongation and deflection of the beam when X=L along the Y-axis, and has symmetric and constant values of elongation and deflection. At this time, the analytical values will be compared with various finite element methods.

2.3.2 The cantilever beam with a tip load no transition zone

In the four-node quadrilateral element, known as QUAD, each element has four nodes and each node has two degrees of freedom. In this mesh, the number of nodes and number of elements are 18 and 10 respectively. Since the number of division on the vertical axis is 2, and it is constant over the entire domain, merging in the domain will not be present. In other words, all the nodes in the domain are connected and there are no meshes with different resolutions.

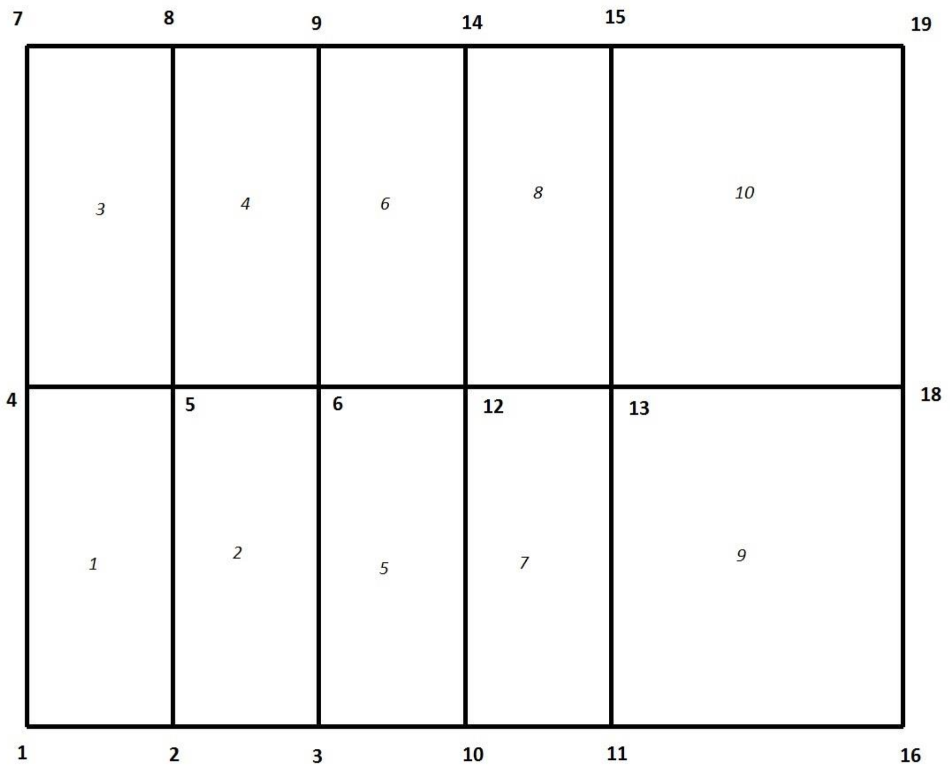

Figure 2-4. Four-noded quadrilateral element (QUAD) initial mesh

In Figure 2-4, it can be readily seen that meshes are consistent over the entire domain. Let’s assume the number of divisions of a domain vertically is “W-span”, and horizontally is “S-span.” As a result, in terms of domain divisions, “W-span” is equal to two’and “S-span” is equal to five.

By comparing displacement values at the end of a beam with analytical solutions, the error might seem to be large; however, nodes 16 and 17 follow the identical format by having symmetric values as analytical solutions. Also, nodes 1 and 7 not only have the highest Von-Mises stresses but they also have the highest stresses within the entire domain, regardless of negligible error presence. Basically, the nodal stresses are derived by method of curve fitting. To state it differently, the Von-Mises stresses will be computed at four Gaussian points of each element. Afterwards, nodal stresses will be readily generated by using curve fitting. Nodal stresses are not very close to the exact solution since the method of curve fitting is implemented to provide the approximate values. So, nodal Von-Mises values are used for plotting the Von-Mises contour plot, given in Figure 2-5.

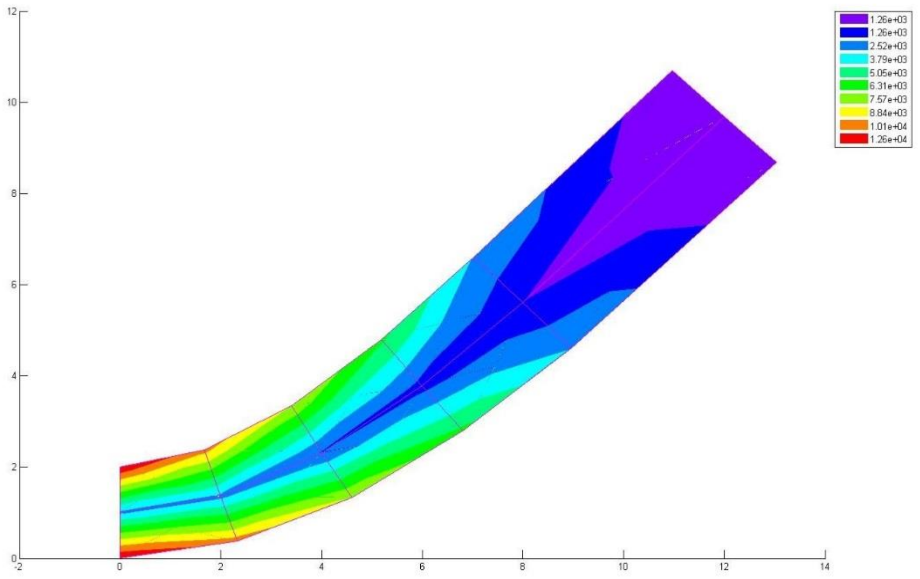

Figure 2-5. Von_Mises contour plot for cantilever beam with a tip load with no transition zone

2.3.3 The cantilever beam with a tip load with one variable node super-element

In Figure 2-6 the same cantilever beam has been used, with the only difference being that it has a transition element. The transition element contains 9 nodes which illustrated by hatch lines within the domain. This transition element contains four elements such as 5, 6, 7, and 8. In this case, since two fine elements merge with one coarse element, the Elimination method is implemented for variable node super-element construction. Basically, based upon the derivation shown earlier, it is needed to maintain the connected nodes in order to provide a merging mechanism for different mesh resolutions. In Figure 2-6 the connected nodes are: 3, 11, 6, 9, and 15, which are each maintained to provide the variable node super-element. However, the unconnected nodes are 10, 13, and 14, and the interior node is 12. In Chapter 2, the derivation of connected nodes is discussed, and the relative unconnected and interior node displacements revived.

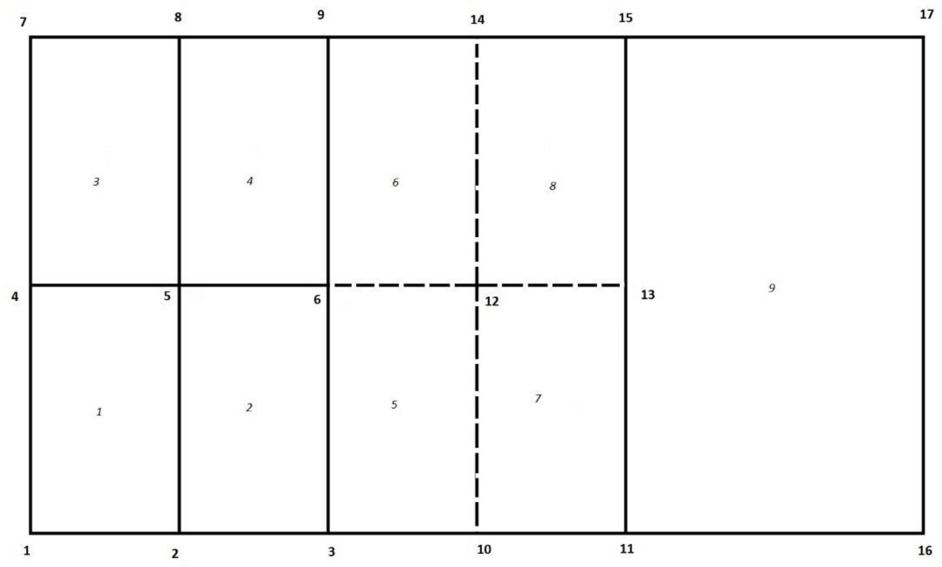

Figure 2-6. Initial mesh plot for the cantilever beam with a tip load with one variable node super-element

After maintaining connected nodes, the overview of the model with a variable node element is shown in Figure 2-7 all five connected nodes are maintained to connect the elements.

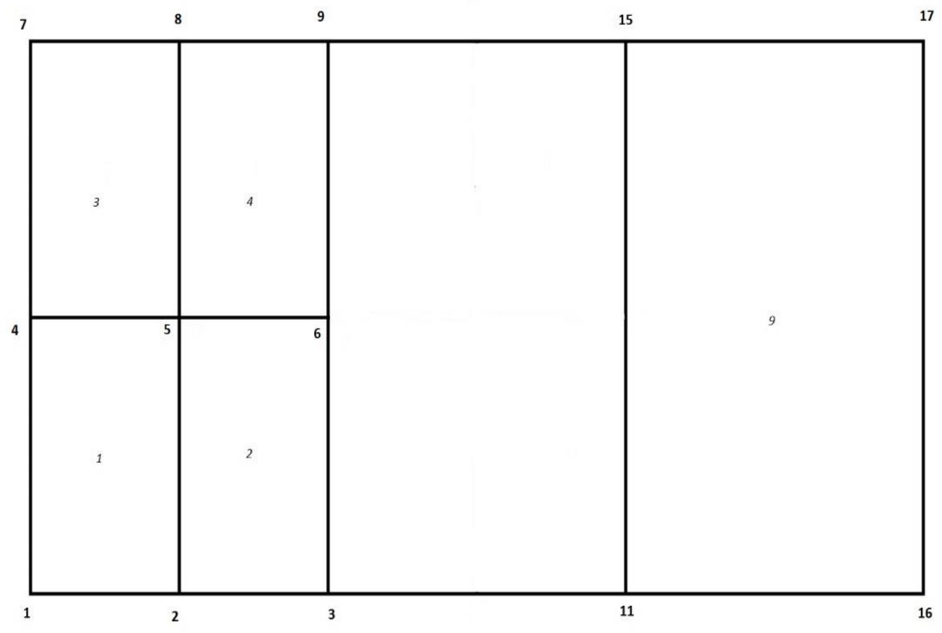

Figure 2-7. Mesh plot Cantilever beam with a tip load after maintaining connected node with a new variable node super-element

Two methods, the penalty method and the Elimination method, are used for Figure 2-7. Both resulted in the same displacements at the end of the tip beam reactions and Von-Mises stresses over the support. Therefore, their contour plots, in terms of Von-Mises stresses, are identical. Figure 2-8 shows the contour plot for Von-Mises stress in the entire domain. Figure 2-8 follows the same format as Figure 2-5; however, there might be some discrepancies within the desired coordinates and Von-Mises stresses possibly due to coarse meshes that have settled over the domains. If finer mesh is plotted in the domain, closer output values can be expected in terms of data validation.

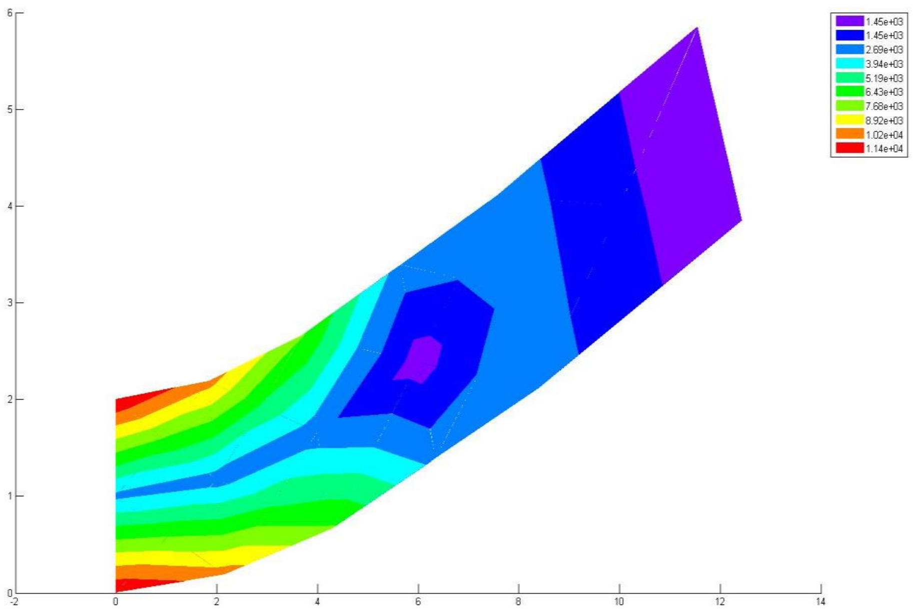

Figure 2-8. Von_Mises contour plot for cantilever beam with a tip load with one variable node super-element

2.3.4 The cantilever beam with a tip load with two variable node super-elements

Figure 2-9 shown below for the cantilever beam with a tip load at the end is demonstrated to provide more than one variable node super-element in the domain. In this case, two variable node elements have been settled in the domain and compared with previous examples in which there was only one variable node element. The purpose of having two transition elements within the domain is to increase the precision of validation of output data. The first variable node element which contains four elements such as 9, 10, 11, and 12 have connected nodes such as 3, 17, 6, 9 and 21; unconnected nodes such as 16, 19 and 20; and a single interior node, being 18. The first variable node element will be merging two element numbers, 2 and 4, to the element number 17. The second variable node element' which is located above the first variable node element' is the combination of element numbers such as 13, 14, 15, and 16. The second variable node element has connected nodes such as 9, 21, 22, 15, and 25. The connected nodes will remain in domain to build up the variable node element. Due to the fact that the second variable node element contains element numbers 13, 14, 15, and 16, maintaining connected nodes with element 6 and 8 allows them to merge to element 18.

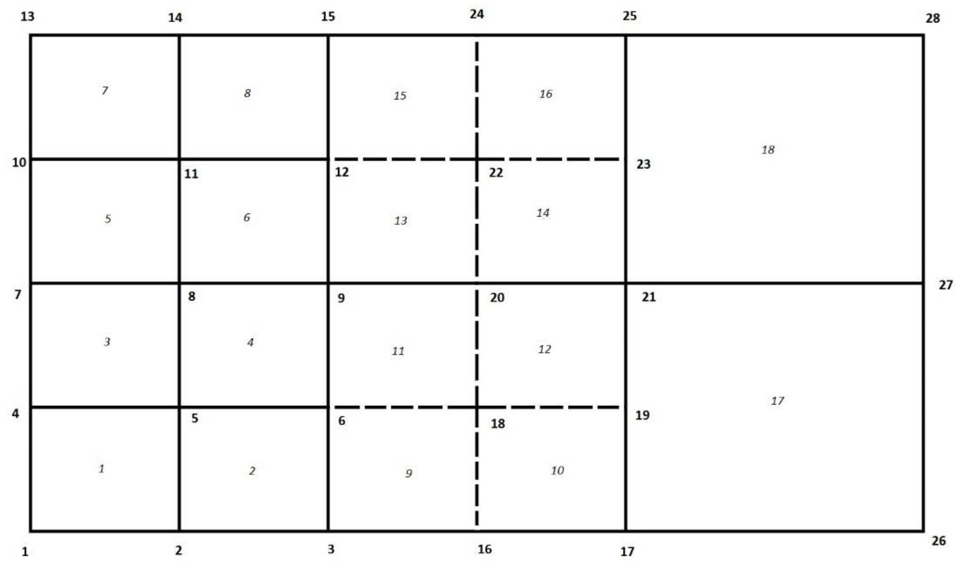

Figure 2-9. Initial mesh for the cantilever beam with a tip load with two variable node super-elements

After constructing the two variable node elements (which had been discussed earlier) and maintaining the desired nodes, the domain will come up with two five-noded variable node elements. This situation is illustrated in Figure 2-10.

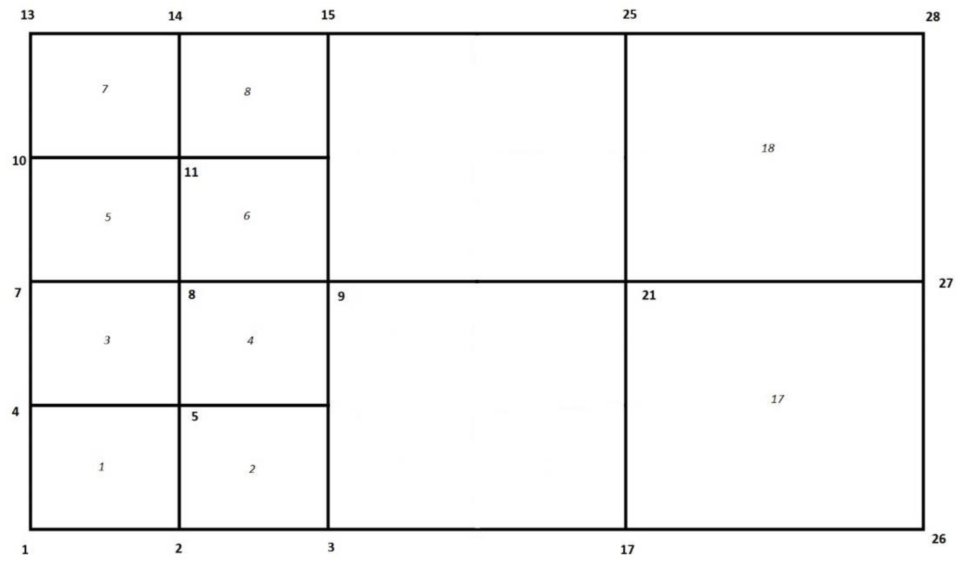

Figure 2-10. Cantilever beam with a tip load after maintaining connected node in two variables node super-elements

By generating output data such as displacements, nodal reactions, and Von-Mises stresses, all results such as the nodal displacements, nodal reactions, and elemental Von-Mises stresses can be generated by using Elimination matlab code in Appendix 1. Due to the fact that all the output values, in terms of the Elimination approach, are the same as the MPC approach, Von-Mises contour plots are identical. The Von-Mises contour plot is shown in Figure 2-11. The figure below is for two variable node super-elements using a cantilever beam, conveying that the highest stress is on the fixed points marked with a red coloration. The top half is compressional Von-Mises stress and the bottom half is tensional Von-Mises stress. The figure below portrays this.

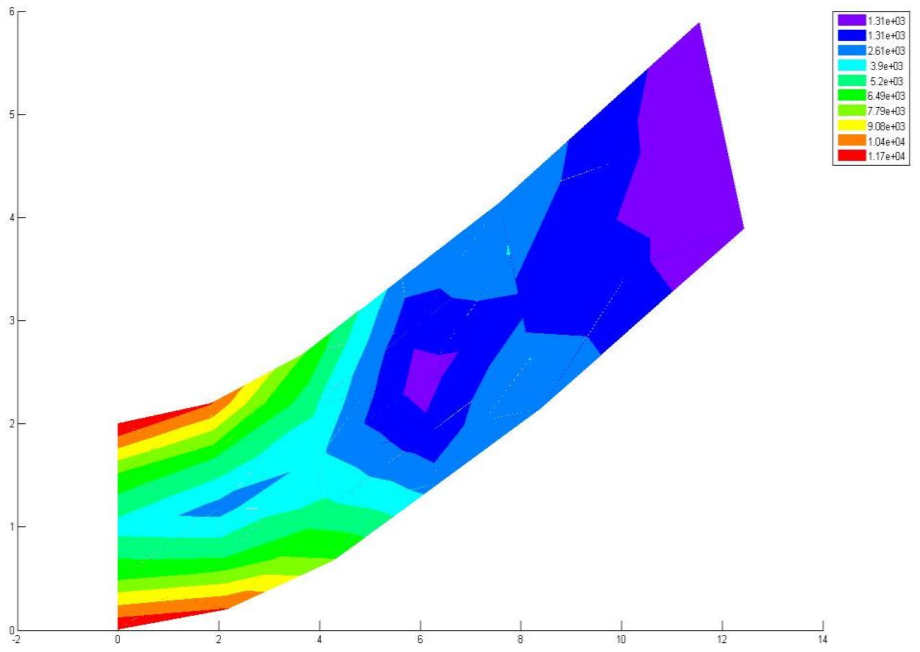

Figure 2-11. Von Mises contour plot for cantilever beam with a tip load with two variable node super-element

2.3.5 The cantilever beam with a tip load with three variable super-elements

Figure 2-12 is shown below for the cantilever beam, with a tip load at the end. This figure provides more than two variable node super-elements in the domain. In this case, three variable node super-elements have been located in the domain in order to draw a comparison with the previous example with fewer variable node super-elements. The purpose of having three variable node elements within the domain is to increase the precision of validation for the output data. The first variable node super-element contains four elements such as 3, 4, 7, and 8, and has connected nodes such as 3, 5, 8, 13, and 15. Unconnected nodes are realized in 4, 10, and 14, and one interior node in 9. The first variable node element will be merging two elements, numbers 2 and 6, to the element number 25. The second variable node super-element located above the first variable node element is the combination of element numbers such as 11, 12, 15 and 16. The second variable has connected nodes such as 13, 15, 18, 23, and 25. The connected nodes will remain in the domain to build up the variable node super-element. The unconnected nodes are identified in 14, 20, and 24, and the interior node is found in 19 within the second variable node element. The third and the last variable node super-element, above the second variable node super-element, contains elements 19, 20, 23, and 24. This variable node super-element has five connected nodes - 23, 25, 28, 33, and 35. In addition, the unconnected nodes are 24, 30, and 34. Here, the interior node is 29. The three variable node elements are to merge two elements to one element in the set of transition elements.

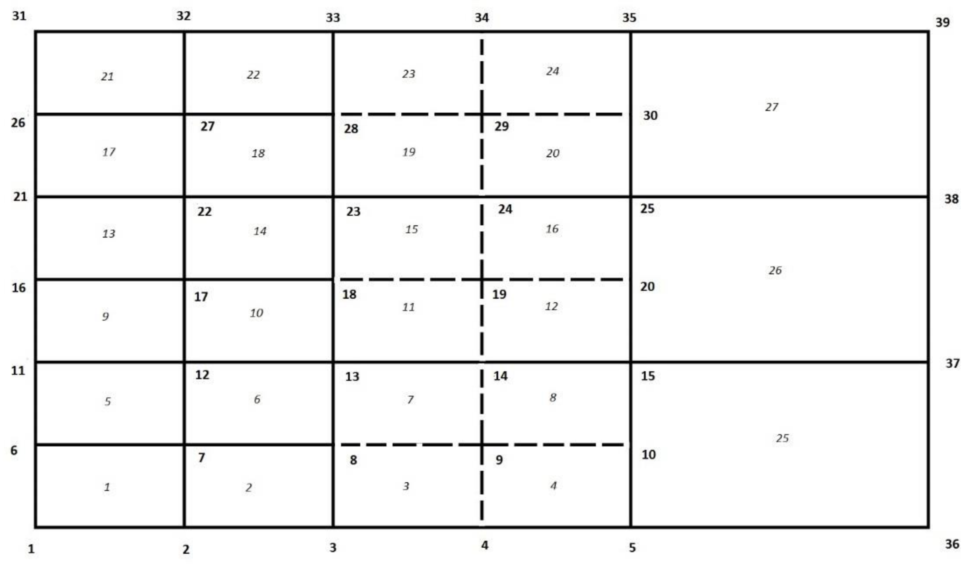

Figure 2-12. Initial mesh plot for the cantilever beam with a tip load with three variable node elements

Figure 2-13 shows that each variable super-element retains its five connected nodes and will be considered as one variable element. As a result, Figure 2-12 can be extended to Figure 2-13.

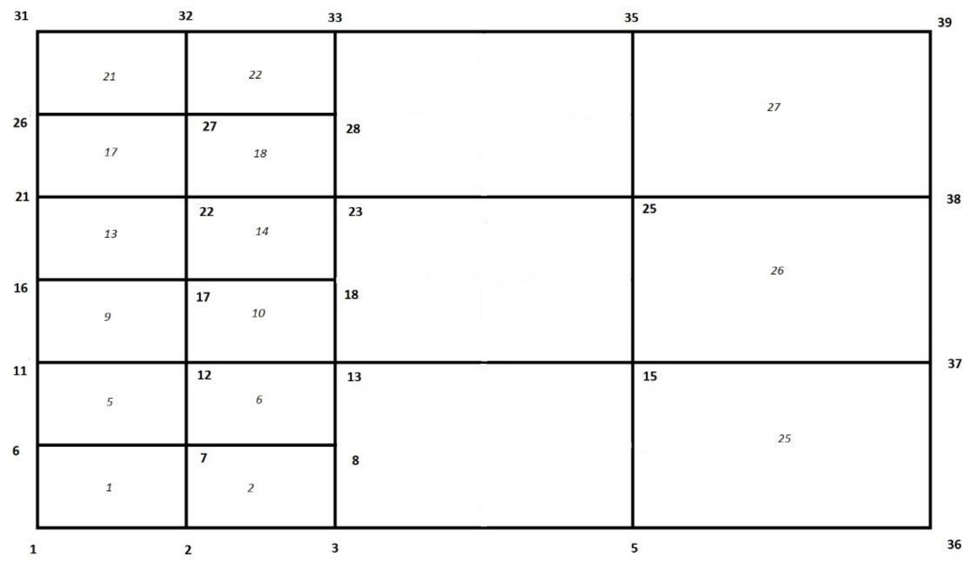

Figure 2-13. Cantilever beam with a tip load after maintaining connected node in three variable node super-elements

Figure 2-14 illustrates the contour plot for Von-Mises stress in the three variable node elements. Also, it can be seen that the stress contour plot follows the same format as the preceding variable super-elements; however, three variable node super-elements will provide a better approximation as opposed to using one variable node super-element, or even two variable node super-elements. The output result for displacements and stresses can be generated with the matlab program in Appendix 1.

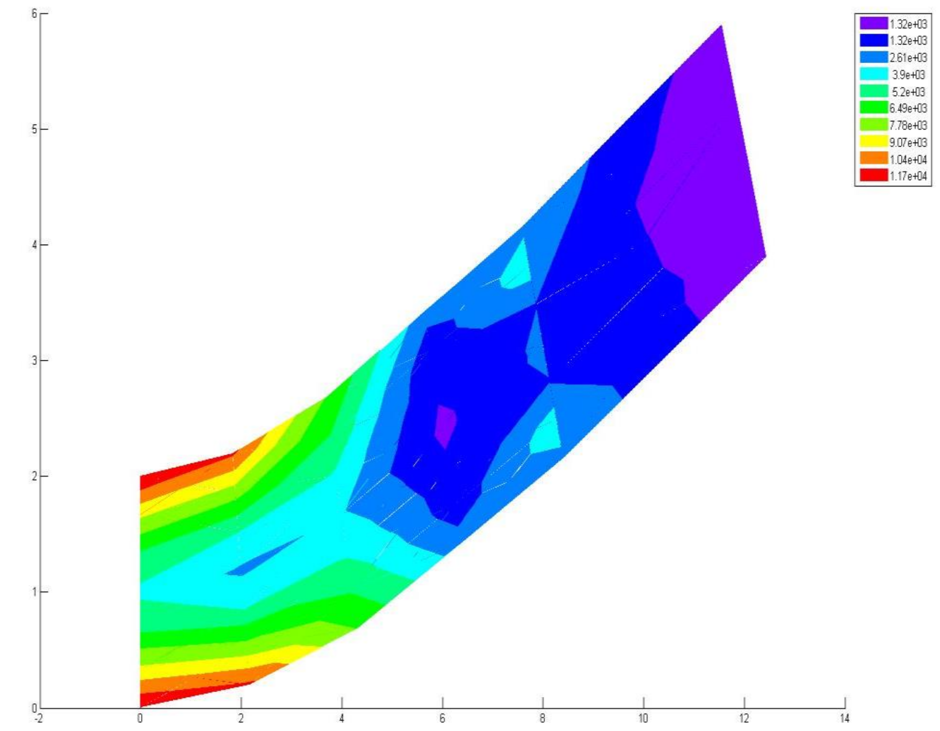

Figure 2-14. Von_Mises contour plot for cantilever beam with a tip load with three variable node super-elements

In this chapter, the basic derivation and the problem of a simple cantilever beam with a tip load at the end have been discussed.

CHAPTER 3

EXAMPLES AND VERIFICATIONS

Two plane strain examples are presented here to demonstrate the use of the MPC-based variable node super-element. One example is a cantilever beam loaded at the free end' and the other is a problem with stress concentration. Exact solutions are available in the literature for these two problems, which will be used to verify the numerical results. Each problem is discretized into four different mesh patterns. Two are using the standard quadrilateral elements; one with coarse mesh, the other with fine mesh. The other two meshes include mismatched elements transient from fine to coarse mesh. One has only one such transient zone and the other has two.

3.1 Cantilever beam with a tip load

A cantilever beam is treated to check the performance under generic loading. The solutions for the plane strain are given by Eqs. (3.1-3.2) [Lim, 2010,& Timoshenko, 1970]. Since the plane strain condition is considered in this example, *E* is substituted with *E/(1-v²)* and *v* with *v/(1-v),* respectively. The parameters such as *L = 8 mm*, *D = 1mm*, *E = 200000 Mpa.*, *v=0.3* and *P = 1N* are chosen where *D* and *L* are the depth and length of the beam.

$$U_x(x,y) = -\frac{Py}{6\frac{E}{(1-v^2)}I}\left\{(6L-3x)x + \left(2+\frac{v}{1-v}\right)\times\left(y^2-\frac{D^2}{4}\right)\right\} \tag{3.1}$$

$$U_y(x,y) = \frac{P}{6\frac{E}{(1-v^2)}I}\left\{3(\frac{v}{1-v})y^2(L-x) + \left(4+5(\frac{v}{1-v})\right)\frac{D^2x}{4} + (3L-x)x^2\right\} \tag{3.2}$$

Since the aim is to provide the displacements along the tip load (*x*=*l*), the simplified equations are given below:

$$U_x(y) = -\frac{Py}{6\frac{E}{(1-v^2)}I}\left\{(6L^2) + \left(2+\frac{v}{1-v}\right)\times\left(y^2-\frac{D^2}{4}\right)\right\} \tag{3.3}$$

$$U_y(y) = \frac{P}{6\frac{E}{(1-v^2)}I}\left\{\left(4+5(\frac{v}{1-v})\right)\frac{D^2L}{4}+2L^3\right\}. \tag{3.4}$$

The results based upon the exact solution for elongation over the nodes at the end of the cantilever beam, can be found in the Table 3-1.

The error norms can be calculated for the nodes along with tip load by the Eq. (3.5):

$$Re_d = (\sum_{i=1}^{nnode}((u_i^h - u_i^{exact})/(u_i^{exact}))^2 \,/\, \sum_{i=1}^{nnode} nnode)^{1/2}\ . \tag{3.5}$$

Basically, the error norm is used to compare the displacements in a desired area.

3.1.1 Coarse mesh

Figure 3-1 displays a coarse mesh model. In this model, the division of “W-span” and “S-span” are 8 and 10 respectively while the number of nodes and elements are 99 and 80 respectively. Figure 3-2 displays a contour plot for Von-Mises stress which technically shows where the highest stress is concentrated. By the way, since the Von-Mises stresses are symmetrically distributed, it can be obvious that they can be reliable results. However, in terms of coarse mesh since displacements are not very accurate due to the small stiffness matrix size and insufficient degrees of freedom in the domain, the stress contour plot will not be smooth enough.

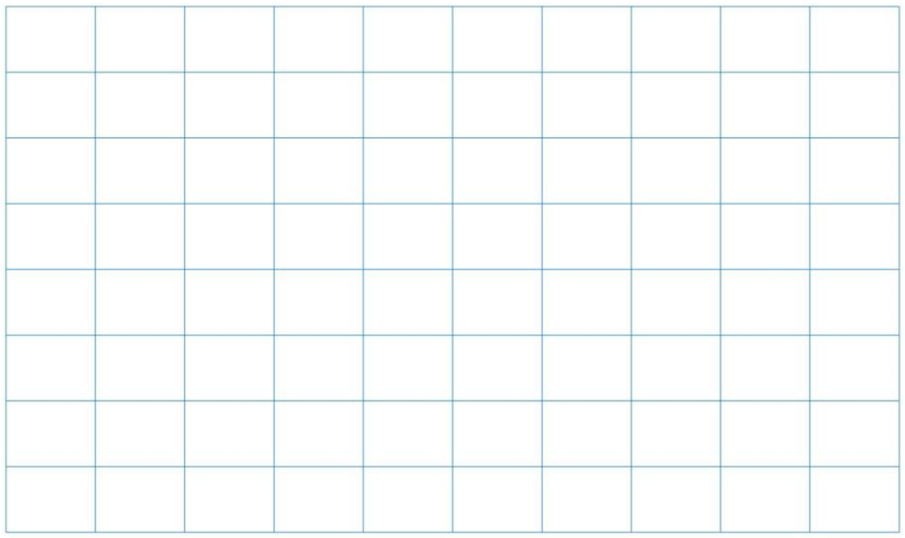

Figure 3-1. Coarse mesh plot for the cantilever beam with a tip load

The elongation output values are available in Table 3-1. The corresponding error norm in terms of coarse mesh is the largest value compared to the rest of the methods, although the number of nodes at the end of beam are less than the exact solution. In other words, if the number of “S-span” was less, and the number of “W-span” was more than the number of “S-span,” error norm in terms of elongation would have been a higher value. The Von-Mises stress can also be found at the constraint point at top left and bottom left where the support is located. In addition, in Figure 3-5 the fourth node of element 71 at the integration point has the highest Von-Mises stress. Also, since the method of curve fitting is used to convert stresses at Gaussian points of each element to the nodal points, the highest stress is at node 89. The results for Von-Mises stresses are in Table 3-2. Moreover, the curve fitting values will not be a very well-estimated method for the Von-Mises stress because the method is used for plotting the contour plot. However, consideration of nodal stresses will prove to be the closest to the exact solution.

3.1.2 Fine mesh

Figure 3-2 displays a fine mesh model. In this model, the division of “W-span” and “S-span” are 16 and 128. Also, the number of elements and number of nodes are 2048 and 2193, respectively. The Von-Mises contour plot in Figure 3-6 is shown; however, the finer the mesh the better the solution. By this it means that, in Figure 3-6, the maximum Von-Mises stress is 42.14 at node number 2065 by means of the curve fitting method. On the other hand, the fourth node of element 1921 is at the Gaussian point. So, in this case the fine mesh solution is much closer to the reference solution explained in Section 3-1. Nodal displacements are in Table 3-1 for elongation and Table 3-3 for deflection.

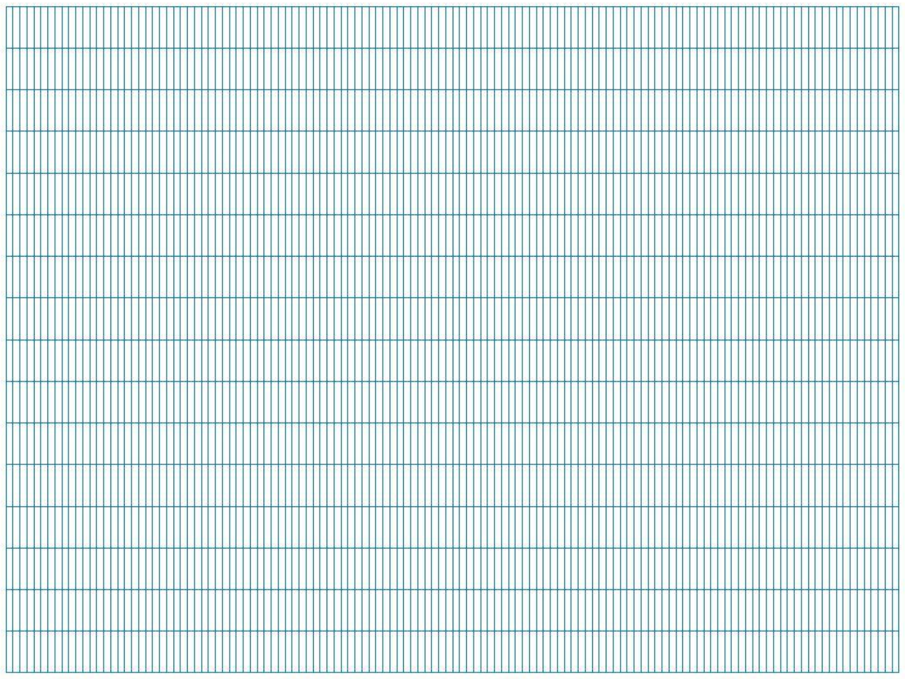

Figure 3-2. Fine mesh plot for a cantilever beam with a tip load

3.1.3 Quadrilateral elements with transition zones

In Figure 3-3 and Figure 3-4 it can be noticed that not only have transition zones been presented within the models but also the computational time will be reduced by using fewer total degrees of freedom in a model. In Figure 3-3, the fine mesh subdomain merges with the coarse mesh and in Figure 3-4 there are three different mesh resolutions considered: fine level, intermediate level, and coarse level.

In Figure 3-3, the number of nodes and elements are 1342 and 1240, respectively. Basically, eight variable node elements are located in the domain; each transition element contains four elements which, after derivation, will come up with one new variable node element. Therefore, every two fine elements on one side of a variable node element can merge with one coarse element. .

In the MPC-elimination method, every element has a 10 by 10 stiffness matrix after derivation' and that will be combined with regular quadrilateral elements which are 8 by 8 in size. In this method, only the element numbers within each variable node super-element will be given to the input' and the program will be deriving the K_{cc} stiffness matrix for each variable node element. Due to the presence of eight variable node elements in Figure 3-3, eight K_{cc} stiffness matrixes are constructed. Afterwards, by choosing the proper element identifier' which is used in the matlab code in Appendix 1, the new global stiffness matrix can be formed. Technically, the unconnected nodes and interior nodes of each variable node element will be constrained and the new global stiffness matrix size will be reduced. Then, unconnected nodes and interior nodes will be re-derived which has been previously discussed in Chapter 2.

In the MPC-penalty method, the global stiffness matrix will be formed by plugging the proper coefficient into the connected and unconnected degrees of freedom in the global stiffness matrix. Thus, a new stiffness matrix will be formed. In this method, the unconnected and interior node will be retained. The coefficients corresponding to connected and unconnected nodes in each variable node element will be 0.5 and -1, respectively. In each variable node element, there are three unconnected nodes, each of them being between two connected nodes. Since each node has two degrees of freedom, each variable node element should have six MPC inputs. However, the common MPCs in between two adjacent variable node elements will not be counted twice. As a result, 34 MPC inputs are considered for Figure 3-3.

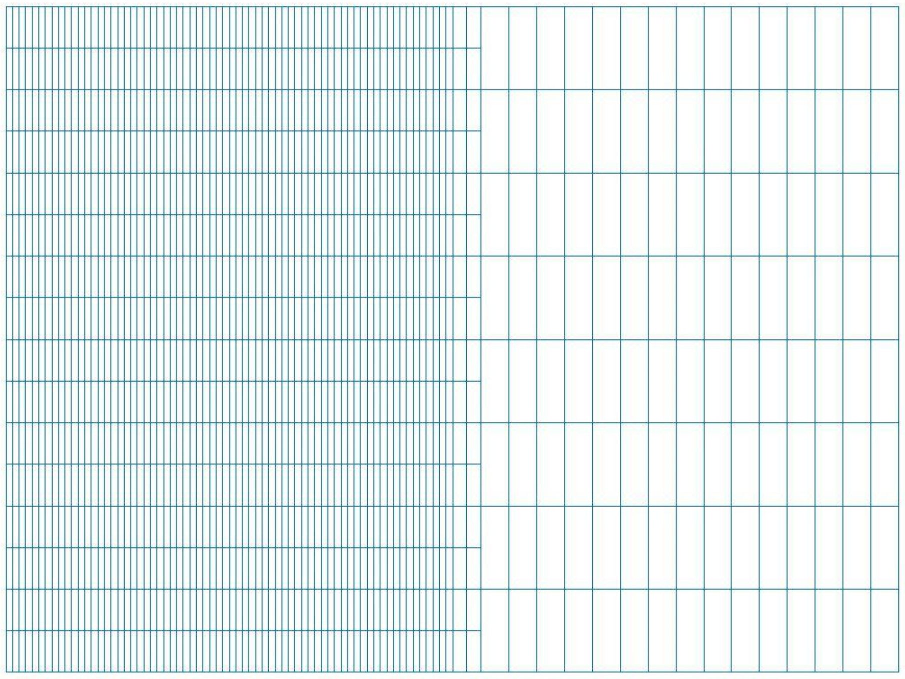

Figure 3-3. Mesh plot of cantilever beam with one transition zone

In Figure 3-4, the number of nodes and elements are 1395 and 1284. Basically, three different mesh resolutions are constructed within the domain such as fine mesh, intermediate mesh, and coarse mesh. In other words, the fine mesh level will merge with the intermediate mesh level with eight transition elements. In addition to this, the intermediate mesh level will merge with four transition elements. Therefore, all the transition elements will be 12. In the Elimination method, only 12 variable node super-elements will be constructed to provide a new global stiffness matrix, however, in the penlaty method 52 MPCs will be possessed in the input data.

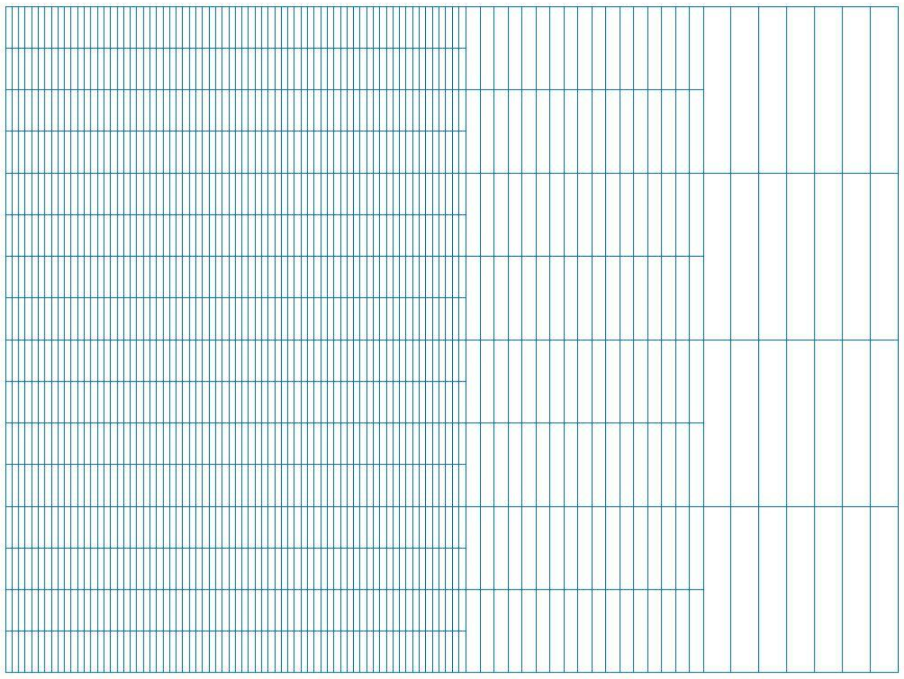

Figure 3-4. Mesh plot of cantilever beam with two transition zones

Generally, the penalty method is used to validate the Elimination method. Since, based upon the derivation, both methods are constructed to evaluate the unconnected nodes which are located in between two neighboring connected nodes in each variable node element, the unconnected node displacements will be the average of two neighboring connected nodes.

The stress contour plots of these four cases are listed in Figure (3-5) to Figure (3-10). It was the Von-Mises stress plotted here. The matlab code given in the textbook of Chandrupatla and Belegundu (2002) was modified to support the current study. The von Mises stress was reported at the four Gaussian integration points in each element. The nodal stresses were obtained through a curve fitting process in which the stress is assumed to be linearly distributed in an element. The nodal stresses so obtained were then used for the stress contour plot. The maximal stress happened at the top and the bottom corners of the beam at the fixed end. The maximal stresses reported in different cases were listed in Table 3-2 in comparison with the exact solution. The maximal displacements reported in different cases were also listed collectively in Table 3-3, along with the exact solution for comparison.

Table 3-1. The total U_x-displacement values at the end of beam in *mm*

	Exact_sol	***QUAD***	***QUAD***	***MPC_elim***	***MPC_elim.***	***MPC_pen.***
Y	***Ux***	***Fine***	***Coarse***	***1_Trans.***	***2_Trans.***	***1_Trans.***
-0.5	-8.736E-04	-8.73E-04	-7.14E-04	-8.64E-04	-8.66E-04	-8.64E-04
-0.4375	-7.638E-04	-7.62E-04				
-0.375	-6.543E-04	-6.53E-04	-5.34E-04	-6.47E-04		-6.47E-04
-0.3125	-5.449E-04	-5.44E-04				
-0.25	-4.358E-04	-4.35E-04	-3.56E-04	-4.31E-04	-4.32E-04	-4.31E-04
-0.1875	-3.267E-04	-3.26E-04				
-0.125	-2.178E-04	-2.17E-04	-1.78E-04	-2.15E-04		-2.15E-04
-0.0625	-1.089E-04	-1.09E-04				
0	0.000E+00	2.43E-15	1.01E-17	1.33E-16	1.16E-16	1.43E-09
0.0625	1.089E-04	1.09E-04				
0.125	2.178E-04	2.17E-04	1.78E-04	2.15E-04		2.15E-04
0.1875	3.267E-04	3.26E-04				
0.25	4.358E-04	4.35E-04	3.56E-04	4.31E-04	4.32E-04	4.31E-04
0.3125	5.449E-04	5.44E-04				
0.375	6.543E-04	6.53E-04	5.34E-04	6.47E-04		6.47E-04
0.4375	7.638E-04	7.62E-04				
0.5	8.736E-04	8.73E-04	7.14E-04	8.64E-04	8.66E-04	8.64E-04
Error Norm with respect to exact sol.						
		2.28E-03	**7.33E-02**	**7.17E-03**	**1.20E-04**	**1.72E-04**

Table 3-2. Maximal Von-Mises stresses for cantilever beam

Different Mesh Model	Exact-sol.	Gaussian_point	Nodal_value
Exact Solution	48.01920768		
Fine mesh		40.9600	42.1436
Coarse mesh		34.1900	37.0010
MPC-elim._1		40.9710	42.1502
MPC-elim._2		40.9710	42.1502
MPC-pen._1		40.9590	42.1377
MPC-pen._2		40.9630	42.1419

Table 3-3. Maximal U_y-displacement at the free end in *mm*

Differnet Mesh Model	U_y
Exact Solution	-9.43E-03
Fine mesh	-9.41E-03
Coarse mesh	-7.71E-03
MPC-elim._1	-9.38E-03
MPC_elim_2	-9.40E-03
MPC_pen._1	-9.38E-03
MPC_pen_2	-9.40E-03

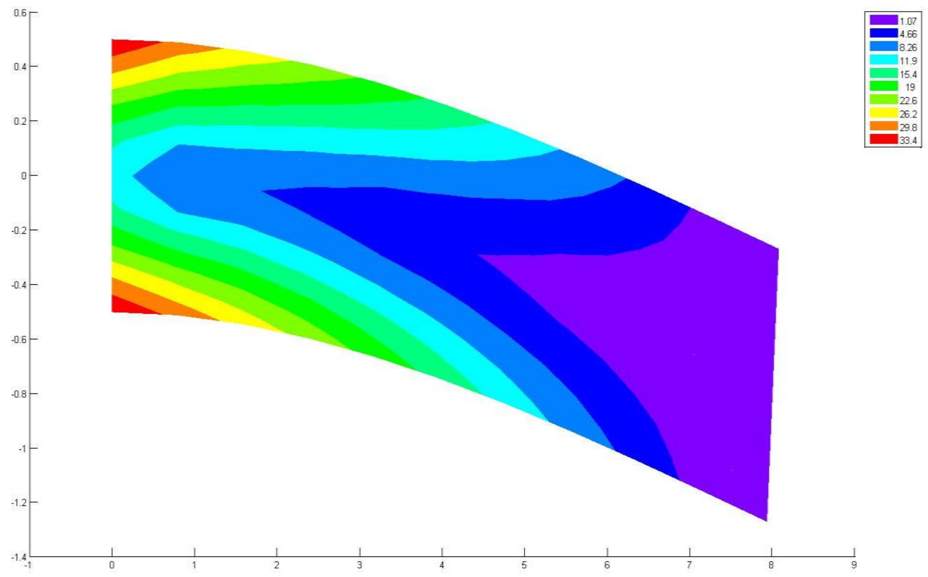

Figure 3-5. Von_Mises stress contour plot for the coarse mesh

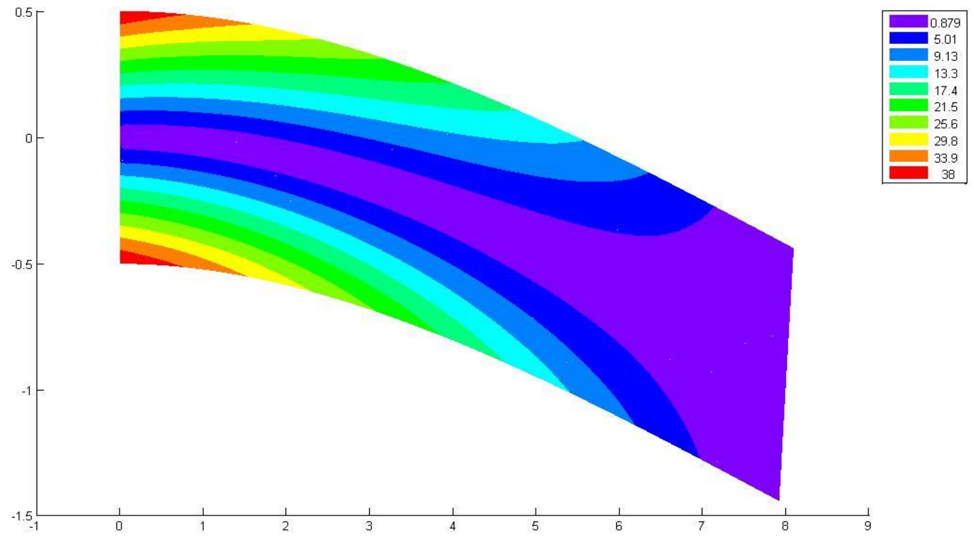

Figure 3-6. Von-Mises contour plot for the fine mesh

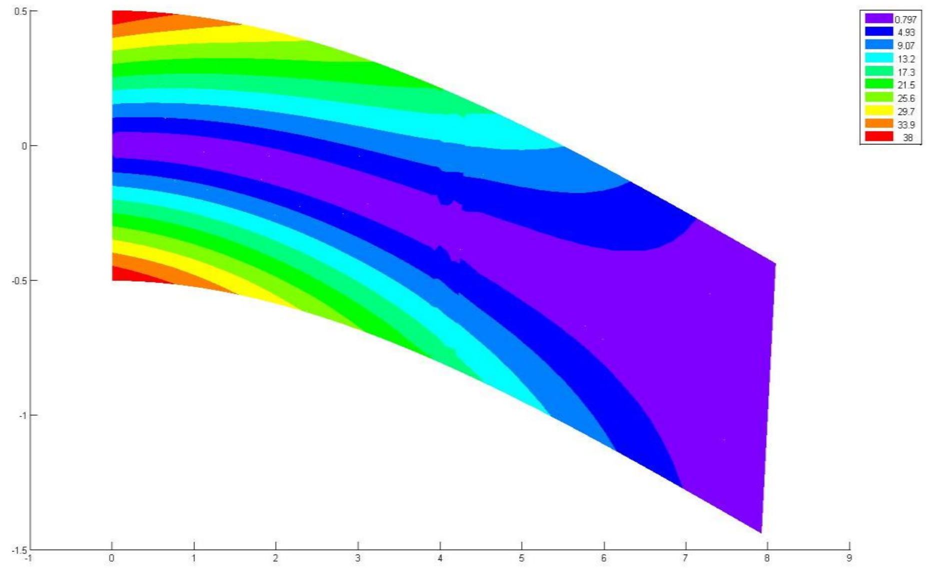

Figure 3-7. Von-Mises contour plot for MPC-elimination with one transition zone

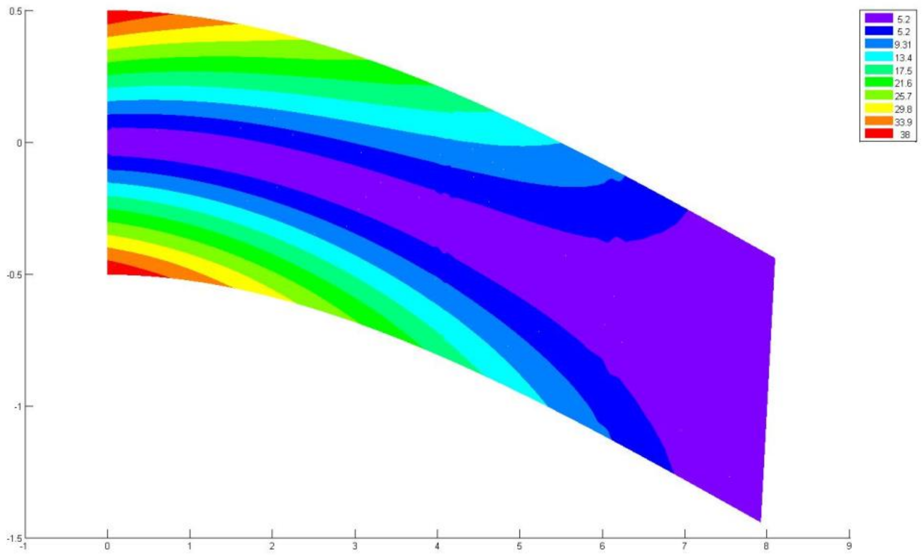

Figure 3-8. Von-Mises contour plot for MPC-elimination with two transition zones

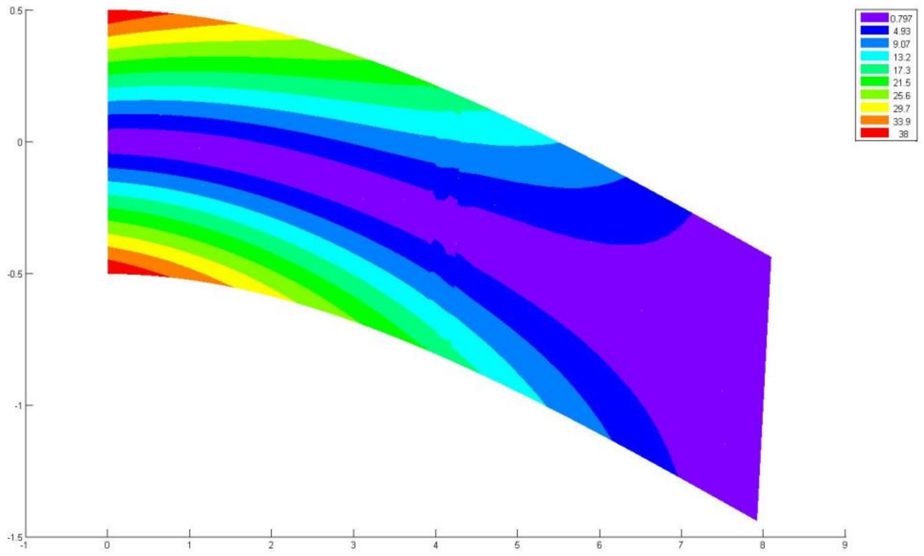

Figure 3-9. Von-Mises contour plot for MPC-penalty with one transition zone

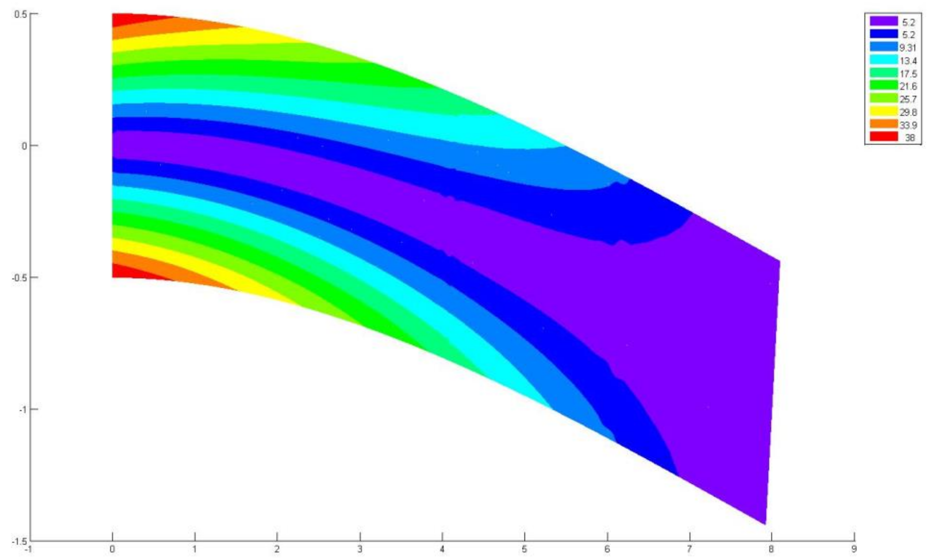

Figure 3-10. Von-Mises contour plot for MPC-penalty with two transition zones

3.2 Infinite plate with a circular hole

Figure 3-11 represents an infinite plate with a central circular hole where the radius is $a = 1\ m$, and is subject to an unidirectional tensile load of $T = 1.0\ {}^{N}\!/_{m}$ at infinity in the x-direction. The plate is now set as a $L\times L$ square for finite element analysis. Due to its symmetry, only the upper right quadrant of the plate is modeled. Figure 3-12 represents the quadrant plate with a hole which is set to be a 12×12 meter square. In this circumstance, the plane strain condition is considered. Moreover, the modulus of elasticity and passion ratios are $E = 10^3\ {}^{N}\!/_{m^2}$ and $v = 0.3$, respectively. Symmetric conditions are imposed on the left as well as the bottom edges, while the inner boundary of the hole is traction free. [Lui, 2009, & Timoshenko 1970].

$$\sigma_r = 1 - \frac{a^2}{r^2}\left[\frac{3}{2}\cos 2\theta + \cos 4\theta\right] + \frac{3a^4}{2r^4}\cos 4\theta \tag{3.6}$$

$$\sigma_\theta = -\frac{a^2}{r^2}\left[\frac{1}{2}\cos 2\theta - \cos 4\theta\right] - \frac{3a^4}{2r^4}\cos 4\theta \tag{3.7}$$

$$\tau_{r\theta} = -\frac{a^2}{r^2}\left[\frac{1}{2}\sin 2\theta - \sin 4\theta\right] + \frac{3a^4}{2r^4}\sin 4\theta \tag{3.8}$$

where (r, θ) are the polar coordinates and θ is the measured counterclockwise from the positive *x*-axis. Traction boundary conditions are imposed on the right and top edges based on the exact solutions.

The displacement components corresponding to the stresses in terms of polar coordinate are given in the equations below [Lui, 2009, & Timoshenko 1970]:

$$U_r(r,\theta) = \frac{a}{8\mu}\left[\frac{r}{a}(k+1)cos\theta + 2\frac{a}{r}\big((1+k)cos\theta + cos3\theta\big) - 2\frac{a^3}{r^3}cos3\theta\right] \tag{3.9}$$

$$U_\theta(r,\theta) = \frac{a}{8\mu}\left[\frac{r}{a}(k-3)sin\theta + 2\frac{a}{r}\big((1-k)sin\theta + sin3\theta\big) - 2\frac{a^3}{r^3}sin3\theta\right] \tag{3.10}$$

where $\mu = \dfrac{E}{(1+\nu)}$ and $k = 3 - 4v$ are defined in terms of Poisson's ratio for "plane strain" condition. In theory, the stress concentration factor is a function of the ratio, (a/L). For an infinite plate, the stress concentration factor is 3, as (a/L) approaches to zero. Roark's formulas [Young and Budynas, 2002] gives the following curve-fitting equation for the stress concentration,

$$\frac{\sigma}{T} = 3.0 - 3.13\left(\frac{a}{L}\right) + 3.66\left(\frac{a}{L}\right)^2 - 1.53\left(\frac{a}{L}\right)^3 \tag{3.11}$$

which yields a maximal stress of 2.763 for (α/L).=1/12.

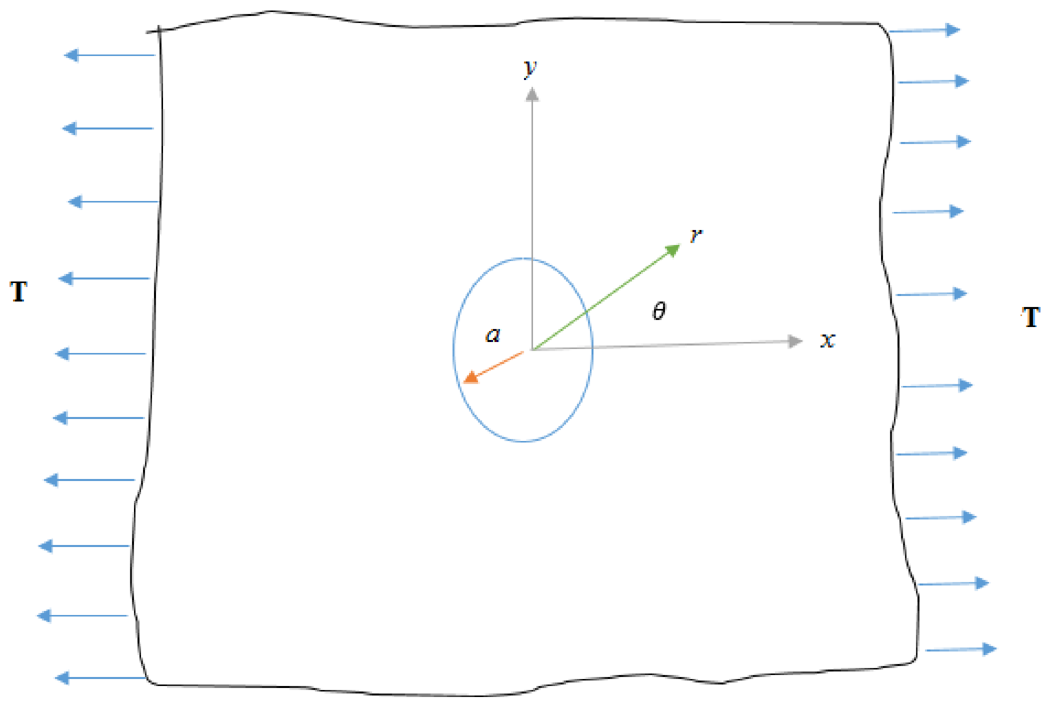

Figure 3-11. Infinite plate with a circular hole subjected to unidirectional tension

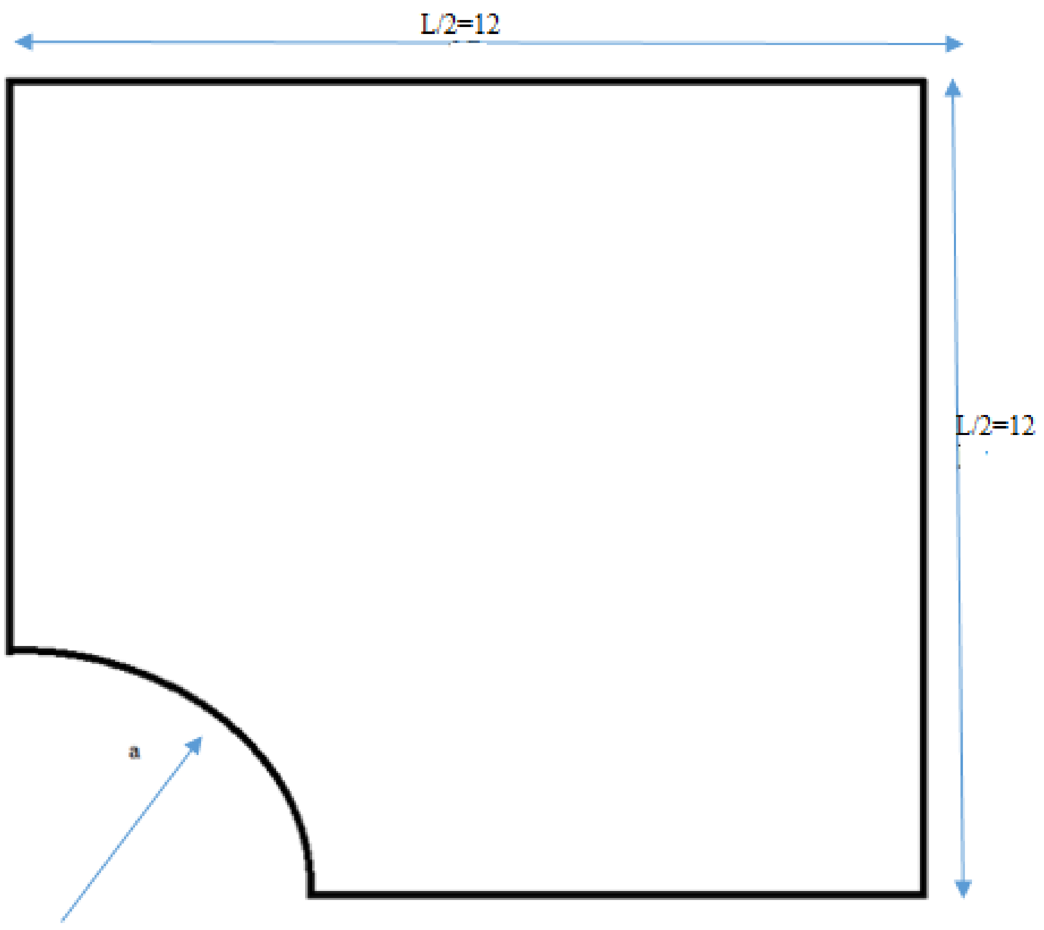

Figure 3-12. Quadrant of an infinite plate with a hole

3.2.1 Coarse mesh

Figure 3-13 represents the coarse mesh of the quadrant plate. In the figure below, the “W-span” and “S-span” divisions are equal to 12 and 6. In this case, the numbers of nodes and elements are 91 and 72, respectively.

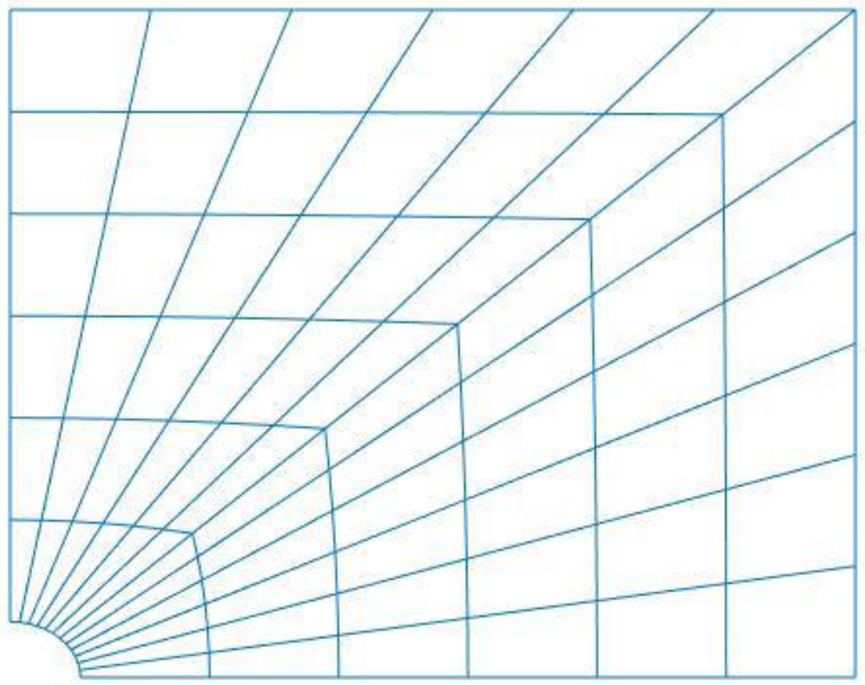

Figure 3-13. Coarse mesh for a plate with a hole

Figure 3-14 presents the Von-Mises contour plot for a coarse mesh. On the left bottom of the circular hole the highest Von-Mises stress can be seen. The Von-Mises stress for coarse mesh is given in Table 3.5. The nodal value is slightly larger than the integration value due to the fact that the nodal point is closer to the high stress area by the method of curve fitting.

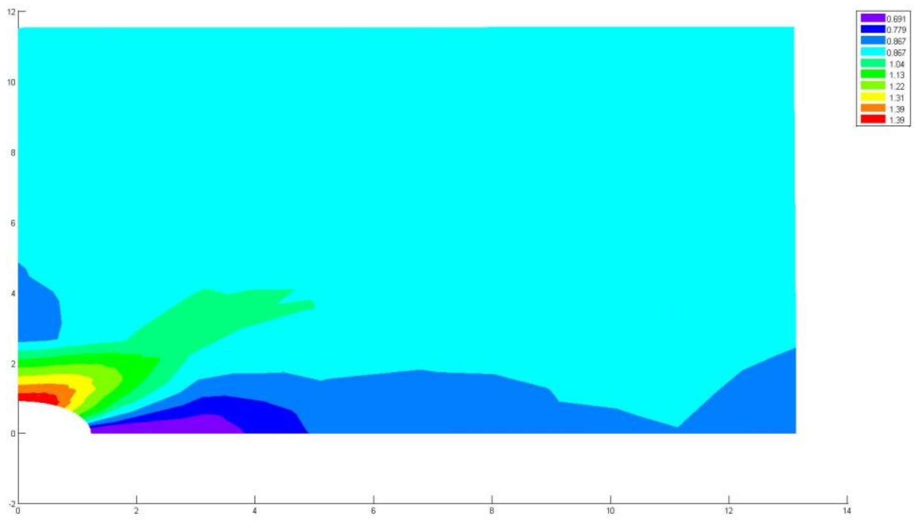

Figure 3-14. Von-Mises contour plot for a coarse mesh

3.2.2 Fine mesh

Figure 3-15 represents the fine mesh of a plate with a hole. The total number of nodes and elements are 1221 and 1152 respectively. Also, the forces and boundary conditions are distributed more than the coarse mesh.

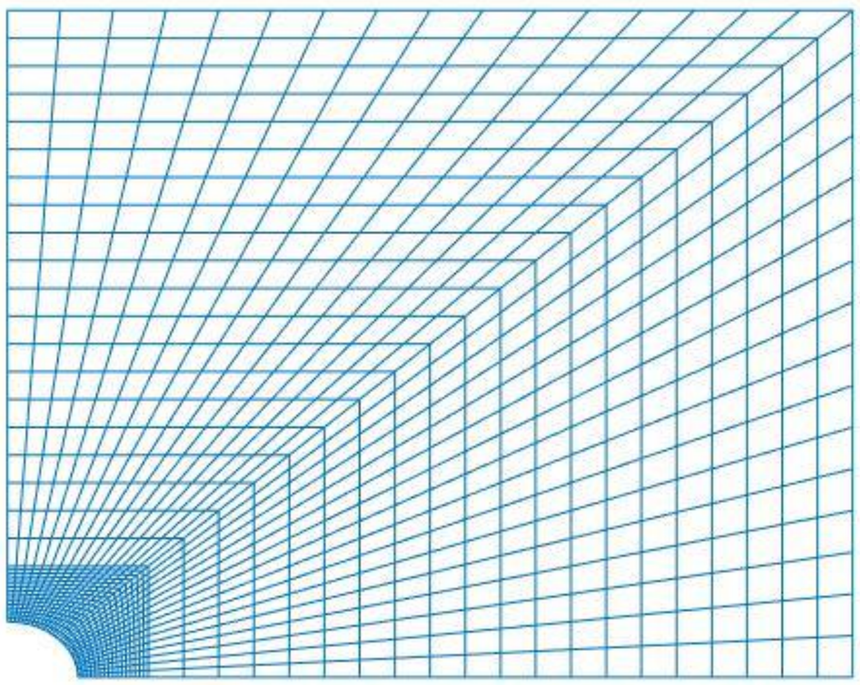

Figure 3-15. Fine mesh for a plate with a hole

After solving the problem, Figure 3-16 shows the Von-Mises contour plot in terms of nodal Von-Mises stress values. However, in comparison to the coarse mesh in Figure 3-14 the highest Von-Mises stress in the fine mesh has a closer value to the Von-Mises analytical solution.

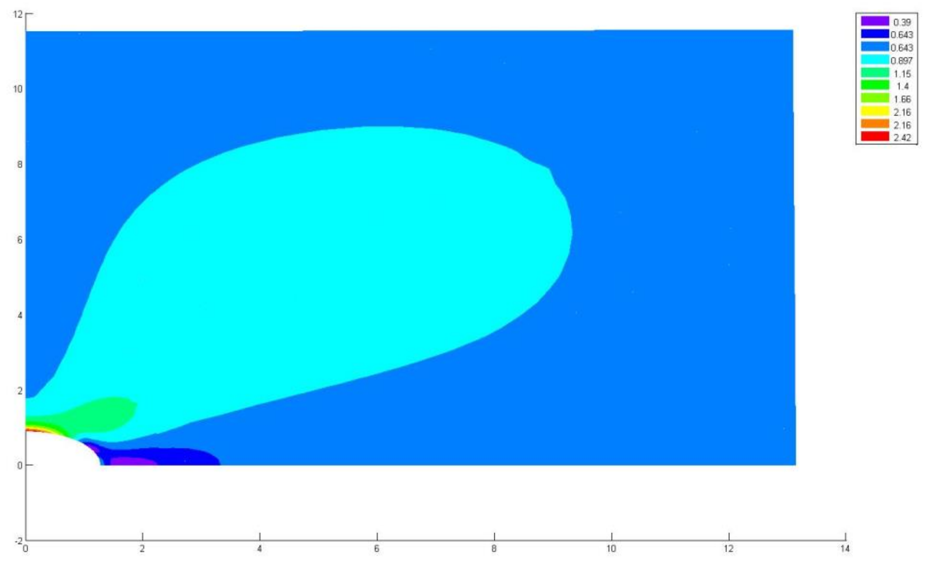

Figure 3-16. Von-Mises contour plot fine mesh for a plate with a hole

3.2.3 Quadrilateral elements with transition zones

Figure 3-17 shows a plate with a hole which is divided into two subdomains. The first subdomain has fine elements' and the second subdomain consists of coarse elements. The total number of nodes and elements are 818 and 752, respectively. 18 variable node elements are considered for one set of transition elements problem and each variable node super-element comprises 4 elements. In all variable node super-elements in Figure 3-17, 33 unconnected nodes are available; since each node has two degrees of freedom, there will be 66 MPCs.

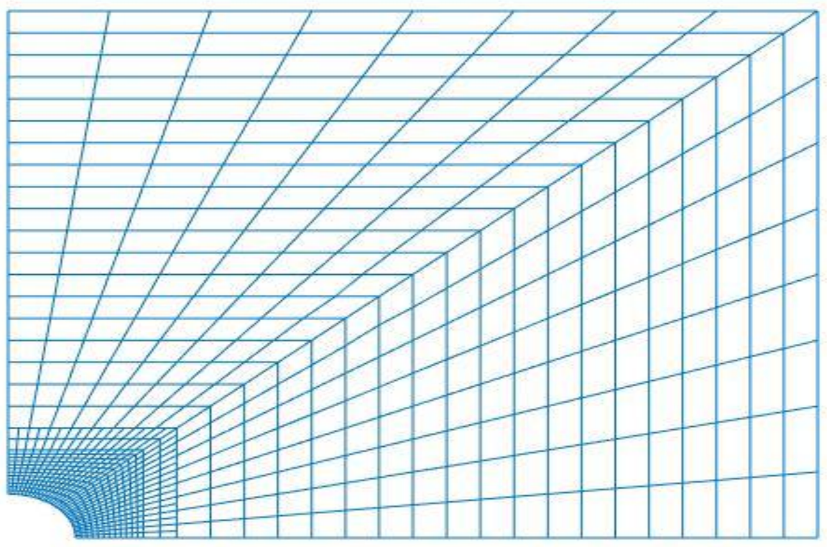

Figure 3-17. Plate with a hole with one transition zone

Figure 3-18 shows a plate with a hole which is divided into three subdomains. The first subdomain has fine elements and the second and third subdomains consist of intermediate and coarse elements. The total number of nodes and elements are 644 and 584, respectively. 24 variable node elements are considered for two transition zones problem and each variable node element includes 4 elements. In all variable node super-elements in Figure 3.18, 50 unconnected nodes are available; since each node has two degrees of freedom, there will be 100 MPCs.

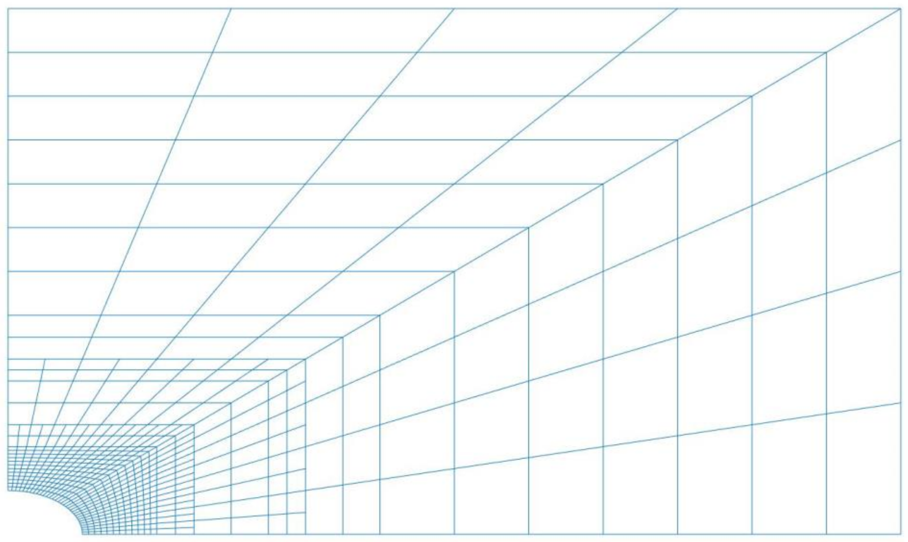

Figure 3-18. Plate with a hole with two transition zones

Figure 3-19 represents the Von-Mises contour plot in terms of Elimination which is very similar to Figure 3-21. Both Figures are the result of one transition problem. In Table 3-5 the Von-Mises stress in the MPC-elimination method and the MPC-penalty method are identical. Also, according to displacement values in Table 3.4 over the right boundary of a plate with a hole, both methods have ended up with the same solutions.

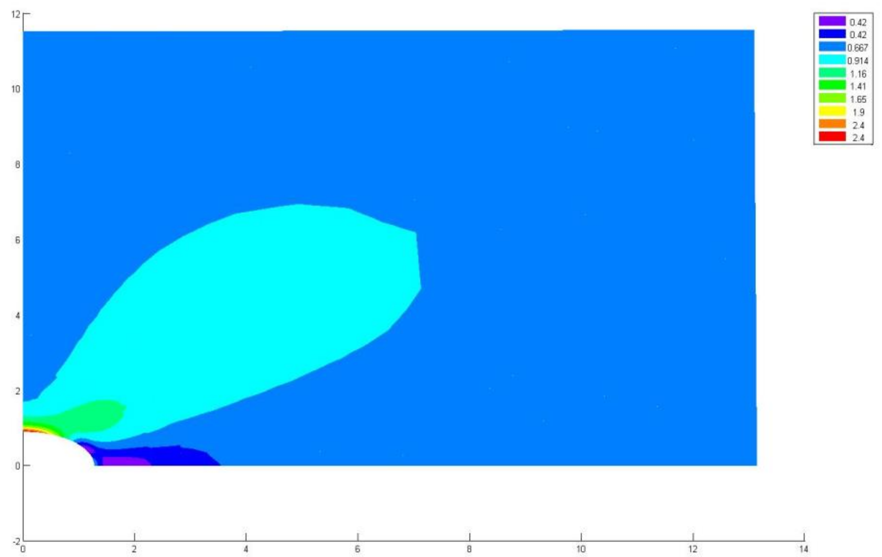

Figure 3-19. Von-Mises stress contour plot for MPC-elimination with one transition zone

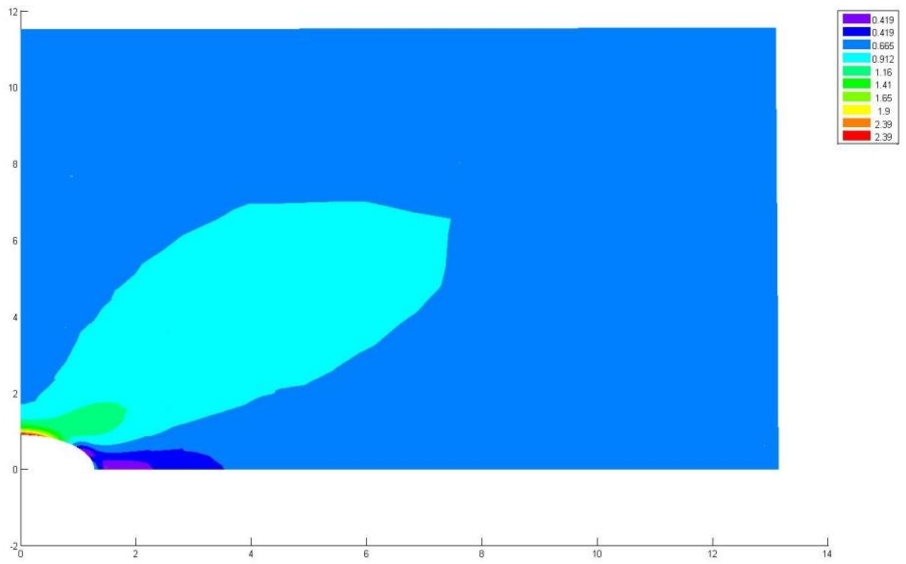

Figure 3-20. Von-Mises stress contour plot for MPC-elimination with two transition zones

Figure 3-20 represents the Von-Mises contour plot in terms of Elimination which is pretty similar to Figure 3-22. Both figures are the result of two transitions. In Table 3.5

the Von-Mises stress in the MPC-elimination method and the MPC-penalty method are identical. Also, according to displacement values in Table 3.4 over the right boundary of a plate with a hole, both methods ended up with the same solutions.

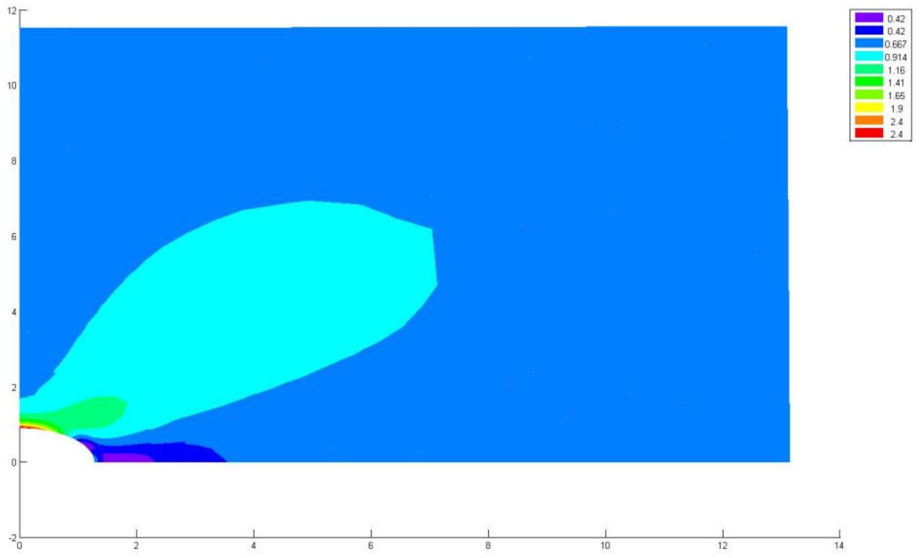

Figure 3-21. Von-Mises stress contour plot for MPC-penalty with one transition zone

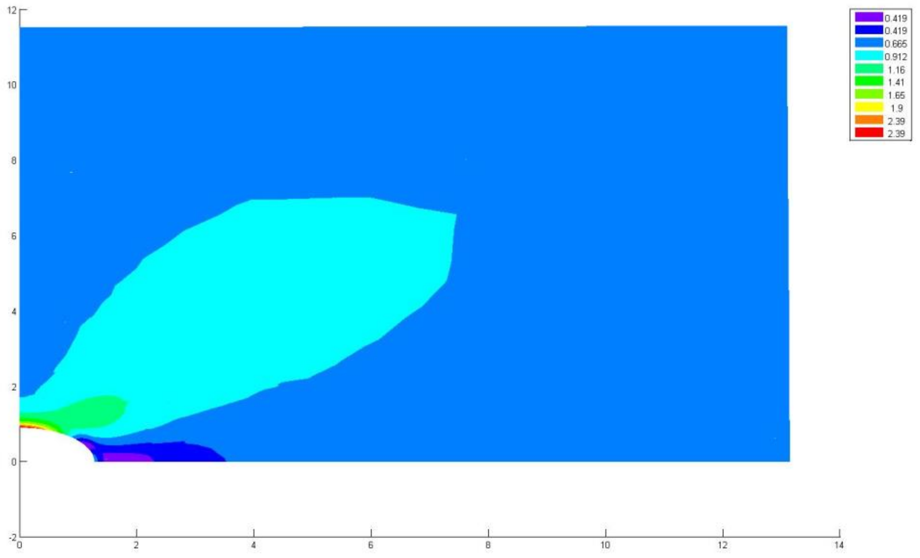

Figure 3-22. Von-Mises stress contour plot for MPC-penalty with two transition zones

Table 3-4. U_y -displacements at the end of the plate

Y (X=12m)	***Exact-sol.***	***Fine***	***Coarse***	***MPC-elim-1***	***MPC-elim-2***	***MPC-pen-1***
0	0.011125	0.011307	0.011265	0.01130717	0.01130837	0.011307171
0.75	0.011124	0.011303				
1.5	0.011119	0.011291	0.010429	0.0112906		0.011290604
2.25	0.011111	0.011271				
3	0.011101	0.011245	0.011209	0.01124512	0.011247231	0.011245123
3.75	0.01109	0.011214				
4.5	0.011077	0.011181	0.011152	0.01118094		0.011180939
5.25	0.011063	0.011145				
6	0.01105	0.011109	0.011088	0.01110904	0.011111923	0.011109035
6.75	0.011037	0.011072				
7.5	0.011024	0.011037	0.011025	0.0110371		0.011037098
8.25	0.011013	0.011002				
9	0.011002	0.010968	0.010964	0.01096841	0.010970017	0.010968406
9.75	0.010992	0.010935				
10.5	0.010983	0.010903	0.010906	0.01090294		0.010902944
11.25	0.010976	0.010871				
12	0.010969	0.01084	0.01085	0.01083916	0.010837824	0.010839164
Error_Norm		**0.000104**	**0.000482**	**0.00011036**	**0.000125041**	**0.000110357**

Table 3-5. Maximal Von-Mises stresses for the plate

Different Mesh Model	Exact_Solution	Gaussian_point	Nodal_value
Exact_solution for an infinite plate	3		
Analytical solution for (a/L)=1/12	2.763		
Fine mesh		2.5632	2.670481
Coarse mesh		1.6291	1.829874
MPC_elim_1		2.5136	2.642626
MPC_elim_2		2.5091	2.637845
MPC_pen_1		2.5136	2.642626
MPC_pen_2		2.5091	2.637845

Considering Table 3-5, it can be concluded that the MPC-elimination elicits the same solutions exactly. Not only is the fourth node at Gaussian integration of the corresponding element close to the exact solution within an error of less than 8 percent, but also the nodal value will be a good estimate for the Von-Mises stress, with the error within 3 percent.

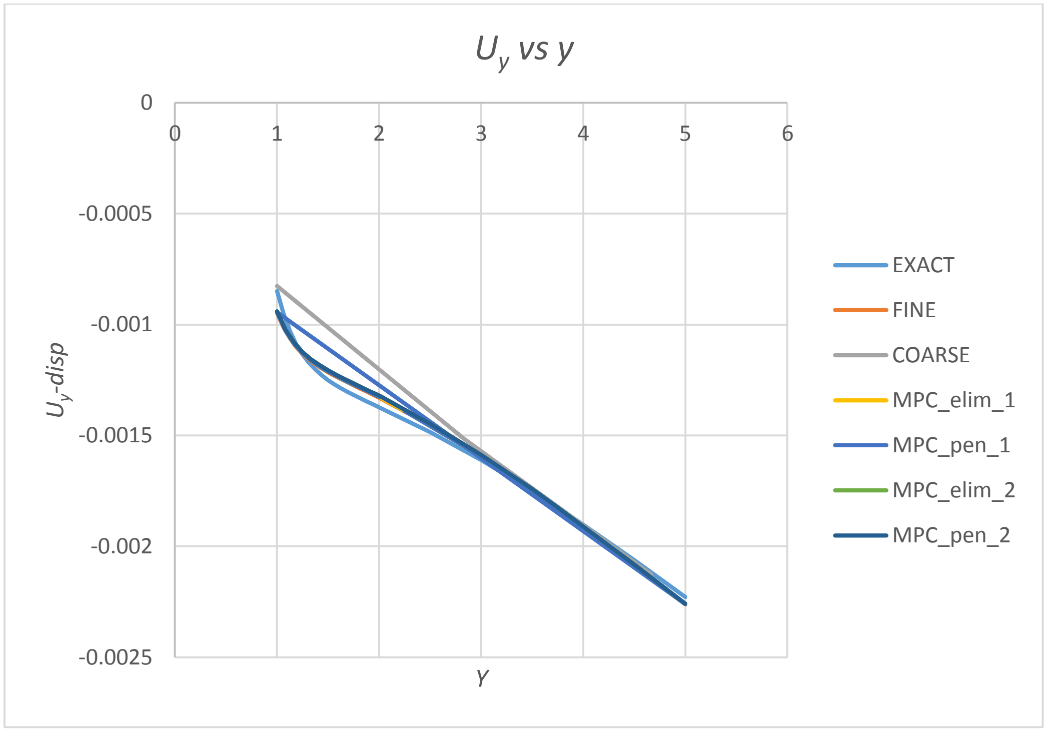

Figure 3-23 U_y along the left edge of the plate model

Figure 3-23 shows that by progressing along the left boundary, the deflection will be reduced. However, their absolute values will be incremental. In other words, the closer a point is to the central hole, the less displacement will be expected. Basically, all the finite element method solutions in terms of U_y nearly fall on each other. The maximum error for U_y between the finite element methods and reference solution according to the figure above is *0.5* percent.

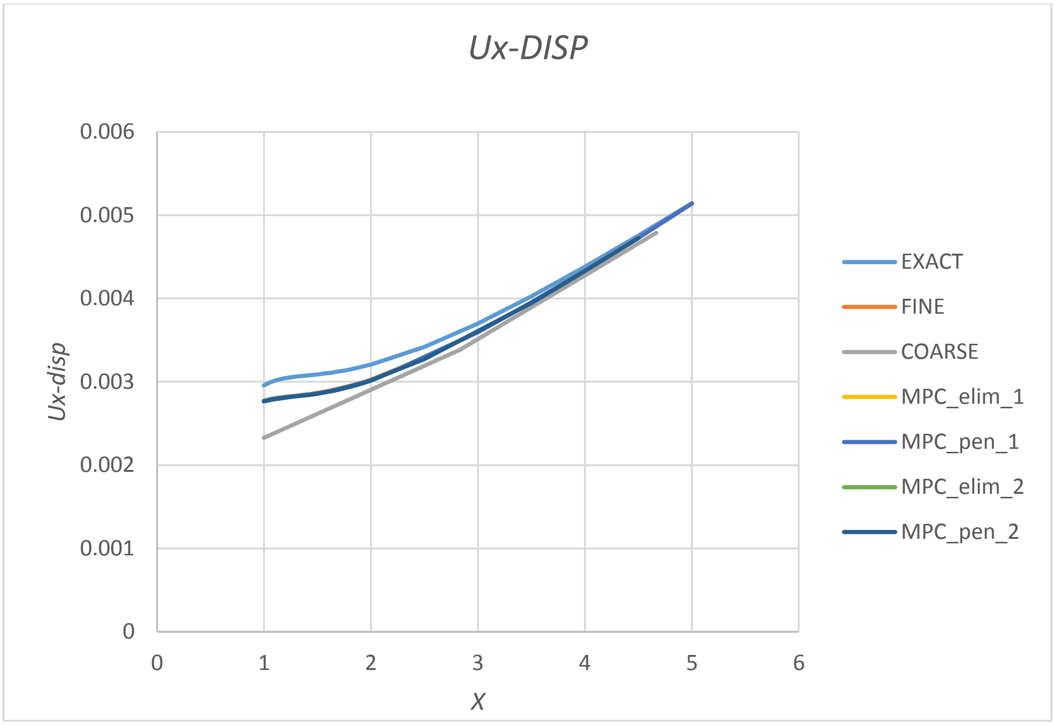

Figure 3-24. U_x along the bottom edge of the plate model

Figure 3-24 signifies the elongation over the lower boundary which starts at the central hole and goes to the coordinate of $X=12$ and $Y=0$. This demonstrates that by continuing to make progress along the X axis, the U_x-displacements increase. All the finite element method solutions in terms of U_x are nearly fall on each other. The Maximum error for U_x between the finite element methods and reference solution the according to the Figure above is less than 0.5 percent.

CHAPTER 4

CONCLUDING REMARKS

In this chapter, the work performed in this study, and the ideas behind it, will be summarized. Also, the efficiency and priority of the method will be discussed here.

4.1 Summary of the work

The MPC-elimination method and the MPC-penalty method were discussed in Chapter 2. Both methods are used in the domains with transition elements. The idea driving the MPC-elimination method is to construct the variable node super-elements in multi-scale problems. After conducting and assembling the new global stiffness matrix including the connected nodes of variable node super-elements, the interior and unconnected nodes were constrained. Then, the displacements of unconnected and interior nodes were revived based upon the derivation in Chapter 2.

4.2 Conclusions

The derivation of the MPC-elimination method was introduced in Chapter 2. Also, several examples were used that revolve around two particularly challenging problems. In this study, the cantilever beam with a tip load as well as a plate with a central hole, were the challenges addressed and modelled in this work. In addition, the validation of data in terms of displacements and Von-Mises stresses were surveyed in the preceding examples also. The MPC-variable node super-element was used for the models which comprise any four noded quadrilateral elements with transition elements.

Since the models with transition zones have fewer degrees of freedom compared to models that do not have transition elements, the size of the global stiffness matrix of the domain was reduced. By reducing the global stiffness matrix size, computational time was reduced. In addition to reducing computational time, the Elimination method can solve multi-scale problems by constructing five noded variable node super-elements with the connected nodes within transition zones. The MPC-elimination method constructs the local stiffness matrix of each local variable node super-element in the model. Typically, the stiffness matrix of each variable node super-element is derived, and contains ten degrees of freedom due to the elimination of the unconnected and interior nodes. Again, the derivation of the stiffness matrix of variable node super-elements was covered in Chapter 2. As a result, the MPC-elimination method can solve multi-scale two dimensional problems with quadrilateral elements.

4.3 Suggestion for future work

Only a (1+4)-variable node element with 4-node quadrilateral elements was introduced here. The procedure developed in Chapter 2 is simple but quite general and can be extended to other types of variable node super-elements. For example, a (2+4)-variable node super-element can be generated by a collection of six 4-node quadrilateral elements as shown in Figure 4-1. This (2+4)-node super-element can be used along with a (1+4)-one to handle the case even when the corner nodes of the mismatched quadrilateral elements don’t join together. This scenario is shown in Figure 4-2.

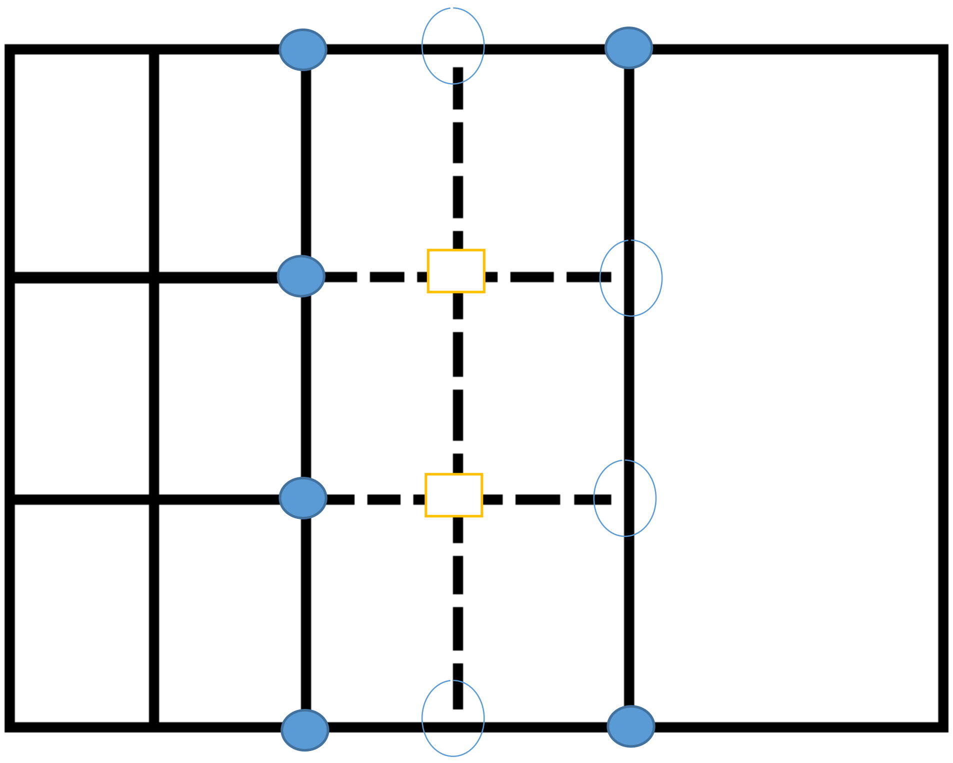

Figure 4-1. (2+4)-node super-element: (a) ●: the connected node, (b) ○: the unconnected node, and (c) □: the interior node

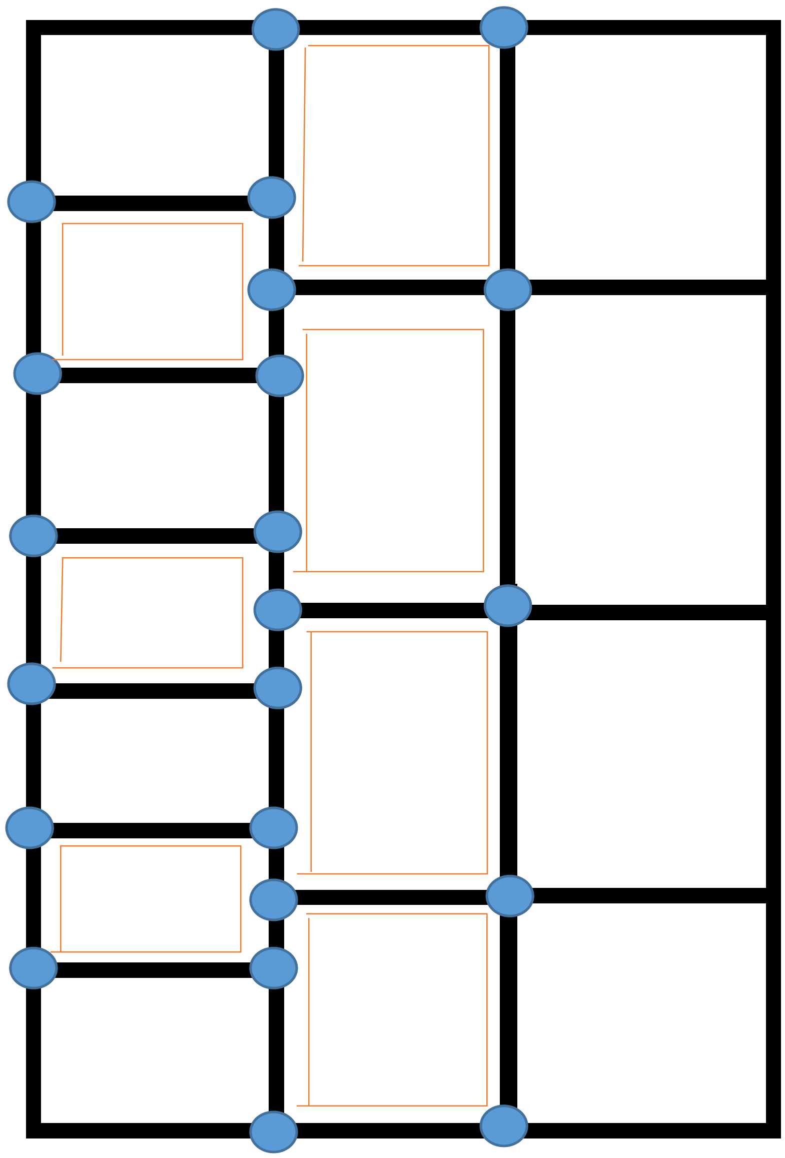

Figure 4-2. Combination of (1+4) - and (2+4)-node super-element for transition between mismatched elements

High order variable node super-elements can also be developed based upon the procedure described in Chapter 2. For example, a single 8-node quadrilateral element can be a variable node element to connect 8-node quadrilateral element to a 4-node one. This is done by imposing a MPC constraint along its edge that is connected to the 4-node quadrilateral element. Similarly, a pair of two 8-node quadrilateral elements can form a variable node super-element which connected two 8-node quadrilateral elements to one. These two scenarios are presented in Figure 4-3 and Figure 4-4.

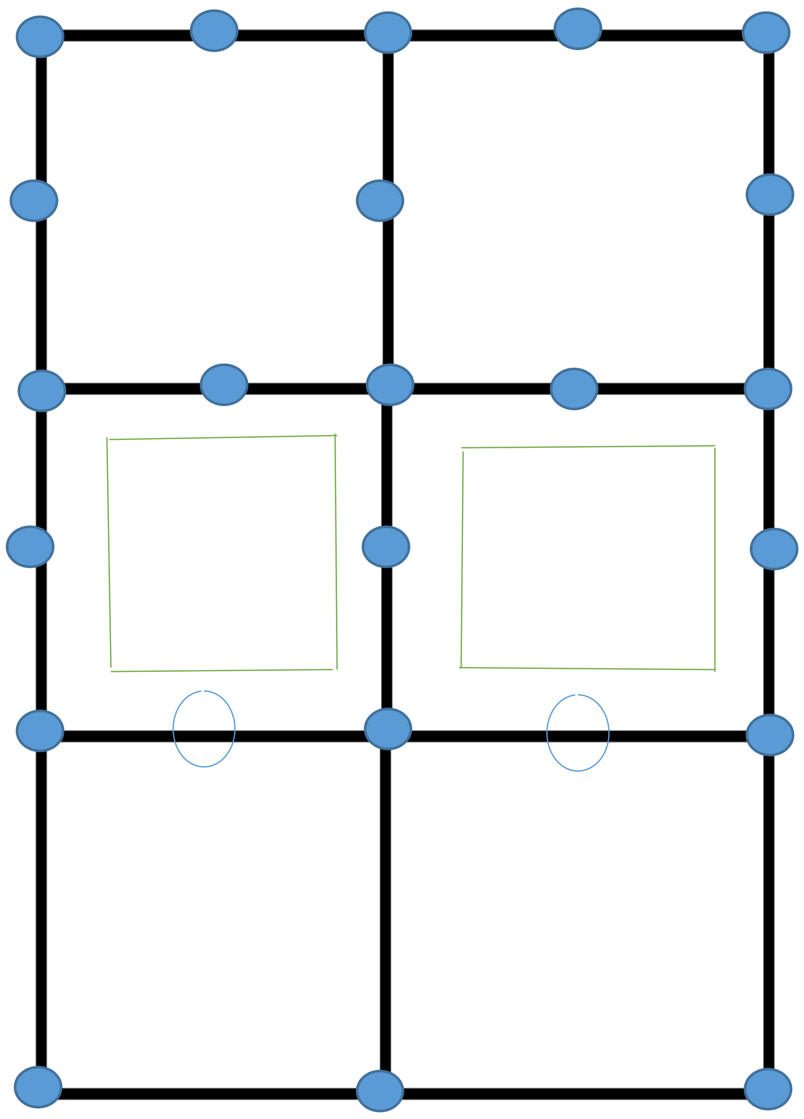

Figure 4-3. Constraint 8-node quadrilateral element

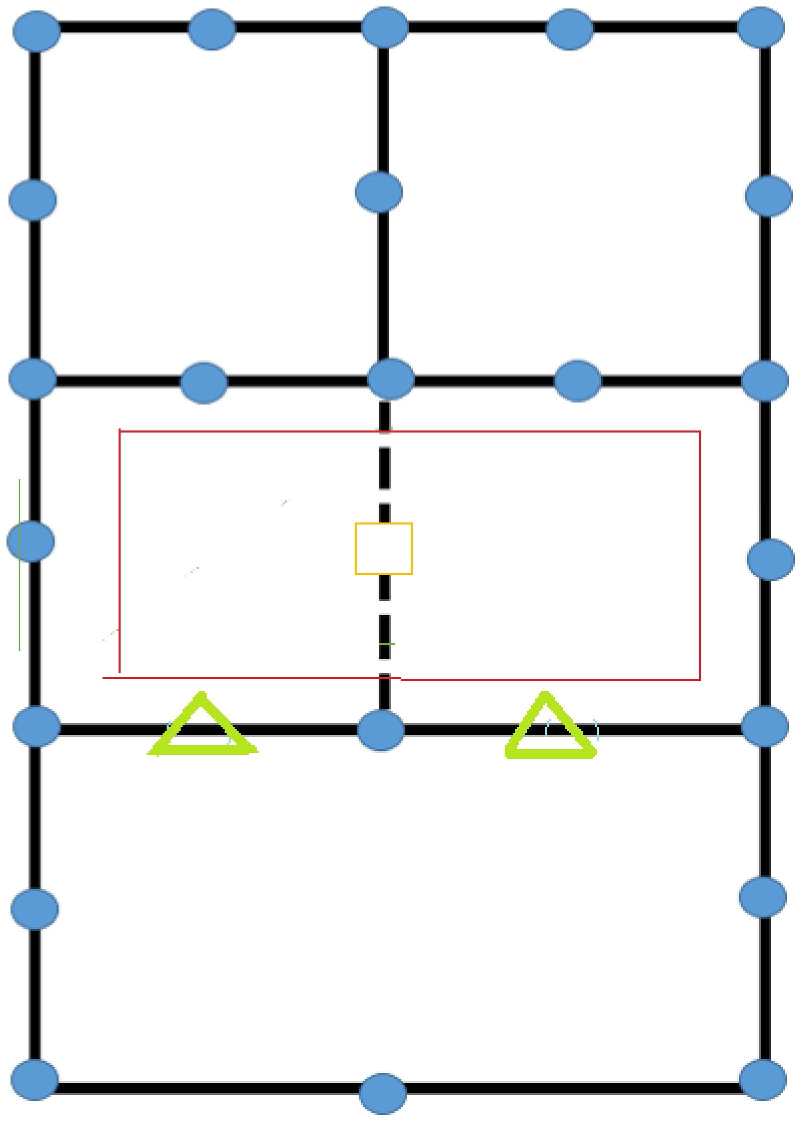

Figure 4-4. (2+8)-node variable node super-element

REFERENCES

1. Aminpour, M., Pageau, S., and Shin, Y., “Improved interface modeling technology”, *American institute of aeronautics & astronautics*, 2001, pp. 16-19
2. Barber, J. R., Elasticity, *Kluwer academic publisher,* 2nd. ed, 2002. Print.
3. Chandrupatla, T. R., and Belegundu A. D., Introduction to finite elements in engineering, *Prentice Hall,* 3rd. ed, 2002. Print.
4. Cho, Y., and Im, S., “MLS-based variable-node elements compatible with quadratic interpolation. Part I: formulation and application for non-matching meshes”, *International journal for numerical methods in engineering,* Vol. 65, 2005, pp.494-516
5. Cho, Y., Jun S., Im, S., and Kim, H., “An improved interface element with variable nodes for non-matching finite element meshes”, *Comput. methods appl. mech. engrg.*, Vol. 194, 2005, pp. 3022-304
6. Du, Y. G., Chan, A. H. C., LClarck, L. A., Wang, X. T , Gurkalo, F., and Bartos, S., “Finite element analysis of cracking and delamination of concrete beam due to steel corrosion”, *Engineering structures*, Vol. 56, 2013, pp. 8-21
7. Gill, P., Davey, K., “Analysis of thermo-mechanical behavior of a crack using XEFM for Leak-before-break assessments”, *International journal of solids and structures*, Vol. 51, 2014, pp. 2062-2072In
8. Kim, H. G., “interface element method for a partitioned system with non-matching interfaces”, Computer methods in applied mechanics and engineering, Vol. 191, 2002, 3165-3194
9. Kim, H., “A new coupling strategy for fluid-solid interaction problems by using the interface element method”, *International journal for numerical methods in engineering*, Vol. 81, 2010, pp. 81:403-428
10. Lee, K., Son, Y., and Im, S., “Three dimensional variable-node elements based upon CS-FEM for elastic-plastic analysis”, *Computers and structures*, Vol. 158, 2015, pp. 308-332
11. Lim, J. H., and Im, S., “(4+n)-noded Moving Least Square (MLS)-based finite elements for mesh gradation”, *Structural engineering and mechanics,* Vol. 25, No. 1, 2007, pp. 91-106
12. Lim, J. H., Im, S., and Cho, Y., “MLS (moving least square)-based finite elements for three-dimensional nonmatching meshes and adaptive mesh refinement”, *Comput. methods Appl. engrg.,* Vol.196, 2007, pp. 2216-2228

13. Lim, J. H., Lim, H. H., Lee, H. J., and Im, S., “A new computational approach to contact mechanics using variable-node finite elements”, *International journal for numerical methods in engineering*, Vol.73, 2008, pp. 1966-1988
14. Lim, J. H., Sohn, D., Lee, J. H. and Im, S., “Variable-node finite elements with smoothed integration techniques and their applications for multiscale mechanics problems”, *Computers and structures*, Vol. 88, 2010, pp. 413-425
15. Lim., J. H., Im, S., and Cho, Y., “Variable-node elements for non-matching meshes by means of MLS (moving least-square) scheme”, *International journal for numerical methods in engineering*, Vol. 72, 2007, pp.835-857
16. Liu[a,b], G. R, Nguyen-Thoi[a,*],T., and Lam[c], K. Y., “An edge-based smoothed finite element method (ES-FEM) for static, free and forced vibration analyses of solid”, *Journal of sound and vibration*, Vol. 320,2009, pp. 1100-1130
17. Mosher, M. C., “ A variable node finite element method”, *Journal of computational physics*, Vol. 57, pp. 157-187, 1987
18. Pantano, A., Averill, R. C., “A penalty-based finite element interface technology”, *Computers and structures*, Vol. 80, 2002, pp. 1725-1748
19. Park, K. C., Felippa, C. A., and Rebel G., “A simple algorithm for localized construction of non-matching structural interfaces, *International journal for numerical methods in engineering*, Vol. 53, 2002, pp. 2117-2142
20. Sohn, D., Lim, J. H., and Im, S., “An efficient scheme for coupling dissimilar hexahedral meshes with the aid of variable-node transition elements”, *Advances in engineering software*, Vol. 65, 2013, pp. 200-215.
21. Sohn, D., Lim, J. H., Cho, Y. S, Kim, J. H., and Im, S., “Finite element analysis of quasistatic crack propagation in brittle media with voids or inclusions”, *Journal of computational physics*, Vol. 230, 2011, pp. 6866-6899
22. Timoshenko, S. P., and Goodier, J., N., “Theory of elasticity”, *McGraw-Hill, Inc.* 3rd. ed, 1970. Print.
23. Zhou, M., Cen, S., Bao, Y., and Li, C., “A quadsi-static crack propagation simulation based on shape-free hybrid stress-function finite elements with simple remeshing”, *Comput. methods appl. mech. engrg*, Vol. 275, 2014, pp. 159-188
24. Young, W, C and Budynas, R. G., *Roark’s Formulas for Stress and Strain,* 7th edition, McGraw-Hill, 2002

APPENDIX 1

The Elimination Matlab Code

```
function []=variable_quad()
clear all
close all
global NOC_V V_NOC
%------------------------ QUAD2  ---------------------------
disp('==========================================');
disp('         Revised form of-PROGRAM QUAD2                    ');
disp('    2-D STRESS ANALYSIS USING 4-NODE      ');
disp(' QUADRILATERAL ELEMENTS WITH TEMPERATURE  ');
disp('   T.R.Chandrupatla and A.D.Belegundu     ');
disp('==========================================');

InputData;
Variable_Node;
k=1;
disp('main')
       for gh = 1 : 4
           NC1=NOC_V(k,gh,1);
           NC2=NOC_V(k,gh,2);
           NC3=NOC_V(k,gh,3);
           NC4=NOC_V(k,gh,4);
           disp(sprintf('k, ele, NC1, NC2, NC3, NC4, %d %d %d %d %d %d', k, gh, NC1,NC2,
NC3, NC4))
       end
Bandwidth;
Stiffness;
ModifyForBC;
BandSolver_VarNode;
StressCalc;
ReactionCalc;
Output;
%------------------------  function InputData  ---------------------------
function [] = InputData();
% add a global statement for variable-node element
% global NVE NE_VarNode NE_V NE_Eliminat
%------------------------------
% NE_VarNode --- Number of Variable-Node Elements
% NVE --- Number of QUAD elements in a variable-node element
% NE_V(i,j); i = 1, NE_VarNode, j = 1, NVE ( Element ID included in
%                                      a variable-node element
% NE_Eliminate(i)=1 or -1, i = 1:NE---- = 1, regular CQUD
%                                       = -1, as a part of nodal
%                                             variable element
% NOC_V(i,j,k)---- i = 1, NE_VarNode, j = 1 : NVE (element), k = 1 : NEN (node) ( Mapping
%   between global node numbers to the local one ( 1 to 9 ) in the ith
%   variable-node element)
% V_NOC(i,j) --- i = 1, NE_VarNode, j = 1 : 9, ( Mapping the local
%   nodal number 1-9 to the global ones for the ith variable-node element
%------------------------------
global NN NE NM NDIM NEN NDN
%---------------------
% newly added for variable node elements
%---------------------
global NVE NE_VarNode NE_V NE_Eliminate
global I_Node CC_Node UU_Node
global NOC_V V_NOC
global No_I
%------------------------------
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2 FILE3
global LINP LOUT LOUT2
global NQ
global LC IPL

% disp('  1) Plane Stress Analysis');
% disp('  2) Plane Strain Analysis');
% LC = input('  Choose 1(default) or 2 :');
LC = 1;
if isempty(LC) | LC<1 | LC>2
   LC = 1;
end

disp(blanks(1));
FILE1 = '9_elem.m';
LINP  = fopen(FILE1,'r');
FILE2 = '9_elimination.doc';
LOUT  = fopen(FILE2,'w');

DUMMY = fgets(LINP);
TITLE = fgets(LINP);
% NVE: Number of QUAD elements in a single variable-node element
% NE_VarNode: Number of variable-node element
DUMMY = fgets(LINP);
TMP = str2num(fgets(LINP));
[NN, NE, NM, NDIM, NEN, NDN, NVE] = deal(TMP(1),TMP(2),TMP(3),TMP(4),TMP(5),TMP(6),
TMP(7));
NQ = NDN * NN;
% Indicate the connected CQUAD elements made of a variable-node element
DUMMY = fgets(LINP);
TMP = str2num(fgets(LINP));
[ND, NL, NMPC NE_VarNode]= deal(TMP(1),TMP(2),TMP(3),TMP(4));
%----- Connectivity for Elements of Variable Node -----
DUMMY = fgets(LINP);
NVE
for I=1:NE_VarNode
   TMP = str2num(fgets(LINP));
   [N,NE_V(N,:)] = ...
      deal(TMP(1),TMP(2:1+NVE));
end
NVE
NE_V
NE_VarNode

NPR=3; %E, NU, ALPHA

% Dimensioned for minimum 3 properties
% disp(blanks(1));
% disp('PLOT CHOICE');
% disp('  1) No Plot Data');
% disp('  2) Create Data File for in-plane Shear Stress');
% disp('  3) Create Data File for Von Mises Stress');
% IPL = input('  Choose 1(defalut), 2, or 3 :');
%      --- default is no data
IPL = 1;
if isempty(IPL) | IPL<1 | IPL>3
   IPL = 1;
end
if IPL > 1
    disp(blanks(1));
    FILE3 = input('Give Data File Name for Element Stresses ','s');
    LOUT2  = fopen(FILE3,'w');
end

%----- Coordinates -----
DUMMY = fgets(LINP);
for I=1:NN
   TMP = str2num(fgets(LINP));
   [N, X(N,:)]=deal(TMP(1),TMP(2:1+NDIM));
end
%----- Connectivity -----
DUMMY = fgets(LINP);
for I=1:NE
```

```
   TMP = str2num(fgets(LINP));
   [N,NOC(N,:), MAT(N,:), TH(N,:), DT(N,:)] = ...
      deal(TMP(1),TMP(2:1+NEN), TMP(2+NEN), TMP(3+NEN), TMP(4+NEN));
end

%----- Specified Displacements -----
DUMMY = fgets(LINP);
for I=1:ND
   TMP = str2num(fgets(LINP));
   [NU(I,:),U(I,:)] = deal(TMP(1), TMP(2));
end
%----- Component Loads -----
DUMMY = fgets(LINP);
F = zeros(NQ,1);
for I=1:NL
   TMP = str2num(fgets(LINP));
   [N,F(N)]=deal(TMP(1),TMP(2));
end

%----- Material Properties -----
DUMMY = fgets(LINP);
NPR
for I=1:NM
   TMP = str2num(fgets(LINP));
   [N, PM(N,:)] = deal(TMP(1), TMP(2:NPR+1));
end

PM
%----- Multi-point Constraints B1*Qi+B2*Qj=B0
if NMPC > 0
   DUMMY = fgets(LINP);
   for I=1:NMPC
    TMP = str2num(fgets(LINP));
      [BT(I,1), MPC(I,1), BT(I,2), MPC(I,2), BT(I,3)] = ...
                     deal(TMP(1),TMP(2),TMP(3),TMP(4),TMP(5));
   end
end
fclose(LINP);
%----------- function Re-connectivity for Variable-Node Element -----------
function []=Variable_Node();
global NN NE NM NDIM NEN NDN
global ND NL NCH NPR NMPC NBW
%---------------------
% newly added for variable node elements
%---------------------
global NVE NE_VarNode NE_V NE_Eliminate
global I_Node CC_Node UU_Node
global NOC_V V_NOC
global No_INode No_UNode No_CNode
%------------------------------
global X NOC F AREA MAT TH DT S
% Be sure NU is redefined in the subroutine - The originial NU is defined
% as NUU here
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
% Assign a negative ID to those QUAD elements made of variable-node
% elements
for i = 1 : NE
    NE_Eliminate(i)=1;
end
for i = 1 : NE_VarNode
    for j = 1 : NVE
        IJ=NE_V(i,j);
        NE_Eliminate(IJ)=-1;
    end
end
NE_Eliminate
% Set up the connectivity table for a variable-node element
```

```
if NVE == 4
    Nnode=9;
end
for i = 1 : NE_VarNode
    for j = 1 : NVE
        IE=NE_V(i,j);
        for k = 1 : NEN
            jk=(j-1)*NEN+k;
            VE(i,jk)=NOC(IE,k);
        end
    end
end
    NOC_V=zeros(NE_VarNode,NVE,NEN)
    NOC_V
%------------------------------------------------------------
% Identify the connected, the unconnected and the interior points
% Total number of nodes per variable-node element
% k index is reserved for NE_VarNode
NT=NEN*NVE;
for k = 1 : NE_VarNode
 disp(sprintf('ID of Var Node element = %d', k))
 Node_Count=zeros(NVE*2,4);
for j = 1 : NVE
    for jj = 1 : NEN
        jjj=(j-1)*NEN+jj;
        Node_Count(jjj,1)=1;
        Node_Count(jjj,2)=NE_V(k,j);
        Node_Count(jjj,3)=0;
        Node_Count(jjj,4)=VE(k,jjj);
    end
    % Node_Count(1) : # of repeatness , Node_Count(2): QUAD element ID
end
Node_Count
    for j = 1 : NT
        Nstart=VE(k,j);
        Ncount=1;
    for kkn = 1 : NT
        if kkn ~= j
        JJ=VE(k,kkn);
        if Nstart == JJ
            Node_Count(j,1)=Node_Count(j,1)+1;
            Node_Count(j,3)=Node_Count(kkn,2);
        end
        end
    end
    end
%end - for i = 1 : NE_VarNode
    Node_Count
    [B,BI]=sort(Node_Count(:,1))
    maxN=B(NT);
    I_Node(1)=Node_Count(BI(NT),4);
    icount=0;
    ucount=1;
    for i = 1 : NT
        if B(i)==1
            icount=icount+1;
            C_Node(icount,1)=Node_Count(BI(i),4);
            C_Node(icount,2)=Node_Count(BI(i),2);
            C_Node(icount,3)=0;
        end
        if B(i)== 2
            if ucount == 1
            U_Node(ucount,1)=Node_Count(BI(i),4);
            U_Node(ucount,2)=Node_Count(BI(i),2);
            U_Node(ucount,3)=Node_Count(BI(i),3);
            ucount=ucount+1;
            else
                UNode=Node_Count(BI(i),4);
                UNode
                double=0;
                for iii = 1 : ucount-1
                if UNode == U_Node(iii,1)
                    double=double+1;
                end
            end
            double
            if double == 0
                BI(i)
        U_Node(ucount,1)=Node_Count(BI(i),4)
        U_Node(ucount,2)=Node_Count(BI(i),2)
        U_Node(ucount,3)=Node_Count(BI(i),3)
        ucount=ucount+1;
        double=0;
            end
        end
    end
end
No_UNode=ucount-1;
No_INode=1;
No_CNode=icount;
No_UNode
No_INode
No_CNode
I_Node
C_Node
U_Node
UC_ID=0;
for i = 1 : No_UNode
    NUU=U_Node(i,1);
     % Check if any internally unconnected node is a connected one
     % see if the node connected to any CQUAD which is not involved in
     % variable-node elements
     if UC_ID == 0
    for ii = 1 : NE
        if NE_Eliminate(ii) ~= -1
        for jj = 1 : NEN
    NOther=NOC(ii,jj);
    if NUU == NOther
        UC_ID=i;
        break
    end
       end
        end
    end
     end
end
UC_ID
 % The above is to check if any unconnected node is a connected one.
% check if the unconnected node is connected to any boundary nodes
if UC_ID == 0
for i = 1 : No_UNode
    NUU=U_Node(i,1);
    if UC_ID == 0
    for idof = 1 : 2
        udof=(NUU-1)*NDN+idof;
    for ii = 1 : ND
        gdof=NU(ii)
        udof
        if udof == gdof
        UC_ID=i;
        break
        end
    end
    end
    end
 end
end
UC_ID
for i = 1 : UC_ID-1
    for j = 1 : 3
        UU_Node(i,j)=U_Node(i,j);
    end
    end
    if UC_ID+1 <= No_UNode
    for i = UC_ID+1 : No_UNode
        for j = 1 : 3
            UU_Node(i-1,j)=U_Node(i,j);
        end
    end
    end
    No_UNode=No_UNode-1;
    for i = 1 : No_CNode
        for j = 1 : 3
        CC_Node(i,j)=C_Node(i,j);
        end
    end
    for j = 1 : 3
    CC_Node(No_CNode+1,j)=U_Node(UC_ID,j);
    end
    No_CNode=No_CNode+1;
    I_Node
    CC_Node
    UU_Node
    No_CNode
    No_UNode
    for iii = 1 : No_CNode
    for i = 1 : 2
        NCE(i)=CC_Node(iii, i+1);
    end
    NCE_S=sort(NCE)
    if NCE_S(1) > 0
        for j = 1 : No_UNode
            for kp = 1 : 2
            NUE(kp)=UU_Node(j,kp+1);
            end
            NUE_S=sort(NUE)
            ncount=0;
            for ijk= 1 : 2
                for jki = 1 : 2
                if NCE_S(ijk)~=NUE_S(jki)
                ncount=ncount+1;
                end
                end
            end
            ncount
            if ncount == 4
                N_CDisconnect=iii
                N_UDisconnect=j
            end
        end
    end
    end
    N_CDisconnect
    N_UDisconnect
    I_Node(1)
    CC_Node(N_CDisconnect,1)
    UU_Node(N_UDisconnect,1)
    % Connectivity Table for a Variable_node element
    NVE
    % Reorder the interior node
    Ii=I_Node(1);
    Ij=CC_Node(N_CDisconnect, 1);
    Ik=UU_Node(N_UDisconnect, 1);
    V_NOC(k,5)=Ii
    V_NOC(k,4)=Ij
    V_NOC(k,6)=Ik

%      for k = 1 : NE_VarNode
    for i = 1 : NVE
        IJ=NE_V(k,i);
        for j = 1 : NEN
            IJK=NOC(IJ,j);
        if IJK == Ii
```

```
      NOC_V(k,i,j) = 5;
     end
     if IJK == Ij
         NOC_V(k,i,j)=4;
     end
     if IJK == Ik
         NOC_V(k,i,j)=6;
     end
     end
end
i5= N_UDisconnect;
ii9(1)=3;
ii9(2)=9;
ii10(1)=2;
ii10(2)=8;
GG(1)=1;
GG(2)=7;
    EEU=UU_Node(i5,1);
    EE(1)=UU_Node(i5,2);
    EE(2)=UU_Node(i5,3);
    for i6 = 1 : 2
        IEL=EE(i6);
        for i8 = 1 : NVE
            IVL=NE_V(k,i8);
            if IVL == IEL
                KKKK(i6) = i8;
            end
        end
        for i9 = 1 : No_CNode
            if i9 ~= N_CDisconnect
                FFC=CC_Node(i9,1);
                FF(1)=CC_Node(i9, 2);
                FF(2)=CC_Node(i9, 3);
                count=0;
                for i10 = 1 : 2
                    if IEL == FF(i10)
                        count=count+1;
                        EEFF=EE(i6);
                    end
                end
                if count == 1
                for i11 = 1 : NEN
                    NEE=NOC(EEFF,i11);
                    if FFC == NEE
                        V_NOC(k,ii9(i6))=NEE;
                        NOC_V(k,KKKK(i6), i11)=ii9(i6);
                    end
                end
                end
            end
        end
    end
    UU_Node
    for ix = 1 : No_UNode
        if ix ~= N_UDisconnect
            IXN = UU_Node(ix,1);
            IXE(1)=UU_Node(ix,2);
            IXE(2)=UU_Node(ix,3);
            for iy = 1 : 2
                   IXX = IXE(iy);
                for iz = 1 : 2
                   IEN=EE(iz);
                   if IXX == IEN
                       for ixy = 1 : NEN
                           NC4=NOC(IEN,ixy);
                           if NC4 == IXN
                               V_NOC(k,ii10(iz))=NC4;
                               NOC_V(k,KKKK(iz),ixy)=ii10(iz);
                           end
                       end
                   end
                end
            end
        end
    end
    EE
    NOC(NE_V(1),:)
    NOC_V(1,1,:)
    NOC(NE_V(2),:)
    NOC_V(1,2,:)
    NOC(NE_V(3),:)
    NOC_V(1,3,:)
    NOC(NE_V(4),:)
    NOC_V(1,4,:)
    UU_Node
    count=0;
    for ia = 1 : No_UNode
        if count == 0
        if ia ~= N_UDisconnect
        ix = UU_Node(ia,1)
        UA(1)=UU_Node(ia,2)
        UA(2)=UU_Node(ia,3)
        for ib = 1 : 2
             UAA=UA(ib)
                 if UAA == EE(1)
                     IBB=ib
                     count=count+1;
                     IAA=ia
                 end
        end
        for ib = 1 : 2
            if ib ~=IBB
                UAEL=UA(ib)
                for id = 1 : NVE
                    if UAEL == NE_V(k,id)
                        UAID=id
                    end
                end
                for ic = 1 : NEN
                    NOCU=NOC(UAEL,ic)
                    if NOCU == ix
                        V_NOC(k,ii10(1))=NOCU;
                        NOC_V(k,UAID,ic)=ii10(1);
                    end
                end
            end
        end
        end
        end
    end
    count=0;
    for ia = 1 : No_UNode
        if ia ~= IAA
        if ia ~= N_UDisconnect
        ix = UU_Node(ia,1);
        UA(1)=UU_Node(ia,2);
        UA(2)=UU_Node(ia,3);
        for ib = 1 : 2
             UAA=UA(ib)
                 if UAA == EE(2)
                     IBB=ib
                     count=count+1;
                 end
        end
        for ib = 1 : 2
            if ib ~=IBB
                UAEL=UA(ib)
                for id = 1 : NVE
                    if UAEL == NE_V(k,id);
                        UAID=id;
                    end
                end
```

```
                    for ic = 1 : NEN
                        NOCU=NOC(UAEL,ic);
                        if NOCU == ix
                            ix;
                            ic;
                            ii10(2);
                            V_NOC(k,ii10(2))=NOCU;
                            NOC_V(k,UAID,ic)=ii10(2);
                        end
                    end
                end
            end
            end
            end

        end
        V_NOC
        count = 0;
        for if3 = 1 : NVE
            for if2 = 1 : NEN
            IIN = NOC_V(k,if3,if2);
            if IIN ==0
                count = count + 1
                IINE(count) = if3;
            end
            end
        end
        IINE
       for if4 = 1 : count
           IIN = IINE(if4);
           for if5 = 1 : NEN
               if NOC_V(k,IIN,if5) == ii10(1)
                   INN = IIN;
               end
           end
       end
       for if6 = 1 : NEN
           if NOC_V(k,INN,if6) == 0;
               NELE=NE_V(k,INN);
               V_NOC(k,1)=NOC(NELE,if6);
               NOC_V(k,INN,if6)=1;
           end
       end
       IINE
       INN
       for if4 = 1 : count
           IIN = IINE(if4);
           if IIN ~= INN
               ANN = IIN
           end
       end
       ANN
       for if6 = 1 : NEN
           if NOC_V(k,ANN,if6) == 0;
               NELE=NE_V(k,ANN);
               V_NOC(k,7)=NOC(NELE,if6);
               NOC_V(k,ANN,if6)=7;
           end
       end
       for gh = 1 : 4
           NC1=NOC_V(k,gh,1);
           NC2=NOC_V(k,gh,2);
           NC3=NOC_V(k,gh,3);
           NC4=NOC_V(k,gh,4);
           disp('Variable_Node')
           disp(sprintf('k, ele, NC1, NC2, NC3, NC4, %d %d %d %d %d %d', k, gh, NC1,NC2,
NC3, NC4))
       end
       V_NOC(k,:)
    end
%------------------------  function Bandwidth  ---------------------------
```

```
function []=Bandwidth();
global NN NE NM NDIM NEN NDN
%----------------------
% newly added for variable node elements
%----------------------
global NVE NE_VarNode NE_V NE_Eliminate
global I_Node CC_Node UU_Node
global NOC_V V_NOC
global  No_UNode No_CNode
%------------------------------
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
%----- Bandwidth NBW from Connectivity NOC() and MPC
NBW = 0;
% Bandwidth for the regular CQUAD
for I = 1:NE
    NEI = NE_Eliminate(I);
    if NEI ~= -1
   NMIN = NOC(I, 1);
   NMAX = NOC(I, 1);
   for J = 2:NEN
      if NMIN > NOC(I, J); NMIN = NOC(I, J); end
      if NMAX < NOC(I, J); NMAX = NOC(I, J); end
   end
   NTMP = NDN * (NMAX - NMIN + 1);
   if NBW < NTMP; NBW = NTMP; end
    end
end
% Bandwidth for the variable-node elements
for I = 1:NE_VarNode
   NMIN = V_NOC(I, 1);
   NMAX = V_NOC(I, 1);
   for J = 2:9
      if NMIN > V_NOC(I, J); NMIN = V_NOC(I, J); end
      if NMAX < V_NOC(I, J); NMAX = V_NOC(I, J); end
   end
   NTMP = NDN * (NMAX - NMIN + 1);
   if NBW < NTMP; NBW = NTMP; end
end
% Bandwidth for MPC
for I = 1:NMPC
   NABS = abs(MPC(I, 1) - MPC(I, 2)) + 1;
   if (NBW < NABS); NBW = NABS; end
end
disp(blanks(1));
disp(sprintf('Bandwidth = %d', NBW));
kN=1;
       for gh = 1 : 4
           NC1=NOC_V(kN,gh,1);
           NC2=NOC_V(kN,gh,2);
           NC3=NOC_V(kN,gh,3);
           NC4=NOC_V(kN,gh,4);
           disp(sprintf('kB, ele, NC1, NC2, NC3, NC4, %d %d %d %d %d %d', kN, gh,
NC1,NC2, NC3, NC4))
       end

%------------------------  function Stiffness  ---------------------------
function []=Stiffness();
global NN NE NM NDIM NEN NDN
%----------------------
% newly added for variable node elements
%----------------------

global NVE NE_VarNode NE_V NE_Eliminate
global I_Node CC_Node UU_Node
global NOC_V V_NOC
global No_INode No_UNode No_CNode
```

```
%------------------------------
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
global NQ
global LC IPL
global XNI
global G_KIC
disp('stiffness')
NOC_V(1,:,:)
NOC_V
size(NOC_V)
NOC_V(1,1,1)
NOC_V(1,1,2)
kN=1
       for gh = 1 : 4
           NC1=NOC_V(kN,gh,1);
           NC2=NOC_V(kN,gh,2);
           NC3=NOC_V(kN,gh,3);
           NC4=NOC_V(kN,gh,4);
           disp(sprintf('kN, ele, NC1, NC2, NC3, NC4, %d %d %d %d %d %d', kN, gh,
NC1,NC2, NC3, NC4))
       end
       V_NOC

%----- Global Stiffness Matrix
S = zeros(NQ,NBW);

%----- Corner Nodes and Integration Points
C = .57735026919;
XNI(1, 1) = -C;
XNI(1, 2) = -C;
XNI(2, 1) = C;
XNI(2, 2) = -C;
XNI(3, 1) = C;
XNI(3, 2) = C;
XNI(4, 1) = -C;
XNI(4, 2) = C;
% Formation of Stiffness Matrices for Regular CQUAD Elements
for N = 1:NE
        NEI = NE_Eliminate(N);
    if NEI ~= -1
   disp(sprintf('Forming Stiffness Matrix of Regular Element %d', N));

%--------  Element Stiffness and Temperature Load  -----
   TL = zeros(8,1);
   SE = zeros(8);
   DTE = DT(N);
%  --- Weight Factor is ONE
%  --- Loop on Integration Points
   for IP = 1:4
%  ---  Get DB Matrix at Integration Point IP
        XI = XNI(IP, 1);
        ETA = XNI(IP, 2);
        [DJ, D, B, DB] = dbmat(N, LC, MAT, PM, NOC, X ,XI,ETA);
        THICK = TH(N);

%  --- Element Stiffness Matrix  SE
       for I = 1:8
          for J = 1:8
              C = 0;
              for K = 1:3
                 C = C + B(K, I) * DB(K, J) * DJ * THICK;
              end
                SE(I, J) = SE(I, J) + C;
           end
       end
```

```
%  --- Determine Temperature Load TL
       AL = PM(MAT(N), 3);
     PNU = PM(MAT(N), 2);
       C = AL * DTE;
       if (LC == 2); C = (1 + PNU) * C; end
       for I = 1:8
          TL(I) = TL(I) + THICK * DJ * C * (DB(1, I) + DB(2, I));
       end
   end
   disp('.... Placing in Global Locations');
   for II = 1:NEN
      NRT = NDN * (NOC(N, II) - 1);
      for IT = 1:NDN
         NR = NRT + IT;
         I = NDN * (II - 1) + IT;
         for JJ = 1:NEN
            NCT = NDN * (NOC(N, JJ) - 1);
            for JT = 1:NDN
               J = NDN * (JJ - 1) + JT;
               NC = NCT + JT - NR + 1;
               if (NC > 0)
                  S(NR, NC) = S(NR, NC) + SE(I, J);
               end
            end
         end
         F(NR) = F(NR) + TL(I);
      end
   end
    end
end
for N = 1:NE_VarNode
   disp(sprintf('Forming Stiffness Matrix of Variable Node Element %d', N));
   NOC_V(1,:,:)
   [KCC, KIC]=Stiffness_Variable(N);
   % G_KIC * Uc will gives the displacement at the interior node for each
   % element G_KIC(N,:,:)=KIC(:,:)
   for nrow = 1 : NDN
   for nki = 1 : No_CNode
       for nkj = 1 : NDN
           I_nk=(nki-1)*NDN+nkj;
           G_KIC(N,nrow,I_nk)=KIC(nrow,I_nk);
%            disp(sprintf('N, nkj,I_nk, G_%5d %5d %5d %10.4E', N, nrow, I_nk,
G_KIC(N,nrow,I_nk)))
       end
   end
   end
% Place KCC (a 10 x10 matrix for a variable-node element )
% in the global matrix
   disp('.... Placing in Global Locations');
   No_INode;
   No_UNode;
   No_CNode;
   No_CBoundary = No_CNode;
   No_UBoundary = No_UNode;
   No_IBoundary = No_INode;
  C_Node(1)=1;
  C_Node(2)=3;
  C_Node(3)=7;
  C_Node(4)=9;
  C_Node(5)=4;
  U_Node(1)=2;
  U_Node(2)=6;
  U_Node(3)=8;
  I_Node(1)=5;
   for II = 1:No_CBoundary
       Node_I=C_Node(II);
       NRT = NDN * (V_NOC(N, Node_I) - 1);
      for IT = 1:NDN
         NR = NRT + IT;
         I = NDN * (II - 1) + IT;
         for JJ = 1:No_CBoundary
```

```
             Node_J=C_Node(JJ);
             NCT = NDN * (V_NOC(N, Node_J) - 1);
            for JT = 1:NDN
               J = NDN * (JJ - 1) + JT;
               NC = NCT + JT - NR + 1;
               if (NC > 0)
%                   disp(sprintf('NVI V_NOC(N,NVI) NVJ V_NOC(N,NVJ) %d %d %d %d', Node_I,
V_NOC(N,Node_I), Node_J, V_NOC(N,Node_J)));
%                   disp(sprintf('I J NR NC %d %d %d %d', I, J, NR, NC));
%                    NR
%                    NC
                  S(NR, NC) = S(NR, NC) + KCC(I, J);
               end
            end
         end
         % No thermal load accepted
         % F(NR) = F(NR) + TL(I);
      end
   end
%    S
%    V_NOC
 % Constrain the displacements at the interior
 % and the unconnected nodes to be zero
%  The process is done in the global matrix
    for II = 1:No_UBoundary
       Node_I=U_Node(II);
       NRT = NDN * (V_NOC(N, Node_I) - 1);
      for IT = 1:NDN
         NR = NRT + IT;
         I = NDN * (NR - 1) + IT;
         S(NR,1)= 100000*100000;
         % No thermal load accepted
         % F(NR) = F(NR) + TL(I);
      end
    end
      for II = 1:No_IBoundary
       Node_I=I_Node(II);
       NRT = NDN * (V_NOC(N, Node_I) - 1);
      for IT = 1:NDN
         NR = NRT + IT;
         S(NR,1)= 100000*100000;
         % No thermal load accepted
         % F(NR) = F(NR) + TL(I);
      end
      end
end
%------------------------  function ModifyForBC  ---------------------------
function []=ModifyForBC();
global NN NE NM NDIM NEN
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
global NQ
%----- Decide Penalty Parameter CNST -----
CNST = 0;
for I = 1:NQ
   if CNST < S(I, 1); CNST = S(I, 1); end
end
CNST = CNST * 1000000;

%----- Modify for Boundary Conditions -----
%    --- Displacement BC ---
for I = 1:ND
   N = NU(I);
   S(N, 1) = S(N, 1) + CNST;
   F(N) = F(N) + CNST * U(I);
end
%--- Multi-point Constraints ---
```

```
for I = 1:NMPC
   I1 = MPC(I, 1);
   I2 = MPC(I, 2);
   S(I1, 1) = S(I1, 1) + CNST * BT(I, 1) * BT(I, 1);
   S(I2, 1) = S(I2, 1) + CNST * BT(I, 2) * BT(I, 2);
   IR = I1;
   if IR > I2; IR = I2; end
   IC = abs(I2 - I1) + 1;
   S(IR, IC) = S(IR, IC) + CNST * BT(I, 1) * BT(I, 2);
   F(I1) = F(I1) + CNST * BT(I, 1) * BT(I, 3);
   F(I2) = F(I2) + CNST * BT(I, 2) * BT(I, 3);
end

%------------------------  function BandSolver  ---------------------------
function []=BandSolver_VarNode();
global NN NE NM NDIM NEN NDN
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
%---------------------
% newly added for variable node elements
%---------------------
global NVE NE_VarNode NE_V NE_Eliminate
global I_Node CC_Node UU_Node
global NOC_V V_NOC
global No_INode No_UNode No_CNode
%------------------------------
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
global NQ
global G_KIC
%----- Equation Solving using Band Solver -----
disp('Solving using Band Solver(bansol.m)');
[F] = bansol(NQ,NBW,S,F);
F
  C_Node(1)=1;
  C_Node(2)=3;
  C_Node(3)=7;
  C_Node(4)=9;
  C_Node(5)=4;
  U_Node(1)=2;
  U_Node(2)=6;
  U_Node(3)=8;
C_UC(1,1)=0.5;
C_UC(1,3)=0.5;
C_UC(2,2)=0.5;
C_UC(2,4)=0.5;
C_UC(3,3)=0.5;
C_UC(3,7)=0.5;
C_UC(4,4)=0.5;
C_UC(4,8)=0.5;
C_UC(5,5)=0.5;
C_UC(5,7)=0.5;
C_UC(6,6)=0.5;
C_UC(6,8)=0.5;
NUN=NDN*No_INode;
NCN=NDN*No_CNode;
%***************************
  for i = 1 : NN
      Node_ID(i)=i;
  end
%***************************
for N = 1 : NE_VarNode
    Int_I=V_NOC(N,5);
   % Recover the displacements at the interior nodes
  for jt = 1 : NDN
    Int_II=(Int_I-1)*NDN+jt;
    sum=0;
    for i = 1 : No_CNode
        CI=C_Node(i);
        GI=V_NOC(N,CI);
        for j = 1 : NDN
            GII=(GI-1)*NDN+j;
            ij = (i-1)*NDN+j;
            disp_G(ij)=F(GII);
            sum=sum+G_KIC(N,jt,ij)*disp_G(ij);
        end
    end
    disp_G;
    sum;
    F(Int_II)=F(Int_II)+sum;
  end
 % Recover the displacement at the unconnected nodes
  for i = 1 : No_UNode
      CoU=U_Node(i); % local nodal number
      GU=V_NOC(N,CoU); % global nodal number
      %******************************
      NGU=Node_ID(GU)
      if NGU >= 0
      %*************************
      for j = 1 : NDN
          ij = (i-1)*NDN+j;
          Gj = (GU-1)*NDN+j;
          sum = 0;
           for L = 1 : (No_CNode-1)
               for M = 1 : NDN
                   LM = (L-1)*NDN+M;
                   sum=sum+C_UC(ij,LM)*disp_G(LM);
               end
           end
           sum
         F(Gj)=F(Gj)+sum;
      end
      %*******************************
      Node_ID(GU)= -1 * GU
      %*******************************
      end
  end
end



%------------------------  function StressCalc  ---------------------------
function []=StressCalc();
global NN NE NM NDIM NEN NDN
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS VSTRESS MSTRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
global LC IPL
global XNI

%-----  Stress Calculations -----
%--- Stresses at Integration Points
fprintf(LOUT,'ELEM#  von Mises Stresses at 4 Integ_points\n');

for N = 1:NE
    fprintf(LOUT,'  %d',N);
    for IP = 1:4
       XI = XNI(IP,1); ETA = XNI(IP,2);
      [DJ, D, B, DB] = dbmat(N, LC, MAT, PM, NOC, X ,XI,ETA);
%     --- Stress Evaluation
      for I = 1:NEN
         IN = NDN * (NOC(N, I) - 1);
         II = NDN * (I - 1);
         for J = 1:NDN
            Q(II + J) = F(IN + J);
         end
      end
      AL = PM(MAT(N), 3);
      PNU = PM(MAT(N), 2);
      C1 = AL * DT(N);
      if LC == 2; C1 = C1 * (1 + PNU); end
       for I = 1:3
         C = 0;
         for K = 1:8
            C = C + DB(I, K) * Q(K);
         end
         STR(I) = C - C1 * (D(I, 1) + D(I, 2));
      end
%     --- Von Mises Stress at Centroid
      C = 0;
      if LC == 2; C = PNU * (STR(1) + STR(2)); end
      C1 = (STR(1)-STR(2))^2 + (STR(2)-C)^2 + (C-STR(1))^2;
      SV = sqrt(.5 * C1 + 3 * STR(3)^2);
      VSTRESS(N,IP) = SV;
%      --- Maximum Shear Stress R
      R = sqrt(.25 * (STR(1) - STR(2))^2 + (STR(3))^2);
      MSTRESS(N,IP) = R;
   end
end

%------------------------  function ReactionCalc  ---------------------------
function []=ReactionCalc();
global NN NE NM NDIM NEN NDN
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
%----- Reaction Calculation -----
disp(blanks(1));

for I = 1:ND
   N = NU(I);
   REACT(I) = CNST * (U(I) - F(N));
end

%------------------------  function Output  ---------------------------
function []=Output();
global NN NE NM NDIM NEN NDN
global ND NL NCH NPR  NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS VSTRESS MSTRESS REACT
global CNST
global TITLE FILE1 FILE2 FILE3
global LINP LOUT LOUT2
global LC IPL

disp(sprintf('Output for Input Data from file %s\n',FILE1));
fprintf(LOUT,'Output for Input Data from file %s\n',FILE1);

disp(TITLE);
fprintf(LOUT,'%s\n',TITLE);
if LC == 1; fprintf(LOUT,'Plane Stress Analysis\n'); end
if LC == 2; fprintf(LOUT,'Plane Strain Analysis\n'); end

disp(' Node#    X-Displ          Y-Displ');
fprintf(LOUT,' Node#    X-Displ          Y-Displ\n');
I=[1:NN]';
% print a matrix
disp(sprintf(' %4d %15.4E %15.4E\n',[I,F(2*I-1),F(2*I)]'));
fprintf(LOUT,' %4d %15.4E %15.4E\n',[I,F(2*I-1),F(2*I)]');

%----- Reaction Calculation -----
disp(sprintf('  DOF#     Reaction'));
```

```
fprintf(LOUT,'  DOF#     Reaction\n');
for I = 1:ND
   N = NU(I);
   disp(sprintf(' %4d %15.4E',N,REACT(I)));
   fprintf(LOUT,' %4d %15.4E\n',N,REACT(I));
end

if IPL ==2
   fprintf(LOUT2,'Max. in-plane Shear Stress\n');
elseif IPL ==3
    fprintf(LOUT2,'Von Mises Stress\n');
end

%-----  Stress Calculations -----
%--- Stresses at Integration Points
disp(sprintf('ELEM#  von Mises Stresses at 4 Integ_points'));
fprintf(LOUT,'ELEM#  von Mises Stresses at 4 Integ_points\n');

for N = 1:NE
    disp(sprintf('%5d  %14.4E %14.4E %14.4E %14.4E',N,VSTRESS(N,1:4)));
    fprintf(LOUT,'%5d  %14.4E %14.4E %14.4E %14.4E\n',N,VSTRESS(N,1:4));
    if IPL == 2
%--- Maximum Shear Stress R
    fprintf(LOUT2,'%14.4E %14.4E %14.4E %14.4E\n',MSTRESS(N,1:4));
   elseif IPL == 3
%--- Von Mises Stress at Integration Point
    fprintf(LOUT2,'%14.4E %14.4E %14.4E %14.4E\n',VSTRESS(N,1:4));
   end
end



disp(blanks(1));
disp('-----    All Calculations are done    -----');
disp(sprintf('The Results are available in the text file %s', FILE2));
disp('View using a text processor');
if (IPL > 1)
   fclose(LOUT2);
   disp(sprintf('Element Stress Data in file %s', FILE3));
   disp('Run BESTFITQ and then CONTOUR1 or CONTOUR2 to plot stresses');
end
ALPHA=0.5;
%disp('coefficient of displacement','%d',ALPHA);
fprintf(LOUT,'DEFORMATION COORDINATE \n')
for I=1:NN;
    X_DEFORMED(I,1)=X(I,1)+ALPHA*F(2*I-1);
    X_DEFORMED(I,2)=X(I,2)+ALPHA*F(2*I);
    I_DEFORMED=I;

%     disp(sprintf(' %d  %15.4E
%15.4E\n',[I_DEFORMED,X_DEFORMED(I,1),X_DEFORMED(I,2)]'));
     fprintf(LOUT,' %d  %15.4E %15.4E\n',[I_DEFORMED,X_DEFORMED(I,1),X_DEFORMED(I,2)]');

end
%----------------------------(ONLY FOR PLOT_2D)-----------------
%axis off;
hold on;
%figure;
title('Actual-Mesh');
for N=1 : NE
  for j=1 : NEN
    xe(j)=X(NOC(N,j),1);
    ye(j)=X(NOC(N,j),2);
  end
  xe(j+1)=xe(1);ye(j+1)=ye(1);

  line(xe,ye)
end
%%--------------------- Calculating the MAX range of X and Y coordinates
LX = abs(max(X(:,1))-min(X(:,1)));
```

```
LY = abs(max(X(:,2))-min(X(:,2)));
LMAX = max (LX,LY);
epsilon =LMAX*0.001;
for i=1:NN
    xc = X(i,1)-epsilon;yc=X(i,2)-epsilon;
    text(xc,yc,num2str(i),'FontSize',7);
end
hold off;

figure;
title('Deformed-Mesh');
for N=1 : NE
  for j=1 : NEN
    XE_DEFORMED(j)=X_DEFORMED(NOC(N,j),1);
    YE_DEFORMED(j)=X_DEFORMED(NOC(N,j),2);
  end
  XE_DEFORMED(j+1)=XE_DEFORMED(1);YE_DEFORMED(j+1)=YE_DEFORMED(1);
  line(XE_DEFORMED,YE_DEFORMED,'linestyle','-','color','m');
end
fclose(LOUT);




%------------------------  dbmat  ---------------------------
function [DJ, D, B, DB] = dbmat(N, LC, MAT, PM, NOC, X,XI,ETA);

%  --- Material Properties
   M = MAT(N);
   E = PM(M, 1);
   PNU = PM(M, 2);
   AL = PM(M, 3);
%  --- D() Matrix
   if LC == 1
%  --- Plane Stress
      C1 = E / (1 - PNU^2);
      C2 = C1 * PNU;
   else
%  --- Plane Strain
      C = E / ((1 + PNU) * (1 - 2 * PNU));
      C1 = C * (1 - PNU);
      C2 = C * PNU;
   end
   C3 = .5 * E / (1 + PNU);

   D(1, 1) = C1;
   D(1, 2) = C2;
   D(1, 3) = 0;
   D(2, 1) = C2;
   D(2, 2) = C1;
   D(2, 3) = 0;
   D(3, 1) = 0;
   D(3, 2) = 0;
   D(3, 3) = C3;

%  -------  DB()  MATRIX  ------
%  --- Nodal Coordinates
      N1 = NOC(N, 1);
      N2 = NOC(N, 2);
      N3 = NOC(N, 3);
      N4 = NOC(N, 4);
      X1 = X(N1, 1);
      Y1 = X(N1, 2);
      X2 = X(N2, 1);
      Y2 = X(N2, 2);
      X3 = X(N3, 1);
      Y3 = X(N3, 2);
      X4 = X(N4, 1);
      Y4 = X(N4, 2);
```

```
%  --- Formation of Jacobian  TJ
      TJ11 = ((1 - ETA) * (X2 - X1) + (1 + ETA) * (X3 - X4)) / 4;
      TJ12 = ((1 - ETA) * (Y2 - Y1) + (1 + ETA) * (Y3 - Y4)) / 4;
      TJ21 = ((1 - XI) * (X4 - X1) + (1 + XI) * (X3 - X2)) / 4;
      TJ22 = ((1 - XI) * (Y4 - Y1) + (1 + XI) * (Y3 - Y2)) / 4;
%  --- Determinant of the JACOBIAN
      DJ = abs(TJ11 * TJ22 - TJ12 * TJ21);
%  --- A(3,4) Matrix relates Strains to
%  --- Local Derivatives of u
      A(1, 1) = TJ22 / DJ;
      A(2, 1) = 0;
      A(3, 1) = -TJ21 / DJ;
      A(1, 2) = -TJ12 / DJ;
      A(2, 2) = 0;
      A(3, 2) = TJ11 / DJ;
      A(1, 3) = 0;
      A(2, 3) = -TJ21 / DJ;
      A(3, 3) = TJ22 / DJ;
      A(1, 4) = 0;
      A(2, 4) = TJ11 / DJ;
      A(3, 4) = -TJ12 / DJ;
%  --- G(4,8) Matrix relates Local Derivatives of u
%  --- to Local Nodal Displacements q(8)
      G = zeros(4, 8);

      G(1, 1) = -(1 - ETA) / 4;
      G(2, 1) = -(1 - XI) / 4;
      G(3, 2) = -(1 - ETA) / 4;
      G(4, 2) = -(1 - XI) / 4;
      G(1, 3) = (1 - ETA) / 4;
      G(2, 3) = -(1 + XI) / 4;
      G(3, 4) = (1 - ETA) / 4;
      G(4, 4) = -(1 + XI) / 4;
      G(1, 5) = (1 + ETA) / 4;
      G(2, 5) = (1 + XI) / 4;
      G(3, 6) = (1 + ETA) / 4;
      G(4, 6) = (1 + XI) / 4;
      G(1, 7) = -(1 + ETA) / 4;
      G(2, 7) = (1 - XI) / 4;
      G(3, 8) = -(1 + ETA) / 4;
      G(4, 8) = (1 - XI) / 4;
%  --- B(3,8) Matrix Relates Strains to q
      for I = 1:3
         for J = 1:8
            C = 0;
            for K = 1:4
               C = C + A(I, K) * G(K, J);
            end
             B(I, J) = C;
         end
      end
%  --- DB(3,8) Matrix relates Stresses to q(8)
      for I = 1:3
         for J = 1:8
           C = 0;
            for K = 1:3
              C = C + D(I, K) * B(K, J);
           end
              DB(I, J) = C;
         end
      end
function [F] = bansol(NN,NBW,S,F)
% Band Solver

N = NN;
%----- Forward Elimination -----
for K=1:N-1
   NBK = N - K + 1;
   if (N - K + 1) > NBW
      NBK = NBW;
   end
   for I=K+1:NBK+K-1
      I1 = I - K + 1;
      C = S(K, I1) / S(K, 1);
      for J=I: NBK+K-1
         J1 = J - I + 1;
         J2 = J - K + 1;
         S(I, J1) = S(I, J1) - C * S(K, J2);
      end
      F(I) = F(I) - C * F(K);
   end
end
%----- Back Substitution -----
F(N) = F(N) / S(N, 1);
for II=1:N-1
   I = N - II;
   NBI = N - I + 1;
   if (N - I + 1) > NBW
      NBI = NBW;
   end
   SUM = 0.;
   for J=2:NBI
      SUM = SUM + S(I, J) * F(I + J - 1);
   end
   F(I) = (F(I) - SUM) / S(I, 1);
end
%------------------------  function Stiffness  ---------------------------
function [KCC, KIC]=Stiffness_Variable(NN);
% Construct the 10x10 stiffness matrix for a variable-node element
global NNODE NE NM NDIM NEN NDN
%------------------------------
% NE_VarNode --- Number of Variable-Node Elements
% NVE --- Number of QUAD elements in a variable-node element
% NE_V(i,j); i = 1, NE_VarNode, j = 1, NVE ( Element ID included in
%                                      a variable-node element
% NE_Eliminate(i)=1 or -1, i = 1:NE---- = 1, regular CQUD
%                                       = -1, as a part of nodal
%                                             variable element
% NOC_V(i,j,k)---- i = 1, NE_VarNode, j = 1 : NVE, k = 1 : NEN ( Mapping
%   between global node numbers to the local one ( 1 to 9 ) in the ith
%   variable-node element)
% V_NOC(i,j) --- i = 1, NE_VarNode, j = 1 : 9, ( Mapping the local
%   nodal number 1-9 to the global ones for the ith variable-node element
%------------------------------
% I_Node
% CC_Node
% UU_Node
% No_I
%---------------------
global NVE NE_VarNode NE_V NE_Eliminate
global NOC_V V_NOC
%------------------------------
global ND NL NCH NPR NMPC NBW
global X NOC F AREA MAT TH DT S
global PM NU U MPC BT STRESS REACT
global CNST
global TITLE FILE1 FILE2
global LINP LOUT
global NQ
global LC IPL
global XNI
% newly added
% Define Interior, Unconnected and Connected boundary nodes for
% a standard 9-node variable-node element
       for gh = 1 : 4
           NC1=NOC_V(NN,gh,1);
           NC2=NOC_V(NN,gh,2);
           NC3=NOC_V(NN,gh,3);
           NC4=NOC_V(NN,gh,4);
           disp(sprintf('STIFFNESS-VAR k, ele, NC1, NC2, NC3, NC4, %d %d %d %d %d %d',
NN, gh, NC1,NC2, NC3, NC4))
       end
```

```
    KCC=zeros(10,10);
    NOC_V(1,:,:)
  No_INode=1;
  No_CBoundary=5;
  No_UBoundary=3;
  C_Node(1)=1;
  C_Node(2)=3;
  C_Node(3)=7;
  C_Node(4)=9;
  C_Node(5)=4;
  U_Node(1)=2;
  U_Node(2)=6;
  U_Node(3)=8;
  I_Node(1)=5;
  C_UC=zeros(6,10);
C_UC(1,1)=0.5;
C_UC(1,3)=0.5;
C_UC(2,2)=0.5;
C_UC(2,4)=0.5;
C_UC(3,3)=0.5;
C_UC(3,7)=0.5;
C_UC(4,4)=0.5;
C_UC(4,8)=0.5;
C_UC(5,5)=0.5;
C_UC(5,7)=0.5;
C_UC(6,6)=0.5;
C_UC(6,8)=0.5;
%----- Global Stiffness Matrix in an Node_Varaible Element
SN=zeros(18,18);
%----- Corner Nodes and Integration Points
C = .57735026919;
XNI(1, 1) = -C;
XNI(1, 2) = -C;
XNI(2, 1) = C;
XNI(2, 2) = -C;
XNI(3, 1) = C;
XNI(3, 2) = C;
XNI(4, 1) = -C;
XNI(4, 2) = C;

%for NN = 1:NE_VarNode
       disp(sprintf('Forming Stiffness Matrix of Var-Node Element %d', NN));
    for INN = 1 : NVE
        N = NE_V(NN,INN);
    disp(sprintf('Forming Stiffness Matrix of Element %d', N));
%--------  Element Stiffness and Temperature Load  -----
   TL = zeros(8,1);
   SE = zeros(8);
   DTE = DT(N);
%  --- Weight Factor is ONE
%  --- Loop on Integration Points
   for IP = 1:4
%  ---  Get DB Matrix at Integration Point IP
        XI = XNI(IP, 1);
        ETA = XNI(IP, 2);
        [DJ, D, B, DB] = dbmat(N, LC, MAT, PM, NOC, X ,XI,ETA);
        THICK = TH(N);

%  --- Element Stiffness Matrix  SE
       for I = 1:8
          for J = 1:8
              C = 0;
              for K = 1:3
                 C = C + B(K, I) * DB(K, J) * DJ * THICK;
              end
                SE(I, J) = SE(I, J) + C;
           end
       end
%  --- Determine Temperature Load TL
       AL = PM(MAT(N), 3);
     PNU = PM(MAT(N), 2);
```

```
        C = AL * DTE;
        if (LC == 2); C = (1 + PNU) * C; end
        for I = 1:8
           TL(I) = TL(I) + THICK * DJ * C * (DB(1, I) + DB(2, I));
        end
    end
    SE
    disp('.... Placing in 1-9 Locations in a Variable-Node Element');
    NOC_V(1,:,:)

    for II = 1:NEN

        NOCVI=NOC_V(NN,INN,II);
       NRT = NDN * (NOCVI - 1);
       for IT = 1:NDN
          NR = NRT + IT;
          I = NDN * (II - 1) + IT;
          for JJ = 1:NEN
             NOCVJ=NOC_V(NN,INN,JJ);
             NCT = NDN * (NOCVJ - 1);
             for JT = 1:NDN
                J = NDN * (JJ - 1) + JT;
                NC = NCT + JT ;
                disp(sprintf('Forming NN, INN, NR NC NOCVI NOCVJ I J %d %d %d %d %d %d %d
%d %d', NN, INN, NR, NC, NOCVI, NOCVJ, I, J));
                SN(NR, NC) = SN(NR, NC) + SE(I, J);
             end
          end
          F(NR) = F(NR) + TL(I);
       end
    end
end
S
SN
%local  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%local  N_edge C_Node U_Node I_Node
for i = 1 : No_INode
        NIi=I_Node(i);
    for j = 1 : No_INode
        NIj=I_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Irow=(NIi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Icol=(NIj-1)*NDN+m;
    S_II(irow,jcol)= SN(Irow,Icol);
            end
        end
    end
end
S_II_inv=inv(S_II);
%local  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%local  N_edge C_Node U_Node I_Node
for i = 1 : No_CBoundary
        UCi=C_Node(i);
    for j = 1 : No_CBoundary
        UCj=C_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Crow=(UCi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Ccol=(UCj-1)*NDN+m;
    S_CC(irow,jcol)= SN(Crow,Ccol);
            end
        end
    end
end
%global  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%global  N_edge C_Node U_Node I_Node
```

```
for i = 1 : No_UBoundary
        NUi=U_Node(i);
    for j = 1 : No_CBoundary
        NCj=C_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Urow=(NUi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Ccol=(NCj-1)*NDN+m;
    S_UC(irow,jcol)= SN(Urow,Ccol);
            end
        end
    end
end
S_CU=S_UC';
%global  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%global  N_edge C_Node U_Node I_Node
for i = 1 : No_UBoundary
        NUi=U_Node(i);
    for j = 1 : No_INode
        NIj=I_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Urow=(NUi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Icol=(NIj-1)*NDN+m;
    S_UI(irow,jcol)= SN(Urow,Icol);
            end
        end
    end
end
S_IU=S_UI';
%global  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%global  N_edge C_Node U_Node I_Node
for i = 1 : No_CBoundary
        NCi=C_Node(i);
    for j = 1 : No_INode
        NIj=I_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Crow=(NCi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Icol=(NIj-1)*NDN+m;
    S_CI(irow,jcol)= SN(Crow,Icol);
            end
        end
    end
end
S_IC=S_CI';
%global  No_INode No_CBoundary No_UBoundary No_Boundary_Segment
%global  N_edge C_Node U_Node I_Node
for i = 1 : No_UBoundary
        UNi=U_Node(i);
    for j = 1 : No_UBoundary
        UNj=U_Node(j);
        for k = 1 : NDN
            irow=(i-1)*NDN+k;
            Urow=(UNi-1)*NDN+k;
            for m = 1 : NDN
            jcol=(j-1)*NDN+m;
            Ucol=(UNj-1)*NDN+m;
    S_UU(irow,jcol)= SN(Urow,Ucol);
            end
        end
    end
end
S_UU
S_II
```

```
S_UI
S_CI
S_UC
S_CC
C_UC
S_II_inv=inv(S_II);
K_CC_1=S_UU-S_UI*S_II_inv*S_UI';
K_CC_2=S_UC-S_UI*S_II_inv*S_CI';
K_CC_3=S_UC'-S_CI*S_II_inv*S_UI';
K_CC_4=S_CC-S_CI*S_II_inv*S_CI';
KCC=KCC+C_UC'*K_CC_1*C_UC;
KCC=KCC+K_CC_3*C_UC;
KCC=KCC+C_UC'*K_CC_2;
KCC=KCC+K_CC_4;
KCC
KIC=-S_II_inv*(S_UI'*C_UC+S_CI')
```

VITA

Mohamad Eftekharjoo
Mechanical and Aerospace Engineering
Kaufman Hall 238
Old Dominion University
Norfolk, VA 23529

Mohamad Eftekharjoo was born in Tehran in 1987. He received his Bachelor’s degree in Industrial Engineering in 2011. He plans to pursue a PhD in Mechanical and Aerospace Engineering from Old Dominion University.