

**FREE VIBRATION ANALYSIS OF RECTANGULAR THIN
PLATE USING FINITE ELEMENT
METHOD**

BY

MICHAEL UCHENNA NDUKWU (B.Eng)

REG. NO. 20124766338

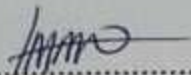
**A THESIS SUBMITTED TO THE POSTGRADUATE
SCHOOL, FEDERAL UNIVERSITY OF TECHNOLOGY,
OWERRI**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF MASTER OF ENGINEERING
(M.Eng) DEGREE IN CIVIL ENGINEERING
(STRUCTURES)**


SEPTEMBER 2021

CERTIFICATION

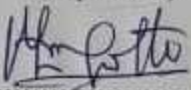
This is to certify that this research project "Free Vibration Analysis of Rectangular Thin Plate using Finite Element Method" by Michael Uchenna Ndukwu, Registration number 20124766338, is hereby approved for the award of Masters of Engineering (M.Eng) degree in Civil Engineering (Structures) of Federal University of Technology, Owerri.


.....
Engr. Dr O.M. Ibearugbulem
(Supervisor)

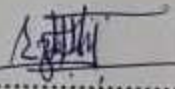
07/09/2021
.....
Date


.....
Engr. Dr. L. Anyaogu
(Supervisor)

07/09/2021
.....
Date


.....
Rev. Engr. Prof. L.O. Ettu
(Head of Department)

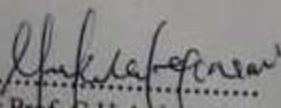
25/01/2022
.....
Date


.....
Engr. Prof. J.C. Ezeh
(Dean, SEET)

25/01/2022
.....
Date

.....
Prof. C.C Eze
(Dean of Postgraduate School)

.....
Date


.....
Engr. Prof. C.H Aginam
(External Examiner)

07/09/21
.....
Date

DEDICATION

This project report is dedicated to Jesus Christ, Our Lady of the Rosary and to my parents.

ACKNOWLEDGEMENTS

Firstly, I acknowledge God for his sustenance and gift of life throughout the duration of this project.

I would like to express my sincere gratitude to my supervisors, Engr. Dr. O.M. Ibearugbulem and Engr. Dr. L Anyaogu for their continuous support for my thesis and related research, for their patience, motivation, immense knowledge, insightful comments and encouragement which incited me to widen my research from various perspectives.

Their guidance helped me all through my research and writing of this thesis.

Besides my supervisors, I wish to acknowledge the Dean, School of Engineering and Engineering Technology, Engr. Prof. J.C. Ezeh and the Head of Department of Civil Engineering, Engr. Dr. L. Ettu

My sincere thanks also go to Engr.Prof. D.O. Onwuka, Engr. Dr. L.O. Osuagwu, Engr.Prof. B.C. Okoro, Engr. Prof. (Mrs.) B.U. Dike, Engr. Dr. U.C. Anya, Engr. Dr. (Mrs.) C.E. Okere, Engr. Dr. H.U. Nwoke, Engr A.N. Nwachukwu, Engr. F.C. Njoku, Engr. Rev. Dr. N.L. Nwakwasi, Engr. Dr. (Mrs.) J.I. Arimanwa, Engr. K.C. Nwachukwu, Engr. K.O. Njoku, C.O. Igbojiaku, Engr. S. Agbo, Engr.S.E. Iwuoha, Engr. A.U Igbojaku Engr. O.P. Okorie, Engr. E.O. Ithemgbulam, Engr. (Mrs.) J. Maduagwu, Engr. A.P.C Amanze, Engr. Dr. (Mrs.) C.T. Awodiji and Engr.C.A. Ajoku.

Finally, I remember my parents Mr. and Mrs. Ndukwu, my brothers, and sisters for supporting me throughout the period of writing this research.

TABLE OF CONTENTS

	Page
Title Page	i
Certification	ii
Dedication	iii
Acknowledgements	iv
Abstract	v
Table of contents	vi
List of tables	ix
List of figures	xi
Definition of Notations	xii
CHAPTER ONE: INTRODUCTION	1
1.1 Background of Study	1
1.2 Statement of Problem	5
1.3 Objectives of the Study	6
1.4 Justification of Study	6
1.5 Scope of Study	6
CHAPTER TWO: LITERATURE REVIEW	8
2.1 Vibration of plates	8
2.2 Past work on vibration of plates	10
2.3 Methods of plate analysis	18
2.3.1 Analytical method of plate analysis	18
2.3.1.1 Variational Iteration method-II	18

2.3.1.2	Superposition method	19
2.3.1.3	Integral Transform method	20
2.3.2	Approximate method of plate analysis	21
2.3.2.1	Rayleigh-Ritz method	21
2.3.2.2	Galerkin's method	23
2.3.3	Numerical method of plate analysis	24
2.3.3.1	Finite difference method	25
2.3.3.2	Boundary element method	27
2.4	Finite element analysis of plates	29
2.4.1	Finite element method	29
2.4.2	Finite element method historical review	31
2.4.3	Recent analysis of plates using finite element method	31
2.5	MATLAB as Software for Analysis	34
CHAPTER THREE: METHODOLOGY		36
3.1	Formulation of the general flexural element stiffness matrix of a thin rectangular plate.	36
3.2	Formulation of inertia matrix of thin rectangular plate	55
3.3	Development of Computer program	61
3.4	Comparism of the present study and that obtained from previous Methods.	62
CHAPTER FOUR: RESULTS AND DISCUSSIONS		63
4.1	Results	63
4.1.1	Formulated General Flexural Element Stiffness Matrix of a Thin Rectangular Plate.	63
4.1.2	Formulated Inertia Matrix of a Thin Rectangular Plate.	64

4.1.3	MATLAB Program	65
4.1.4	Numerical Example.	65
4.1.4.1	Detailed graphical representation of results for all boundary Conditions of thin rectangular plates used in this study.	67
4.2	Discussions of results	73
4.2.1	Discussion of results of CCCC rectangular thin plates.	73
4.2.2	Discussion of results of CCSS rectangular thin plates.	74
4.2.3	Discussion of results of CSCS rectangular thin plates.	75
4.2.4	Discussion of results of CSSS rectangular thin plates.	76
4.2.5	Discussion of results of CCCS rectangular thin plates.	77
CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS		79
5.1	Conclusions	79
5.2	Recommendations	79
5.3	Contributions to knowledge	80
	References	81
	Appendices	91

LIST OF TABLES

Table 4.1	The natural frequency (λ) for different aspect ratio and grid size (n) for CCCC plates.	135
Table 4.2	Results of natural frequency (λ) for CCCC plate of present study and the results of previous studies with their percentage difference for different aspect ratios.	136
Table 4.3	The natural frequency (λ) for different aspect ratio and grid size (n) for CCSS plates.	137
Table 4.4	Results of natural frequency (λ) for CCSS plate of present study and the results of previous studies with their percentage difference for different aspect ratios.	138
Table 4.5	The natural frequency (λ) for different aspect ratio and grid size (n) for CSCS plates.	139
Table 4.6	Results of natural frequency (λ) for CSCS plates of present study and results of previous studies with their percentage difference for different aspect ratios.	140
Table 4.7	The natural frequency (λ) for different aspect ratio and grid size (n) for CSSS plates.	141
Table 4.8	Results of natural frequency (λ) for CSSS plates of present study and results of previous studies with their percentage difference for different aspect ratios.	142
Table 4.9	The natural frequency (λ) for different aspect ratio and grid	

size (n) for CCCS plates. 143

Table 4.10 Results of natural frequency (λ) for CCCS plates of present study and results of previous studies with their percentage difference for different aspect ratios. 144

LIST OF FIGURES

Figure 1.1	A thin rectangular plate divided into finite element with numbered edges and X and Y axis, including plate dimension 'a' and 'b'	4
Figure 1.2	Rectangular finite plates of different boundary condition.	5
Figure 3.1	Pascal Triangular Shape function for non dimensional coordinates	42
Figure 3.2	A finite element mesh of a plate simply supported on opposite short edge	42
Figure 3.3	A thin plate with 12 degrees of freedom.	43
Figure 3.4	A thin plate with 12 degrees of freedom and nodal coordinates	44
Figure 4.1	CCCC rectangular plate.	65
Figure 4.2	CCSS rectangular plate.	65
Figure 4.3	CSCS rectangular plate.	66
Figure 4.4	CSSS rectangular plate.	66
Figure 4.5	CCCS rectangular plate.	67
Figure 4.6	Graphical Representation of Fundamental Natural frequency for CCCC Plates support conditions of Aspect Ratios of b/a	68
Figure 4.7	Graphical Representation of Fundamental Natural frequency for CCSS Plates support conditions of Aspect Ratios of b/a	69
Figure 4.8	Graphical Representation of Fundamental Natural frequency for CSCS Plates support conditions of Aspect Ratios of b/a	70
Figure 4.9	Graphical Representation of Fundamental Natural frequency for CSSS Plates support conditions of Aspect Ratios of b/a	71
Figure 4.10	Graphical Representation of Fundamental Natural frequency for CCCS Plates support conditions of Aspect Ratios of b/a	72

DEFINITION OF NOTATIONS

NOTATIONS	MEANING
a	Width of a rectangular plate.
b	Length of a rectangular plate.
ρ	Mass density of material.
h	Thickness of plate.
p	Time factor for free vibration.
L_u	Linear operator.
N_u	Non-linear operator.
g(t)	Forcing term.
U	Restricted variation.
n	nth variation.
ξ	Non dimensional coordinates.
η	Non dimensional coordinates.
J	Number of separate problems.
Φ	Unknown auxiliary function.
D	Flexural rigidity.
(U)	Strain energy.
E	Young's modulus.
ν	Poisson's ratio.
$\nabla^2 \nabla^2$	Laplacian operator.
(W)	Work.
T	Kinetic energy.

Π	Total potential energy.
ω	Circular frequency of plate.
F	Displacement force.
P_0	Applied uniformly distributed static load.
M	Mass.
C	Damping
K	Stiffness matrices.
F_t	Vector of active nodal forces.
\dot{m}	Mass per unit area of the plate.
λ	Fundermental natural frequency of plate.
w	Deflection equation of plate.
$[N]$	Deflection shape function of plate.
$[\psi]$	Coefficients of the displacements
$[N_i]$	Nodal value displacement profile.
$\frac{\partial^2 w}{\partial x^2}$	Second partial derivative of w with respect to x.
$\frac{\partial^2 w}{\partial xy}$	Second partial derivative of w with respect to xy
$\frac{\partial^2 w}{\partial y^2}$	Second partial derivative of w with respect to y.
θ_R	Rotation of the node about R axis.
θ_Q	Rotation of the node about Q axis.
P	Aspect ratio

ABSTRACT

This research work presents Free Vibration Analysis of Rectangular Thin Plate Using Finite Element Method. The analysis is limited to five boundary conditions. These boundary conditions are plates clamped on four edges (CCCC), plates clamped and simply supported on adjacent edges (CCSS), plates clamped on two opposite edges and simply supported on the other two opposite edges (CSCS), plates clamped on one edge and simply supported on three edges (CSSS) and plates clamped on three edges and simply supported on one edge (CCCS). A shape function which satisfies the twelve degrees of freedom of plate was assembled from the Pascal triangle to formulate a stiffness matrix which is referred to as the general flexural element stiffness matrix of thin rectangular plate. The fourth order differential equation of plate in vibration was analyzed with the shape function to derive the general stiffness (K) of the plate and inertia stiffness (K_i) subject to vibration. Analysis with the finite element method, the individual stiffness for CCCC, CCSS, CSCS, CSSS, CCCS was obtained. A grid size discretization, one of the major importance of finite element method; was applied to determine the approximate values of the fundamental natural frequencies of the rectangular thin plates in vibration. A MATLAB program was generated to compute the fundamental natural frequencies (λ) for plates of various aspect ratios (from 1.0 to 2.0 at an increment of 0.1) and grid size (n) (from 3 to 21 etc. at odd number increment) and the results were tabulated. The odd number increment for the grid size is to make the central deflection of the plate coincide with the central node of the plate for easy calculation. The maximum percentage difference of the natural frequencies obtained between this study and previous research works done by Onwuka et al., Njoku, Leissa et al., Sakata et al., Chakraverty and Gorman are 0.4060%, 0.4070% and 3.1600% for CCCC boundary condition, 0.5503%, 0.5675% and 0.1283% for CCSS boundary condition, 0.4025%, 0.1629% and 0.1208% for CSCS boundary condition, 3.6188%, 1.4968% and 0.1034% for CSSS boundary condition and 0.4394% and 0.1461% for CCCS boundary condition. The natural frequency values obtained by the present study are close when compared with other approximate methods of Onwuka et al., Njoku, Leissa et al., Sakata et al., Chakraverty and Gorman which are clearly shown on the line graph.

Keywords: Vibration Analysis, Rectangular Thin Plate, Finite element, Stiffness, MATLAB.

CHAPTER ONE

INTRODUCTION

1.1 Background Information

A plate is a structural element that is relatively thin in Y direction compared with the X and Z direction, and it's flat. From a geometrical point of view, a plate is very much like a membrane. Typically, the flexural (bending) stiffness arises because a plate is considerably thicker than a membrane relative to its other dimensions. Although a plate is typically thicker than a membrane, the ratio of its thickness to its average lateral dimension is usually taken to exceed $1/20$ to represent its fundamental vibration mode reasonably accurately with classical thin plate theory (Leissa and Qatu, 2011).

Plates are important structural elements. They may exist in many applications like in Aeroplane, Ship deck, concrete slab etc. In civil engineering, flat panels exist in various steel or concrete structures. They may be various shapes like rectangular, circular, rhombic, triangular, trapezoidal and others. Although shells have indeed, wider application in almost every engineering field, plates also present a reasonable introduction to the analysis of those complex structures.

Plates give us the chance to study the influence of bending in two dimensions without having tangential stresses caused by transverse vibration motion (Leissa and Qatu, 2011).

Structural components like plates are generally subjected to dynamic loading during their working life. Very often these components may have to perform in a severe dynamic environment where the maximum damage results from the resonant vibration. In the vehicles used in transport such as ships and aircrafts, vibration results in discomfort to the crew and

passengers. Human reaction to vibration in the range of 4Hz to 16Hz is more sensitive because natural frequencies of the body can be excited and may cause serious physiological effects.

Susceptibility to fracture of materials due to vibration is determined by stress and frequency. The maximum amplitude of the vibration must be limited for the safety of the structure. Hence vibration analysis has become very important in designing a structure to know in advance its response and to take necessary steps to control the structural vibrations and its amplitudes (Asmin, 2008).

The finite element analysis is a powerful computational technique for solving engineering problems having complex geometries that are subjected to general boundary conditions. While the analysis is being carried out, the field variables are varied from point to point, thus possessing an infinite number of solutions in the domain. To overcome this difficulty, finite element analysis is used; the system is discretized into a finite number of parts known as elements by expressing the unknown field variables in terms of the assumed approximating functions within each element. For each element, the systematic approximate solution is constructed by applying the variational or weighted residual methods. These functions (also called interpolation function) are included in terms of field variables or specific points referred to as nodes (Vanam Rajyalakshmi and Inala, 2012).

The essential feature of the finite element method is the means by which the differential equations of equilibrium of the elastic continuum are approximated by a set of algebraic equilibrium equations. This procedure is generally looked upon as the situation for the actual continuum of an assemblage of discrete structural elements, interconnected at a finite number of modal points. In effect, the continuum may be visualized as being physically cut up into finite

element system, the material properties of the original material being retained in the elements. The analysis involves the evaluation of the element elastic properties, which are represented by the stiffness matrix expressing the relationship between element nodal forces and displacements. By appropriate superposition of these element stiffnesses, the stiffness matrix of the entire assemblage may be obtained. Finally, the nodal equilibrium equations, expressed in terms of this structural stiffness matrix, must be solved simultaneously for nodal displacement of the complete system. (Ray and Joseph, 2003).

In addition, the finite element method is endowed with three basic features that account for its superiority over other competing methods. First, a geometrically simple subdomain, called finite elements. Second, over each finite element, the approximation functions are derived using the basic idea that any continuous function can be represented by a linear combination of algebraic polynomials. Third, algebraic relations among the undetermined coefficients (i.e., nodal values) are obtained by satisfying the governing equations, often in a weighted-integral sense, over each element. Thus, the finite element method can be viewed, in particular, as an element-wise application of the Rayleigh-Ritz or Weighted-Residual methods (Reddy, 1993).

In conclusion, this study is a finite element method and its approach is to analyze a thin rectangular plate under free vibration. The Paschal triangular shape function and the fourth order differential equation of plate in vibration is used to generate a stiffness matrix and the inertia matrix which when applied to the plate model produces a numerical solution of the plate under free vibration. This research developed a MATLAB program to reduce the calculation errors and the time spent during the plate analysis. Application of different

support condition in the finite element model gives the value of the natural frequencies with respect to the aspect ratio provided. The fundamental natural frequencies are the most important part of free vibration of plates and this research sought to obtain such frequencies. Figure 1.1 shows a typical plate model that will be used in this study.

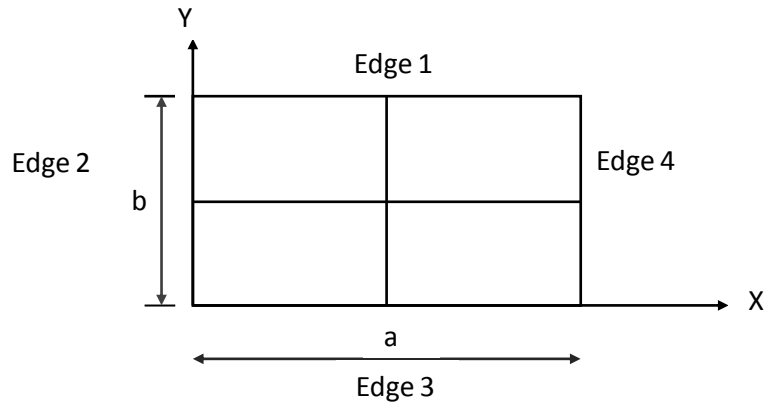
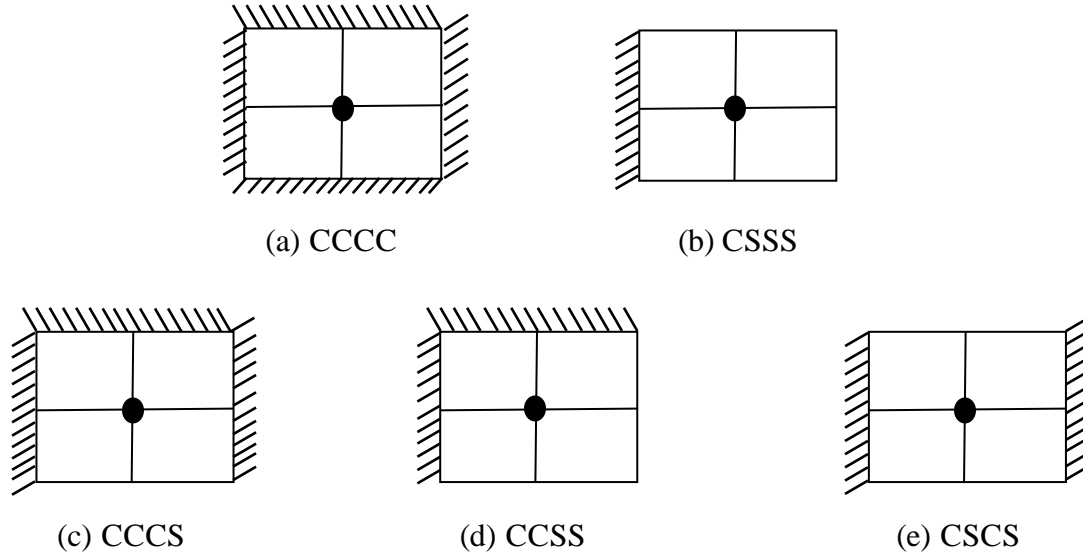


Figure 1.1 A thin rectangular plate divided into finite element with numbered edges and X, Y axis, including plate dimension 'a' and 'b'

The configuration of the plate shown in Figure 1.2 will be adopted in the analysis of plate problems in this study. The meaning of each plate boundary condition is listed as follows.

- CCCC Clamped on four edges.
- CSSS Clamped on one edge and simply supported on the remaining three edges.
- CCCS Clamped on three edges and simply supported on one edge.
- CCSS Clamped on adjacent edges and simply supported on the other edge.
- CSCS Clamped on two opposite edges and simply supported on the other two opposite edges.



Figures 1.2 (a to f) Rectangular finite plates of different boundary condition

1.2 Problem Statement

Formulating the general flexural element stiffness matrix of thin rectangular plate becomes a significant issue due to the boundary conditions and chosen deflection shape function constraints.

Application of the general flexural element stiffness matrix equation of a plate in order to obtain the stiffness matrix and the inertia matrix took some time and require a great deal of concentration to avoid confusion. The stiffness matrix of the plate becomes tedious and cumbersome as the plate is discretized further. Hence, there is a need to employ a mathematical tool that can handle the analysis effectively. The mathematical tool used in the analysis of plate is MATLAB computer program.

1.3 Objectives

The main objective of this study is to perform free vibration analysis of rectangular thin plate using Finite Element Method (FEM). The specific objectives are as follows:

- (i) To formulate the general flexural element stiffness matrix of a thin rectangular plate.
- (ii) To formulate the inertia matrix of a thin rectangular plate.
- (iii) To develop a MATLAB computer program that will use the stiffness matrix and inertia matrix to obtain the resonating frequencies of plates of five boundary conditions.
- (iv) To compare the result obtained from the present study with those of other approximate methods.

1.4 Justification of Study

On completion of this work, the following should be benefited from it:

- (i) A good knowledge of the use of Finite Element Method to perform the free vibration analysis of a thin rectangular plate.
- (ii) A reliable computer program that gives the approximate resonating frequencies within a short time with minimum error

1.5 Scope of Study

This research work is limited to the free vibration analysis of rectangular thin plates using finite element method. The boundary conditions considered for analysis were CCCC, CSSS, CCCS, CCSS and CSCS plates.

Chapter two titled “Literature Review” presents the investigation of different research work carried out on vibration and methods of plate analysis by many scholars. Chapter three deals with research methodology. This shows the formulation of the general flexural element stiffness

matrix of a thin rectangular plate. The general stiffness matrix and inertia matrix equations were derived and the fundamental natural frequency equation for the various boundary conditions of the plate was obtained. Also, in this chapter, the algorithm for the computer program was listed. Results and discussions which is the title of chapter four dealt with obtaining the fundamental natural frequency of the plate with various boundary conditions using the MATLAB program generated. Aspect ratios ranging from 1.0 to 2.0 with 0.1 increments were also applied.

Solutions from the present study were compared with the previous works and general discussion and comments were made with regards to the comparisons. Chapter five titled “Conclusions and Recommendations” dealt with drawing of conclusion based on the results obtained in chapter four and the necessary recommendations that follow it.

CHAPTER TWO

LITERATURE REVIEW

2.1 Vibration of Plates

Vibration is the mechanical oscillation of a particle, member, or a body from its position of equilibrium. It is the study that relates the motion of physical bodies to the forces acting on them. Plates with different shapes and boundary conditions have applications in different structures such as aerospace, machine design, telephone industry, nuclear reactor technology, naval structures and earthquake-resistant structures. The dynamic behavior of flexible, flat, thin, rectangular plate has received huge attention in recent years due to its technical importance (Chakraverty, 2009).

Thin rectangular plate structures are mostly used in the industrialized world and in a broad range of engineering applications, for example, electronic circuit board design, solar panels, and bridge decks.

The plate stability which means the ability of a plate structure to withstand sudden change, dislodgment, or overthrow when subjected to loading is associated with various physical effects that lead to vibration.

High vibrations of flexible structure systems cause noise, fatigue, wear, destruction, human discomfort, and reduced system effectiveness. Due to above defects, the vibration of flexible thin plate structures needs to be controlled. In the process of looking for ways to control these defects, the vibration of the elastic plates has been treated widely from researchers with different boundary conditions, both from theoretical and experimental points of view. It is necessary to find an approximate or accurate model of the plate structure to control the vibration of a plate

well. Suitable modeling of a dynamic system would result in good control (Tavakolpour, Mat Darus, Mailah, and Tokhi, 2010).

Dynamic effects of time-dependent loads on a structure are studied in structural dynamics. Structural dynamics deals with time-independent motions of structures, primarily, with the vibration of structures, and analysis of the internal forces associated with them. The dynamics of plates can be modeled mathematically by partial differential equations based on Newton's laws or by integral equations based on the considerations of virtual work.

The derivation of the governing differential equation of motion is a simple extension of a static case by adding effective forces to the plate that result from accelerations of the mass of the plate. The various kinds of motion of the plate are the free vibration which deals with the natural characteristics of the plates, and these natural vibrations occur at discrete natural frequencies, depending only on geometry and materials of plates. The forced vibration which is another kind of motion of plate comes in two kinds: a harmonic response, when a periodic force is applied to the plate; and a transient response, when the applied force is not a periodic force (Ali-Reza, Jalil, Majid, and Habibolahiyani, 2011).

According to Ali-Reza et al. (2011), the differential equation of undamped forced motion of plate is as expressed in Equation (2.1).

$$D\nabla^2\nabla^2w(x, y, t) = P(x, y, t) - \rho h \frac{\partial^2 w}{\partial t^2}(x, y, t) \quad (2.1)$$

Where ρ is the mass density of the material, h is the plate thickness. P and W , are functions of time for natural or free vibrations, $P(x,y,t)$ is set to zero i.e. Equation (2.1) was reduced to Equation (2.2).

$$D\nabla^2\nabla^2w(x, y, t) + \rho h \frac{\partial^2 w}{\partial t^2}(x, y, t) = 0 \quad (2.2)$$

2.2 Past Work on Vibration of Plates

The first mathematical approach to the membrane of very thin plates was formulated by L. Euler in 1766. He solved the problems of free vibrations of rectangular, triangular and circular elastic membranes by using the analogy of two system of stretched strings perpendicular to each other. His student, Jacques Bernoulli, extended Euler's analogy to plates by replacing the net of spring with a gridwork of beams having only bending rigidity (Szilard, 2004).

The first-order shear deformation theory (FSDT) has been employed widely to establish finite element models for free vibration analysis of the composite laminated plates. The effects of lamination and extension-bending coupling, shear and twist-curvature couplings on the lowest frequencies and corresponding mode shapes for free vibration of laminated anisotropic composite plates was investigated using a finite element method with quadratic interpolation functions and five engineering degrees of freedom (DOF). The free and forced vibration response of laminated composite folded plate structures was predicted by a nine-node Lagrangian plate bending finite element with five engineering DOF per node that incorporated rotary inertia. A nine-node isoparametric plate bending elements was used for the analysis of free undamped vibration of rectangular isotropic and fiber-reinforced laminated composite plates, and an effective mass lumping scheme with rotary inertia was introduced (Zhang and Yang, 2008).

The Free Vibration analysis of stiffened laminated composite plates was performed using the layered (Zig Zag) finite element method based on the first-order shear deformation theory. The layers of the laminated plate were modeled using nine-node isoparametric degenerated flat shell element, and the stiffeners were modeled as three-node isoparametric beam elements based on Timoshenko beam theory. Bilinear in-plane displacement constraints were used to maintain the

inter-layer continuity, and a special lumping technique was used in deriving the lumped mass matrices (Guo, Harik, Wei-Xin, 2002).

Latheswary, Valsarajan, Sadasiva. (2004) investigated the static and free vibration analysis of moderately thick laminated composite plates using a 4-node finite element formulation, based on higher-order shear deformation theory and the transient analysis of layered anisotropic plates using a shear deformable 9-noded Lagrangian element, based on first-order shear deformation theory.

The free vibration analysis of multi-layered thick composite plates was studied by a finite element procedure based on an accurate higher-order theory which accounted for the realistic vibration of in-plane and transverse displacements through the thickness. The vibration and stability problems of cross-ply and angle-ply laminated composite plates were investigated using general higher-order theories of laminates which considered the complete effects of transverse shear and normal deformations (Matsunaga, 2000 and Matsunaga, 2002).

The free in-plane vibration of rectangular plates involves two independent displacement variables, so it becomes more difficult to get its exact solution compared with thin plate vibration where there is only a single independent displacement variable. The pioneering work was done by Lord Rayleigh, who dealt with what was referred to as 'Simply supported' plates. Later Gorman obtained the exact solutions for the rectangular plates with two opposite edges simply supported, and the remaining edges being both clamped or both free. In Gorman's work, a quarter of the rectangular plate was analyzed to avoid the calculation difficulties and missing

problems of repeated Eigenvalues and the interpretations of the mode shapes as well (Gorman, 2006).

Recently, Xing and Liu (2009) and Xing and Liu (2011) obtained all possible exact solutions for isotropic and orthotropic rectangular plate via the direct separation of variables. The exact solutions for the isotropic plates with two opposite edge simply supported and the other two opposite edges being asymmetrical were not achieved before. In addition, Liu and Xing (2011) obtained the exact solutions for free in-plane vibration of rectangular orthotropic plates with four edges not being simply supported.

According to Xing, Xu, Liu (2013), the free vibrations of rectangular Mindlin plates; Mindlin was the first who gave the exact solution of simply supported plate and investigated the frequency variation of plates; a pair of parallel edges of which are simply supported, and the other pair are free. Later Srinivas (1970) obtained the exact solutions for the homogeneous and laminated rectangular Mindlin plates with four simply supported edges while Mindlin obtained the exact solutions, upon two main restrictions:

- (i) That the length/width must be a ratio of integers.
- (ii) That for each value of Poisson's ratio, all the allowable modes must have the same frequency.

Xiang and Wei (2004) employed the Levy solution approach in conjunction with the state space technique to derive the analytical Eigensolutions of rectangular Mindlin plates with two opposite

edges simply supported. Hashemi and Arsanjani (2005) presented the exact characteristic equations for plates with one pair of opposite plate edges simply supported.

Shimpi and Patel (2006) obtained the exact solutions of a simply supported plate. Thai and Kim (2012) obtained the Levy-type exact solutions for plates with two opposite edges simply supported and the other two edges having arbitrary boundary conditions by applying the state space approach.

Senjanovic and Tomic (2013) reduced a system of three equations to a single equation in terms of bending deflection only which was fundamental variable and obtained closed-form solutions for a simply supported plate and the characteristic equation for a plate with two opposite edges simply supported.

Gupta, Anupam and Dharam (2009) discussed the effect of thermal gradient on some vibration problems of orthotropic visco-elastic plates of variable thickness. He solved the problem of thermal effect on vibration of a non-homogeneous orthotropic rectangular plate having a bi-directional parabolically varying thickness and free vibration of a clamped visco-elastic rectangular plate having bi-direction exponentially thickness variations.

Kharde, Mahale, Bhosale and Thorat (2013) developed flexural vibration of thick isotropic plates by using exponential shear deformation theory. He used the exponential term in its displacement field to calculate the strain and stresses, satisfying the stress-free boundary conditions at the top and bottom of the plate. The principle of virtual work approach was used to obtain the governing differential equations and boundary conditions.

Aginam, Chdolie and Ezeagu (2012) used the direct variation method in the analysis of isotropic thin rectangular plate. A mathematical model that is based on direct variational procedures and potential energy principle was developed and successfully applied to thin rectangular plates with two opposite edges clamped and other edges simply supported and thin rectangular plate with one edge clamped and the three other sides simply supported. The coordinate functions, which must satisfy the geometric and natural boundary conditions were carefully constructed and applied to classical plate equation.

In vibration problems of rectangular Kirchhoff plates, exact solutions were limited to some boundary and support conditions. For plates with all edges simply supported or with a pair of opposite edges simply supported the Navier and Levy solution methods can be used to generate exact value. Reddy (2007). Xing and Liu (2009) presented a new method to obtain exact eigensolutions for any combination of separable simply supported and clamped boundary conditions. When the plate is arbitrarily supported or elastically restrained along its edges with attached concentrated mass and spring elements, its solution resorts to numerical approximate method which comes in different approaches i.e. the finite differences and the finite element method, the differential quadrature (DQ) and the generalized differential quadrature method (GDQ), the supper position method, the analytical numerical combined method, the discrete singular convolution method (DSC) and the Rayleigh and Ritz method.

Recently, Li, Zhang, Du and Liu (2009) described a mathematical approach for analysis of rectangular plates with general elastic supports. They assume displacement solution in the form

of double Fourier series with supplementary terms given by products of single Fourier series and sufficient smooth functions.

Lorenzo (2011) highlighted that Ritz technique has conceptual simplicity, wide flexibility, high reliability and computational efficiency in solving the vibrational problem of thin plates. The Ritz procedure consists in approximating the normal displacement variable through a linear combination of global assumed functions, known as admissible functions, trial functions or basis functions, each satisfying at least the geometrical boundary conditions of the plate. The unknown coefficients of the combination can be obtained from the minimization of the energy functional of the system. Convergence of the exact solution is guaranteed if the trial functions are linearly independent and form a mathematically complete set. In the use of the Trigonometric Ritz method for general vibration analysis of rectangular Kirchhoff plates, Lorenzo predicted the natural flexural frequencies of plates using in-plane loads, elastically restrained edges, elastic concentrated masses, intermediate line and point supports.

Wang & Wang (2008) developed the transverse free vibration of viscoelastic rectangular thin plates with linearly varying thickness and multiply all-over part-through cracks using the differential quadrature method. Based on Mindlin plate theory (MPT), a set of exact close-form characteristic equations incorporating shear deformation and rotary inertia was proposed by Hosseini – Hashemi, Heydar and Hossein (2010) to analyze free vibration problem of moderate thick rectangular plates with an arbitrary number of all-over part-through cracks.

Huang & Hung (2011) examined the vibration suppression of a simply supported plate under a harmonic force at its center by utilizing an active dynamic absorber. The absorber consists of two

piezo patches bonded to the top and bottom surfaces of the plate and proper electric circuits. One patch is acting as the sensor to sense the velocity information while the other one acts as an actuator. Their results revealed this active absorber could effectively reduce the plate vibration for both controlled modes and the uncontrolled resonances.

Ali (2013) employed Hamilton's principle to derive the differential equation governing the vibration of single – span thin beams having a number of piezoelectric actuators attached to their surfaces. He employed a linear classical optimal control algorithm with displacement – velocity, velocity – acceleration feedbacks and three types of external excitation, including a simple rectangular impulse, a moving load, and a moving mass.

In rectangular plate supports, Huang, Ma, Sakiyama, Mastuda and Morita (2007) offered a discrete green function method for free vibration analysis of rectangular plates with point supports. Altekin (2010) investigated the bending of orthotropic super-elliptical plates on intermediate point supports. The Ritz method was used and the total potential energy functional was modified by introducing the Lagrange multipliers to improve the accuracy of the stress resultants.

Predicting the dynamic performance of continuous systems under moving loads, Rofoei and Nikkhoo (2009) proposed a classical closed-loop control algorithm to suppress the vibration of a simply supported rectangular plate excited by a moving mass via a number of bounded active piezoelectric patches. In 2012 they also scrutinized the inertial effects and the trajectory of the moving load on the dynamic response of a simply – supported edge thin rectangular plate.

Zarfam, Khaloo and Nikkhoo (2013) performed an investigation on the vibration of a Euler-Bernoulli beam traversed by a moving mass accompanied with horizontal support excitation. The analytical solution to the dynamic response of a circular thin plate carrying a moving load was obtained by Uzal and Sakman (2010) by disregarding the load inertia.

Vaseghi, Amiri, Nikkhoo, Davoodi and Ebrahimzadeh (2013) provided a semi-analytical simulation of a shear deformable plate vibration due to traveling inertial loads considering a general load distribution pattern and plate boundary condition.

Ali and Nikkhoo (2013) determined the resonance of a two-lane slab-type bridge caused by the traveling vehicles by using eigenfunction expansion method. A single-span rectangular plate with SFSF (F: free edge), (S: simply supported) boundary condition is considered to simulate the bridge deck. The spacing of the traversing mass is focused to detect the maximum DAF (dynamic amplification factor) of the plate.

Li, Iu and Kou (2008) studied the free vibration of rectangular sandwich plates with functionally graded cores and functionally graded (FG) plates in a thermal environment based on the three-dimensional linear theory of elasticity. They expanded the displacements of the plates by Chebyshev polynomials and obtained the natural frequencies by using Ritz method. Amini et al. (2009) developed a method for three-dimensional free vibration analysis of rectangular FG plates resting on an elastic foundation using Chebyshev polynomials and Ritz method.

Xing and Liu (2009) calculated the in-plane vibrational frequencies of rectangular plates by employing variable separation method and using the Rayleigh quotient variational principle.

2.3 Methods of Plate Analysis

There are various methods of analyzing plates in vibration. They are listed as follows:

2.3.1 Analytical Methods of Plate Analysis

2.3.1.1 Variational Iteration Method – II

In recent times, scientists have proposed and applied some analytical methods to nonlinear equations, like the vibration behavior of quintic nonlinear in an extensional beam or two-parameter elastic substrate based on the three mode assumptions which were investigated. The parameter expansion method was used to obtain the approximate expressions of nonlinear frequency – amplitude relationship for the first, second and third modes of vibrations. According to Pakar and Bayat (2012), Hamiltonian approach was applied to the analysis of the nonlinear free vibration of a tapered beam.

A new analytical method called the Variational Iteration method – II (VIM – II) for the differential equation of the large deformation of a cantilever beam under point load at the free tip was applied by Ghaffarzadeh and Nikkar (2013).

The basic concept of one of the analytical methods of vibration called Variational Iteration method is highlighted below.

The general differential equation is as expressed in Equation (2.3)

$$L_u + N_u = g(t) \tag{2.3}$$

Where L is a linear operator, and N is a nonlinear operator, g(t) an inhomogeneous or forcing term.

The correct functional for variation iteration method is as expressed in Equation (2.4)

$$U_{n+1}(t) = U_n(t) + \int_0^1 \lambda (L_{u_n}(\tau) + N_{\tilde{u}_n}(\tau) - g(\tau)) dt \quad (2.4)$$

Where λ is a general Lagrange multiplier, which can be identified optimally via the variational theory, the subscription n denotes the nth approximation, \tilde{U}_n is considered as a restricted variation. For linear problems, its exact solution can be obtained by only one iteration step due to the fact that the Lagrange multiplier can be exactly identified. In this method, the problems are initially approximated with possible unknowns and it can be applied to non-linear problems without linearization or small parameters. The approximate solutions obtained by this method rapidly converge to the exact solution (Jodeiri and Imani, 2015).

2.3.1.2 Superposition Method

Superposition method is one of the analytical methods used in determining the natural frequencies of vibrating plates. This method was developed by Gorman in 1999. He analyzed free vibration of rectangular plates with numerous different types of boundary conditions and point supports. This powerful method gives the best values for the natural frequencies of plates having various aspect ratios when compared with other computing methods.

In many cases, the results from the superposition method are the benchmarks for the natural frequencies because it is very efficient and accurate for a range of geometric shapes. The solutions obtained by this method would give an upper bound and lower bound or lower bound result depending on whether the boundary conditions of the building are stiffer or more flexible than those of the actual system. The superposition method is an analytical-type method of a

solution because it is closed form of an infinite series. Its solution always satisfies the governing differential equation and all boundary condition (Patiphan, 2015).

Mathematically, the superposition method requires Equation (2.5) to be true.

$$w(\xi, \eta) = \sum_{i=1}^j w_i(\xi, \eta) \quad (2.5)$$

Equation (2.5) is decomposed further to Equation (2.5a)

$$w(\xi, \eta) = w_1(\xi, \eta) + w_2(\xi, \eta) + \dots + w_j(\xi, \eta) \quad (2.5a)$$

Where j is the number of separate problems ξ and η is the non-dimensional coordinates.

Each building block $w_i(\xi, \eta)$ is solved separately and the solutions are superimposed to obtain original plate solution $w(\xi, \eta)$.

2.3.1.3 Integral Transform Method

This is another type of analytical method of which its various types have been successfully used to solve many kinds of mixed boundary value problems in engineering.

By the integral transform method, most problems are usually reduced to the final equation form of integral equations of a special type depending on the integral transform techniques used and some of their solutions have already been provided (Polyanin and Manzhirov, 2008).

The integral transform applied in the static bending problem of rectangular plates are in the form of inhomogeneous Fredholm-type of the second kind as expressed in Equation (2.6) and Equation (2.7)

$$\Phi_1(p) + \int_0^1 [k_{11}(p, r)\Phi_1(r) + k_{12}(p, r)\Phi_2(r)]dr = f_1(p) \quad (2.6)$$

$$\Phi_2(p) + \int_0^1 [k_{21}(p,r)\Phi_1(r) + k_{22}(p,r)\Phi_2(r)]dr = f_2(p) \quad (2.7)$$

where p is an independent variable and $0 \leq p \leq 1$, r is a dummy variable, $\Phi_1(p)$ and $\Phi_2(p)$ are the unknown auxiliary functions to be determined and introduced in the Hankel integral transform techniques which is related to the unknown constants of integration of plate's differential equation. $f_1(p)$ and $f_2(p)$ are known functions that involved to the load applied on the plate, and $k_{11}(p,r)$ through $k_{22}(p,r)$ are the kernels of integral equation (Thonchangya, 2001).

2.3.2 Approximate Method of Plate Analysis

2.3.2.1 Rayleigh-Ritz Method

According to Leissa (2005), Rayleigh-Ritz method is an approximate method that makes use of the energy concept by determining the maximum strain energy and maximum kinetic energy of the systems and then, minimizing these energy terms to obtain the condition of equilibrium equations. An inherent advantage of this method is that no need to solve the governing differential equation, only the suitable series of functions are required to be chosen in satisfying exactly the prescribed geometric boundary conditions of the problem.

According to Thongperm, Patiphan, Jakarin and Yos (2015), Rayleigh method was first introduced by Lord Raleigh in 1877 for solving the vibration problems to obtain the fundamental natural frequency of continuum systems by assuming the mode shape and setting the maximum values of potential and kinetic energy in a cycle of motion equal to each other. In 1908, Ritz described his method for solving the boundary value problems and the eigenvalue problems to determine the frequencies and their corresponding mode shapes by choosing multiple admissible

displacement functions and minimizing a functional involving both potential and kinetic energies.

The rayleigh-ritz method can be illustrated in Equation (2.8) using the governing biharmonic differential equation for the static bending problem of isotropic rectangular thin plates with uniform thickness(h) in terms of deflection $W(x,y)$ and subject to a distributed load $Q(x,y)$.

$$\frac{\partial^4 W(x, y)}{\partial x^4} + 2 \frac{\partial^4 W(x, y)}{\partial x^2 \partial y^2} + \frac{\partial^4 W(x, y)}{\partial y^4} = \frac{Q(x, y)}{D} \quad (2.8)$$

$$D = \frac{Eh^3}{12(1-\nu^2)} \quad (2.9)$$

Where D is the plate rigidity, E and ν are Young's modulus and Poisson's ratio of the plate, respectively.

The strain energy of plate is expressed as shown in Equation (2.10)

$$U = \frac{D}{2} \iint_A \left\{ \left[\nabla^2 w(x, y) \right]^2 - 2(1-\nu) \left[\left(\frac{\partial^2 W(x, y)}{\partial x^2} \right) \left(\frac{\partial^2 W(x, y)}{\partial y^2} \right) - \left(\frac{\partial^2 W(x, y)}{\partial x \partial y} \right)^2 \right] \right\} dx dy \quad (2.10)$$

Where A is the total area of the plate and ∇^2 is the Laplacian operator.

The work (W) that produced by the external load is expressed in Equation (2.11)

$$W = \frac{1}{D} \iint_A [Q(x, y) W(x, y)] dx dy. \quad (2.11)$$

The total potential energy $\Pi = U - W$. (2.11a)

For free vibration of plates, the kinetic energy (T) is as given in Equation (2.12)

$$T = \frac{ph\omega^2}{2} \iint_A [W(x, y)]^2 dx dy \quad (2.12)$$

Where p is the density of mass per unit area of the plate and ω is the circular frequency of plate vibration. Rayleigh-Ritz method applied the deflection function $W(x,y)$ is terms of admissible function $\psi_{ij}(x,y)$ satisfying boundary conditions of the plate as expressed in Equation (2.13)

$$W_{(x,y)} = \sum_{i=1}^I \sum_{j=1}^J a_{ij} \psi_{ij}(x,y) \quad (2.13)$$

where I, J are the highest integer numbers to be used in approximation of $W_{(x,y)}$, a_{ij} is defined to be the adjustable arbitrary coefficients, and the function $\psi_{ij}(x,y)$ may assume to be the algebraic polynomials or trigonometric functions satisfying essential (geometric) boundary conditions i.e. (slope or deflection) of the plate.

Rayleigh-Ritz method always provides upper bound solutions for vibration frequencies, and the upper bound solutions will come to the exact solution as the number of admissible functions is sufficiently large and if the used admissible functions are from a mathematically complete set of functions.

Rayleigh's principle says that for an unknown system, in steady state, the maximum kinetic energy during free vibration must be equal to the maximum potential energy, which in case of elastic bodies is predominately the strain energy evolved (Amitabha, 2015).

2.3.2.2 Galerkin's Method

Galerkin's method is a variation of the method of weighted residual which provides a very powerful approximate solution procedure that is applicable to a wide variety of problems. In this method of weighted residual, the weighing functions are chosen to be identical to the trial functions. The usual way of obtaining the solution of a system for an extreme condition is to solve the differential equation with corresponding boundary conditions and since such a solution

is not possible except in very simple cases, we resort to approximate methods such as the Galerkin's method for approximate solutions to the problem (Njoku, 2013).

The functional presented by Galerkin is as given in Equation (2.14) by Ezech et al. (2013)

$$\iint_A [L(W_N) - F] f_i(x, y) dx dy = 0 \quad (2.14)$$

Where L indicates either a linear or nonlinear differential operator, W is the displacement function and $f_i(x, y)$ is the linearly independent co-ordinate functions called trial functions, where i tends from 1 to n, which depends on the mode of vibration. $L(W_N)$ is equal to $D \nabla_w^4$ and F is the

inertia force. $D = \frac{Eh^3}{12(1-\nu^2)}$ and ∇^4 is the biharmonic differential operator and w is displacement.

This method is an energy approach that sums up all the work (strain energy and potential energy or external work) on the continuum to be equal to total potential energy. It seeks to minimize the total potential energy functional in order to get the stability matrix. The accuracy of the solution is dependent on the accuracy of the approximate deflection function called the shape function (Ezech et al., 2014).

On the vibration of a plate, Galerkin formulated an equation for free vibration of thin plate in non-tribal form as expressed in Equation (2.15)

$$\iint_A [D \nabla^4 W - m w^2] W dx dy = G . \quad (2.15)$$

(Ventsel & Krauthammer, 2001)

w is evaluated and the fundamental frequency of the vibrating plate is obtained. The use of this method can be recommended when computers are not available, but its computation can be lengthy, and the accuracy of this method depends on the selected shape function.

2.3.3 Numerical Method of Plate Analysis

According to Oxford dictionary by John (2004), the numerical method is a complete and unambiguous set of procedures for a problem, together with computable error estimates.

Simple cases in plate structures with a limited set of boundary conditions and geometries results from the analysis are readily available. With the advent of fast computers and various efficient numerical methods, big amount of research done for getting better accuracy in the results evolve. Numerical methods offer a reasonable and accepted solution but with complex shapes of plate sometimes lead to inaccuracies and more computing time (Chakraverty, 2009).

Different numerical methods as it applied to plate are discussed below.

2.3.3.1 Finite Difference Method

Finite difference method is a numerical method based on the mathematical discretization of differential equations, which the finite difference equations are translated into algebraic. By using this method, a continuous process is studied in a finite number of sufficiently small intervals. In these small intervals, the function is approximated by approximate expressions. In each elementary interval is the integration, with the results of integration in the previous interval, are taken as initial for the next time interval (Dolicanin, Nikolic and Dolicanin, 2010).

The function $f(x)$ is continuous in the interval divided into equal divisions ($\Delta x = h$). The first derivative of 'f' at x is as shown in Equation (2.16) and the first finite difference is as shown in Equation (2.17) respectively.

$$f' = \Delta f |_x / 2h \quad (2.16)$$

$$\Delta F |_x = f(x+h) - f(x-h) / 2. \text{ (First finite difference).} \quad (2.17)$$

In applying ordinary finite difference method, the derivatives in the governing plate differential equation are replaced by different quantities at some selected point and plate differential equation has been transformed into a set of algebraic equations. The convergence to the exact solution of plate bending problem with classical finite difference method is slow due to the errors used in deriving finite difference expressions, the approximation of boundary conditions and use of coarse load averaging rules (Ali, 2011).

By the application of the boundary conditions e.g. (SSSS and CCCC plates), ordinary finite difference method can be used to obtain solutions for the pure bending analysis of thin rectangular flat plates carrying uniformly distributed load. Using the governing differential equation for deflection of thin plates which is shown in Equation (2.18).

$$\nabla^4 W_{(x,y)} = \frac{\partial^4 W}{\partial x^4} + 2 \frac{\partial^4 W}{\partial x^2 \partial y^2} + \frac{\partial^4 W}{\partial y^4} = \frac{P_o(x,y)}{D} \quad (2.18)$$

Where W = deflection, x, and y are plate coordinates

P_o = applied uniformly distributed static load

D = flexural rigidity of the plate

$$D = \frac{Eh^3}{12(1-\nu^2)} \quad (2.18a)$$

E, ν , and h are Young's modulus, Poisson's ratio, and the thickness of the plate respectively; a pattern for the derived coefficients is formed. The pattern for the derived coefficients and pattern for the governing equation are applied at each of the internal nodes with the appropriate boundary conditions to generate the required matrix equation which the deflections and moments are determined (Ezeh, Ibearugbulem and Onyechere, 2013).

According to Ezeh, Ibearugbulem and Onyechere (2013), an ordinary finite difference was used to obtain solutions for the free vibration of thin rectangular flat plates carrying uniformly distribution load. A free harmonic vibration of a thin plate with constant thickness h, governed by differential equation was given by (Jiu, Liu and Chen., 2007) as expressed in Equation (2.19).

$$\nabla^4 W_{(x,y)} - \frac{\omega^2 \rho h}{D} W(x,y) = 0 \quad (2.19)$$

Where $\nabla^4 = \nabla^2 \nabla^2$ is called the biharmonic differential operator.

ω is the natural circular frequency for the vibrating plate given in rad/sec.

ρ is mass density of the plate.

h is the thickness of the plate.

Considering the boundary conditions and using the central difference, the ordinary finite difference for the differentials is formulated. The ordinary finite-difference and patterns derived were used to replace the derivatives in the governing equation and applied at each of the internal nodes noting the boundary conditions to generate the required eigenvalue equation.

The finite difference method is relatively easy to program, fast enough to analyze and also seems to be more convenient for uniform structures such as plate system. The main drawback of finite

difference method is that it is not suitable for problems with awkward and irregular geometries (Wu, Shu, Xiang and Wang., 2010).

Furthermore, since it is difficult to vary the size of the different cell in particular regions, it is not suitable for problems with rapidly changing variables such as stress concentration problems.

2.3.3.2 Boundary Element Method

Boundary element method is a numerical/computational method that has emerged as a powerful alternative to finite elements particularly in cases where better accuracy is required due to problems such as stress concentration or where domain extends to infinity. The most important feature of boundary elements is that the methodology of formulating boundary values problems as boundary integral equations describes problems only by equations with known and unknown boundary states. Hence, it only requires discretization of the surface rather than the volume. The necessary discretization effort is mostly much smaller and meshes can easily be generated and design changes do not require a complete remeshing.

The Boundary Element Method is especially advantageous in the case of problems with infinite or semi-infinite domains, although only the finite surface of the infinite domain has to be discretized, the solution at any arbitrary point of the domain can be found after determining the unknown boundary data (Heinz, 2010).

According to Michal and Tomasz (2007), the most popular approach to Boundary Element method was proposed by Benzine in which, the forces at the internal supports are treated as unknown variables. The second approach was proposed by Rashed in the application of a

coupled Boundary Element Method – flexibility force method in bending analysis of plates with internal supports.

Advantages of the Boundary Element Method

The advantages of Boundary Element Method are as follows:

- i. Less data preparation time.
- ii. Stresses are accurate because no further approximation is imposed on the solution at interior points.
- iii. Less computer time and storage.
- iv. Less unwanted information.

Disadvantages of the Boundary Element Method

The disadvantages of Boundary Element Method are listed below:

- i. The mathematics used in Boundary Element formulation may seem unfamiliar to engineers.
- ii. In non-linear material problems, the interior must be modeled.
- iii. The solution matrix resulting from the Boundary Element formulation is unsymmetric and fully populated with no zero coefficients whereas finite element solution matrices are usually much larger but sparsely populated.
- iv. Poor for thin structures (shell) three-dimensional analyses.

2.4 Finite Element Analysis of Plates

2.4.1 Finite Element Method

The finite element method is a general discretization method for the solution of partial derivative differential equations, and it is at present the most common discretization method.

Its success is due to the possibility of using it for a wide variety of problems, but above all to the availability of computing machines of ever-increasing power. The method yield usually models with a large number of degrees of freedom. The ordinary differential equations obtained are easily implemented in general-purpose codes for digital computers.

The Finite Element Method (FEM) is based on the subdivision of the structure into finite elements where a continuum is divided into small regions called elements, interconnected at selected nodes. The deformation of each element is expressed by interpolating polynomials. The coefficient of this polynomials is defined in terms of the element nodal degree of freedom (DOF) that describe the displacements and slopes of selected nodes on the element (Nikolas, 2014).

FEM converts the set of governing differential equations for a problem into a set of an equation in the form expressed in Equation (2.20).

$$M \ddot{x} + C \dot{x} + Kx = F_{(t)} \quad (2.20)$$

Vector x represents nodal displacements, and the dot indicates the first time derivative. M , C & K are mass, damping and stiffness matrices, respectively. $F_{(t)}$ is the vector of active nodal forces.

Finite element models and MATLAB of complex mechanical systems can be constructed in many engineering programs.

It can be used to analyze systems by defining their systems and the inputs. These programs develop the mathematical model of the systems and performed their solutions. These mathematical models could be extracted from the finite element programs and then it can be used with other control programs such as MATLAB to solve closed-loop problems. By the MATLAB directly into the FE programs, the closed loop control problems with complex structures can be analyzed more easily. The FE models of smart structures such as beam and rectangular plate are constructed by MATLAB. Closed loop- FE simulations of these smart structures are studied with the integrated procedure to reduce vibration amplitudes (Kapil, 2012).

Steps Used in the Application of FEM (Szilard, 2004).

The following steps were used in Finite Element Method for analyzing structures:

- (i) Discretization of the continuum,
- (ii) Selection of suitable shape functions,
- (iii) Element formulation.
- (iv) Treatment of the boundary conditions and loads.
- (v) Assembly of the discretized system.
- (vi) A solution of the resulting system of equations.
- (vii) Computation of stress resultants

2.4.2 Finite Element Method Historical Review

According to Nikolaos (2014), the principles of the Finite Element Method were proposed by Ritz in 1909. Galerkin, a Russian mathematician further developed this method. As a result of the absence of computers, the development of Finite Element Method was delayed.

Another German Mathematician Courant in 1943 solved the torsion problem using an assemblage of triangular elements and the principle of Minimum Potential Energy, which he named Rayleigh-Ritz Method. In 1955, Greek J.H. Argyris wrote a paper on ‘Energy theorems and structural analysis’, introducing the principles of finite elements. The stiffness matrices for beams and other elements were derived by Turner, Clough Martin, and Topp in 1956. In 1967, the first book on Finite Elements was written by Zienkiewicz and Chung.

2.4.3 Recent Analysis of Plates Using Finite Element Method

Ramu and Mohanty (2012) used the Kirchhoff plate theory to study the free Vibration Analysis of Rectangular plate using Finite Element Method. The matrixes formed are used to calculate the natural frequencies of a rectangular plate by solving the eigenvalue problem. The calculated natural frequencies using FEM were compared with those obtained from exact Levy type solution which gave a close result.

Ugurlu, Kutlu, Ergin and Omurtag (2008) investigated the effects of elastic foundation and fluid on the dynamic response characteristic of rectangular Kirchhoff plates using a mixed-type finite element formulation.

Phadikar and Pradhan (2010) presented a finite element on Eringen non-classical elasticity theory. In 2012, Pradhan developed a finite element formulation for nonlocal elastic Euler-Bernoulli beam theory and Timoshenko beam theory.

Nguyen-Xuan, Rabezuk, Nguyen-Thanh, Nguyen-Thoi and Bordas (2010) presented a formulation of the Node-based smoothed finite element method (NS-FEM) for Reissner-Mindlin plates using 3-node triangular elements. The discrete weak form of the NS-FEM is obtained

based on the strain smoothing technique over smoothing domains associated with the nodes of elements.

Lin and Nien (2007) discussed the adaptive modeling and shape control of laminate plates with Piezoelectric Actuators. A finite element formulation was developed for dynamics and static response of laminated plates. Paquin and St. Amant (2010) studied the effect of a variable thickness beam harvester on its electromagnetic performance. Rayleigh-Ritz approximations were used to develop a semi-analytical mechanical model. Finite element technique was used to validate the model and numerical simulations were then performed to find the optimum for a given maximal strain across the piezoelectric elements for different beam slope angles.

Gergely, Laszlo, Gyorgy, and Laszlo, D. (2012) analyzed the laminated structural glass plates with Polyvinyl butyral (PVB) interlayer using finite element method. Their present study aimed at developing a numerical model verified by experimental results and to examine the elastic mechanical behavior of structural glass laminated with PVB film.

Limitation of FEM

The limitations of Finite Element Method are:

- i. It requires the use of powerful computers of considerable speed and storage capacity.
- ii. It is difficult to ascertain the accuracy of numerical results when large structural systems are analyzed.
- iii. The method is poorly adapted to a solution of singular problems e.g. plates and shells with cracks, corner points, discontinuity internal actions and problems for unbounded domains.

According to Neffati and Abobaker (2012). The governing equation that describes the flexural vibration of thin plates subjected to transverse loading, based on classical plate theory, is as expressed in Equation (2.21).

$$p(x, y)h \frac{\partial^2 w(x, y, t)}{\partial t^2} + D \left[\frac{\partial^4 w(x, y, t)}{\partial x^4} + 2 \frac{\partial^4 w(x, y, t)}{\partial x^2 \partial y^2} + \frac{\partial^4 w(x, y, t)}{\partial y^4} \right] = p_z(x, y, t). \quad (2.21)$$

Where $w(x, y, t)$, is the out of plane motion in positive z - direction, p_z is the exciting load per unit area, E , h , ν and $p(x, y)$, are the modulus of elasticity, plate thickness, Poisson's ratio, and density respectively. D is the flexural rigidity which is shown in Equation (2.21a).

$$D = \frac{Eh^3}{12(1-\nu^2)} \quad (2.21a)$$

The governing equation of a vibrating thin plate above is a fourth-order partial differential equation which requires the continuity of both deflection and slope with respect to both x - and y - directions, namely w , Θ and ϕ . At least three degrees of freedom are required at each node of the selected element to get a unique solution. The finite element equation can now be expressed in Equation (2.22) as:

$$R(x, y, t) = p(x, y)h \frac{\partial^2 w_a}{\partial t^2} + D \left[\frac{\partial^4 w_a}{\partial x^4} + 2 \frac{\partial^4 w_a}{\partial x^2 \partial y^2} + \frac{\partial^4 w_a}{\partial y^4} \right] - p_z(x, y, t) \neq 0 \quad (2.22)$$

2.5 MATLAB as Software for Analysis

MATLAB is interactive software which has been used recently in various areas of Engineering. One attractive aspect of MATLAB is that is relatively easy to learn. It is written on an intuitive basis and it does not require in-depth knowledge of operational principle of computer programming like compiling and linking most of the other programming languages.

The power of MATLAB is represented by the length and simplicity of the code. For example, one page of MATLAB code may be equivalent to many pages of other computer source codes. The numerical calculation in MATLAB uses collections of well written scientific/mathematical subroutines such as LINPACK and EISPACK. MATLAB provides a Graphical user interface (GUI) as well as three-dimensional graphical animation (Young, 1997).

It is a software tool with powerful computational and graphics presentation capabilities widely used in education and research. It is valuable for teaching structural analysis; in particular, modern matrix procedures like the direct stiffness and finite element methods. The popularity of MATLAB in teaching analysis in structural engineering is due to its ease of use through a variety of built-in functions well suited for structural analysis. The main objective of working with MATLAB is to appreciate the principles and concepts of structural analysis and clarity of their formulation (Ansgar, 2009).

The great advantage of MATLAB as an interactive system is that programs can be tested and debugged quickly, allowing the user to concentrate more on the principles behind the program and less on programming itself. Since there is no need to compile, link and execute after each correction, MATLAB programs can be developed in much shorter time than equivalent FORTRAN or C programs. Other advantages of MATLAB include:

- i. There is extensive graphics support that follows the results of computations to be plotted with a few statements.
- ii. All numerical objects are treated as double-precision arrays, thus there is no need to declare types and carry out type conversions (Jaan, 2005).

CHAPTER THREE
METHODOLOGY

3.1 Formulation of the General Flexural Element Stiffness Matrix of Thin Rectangular Plate.

In the formulation of the general element flexural stiffness matrix of a rectangular thin plate, the following general plate assumptions governed by the Kirchoff hypotheses were made.

- i. The deflection (w) of the mid-surface is small when compared with the thickness of the plate.
- ii. The mid-surface of the plate remains unrestrained before, during or after bending.
- iii. A cross section that is initially straight before bending shall remain straight after bending.
- iv. The stress normal to xy plane σ_z is assumed to be zero.

Based on the above assumptions the Equation of plate that delineates the flexural vibration of thin isotropic rectangular plate and was selected as expressed in Equation (3.1).

$$D \left(\frac{\partial^4 w_{(x,y)}}{\partial x^4} + \frac{2\partial^4 w_{(x,y)}}{\partial x^2 \partial y^2} + \frac{\partial^4 w_{(x,y)}}{\partial y^4} \right) = \bar{m} \lambda^2 w_{(x,y)} \quad (3.1)$$

Where w is the displacement in positive Z - direction

\bar{m} is the mass per unit area of the plate

λ is the fundamental natural frequency of the plate

D is the flexural rigidity expressed in Equation (3.1a) as

$$D = \frac{Eh^3}{12(1-\nu^2)} \quad (3.1a)$$

E = Modulus of elasticity

h = Plate thickness

ν = poison's ratio

For a plane continuum the governing differential equation for total strain energy for plate in vibration is given as;

$$\Pi = \frac{D}{2} \int_0^a \int_0^b \left[\left[\frac{\partial^2 w}{\partial x^2} \right]^2 + 2 \left[\frac{\partial^2 w}{\partial x \partial y} \right]^2 + \left[\frac{\partial^2 w}{\partial y^2} \right]^2 \right] \partial x \partial y - \frac{m \lambda^2}{2} \int \int w^2 \partial x \partial y \quad (3.2)$$

Let's define 'w' as the deflection equation.

$$[w] = [N] [\psi] \quad (3.3)$$

Where, w = deflection of plate

$[N]$ = deflection shape function

$[\psi]$ = coefficients of the displacements (deflection and rotation)

Let $[w_i]$ = deflection and rotation at the nodes of the plate.

$[N_i]$ = nodal value displacement profile

$$[w_i] = [N_i] [\psi] \quad (3.4)$$

Where $[N_i]$ is a square matrix

Re-arranging Equation (3.4) and making $[\psi]$ subject of the Equation, this yield

Equation (3.5).

$$[\psi] = [N_i^{-1}]^T [w_i] \quad (3.5)$$

Substituting Equation (3.5) into (3.3) yields Equation (3.6)

$$w = [N] [N_i^{-1}]^T [w_i] \quad (3.6)$$

By transforming Equation (3.2);

$$\text{Let } \frac{\partial^2 w}{\partial x^2} = w''_x, \frac{\partial^2 w}{\partial x \partial y} = w''_{xy} \text{ and } \frac{\partial^2 w}{\partial y^2} = w''_y \quad (3.7)$$

Substituting Equation (3.7) into Equation (3.2), gives Equation (3.8)

$$\Pi = \frac{D}{2} \int_0^a \int_0^b \left[[w_x]''^2 + 2[w_{xy}]''^2 + [w_y]''^2 \right] \partial x \partial y - \frac{m\lambda^2}{2} \int \int w^2 \partial x \partial y \quad (3.8)$$

Equation (3.8) is divided into two sections ‘‘A’’ and ‘‘B’’ such that section A comprises of Equation (3.8a) and section B is made up Equation (3.8b) as expressed below.

$$\text{Section A} = \frac{D}{2} \int_0^a \int_0^b \left[[w_x]''^2 + 2[w_{xy}]''^2 + [w_y]''^2 \right] \partial x \partial y \quad (3.8a)$$

$$\text{Section B} = -\frac{m\lambda^2}{2} \int \int w^2 \partial x \partial y \quad (3.8b)$$

Substituting Equation (3.6) into section A and B of Equation (3.8), yields Equation (3.9)

$$\begin{aligned} \Pi = & \frac{D}{2} \left[\int_0^a \int_0^b \left[[N][N_i^{-1}]^T [w_i] \right]_x'' \right]^2 dx dy + D \left[\int_0^a \int_0^b \left[[N][N_i^{-1}]^T [w_i] \right]_{xy}'' \right]^2 dx dy \\ & + \frac{D}{2} \left[\int_0^a \int_0^b \left[[N][N_i^{-1}]^T [w_i] \right]_y'' \right]^2 dx dy - \frac{m\lambda^2}{2} \left[\int \int \left[[N][N_i^{-1}]^T [w_i] \right]^2 dx dy \right] \end{aligned} \quad (3.9)$$

By expanding section, A of Equation (3.9), Equations (3.9i to 3.9vi) were obtained;

$$\frac{D}{2} \int_0^a \int_0^b \left[\left[[N][N_i^{-1}]^T [w_i] \right]_x'' \right]^2 dx dy = \frac{D}{2} \int_0^a \int_0^b \left[\left[[N][N_i^{-1}]^T [w_i] \right]_x'' \right]^T \left[\left[[N][N_i^{-1}]^T [w_i] \right]_x'' \right] dx dy \quad (3.9i)$$

$$\frac{D}{2} \int_0^a \int_0^b \left[\left[[N][N_i^{-1}]^T [w_i] \right]_x'' \right]^2 dx dy = \frac{D}{2} [N_i^{-1}]^T [w_i] \int_0^a \int_0^b \left[[N]^T [N] \right]_x'' dx dy \cdot [N_i^{-1}] [w_i] \quad (3.9ii)$$

Also

$$D \int_0^a \int_0^b \left[\left[[N][N_i^{-1}]^T [w_i] \right]_{xy}'' \right]^2 dx dy = D \int_0^a \int_0^b \left[\left[[N][N_i^{-1}]^T [w_i] \right]_{xy}'' \right]^T \left[\left[[N][N_i^{-1}]^T [w_i] \right]_{xy}'' \right] dx dy \quad (3.9iii)$$

$$D \int_0^a \int_0^b \left[\left[[N] [N_i^{-1}]^T [w_i] \right]_{xy} \right]^2 dx dy = D [N_i^{-1}]^T [w_i]^T \int_0^a \int_0^b \left[[N]^T [N] \right]_x dx dy \cdot [N_i^{-1}] [w_i] \quad (3.9iv)$$

And

$$\frac{D}{2} \int_0^a \int_0^b \left[\left[[N] [N_i^{-1}]^T [w_i] \right]_y \right]^2 dx dy = \frac{D}{2} \int_0^a \int_0^b \left[\left[[N] [N_i^{-1}]^T [w_i] \right]_y \right]^T \left[\left[[N] [N_i^{-1}]^T [w_i] \right]_y \right] dx dy \quad (3.9v)$$

$$\frac{D}{2} \int_0^a \int_0^b \left[\left[[N] [N_i^{-1}]^T [w_i] \right]_y \right]^2 dx dy = \frac{D}{2} [N_i^{-1}]^T [w_i]^T \int_0^a \int_0^b \left[[N]^T [N] \right]_y dx dy \cdot [N_i^{-1}] [w_i] \quad (3.9vi)$$

Also by expanding section B of Equation (3.9), Equations (3.9vii to 3.9viii) were obtained;

$$\frac{m\lambda^2}{2} \int \int \left[\left[[N] [N_i^{-1}]^T [w_i] \right] \right]^2 dx dy = \frac{m\lambda^2}{2} \int \int \left[[N] [N_i^{-1}]^T [w_i] \right]^T \left[[N] [N_i^{-1}] [w_i] \right] dx dy \quad (3.9vii)$$

$$\frac{m\lambda^2}{2} \int \int \left[\left[[N] [N_i^{-1}]^T [w_i] \right] \right]^2 dx dy = \frac{m\lambda^2}{2} [N_i^{-1}]^T [w_i]^T \int \int \left[[N]^T [N] \right]_y dx dy \cdot [N_i^{-1}] [w_i] \quad (3.9viii)$$

Now, bringing out $[N_i^{-1}]^T$ and $[w_i]^T$ which are constants and minimizing Equations

(3.9ii),(3.9iv),(3.9vi) and (3.9viii) by differentiating with respect to $[w_i]$ yields Equation (3.10);

$$\begin{aligned} \Pi = & [N_i^{-1}]^T \frac{D}{2} \int_0^a \int_0^b \left[[N]^T [N] \right]_x dx dy \cdot [w_i] [N_i^{-1}] \\ & + [N_i^{-1}]^T D \int_0^a \int_0^b \left[[N]^T [N] \right]_{xy} dx dy \cdot [w_i] [N_i^{-1}] \\ & + [N_i^{-1}]^T \frac{D}{2} \int_0^a \int_0^b \left[[N]^T [N] \right]_y dx dy \cdot [w_i] [N_i^{-1}] \\ & - [N_i^{-1}]^T \frac{m\lambda^2}{2} \int \int \left[[N]^T [N] \right] dx dy \cdot [w_i] [N_i^{-1}] \end{aligned} \quad (3.10)$$

Equation (3.10) is called the general flexural element stiffness matrix of a thin rectangular plate.

By introducing non-dimensional parameter R and Q

$$\text{Let } R = \frac{x}{a} \text{ and } Q = \frac{y}{b} \quad (3.11)$$

$$x = Ra \text{ and } y = Qb \quad (3.11a)$$

Note $dx = adR$, $dy = bdQ$

$$\text{Let } \alpha = \frac{b}{a} \text{ then } b = \alpha a \quad (3.12)$$

Recall Equation (3.7) and Equation (3.9).

By solving Equation (3.9) and substituting Equations (3.11a) yields Equation (3.13)

$$\begin{aligned} \Pi = & \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{a^2 dR^2} \right]^2 \right] adR . bdQ + D \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{adR . bdQ} \right]^2 \right] adR . bdQ \\ & + \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{b^2 dQ^2} \right]^2 \right] adR . bdQ - \frac{m\lambda^2}{2} \left[\int \int [N] [N_i^{-1}]^T [w_i]^2 adR . bdQ \right] \end{aligned} \quad (3.13)$$

Bringing out a and b terms in the bracket of Equation (3.13) yield Equation (3.13a)

$$\begin{aligned} \Pi = & \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dR^2} \right]^2 \right] \frac{ab . dRdQ}{a^4} + D \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dRdQ} \right]^2 \right] \frac{ab . dRdQ}{a^2 b^2} \\ & + \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dQ^2} \right]^2 \right] \frac{ab . dRdQ}{b^4} - \frac{m\lambda^2}{2} \left[\int \int [N] [N_i^{-1}]^T [w_i]^2 ab . dRdQ \right] \end{aligned} \quad (3.13a)$$

Multiplying Equation (3.13a) with 1/ab yields Equation (3.13b)

$$\begin{aligned}
\Pi = & \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dR^2} \right]^2 \right] \frac{dRdQ}{a^4} + D \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dRdQ} \right]^2 \right] \frac{dRdQ}{a^2 b^2} \\
& + \frac{D}{2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dQ^2} \right]^2 \right] \frac{dRdQ}{b^4} - \frac{m\lambda^2}{2} \left[\int \int [N] [N_i^{-1}]^T [w_i]^2 dRdQ \right] \quad (3.13b)
\end{aligned}$$

Substituting Equation (3.12) into Equation (3.13b) yield Equation (3.13c)

$$\begin{aligned}
\Pi = & \frac{D}{2a^4} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dR^2} \right]^2 \right] dRdQ + \frac{D}{a^4 \alpha^2} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dRdQ} \right]^2 \right] dRdQ \\
& + \frac{D}{2a^4 \alpha^4} \left[\int_0^a \int_0^b \left[\frac{d^2 [N] [N^{-1}]^T [w_i]}{dQ^2} \right]^2 \right] dRdQ - \frac{m\lambda^2}{2} \left[\int \int [N] [N_i^{-1}]^T [w_i]^2 dRdQ \right] \quad (3.13c)
\end{aligned}$$

Expanding Equation (3.13c) using the same process for Equation (3.9i) to Equation (3.9viii)

yields Equation (3.13d)

$$\begin{aligned}
\Pi = & [N_i^{-1}]^T \frac{D}{2a^4} \int_0^1 \int_0^1 [N]^T [N]_R dRdQ. [w_i] [N_i^{-1}] \\
& + [N_i^{-1}]^T \frac{D}{a^4 \alpha^2} \int_0^1 \int_0^1 [N]^T [N]_{RQ} dRdQ. [w_i] [N_i^{-1}] \\
& + [N_i^{-1}]^T \frac{D}{2a^4 \alpha^4} \int_0^1 \int_0^1 [N]^T [N]_Q dRdQ. [w_i] [N_i^{-1}] \\
& - [N_i^{-1}]^T \frac{m\lambda^2}{2} \int \int [N]^T [N] dRdQ. [w_i] [N_i^{-1}] \quad (3.13d)
\end{aligned}$$

Equation (3.13d) is called the general flexural element stiffness matrix of thin rectangular plate for non- dimensional coordinates.

Figure 3.1 shows the Pascal triangular shape function of a rectangular plate element in terms of non-dimensional parameter.

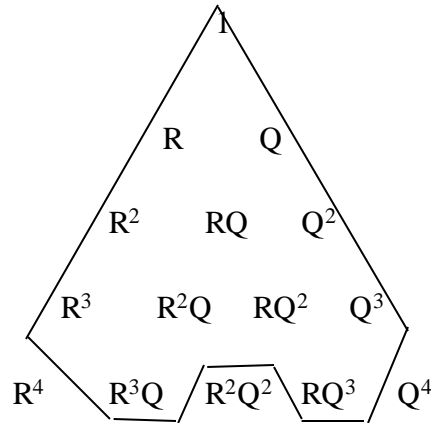


Figure 3.1 The enclosed section of the pascal triangle is the area of concentration for the deflection.

It should be noted that the shape function in terms of non dimensional parameter is expressed in a Polynomial fuction and the parameters expressed in the Pascal triangle are the coefficients of the Polynomial Equation of deflection.

The figure 3.2 shows a sample of finite element mesh of a plate simply supported on opposite short edge. The plate is descritized into 49 elements and has 48 nodes.

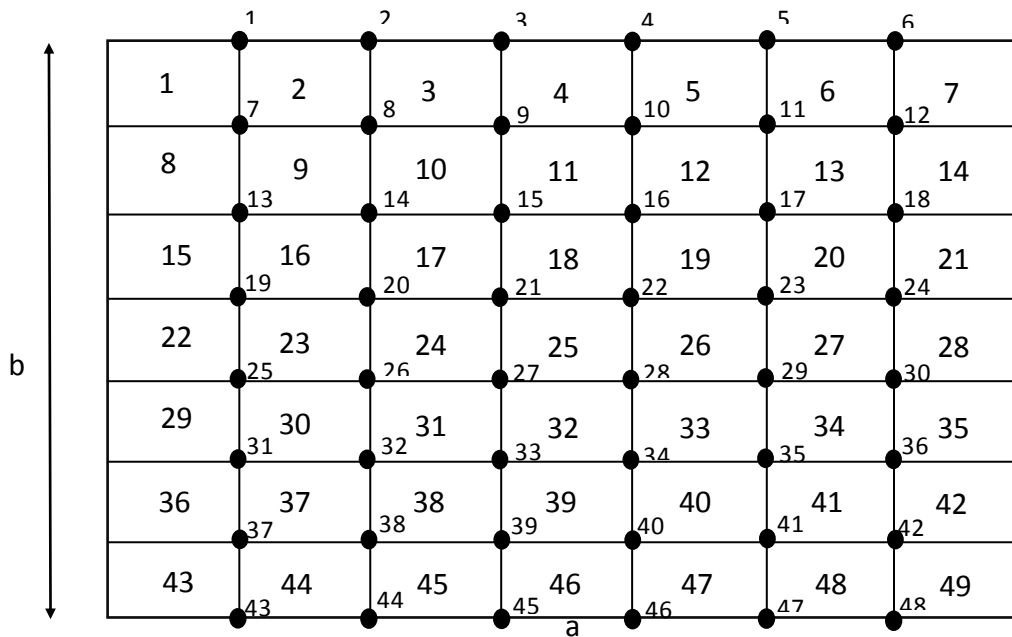


Figure 3.2 A finite element mesh of a plate simply supported on opposite short edge

The figure 3.3 represents a plate model that shows three deformable quantities acting on it. The plate contains in it 12 degrees of freedom which must be satisfied under the application of flexural load. Each node on the plate can be numbered in anti-clockwise setting and each node bears 3 degrees of freedom i.e. deflection, rotation about the vertical direction and rotation about the horizontal direction. The solution of these displacements can be obtained by the use of the general flexural element stiffness matrix of a thin rectangular plate derived in equation (3.13d).

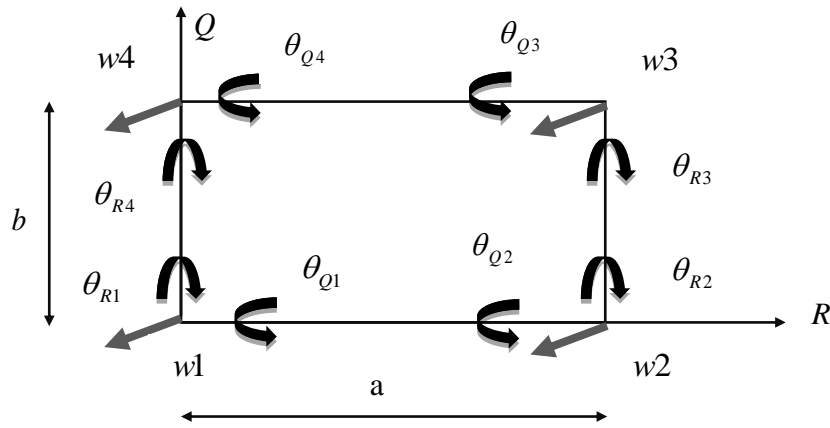


Figure 3.3. A thin plate with 12 degrees of freedom

Where:

w = deflection of the plate at the node.

θ_R = Rotation at the node about R axis

θ_Q = Rotation at the node about Q axis

a and b are the dimension of the plate along R and Q axis respectively.

In the determination of the general stiffness matrix of thin plate under free vibration, R and Q are the non-dimensional parameter and N is the deflection shape function of which its values are deduced from the Pascal triangle.

$$N = [1 \quad R \quad Q \quad R^2 \quad RQ \quad Q^2 \quad R^3 \quad R^2Q \quad RQ^2 \quad Q^3 \quad R^3Q \quad RQ^3] \quad (3.14)$$

Recall that a plate with 12 degrees of freedom will result to a 12 x 12 matrix required for the solution. That means we still have 12 deformable terms to be used.

Bringing N , θR and θQ together we have;

$$N_i = \begin{bmatrix} 1 & R & Q & R^2 & RQ & Q^2 & R^3 & R^2Q & RQ^2 & Q^3 & R^3Q & RQ^3 \\ 0 & 1 & 0 & 2R & Q & 0 & 3R^2 & 2RQ & Q^2 & 0 & 3R^2Q & Q^3 \\ 0 & 0 & 1 & 0 & R & 2Q & 0 & R^2 & 2RQ & 3Q^2 & R^3 & 3RQ^2 \end{bmatrix} \begin{matrix} N \\ \theta R \\ \theta Q \end{matrix} \quad (3.14a)$$

Note: R^2 denotes R^2 and Q^2 denote Q^2 .

The values of θR and θQ are obtained by differentiating N with respect to R and Q respectively.

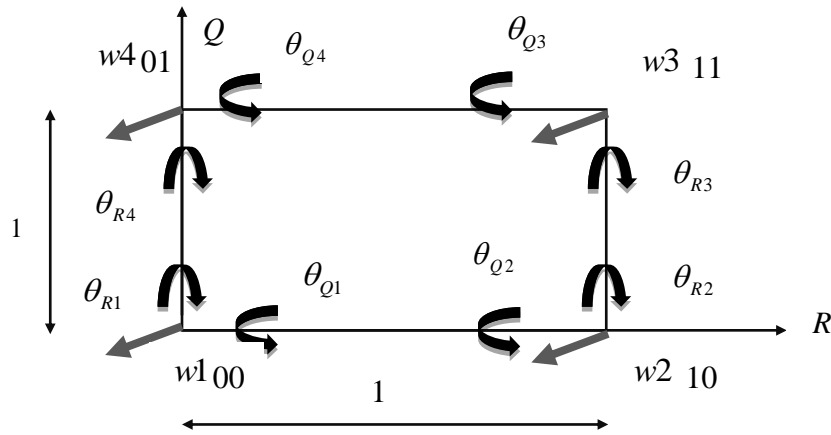


Figure 3.4. A thin plate with 12 degrees of freedom and nodal coordinates

Figure 3.4 is a plate continuum with nodal values. The plate dimensions a and b are assumed to be 1 which led to the nodal values. These values are substituted in Equation (3.14a) after differentiating the deflection Equation (N) at all the nodes with respect to R and Q respectively.

At node 1 we have three degrees of freedom which is w_1 , θR_1 , θQ_1 and coordinate at this node is $(0,0)$. When this coordinate value is substituted in Equation (3.14a) it yields Equation (3.14b).

$$N_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.14b)$$

At node 2 the three degrees of freedom are $w_2, \theta R_2, \theta Q_2$ and the nodal values are (1,0).

Substituting these nodal values in Equation (3.14a) yields Equation (3.14c).

$$N_i = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (3.14c)$$

At node 3 the three degrees of freedom are $w_3, \theta R_3, \theta Q_3$ and the nodal values are (1,1). These nodal values are substituted in Equation (3.14a) which yields Equation (3.14d).

$$N_i = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 2 & 1 & 0 & 3 & 2 & 1 & 0 & 3 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 3 & 1 & 3 \end{bmatrix} \quad (3.14d)$$

At node 4 the three degrees of freedom are $w_4, \theta R_4, \theta Q_4$ and the coordinate at the node is (0,1).

Equation (3.14e) is obtained by substituting these values in Equation (3.14a).

$$N_i = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \end{bmatrix} \quad (3.14e)$$

Combining Equations (3.14b) to (3.14e) yields a 12 by 12 square matrix shown in Equation

(3.15). The application of boundary conditions is done in chapter 4.

Nodal Values
of R and Q

$$[N_i] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 2 & 1 & 0 & 3 & 2 & 1 & 0 & 3 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 3 & 1 & 3 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \end{bmatrix} \quad (3.15)$$

0,0
1,0
1,1
0,1

The inverse of matrix in Equation (3.15) is as shown in Equation (3.16);

$$[N_i]^{-1} = \begin{matrix} w1 \\ \theta R1 \\ \theta Q1 \\ w2 \\ \theta R2 \\ \theta Q2 \\ w3 \\ \theta R3 \\ \theta Q3 \\ w4 \\ \theta R4 \\ \theta Q4 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -2 & 0 & 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & 0 \\ -3 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & -1 \\ 2 & 1 & 0 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & -3 & 1 & 0 & 3 & -1 & 0 & -3 & -2 & 0 \\ 3 & 0 & 2 & -3 & 0 & -2 & 3 & 0 & -1 & -3 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1 \\ -2 & -1 & 0 & 2 & -1 & 0 & -2 & 1 & 0 & 2 & 1 & 0 \\ -2 & 0 & -1 & 2 & 0 & 1 & -2 & 0 & 1 & 2 & 0 & -1 \end{bmatrix} \quad (3.16)$$

Equation (3.16) was transposed as shown in Equation (3.17);

$$[N_i]^{-1T} = \begin{matrix} w1 \\ \theta R1 \\ \theta Q1 \\ w2 \\ \theta R2 \\ \theta Q2 \\ w3 \\ \theta R3 \\ \theta Q3 \\ w4 \\ \theta R4 \\ \theta Q4 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & -3 & -1 & -3 & 2 & 3 & 3 & 2 & -2 & -2 \\ 0 & 1 & 0 & -2 & -1 & 0 & 1 & 2 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -2 & 0 & 0 & 2 & 1 & 0 & -1 \\ 0 & 0 & 0 & 3 & 1 & 0 & -2 & -3 & -3 & 0 & 2 & 2 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 3 & 3 & 0 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 3 & 0 & -3 & -3 & -2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & -1 \end{bmatrix} \quad (3.17)$$

The deflection shape function is differentiated with respect to R and Q by considering three cases.

Case 1: For R direction;

$$[N] = [1 \quad R \quad Q \quad R^2 \quad RQ \quad Q^2 \quad R^3 \quad R^2Q \quad RQ^2 \quad Q^3 \quad R^3Q \quad RQ^3] \quad (3.18)$$

$$[N]_R'' = \left[\frac{\partial^2 N}{\partial R^2} \right] = [0 \quad 0 \quad 0 \quad 2 \quad 0 \quad 0 \quad 6R \quad 2Q \quad 0 \quad 0 \quad 6RQ \quad 0] \quad (3.19)$$

Transposing Equation (3.19) yields Equation (3.20);

$$[N]_R'^T = \left[\frac{\partial^2 N}{\partial R^2} \right]^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 6R \\ 2Q \\ 0 \\ 0 \\ 6RQ \\ 0 \end{bmatrix} \quad (3.20)$$

The matrix multiplication of Equation (3.19) and (3.20) gives Equation (3.21);

$$[N]_R'^T \cdot [N]_R'' = \left[\frac{\partial^2 N}{\partial R^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2} \right] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 12R & 4Q & 0 & 0 & 12RQ & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12R & 0 & 0 & 36R^2 & 12RQ & 0 & 0 & 36R^2Q & 0 \\ 0 & 0 & 0 & 4Q & 0 & 0 & 12RQ & 4Q^2 & 0 & 0 & 12RQ^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12RQ & 0 & 0 & 36R^2Q & 12RQ^2 & 0 & 0 & 36R^2Q^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.21)$$

Integration of Equation (3.21) yields Equation (3.22);

$$\iint \left[\frac{\partial^2 N}{\partial R^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 6 & 2 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 12 & 3 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 3 & 1.33333 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 6 & 2 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.22)$$

Multiplying Equation (3.17) and Equation (3.22) gives Equation (3.23);

$$[N_i]^{-1T} \iint \left[\frac{\partial^2 N}{\partial R^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -0.33333 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0.33333 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0.66667 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -0.66667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.23)$$

The stiffness K_R for case one as shown in Equation (3.24) is obtained by multiplying together Equation (3.23) and Equation (3.16);

$$K_R = [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial R^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2} \right] dRdQ \cdot [N_i]^{-1} =$$

$$\begin{bmatrix} 4 & 2 & 0 & -4 & 2 & 0 & -2 & 1 & 0 & 2 & 1 & 0 \\ 2 & 1.33333 & 0 & -2 & 0.66667 & 0 & -1 & 0.33333 & 0 & 1 & 0.66667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -2 & 0 & 4 & -2 & 0 & 2 & -1 & 0 & -2 & -1 & 0 \\ 2 & 0.66667 & 0 & -2 & 1.33333 & 0 & -1 & 0.66667 & 0 & 1 & 0.33333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 2 & -1 & 0 & 4 & -2 & 0 & -4 & -2 & 0 \\ 1 & 0.33333 & 0 & -1 & 0.66667 & 0 & -2 & 1.33333 & 0 & 2 & 0.66667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & -2 & 1 & 0 & -4 & 2 & 0 & 4 & 2 & 0 \\ 1 & 0.66667 & 0 & -1 & 0.33333 & 0 & -2 & 0.66667 & 0 & 2 & 1.33333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.24)$$

Case 2: For RQ direction.

$$[N] = [1 \quad R \quad Q \quad R2 \quad RQ \quad Q2 \quad R3 \quad R2Q \quad RQ2 \quad Q3 \quad R3Q \quad RQ3]$$

$$[N]_{RQ}'' = \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right] = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 2R \quad 2Q \quad 0 \quad 3R2 \quad 3Q2] \quad (3.25)$$

Transposing Equation (3.25) results to Equation (3.26);

$$[N]_{RQ}''^T = \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right]^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 2R \\ 2Q \\ 0 \\ 3R2 \\ 3Q2 \end{bmatrix} \quad (3.26)$$

The matrix multiplication of Equation (3.25) and Equation (3.26) gives Equation (3.27);

$$[N]_{RQ}^{T} \cdot [N]_{RQ} = \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right] =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2R & 2Q & 0 & 3R2 & 3Q2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2R & 0 & 0 & 4R2 & 4RQ & 0 & 6R3 & 6RQ2 \\ 0 & 0 & 0 & 0 & 2Q & 0 & 0 & 4RQ & 4Q2 & 0 & 6R2Q & 6Q3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3R2 & 0 & 0 & 6R3 & 6R2Q & 0 & 9R4 & 9R2Q2 \\ 0 & 0 & 0 & 0 & 3Q2 & 0 & 0 & 6RQ2 & 6Q3 & 0 & 9R2Q2 & 9Q4 \end{bmatrix} \quad (3.27)$$

By integrating Equation (3.27), Equation (3.28) is obtained;

$$\iint \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.33333 & 1 & 0 & 1.5 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1.33333 & 0 & 1 & 1.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.5 & 1 & 0 & 1.8 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1.5 & 0 & 1 & 1.8 \end{bmatrix} \quad (3.28)$$

Multiplication of Equation (3.17) and Equation (3.28) yields Equation (3.29);

$$[N_i]^{-1T} \iint \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0.9 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & -0.9 & -0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1667 & 0 & 0 & -0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1667 & 0 & 0 & -0.2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0.9 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & -0.9 & -0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1667 & 0 & 0 & -0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1667 & 0 & 0 & 0.3 \end{bmatrix} \quad (3.29)$$

The stiffness K_{RQ} for case two as shown in Equation (3.30) is obtained by multiplying together 2, Equation (3.29) and (3.16);

$$K_{RQ} = 2 * [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2 \partial Q^2} \right] dRdQ \cdot [N_i]^{-1} =$$

$$\begin{bmatrix} 2.8 & 0.2 & 0.2 & -2.8 & 0.2 & -0.2 & 2.8 & -0.2 & -0.2 & -2.8 & -0.2 & 0.2 \\ 0.2 & 0.2667 & 0 & -0.2 & -0.067 & 0 & 0.2 & 0.0667 & 0 & -0.2 & -0.267 & 0 \\ 0.2 & 0 & 0.2667 & -0.2 & 0 & -0.267 & 0.2 & 0 & 0.0667 & -0.2 & 0 & -0.067 \\ -2.8 & -0.2 & -0.2 & 2.8 & 0.2 & 0.2 & -2.8 & 0.2 & 0.2 & 2.8 & 0.2 & -0.2 \\ 0.2 & -0.067 & 0 & -0.2 & 0.2667 & 0 & 0.2 & -0.267 & 0 & -0.2 & 0.0667 & 0 \\ -0.2 & 0 & -0.267 & 0.2 & 0 & 0.2667 & -0.2 & 0 & -0.067 & 0.2 & 0 & 0.0667 \\ 2.8 & 0.2 & 0.2 & -2.8 & 0.2 & -0.2 & 2.8 & -0.2 & -0.2 & -2.8 & -0.2 & 0.2 \\ -0.2 & 0.0667 & 0 & 0.2 & -0.267 & 0 & -0.2 & 0.2667 & 0 & 0.2 & -0.067 & 0 \\ -0.2 & 0 & 0.0667 & 0.2 & 0 & -0.067 & -0.2 & 0 & 0.2667 & 0.2 & 0 & -0.267 \\ -2.8 & -0.2 & -0.2 & 2.8 & -0.2 & 0.2 & -2.8 & 0.2 & 0.2 & 2.8 & 0.2 & -0.2 \\ -0.2 & -0.267 & 0 & 0.2 & 0.0667 & 0 & -0.2 & -0.067 & 0 & 0.2 & 0.2667 & 0 \\ 0.2 & 0 & -0.067 & -0.2 & 0 & 0.0667 & 0.2 & 0 & -0.267 & -0.2 & 0 & 0.2667 \end{bmatrix} \quad (3.30)$$

Case 3: For Q direction.

$$\begin{aligned}
[N] &= [1 \quad R \quad Q \quad R^2 \quad RQ \quad Q^2 \quad R^3 \quad R^2Q \quad RQ^2 \quad Q^3 \quad R^3Q \quad RQ^3] \\
[N]_Q'' &= \left[\frac{\partial^2 N}{\partial Q^2} \right] = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 2 \quad 0 \quad 0 \quad 2R \quad 6Q \quad 0 \quad 6RQ] \quad (3.31)
\end{aligned}$$

The transpose of Equation (3.31) yields Equation (3.32);

$$[N]_Q'^T = \left[\frac{\partial^2 N}{\partial Q^2} \right]^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 2R \\ 6Q \\ 0 \\ 6RQ \end{bmatrix} \quad (3.32)$$

The matrix multiplication of Equation (3.31) and Equation (3.32) gives Equation (3.33);

$$\begin{aligned}
[N]_Q'^T \cdot [N]_Q'' &= \left[\frac{\partial^2 N}{\partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial Q^2} \right] = \\
&\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4R & 12Q & 0 & 12RQ \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4R & 0 & 0 & 4R^2 & 12RQ & 0 & 12R^2Q \\ 0 & 0 & 0 & 0 & 0 & 12Q & 0 & 0 & 12RQ & 36Q^2 & 0 & 36RQ^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12RQ & 0 & 0 & 12R^2Q & 36RQ^2 & 0 & 36R^2Q^2 \end{bmatrix} \quad (3.33)
\end{aligned}$$

Integrating Equation (3.33) gives Equation (3.34);

$$\iint \left[\frac{\partial^2 N}{\partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial Q^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 2 & 6 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1.3333 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 3 & 12 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 6 & 0 & 4 \end{bmatrix} \quad (3.34)$$

Multiplication of Equation (3.17) and Equation (3.34) gives Equation (3.35);

$$[N_i]^{-1r} \iint \left[\frac{\partial^2 N}{\partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial Q^2} \right] dRdQ =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -0.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -0.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.6667 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.3333 & 3 & 0 & 1 \end{bmatrix} \quad (3.35)$$

The stiffness K_Q for case three i.e. Q axis, as shown in Equation (3.36) is obtained by

multiplying together Equation (3.35) and Equation (3.16)

$$K_Q = [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial Q^2} \right] dRdQ \cdot [N_i]^{-1} =$$

$$\begin{bmatrix} 4 & 0 & 2 & 2 & 0 & 1 & -2 & 0 & 1 & -4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1.3333 & 1 & 0 & 0.6667 & -1 & 0 & 0.3333 & -2 & 0 & 0.6667 \\ 2 & 0 & 1 & 4 & 0 & 2 & -4 & 0 & 2 & -2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.6667 & 2 & 0 & 1.3333 & -2 & 0 & 0.6667 & -1 & 0 & 0.3333 \\ -2 & 0 & -1 & -4 & 0 & -2 & 4 & 0 & -2 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.3333 & 2 & 0 & 0.6667 & -2 & 0 & 1.3333 & -1 & 0 & 0.6667 \\ -4 & 0 & -2 & -2 & 0 & -1 & 2 & 0 & -1 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0.6667 & 1 & 0 & 0.3333 & -1 & 0 & 0.6667 & -2 & 0 & 1.3333 \end{bmatrix} \quad (3.36)$$

The general flexural element stiffness matrix for thin rectangular plate is obtained by adding the individual stiffness along R , Q and RQ axis. Summing up Equation (3.24), Equation (3.30) and Equation (3.36) gives Equation (3.37);

$$K = K_R + K_{RQ} + K_Q =$$

$$\begin{bmatrix} 10.8 & 2.2 & 2.2 & -4.8 & 2.2 & 0.8 & -1.2 & 0.8 & 0.8 & -4.8 & 0.8 & 2.2 \\ 2.2 & 1.6 & 0 & -2.2 & 0.6 & 0 & -0.8 & 0.4 & 0 & 0.8 & 0.4 & 0 \\ 2.2 & 0 & 1.6 & 0.8 & 0 & 0.4 & -0.8 & 0 & 0.4 & -2.2 & 0 & 0.6 \\ -4.8 & -2.2 & 0.8 & 10.8 & -2.2 & 2.2 & -4.8 & -0.8 & 2.2 & -1.2 & -0.8 & 0.8 \\ 2.2 & 0.6 & 0 & -2.2 & 1.6 & 0 & -0.8 & 0.4 & 0 & 0.8 & 0.4 & 0 \\ 0.8 & 0 & 0.4 & 2.2 & 0 & 1.6 & -2.2 & 0 & 0.6 & -0.8 & 0 & 0.4 \\ -1.2 & -0.8 & -0.8 & -4.8 & -0.8 & -2.2 & 10.8 & -2.2 & -2.2 & -4.8 & -2.2 & -0.8 \\ 0.8 & 0.4 & 0 & -0.8 & 0.4 & 0 & -2.2 & 1.6 & 0 & 2.2 & 0.6 & 0 \\ 0.8 & 0 & 0.4 & 2.2 & 0 & 0.6 & -2.2 & 0 & 1.6 & -0.8 & 0 & 0.4 \\ -4.8 & 0.8 & -2 & -1.2 & 0.8 & -0.8 & -4.8 & 2.2 & -0.8 & 10.8 & 2.2 & -2.2 \\ 0.8 & 0.4 & 0 & -0.8 & 0.4 & 0 & -2.2 & 0.6 & 0 & 2.2 & 1.6 & 0 \\ 2.2 & 0 & 0.6 & 0.8 & 0 & 0.4 & -0.8 & 0 & 0.4 & -2.2 & 0 & 1.6 \end{bmatrix} \quad (3.37)$$

3.2 Formulation of Inertia Matrix of a Thin Rectangular Plate

It is clear that the general flexural element stiffness of the thin rectangular plate has been formulated. For a plate under free vibration, the inertia work for a flexural plate extracted from Equation (3.2) is as expressed in Equation (3.38);

$$Wk_i = \frac{m\lambda^2}{2} \int_0^1 \int_0^1 w^2 dx dy \quad (3.38)$$

Applying the non-dimensional parameter shown in Equation (3.11) gives

Equation (3.39);

$$Wk_i = \frac{m\lambda^2}{2} \int_0^1 \int_0^1 w^2 dR dQ \quad (3.39)$$

Recall from Equation (3.6);

$$w = [N] [N_i^{-1}]^T [w_i] \quad (3.6)$$

Substituting Equation (3.6) into Equation(3.39) and expanding it gives

Equations (3.40), (3.40a) and (3.40b);

$$Wk_i = \frac{m\lambda^2}{2} \int_0^1 \int_0^1 \left[[N] [N_i^{-1}]^T [w_i] \right]^2 dR dQ \quad (3.40)$$

$$Wk_i = \frac{m\lambda^2}{2} \int_0^1 \int_0^1 \left[[N] [N_i^{-1}]^T [w_i] \right]^T \left[[N] [N_i^{-1}] [w_i] \right] dR dQ \quad (3.40a)$$

$$Wk_i = [N_i^{-1}]^T [w_i]^T \left[\frac{m\lambda^2}{2} \int_0^1 \int_0^1 [N]^T [N] dR dQ \right] [N_i^{-1}] [w_i] \quad (3.40b)$$

Minimizing Equation (3.40b) by differentiating with respect to $[w_i]$ yields Equation (3.41);

$$K_i = [N_i]^{-1T} m \lambda^2 \int_0^1 \int_0^1 [N]^T [N] \partial R \partial Q \cdot [N_i]^{-1} \quad (3.41)$$

Equation (3.41) is called the inertia stiffness matrix of thin rectangular plate.

It can also be in this form as expressed in Equation (3.42)

$$K_i = m \lambda^2 [K_{ci}] \quad (3.42)$$

$$\text{Where } [K_{ci}] = [N_i]^{-1T} \cdot \int_0^1 \int_0^1 [N]^T [N] \partial R \partial Q \cdot [N_i]^{-1} \quad (3.43)$$

Recall that Equation (3.13) is the Equation that is used in obtaining the general flexural element stiffness matrix which is expressed in Equation (3.37)

$$K = K_R + K_{RQ} + K_Q \quad (3.37)$$

Where K_R , K_{RQ} , K_Q are all expressed in the following Equations.

$$K_R = [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial R^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2} \right] dR dQ \cdot [N_i]^{-1} \quad (3.24)$$

$$K_{RQ} = 2 * [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial R^2 Q^2} \right] dR dQ \cdot [N_i]^{-1} \quad (3.30)$$

$$K_Q = [N_i]^{-1T} \cdot \iint \left[\frac{\partial^2 N}{\partial Q^2} \right]^T \cdot \left[\frac{\partial^2 N}{\partial Q^2} \right] dR dQ \cdot [N_i]^{-1} \quad (3.36)$$

Note that in obtaining these stiffness along these directions K_R, K_{RQ}, K_Q , these terms $\frac{Db}{a^3}$,

$\frac{1}{\alpha^2}, \frac{1}{\alpha^4}$ were not applied in the matrix. Therefore, for a complete stiffness Equation these terms must be introduced back into the stiffness Equation which yields Equation (3.44).

$$K = \frac{Db}{a^3} \left[K_R + \frac{1}{\alpha^2} K_{RQ} + \frac{1}{\alpha^4} K_Q \right] \quad (3.44)$$

The governing differential equation for total strain energy of plate in vibration states that,

$$\Pi = K - K_i = 0 \quad (3.45)$$

Where K is expressed in Equation (3.44) and K_i is expressed in Equation (3.42).

Substituting the values of K and K_i yields Equation (3.46).

$$\frac{Db}{a^3} \left[K_R + \frac{1}{\alpha^2} K_{RQ} + \frac{1}{\alpha^4} K_Q \right] - m \lambda^2 [K_{ci}] = 0 \quad (3.46)$$

Solving Equation (3.46) yields Equation (3.47).

$$\lambda^2 = \frac{\frac{Db}{ma^3} \left[K_R + \frac{1}{\alpha^2} K_{RQ} + \frac{1}{\alpha^4} K_Q \right]}{K_{ci}} \quad (3.47)$$

The resonating frequency can be obtained as expressed in Equation (3.47).

The numerical formulation of the inertia matrix is done below.

By multiplying the Transpose of Equation (3.14) and Equation (3.14) yields Equation (3.48)

$$[N]^T [N] =$$

$$\begin{bmatrix} 1 & R & Q & R2 & RQ & Q2 & R3 & R2Q & RQ2 & Q3 & R3Q & RQ3 \\ R & R2 & RQ & R3 & R2Q & RQ2 & R4 & R3Q & R2Q2 & RQ3 & R4Q & R2Q3 \\ Q & RQ & Q2 & R2Q & RQ2 & Q3 & R3Q & R2Q2 & RQ3 & Q4 & R3Q2 & RQ4 \\ R2 & R3 & R2Q & R4 & R3Q & R2Q2 & R5 & R4Q & R3Q2 & R2Q3 & R5Q & R3Q3 \\ RQ & R2Q & RQ2 & R3Q & R2Q2 & RQ3 & R4Q & R3Q2 & R2Q3 & RQ4 & R4Q2 & R2Q4 \\ Q2 & RQ2 & Q3 & R2Q2 & RQ3 & Q4 & R3Q2 & R2Q3 & RQ4 & Q5 & R3Q3 & RQ5 \\ R3 & R4 & R3Q & R5 & R4Q & R3Q2 & R6 & R5Q & R4Q2 & R3Q3 & R6Q & R4Q3 \\ R2Q & R3Q & R2Q2 & R4Q & R3Q2 & R2Q3 & R5Q & R4Q2 & R3Q3 & R2Q4 & R5Q2 & R3Q4 \\ RQ2 & R2Q2 & RQ3 & R3Q2 & R2Q3 & RQ4 & R4Q2 & R3Q3 & R2Q4 & RQ5 & R4Q3 & R2Q5 \\ Q3 & RQ3 & Q4 & R2Q3 & RQ4 & Q5 & R3Q3 & R2Q4 & RQ5 & Q6 & R3Q4 & RQ6 \\ R3Q & R4Q & R3Q2 & R5Q & R4Q2 & R3Q3 & R6Q & R5Q2 & R4Q3 & R3Q4 & R6Q2 & R4Q4 \\ RQ3 & R2Q3 & RQ4 & R3Q3 & R2Q4 & RQ5 & R4Q3 & R3Q4 & R2Q5 & RQ6 & R4Q4 & R2Q6 \end{bmatrix} \quad (3.48)$$

Integrating Equation (3.48) and substituting the boundary values of R and Q coordinates which is assumed to be 1 as shown in Figure 3.1b yields Equation (3.49)

$$\int_0^1 \int_0^1 [N]^T [N] dR dQ =$$

1.000	0.500	0.500	0.333	0.250	0.333	0.250	0.167	0.167	0.250	0.125	0.125
0.500	0.333	0.250	0.250	0.167	0.167	0.200	0.125	0.111	0.125	0.100	0.083
0.500	0.250	0.333	0.167	0.167	0.250	0.125	0.111	0.125	0.200	0.083	0.100
0.333	0.250	0.167	0.200	0.125	0.111	0.167	0.100	0.083	0.083	0.083	0.063
0.250	0.167	0.167	0.125	0.111	0.125	0.100	0.083	0.083	0.100	0.067	0.067
0.333	0.167	0.250	0.111	0.125	0.200	0.083	0.083	0.100	0.167	0.063	0.083
0.250	0.200	0.125	0.167	0.100	0.083	0.143	0.083	0.067	0.063	0.071	0.050
0.167	0.125	0.111	0.100	0.083	0.083	0.083	0.067	0.063	0.067	0.056	0.050
0.167	0.111	0.125	0.083	0.083	0.100	0.067	0.063	0.067	0.083	0.050	0.056
0.250	0.125	0.200	0.083	0.100	0.167	0.063	0.067	0.083	0.143	0.050	0.071
0.125	0.100	0.083	0.083	0.067	0.063	0.071	0.056	0.050	0.050	0.048	0.040
0.125	0.083	0.100	0.063	0.067	0.083	0.050	0.050	0.056	0.071	0.040	0.048

(3.49)

Multiplication of inverse of Equation (3.14) and Equation (3.49) yields Equation (3.50)

$$[N]^{-1} \int_0^1 \int_0^1 [N]^T [N] dR dQ =$$

0.25000	0.07500	0.07500	0.03333	0.02222	0.03333	0.01786	0.00972	0.00972	0.01786	0.00512	0.00512
0.04167	0.01667	0.01389	0.00833	0.00556	0.00694	0.00476	0.00278	0.00278	0.00417	0.00159	0.00167
0.04167	0.01389	0.01667	0.00694	0.00556	0.00833	0.00417	0.00278	0.00278	0.00476	0.00167	0.00159
0.25000	0.17500	0.07500	0.13333	0.05278	0.03333	0.10714	0.04028	0.02361	0.01786	0.03238	0.01274
-0.04167	-0.02500	-0.01389	-0.01667	-0.00833	-0.00694	-0.00119	-0.00556	-0.00417	-0.00417	-0.00397	-0.00250
0.04167	0.02778	0.01667	0.02083	0.01111	0.00833	0.01667	0.00833	0.00556	0.00476	0.00667	0.00317
0.25000	0.17500	0.17500	0.13333	0.12222	0.13333	0.10714	0.09306	0.09306	0.10714	0.07476	0.07476
-0.04167	-0.02500	-0.02778	-0.01667	-0.01667	-0.02083	-0.01190	-0.01111	-0.01250	-0.01667	-0.00794	-0.01000
-0.04167	-0.02778	-0.02500	-0.02083	-0.01667	-0.01667	-0.01667	-0.01250	-0.01111	-0.01190	-0.01000	-0.00794
0.25000	0.07500	0.17500	0.03333	0.05278	0.13333	0.01786	0.02361	0.04028	0.10714	0.01274	0.03238
0.04167	0.01667	0.02778	0.00833	0.01111	0.02083	0.00476	0.00556	0.00833	0.01667	0.00317	0.00667
-0.04167	-0.01389	-0.02500	-0.00694	-0.00833	-0.01667	-0.00417	-0.00417	-0.00556	-0.01190	-0.00250	-0.00397

(3.50)

Finally, the matrix multiplication of transpose of Equation (3.14) inverse and Equation (3.50) and inverse of Equation (3.14), yields the inertia matrix shown in Equation (3.51);

$$K_i = [N^{-1}]^T \left[\int_0^1 \int_0^1 [N]^T [N] dR dQ \right] [N]^{-1} =$$

$$\begin{bmatrix} 0.13706 & 0.01829 & 0.01829 & 0.04865 & -0.01087 & 0.00790 & 0.01563 & -0.00460 & -0.00460 & 0.04865 & 0.00790 & -0.01087 \\ 0.01829 & 0.00317 & 0.00250 & 0.01087 & -0.00238 & 0.00167 & 0.00460 & -0.00119 & -0.00111 & 0.00790 & 0.00159 & -0.00167 \\ 0.01829 & 0.00250 & 0.00317 & 0.00790 & -0.00167 & 0.00159 & 0.00460 & -0.00111 & -0.00119 & 0.01087 & 0.00167 & -0.00238 \\ 0.04865 & 0.01087 & 0.00790 & 0.13706 & -0.01829 & 0.01829 & 0.04865 & -0.00790 & -0.01087 & 0.01563 & 0.00460 & -0.00460 \\ -0.01087 & -0.00238 & -0.00167 & -0.01829 & 0.00317 & -0.00250 & -0.00790 & 0.00159 & 0.00167 & -0.00460 & -0.00119 & 0.00111 \\ 0.00790 & 0.00167 & 0.00159 & 0.01829 & -0.00250 & 0.00317 & 0.01087 & -0.00167 & -0.00238 & 0.00460 & 0.00111 & -0.00119 \\ 0.01563 & 0.00460 & 0.00460 & 0.04865 & -0.00790 & 0.01087 & 0.13706 & -0.01829 & -0.01829 & 0.04865 & 0.01087 & -0.00790 \\ -0.00460 & -0.00119 & -0.00111 & -0.00790 & 0.00159 & -0.00167 & -0.01829 & 0.00317 & 0.00250 & -0.01087 & -0.00238 & 0.00167 \\ -0.00460 & -0.00111 & -0.00119 & -0.01087 & 0.00167 & -0.00238 & -0.01829 & 0.00250 & 0.00317 & -0.00790 & -0.00167 & 0.00159 \\ 0.04865 & 0.00790 & 0.01087 & 0.01563 & -0.00460 & 0.00460 & 0.04865 & -0.01087 & -0.00790 & 0.13706 & 0.01829 & -0.01829 \\ 0.00790 & 0.00159 & 0.00167 & 0.00460 & -0.00119 & 0.00111 & 0.01087 & -0.00238 & -0.00167 & 0.01829 & 0.00317 & -0.00250 \\ -0.01087 & -0.00167 & -0.00238 & -0.00460 & 0.00111 & -0.00119 & -0.00790 & 0.00167 & 0.00159 & -0.01829 & -0.00250 & 0.00317 \end{bmatrix} \quad (3.51)$$

3.3 Development of Computer Program

A computer program for free vibration analysis of rectangular thin plate was developed using Mathematical laboratory program (MATLAB) version 8.1 and model R2013a. The program is user friendly, and its interphase is shown in the Appendixes.

The algorithm for the MATLAB program that used the stiffness matrix and inertia matrix to obtain the resonating frequencies of plates of various boundary conditions are listed below.

3.3.1 Algorithm

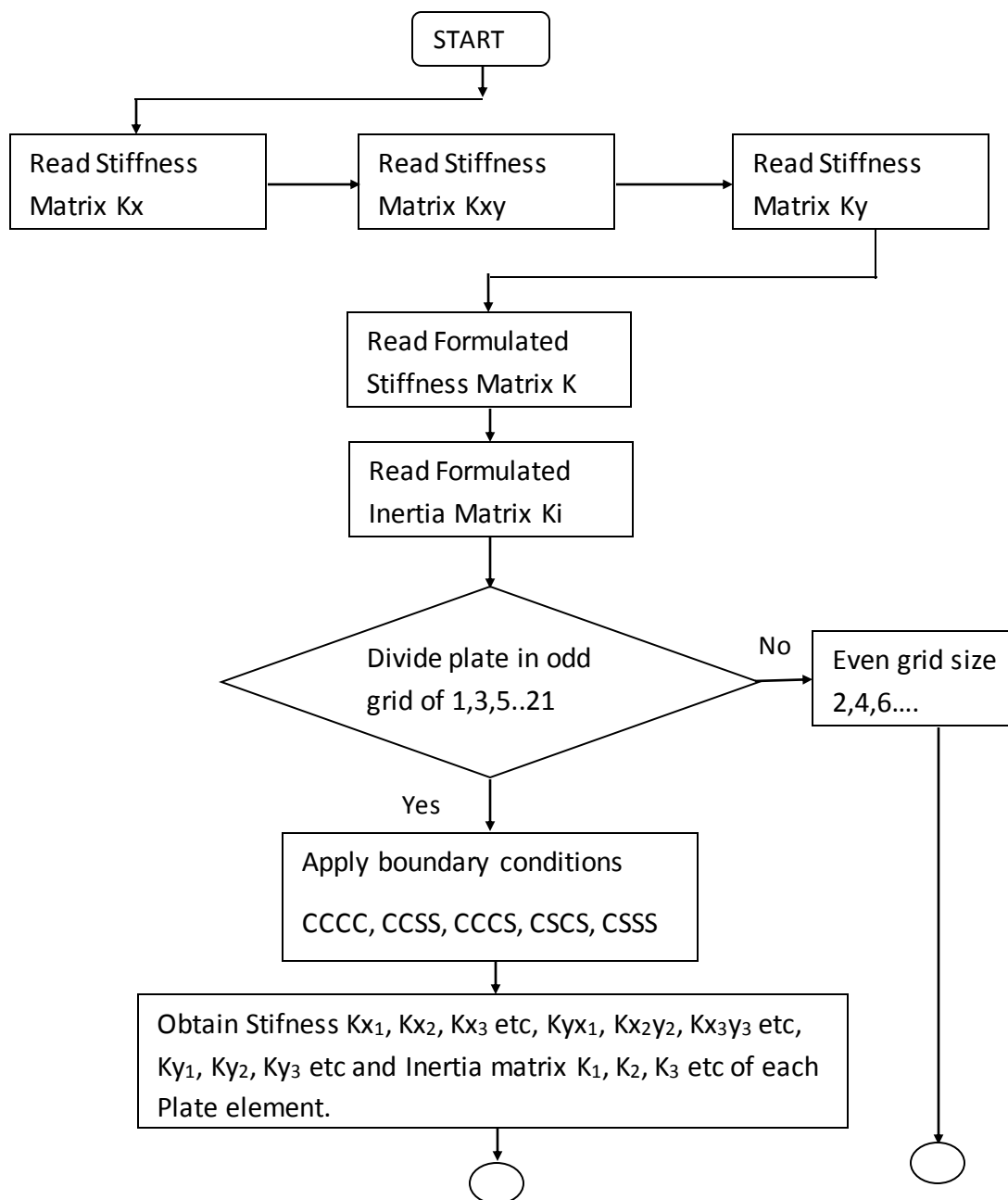
- i. Discretize the plate according to grid size i.e. 1,3,5,7etc.
- ii. Considering the boundary conditions, number all the deflections and rotations in each node starting from the central node coinciding with the central deflection of the plate. The boundary conditions and the number of grid size determine the size of matrix formed.
- iii. Using the deflections and rotations in the K_x , K_{xy} and K_y which are constants in the general plate model, obtain the stiffness matrix K_x , K_{xy} , K_y of each element of the discretized plate.
- iv. Based on the aspect ratio, obtain the general stiffness K of the entire plate by summing up all the individual stiffnesses i.e. $(K_{x1} + K_{x1y1} + K_{y1}) + (K_{x2} + K_{x2y2} + K_{y2})$ etc of each element of the discretized plate.
- v. Determine the inertia stiffness K_i of each element of discretized plate using the method of posting from the inertia stiffness of a plate model.
- vi. The inertia matrix K_i of the discretized plate is obtained by adding together all the individual inertia matrix of each element.
- vii. The square of the resonating frequency λ^2 is obtained by dividing the stiffness matrix K with the inertia matrix K_i .

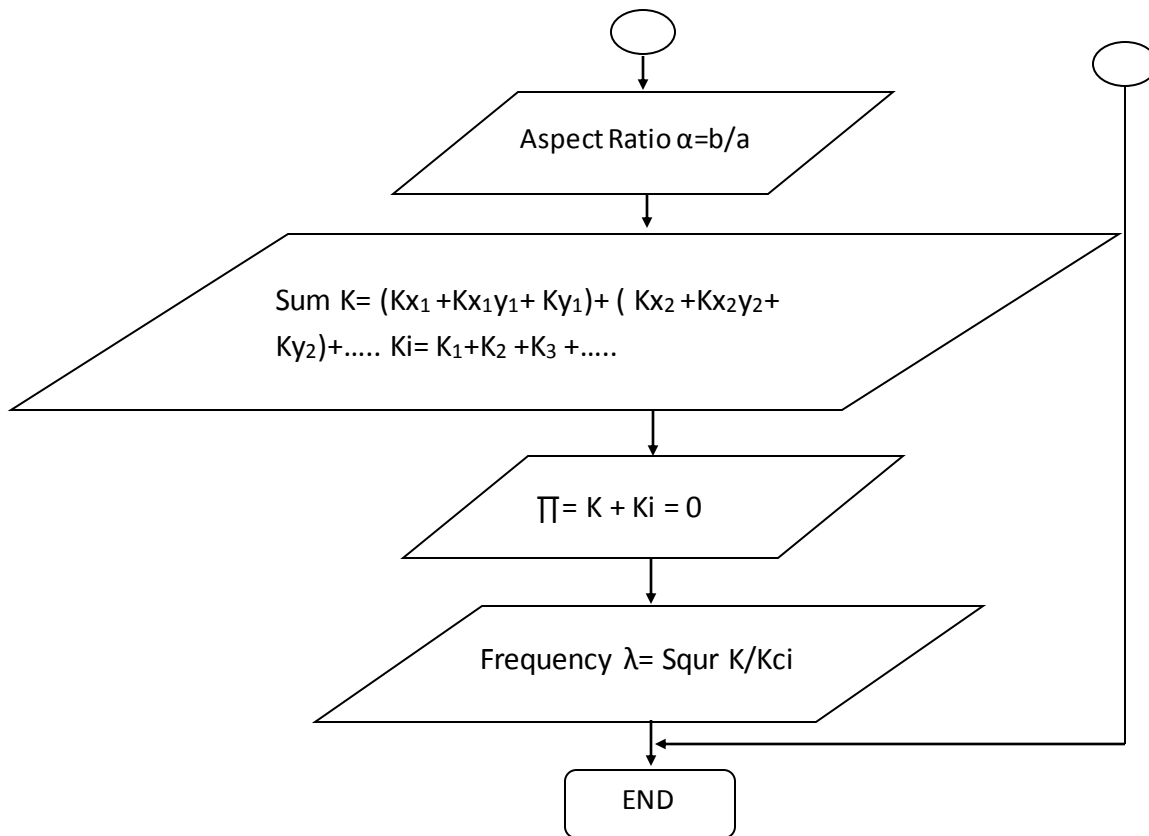
viii. The resonating frequency λ which is an eigenvalue problem is obtained by the square

root of. $\frac{K}{K_{ci}}$

3.3.2 Flow Chart

The flow chart presented below is based on the algorithm which is used to develop a MATLAB program.





3.4 Comparison of the Present Study and that Obtained from the Previous Methods.

The following are the list of previous methods used in comparing the results of the present study.

1. Free Vibration Analysis of an Around-Clamped Rectangular Thin Orthotropic Plate Using Taylor-Mclaurin Shape function by Onwuka et al. (2016)
2. Dynamic Analysis of Thin Rectangular Flat Isotropic Plates Using Galerkin's Method by Njoku. (2013)
3. Vibration of Plates by Chakraverty. (2009)
4. Vibration of Plates by Leissa. (1969)
5. Natural Frequencies of Orthotropic Rectangular Plates Obtained by Iterative Reduction of the Partial Differential Equation by Sakata et al. (1996)
6. Free Vibration Analysis of Rectangular Plates by Gorman. (1982)

CHAPTER FOUR
RESULTS AND DISCUSSIONS

4.1 Results

The results obtained from this study are as detailed in this order:

**4.1.1 Formulated General Flexural Element Stiffness Matrix
of Thin Rectangular Plate.**

$$K = K_R + K_{RQ} + K_Q =$$

$$\begin{bmatrix} 10.8 & 2.2 & 2.2 & -4.8 & 2.2 & 0.8 & -1.2 & 0.8 & 0.8 & -4.8 & 0.8 & 2.2 \\ 2.2 & 1.6 & 0 & -2.2 & 0.6 & 0 & -0.8 & 0.4 & 0 & 0.8 & 0.4 & 0 \\ 2.2 & 0 & 1.6 & 0.8 & 0 & 0.4 & -0.8 & 0 & 0.4 & -2.2 & 0 & 0.6 \\ -4.8 & -2.2 & 0.8 & 10.8 & -2.2 & 2.2 & -4.8 & -0.8 & 2.2 & -1.2 & -0.8 & 0.8 \\ 2.2 & 0.6 & 0 & -2.2 & 1.6 & 0 & -0.8 & 0.4 & 0 & 0.8 & 0.4 & 0 \\ 0.8 & 0 & 0.4 & 2.2 & 0 & 1.6 & -2.2 & 0 & 0.6 & -0.8 & 0 & 0.4 \\ -1.2 & -0.8 & -0.8 & -4.8 & -0.8 & -2.2 & 10.8 & -2.2 & -2.2 & -4.8 & -2.2 & -0.8 \\ 0.8 & 0.4 & 0 & -0.8 & 0.4 & 0 & -2.2 & 1.6 & 0 & 2.2 & 0.6 & 0 \\ 0.8 & 0 & 0.4 & 2.2 & 0 & 0.6 & -2.2 & 0 & 1.6 & -0.8 & 0 & 0.4 \\ -4.8 & 0.8 & -2 & -1.2 & 0.8 & -0.8 & -4.8 & 2.2 & -0.8 & 10.8 & 2.2 & -2.2 \\ 0.8 & 0.4 & 0 & -0.8 & 0.4 & 0 & -2.2 & 0.6 & 0 & 2.2 & 1.6 & 0 \\ 2.2 & 0 & 0.6 & 0.8 & 0 & 0.4 & -0.8 & 0 & 0.4 & -2.2 & 0 & 1.6 \end{bmatrix} \quad (3.37)$$

4.1.2 Formulated Inertia Matrix of Thin Rectangular Plate.

$$K_i = [N^{-1}]^T \left[\int_0^1 \int_0^1 [N]^T [N] dRdQ \right] [N]^{-1} =$$

$$\begin{bmatrix} 0.13706 & 0.01829 & 0.01829 & 0.04865 & -0.01087 & 0.00790 & 0.01563 & -0.00460 & -0.00460 & 0.04865 & 0.00790 & -0.01087 \\ 0.01829 & 0.00317 & 0.00250 & 0.01087 & -0.00238 & 0.00167 & 0.00460 & -0.00119 & -0.00111 & 0.00790 & 0.00159 & -0.00167 \\ 0.01829 & 0.00250 & 0.00317 & 0.00790 & -0.00167 & 0.00159 & 0.00460 & -0.00111 & -0.00119 & 0.01087 & 0.00167 & -0.00238 \\ 0.04865 & 0.01087 & 0.00790 & 0.13706 & -0.01829 & 0.01829 & 0.04865 & -0.00790 & -0.01087 & 0.01563 & 0.00460 & -0.00460 \\ -0.01087 & -0.00238 & -0.00167 & -0.01829 & 0.00317 & -0.00250 & -0.00790 & 0.00159 & 0.00167 & -0.00460 & -0.00119 & 0.00111 \\ 0.00790 & 0.00167 & 0.00159 & 0.01829 & -0.00250 & 0.00317 & 0.01087 & -0.00167 & -0.00238 & 0.00460 & 0.00111 & -0.00119 \\ 0.01563 & 0.00460 & 0.00460 & 0.04865 & -0.00790 & 0.01087 & 0.13706 & -0.01829 & -0.01829 & 0.04865 & 0.01087 & -0.00790 \\ -0.00460 & -0.00119 & -0.00111 & -0.00790 & 0.00159 & -0.00167 & -0.01829 & 0.00317 & 0.00250 & -0.01087 & -0.00238 & 0.00167 \\ -0.00460 & -0.00111 & -0.00119 & -0.01087 & 0.00167 & -0.00238 & -0.01829 & 0.00250 & 0.00317 & -0.00790 & -0.00167 & 0.00159 \\ 0.04865 & 0.00790 & 0.01087 & 0.01563 & -0.00460 & 0.00460 & 0.04865 & -0.01087 & -0.00790 & 0.13706 & 0.01829 & -0.01829 \\ 0.00790 & 0.00159 & 0.00167 & 0.00460 & -0.00119 & 0.00111 & 0.01087 & -0.00238 & -0.00167 & 0.01829 & 0.00317 & -0.00250 \\ -0.01087 & -0.00167 & -0.00238 & -0.00460 & 0.00111 & -0.00119 & -0.00790 & 0.00167 & 0.00159 & -0.01829 & -0.00250 & 0.00317 \end{bmatrix} \quad (3.51)$$

4.1.3 MATLAB Program

A MATLAB program is written for the various boundary conditions including CCCC, CCSS, CSCS, CSSS and CCCS.

All of them are shown in the Appendixes.

4.1.4 Numerical Examples

1. Determine the natural frequency (λ) of a rectangular thin plate clamped on all edges when the plate is divided into finite elements as shown in Figure 4.1.

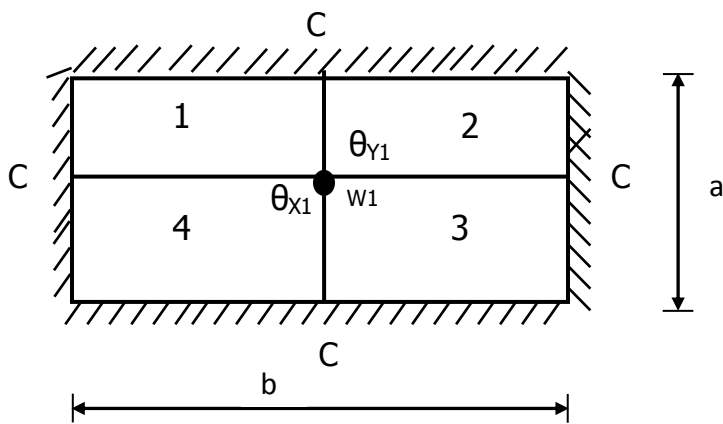


Figure 4.1 CCCC rectangular plate

2. Determine the natural frequency (λ) of a rectangular thin plate clamped and simply supported on adjacent edges when the plate is divided into finite elements as shown in

Figure 4.2.

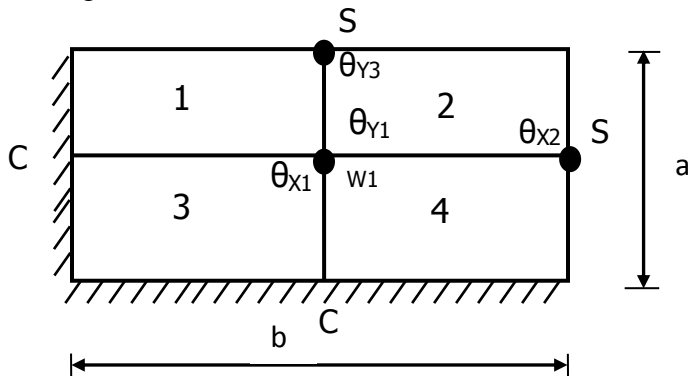


Figure 4.2 CCSS rectangular plate

3. Determine the natural frequency of a rectangular thin plate clamped on two opposite edges and simply supported on two opposite edges when the plate is divided into finite elements as shown in Figure 4.3.

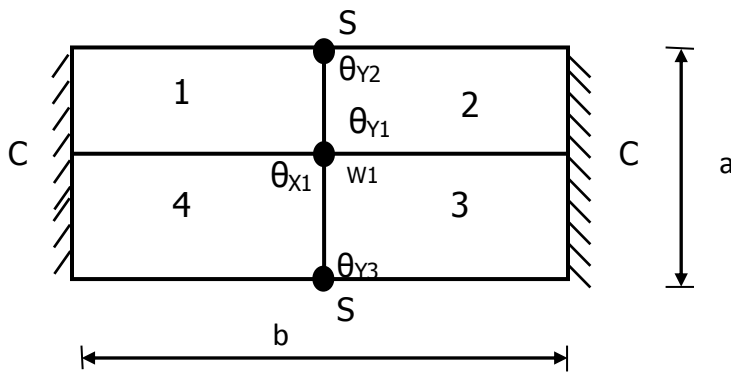


Figure 4.3 CSCS rectangular plate

4. Determine the natural frequency (λ) of a rectangular thin plate clamped on one edge and simply supported on three edges when the plate is divided into finite elements as shown in Figure 4.4.

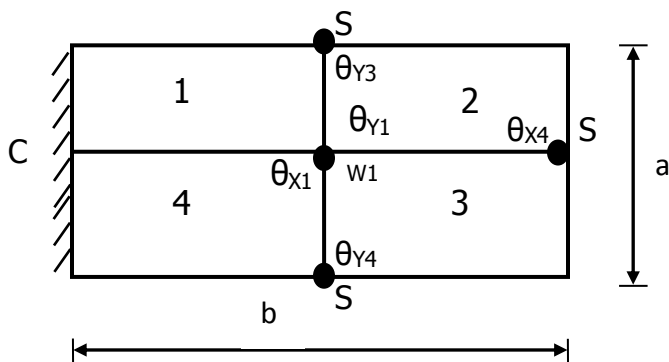


Figure 4.4 CSSS rectangular plate

5. Determine the natural frequency (λ) of a rectangular thin plate clamped on three edges and simply supported on one edge when the plate is divided into finite elements as shown in Figure 4.5.

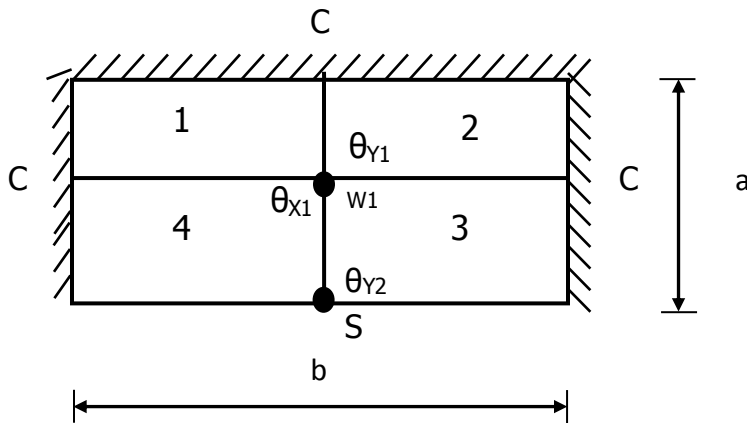


Figure 4.5 CCCS rectangular plate

The results of the numerical example shown in section 4.14 are highlighted accordingly including the aspect ratio (α) which is a factor that depends on the dimensions of the plate was used to determine the fundamental natural frequency of plate for different plate dimensions. The aspect ratio is in the form of $\alpha=b/a$. In the program formulated, the natural frequency has in it the minimum and maximum values, but our interest was the minimum values. It was the square root of the minimum values that gave the natural frequencies of the plates.

4.1.4.1 Detailed Graphical Representation of Results for all Boundary Conditions of Thin Rectangular Plates used in this Study.

The following graphical representation explains the results of the fundamental natural frequency of the chosen boundary conditions tabulated.

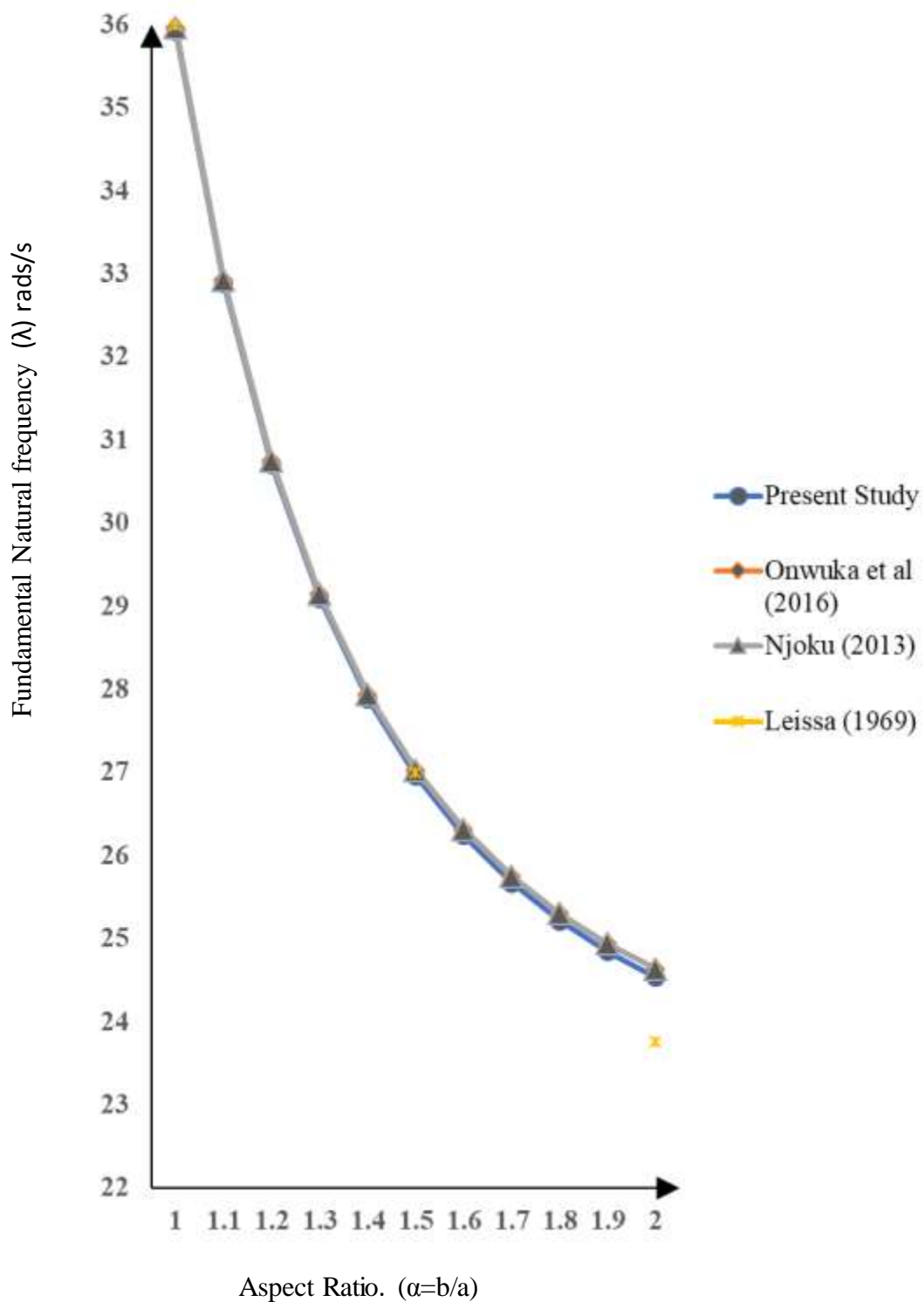


Fig. 4.6 Graphical Representation of Fundamental Natural frequency for CCCC Plates support conditions of Aspect Ratios of b/a

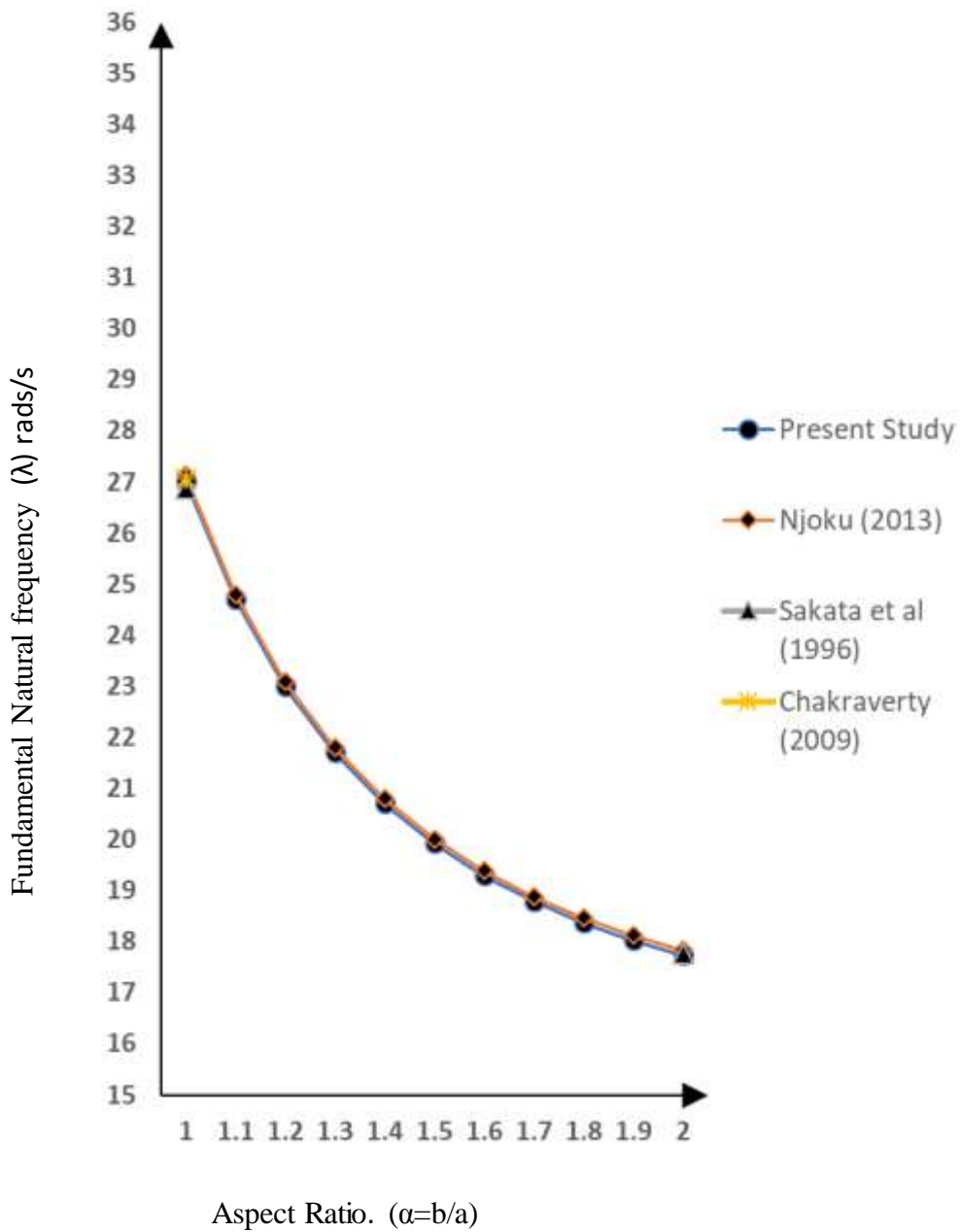


Fig. 4.7 Graphical Representation of Fundamental Natural frequency for CCSS Plates support conditions of Aspect Ratios of b/a

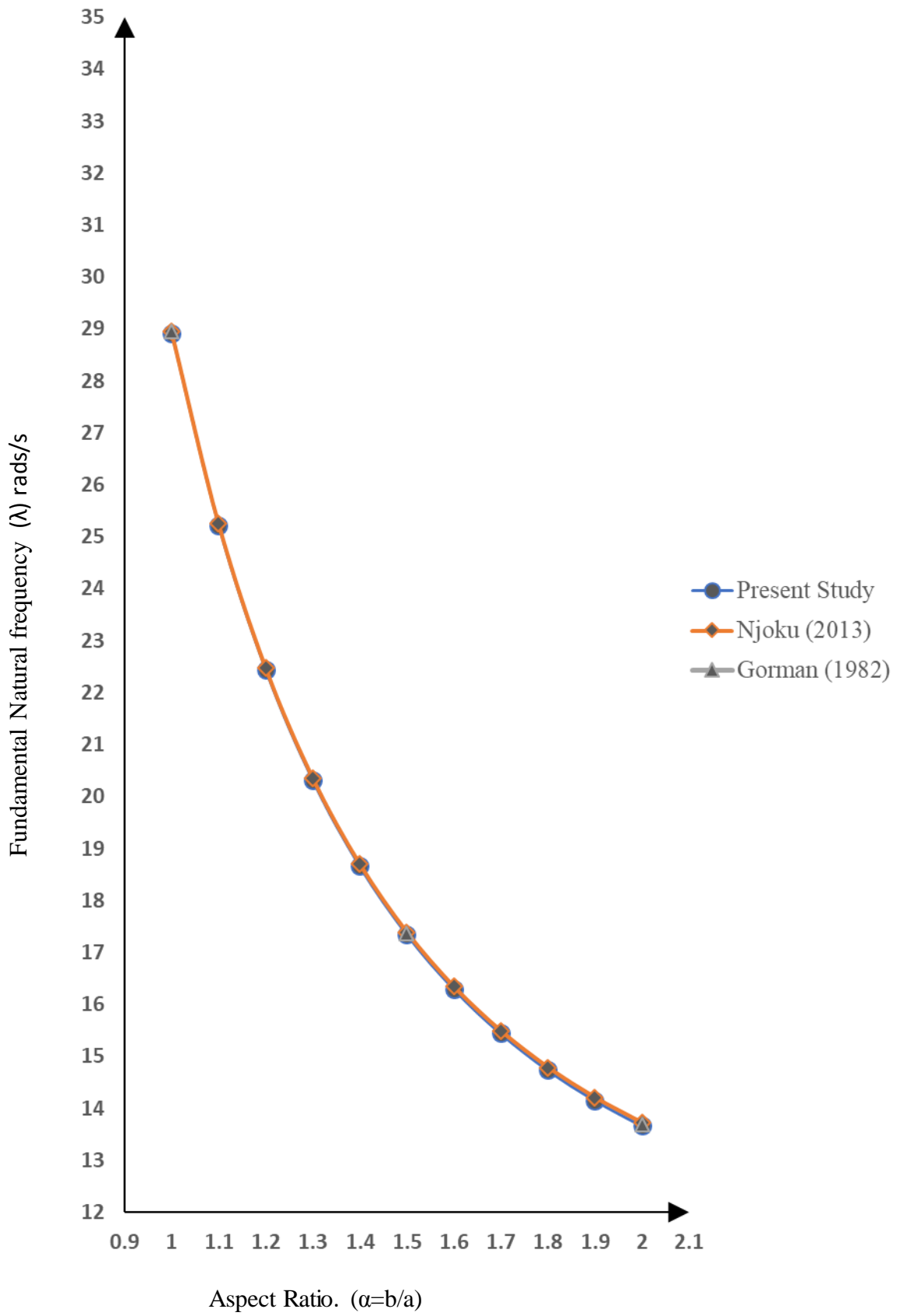


Fig. 4.8 Graphical Representation of Fundamental Natural frequency for CSCS Plates support conditions of Aspect Ratios of b/a

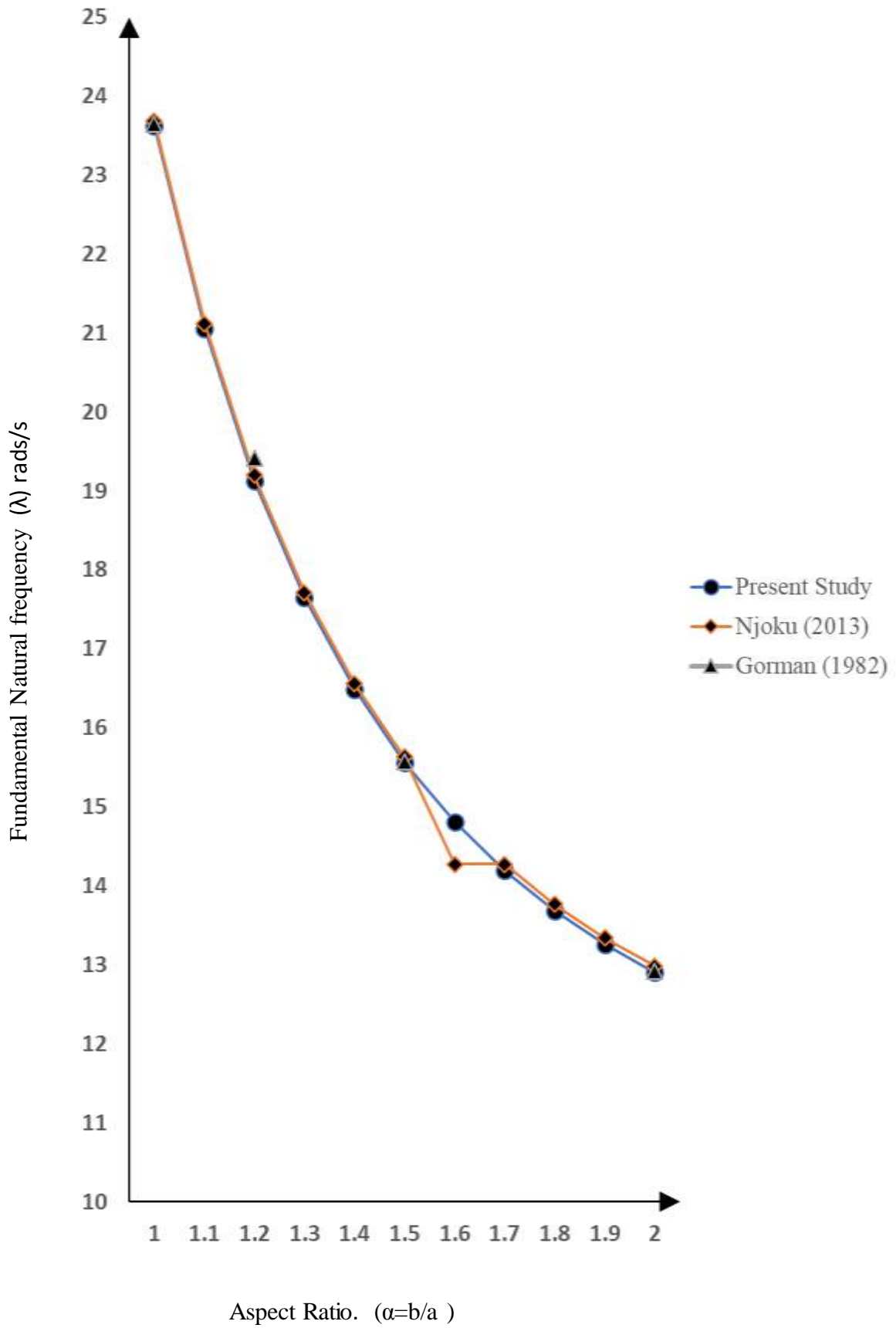


Fig. 4.9 Graphical Representation of Fundamental Natural frequency for CSSS Plates support conditions of Aspect Ratios of b/a

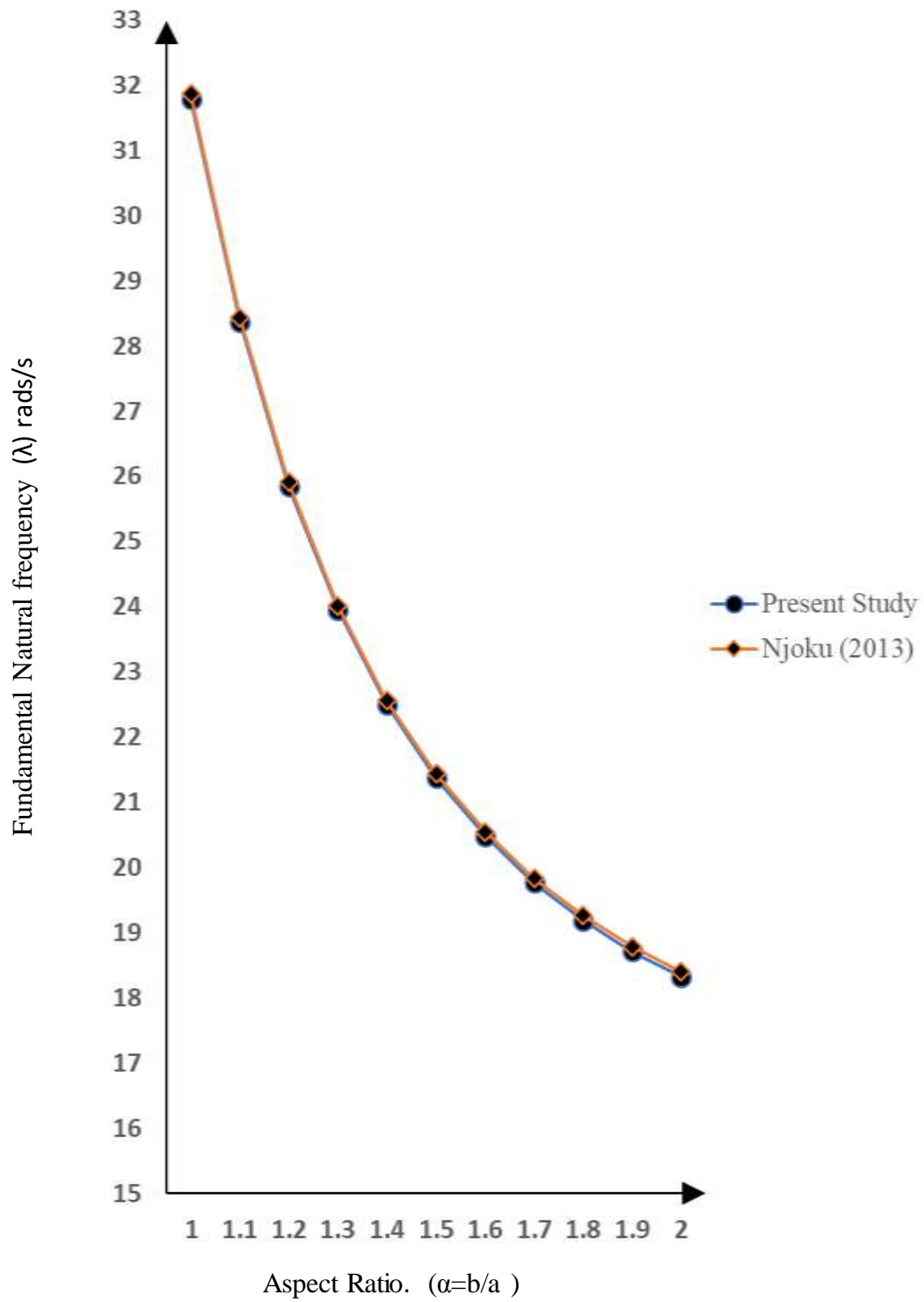


Fig. 4.10 Graphical Representation of Fundamental Natural frequency for CCCS Plates support conditions of Aspect Ratios of b/a

4.2 Discussion of Results

In this section, the natural frequency results for various boundary conditions of thin plates obtained using finite element method are discussed in detail.

4.2.1 CCCC Rectangular Thin Plates

Considering the numerical example one of section 4.14, the program of Appendix A was used to determine the resonating frequency (λ) of a rectangular plate starting with a total number of three displacements which is equivalent to one grid size. A grid size (n) of increasing odd numbers i.e. ($n=3$), ($n=5$), ($n=7$) etc. were used in the program. The results of (λ) which is an eigenvalue problem were tabulated as shown in Table 4.1.

In addition, for verification purpose, the accuracy of the results obtained by the present method is compared with those given by other approximate methods. In table 4.1, the aspect ratio (α) ranging from 1.0 to 2.0 was used to determine the percentage difference between the present study and that of other approximate methods.

Looking at table 4.1 critically, it was observed that as the grid size (n) increases horizontally across the table the accuracy of the resonating frequency (λ) increases. This is as a result of an increase in the discretization of the plate element which is one of the major factors of finite element method. Investigating vertically on the table, it was observed that as the aspect ratio of the plate increases, the corresponding natural frequency decreases for each grid size. For the aspect ratio of 1.0 under the grid size $n=3$ and $n=21$, the natural frequencies (λ) are 34.70339 and 35.92795 respectively while for the aspect ratio of 2.0 the natural frequency (λ) are 23.59059 and 24.53527 respectively.

In table 4.2, the percentage difference between the present study and other scholars for this boundary condition were computed. Based on the grid size $n=21$ the results of the resonating

frequency under this grid size was compared with the results of Onwuka et al (2016), Njoku (2013) and Liessa (1969). For the present study, the resonating frequency (λ_1) corresponding to grid size $n=21$ and aspect ratio 1.0 was 35.92795, while the results of Onwuka et al (2016), Njoku (2013) and Liessa (1969) were 35.967, 35.96736 and 35.9866 respectively. The percentage difference between the present study (λ_1) and that of the previous studies (λ_2), (λ_3) and (λ_4) are 0.108%, 0.110% and 0.163%. Hence, these values show that the present study gives a solution that is very close to the previous research work. The graph of the fundamental natural frequency (λ) against the aspect ratio α shown in Figure 4.6 explains in detail how close the fundamental natural frequency of this present study for CCCC plate is with other approximate methods.

4.2.2 CCSS Rectangular Thin Plates

Addressing the problem two of section 4.14 by using MATLAB program formed in Appendix B, the natural frequencies (λ) were obtained and tabulated on table 4.3 with respect to the aspect ratio. A range of grid size ($n=3$) and ($n=21$) was used.

Table 4.3 shows the resonating frequency (λ) of different grid size (n). Inspecting the table horizontally, the accuracy of the result increases with the increase in grid size. The increase in the grid size shows the number of discretization the plate element undergoes which is one of the importance of finite element method. For the aspect ratio of 1.5, the results of the natural frequency (λ) for grid size $n=3$ and $n=21$ are 19.27194 and 19.92419 respectively.

Based on the results on Table 4.3, Table 4.4 shows the results of the present study (λ_1) for $n=21$ which was compared with the results of previous studies from Njoku (2013) (λ_2), Saka et al (1996) (λ_3) and Charkraverty (2009) (λ_4). The results of these previous studies for the

aspect ratio of 1.0 are 27.12846, 26.867 and 27.055 while that of the present study is 27.02033. Also, the percentage differences corresponding to the above results for the present study are 0.4001% for Njoku (2013), 0.5675% for Saka et al (1996) and 0.1283% for Charkraverty (2009) under the aspect ratio of 1.0. These values show that the natural frequency (λ_1) of present study is close to the natural frequency of the previous study, (λ_2), (λ_3) and (λ_4). The graph on figure 4.7 presents the clear picture that the fundamental natural frequency (λ) obtained in this present study agrees well with other approximate solution of Njoku, Saka et al, and Charkraverty under the same aspect ratio α .

4.2.3 CSCS Rectangular Thin Plates

Table 4.5 shows the solution of problem three of section 4.14. The solution of the resonating frequency (λ) was obtained with the MATLAB program presented in Appendix C. The program solved for a grid size of (n=3) down to (n=21).

Looking at the grid size n=3 to the grid size of n=21, the natural frequency accuracy increases for all aspect ratios i.e. 1.0 to 2.0. This gradual increase in accuracy across table 4.5 shows that the higher the discretizations of the plate with respect to the grid size (n), the more accurate the result; which is one of the advantages of finite element method. For aspect ratio 1.3, the results of the natural frequency (λ) for n=3 and n=21 respectively are 19.74984 and 20.31927. Looking vertically on table 4.5 the value of the natural frequency reduces as the aspect ratio of the plate increases.

Further analysis of table 4.5 is shown in table 4.6. The natural frequency (λ) under grid size n=21 is selected from table 4.5 to calculate the percentage differences between the present study and the previous approximate methods. The results of the present study (λ_1) for n=21 were compared with the results of previous studies from Njoku (2013) (λ_2), Goman

(1982) (λ_3) and Charkraverty (2009) (λ_4) and was found to be close. Under aspect ratio of 1.0, the result of natural frequency (λ_1) for the present study is 22.44284, while the results of Njoku (2013) (λ_2), Goman (1982) (λ_3) and Charkraverty (2009) (λ_4) are 28.95663, 28.95 and 28.950 respectively. Also, the respective percentage difference between present study and previous studies are 0.1437%, 0.1208%, and 0.1208%. The graph of the fundamental natural frequency (λ) against the aspect ratio α was shown in Figure 4.8 for a more detailed result. When studying the graph closely you will notice that the values of the fundamental natural frequency obtained from the present study follow the same curve with the one obtained from other approximate method of Njoku and Gorman when plotted with the same aspect ratio α . This observation shows that the present study obtained an accurate result from the finite element method used when compared with other methods.

4.2.4 CSSS Rectangular Thin Plates

Referring to numerical example, four of section 4.14, the natural frequency (λ) with different aspect ratios was obtained using the MATLAB program in Appendix D. The grid size of odd numbers with odd number increment was used i.e. $n=3$, $n=5$, to $n=21$ etc. This was tabulated in table 4.7.

Critically looking at table 4.7, it was observed that as the grid size (n) increases horizontally across the table the accuracy of the natural frequency (λ) increases. This increment in accuracy is as a result of increase in the discretization of the plate element which is one of the importance of finite element method. Investigating vertically on the table, it was observed that as the aspect ratio of the plate increases, the corresponding natural frequency decreases for each grid size. For aspect ratio of 1.2 under grid size $n=3$ and $n=21$ the natural

frequencies (λ) are 18.64727 and 19.13359 respectively while for the aspect ratio of 2.0, the natural frequencies (λ) are 12.88109 and 12.90251 respectively.

In view of this, from table 4.8, the percentage difference between the present study and other scholars for this boundary condition were computed. The natural frequencies of the present study were compared with the results of Njoku(2013), Goman (1982) and Charkraverty (2009). For the present study, the resonating frequency (λ_1) corresponding to grid size $n=21$ and aspect ratio 1.0 was 23.62158, while the results of Njoku(2013) (λ_2), Goman(1982) (λ_3) and Charkraverty (2009) (λ_4) are 23.67982, 23.65 and 23.646 respectively. The percentage difference between present study (λ_1) and that of previous studies (λ_2), (λ_3) and (λ_4) are 0.2465%, 0.1203% and 0.1034%. Hence the value of present study shows that is very close to the previous research work. The graph of the fundamental natural frequency (λ) against the aspect ratio α was shown in Figure 4.9 for a more detailed result.

4.2.5 CCCS Rectangular Thin Plates

The solution of example five of section 4.14 was given on table 4.9. The MATLAB program outlined in Appendix E gives the fundamental natural frequency for this boundary condition of a thin plate by using different aspect ratios and corresponding grid size (n).

Table 4.9 shows the natural frequency of different grid size (n) from $n=3$ to $n=21$. Looking at the table horizontally, the accuracy of the natural frequency for this boundary condition increases with increase in grid size. The discretization of the plate element which is represented by the grid size shows the importance of finite element method. Also looking vertically downwards on table 4.9, the natural frequency of the plate reduces with the

increase in the aspect ratio of the plate. For the aspect ratio of 1.0, the results of the natural frequency (λ) for grid size $n=3$ and $n=21$ are 30.71895 and 31.78051 respectively.

In view of the results in Table 4.9, Table 4.10 shows the results of the present study (λ_1) for $n=21$ which was compared with the results of previous studies from Njoku(2013) (λ_2) and Charkraverty(2009) (λ_3).

For the present study, the resonating frequency (λ_1) corresponding to grid size $n=21$ and aspect ratio 1.0 was 31.78051, while the results of Njoku(2013) (λ_2) and Charkraverty (2009) (λ_3) are 31.86803 and 31.827 respectively. The percentage difference between present study (λ_1) and that of previous studies (λ_2) and (λ_3) are 0.2753% and 0.1461%. Hence the value of present study shows that is very close to the previous research work. The graph of the fundamental natural frequency (λ) against the aspect ratio α was shown in Figure 4.10 for a more detailed result. The graph shows that the result obtained from the present study and other approximate methods follow the same curve. From the graph, you will discover that the curves are almost on the same pathway because of close values of very small percentage difference. Hence the finite element method gives the fundamental natural frequency of a plate in free vibration.

DETAILED RESULT OF VARIOUS BOUNDARY CONDITIONS OF THIN RECTANGULAR PLATES.

Table 4.1 The natural frequency (λ) for different aspect ratio and grid size (n) for CCCC plates.

Aspect ratio $\alpha = b/a$	Natural Frequency (λ)For Grid size n = 3	Natural Frequency (λ)For Grid size n = 5	Natural Frequency (λ)For Grid size n = 7	Natural Frequency (λ)For Grid size n = 9	Natural Frequency (λ)For Grid size n = 11	Natural Frequency (λ)For Grid size n = 13	Natural Frequency (λ)For Grid size n = 15	Natural Frequency (λ)For Grid size n = 17	Natural Frequency (λ)For Grid size n = 19	Natural Frequency (λ)For Grid size n = 21
1.0	34.70339	35.30331	35.57776	35.71716	35.79633	35.84528	35.87755	35.89989	35.91598	35.92795
1.1	31.76597	32.31657	32.56836	32.69627	32.76892	32.81384	32.84345	32.86396	32.87872	32.8897
1.2	29.64798	30.16551	30.40188	30.52199	30.59023	30.63243	30.66025	30.67952	30.6934	30.70371
1.3	28.08517	28.58056	28.80627	28.92099	28.98619	29.02653	29.05312	29.07154	29.0848	29.09466
1.4	26.90753	27.38847	27.60671	27.71764	27.78071	27.81974	27.84547	27.8633	27.87613	27.88568
1.5	26.00278	26.47495	26.68792	26.79613	26.85768	26.89576	26.92088	26.93829	26.95082	26.96014
1.6	25.29505	25.76289	25.97212	26.07836	26.13879	26.17619	26.20086	26.21795	26.23027	26.23942
1.7	24.73195	25.19915	25.40577	25.51056	25.57015	25.60704	25.63138	25.64824	25.66039	25.66941
1.8	24.27661	24.74645	24.95131	25.05502	25.11398	25.15048	25.17455	25.19124	25.20326	25.21219
1.9	23.9026	24.37815	24.58193	24.68483	24.7433	24.77949	24.80336	24.81991	24.83183	24.84069
2.0	23.59059	24.0749	24.27817	24.38046	24.43854	24.47449	24.4982	24.51464	24.52648	24.53527

Table 4.2 Results of natural frequency (λ) for CCCC plate of present study and the results of previous studies with their percentage difference for different aspect ratios.

Aspect ratio $\alpha = b/a$	Present Study (λ_1) For grid size n=21	Onwuka et al (2016) (λ_2)	Njoku (2013) (λ_3)	Leissa (1969) (λ_4)	Percentage Difference For (λ_2) & (λ_1)	Percentage Difference For (λ_3) & (λ_1)	Percentage Difference For (λ_4) & (λ_1)
1.0	35.92795	35.967	35.96736	35.9866	0.108	0.110	0.163
1.1	32.88970	32.929	32.92891	-	0.119	0.119	-
1.2	30.70371	30.748	30.7477	-	0.144	0.143	-
1.3	29.09466	29.146	29.1459	-	0.176	0.176	-
1.4	27.88568	27.945	27.94509	-	0.213	0.213	-
1.5	26.96014	27.028	27.0278	27.0000	0.252	0.251	0.148
1.6	26.23942	26.315	26.31492	-	0.288	0.288	-
1.7	25.66941	25.752	25.75212	-	0.322	0.322	-
1.8	25.21219	25.301	25.30138	-	0.352	0.354	-
1.9	24.84069	24.936	24.93561	-	0.384	0.382	-
2.0	24.53527	24.635	24.63524	23.7600	0.406	0.407	3.160

Table 4.3 The natural frequency (λ) for different aspect ratio and grid size (n) for CCSS plates.

Aspect ratio $\alpha = b/a$	Natural Frequency (λ) For Grid size n = 3	Natural Frequency (λ) For Grid size n = 5	Natural Frequency (λ) For Grid size n = 7	Natural Frequency (λ) For Grid size n = 9	Natural Frequency (λ) For Grid size n = 11	Natural Frequency (λ) For Grid size n = 13	Natural Frequency (λ) For Grid size n = 15	Natural Frequency (λ) For Grid size n = 17	Natural Frequency (λ) For Grid size n = 19	Natural Frequency (λ) For Grid size n = 21
1.0	26.20237	26.63259	26.80778	26.89367	26.94168	26.9711	26.99038	27.00369	27.01324	27.02033
1.1	23.95566	24.35108	24.51207	24.591	24.63512	24.66215	24.67987	24.6921	24.70087	24.70738
1.2	22.2895	22.66248	22.81424	22.88865	22.93024	22.95573	22.97244	22.98396	22.99223	22.99836
1.3	21.02467	21.38322	21.52896	21.60042	21.64037	21.66485	21.68089	21.69196	21.6999	21.70579
1.4	20.04478	20.39426	20.53606	20.60559	20.64446	20.66828	20.6839	20.69467	20.70239	20.70812
1.5	19.27194	19.61599	19.75524	19.82351	19.86168	19.88507	19.9004	19.91098	19.91856	19.92419
1.6	18.65261	18.99381	19.13144	19.19888	19.23695	19.2597	19.27485	19.2853	19.2928	19.29835
1.7	18.14914	18.48938	18.62601	18.69293	18.73034	18.75328	18.76831	18.77867	18.78611	18.79162
1.8	17.73452	18.07524	18.21131	18.2779	18.31513	18.33795	18.35291	18.36322	18.37062	18.3761
1.9	17.38896	17.73136	17.86717	17.93358	17.9707	17.99344	18.00835	18.01864	18.02601	18.03148
2.0	17.09779	17.44289	17.57867	17.64498	17.68203	17.70474	17.71962	17.72989	17.73725	17.74271

Table 4.4 Results of natural frequency (λ) for CCSS plate of present study and the results of previous studies with their percentage differences for different aspect ratios.

Aspect ratio $\alpha = b/a$	Present Study (λ_1) For grid size n=21	Njoku (2013) (λ_2)	Sakata et al (1996) (λ_3)	Chakraverty (2009) (λ_4)	Percentage Difference For (λ_2) & (λ_1)	Percentage Difference For (λ_3) & (λ_1)	Percentage Difference For (λ_4) & (λ_1)
1.0	27.02033	27.12846	26.867	27.055	0.4001	0.5675	0.1283
1.1	24.70738	24.8076	-	-	0.4056	-	-
1.2	22.99836	23.09477	-	-	0.4191	-	-
1.3	21.70579	21.80073	-	-	0.4374	-	-
1.4	20.70812	20.80073	-	-	0.4472	-	-
1.5	19.92419	20.01935	-	-	0.4776	-	-
1.6	19.29835	19.39417	-	-	0.4965	-	-
1.7	18.79162	18.88809	-	-	0.5134	-	-
1.8	18.3761	18.47313	-	-	0.5280	-	-
1.9	18.03148	18.12889	-	-	0.5402	-	-
2.0	17.74271	17.84034	17.770	-	0.5503	0.5503	-

Table 4.5 The natural frequency (λ) for different aspect ratio and grid size (n) for CSCS plates.

Aspect ratio $\alpha = b/a$	Natural Frequency (λ) For Grid size n = 3	Natural Frequency (λ) For Grid size n = 5	Natural Frequency (λ) For Grid size n = 7	Natural Frequency (λ) For Grid size n = 9	Natural Frequency (λ) For Grid size n = 11	Natural Frequency (λ) For Grid size n = 13	Natural Frequency (λ) For Grid size n = 15	Natural Frequency (λ) For Grid size n = 17	Natural Frequency (λ) For Grid size n = 19	Natural Frequency (λ) For Grid size n = 21
1.0	28.0212	28.4999	28.68858	28.78036	28.83149	28.86278	28.88327	28.8974	28.90755	28.91507
1.1	24.46694	24.8644	25.02354	25.10134	25.14479	25.1714	25.18884	25.20087	25.20951	25.21591
1.2	21.79777	22.13671	22.27474	22.34258	22.38056	22.40385	22.41912	22.42966	22.43723	22.44284
1.3	19.74984	20.04591	20.16863	20.22929	20.26333	20.28424	20.29796	20.30743	20.31423	20.31927
1.4	18.14973	18.41399	18.52549	18.58093	18.61211	18.63129	18.64389	18.65259	18.65883	18.66346
1.5	16.87975	17.12017	17.22338	17.27497	17.30407	17.32199	17.33377	17.34191	17.34775	17.35208
1.6	15.85781	16.08024	16.17727	16.22603	16.2536	16.27059	16.28177	16.28949	16.29504	16.29915
1.7	15.02535	15.23412	15.32652	15.37318	15.39962	15.41593	15.42667	15.43409	15.43942	15.44337
1.8	14.33976	14.53812	14.62704	14.67213	14.69772	14.71354	14.72396	14.73115	14.73633	14.74016
1.9	13.76946	13.95991	14.04618	14.0901	14.11507	14.13052	14.14069	14.14773	14.15279	14.15653
2.0	13.29074	13.47518	13.55946	13.60249	13.62700	13.64217	13.65216	13.65908	13.66405	13.66773

Table 4.6 Results of natural frequency (λ) for CSCS plates of present study and results of previous studies with their percentage difference for different aspect ratios.

Aspect ratio $\alpha = b/a$	Present Study (λ_1) For grid size n=21	Njoku (2013) (λ_2)	Goman (1982) (λ_3)	Chakraborty (2009) (λ_4)	Percentage Difference For (λ_2) & (λ_1)	Percentage Difference For (λ_3) & (λ_1)	Percentage Difference For (λ_4) & (λ_1)
1.0	28.91507	28.95663	28.95	28.950	0.1437	0.1208	0.1208
1.1	25.21591	25.25798	-	-	0.1668	-	-
1.2	22.44284	22.48099	-	-	0.1699	-	-
1.3	20.31927	20.35661	-	-	0.1838	-	-
1.4	18.66346	18.7019	-	-	0.2059	-	-
1.5	17.35208	17.39274	17.37	-	0.2343	0.1033	-
1.6	16.29915	16.34263	-	-	0.2667	-	-
1.7	15.44337	15.48987	-	-	0.3011	-	-
1.8	14.74016	14.78968	-	-	0.3359	-	-
1.9	14.15653	14.20892	-	-	0.3701	-	-
2.0	13.66773	13.72275	13.69	-	0.4025	0.1629	-

Table 4.7 The natural frequency (λ) for different aspect ratio and grid size (n) for CSSS plates.

Aspect ratio $\alpha = b/a$	Natural Frequency (λ) For Grid size n = 3	Natural Frequency (λ) For Grid size n = 5	Natural Frequency (λ) For Grid size n = 7	Natural Frequency (λ) For Grid size n = 9	Natural Frequency (λ) For Grid size n = 11	Natural Frequency (λ) For Grid size n = 13	Natural Frequency (λ) For Grid size n = 15	Natural Frequency (λ) For Grid size n = 17	Natural Frequency (λ) For Grid size n = 19	Natural Frequency (λ) For Grid size n = 21
1.0	23.00074	23.33312	23.46452	23.52829	23.56376	23.58543	23.59961	23.60938	23.61639	23.62158
1.1	20.51997	20.80805	20.92285	20.9787	21.0098	21.02882	21.04126	21.04983	21.05598	21.06053
1.2	18.64727	18.90537	19.00896	19.05948	19.08764	19.10486	19.11614	19.1239	19.12947	19.13359
1.3	17.20151	17.43921	17.53518	17.58208	17.60825	17.62426	17.63474	17.64197	17.64714	17.65097
1.4	16.06391	16.28768	16.37848	16.42293	16.44774	16.46293	16.47288	16.47973	16.48464	16.48828
1.5	15.15394	15.36828	15.45557	15.49836	15.52226	15.5369	15.54649	15.55309	15.55782	15.56132
1.6	14.41556	14.62358	14.70852	14.7502	14.77349	14.78776	14.79711	14.80354	14.80816	14.81156
1.7	13.80877	14.0127	14.09608	14.13703	14.15993	14.17396	14.18314	14.18947	14.1940	14.19735
1.8	13.30448	13.5059	13.58829	13.62877	13.65141	13.66529	13.67437	13.68063	13.68512	13.68843
1.9	12.88109	13.08117	13.16295	13.20315	13.22564	13.23943	13.24846	13.25467	13.25913	13.26242
2.0	12.52237	12.72196	12.80342	12.84346	12.86587	12.8796	12.8886	12.89479	12.89923	12.90251

Table 4.8 Results of natural frequency (λ) for CSSS plates of present study and results of previous studies with their percentage difference for different aspect ratios.

Aspect ratio $\alpha = b/a$	Present Study (λ_1) For grid size n=21	Njoku (2013) (λ_2)	Gorman (1982) (λ_3)	Chakraverty (2009) (λ_4)	Percentage Difference For (λ_2) & (λ_1)	Percentage Difference For (λ_3) & (λ_1)	Percentage Difference For (λ_4) & (λ_1)
1.0	23.62158	23.67982	23.65	23.646	0.2465	0.1203	0.1034
1.1	21.06053	21.1211	-	-	0.2876	-	-
1.2	19.13359	19.19747	19.42	-	0.3388	1.4968	-
1.3	17.65097	17.7184	-	-	0.3820	-	-
1.4	16.48828	16.55908	-	-	0.4294	-	-
1.5	15.56132	15.63509	15.58	-	0.4741	0.1200	-
1.6	14.81156	14.27555	-	-	3.6188	-	-
1.7	14.19735	14.27555	-	-	0.5505	-	-
1.8	13.68843	13.76808	-	-	0.5818	-	-
1.9	13.26242	13.34306	-	-	0.6080	-	-
2.0	12.90251	12.98373	12.92	-	0.6295	0.1355	-

Table 4.9 The natural frequency (λ) for different aspect ratio and grid size (n) for CCCS plates.

Aspect ratio $\alpha = b/a$	Natural Frequency (λ) For Grid size n = 3	Natural Frequency (λ) For Grid size n = 5	Natural Frequency (λ) For Grid size n = 7	Natural Frequency (λ) For Grid size n = 9	Natural Frequency (λ) For Grid size n = 11	Natural Frequency (λ) For Grid size n = 13	Natural Frequency (λ) For Grid size n = 15	Natural Frequency (λ) For Grid size n = 17	Natural Frequency (λ) For Grid size n = 19	Natural Frequency (λ) For Grid size n = 21
1.0	30.71895	31.26736	31.4972	31.6111	31.67511	31.71445	31.74029	31.75814	31.77098	31.78051
1.1	27.43187	27.90753	28.10958	28.2101	28.26669	28.3015	28.32437	28.34018	28.35155	28.35999
1.2	25.008	25.43231	25.61491	25.70612	25.75755	25.78923	25.81005	25.82445	25.8348	25.84248
1.3	23.18205	23.56964	23.73843	23.82308	23.8709	23.90037	23.91976	23.93317	23.94281	23.94997
1.4	21.78061	22.1417	22.30058	22.38055	22.4258	22.45371	22.47209	22.4848	22.49394	22.50073
1.5	20.68695	21.02882	21.18051	21.2571	21.30051	21.32731	21.34496	21.35717	21.36596	21.37248
1.6	19.82053	20.14852	20.29494	20.36907	20.4113	20.43713	20.45425	20.46611	20.47464	20.48097
1.7	19.12465	19.44273	19.58527	19.65757	19.69866	19.72406	19.7408	19.75239	19.76073	19.76693
1.8	18.55863	18.86987	19.00951	19.08046	19.12081	19.14577	19.16223	19.17363	19.18183	19.18792
1.9	18.0928	18.39961	18.53712	18.60704	18.64683	18.67147	18.68771	18.69869	18.70706	18.71307
2.0	17.7052	18.00958	18.1455	18.21464	18.25401	18.27839	18.29447	18.30561	18.31363	18.31958

Table 4.10 Results of natural frequency (λ) for CCCS plates of present study and results of previous studies with their percentage difference for different aspect ratios.

Aspect ratio $\alpha = b/a$	Present Study (λ_1) For grid size $n=21$	Njoku (2013) (λ_2)	Chakra- verty (2009) (λ_3)	Percentage Difference For (λ_2) & (λ_1)	Percentage Difference For (λ_3) & (λ_1)
1.0	31.78051	31.86803	31.827	0.2753	0.1461
1.1	28.35999	28.43333	-	0.2586	-
1.2	25.84248	25.90856	-	0.2551	-
1.3	23.94997	24.01317	-	0.2632	-
1.4	22.50073	22.56374	-	0.2793	-
1.5	21.37248	21.43689	-	0.3005	-
1.6	20.48097	20.54761	-	0.3243	-
1.7	19.76693	19.83618	-	0.3491	-
1.8	19.18792	19.25986	-	0.3735	-
1.9	18.71307	18.78759	-	0.3966	-
2.0	18.31958	18.39649	-	0.4394	-

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The following conclusions could be drawn based on the results of this study.

The general flexural element stiffness matrix of a thin rectangular plate with the inertia matrix is satisfactory for all the boundary condition used in this study. This means that for each chosen boundary condition, the same stiffness matrix formulated can be used to obtain the required approximate result.

The MATLAB program formed for different boundary conditions using finite element method is fast, efficient and easy to understand. It is fast because it can perform analysis of plate within a short time. Yielding results in a short time with minimal error makes the program efficient.

The finite element method used for the free vibration analysis of rectangular thin plate yielded satisfactory approximate fundamental natural frequencies for the plates.

The fundamental natural frequencies obtained by previous research works that used different methods of analysis are very close to those obtained in the present study. That is the difference in value of results obtained in this study with other approximate methods are negligible.

5.2 Recommendations

- (i) It is recommended that the method used in this research in deriving the stiffness and inertia matrix of a thin rectangular plate under free vibration should be applied to a thick plate subjected to forced vibration.

- (ii) It is recommended that the free vibration analysis and finite element method used in this study should be applied to rectangular plates with central cutout, on thin circular plates with multiple opening and plates with various shapes like skew, elliptical, hexagonal etc.
- (iii) Other programming languages like Java, FORTRAN, C++, and Python can be recommended for future research work on free vibration and forced vibration analysis of rectangular plates using finite element method.
- (iv) Further application of degrees of freedom to 16 deformable terms or more on rectangular plates can be recommended for the purpose of accuracy.

5.3 Contribution to Knowledge

This research work, free vibration of a thin rectangular plate using finite element method has contributed to the following knowledge.

- (i) Successfully formulated the General Flexural Element Stiffness Matrix as expressed in the Equation (3.13d).

$$\begin{aligned}
\Pi = & \left[N_i^{-1} \right]^T \frac{D}{2a^4} \int_0^1 \int_0^1 \left[[N]^T [N] \right]_R dRdQ. [w_i] \left[N_i^{-1} \right] \\
& + \left[N_i^{-1} \right]^T \frac{D}{a^4 \alpha^2} \int_0^1 \int_0^1 \left[[N]^T [N] \right]_{RQ} dRdQ. [w_i] \left[N_i^{-1} \right] \\
& + \left[N_i^{-1} \right]^T \frac{D}{2a^4 \alpha^4} \int_0^1 \int_0^1 \left[[N]^T [N] \right]_Q dRdQ. [w_i] \left[N_i^{-1} \right] \\
& - \left[N_i^{-1} \right]^T \frac{m\lambda^2}{2} \int \int \left[[N]^T [N] \right] dRdQ. [w_i] \left[N_i^{-1} \right] \tag{3.13d}
\end{aligned}$$

and the Inertia Matrix as expressed in Equation (3.41).

$$K_i = \left[N_i \right]^{-1T} m \dots \lambda^2 \int_0^1 \int_0^1 \left[[N]^T [N] \right] \partial R \partial Q. \left[N_i \right]^{-1} \tag{3.41}$$

- (ii) A MATLAB program developed using the stiffness matrix and inertia matrix in Appendix A to D in obtaining natural frequencies of thin plate free vibration problems.

REFERENCES

- Aginam, C.H., Chidolue, C.A., and Ezeagu, C.A. (2012). Application of Direct Variational Method in Analysis of Isotropic Thin Rectangular Plates. *ARPJ Journal of Engineering and Applied Sciences*, 7(9), 1128-1129.
- Ali Reza, P., Jalil, E., Majid, S. and Habibolahiyani, H. (2011). The Vibration of Thin Plates by using Modal Analysis. *Journal of World Academy of Science, Engineering and Technology*. 59, 2880.
- Ali, E., and Nahit, K. (2011). A New Approach of Improved Finite Difference Scheme on Plate Bending Analysis. *Scientific Research and Essays*, 6 (1), 6-17.
- Ali, Nikkhoo, and Rofooei, F.R. (2012). Parametric Study of the Dynamic Response of Thin Rectangular Plates Traversed by a Moving Mass. *Acta Mechanica*, 223(1), 15-27.
- Ali, Nikkhoo. (2013). Investigating the Behaviour of Smart Thin Beams with Piezoelectric Actuators Under Dynamic Loads. *Journal Mechanical System and Signal Processing*, 45, 1-18.
- Ali, N., Mohsen, E.H., Saeed, E.A. and Javad, V.A. (2013). Vibration of Thin Rectangular Plate Subjected to Series of Moving Inertial Loads. *Mechanics Research Communications*. 1-9.
- Amini, M.H., Soleimani M. and Rastgoo, A. (2009). Three-Dimensional Free Vibration Analysis of Functionally Graded Material Plates Resting on an Elastic Foundation. *Smart Materials and Structures*, 18, 1-9.
- Amitabha (2015). Dynamics of Machines: Approximate Method of Vibration analysis Rayleigh's Method. *Department of Mechanical Engineering Indian Institute of Technology, Kanpur*. URL: textofvideo.nptel.iitm.ac.in/112104114/lec41.pdf
- Ansgar, N. (2009). Teaching and Learning Structural Analysis with MATLAB. *Journal American Society for Engineering Education*, 167, 2.

- Asim Kumar, M. (2008). *Finite Element Large Amplitude Free Flexural Vibration Analysis of Isotropic Plates*. Thesis, Master of Technology in Civil Engineering, National Institute of Technology Rourkela, 1-2.
- Atekin, M. (2010). Bending of Orthotropic Super-Elliptical Plates on Intermediate Point Supports. *Ocean Engineering*, 37, 1048-1060.
- Chakraverty, S. (2009). *Vibration of Plates (1st Ed)*. Taylor and Francis Group, LLC.
- Dolicanin, C.B., Nikolic, V.B. and Dolicanin, D.C. (2010). Application of Finite Difference Method to Study of the Phenomenon in the Theory of Thin Plates. *Appl. Math. Inform. and Mech*, 2 (1), 29-43.
- Ezeh, J.C., Ibearugbulem, O.M. and Onyechere, C.I. (2013). Free Vibration Analysis of Thin Rectangular Flat Plates Using Ordinary Finite Difference Method. *Journal Natural and Applied Science*, 4, 187-192.
- Ezeh, J.C., Njoku, K.O., Ibearugbulem, O.M., Ettu, L. O. and Anyaogu, L. (2013). Free Vibration Analysis of Thin Rectangular Isotropic CCCC Plate Using Taylor Series Formulated Shape Function in Galerkin's Method. *Journal Academic Research International*, 4 (4), 126-132.
- Ezeh, J.C., Ibearugbulem, O.M. and Onyechere, C.I. (2013). Pure Bending Analysis of Thin Rectangular Flat Plates Using Ordinary Finite Difference Method. *International Journal of Emerging Technology and Advanced Engineering*, 3(3), 20-23.
- Ezeh, J.C., Ibearugbulem, O.M., Opara, H.E., and Oguaghamba. (2014). Galerkin's Indirect Variational Method in Elastic Stability Analysis of all Edges Clamped Thin Rectangular Flat Plates. *International Journal of Research in Engineering and Technology*, 3, 674-679.

- Gergely, M., Laszlo G.V., Gyorgy, S. and Laszlo, D., (2012). Finite Element Analysis of Laminated Structural Glass Plates with Polyvinyl Butyral (PVB) interlayer. *Periodica Polytechnica*, 56 (1), 35-42.
- Ghaffarzadeh, H. and Nikkar, A. (2013). Explicit Solution to the Large Deformation of a Cantilever Beam Under Point Load at the Free Tip Using Variational Iteration Method –II. *Journal of Mechanical Science and Technology*, 27 (11), 3433-3438.
- Gorman, D.J. (1982). Free Vibration Analysis of Rectangular Plates. North Holland. *Elsevier*.
- Gorman, D.J. (2006). Exact Solutions for Free In-plane Vibration of Rectangular Plates with Two Opposite Edges Simply Supported. *Journal of Sound Vibration*, 294, 131-161.
- Guo, M., Harik, I.E. and Wei-Xin. (2002). Free Vibration Analysis of Stiffened Laminated Plates using Layered Finite Element. *Journal for Structural Engineering Mechanics*, 14(3), 245-262.
- Gupta, A.K., Anupam, K. and Dharam, V.G. (2009). Free Vibration of Clamped Visco-Elastic Rectangular Plate having Bi-Direction Exponentially Thickness Variations. *Journal of Theoretical and Applied Mechanics*, 47 (2), 457-471.
- Hashemi, S.H. and Arsanjani, M. (2005). Exact Characteristic Equations for some of Classical Boundary Conditions of Vibrating Moderately Thick Rectangular Plates. *International Journal Solid Structures*, 42, 819-853.
- Heinz, A. (2010). *A Short Course on Boundary Element Methods*. Institute for Angewandte Mechanik. 1-84.
- Hosseini-Hashemi, Sh., Heydar Roohi, Gh. and Hossein Rokni D.T. Exact Free Vibration Study of Rectangular Mindlin Plates with All-Over Part-Through Open Cracks. *Journal Computers and Structures*, 88 (17-18), 1015-1032.

- Huang, Y.M and Hung, S.C. (2011). Analytical Study of an Active Piezoelectric Absorber on Vibration Attenuation of a Plate. *Journal of Sound and Vibration*, 330 (3), 361-373.
- Huang, M., Ma, X.Q., Sakiyama, T., Matsuda, H. and Morita C. (2007). Free Vibration of Rectangular Plates with Variable Thickness and Point Supports. *Journal of Sound and Vibration*, 300, 435-452.
- Jaen, K. (2005). *Numerical Methods in Engineering with MATLAB*. UK: Cambridge University Press. 1.
- Jodeiri, H. and Imani, V.H. (2015). An Efficient Analytical Method for Vibration Problems. *ARPJ Journal of Engineering and Applied Sciences*, 10 (2) 740-743.
- Jiu, H. W., Liu, A.Q. and Chen, H.L. (2007). Exact Solutions for Free- Vibration Analysis of Rectangular Plates Using Bessel Functions. *Journal of Applied Mechanics ASME*, 74, 1247-1251.
- John, D. (2004). *A dictionary of computing*. Published by Oxford University Press.
- Kapil, (2012). Analysis of Simple Supported Plate for Active Vibration Control with Piezoelectric Sensors and Actuators. *M. Tech, Mechanical Engineering*, 1-25.
- Kharde, S.B., Mahale, A.K., Bhosale, K.C. and Thorat, S.R. (2013). Flexural Vibration of Thick Isotropic Plates by Using Exponential Shear Deformation Theory. *International Journal of Emerging Technology and Advanced Engineering*, 3, 369-374.
- Latheswary, S., Valsarajan, K.V., and Sadasiva, R. (2004). Dynamic Response of Moderately Thick Composite Plates. *Journal of Sound and Vibration*, 270 (1-2) 417-426.
- Leissa, A.W. (1969). *Vibration of Plates*. Scientific and Technical information Division office of Technology Utilization. Ohio State University of Columbus, Ohio.
- Leissa, A.W. (2005). The Historical Bases of the Rayleigh and Ritz Method. *Journal of Sound and Vibration*, 278, 961-978.

- Leissa, A.W., Qatu, M.S. (2011). *Vibrations of Continuous Systems*. Mc Graw-Hill Professional Publication: Blacklick.
- Li, W.L., Zhang, X., Du, J. and Liu, Z. (2009). An Exact Series Solution for the Transverse Vibration of Rectangular Plates with General Elastic Boundary Supports. *Journal of Sound and Vibration*, 321, 254-269.
- Li, Q., Iu, V.P. and Kou K.p. (2009). Three- Dimensional Vibration Analysis of Functionally Graded Material Plates in Thermal Environment. *Journal Sound Vibration*, 324, 733-750.
- Li, Q., Iu, V.P. and Kou K.p. (2008). Three- Dimensional Vibration Analysis of Functionally Graded Material Sandwich Plates. *Journal Sound Vibration*, 311, 498-515.
- Lin, J.C. and Nien, M.H. (2007). Adaptive Modeling and Shape Control of Laminated Plates using Piezoelectric Actuators. *Journal of Materials Processing Technology*, 189, 231-236.
- Liu, B. and Xing, Y.F. (2011). Comprehensive Exact Solutions for Free In-Plane Vibrations of Orthotropic Rectangular Plates. *Eur J Mech A- Solid*, 30, 383-395.
- Lorenzo, D. (2011). On the Use of the Trigonometric Ritz Method for General Vibration Analysis of Rectangular Kirchhoff Plates. *Thin-Walled Structures*, 49, 129-144.
- Matsunaga, H. (2002). Vibration of Cross-ply Laminated Composite Plates Subjected to Initial In-plane Stress. *Thin-Walled Structures*, 1925-1944.
- Matsunaga, H. (2000). Vibration and Stability of Cross-Ply Laminated Composite Plates According to a Global Higher-Order Plate Theory, *Composite Structures 2002*, 231-244.
- Michal. G. and Tomasz, J. (2007). The Analysis of Internally Supported Thin Plates by the Boundary Element Method. *Part 3-Initial Stability Analysis*, 51-60.

- Neffati, M.W. and Abobaker, A.K. (2012). Free Vibration Analysis of Rectangular Plates Using Galerkin-Based Finite Element Method. *Vol.2 (4),59*.
- Nguyen-Xuan, H., Rabczuk, T., Nguyen-Thanh, N., Nguyen-Thoi, T. and Bordas, S. (2010), A Node-Base Smoothed Finite Element Method with Stabilized Discrete Shear Gap Technique for Analysis of Reissner-Mindlin Plates. *Computational Mechanics, 46*, 679-701.
- Nikolas, T. (2014). Educational Examples in Structural Acoustics Using the Finite Element Method. 29-30.
- Njoku, K.O. (2013). *Dynamic Analysis of Thin Rectangular Flat Isotropic Plates Using Galerkin's Method*. M. Eng. Thesis Federal University of Technology Owerri.
- Onwuka, D.O., Ibearugbulem, O.M., Abamara, A.C., Njoku, C.F. and Agbo, S.I. (2016). Free Vibration Analysis of an Around-Clamped Rectangular Thin Orthotropic Plate Using Taylor-Mclaurin Shape function. *American Journal of Engineering Research,5 (5)*, 190-197.
- Pakar, I. and Bayat, M. (2012). Analytical Study on the Non- Linear Vibration of Euler-Bernoulli Beams. *Journal of Vibroengineering,14 (1)*, 3433-3438.
- Paquin, S. and St-Amant Y. (2010). Improving the Performance of a Piezoelectric Energy Harvester Using a Variable Thickness Beam. *Smart Mater Structures, 19*, 10502.
- Patiphan, C.W. (2015). *Analytical Methods of Vibration Problems of Plates*. Vibool Choke Ltd, Bangkok, Thailand. 1705-1708.
- Phadikar, J.K., and Pradhan, S.C. (2010). Variational Formulation and Finite Element Analysis for Non-Local Elastic Nanobeams and Nanoplates. *Computational Material Science, 49*, 492-499.
- Polyanin, A.D. and Manzhirov, A.V. (2008). *Handbook of integral Equations, 2nd Edition.*, Chapman and Hall.

- Pradhan, S.C., (2012). Nonlocal Finite Element Analysis and Small-Scale Effects of CNTS with Timoshenko Beam Theory. *Finite Elements in Analysis and Design*, 50, 8-20.
- Ramu, I., and Mohanty, S.C. (2012). Study on Free Vibration Analysis of Rectangular Plate Structures Using Finite Element Method. *Procedia Eng.*, 38, 2758-2766.
- Ray, W.C. and Joseph, P. (2003). *Dynamics of Structures* (3rd Edition), USA: Computers and Structures Inc.
- Reddy, J.N. (2007). *Theory and Analysis of Elastic Plates and Shells*. CRC Press.
- Reddy, J.N. (1993). *An Introduction to the Finite Element Method* (2nd Edition). New York: Mc Graw Hill, 18-64.
- Rofooei, F.R. and Nikkhoo, A. (2009). Application of Active Piezoelectric Patches in Controlling the Dynamic Response of Thin Rectangular Plate Under a Moving Mass. *International Journal of Solids and Structures*, 46 (11-12), 2429-2443.
- Rudolph, S. (2004). *Theories and Applications of Plate Analysis*. (Classical, Numerical and Engineering Methods). 10, 364-366.
- Sakata, T., Takahashi, K. and Bhat, R.B. (1996). Natural Frequencies of Orthotropic Rectangular Plates Obtained by Iterative Reduction of the Partial Differential Equation. *Journal of sound Vibration*, 18, 89-101.
- Senjanovic, V. N. and Tomic, M. (2013). An Advanced Theory of Moderately Thick Plate Vibrations. *Journal of sound Vibration*, 332, 1868-1880.
- Shimpi, R.P. and Patel, H.G. (2006). Free Vibration of Plate Using Variable Refined Plate Theory. *Journal Sound Vibration*, 296, 979-999.
- Sirinivas, S., Rao A.K. and Rao. C.V. (1970). An exact analysis for Vibration of Simple Supported Homogeneous and Laminated Thick Rectangular Plates. *Journal Sound Vibration*, 12, 187-199.

- Tavakolpour, A.R., Mat Darus, I.Z., Mailah, M. and Tokhi, M.O. (2010). Genetic Algorithm-Based Identification of a Rectangular Flexible Plate System for Active Vibration Control. *International journal of Engineering and Applied Artificial intelligence*, 23, 1388-1397.
- Thai, H.T. and Kim, S.E. (2012). Levy-Type Solution for Free Vibration Analysis of Orthotropic Plates Based on Two Variable Refined Theory, *Applied Mathematical Model*, 36, 3870-3882.
- Thonchangya, N. (2001). Analysis of the Bending of a Uniformly Loaded Rectangular Plate with Variable Width Conner Supports. *International Journal of Computational Engineering Science*, 2, 31-42.
- Thongperm, Y., Patiphan, C., Jakarin, V. and Yos, S. (2015). Accurate Approximate and Analytical Method for Vibration and Bending Problems of Plates: A Literature Survey. *Applied Mathematical Sciences*, 9 (35), 1697-1719.
- Ugurlu, B., Kutlu, A., Ergin A., and Omurtag, M.H. Dynamics of a Rectangular Plate Resting on an Elastic Foundation and Partial in Contact with a Quiescent Fluid. *Journal Sound Vibration*, 317, 308-328.
- Uzale, E. and Sakman, L.E., (2010). Dynamic Response of a Circular Plate to a Moving Load. *Acta Mechanica*, 210 (3-4) 351-359.
- Vanam, B.C.L., Rajyalakshmi, M. and Inala, R. (2012). Static Analysis of an Isotropic Rectangular Plate Using Finite Element Analysis (FEA). *Journal of Mechanical Engineering Research*, 4 (4), 148-142.
- Vaseghi- Amiri, J., Nikkhoo, A., Davoodi, M.R. and Ebrahimzadeh H. M. (2013). Vibration Analysis of a Mindlin Elastic Plate Under a Moving Mass Excitation by Eigenfunction Expansion Method. *Thin Walled Structures*, 62, 53-64.

- Venstel, E. and Krauthammer, T. (2001). *Thin Plates and Shell (Theory, Analysis and Applications)*. New York Press.
- Wang, Y. and Wang, Z. (2008). Transverse Vibration of Viscoelastic Rectangular Plate with Linearly Varying Thickness and Multiple Cracks. *Journal of Sound and Vibration*, 318 (4-5), 1005-1023.
- Wu, W., Shu, C., Xiang, Y. and Wang, C. (2010). Free Vibration and Buckling Analysis of Highly Skewed Plates by Least Squares Based Finite Difference Method. *International Journal of Structural Stability and Dynamics*, 10, 225-250.
- Xiang, Y. and Wei, G.W. (2004). Exact Solution for Buckling and Vibration of Stepped Rectangular Midlin Plates. *International Journal Solid Structures*, 41, 279-294.
- Xing, Y.F. and Liu, B. (2009). Exact Solutions for Free In-plane Vibration of Rectangular Plates. *International Journal of Mechanical Science*, 51, 245-255.
- Xing, Y.F. and Liu, B. (2009). New Exact Solutions for Free Vibrations of Thin Orthotropic Rectangular Plates. *Composite Structures*, 89, 567-574.
- Xing, Y.F. and Liu, B. (2011). Comprehensive Exact Solutions for Free In-plane Vibrations of Orthotropic Rectangular Plates. *European Journal of Mechanics - A/Solids*, 30 (3), 383-395.
- Xing, Y.F., Xu, T.F. and Liu, B. (2013). Recent Research Advances of Exact Solutions for Free Vibration of Plates and Shells. *11th International Conference on Vibration Problems*, 5.
- Young, W.K. and Hyochoong, B. (1997). *The Finite Element Method Using MATLAB*. CRC Mechanical Engineering Series. 2.
- Zarfam, R., Khaloo, A.R. and Nikkhoo, A. (2013). On the Response Spectrum of Euler-Bernolli Beams with a Moving Mass and Horizontal Support Excitation. *Mechanics Research Communications*, 47, 77-83.

Zhang, Y.X. and Yang, C.H. (2008). Recent Developments in Finite Element Analysis for Laminated Composite Plates. *Composite structures*, 87 (1), 148.

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

%ReDim kx(12, 12), kxy(12, 12), ky(12, 12), k(12, 12), ki(12, 12)
for x = 1 : 12
for y = 1 : 12
k(x, y) = 0; kx(x, y) = 0; kxy(x, y) = 0; ky(x, y) = 0;
end
end
af = input('WHAT IS THE aspect ratio, b/a?'); af = af * 1;
kx(1, 1) = 4; kx(1, 2) = 2; kx(1, 3) = 0; kx(1, 4) = -4; kx(1, 5) = 2; kx(1, 6) = 0; kx(1, 7) = -2; kx(1, 8) = 1; kx(1, 9) = 0; kx(1, 10) = 2; kx(1, 11) = 1; kx(1, 12) = 0;
kx(2, 1) = 2; kx(2, 2) = 1.3333333; kx(2, 3) = 0; kx(2, 4) = -2; kx(2, 5) = 0.66666667; kx(2, 6) = 0; kx(2, 7) = -1; kx(2, 8) = 0.3333333; kx(2, 9) = 0; kx(2, 10) = 1; kx(2, 11) = 0.66666667; kx(2, 12) = 0;
kx(3, 1) = 0; kx(3, 2) = 0; kx(3, 3) = 0; kx(3, 4) = 0; kx(3, 5) = 0; kx(3, 6) = 0; kx(3, 7) = 0; kx(3, 8) = 0; kx(3, 9) = 0; kx(3, 10) = 0; kx(3, 11) = 0; kx(3, 12) = 0;
kx(4, 1) = -4; kx(4, 2) = -2; kx(4, 3) = 0; kx(4, 4) = 4; kx(4, 5) = -2; kx(4, 6) = 0; kx(4, 7) = 2; kx(4, 8) = -1; kx(4, 9) = 0; kx(4, 10) = -2; kx(4, 11) = -1; kx(4, 12) = 0;
kx(5, 1) = 2; kx(5, 2) = 0.66666667; kx(5, 3) = 0; kx(5, 4) = -2; kx(5, 5) = 1.3333333; kx(5, 6) = 0; kx(5, 7) = -1; kx(5, 8) = 0.66666667; kx(5, 9) = 0; kx(5, 10) = 1; kx(5, 11) = 0.3333333; kx(5, 12) = 0;
kx(6, 1) = 0; kx(6, 2) = 0; kx(6, 3) = 0; kx(6, 4) = 0; kx(6, 5) = 0; kx(6, 6) = 0; kx(6, 7) = 0; kx(6, 8) = 0; kx(6, 9) = 0; kx(6, 10) = 0; kx(6, 11) = 0; kx(6, 12) = 0;
kx(7, 1) = -2; kx(7, 2) = -1; kx(7, 3) = 0; kx(7, 4) = 2; kx(7, 5) = -1; kx(7, 6) = 0; kx(7, 7) = 4; kx(7, 8) = -2; kx(7, 9) = 0; kx(7, 10) = -4; kx(7, 11) = -2; kx(7, 12) = 0;
kx(8, 1) = 1; kx(8, 2) = 0.3333333; kx(8, 3) = 0; kx(8, 4) = -1; kx(8, 5) = 0.66666667; kx(8, 6) = 0; kx(8, 7) = -2; kx(8, 8) = 1.3333333; kx(8, 9) = 0; kx(8, 10) = 2; kx(8, 11) = 0.66666667; kx(8, 12) = 0;
kx(9, 1) = 0; kx(9, 2) = 0; kx(9, 3) = 0; kx(9, 4) = 0; kx(9, 5) = 0; kx(9, 6) = 0; kx(9, 7) = 0; kx(9, 8) = 0; kx(9, 9) = 0; kx(9, 10) = 0; kx(9, 11) = 0; kx(9, 12) = 0;
kx(10, 1) = 2; kx(10, 2) = 1; kx(10, 3) = 0; kx(10, 4) = -2; kx(10, 5) = 1; kx(10, 6) = 0; kx(10, 7) = -4; kx(10, 8) = 2; kx(10, 9) = 0; kx(10, 10) = 4; kx(10, 11) = 2; kx(10, 12) = 0;
kx(11, 1) = 1; kx(11, 2) = 0.66666667; kx(11, 3) = 0; kx(11, 4) = -1; kx(11, 5) = 0.3333333; kx(11, 6) = 0; kx(11, 7) = -2; kx(11, 8) = 0.66666667; kx(11, 9) = 0; kx(11, 10) = 2; kx(11, 11) = 1.3333333; kx(11, 12) = 0;
kx(12, 1) = 0; kx(12, 2) = 0; kx(12, 3) = 0; kx(12, 4) = 0; kx(12, 5) = 0; kx(12, 6) = 0; kx(12, 7) = 0; kx(12, 8) = 0; kx(12, 9) = 0; kx(12, 10) = 0; kx(12, 11) = 0; kx(12, 12) = 0;

ky(1, 1) = 4; ky(1, 2) = 0; ky(1, 3) = 2; ky(1, 4) = 2; ky(1, 5) = 0; ky(1, 6) = 1; ky(1, 7) = -2; ky(1, 8) = 0; ky(1, 9) = 1; ky(1, 10) = -4; ky(1, 11) = 0; ky(1, 12) = 2;
ky(2, 1) = 0; ky(2, 2) = 0; ky(2, 3) = 0; ky(2, 4) = 0; ky(2, 5) = 0; ky(2, 6) = 0; ky(2, 7) = 0; ky(2, 8) = 0; ky(2, 9) = 0; ky(2, 10) = 0; ky(2, 11) = 0; ky(2, 12) = 0;
ky(3, 1) = 2; ky(3, 2) = 0; ky(3, 3) = 1.3333333; ky(3, 4) = 1; ky(3, 5) = 0; ky(3, 6) = 0.66666667; ky(3, 7) = -1; ky(3, 8) = 0; ky(3, 9) = 0.3333333; ky(3, 10) = -2; ky(3, 11) = 0; ky(3, 12) = 0.66666667;
ky(4, 1) = 2; ky(4, 2) = 0; ky(4, 3) = 1; ky(4, 4) = 4; ky(4, 5) = 0; ky(4, 6) = 2; ky(4, 7) = -4; ky(4, 8) = 0; ky(4, 9) = 2; ky(4, 10) = -2; ky(4, 11) = 0; ky(4, 12) = 1;
ky(5, 1) = 0; ky(5, 2) = 0; ky(5, 3) = 0; ky(5, 4) = 0; ky(5, 5) = 0; ky(5, 6) = 0; ky(5, 7) = 0; ky(5, 8) = 0; ky(5, 9) = 0; ky(5, 10) = 0; ky(5, 11) = 0; ky(5, 12) = 0;
ky(6, 1) = 1; ky(6, 2) = 0; ky(6, 3) = 0.66666667; ky(6, 4) = 2; ky(6, 5) = 0; ky(6, 6) = 1.3333333; ky(6, 7) = -2; ky(6, 8) = 0; ky(6, 9) = 0.66666667; ky(6, 10) = -1; ky(6, 11) = 0; ky(6, 12) = 0.3333333;
ky(7, 1) = -2; ky(7, 2) = 0; ky(7, 3) = -1; ky(7, 4) = -4; ky(7, 5) = 0; ky(7, 6) = -2; ky(7, 7) = 4; ky(7, 8) = 0; ky(7, 9) = -2; ky(7, 10) = 2; ky(7, 11) = 0; ky(7, 12) = -1;
ky(8, 1) = 0; ky(8, 2) = 0; ky(8, 3) = 0; ky(8, 4) = 0; ky(8, 5) = 0; ky(8, 6) = 0; ky(8, 7) = 0; ky(8, 8) = 0; ky(8, 9) = 0; ky(8, 10) = 0; ky(8, 11) = 0; ky(8, 12) = 0;
ky(9, 1) = 1; ky(9, 2) = 0; ky(9, 3) = 0.3333333; ky(9, 4) = 2; ky(9, 5) = 0; ky(9, 6) = 0.66666667; ky(9, 7) = -2; ky(9, 8) = 0; ky(9, 9) = 1.3333333; ky(9, 10) = -1; ky(9, 11) = 0; ky(9, 12) = 0.66666667;
ky(10, 1) = -4; ky(10, 2) = 0; ky(10, 3) = -2; ky(10, 4) = -2; ky(10, 5) = 0; ky(10, 6) = -1; ky(10, 7) = 2; ky(10, 8) = 0; ky(10, 9) = -1; ky(10, 10) = 4; ky(10, 11) = 0; ky(10, 12) = -2;

```

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```
ky(11, 1) = 0; ky(11, 2) = 0; ky(11, 3) = 0; ky(11, 4) = 0; ky(11, 5) = 0; ky(11, 6) = 0; ky(11, 7) = 0; ky(11, 8) = 0; ky(11, 9) = 0; ky(11, 10) = 0; ky(11, 11) = 0; ky(11, 12) = 0;
ky(12, 1) = 2; ky(12, 2) = 0; ky(12, 3) = 0.66666667; ky(12, 4) = 1; ky(12, 5) = 0; ky(12, 6) = 0.33333333; ky(12, 7) = -1; ky(12, 8) = 0; ky(12, 9) = 0.66666667; ky(12, 10) = -2; ky(12, 11) = 0; ky(12, 12) = 1.33333333;

kxy(1, 1) = 2.8; kxy(1, 2) = 0.2; kxy(1, 3) = 0.2; kxy(1, 4) = -2.8; kxy(1, 5) = 0.2; kxy(1, 6) = -0.2; kxy(1, 7) = 2.8; kxy(1, 8) = -0.2; kxy(1, 9) = -0.2; kxy(1, 10) = -2.8; kxy(1, 11) = -0.2; kxy(1, 12) = 0.2;
kxy(2, 1) = 0.2; kxy(2, 2) = 0.26666667; kxy(2, 3) = 0; kxy(2, 4) = -0.2; kxy(2, 5) = -0.06666667; kxy(2, 6) = 0; kxy(2, 7) = 0.2; kxy(2, 8) = 0.06666667; kxy(2, 9) = 0; kxy(2, 10) = -0.2; kxy(2, 11) = -0.26666667; kxy(2, 12) = 0;
kxy(3, 1) = 0.2; kxy(3, 2) = 0; kxy(3, 3) = 0.26666667; kxy(3, 4) = -0.2; kxy(3, 5) = 0; kxy(3, 6) = -0.26666667; kxy(3, 7) = 0.2; kxy(3, 8) = 0; kxy(3, 9) = 0.06666667; kxy(3, 10) = -0.2; kxy(3, 11) = 0; kxy(3, 12) = -0.06666667;
kxy(4, 1) = -2.8; kxy(4, 2) = -0.2; kxy(4, 3) = -0.2; kxy(4, 4) = 2.8; kxy(4, 5) = -0.2; kxy(4, 6) = 0.2; kxy(4, 7) = -2.8; kxy(4, 8) = 0.2; kxy(4, 9) = 0.2; kxy(4, 10) = 2.8; kxy(4, 11) = 0.2; kxy(4, 12) = -0.2;
kxy(5, 1) = 0.2; kxy(5, 2) = -0.06666667; kxy(5, 3) = 0; kxy(5, 4) = -0.2; kxy(5, 5) = 0.26666667; kxy(5, 6) = 0; kxy(5, 7) = 0.2; kxy(5, 8) = -0.26666667; kxy(5, 9) = 0; kxy(5, 10) = -0.2; kxy(5, 11) = 0.06666667; kxy(5, 12) = 0;
kxy(6, 1) = -0.2; kxy(6, 2) = 0; kxy(6, 3) = -0.26666667; kxy(6, 4) = 0.2; kxy(6, 5) = 0; kxy(6, 6) = 0.26666667; kxy(6, 7) = -0.2; kxy(6, 8) = 0; kxy(6, 9) = -0.06666667; kxy(6, 10) = 0.2; kxy(6, 11) = 0; kxy(6, 12) = 0.06666667;
kxy(7, 1) = 2.8; kxy(7, 2) = 0.2; kxy(7, 3) = 0.2; kxy(7, 4) = -2.8; kxy(7, 5) = 0.2; kxy(7, 6) = -0.2; kxy(7, 7) = 2.8; kxy(7, 8) = -0.2; kxy(7, 9) = -0.2; kxy(7, 10) = -2.8; kxy(7, 11) = -0.2; kxy(7, 12) = 0.2;
kxy(8, 1) = -0.2; kxy(8, 2) = 0.06666667; kxy(8, 3) = 0; kxy(8, 4) = 0.2; kxy(8, 5) = -0.26666667; kxy(8, 6) = 0; kxy(8, 7) = -0.2; kxy(8, 8) = 0.26666667; kxy(8, 9) = 0; kxy(8, 10) = 0.2; kxy(8, 11) = -0.06666667; kxy(8, 12) = 0;
kxy(9, 1) = -0.2; kxy(9, 2) = 0; kxy(9, 3) = 0.06666667; kxy(9, 4) = 0.2; kxy(9, 5) = 0; kxy(9, 6) = -0.06666667; kxy(9, 7) = -0.2; kxy(9, 8) = 0; kxy(9, 9) = 0.26666667; kxy(9, 10) = 0.2; kxy(9, 11) = 0; kxy(9, 12) = -0.26666667;
kxy(10, 1) = -2.8; kxy(10, 2) = -0.2; kxy(10, 3) = -0.2; kxy(10, 4) = 2.8; kxy(10, 5) = -0.2; kxy(10, 6) = 0.2; kxy(10, 7) = -2.8; kxy(10, 8) = 0.2; kxy(10, 9) = 0.2; kxy(10, 10) = 2.8; kxy(10, 11) = 0.2; kxy(10, 12) = -0.2;
kxy(11, 1) = -0.2; kxy(11, 2) = -0.26666667; kxy(11, 3) = 0; kxy(11, 4) = 0.2; kxy(11, 5) = 0.06666667; kxy(11, 6) = 0; kxy(11, 7) = -0.2; kxy(11, 8) = -0.06666667; kxy(11, 9) = 0; kxy(11, 10) = 0.2; kxy(11, 11) = 0.26666667; kxy(11, 12) = 0;
kxy(12, 1) = 0.2; kxy(12, 2) = 0; kxy(12, 3) = -0.06666667; kxy(12, 4) = -0.2; kxy(12, 5) = 0; kxy(12, 6) = 0.06666667; kxy(12, 7) = 0.2; kxy(12, 8) = 0; kxy(12, 9) = -0.26666667; kxy(12, 10) = -0.2; kxy(12, 11) = 0; kxy(12, 12) = 0.26666667;
```

%' Inertia matrix

```
ki(1, 1) = 0.13706; ki(1, 2) = 0.01829; ki(1, 3) = 0.01829; ki(1, 4) = 0.04865; ki(1, 5) = -0.01087; ki(1, 6) = 0.0079; ki(1, 7) = 0.01563; ki(1, 8) = -0.0046; ki(1, 9) = -0.0046; ki(1, 10) = 0.04865; ki(1, 11) = 0.0079; ki(1, 12) = -0.01087;
ki(2, 1) = 0.01829; ki(2, 2) = 0.00317; ki(2, 3) = 0.0025; ki(2, 4) = 0.01087; ki(2, 5) = -0.00238; ki(2, 6) = 0.00167; ki(2, 7) = 0.0046; ki(2, 8) = -0.00119; ki(2, 9) = -0.00111; ki(2, 10) = 0.0079; ki(2, 11) = 0.00159; ki(2, 12) = -0.00167;
ki(3, 1) = 0.01829; ki(3, 2) = 0.0025; ki(3, 3) = 0.00317; ki(3, 4) = 0.0079; ki(3, 5) = -0.00167; ki(3, 6) = 0.00159; ki(3, 7) = 0.0046; ki(3, 8) = -0.00111; ki(3, 9) = -0.00119; ki(3, 10) = 0.01087; ki(3, 11) = 0.00167; ki(3, 12) = -0.00238;
ki(4, 1) = 0.04865; ki(4, 2) = 0.01087; ki(4, 3) = 0.0079; ki(4, 4) = 0.13706; ki(4, 5) = -0.01829; ki(4, 6) = 0.01829; ki(4, 7) = 0.04865; ki(4, 8) = -0.0079; ki(4, 9) = -0.01087; ki(4, 10) = 0.01563; ki(4, 11) = 0.0046; ki(4, 12) = -0.0046;
```

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

ki(5, 1) = -0.01087; ki(5, 2) = -0.00238; ki(5, 3) = -0.00167; ki(5, 4) = -0.01829; ki(5, 5) = 0.00317; ki(5, 6) = -
0.0025; ki(5, 7) = -0.0079; ki(5, 8) = 0.00159; ki(5, 9) = 0.00167; ki(5, 10) = -0.0046; ki(5, 11) = -0.00119; ki(5,
12) = 0.00111;
ki(6, 1) = 0.0079; ki(6, 2) = 0.00167; ki(6, 3) = 0.00159; ki(6, 4) = 0.01829; ki(6, 5) = -0.0025; ki(6, 6) =
0.00317; ki(6, 7) = 0.01087; ki(6, 8) = -0.00167; ki(6, 9) = -0.00238; ki(6, 10) = 0.0046; ki(6, 11) = 0.00111;
ki(6, 12) = -0.00119;
ki(7, 1) = 0.01563; ki(7, 2) = 0.0046; ki(7, 3) = 0.0046; ki(7, 4) = 0.04865; ki(7, 5) = -0.0079; ki(7, 6) =
0.01087; ki(7, 7) = 0.13706; ki(7, 8) = -0.01829; ki(7, 9) = -0.01829; ki(7, 10) = 0.04865; ki(7, 11) = 0.01087;
ki(7, 12) = -0.0079;
ki(8, 1) = -0.0046; ki(8, 2) = -0.00119; ki(8, 3) = -0.00111; ki(8, 4) = -0.0079; ki(8, 5) = 0.00159; ki(8, 6) = -
0.00167; ki(8, 7) = -0.01829; ki(8, 8) = 0.00317; ki(8, 9) = 0.0025; ki(8, 10) = -0.01087; ki(8, 11) = -0.00238;
ki(8, 12) = 0.00167;
ki(9, 1) = -0.0046; ki(9, 2) = -0.00111; ki(9, 3) = -0.00119; ki(9, 4) = -0.01087; ki(9, 5) = 0.00167; ki(9, 6) = -
0.00238; ki(9, 7) = -0.01829; ki(9, 8) = 0.0025; ki(9, 9) = 0.00317; ki(9, 10) = -0.0079; ki(9, 11) = -0.00167;
ki(9, 12) = 0.00159;
ki(10, 1) = 0.04865; ki(10, 2) = 0.0079; ki(10, 3) = 0.01087; ki(10, 4) = 0.01563; ki(10, 5) = -0.0046; ki(10, 6) =
0.0046; ki(10, 7) = 0.04865; ki(10, 8) = -0.01087; ki(10, 9) = -0.0079; ki(10, 10) = 0.13706; ki(10, 11) =
0.01829; ki(10, 12) = -0.01829;
ki(11, 1) = 0.0079; ki(11, 2) = 0.00159; ki(11, 3) = 0.00167; ki(11, 4) = 0.0046; ki(11, 5) = -0.00119; ki(11, 6) =
0.00111; ki(11, 7) = 0.01087; ki(11, 8) = -0.00238; ki(11, 9) = -0.00167; ki(11, 10) = 0.01829; ki(11, 11) =
0.00317; ki(11, 12) = -0.0025;
ki(12, 1) = -0.01087; ki(12, 2) = -0.00167; ki(12, 3) = -0.00238; ki(12, 4) = -0.0046; ki(12, 5) = 0.00111; ki(12,
6) = -0.00119; ki(12, 7) = -0.0079; ki(12, 8) = 0.00167; ki(12, 9) = 0.00159; ki(12, 10) = -0.01829; ki(12, 11) =
-0.0025; ki(12, 12) = 0.00317;

```

```

for x = 1 : 12
for y = 1 : 12
k(x, y) = k(x, y) + kx(x, y) + kxy(x, y) / af ^ 2 + ky(x, y) / af ^ 4;
end
end

```

```

%TYP = InputBox("WHAT TYPE OF PLATE? 1 for SSSS, 2 for CCCC, 3 for CSCS, 4 for CCSS, 5 for
CCCS, 6 for CSSS"); TYP = TYP * 1
g = input("WHAT IS THE SIZE OF THE GRID 3,5,7 etc?"); g = g * 1;
n = 3 * (g ^ 2); NN = (g + 2 + g) * 2; m = 3; mm = 3 * g; nm = n ;
%ReDim kk(nm, nm), q(nm), kki(nm, nm), kii(nm, nm)
for x = 1 : nm
for y = 1 : nm
kk(x, y) = 0; kii(x, y) = 0; kki(x, y) = 0;
end
end

```

```

% STIFFNESS for PURE W one node
x = 1;
%for x = 1 : n - 2 Step 3
while x < n - 1
kk(x, x) = 4 * k(1, 1); kii(x, x) = 4 * ki(1, 1);
x = x + 3;
end
% Pure Tx one node
x = 2;

```

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

% for x = 2 : n - 1 Step 3
while x < n
    kk(x, x) = 4 * k(2, 2); kii(x, x) = 4 * ki(2, 2);
    x = x + 3;
end

% Pure Ty one node
x = 3;
% for x = 3 : n Step 3
while x < n + 1
    kk(x, x) = 4 * k(3, 3); kii(x, x) = 4 * ki(3, 3);
    x = x + 3;
end

% Pure W-Tx; W-Ty one node
x = 3;
%for x = 3 : n - 2 Step 3
while x < n - 1
    kk(x, x + 1) = 0; kii(x, x + 1) = 0;
    kk(x, x + 2) = 0; kii(x, x + 2) = 0;
    x = x + 3;
end

% Pure W two nodes
t = 1; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
        kk(t, t2) = k(1, 10) + k(4, 7); kii(t, t2) = ki(1, 10) + ki(4, 7);
        kk(t, t3) = k(1, 7); kii(t, t3) = ki(1, 7);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 2; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(1, 10) + k(4, 7); kii(t, t1) = ki(1, 10) + ki(4, 7);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 1; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
    t = t1; t1 = t1 + 3;
end

% Pure Tx two nodes
t = 2; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1

```

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

for y = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    kk(t, t2) = k(2, 11) + k(5, 8); kii(t, t2) = ki(2, 11) + ki(5, 8);
    kk(t, t3) = k(2, 8); kii(t, t3) = ki(2, 8);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 1; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(2, 11) + k(5, 8); kii(t, t1) = ki(2, 11) + ki(5, 8);

t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 2; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    t = t1; t1 = t1 + 3;
end

% Pure Ty two nodes
t = 3; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
        kk(t, t2) = k(3, 12) + k(6, 9); kii(t, t2) = ki(3, 12) + ki(6, 9);
        kk(t, t3) = k(3, 9); kii(t, t3) = ki(3, 9);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(3, 12) + k(6, 9); kii(t, t1) = ki(3, 12) + ki(6, 9);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 3; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    t = t1; t1 = t1 + 3;
end

% Pure W two nodes back down
t = 4; t1 = t + 3 * (g - 1); t2 = t1 + 1; t3 = t1 + 2;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1

```


APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

for y = 1 : g - 1
    kk(t, t1) = k(4, 10); kii(t, t1) = ki(4, 10);
    kk(t + 1, t1) = k(5, 10); kii(t + 1, t1) = ki(5, 10);
    kk(t + 2, t1) = k(6, 10); kii(t + 2, t1) = ki(6, 10);

    kk(t + 1, t2) = k(5, 11); kii(t + 1, t2) = ki(5, 11);
    kk(t + 2, t3) = k(6, 12); kii(t + 2, t3) = ki(6, 12);

    kk(t, t2) = k(4, 11); kii(t, t2) = ki(4, 11);
    kk(t, t3) = k(4, 12); kii(t, t3) = ki(4, 12);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

% W-Tx W-Ty two nodes
t = 1; tx1 = t + 4; ty1 = t + 5; tx2 = t + 3 * g + 1; ty2 = t + 3 * g + 2;
tx3 = tx2 + 3; ty3 = ty2 + 3;
tt = t; ttx1 = tx1; tty1 = ty1; ttx2 = tx2; tty2 = ty2;
ttx3 = tx3; tty3 = ty3;

for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
        kk(t + 1, tx1 - 1) = k(2, 4) + k(11, 7); kii(t + 1, tx1 - 1) = ki(2, 4) + ki(11, 7);
        kk(t + 2, tx1 - 1) = k(3, 4) + k(12, 7); kii(t + 2, tx1 - 1) = ki(3, 4) + ki(12, 7);

        kk(t, tx2) = k(1, 11) + k(4, 8); kii(t, tx2) = ki(1, 11) + ki(4, 8);
        kk(t + 1, tx2 - 1) = k(2, 10) + k(5, 7); kii(t + 1, tx2 - 1) = ki(2, 10) + ki(5, 7);
        kk(t + 2, tx2 - 1) = k(3, 10) + k(6, 7); kii(t + 2, tx2 - 1) = ki(3, 10) + ki(6, 7);

        kk(t, tx3) = k(1, 8); kii(t, tx3) = ki(1, 8);
        kk(t + 1, tx3 - 1) = k(2, 7); kii(t + 1, tx3 - 1) = ki(2, 7);
        kk(t + 2, tx3 - 1) = k(3, 7); kii(t + 2, tx3 - 1) = ki(3, 7);

        kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
        kk(t, ty2) = k(1, 12) + k(4, 9); kii(t, ty2) = ki(1, 12) + ki(4, 9);
        kk(t, ty3) = k(1, 9); kii(t, ty3) = ki(1, 9);
        t = t + 3; tx1 = tx1 + 3; tx2 = tx2 + 3; tx3 = tx3 + 3;
        ty1 = ty1 + 3; ty2 = ty2 + 3; ty3 = ty3 + 3;
    end
        t = tt + 3 * g; tx1 = ttx1 + 3 * g; tx2 = ttx2 + 3 * g; tx3 = ttx3 + 3 * g;
        ty1 = tty1 + 3 * g; ty2 = tty2 + 3 * g; ty3 = tty3 + 3 * g;
        tt = t; ttx1 = tx1; ttx2 = tx2; ttx3 = tx3;
        tty1 = ty1; tty2 = ty2; tty3 = ty3;
    end

t = 3 * g - 2; tx1 = t + 3 * g + 1; ty1 = t + 3 * g + 2;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 11) + k(4, 8); kii(t, tx1) = ki(1, 11) + ki(4, 8);
    kk(t + 1, tx1 - 1) = k(5, 7) + k(2, 10); kii(t + 1, tx1 - 1) = ki(5, 7) + ki(2, 10);

```

APENDIX A

MATLAB Program Formulated for CCCC Boundary Condition

```

kk(t + 2, tx1 - 1) = k(6, 7) + k(3, 10); kii(t + 2, tx1 - 1) = ki(6, 7) + ki(3, 10);

kk(t, ty1) = k(1, 12) + k(4, 9); kii(t, ty1) = ki(1, 12) + ki(4, 9);
t = tx1 - 1; tx1 = tx1 + 3 * g; ty1 = ty1 + 3 * g;
end

t = n - 3 * g + 1; tx1 = t + 4; ty1 = t + 5;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(11, 7) + k(2, 4); kii(t + 1, tx1 - 1) = ki(11, 7) + ki(2, 4);
    kk(t + 2, tx1 - 1) = k(12, 7) + k(3, 4); kii(t + 2, tx1 - 1) = ki(12, 7) + ki(3, 4);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    t = tx1 - 1; tx1 = tx1 + 3; ty1 = ty1 + 3;
end

%Complete the symmetry
for x = 1 : nm
    for y = 1 : nm
        kk(y, x) = kk(x, y); kii(y, x) = kii(x, y);
    end
end

%Load Vector

x = 1;
while x < n + 1
%for x = 1 : n Step 3
    %%for x = 1 : n - 2 Step 3
        q(x) = 1;
        q(x + 1) = 0;
        q(x + 2) = 0;
        x = x + 3;
    %end
end

ncd = (n - 1) / 2;
kgv = inv(kk);
dd = kgv * transpose(q);
dc = dd(ncd) * 1000 / (1 + g) ^ 4;
kki = inv(kii) * kk;
ld = eig(kki) * (1 + g) ^ 4;
% ld = eig(kki);

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

%ReDim kx(12, 12), kxy(12, 12), ky(12, 12), k(12, 12), ki(12, 12)
for x = 1 : 12
for y = 1 : 12
k(x, y) = 0; kx(x, y) = 0; kxy(x, y) = 0; ky(x, y) = 0;
end
end
af = input('WHAT IS THE aspect ratio, b/a?'); af = af * 1;
kx(1, 1) = 4; kx(1, 2) = 2; kx(1, 3) = 0; kx(1, 4) = -4; kx(1, 5) = 2; kx(1, 6) = 0; kx(1, 7) = -2; kx(1, 8) = 1; kx(1, 9) = 0; kx(1, 10) = 2; kx(1, 11) = 1; kx(1, 12) = 0;
kx(2, 1) = 2; kx(2, 2) = 1.3333333; kx(2, 3) = 0; kx(2, 4) = -2; kx(2, 5) = 0.6666667; kx(2, 6) = 0; kx(2, 7) = -1; kx(2, 8) = 0.3333333; kx(2, 9) = 0; kx(2, 10) = 1; kx(2, 11) = 0.6666667; kx(2, 12) = 0;
kx(3, 1) = 0; kx(3, 2) = 0; kx(3, 3) = 0; kx(3, 4) = 0; kx(3, 5) = 0; kx(3, 6) = 0; kx(3, 7) = 0; kx(3, 8) = 0; kx(3, 9) = 0; kx(3, 10) = 0; kx(3, 11) = 0; kx(3, 12) = 0;
kx(4, 1) = -4; kx(4, 2) = -2; kx(4, 3) = 0; kx(4, 4) = 4; kx(4, 5) = -2; kx(4, 6) = 0; kx(4, 7) = 2; kx(4, 8) = -1; kx(4, 9) = 0; kx(4, 10) = -2; kx(4, 11) = -1; kx(4, 12) = 0;
kx(5, 1) = 2; kx(5, 2) = 0.6666667; kx(5, 3) = 0; kx(5, 4) = -2; kx(5, 5) = 1.3333333; kx(5, 6) = 0; kx(5, 7) = -1; kx(5, 8) = 0.6666667; kx(5, 9) = 0; kx(5, 10) = 1; kx(5, 11) = 0.3333333; kx(5, 12) = 0;
kx(6, 1) = 0; kx(6, 2) = 0; kx(6, 3) = 0; kx(6, 4) = 0; kx(6, 5) = 0; kx(6, 6) = 0; kx(6, 7) = 0; kx(6, 8) = 0; kx(6, 9) = 0; kx(6, 10) = 0; kx(6, 11) = 0; kx(6, 12) = 0;
kx(7, 1) = -2; kx(7, 2) = -1; kx(7, 3) = 0; kx(7, 4) = 2; kx(7, 5) = -1; kx(7, 6) = 0; kx(7, 7) = 4; kx(7, 8) = -2; kx(7, 9) = 0; kx(7, 10) = -4; kx(7, 11) = -2; kx(7, 12) = 0;
kx(8, 1) = 1; kx(8, 2) = 0.3333333; kx(8, 3) = 0; kx(8, 4) = -1; kx(8, 5) = 0.6666667; kx(8, 6) = 0; kx(8, 7) = -2; kx(8, 8) = 1.3333333; kx(8, 9) = 0; kx(8, 10) = 2; kx(8, 11) = 0.6666667; kx(8, 12) = 0;
kx(9, 1) = 0; kx(9, 2) = 0; kx(9, 3) = 0; kx(9, 4) = 0; kx(9, 5) = 0; kx(9, 6) = 0; kx(9, 7) = 0; kx(9, 8) = 0; kx(9, 9) = 0; kx(9, 10) = 0; kx(9, 11) = 0; kx(9, 12) = 0;
kx(10, 1) = 2; kx(10, 2) = 1; kx(10, 3) = 0; kx(10, 4) = -2; kx(10, 5) = 1; kx(10, 6) = 0; kx(10, 7) = -4; kx(10, 8) = 2; kx(10, 9) = 0; kx(10, 10) = 4; kx(10, 11) = 2; kx(10, 12) = 0;
kx(11, 1) = 1; kx(11, 2) = 0.6666667; kx(11, 3) = 0; kx(11, 4) = -1; kx(11, 5) = 0.3333333; kx(11, 6) = 0; kx(11, 7) = -2; kx(11, 8) = 0.6666667; kx(11, 9) = 0; kx(11, 10) = 2; kx(11, 11) = 1.3333333; kx(11, 12) = 0;
kx(12, 1) = 0; kx(12, 2) = 0; kx(12, 3) = 0; kx(12, 4) = 0; kx(12, 5) = 0; kx(12, 6) = 0; kx(12, 7) = 0; kx(12, 8) = 0; kx(12, 9) = 0; kx(12, 10) = 0; kx(12, 11) = 0; kx(12, 12) = 0;

ky(1, 1) = 4; ky(1, 2) = 0; ky(1, 3) = 2; ky(1, 4) = 2; ky(1, 5) = 0; ky(1, 6) = 1; ky(1, 7) = -2; ky(1, 8) = 0; ky(1, 9) = 1; ky(1, 10) = -4; ky(1, 11) = 0; ky(1, 12) = 2;
ky(2, 1) = 0; ky(2, 2) = 0; ky(2, 3) = 0; ky(2, 4) = 0; ky(2, 5) = 0; ky(2, 6) = 0; ky(2, 7) = 0; ky(2, 8) = 0; ky(2, 9) = 0; ky(2, 10) = 0; ky(2, 11) = 0; ky(2, 12) = 0;
ky(3, 1) = 2; ky(3, 2) = 0; ky(3, 3) = 1.3333333; ky(3, 4) = 1; ky(3, 5) = 0; ky(3, 6) = 0.6666667; ky(3, 7) = -1; ky(3, 8) = 0; ky(3, 9) = 0.3333333; ky(3, 10) = -2; ky(3, 11) = 0; ky(3, 12) = 0.6666667;
ky(4, 1) = 2; ky(4, 2) = 0; ky(4, 3) = 1; ky(4, 4) = 4; ky(4, 5) = 0; ky(4, 6) = 2; ky(4, 7) = -4; ky(4, 8) = 0; ky(4, 9) = 2; ky(4, 10) = -2; ky(4, 11) = 0; ky(4, 12) = 1;
ky(5, 1) = 0; ky(5, 2) = 0; ky(5, 3) = 0; ky(5, 4) = 0; ky(5, 5) = 0; ky(5, 6) = 0; ky(5, 7) = 0; ky(5, 8) = 0; ky(5, 9) = 0; ky(5, 10) = 0; ky(5, 11) = 0; ky(5, 12) = 0;
ky(6, 1) = 1; ky(6, 2) = 0; ky(6, 3) = 0.6666667; ky(6, 4) = 2; ky(6, 5) = 0; ky(6, 6) = 1.3333333; ky(6, 7) = -2; ky(6, 8) = 0; ky(6, 9) = 0.6666667; ky(6, 10) = -1; ky(6, 11) = 0; ky(6, 12) = 0.3333333;
ky(7, 1) = -2; ky(7, 2) = 0; ky(7, 3) = -1; ky(7, 4) = -4; ky(7, 5) = 0; ky(7, 6) = -2; ky(7, 7) = 4; ky(7, 8) = 0; ky(7, 9) = -2; ky(7, 10) = 2; ky(7, 11) = 0; ky(7, 12) = -1;
ky(8, 1) = 0; ky(8, 2) = 0; ky(8, 3) = 0; ky(8, 4) = 0; ky(8, 5) = 0; ky(8, 6) = 0; ky(8, 7) = 0; ky(8, 8) = 0; ky(8, 9) = 0; ky(8, 10) = 0; ky(8, 11) = 0; ky(8, 12) = 0;
ky(9, 1) = 1; ky(9, 2) = 0; ky(9, 3) = 0.3333333; ky(9, 4) = 2; ky(9, 5) = 0; ky(9, 6) = 0.6666667; ky(9, 7) = -2; ky(9, 8) = 0; ky(9, 9) = 1.3333333; ky(9, 10) = -1; ky(9, 11) = 0; ky(9, 12) = 0.6666667;

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```
ky(10, 1) = -4; ky(10, 2) = 0; ky(10, 3) = -2; ky(10, 4) = -2; ky(10, 5) = 0; ky(10, 6) = -1; ky(10, 7) = 2; ky(10, 8) = 0; ky(10, 9) = -1; ky(10, 10) = 4; ky(10, 11) = 0; ky(10, 12) = -2;
ky(11, 1) = 0; ky(11, 2) = 0; ky(11, 3) = 0; ky(11, 4) = 0; ky(11, 5) = 0; ky(11, 6) = 0; ky(11, 7) = 0; ky(11, 8) = 0; ky(11, 9) = 0; ky(11, 10) = 0; ky(11, 11) = 0; ky(11, 12) = 0;
ky(12, 1) = 2; ky(12, 2) = 0; ky(12, 3) = 0.6666667; ky(12, 4) = 1; ky(12, 5) = 0; ky(12, 6) = 0.3333333; ky(12, 7) = -1; ky(12, 8) = 0; ky(12, 9) = 0.6666667; ky(12, 10) = -2; ky(12, 11) = 0; ky(12, 12) = 1.3333333;

kxy(1, 1) = 2.8; kxy(1, 2) = 0.2; kxy(1, 3) = 0.2; kxy(1, 4) = -2.8; kxy(1, 5) = 0.2; kxy(1, 6) = -0.2; kxy(1, 7) = 2.8; kxy(1, 8) = -0.2; kxy(1, 9) = -0.2; kxy(1, 10) = -2.8; kxy(1, 11) = -0.2; kxy(1, 12) = 0.2;
kxy(2, 1) = 0.2; kxy(2, 2) = 0.2666667; kxy(2, 3) = 0; kxy(2, 4) = -0.2; kxy(2, 5) = -0.0666667; kxy(2, 6) = 0; kxy(2, 7) = 0.2; kxy(2, 8) = 0.0666667; kxy(2, 9) = 0; kxy(2, 10) = -0.2; kxy(2, 11) = -0.2666667; kxy(2, 12) = 0;
kxy(3, 1) = 0.2; kxy(3, 2) = 0; kxy(3, 3) = 0.2666667; kxy(3, 4) = -0.2; kxy(3, 5) = 0; kxy(3, 6) = -0.2666667; kxy(3, 7) = 0.2; kxy(3, 8) = 0; kxy(3, 9) = 0.0666667; kxy(3, 10) = -0.2; kxy(3, 11) = 0; kxy(3, 12) = -0.0666667;
kxy(4, 1) = -2.8; kxy(4, 2) = -0.2; kxy(4, 3) = -0.2; kxy(4, 4) = 2.8; kxy(4, 5) = -0.2; kxy(4, 6) = 0.2; kxy(4, 7) = -2.8; kxy(4, 8) = 0.2; kxy(4, 9) = 0.2; kxy(4, 10) = 2.8; kxy(4, 11) = 0.2; kxy(4, 12) = -0.2;
kxy(5, 1) = 0.2; kxy(5, 2) = -0.0666667; kxy(5, 3) = 0; kxy(5, 4) = -0.2; kxy(5, 5) = 0.2666667; kxy(5, 6) = 0; kxy(5, 7) = 0.2; kxy(5, 8) = -0.2666667; kxy(5, 9) = 0; kxy(5, 10) = -0.2; kxy(5, 11) = 0.0666667; kxy(5, 12) = 0;
kxy(6, 1) = -0.2; kxy(6, 2) = 0; kxy(6, 3) = -0.2666667; kxy(6, 4) = 0.2; kxy(6, 5) = 0; kxy(6, 6) = 0.2666667; kxy(6, 7) = -0.2; kxy(6, 8) = 0; kxy(6, 9) = -0.0666667; kxy(6, 10) = 0.2; kxy(6, 11) = 0; kxy(6, 12) = 0.0666667;
kxy(7, 1) = 2.8; kxy(7, 2) = 0.2; kxy(7, 3) = 0.2; kxy(7, 4) = -2.8; kxy(7, 5) = 0.2; kxy(7, 6) = -0.2; kxy(7, 7) = 2.8; kxy(7, 8) = -0.2; kxy(7, 9) = -0.2; kxy(7, 10) = -2.8; kxy(7, 11) = -0.2; kxy(7, 12) = 0.2;
kxy(8, 1) = -0.2; kxy(8, 2) = 0.0666667; kxy(8, 3) = 0; kxy(8, 4) = 0.2; kxy(8, 5) = -0.2666667; kxy(8, 6) = 0; kxy(8, 7) = -0.2; kxy(8, 8) = 0.2666667; kxy(8, 9) = 0; kxy(8, 10) = 0.2; kxy(8, 11) = -0.0666667; kxy(8, 12) = 0;
kxy(9, 1) = -0.2; kxy(9, 2) = 0; kxy(9, 3) = 0.0666667; kxy(9, 4) = 0.2; kxy(9, 5) = 0; kxy(9, 6) = -0.0666667; kxy(9, 7) = -0.2; kxy(9, 8) = 0; kxy(9, 9) = 0.2666667; kxy(9, 10) = 0.2; kxy(9, 11) = 0; kxy(9, 12) = -0.2666667;
kxy(10, 1) = -2.8; kxy(10, 2) = -0.2; kxy(10, 3) = -0.2; kxy(10, 4) = 2.8; kxy(10, 5) = -0.2; kxy(10, 6) = 0.2; kxy(10, 7) = -2.8; kxy(10, 8) = 0.2; kxy(10, 9) = 0.2; kxy(10, 10) = 2.8; kxy(10, 11) = 0.2; kxy(10, 12) = -0.2;
kxy(11, 1) = -0.2; kxy(11, 2) = -0.2666667; kxy(11, 3) = 0; kxy(11, 4) = 0.2; kxy(11, 5) = 0.0666667; kxy(11, 6) = 0; kxy(11, 7) = -0.2; kxy(11, 8) = -0.0666667; kxy(11, 9) = 0; kxy(11, 10) = 0.2; kxy(11, 11) = 0.2666667; kxy(11, 12) = 0;
kxy(12, 1) = 0.2; kxy(12, 2) = 0; kxy(12, 3) = -0.0666667; kxy(12, 4) = -0.2; kxy(12, 5) = 0; kxy(12, 6) = 0.0666667; kxy(12, 7) = 0.2; kxy(12, 8) = 0; kxy(12, 9) = -0.2666667; kxy(12, 10) = -0.2; kxy(12, 11) = 0; kxy(12, 12) = 0.2666667;

%' Inertia matrix
ki(1, 1) = 0.13706; ki(1, 2) = 0.01829; ki(1, 3) = 0.01829; ki(1, 4) = 0.04865; ki(1, 5) = -0.01087; ki(1, 6) = 0.0079; ki(1, 7) = 0.01563; ki(1, 8) = -0.0046; ki(1, 9) = -0.0046; ki(1, 10) = 0.04865; ki(1, 11) = 0.0079; ki(1, 12) = -0.01087;
ki(2, 1) = 0.01829; ki(2, 2) = 0.00317; ki(2, 3) = 0.0025; ki(2, 4) = 0.01087; ki(2, 5) = -0.00238; ki(2, 6) = 0.00167; ki(2, 7) = 0.0046; ki(2, 8) = -0.00119; ki(2, 9) = -0.00111; ki(2, 10) = 0.0079; ki(2, 11) = 0.00159; ki(2, 12) = -0.00167;
ki(3, 1) = 0.01829; ki(3, 2) = 0.0025; ki(3, 3) = 0.00317; ki(3, 4) = 0.0079; ki(3, 5) = -0.00167; ki(3, 6) = 0.00159; ki(3, 7) = 0.0046; ki(3, 8) = -0.00111; ki(3, 9) = -0.00119; ki(3, 10) = 0.01087; ki(3, 11) = 0.00167; ki(3, 12) = -0.00238;
ki(4, 1) = 0.04865; ki(4, 2) = 0.01087; ki(4, 3) = 0.0079; ki(4, 4) = 0.13706; ki(4, 5) = -0.01829; ki(4, 6) = 0.01829; ki(4, 7) = 0.04865; ki(4, 8) = -0.0079; ki(4, 9) = -0.01087; ki(4, 10) = 0.01563; ki(4, 11) = 0.0046; ki(4, 12) = -0.0046;
```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

ki(5, 1) = -0.01087; ki(5, 2) = -0.00238; ki(5, 3) = -0.00167; ki(5, 4) = -0.01829; ki(5, 5) = 0.00317; ki(5, 6) = -
0.0025; ki(5, 7) = -0.0079; ki(5, 8) = 0.00159; ki(5, 9) = 0.00167; ki(5, 10) = -0.0046; ki(5, 11) = -0.00119; ki(5,
12) = 0.00111;
ki(6, 1) = 0.0079; ki(6, 2) = 0.00167; ki(6, 3) = 0.00159; ki(6, 4) = 0.01829; ki(6, 5) = -0.0025; ki(6, 6) =
0.00317; ki(6, 7) = 0.01087; ki(6, 8) = -0.00167; ki(6, 9) = -0.00238; ki(6, 10) = 0.0046; ki(6, 11) = 0.00111;
ki(6, 12) = -0.00119;
ki(7, 1) = 0.01563; ki(7, 2) = 0.0046; ki(7, 3) = 0.0046; ki(7, 4) = 0.04865; ki(7, 5) = -0.0079; ki(7, 6) =
0.01087; ki(7, 7) = 0.13706; ki(7, 8) = -0.01829; ki(7, 9) = -0.01829; ki(7, 10) = 0.04865; ki(7, 11) = 0.01087;
ki(7, 12) = -0.0079;
ki(8, 1) = -0.0046; ki(8, 2) = -0.00119; ki(8, 3) = -0.00111; ki(8, 4) = -0.0079; ki(8, 5) = 0.00159; ki(8, 6) = -
0.00167; ki(8, 7) = -0.01829; ki(8, 8) = 0.00317; ki(8, 9) = 0.0025; ki(8, 10) = -0.01087; ki(8, 11) = -0.00238;
ki(8, 12) = 0.00167;
ki(9, 1) = -0.0046; ki(9, 2) = -0.00111; ki(9, 3) = -0.00119; ki(9, 4) = -0.01087; ki(9, 5) = 0.00167; ki(9, 6) = -
0.00238; ki(9, 7) = -0.01829; ki(9, 8) = 0.0025; ki(9, 9) = 0.00317; ki(9, 10) = -0.0079; ki(9, 11) = -0.00167;
ki(9, 12) = 0.00159;
ki(10, 1) = 0.04865; ki(10, 2) = 0.0079; ki(10, 3) = 0.01087; ki(10, 4) = 0.01563; ki(10, 5) = -0.0046; ki(10, 6) =
0.0046; ki(10, 7) = 0.04865; ki(10, 8) = -0.01087; ki(10, 9) = -0.0079; ki(10, 10) = 0.13706; ki(10, 11) =
0.01829; ki(10, 12) = -0.01829;
ki(11, 1) = 0.0079; ki(11, 2) = 0.00159; ki(11, 3) = 0.00167; ki(11, 4) = 0.0046; ki(11, 5) = -0.00119; ki(11, 6) =
0.00111; ki(11, 7) = 0.01087; ki(11, 8) = -0.00238; ki(11, 9) = -0.00167; ki(11, 10) = 0.01829; ki(11, 11) =
0.00317; ki(11, 12) = -0.0025;
ki(12, 1) = -0.01087; ki(12, 2) = -0.00167; ki(12, 3) = -0.00238; ki(12, 4) = -0.0046; ki(12, 5) = 0.00111; ki(12,
6) = -0.00119; ki(12, 7) = -0.0079; ki(12, 8) = 0.00167; ki(12, 9) = 0.00159; ki(12, 10) = -0.01829; ki(12, 11) =
-0.0025; ki(12, 12) = 0.00317;

```

```

for x = 1 : 12
for y = 1 : 12
k(x, y) = k(x, y) + kx(x, y) + kxy(x, y) / af ^ 2 + ky(x, y) / af ^ 4;
end
end

```

```

% TYP = InputBox("WHAT TYPE OF PLATE? 1 for SSSS, 2 for CCCC, 3 for CSCS, 4 for CCSS, 5 for
CCCC, 6 for CSSS"); TYP = TYP * 1
g = input("WHAT IS THE SIZE OF THE GRID 3,5,7 etc?"); g = g * 1;
n = 3 * (g ^ 2); NN = (g + 2 + g) * 2; m = 3; mm = 3 * g; nm = n + 2 * g;
% ReDim kk(nm, nm), q(nm), kki(nm, nm), kii(nm, nm)
for x = 1 : nm
for y = 1 : nm
kk(x, y) = 0; kii(x, y) = 0; kki(x, y) = 0;
end
end

```

```

% STIFFNESS for PURE W one node
x = 1;
% for x = 1 : n - 2 Step 3
while x < n - 1
kk(x, x) = 4 * k(1, 1); kii(x, x) = 4 * ki(1, 1);
x = x + 3;
end
% Pure Tx one node

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

x = 2;
% for x = 2 : n - 1 Step 3
while x < n
    kk(x, x) = 4 * k(2, 2); kii(x, x) = 4 * ki(2, 2);
    x = x + 3;
end

% Pure Ty one node
x = 3;
% for x = 3 : n Step 3
while x < n + 1
    kk(x, x) = 4 * k(3, 3); kii(x, x) = 4 * ki(3, 3);
    x = x + 3;
end

% Pure W-Tx; W-Ty one node
x = 3;
% for x = 3 : n - 2 Step 3
while x < n - 1
    kk(x, x + 1) = 0; kii(x, x + 1) = 0;
    kk(x, x + 2) = 0; kii(x, x + 2) = 0;
    x = x + 3;
end

% Pure W two nodes
t = 1; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
        kk(t, t2) = k(1, 10) + k(4, 7); kii(t, t2) = ki(1, 10) + ki(4, 7);
        kk(t, t3) = k(1, 7); kii(t, t3) = ki(1, 7);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 2; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(1, 10) + k(4, 7); kii(t, t1) = ki(1, 10) + ki(4, 7);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 1; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
    t = t1; t1 = t1 + 3;
end

% Pure Tx two nodes
t = 2; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
kk(t, t2) = k(2, 11) + k(5, 8); kii(t, t2) = ki(2, 11) + ki(5, 8);
kk(t, t3) = k(2, 8); kii(t, t3) = ki(2, 8);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 1; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(2, 11) + k(5, 8); kii(t, t1) = ki(2, 11) + ki(5, 8);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 2; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    t = t1; t1 = t1 + 3;
end

% Pure Ty two nodes
t = 3; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
        kk(t, t2) = k(3, 12) + k(6, 9); kii(t, t2) = ki(3, 12) + ki(6, 9);
        kk(t, t3) = k(3, 9); kii(t, t3) = ki(3, 9);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(3, 12) + k(6, 9); kii(t, t1) = ki(3, 12) + ki(6, 9);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 3; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    t = t1; t1 = t1 + 3;
end

% Pure W two nodes back down
t = 4; t1 = t + 3 * (g - 1); t2 = t1 + 1; t3 = t1 + 2;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(4, 10); kii(t, t1) = ki(4, 10);
    end
end

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

kk(t + 1, t1) = k(5, 10); kii(t + 1, t1) = ki(5, 10);
kk(t + 2, t1) = k(6, 10); kii(t + 2, t1) = ki(6, 10);

kk(t + 1, t2) = k(5, 11); kii(t + 1, t2) = ki(5, 11);
kk(t + 2, t3) = k(6, 12); kii(t + 2, t3) = ki(6, 12);

kk(t, t2) = k(4, 11); kii(t, t2) = ki(4, 11);
kk(t, t3) = k(4, 12); kii(t, t3) = ki(4, 12);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

% W-Tx W-Ty two nodes
t = 1; tx1 = t + 4; ty1 = t + 5; tx2 = t + 3 * g + 1; ty2 = t + 3 * g + 2;
tx3 = tx2 + 3; ty3 = ty2 + 3;
tt = t; ttx1 = tx1; tty1 = ty1; ttx2 = tx2; tty2 = ty2;
ttx3 = tx3; tty3 = ty3;

for x = 1 : g - 1
for y = 1 : g - 1
kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
kk(t + 1, tx1 - 1) = k(2, 4) + k(11, 7); kii(t + 1, tx1 - 1) = ki(2, 4) + ki(11, 7);
kk(t + 2, tx1 - 1) = k(3, 4) + k(12, 7); kii(t + 2, tx1 - 1) = ki(3, 4) + ki(12, 7);

kk(t, tx2) = k(1, 11) + k(4, 8); kii(t, tx2) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx2 - 1) = k(2, 10) + k(5, 7); kii(t + 1, tx2 - 1) = ki(2, 10) + ki(5, 7);
kk(t + 2, tx2 - 1) = k(3, 10) + k(6, 7); kii(t + 2, tx2 - 1) = ki(3, 10) + ki(6, 7);

kk(t, tx3) = k(1, 8); kii(t, tx3) = ki(1, 8);
kk(t + 1, tx3 - 1) = k(2, 7); kii(t + 1, tx3 - 1) = ki(2, 7);
kk(t + 2, tx3 - 1) = k(3, 7); kii(t + 2, tx3 - 1) = ki(3, 7);

kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
kk(t, ty2) = k(1, 12) + k(4, 9); kii(t, ty2) = ki(1, 12) + ki(4, 9);
kk(t, ty3) = k(1, 9); kii(t, ty3) = ki(1, 9);
t = t + 3; tx1 = tx1 + 3; tx2 = tx2 + 3; tx3 = tx3 + 3;
ty1 = ty1 + 3; ty2 = ty2 + 3; ty3 = ty3 + 3;
end
t = tt + 3 * g; tx1 = ttx1 + 3 * g; tx2 = ttx2 + 3 * g; tx3 = ttx3 + 3 * g;
ty1 = tty1 + 3 * g; ty2 = tty2 + 3 * g; ty3 = tty3 + 3 * g;
tt = t; ttx1 = tx1; ttx2 = tx2; ttx3 = tx3;
tty1 = ty1; tty2 = ty2; tty3 = ty3;
end

t = 3 * g - 2; tx1 = t + 3 * g + 1; ty1 = t + 3 * g + 2;
for x = 1 : g - 1
kk(t, tx1) = k(1, 11) + k(4, 8); kii(t, tx1) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx1 - 1) = k(5, 7) + k(2, 10); kii(t + 1, tx1 - 1) = ki(5, 7) + ki(2, 10);
kk(t + 2, tx1 - 1) = k(6, 7) + k(3, 10); kii(t + 2, tx1 - 1) = ki(6, 7) + ki(3, 10);

```


APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```

kk(t, ty1) = k(1, 12) + k(4, 9); kii(t, ty1) = ki(1, 12) + ki(4, 9);
t = tx1 - 1; tx1 = tx1 + 3 * g; ty1 = ty1 + 3 * g;
end

t = n - 3 * g + 1; tx1 = t + 4; ty1 = t + 5;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(11, 7) + k(2, 4); kii(t + 1, tx1 - 1) = ki(11, 7) + ki(2, 4);
    kk(t + 2, tx1 - 1) = k(12, 7) + k(3, 4); kii(t + 2, tx1 - 1) = ki(12, 7) + ki(3, 4);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    t = tx1 - 1; tx1 = tx1 + 3; ty1 = ty1 + 3;
end

% Boundary conditions
t1 = n + 1; t2 = n + g; t3 = n + g + 1; t4 = n + 2 * g; t5 = n + 2 * g + 1;
t6 = n + 3 * g; t7 = n + 3 * g + 1; t8 = n + 4 * g;
f1 = 1; f2 = n - 3 * g + 1; f3 = 3 * g - 2; f4 = n - 2;

%%Right boundary
x = f3; t1 = t1;
while x < f4 + 1
    %For x = f3 To f4 Step 3 * g
    kk(x, t1) = k(1, 5) + k(10, 8); kii(x, t1) = ki(1, 5) + ki(10, 8);
    kk(x + 1, t1) = k(2, 5) + k(11, 8); kii(x + 1, t1) = ki(2, 5) + ki(11, 8);
    kk(x + 2, t1) = k(3, 5) + k(12, 8); kii(x + 2, t1) = ki(3, 5) + ki(12, 8);
    x = x + 3 * g;
t1 = t1 + 1;
%end
end
x = f3 + 3 * g; t1 = t1;
while x < f4 + 1
    %For x = f3 + 3 * g To f4 Step 3 * g
    kk(x, t1) = k(10, 5); kii(x, t1) = ki(10, 5);
    kk(x + 1, t1) = k(11, 5); kii(x + 1, t1) = ki(11, 5);
    kk(x + 2, t1) = k(12, 5); kii(x + 2, t1) = ki(12, 5);
    % kk(t3, t3 + 1) = k(5, 8);
    x = x + 3 * g;
t1 = t1 + 1;
%end
end

for x = n + 1 : n + g - 1
kk(x, x + 1) = k(5, 8); kii(x, x + 1) = ki(5, 8);
end

x = f3; t1 = t1 + 1;
while x < f4 - 3 * g + 1
    %For x = f3 To f4 - 3 * g Step 3 * g
    kk(x, t1) = k(1, 8); kii(x, t1) = ki(1, 8);
    kk(x + 1, t1) = k(2, 8); kii(x + 1, t1) = ki(2, 8);
    kk(x + 2, t1) = k(3, 8); kii(x + 2, t1) = ki(3, 8);

```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```
x = x + 3 * g;
t1 = t1 + 1;
%end
end

% while x < n + 2 * g + 1
for x = n + 1 : n + g
%for x = n + 1; n + g
    kk(x, x) = k(5, 5) + k(8, 8);kii(x, x) = ki(5, 5) + ki(8, 8);
    %x = x + 1;
%end
end

%% Bottom boundary
x = f2; tt3 = t3;
while x < f4 + 1

%For x = f2 To f4 Step 3
    kk(x, t3) = k(1, 12) + k(4, 9);kii(x, t3) = ki(1, 12) + ki(4, 9);
    kk(x + 1, t3) = k(2, 12) + k(5, 9);kii(x + 1, t3) = ki(2, 12) + ki(5, 9);
    kk(x + 2, t3) = k(3, 12) + k(6, 9);kii(x + 2, t3) = ki(3, 12) + ki(6, 9);
    x = x + 3;
    t3 = t3 + 1;
%end
end

    x = f2 + 3; t3 = tt3;
while x < f4 + 1
%For x = f2 + 3 To f4 Step 3
    kk(x, t3) = k(4, 12);kii(x, t3) = ki(4, 12);
    kk(x + 1, t3) = k(5, 12); kii(x + 1, t3) = ki(5, 12);
    kk(x + 2, t3) = k(6, 12);kii(x + 2, t3) = ki(6, 12);
    % kk(t7, t7 + 1) = k(12, 9);
    x = x + 3;
    t3 = t3 + 1;
%end
end

for x = n + g + 1 : nm - 1
kk(x, x + 1) = k(9, 12);kii(x, x + 1) = ki(9, 12);
end

x = f2; t3 = tt3 + 1;
while x < f4 - 2
%For x = f2 To f4 - 3 Step 3
    kk(x, t3) = k(1, 9);kii(x, t3) = ki(1, 9);
    kk(x + 1, t3) = k(2, 9);kii(x + 1, t3) = ki(2, 9);
    kk(x + 2, t3) = k(3, 9);kii(x + 2, t3) = ki(3, 9);
    x = x + 3;
    t3 = t3 + 1;
%end
```

APENDIX B

MATLAB Program Formulated for CCSS Boundary Condition

```
end

% while x < nm + 1
for x = n + g + 1 : nm
%% for x = n + 1; n + g
    kk(x, x) = k(9, 9) + k(12, 12);kii(x, x) = ki(9, 9) + ki(12, 12);
    % x = x + 1;
%end
end

%Complete the symmetry
for x = 1 : nm
    for y = 1 : nm
        kk(y, x) = kk(x, y); kii(y, x) = kii(x, y);
    end
end

%Load Vector

x = 1;

while x < n + 1
%for x = 1 ; n Step 3
    %%for x = 1 ; n - 2 Step 3
        q(x) = 1;
        q(x + 1) = 0;
        q(x + 2) = 0;
        x = x +3;
%end
end

%right load
for x = n + 1 : n + g
    q(x) = -0.0833333333333333;
end

% bottom load
for x = n + g + 1 : n + 2 * g
    q(x) = -0.0833333333333333;
end

ncd = (n - 1) / 2;
kgv = inv(kk);
dd = kgv * transpose(q);
dc = dd(ncd) * 1000 / (1 + g) ^ 4;
kki = inv(kii) * kk;
ld = eig(kki) * (1 + g) ^ 4;
```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

%ReDim kx(12, 12), kxy(12, 12), ky(12, 12), k(12, 12), ki(12, 12)
for x = 1 : 12
for y = 1 : 12
k(x, y) = 0; kx(x, y) = 0; kxy(x, y) = 0; ky(x, y) = 0;
end
end
af = input('WHAT IS THE aspect ratio, b/a?'); af = af * 1;
kx(1, 1) = 4; kx(1, 2) = 2; kx(1, 3) = 0; kx(1, 4) = -4; kx(1, 5) = 2; kx(1, 6) = 0; kx(1, 7) = -2; kx(1, 8) = 1; kx(1, 9) = 0; kx(1, 10) = 2; kx(1, 11) = 1; kx(1, 12) = 0;
kx(2, 1) = 2; kx(2, 2) = 1.3333333; kx(2, 3) = 0; kx(2, 4) = -2; kx(2, 5) = 0.66666667; kx(2, 6) = 0; kx(2, 7) = -1; kx(2, 8) = 0.3333333; kx(2, 9) = 0; kx(2, 10) = 1; kx(2, 11) = 0.66666667; kx(2, 12) = 0;
kx(3, 1) = 0; kx(3, 2) = 0; kx(3, 3) = 0; kx(3, 4) = 0; kx(3, 5) = 0; kx(3, 6) = 0; kx(3, 7) = 0; kx(3, 8) = 0; kx(3, 9) = 0; kx(3, 10) = 0; kx(3, 11) = 0; kx(3, 12) = 0;
kx(4, 1) = -4; kx(4, 2) = -2; kx(4, 3) = 0; kx(4, 4) = 4; kx(4, 5) = -2; kx(4, 6) = 0; kx(4, 7) = 2; kx(4, 8) = -1; kx(4, 9) = 0; kx(4, 10) = -2; kx(4, 11) = -1; kx(4, 12) = 0;
kx(5, 1) = 2; kx(5, 2) = 0.66666667; kx(5, 3) = 0; kx(5, 4) = -2; kx(5, 5) = 1.3333333; kx(5, 6) = 0; kx(5, 7) = -1; kx(5, 8) = 0.66666667; kx(5, 9) = 0; kx(5, 10) = 1; kx(5, 11) = 0.3333333; kx(5, 12) = 0;
kx(6, 1) = 0; kx(6, 2) = 0; kx(6, 3) = 0; kx(6, 4) = 0; kx(6, 5) = 0; kx(6, 6) = 0; kx(6, 7) = 0; kx(6, 8) = 0; kx(6, 9) = 0; kx(6, 10) = 0; kx(6, 11) = 0; kx(6, 12) = 0;
kx(7, 1) = -2; kx(7, 2) = -1; kx(7, 3) = 0; kx(7, 4) = 2; kx(7, 5) = -1; kx(7, 6) = 0; kx(7, 7) = 4; kx(7, 8) = -2; kx(7, 9) = 0; kx(7, 10) = -4; kx(7, 11) = -2; kx(7, 12) = 0;
kx(8, 1) = 1; kx(8, 2) = 0.3333333; kx(8, 3) = 0; kx(8, 4) = -1; kx(8, 5) = 0.66666667; kx(8, 6) = 0; kx(8, 7) = -2; kx(8, 8) = 1.3333333; kx(8, 9) = 0; kx(8, 10) = 2; kx(8, 11) = 0.66666667; kx(8, 12) = 0;
kx(9, 1) = 0; kx(9, 2) = 0; kx(9, 3) = 0; kx(9, 4) = 0; kx(9, 5) = 0; kx(9, 6) = 0; kx(9, 7) = 0; kx(9, 8) = 0; kx(9, 9) = 0; kx(9, 10) = 0; kx(9, 11) = 0; kx(9, 12) = 0;
kx(10, 1) = 2; kx(10, 2) = 1; kx(10, 3) = 0; kx(10, 4) = -2; kx(10, 5) = 1; kx(10, 6) = 0; kx(10, 7) = -4; kx(10, 8) = 2; kx(10, 9) = 0; kx(10, 10) = 4; kx(10, 11) = 2; kx(10, 12) = 0;
kx(11, 1) = 1; kx(11, 2) = 0.66666667; kx(11, 3) = 0; kx(11, 4) = -1; kx(11, 5) = 0.3333333; kx(11, 6) = 0; kx(11, 7) = -2; kx(11, 8) = 0.66666667; kx(11, 9) = 0; kx(11, 10) = 2; kx(11, 11) = 1.3333333; kx(11, 12) = 0;
kx(12, 1) = 0; kx(12, 2) = 0; kx(12, 3) = 0; kx(12, 4) = 0; kx(12, 5) = 0; kx(12, 6) = 0; kx(12, 7) = 0; kx(12, 8) = 0; kx(12, 9) = 0; kx(12, 10) = 0; kx(12, 11) = 0; kx(12, 12) = 0;

ky(1, 1) = 4; ky(1, 2) = 0; ky(1, 3) = 2; ky(1, 4) = 2; ky(1, 5) = 0; ky(1, 6) = 1; ky(1, 7) = -2; ky(1, 8) = 0; ky(1, 9) = 1; ky(1, 10) = -4; ky(1, 11) = 0; ky(1, 12) = 2;
ky(2, 1) = 0; ky(2, 2) = 0; ky(2, 3) = 0; ky(2, 4) = 0; ky(2, 5) = 0; ky(2, 6) = 0; ky(2, 7) = 0; ky(2, 8) = 0; ky(2, 9) = 0; ky(2, 10) = 0; ky(2, 11) = 0; ky(2, 12) = 0;
ky(3, 1) = 2; ky(3, 2) = 0; ky(3, 3) = 1.3333333; ky(3, 4) = 1; ky(3, 5) = 0; ky(3, 6) = 0.66666667; ky(3, 7) = -1; ky(3, 8) = 0; ky(3, 9) = 0.3333333; ky(3, 10) = -2; ky(3, 11) = 0; ky(3, 12) = 0.66666667;
ky(4, 1) = 2; ky(4, 2) = 0; ky(4, 3) = 1; ky(4, 4) = 4; ky(4, 5) = 0; ky(4, 6) = 2; ky(4, 7) = -4; ky(4, 8) = 0; ky(4, 9) = 2; ky(4, 10) = -2; ky(4, 11) = 0; ky(4, 12) = 1;
ky(5, 1) = 0; ky(5, 2) = 0; ky(5, 3) = 0; ky(5, 4) = 0; ky(5, 5) = 0; ky(5, 6) = 0; ky(5, 7) = 0; ky(5, 8) = 0; ky(5, 9) = 0; ky(5, 10) = 0; ky(5, 11) = 0; ky(5, 12) = 0;
ky(6, 1) = 1; ky(6, 2) = 0; ky(6, 3) = 0.66666667; ky(6, 4) = 2; ky(6, 5) = 0; ky(6, 6) = 1.3333333; ky(6, 7) = -2; ky(6, 8) = 0; ky(6, 9) = 0.66666667; ky(6, 10) = -1; ky(6, 11) = 0; ky(6, 12) = 0.3333333;
ky(7, 1) = -2; ky(7, 2) = 0; ky(7, 3) = -1; ky(7, 4) = -4; ky(7, 5) = 0; ky(7, 6) = -2; ky(7, 7) = 4; ky(7, 8) = 0; ky(7, 9) = -2; ky(7, 10) = 2; ky(7, 11) = 0; ky(7, 12) = -1;
ky(8, 1) = 0; ky(8, 2) = 0; ky(8, 3) = 0; ky(8, 4) = 0; ky(8, 5) = 0; ky(8, 6) = 0; ky(8, 7) = 0; ky(8, 8) = 0; ky(8, 9) = 0; ky(8, 10) = 0; ky(8, 11) = 0; ky(8, 12) = 0;
ky(9, 1) = 1; ky(9, 2) = 0; ky(9, 3) = 0.3333333; ky(9, 4) = 2; ky(9, 5) = 0; ky(9, 6) = 0.66666667; ky(9, 7) = -2; ky(9, 8) = 0; ky(9, 9) = 1.3333333; ky(9, 10) = -1; ky(9, 11) = 0; ky(9, 12) = 0.66666667;

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```
ky(10, 1) = -4; ky(10, 2) = 0; ky(10, 3) = -2; ky(10, 4) = -2; ky(10, 5) = 0; ky(10, 6) = -1; ky(10, 7) = 2; ky(10, 8) = 0; ky(10, 9) = -1; ky(10, 10) = 4; ky(10, 11) = 0; ky(10, 12) = -2;
ky(11, 1) = 0; ky(11, 2) = 0; ky(11, 3) = 0; ky(11, 4) = 0; ky(11, 5) = 0; ky(11, 6) = 0; ky(11, 7) = 0; ky(11, 8) = 0; ky(11, 9) = 0; ky(11, 10) = 0; ky(11, 11) = 0; ky(11, 12) = 0;
ky(12, 1) = 2; ky(12, 2) = 0; ky(12, 3) = 0.6666667; ky(12, 4) = 1; ky(12, 5) = 0; ky(12, 6) = 0.3333333; ky(12, 7) = -1; ky(12, 8) = 0; ky(12, 9) = 0.6666667; ky(12, 10) = -2; ky(12, 11) = 0; ky(12, 12) = 1.3333333;

kxy(1, 1) = 2.8; kxy(1, 2) = 0.2; kxy(1, 3) = 0.2; kxy(1, 4) = -2.8; kxy(1, 5) = 0.2; kxy(1, 6) = -0.2; kxy(1, 7) = 2.8; kxy(1, 8) = -0.2; kxy(1, 9) = -0.2; kxy(1, 10) = -2.8; kxy(1, 11) = -0.2; kxy(1, 12) = 0.2;
kxy(2, 1) = 0.2; kxy(2, 2) = 0.2666667; kxy(2, 3) = 0; kxy(2, 4) = -0.2; kxy(2, 5) = -0.0666667; kxy(2, 6) = 0; kxy(2, 7) = 0.2; kxy(2, 8) = 0.0666667; kxy(2, 9) = 0; kxy(2, 10) = -0.2; kxy(2, 11) = -0.2666667; kxy(2, 12) = 0;
kxy(3, 1) = 0.2; kxy(3, 2) = 0; kxy(3, 3) = 0.2666667; kxy(3, 4) = -0.2; kxy(3, 5) = 0; kxy(3, 6) = -0.2666667; kxy(3, 7) = 0.2; kxy(3, 8) = 0; kxy(3, 9) = 0.0666667; kxy(3, 10) = -0.2; kxy(3, 11) = 0; kxy(3, 12) = -0.0666667;
kxy(4, 1) = -2.8; kxy(4, 2) = -0.2; kxy(4, 3) = -0.2; kxy(4, 4) = 2.8; kxy(4, 5) = -0.2; kxy(4, 6) = 0.2; kxy(4, 7) = -2.8; kxy(4, 8) = 0.2; kxy(4, 9) = 0.2; kxy(4, 10) = 2.8; kxy(4, 11) = 0.2; kxy(4, 12) = -0.2;
kxy(5, 1) = 0.2; kxy(5, 2) = -0.0666667; kxy(5, 3) = 0; kxy(5, 4) = -0.2; kxy(5, 5) = 0.2666667; kxy(5, 6) = 0; kxy(5, 7) = 0.2; kxy(5, 8) = -0.2666667; kxy(5, 9) = 0; kxy(5, 10) = -0.2; kxy(5, 11) = 0.0666667; kxy(5, 12) = 0;
kxy(6, 1) = -0.2; kxy(6, 2) = 0; kxy(6, 3) = -0.2666667; kxy(6, 4) = 0.2; kxy(6, 5) = 0; kxy(6, 6) = 0.2666667; kxy(6, 7) = -0.2; kxy(6, 8) = 0; kxy(6, 9) = -0.0666667; kxy(6, 10) = 0.2; kxy(6, 11) = 0; kxy(6, 12) = 0.0666667;
kxy(7, 1) = 2.8; kxy(7, 2) = 0.2; kxy(7, 3) = 0.2; kxy(7, 4) = -2.8; kxy(7, 5) = 0.2; kxy(7, 6) = -0.2; kxy(7, 7) = 2.8; kxy(7, 8) = -0.2; kxy(7, 9) = -0.2; kxy(7, 10) = -2.8; kxy(7, 11) = -0.2; kxy(7, 12) = 0.2;
kxy(8, 1) = -0.2; kxy(8, 2) = 0.0666667; kxy(8, 3) = 0; kxy(8, 4) = 0.2; kxy(8, 5) = -0.2666667; kxy(8, 6) = 0; kxy(8, 7) = -0.2; kxy(8, 8) = 0.2666667; kxy(8, 9) = 0; kxy(8, 10) = 0.2; kxy(8, 11) = -0.0666667; kxy(8, 12) = 0;
kxy(9, 1) = -0.2; kxy(9, 2) = 0; kxy(9, 3) = 0.0666667; kxy(9, 4) = 0.2; kxy(9, 5) = 0; kxy(9, 6) = -0.0666667; kxy(9, 7) = -0.2; kxy(9, 8) = 0; kxy(9, 9) = 0.2666667; kxy(9, 10) = 0.2; kxy(9, 11) = 0; kxy(9, 12) = -0.2666667;
kxy(10, 1) = -2.8; kxy(10, 2) = -0.2; kxy(10, 3) = -0.2; kxy(10, 4) = 2.8; kxy(10, 5) = -0.2; kxy(10, 6) = 0.2; kxy(10, 7) = -2.8; kxy(10, 8) = 0.2; kxy(10, 9) = 0.2; kxy(10, 10) = 2.8; kxy(10, 11) = 0.2; kxy(10, 12) = -0.2;
kxy(11, 1) = -0.2; kxy(11, 2) = -0.2666667; kxy(11, 3) = 0; kxy(11, 4) = 0.2; kxy(11, 5) = 0.0666667; kxy(11, 6) = 0; kxy(11, 7) = -0.2; kxy(11, 8) = -0.0666667; kxy(11, 9) = 0; kxy(11, 10) = 0.2; kxy(11, 11) = 0.2666667; kxy(11, 12) = 0;
kxy(12, 1) = 0.2; kxy(12, 2) = 0; kxy(12, 3) = -0.0666667; kxy(12, 4) = -0.2; kxy(12, 5) = 0; kxy(12, 6) = 0.0666667; kxy(12, 7) = 0.2; kxy(12, 8) = 0; kxy(12, 9) = -0.2666667; kxy(12, 10) = -0.2; kxy(12, 11) = 0; kxy(12, 12) = 0.2666667;

%' Inertia matrix
ki(1, 1) = 0.13706; ki(1, 2) = 0.01829; ki(1, 3) = 0.01829; ki(1, 4) = 0.04865; ki(1, 5) = -0.01087; ki(1, 6) = 0.0079; ki(1, 7) = 0.01563; ki(1, 8) = -0.0046; ki(1, 9) = -0.0046; ki(1, 10) = 0.04865; ki(1, 11) = 0.0079; ki(1, 12) = -0.01087;
ki(2, 1) = 0.01829; ki(2, 2) = 0.00317; ki(2, 3) = 0.0025; ki(2, 4) = 0.01087; ki(2, 5) = -0.00238; ki(2, 6) = 0.00167; ki(2, 7) = 0.0046; ki(2, 8) = -0.00119; ki(2, 9) = -0.00111; ki(2, 10) = 0.0079; ki(2, 11) = 0.00159; ki(2, 12) = -0.00167;
ki(3, 1) = 0.01829; ki(3, 2) = 0.0025; ki(3, 3) = 0.00317; ki(3, 4) = 0.0079; ki(3, 5) = -0.00167; ki(3, 6) = 0.00159; ki(3, 7) = 0.0046; ki(3, 8) = -0.00111; ki(3, 9) = -0.00119; ki(3, 10) = 0.01087; ki(3, 11) = 0.00167; ki(3, 12) = -0.00238;
ki(4, 1) = 0.04865; ki(4, 2) = 0.01087; ki(4, 3) = 0.0079; ki(4, 4) = 0.13706; ki(4, 5) = -0.01829; ki(4, 6) = 0.01829; ki(4, 7) = 0.04865; ki(4, 8) = -0.0079; ki(4, 9) = -0.01087; ki(4, 10) = 0.01563; ki(4, 11) = 0.0046; ki(4, 12) = -0.0046;
```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

ki(5, 1) = -0.01087; ki(5, 2) = -0.00238; ki(5, 3) = -0.00167; ki(5, 4) = -0.01829; ki(5, 5) = 0.00317; ki(5, 6) = -
0.0025; ki(5, 7) = -0.0079; ki(5, 8) = 0.00159; ki(5, 9) = 0.00167; ki(5, 10) = -0.0046; ki(5, 11) = -0.00119; ki(5,
12) = 0.00111;
ki(6, 1) = 0.0079; ki(6, 2) = 0.00167; ki(6, 3) = 0.00159; ki(6, 4) = 0.01829; ki(6, 5) = -0.0025; ki(6, 6) =
0.00317; ki(6, 7) = 0.01087; ki(6, 8) = -0.00167; ki(6, 9) = -0.00238; ki(6, 10) = 0.0046; ki(6, 11) = 0.00111;
ki(6, 12) = -0.00119;
ki(7, 1) = 0.01563; ki(7, 2) = 0.0046; ki(7, 3) = 0.0046; ki(7, 4) = 0.04865; ki(7, 5) = -0.0079; ki(7, 6) =
0.01087; ki(7, 7) = 0.13706; ki(7, 8) = -0.01829; ki(7, 9) = -0.01829; ki(7, 10) = 0.04865; ki(7, 11) = 0.01087;
ki(7, 12) = -0.0079;
ki(8, 1) = -0.0046; ki(8, 2) = -0.00119; ki(8, 3) = -0.00111; ki(8, 4) = -0.0079; ki(8, 5) = 0.00159; ki(8, 6) = -
0.00167; ki(8, 7) = -0.01829; ki(8, 8) = 0.00317; ki(8, 9) = 0.0025; ki(8, 10) = -0.01087; ki(8, 11) = -0.00238;
ki(8, 12) = 0.00167;
ki(9, 1) = -0.0046; ki(9, 2) = -0.00111; ki(9, 3) = -0.00119; ki(9, 4) = -0.01087; ki(9, 5) = 0.00167; ki(9, 6) = -
0.00238; ki(9, 7) = -0.01829; ki(9, 8) = 0.0025; ki(9, 9) = 0.00317; ki(9, 10) = -0.0079; ki(9, 11) = -0.00167;
ki(9, 12) = 0.00159;
ki(10, 1) = 0.04865; ki(10, 2) = 0.0079; ki(10, 3) = 0.01087; ki(10, 4) = 0.01563; ki(10, 5) = -0.0046; ki(10, 6) =
0.0046; ki(10, 7) = 0.04865; ki(10, 8) = -0.01087; ki(10, 9) = -0.0079; ki(10, 10) = 0.13706; ki(10, 11) =
0.01829; ki(10, 12) = -0.01829;
ki(11, 1) = 0.0079; ki(11, 2) = 0.00159; ki(11, 3) = 0.00167; ki(11, 4) = 0.0046; ki(11, 5) = -0.00119; ki(11, 6) =
0.00111; ki(11, 7) = 0.01087; ki(11, 8) = -0.00238; ki(11, 9) = -0.00167; ki(11, 10) = 0.01829; ki(11, 11) =
0.00317; ki(11, 12) = -0.0025;
ki(12, 1) = -0.01087; ki(12, 2) = -0.00167; ki(12, 3) = -0.00238; ki(12, 4) = -0.0046; ki(12, 5) = 0.00111; ki(12,
6) = -0.00119; ki(12, 7) = -0.0079; ki(12, 8) = 0.00167; ki(12, 9) = 0.00159; ki(12, 10) = -0.01829; ki(12, 11) =
-0.0025; ki(12, 12) = 0.00317;

for x = 1 : 12
for y = 1 : 12
k(x, y) = k(x, y) + kx(x, y) + kxy(x, y) / af ^ 2 + ky(x, y) / af ^ 4;
end
end

%TYP = InputBox("WHAT TYPE OF PLATE? 1 for SSSS, 2 for CCCC, 3 for CSCS, 4 for CCSS, 5 for
CCCC, 6 for CSSS"); TYP = TYP * 1;
g = input("WHAT IS THE SIZE OF THE GRID 3,5,7 etc?"); g = g * 1;
n = 3 * (g ^ 2); NN = (g + 2 + g) * 2; m = 3; mm = 3 * g; nm = n + 2 * g;
%ReDim kk(nm, nm), q(nm), kki(nm, nm), kii(nm, nm)
for x = 1 : nm
for y = 1 : nm
kk(x, y) = 0; kii(x, y) = 0; kki(x, y) = 0;
end
end

% STIFFNESS for PURE W one node
x = 1;
%for x = 1 : n - 2 Step 3
while x < n - 1
kk(x, x) = 4 * k(1, 1); kii(x, x) = 4 * ki(1, 1);
x = x + 3;
end
% Pure Tx one node
x = 2;

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

% for x = 2 : n - 1 Step 3
while x < n
    kk(x, x) = 4 * k(2, 2); kii(x, x) = 4 * ki(2, 2);
    x = x + 3;
end

% Pure Ty one node
x = 3;
% for x = 3 : n Step 3
while x < n + 1
    kk(x, x) = 4 * k(3, 3); kii(x, x) = 4 * ki(3, 3);
    x = x + 3;
end

% Pure W-Tx; W-Ty one node
x = 3;
% for x = 3 : n - 2 Step 3
while x < n - 1
    kk(x, x + 1) = 0; kii(x, x + 1) = 0;
    kk(x, x + 2) = 0; kii(x, x + 2) = 0;
    x = x + 3;
end

% Pure W two nodes
t = 1; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
        kk(t, t2) = k(1, 10) + k(4, 7); kii(t, t2) = ki(1, 10) + ki(4, 7);
        kk(t, t3) = k(1, 7); kii(t, t3) = ki(1, 7);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 2; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(1, 10) + k(4, 7); kii(t, t1) = ki(1, 10) + ki(4, 7);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 1; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
    t = t1; t1 = t1 + 3;
end

% Pure Tx two nodes
t = 2; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    end
end

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

kk(t, t2) = k(2, 11) + k(5, 8); kii(t, t2) = ki(2, 11) + ki(5, 8);
kk(t, t3) = k(2, 8); kii(t, t3) = ki(2, 8);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g - 1; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(2, 11) + k(5, 8); kii(t, t1) = ki(2, 11) + ki(5, 8);
    t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 2; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    t = t1; t1 = t1 + 3;
end

```

% Pure Ty two nodes

```

t = 3; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    kk(t, t2) = k(3, 12) + k(6, 9); kii(t, t2) = ki(3, 12) + ki(6, 9);
    kk(t, t3) = k(3, 9); kii(t, t3) = ki(3, 9);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(3, 12) + k(6, 9); kii(t, t1) = ki(3, 12) + ki(6, 9);
    t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 3; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    t = t1; t1 = t1 + 3;
end

```

% Pure W two nodes back down

```

t = 4; t1 = t + 3 * (g - 1); t2 = t1 + 1; t3 = t1 + 2;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, t1) = k(4, 10); kii(t, t1) = ki(4, 10);

```


APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

kk(t + 1, t1) = k(5, 10); kii(t + 1, t1) = ki(5, 10);
kk(t + 2, t1) = k(6, 10); kii(t + 2, t1) = ki(6, 10);

kk(t + 1, t2) = k(5, 11); kii(t + 1, t2) = ki(5, 11);
kk(t + 2, t3) = k(6, 12); kii(t + 2, t3) = ki(6, 12);

kk(t, t2) = k(4, 11); kii(t, t2) = ki(4, 11);
kk(t, t3) = k(4, 12); kii(t, t3) = ki(4, 12);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;

end

% W-Tx W-Ty two nodes
t = 1; tx1 = t + 4; ty1 = t + 5; tx2 = t + 3 * g + 1; ty2 = t + 3 * g + 2;
tx3 = tx2 + 3; ty3 = ty2 + 3;
tt = t; ttx1 = tx1; tty1 = ty1; ttx2 = tx2; tty2 = ty2;
ttx3 = tx3; tty3 = ty3;

for x = 1 : g - 1
for y = 1 : g - 1
kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
kk(t + 1, tx1 - 1) = k(2, 4) + k(11, 7); kii(t + 1, tx1 - 1) = ki(2, 4) + ki(11, 7);
kk(t + 2, tx1 - 1) = k(3, 4) + k(12, 7); kii(t + 2, tx1 - 1) = ki(3, 4) + ki(12, 7);

kk(t, tx2) = k(1, 11) + k(4, 8); kii(t, tx2) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx2 - 1) = k(2, 10) + k(5, 7); kii(t + 1, tx2 - 1) = ki(2, 10) + ki(5, 7);
kk(t + 2, tx2 - 1) = k(3, 10) + k(6, 7); kii(t + 2, tx2 - 1) = ki(3, 10) + ki(6, 7);

kk(t, tx3) = k(1, 8); kii(t, tx3) = ki(1, 8);
kk(t + 1, tx3 - 1) = k(2, 7); kii(t + 1, tx3 - 1) = ki(2, 7);
kk(t + 2, tx3 - 1) = k(3, 7); kii(t + 2, tx3 - 1) = ki(3, 7);

kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
kk(t, ty2) = k(1, 12) + k(4, 9); kii(t, ty2) = ki(1, 12) + ki(4, 9);
kk(t, ty3) = k(1, 9); kii(t, ty3) = ki(1, 9);
t = t + 3; tx1 = tx1 + 3; tx2 = tx2 + 3; tx3 = tx3 + 3;
ty1 = ty1 + 3; ty2 = ty2 + 3; ty3 = ty3 + 3;
end
t = tt + 3 * g; tx1 = ttx1 + 3 * g; tx2 = ttx2 + 3 * g; tx3 = ttx3 + 3 * g;
ty1 = tty1 + 3 * g; ty2 = tty2 + 3 * g; ty3 = tty3 + 3 * g;
tt = t; ttx1 = tx1; ttx2 = tx2; ttx3 = tx3;
tty1 = ty1; tty2 = ty2; tty3 = ty3;
end

t = 3 * g - 2; tx1 = t + 3 * g + 1; ty1 = t + 3 * g + 2;
for x = 1 : g - 1
kk(t, tx1) = k(1, 11) + k(4, 8); kii(t, tx1) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx1 - 1) = k(5, 7) + k(2, 10); kii(t + 1, tx1 - 1) = ki(5, 7) + ki(2, 10);
kk(t + 2, tx1 - 1) = k(6, 7) + k(3, 10); kii(t + 2, tx1 - 1) = ki(6, 7) + ki(3, 10);

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

kk(t, ty1) = k(1, 12) + k(4, 9); kii(t, ty1) = ki(1, 12) + ki(4, 9);
t = tx1 - 1; tx1 = tx1 + 3 * g; ty1 = ty1 + 3 * g;
end

t = n - 3 * g + 1; tx1 = t + 4; ty1 = t + 5;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(11, 7) + k(2, 4); kii(t + 1, tx1 - 1) = ki(11, 7) + ki(2, 4);
    kk(t + 2, tx1 - 1) = k(12, 7) + k(3, 4); kii(t + 2, tx1 - 1) = ki(12, 7) + ki(3, 4);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    t = tx1 - 1; tx1 = tx1 + 3; ty1 = ty1 + 3;
end

% Boundary conditions
t1 = n + 1; t2 = n + g; t3 = n + g + 1; t4 = n + 2 * g; t5 = n + 2 * g + 1;
t6 = n + 3 * g; t7 = n + 3 * g + 1; t8 = n + 4 * g;
f1 = 1; f2 = n - 3 * g + 1; f3 = 3 * g - 2; f4 = n - 2;
%left boundary
x = 1; t1 = t1;
while x < f2 + 1
%for x = 1 : f2 Step 3 * g
    kk(x, t1) = k(4, 2) + k(7, 11); kii(x, t1) = ki(4, 2) + ki(7, 11);
    kk(x + 1, t1) = k(5, 2) + k(8, 11); kii(x + 1, t1) = ki(5, 2) + ki(8, 11);
    kk(x + 2, t1) = k(6, 2) + k(9, 11); kii(x + 2, t1) = ki(6, 2) + ki(9, 11);
    x = x + 3 * g;
    t1 = t1 + 1;
%end
end
x = 3 * g + 1; t1 = t1;
while x < f2 + 1
%for x = 3 * g + 1 : f2 Step 3 * g
    kk(x, t1) = k(7, 2); kii(x, t1) = ki(7, 2);
    kk(x + 1, t1) = k(8, 2); kii(x + 1, t1) = ki(8, 2);
    kk(x + 2, t1) = k(9, 2); kii(x + 2, t1) = ki(9, 2);
    %kk(t1, t1 + 1) = k(2, 11);
    x = x + 3 * g;
    t1 = t1 + 1;
%end
end

for x = n + 1 : n + g - 1
kk(x, x + 1) = k(2, 11); kii(x, x + 1) = ki(2, 11);
end

x = 1; t1 = t1 + 1;
while x < f2 - 3 * g + 1
%for x = 1 : f2 - 3 * g Step 3 * g
    kk(x, t1) = k(4, 11); kii(x, t1) = ki(4, 11);
    kk(x + 1, t1) = k(5, 11); kii(x + 1, t1) = ki(5, 11);
    kk(x + 2, t1) = k(6, 11); kii(x + 2, t1) = ki(6, 11);
    x = x + 3 * g;

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```

t1 = t1 + 1;
%end
end
x = n + 1 ;
while x < n + g + 1
%for x = n + 1 : n + g
%%for x = n + 1; n + g
    kk(x, x) = k(2, 2) + k(11, 11); kii(x, x) = ki(2, 2) + ki(11, 11);
    x = x + 1;
%end
end

%%Right boundary
x = f3; tt3 = t3;
while x < f4 + 1
%for x = f3 : f4 Step 3 * g
    kk(x, t3) = k(1, 5) + k(10, 8); kii(x, t3) = ki(1, 5) + ki(10, 8);
    kk(x + 1, t3) = k(2, 5) + k(11, 8); kii(x + 1, t3) = ki(2, 5) + ki(11, 8);
    kk(x + 2, t3) = k(3, 5) + k(12, 8); kii(x + 2, t3) = ki(3, 5) + ki(12, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end
x = f3 + 3 * g; t3 = tt3;
while x < f4 + 1
%for x = f3 + 3 * g : f4 Step 3 * g
    kk(x, t3) = k(10, 5); kii(x, t3) = ki(10, 5);
    kk(x + 1, t3) = k(11, 5); kii(x + 1, t3) = ki(11, 5);
    kk(x + 2, t3) = k(12, 5); kii(x + 2, t3) = ki(12, 5);
    % kk(t3, t3 + 1) = k(5, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end

for x = n + g + 1 : n + 2 * g - 1
kk(x, x + 1) = k(5, 8); kii(x, x + 1) = ki(5, 8);
end

x = f3; t3 = tt3 + 1;
while x < f4 - 3 * g + 1
%for x = f3 : f4 - 3 * g Step 3 * g
    kk(x, t3) = k(1, 8); kii(x, t3) = ki(1, 8);
    kk(x + 1, t3) = k(2, 8); kii(x + 1, t3) = ki(2, 8);
    kk(x + 2, t3) = k(3, 8); kii(x + 2, t3) = ki(3, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end

%while x < n + 2 * g + 1
for x = n + g + 1 : n + 2 * g

```

APENDIX C

MATLAB Program Formulated for CSCS Boundary Condition

```
%for x = n + 1; n + g
    kk(x, x) = k(5, 5) + k(8, 8); kii(x, x) = ki(5, 5) + ki(8, 8);
    %x = x + 1;
%end
end
```

```
%Complete the symmetry
for x = 1 : nm
    for y = 1 : nm
        kk(y, x) = kk(x, y); kii(y, x) = kii(x, y);
    end
end
```

```
%Load Vector
```

```
x = 1;
while x < n + 1
%for x = 1 : n Step 3
    %%for x = 1 : n - 2 Step 3
        q(x) = 1;
        q(x + 1) = 0;
        q(x + 2) = 0;
        x = x + 3;
    %end
end
% left load
for x = n + 1 : n + g
    q(x) = 0.0833333333333333;
end
%right load
for x = n + g + 1 : n + 2 * g
    q(x) = -0.0833333333333333;
end
```

```
ncd = (n - 1) / 2;
kgv = inv(kk);
dd = kgv * transpose(q);
dc = dd(ncd) * 1000 / (1 + g) ^ 4;
kki = inv(kii) * kk;
ld = eig(kki) * (1 + g) ^ 4;
```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

%ReDim kx(12, 12), kxy(12, 12), ky(12, 12), k(12, 12), ki(12, 12)
for x = 1 : 12
for y = 1 : 12
k(x, y) = 0; kx(x, y) = 0; kxy(x, y) = 0; ky(x, y) = 0;
end
end
af = input('WHAT IS THE aspect ratio, b/a?'); af = af * 1;
kx(1, 1) = 4; kx(1, 2) = 2; kx(1, 3) = 0; kx(1, 4) = -4; kx(1, 5) = 2; kx(1, 6) = 0; kx(1, 7) = -2; kx(1, 8) = 1; kx(1, 9) = 0; kx(1, 10) = 2; kx(1, 11) = 1; kx(1, 12) = 0;
kx(2, 1) = 2; kx(2, 2) = 1.3333333; kx(2, 3) = 0; kx(2, 4) = -2; kx(2, 5) = 0.6666667; kx(2, 6) = 0; kx(2, 7) = -1; kx(2, 8) = 0.3333333; kx(2, 9) = 0; kx(2, 10) = 1; kx(2, 11) = 0.6666667; kx(2, 12) = 0;
kx(3, 1) = 0; kx(3, 2) = 0; kx(3, 3) = 0; kx(3, 4) = 0; kx(3, 5) = 0; kx(3, 6) = 0; kx(3, 7) = 0; kx(3, 8) = 0; kx(3, 9) = 0; kx(3, 10) = 0; kx(3, 11) = 0; kx(3, 12) = 0;
kx(4, 1) = -4; kx(4, 2) = -2; kx(4, 3) = 0; kx(4, 4) = 4; kx(4, 5) = -2; kx(4, 6) = 0; kx(4, 7) = 2; kx(4, 8) = -1; kx(4, 9) = 0; kx(4, 10) = -2; kx(4, 11) = -1; kx(4, 12) = 0;
kx(5, 1) = 2; kx(5, 2) = 0.6666667; kx(5, 3) = 0; kx(5, 4) = -2; kx(5, 5) = 1.3333333; kx(5, 6) = 0; kx(5, 7) = -1; kx(5, 8) = 0.6666667; kx(5, 9) = 0; kx(5, 10) = 1; kx(5, 11) = 0.3333333; kx(5, 12) = 0;
kx(6, 1) = 0; kx(6, 2) = 0; kx(6, 3) = 0; kx(6, 4) = 0; kx(6, 5) = 0; kx(6, 6) = 0; kx(6, 7) = 0; kx(6, 8) = 0; kx(6, 9) = 0; kx(6, 10) = 0; kx(6, 11) = 0; kx(6, 12) = 0;
kx(7, 1) = -2; kx(7, 2) = -1; kx(7, 3) = 0; kx(7, 4) = 2; kx(7, 5) = -1; kx(7, 6) = 0; kx(7, 7) = 4; kx(7, 8) = -2; kx(7, 9) = 0; kx(7, 10) = -4; kx(7, 11) = -2; kx(7, 12) = 0;
kx(8, 1) = 1; kx(8, 2) = 0.3333333; kx(8, 3) = 0; kx(8, 4) = -1; kx(8, 5) = 0.6666667; kx(8, 6) = 0; kx(8, 7) = -2; kx(8, 8) = 1.3333333; kx(8, 9) = 0; kx(8, 10) = 2; kx(8, 11) = 0.6666667; kx(8, 12) = 0;
kx(9, 1) = 0; kx(9, 2) = 0; kx(9, 3) = 0; kx(9, 4) = 0; kx(9, 5) = 0; kx(9, 6) = 0; kx(9, 7) = 0; kx(9, 8) = 0; kx(9, 9) = 0; kx(9, 10) = 0; kx(9, 11) = 0; kx(9, 12) = 0;
kx(10, 1) = 2; kx(10, 2) = 1; kx(10, 3) = 0; kx(10, 4) = -2; kx(10, 5) = 1; kx(10, 6) = 0; kx(10, 7) = -4; kx(10, 8) = 2; kx(10, 9) = 0; kx(10, 10) = 4; kx(10, 11) = 2; kx(10, 12) = 0;
kx(11, 1) = 1; kx(11, 2) = 0.6666667; kx(11, 3) = 0; kx(11, 4) = -1; kx(11, 5) = 0.3333333; kx(11, 6) = 0; kx(11, 7) = -2; kx(11, 8) = 0.6666667; kx(11, 9) = 0; kx(11, 10) = 2; kx(11, 11) = 1.3333333; kx(11, 12) = 0;
kx(12, 1) = 0; kx(12, 2) = 0; kx(12, 3) = 0; kx(12, 4) = 0; kx(12, 5) = 0; kx(12, 6) = 0; kx(12, 7) = 0; kx(12, 8) = 0; kx(12, 9) = 0; kx(12, 10) = 0; kx(12, 11) = 0; kx(12, 12) = 0;

ky(1, 1) = 4; ky(1, 2) = 0; ky(1, 3) = 2; ky(1, 4) = 2; ky(1, 5) = 0; ky(1, 6) = 1; ky(1, 7) = -2; ky(1, 8) = 0; ky(1, 9) = 1; ky(1, 10) = -4; ky(1, 11) = 0; ky(1, 12) = 2;
ky(2, 1) = 0; ky(2, 2) = 0; ky(2, 3) = 0; ky(2, 4) = 0; ky(2, 5) = 0; ky(2, 6) = 0; ky(2, 7) = 0; ky(2, 8) = 0; ky(2, 9) = 0; ky(2, 10) = 0; ky(2, 11) = 0; ky(2, 12) = 0;
ky(3, 1) = 2; ky(3, 2) = 0; ky(3, 3) = 1.3333333; ky(3, 4) = 1; ky(3, 5) = 0; ky(3, 6) = 0.6666667; ky(3, 7) = -1; ky(3, 8) = 0; ky(3, 9) = 0.3333333; ky(3, 10) = -2; ky(3, 11) = 0; ky(3, 12) = 0.6666667;
ky(4, 1) = 2; ky(4, 2) = 0; ky(4, 3) = 1; ky(4, 4) = 4; ky(4, 5) = 0; ky(4, 6) = 2; ky(4, 7) = -4; ky(4, 8) = 0; ky(4, 9) = 2; ky(4, 10) = -2; ky(4, 11) = 0; ky(4, 12) = 1;
ky(5, 1) = 0; ky(5, 2) = 0; ky(5, 3) = 0; ky(5, 4) = 0; ky(5, 5) = 0; ky(5, 6) = 0; ky(5, 7) = 0; ky(5, 8) = 0; ky(5, 9) = 0; ky(5, 10) = 0; ky(5, 11) = 0; ky(5, 12) = 0;
ky(6, 1) = 1; ky(6, 2) = 0; ky(6, 3) = 0.6666667; ky(6, 4) = 2; ky(6, 5) = 0; ky(6, 6) = 1.3333333; ky(6, 7) = -2; ky(6, 8) = 0; ky(6, 9) = 0.6666667; ky(6, 10) = -1; ky(6, 11) = 0; ky(6, 12) = 0.3333333;
ky(7, 1) = -2; ky(7, 2) = 0; ky(7, 3) = -1; ky(7, 4) = -4; ky(7, 5) = 0; ky(7, 6) = -2; ky(7, 7) = 4; ky(7, 8) = 0; ky(7, 9) = -2; ky(7, 10) = 2; ky(7, 11) = 0; ky(7, 12) = -1;
ky(8, 1) = 0; ky(8, 2) = 0; ky(8, 3) = 0; ky(8, 4) = 0; ky(8, 5) = 0; ky(8, 6) = 0; ky(8, 7) = 0; ky(8, 8) = 0; ky(8, 9) = 0; ky(8, 10) = 0; ky(8, 11) = 0; ky(8, 12) = 0;
ky(9, 1) = 1; ky(9, 2) = 0; ky(9, 3) = 0.3333333; ky(9, 4) = 2; ky(9, 5) = 0; ky(9, 6) = 0.6666667; ky(9, 7) = -2; ky(9, 8) = 0; ky(9, 9) = 1.3333333; ky(9, 10) = -1; ky(9, 11) = 0; ky(9, 12) = 0.6666667;

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

ky(10, 1) = -4; ky(10, 2) = 0; ky(10, 3) = -2; ky(10, 4) = -2; ky(10, 5) = 0; ky(10, 6) = -1; ky(10, 7) = 2; ky(10,
8) = 0; ky(10, 9) = -1; ky(10, 10) = 4; ky(10, 11) = 0; ky(10, 12) = -2;
ky(11, 1) = 0; ky(11, 2) = 0; ky(11, 3) = 0; ky(11, 4) = 0; ky(11, 5) = 0; ky(11, 6) = 0; ky(11, 7) = 0; ky(11, 8) =
0; ky(11, 9) = 0; ky(11, 10) = 0; ky(11, 11) = 0; ky(11, 12) = 0;
ky(12, 1) = 2; ky(12, 2) = 0; ky(12, 3) = 0.66666667; ky(12, 4) = 1; ky(12, 5) = 0; ky(12, 6) = 0.33333333; ky(12,
7) = -1; ky(12, 8) = 0; ky(12, 9) = 0.66666667; ky(12, 10) = -2; ky(12, 11) = 0; ky(12, 12) = 1.33333333;

kxy(1, 1) = 2.8; kxy(1, 2) = 0.2; kxy(1, 3) = 0.2; kxy(1, 4) = -2.8; kxy(1, 5) = 0.2; kxy(1, 6) = -0.2; kxy(1, 7) =
2.8; kxy(1, 8) = -0.2; kxy(1, 9) = -0.2; kxy(1, 10) = -2.8; kxy(1, 11) = -0.2; kxy(1, 12) = 0.2;
kxy(2, 1) = 0.2; kxy(2, 2) = 0.26666667; kxy(2, 3) = 0; kxy(2, 4) = -0.2; kxy(2, 5) = -0.06666667; kxy(2, 6) = 0;
kxy(2, 7) = 0.2; kxy(2, 8) = 0.06666667; kxy(2, 9) = 0; kxy(2, 10) = -0.2; kxy(2, 11) = -0.26666667; kxy(2, 12) =
0;
kxy(3, 1) = 0.2; kxy(3, 2) = 0; kxy(3, 3) = 0.26666667; kxy(3, 4) = -0.2; kxy(3, 5) = 0; kxy(3, 6) = -0.26666667;
kxy(3, 7) = 0.2; kxy(3, 8) = 0; kxy(3, 9) = 0.06666667; kxy(3, 10) = -0.2; kxy(3, 11) = 0; kxy(3, 12) = -
0.06666667;
kxy(4, 1) = -2.8; kxy(4, 2) = -0.2; kxy(4, 3) = -0.2; kxy(4, 4) = 2.8; kxy(4, 5) = -0.2; kxy(4, 6) = 0.2; kxy(4, 7) =
-2.8; kxy(4, 8) = 0.2; kxy(4, 9) = 0.2; kxy(4, 10) = 2.8; kxy(4, 11) = 0.2; kxy(4, 12) = -0.2;
kxy(5, 1) = 0.2; kxy(5, 2) = -0.06666667; kxy(5, 3) = 0; kxy(5, 4) = -0.2; kxy(5, 5) = 0.26666667; kxy(5, 6) = 0;
kxy(5, 7) = 0.2; kxy(5, 8) = -0.26666667; kxy(5, 9) = 0; kxy(5, 10) = -0.2; kxy(5, 11) = 0.06666667; kxy(5, 12) =
0;
kxy(6, 1) = -0.2; kxy(6, 2) = 0; kxy(6, 3) = -0.26666667; kxy(6, 4) = 0.2; kxy(6, 5) = 0; kxy(6, 6) = 0.26666667;
kxy(6, 7) = -0.2; kxy(6, 8) = 0; kxy(6, 9) = -0.06666667; kxy(6, 10) = 0.2; kxy(6, 11) = 0; kxy(6, 12) =
0.06666667;
kxy(7, 1) = 2.8; kxy(7, 2) = 0.2; kxy(7, 3) = 0.2; kxy(7, 4) = -2.8; kxy(7, 5) = 0.2; kxy(7, 6) = -0.2; kxy(7, 7) =
2.8; kxy(7, 8) = -0.2; kxy(7, 9) = -0.2; kxy(7, 10) = -2.8; kxy(7, 11) = -0.2; kxy(7, 12) = 0.2;
kxy(8, 1) = -0.2; kxy(8, 2) = 0.06666667; kxy(8, 3) = 0; kxy(8, 4) = 0.2; kxy(8, 5) = -0.26666667; kxy(8, 6) = 0;
kxy(8, 7) = -0.2; kxy(8, 8) = 0.26666667; kxy(8, 9) = 0; kxy(8, 10) = 0.2; kxy(8, 11) = -0.06666667; kxy(8, 12) =
0;
kxy(9, 1) = -0.2; kxy(9, 2) = 0; kxy(9, 3) = 0.06666667; kxy(9, 4) = 0.2; kxy(9, 5) = 0; kxy(9, 6) = -0.06666667;
kxy(9, 7) = -0.2; kxy(9, 8) = 0; kxy(9, 9) = 0.26666667; kxy(9, 10) = 0.2; kxy(9, 11) = 0; kxy(9, 12) = -
0.26666667;
kxy(10, 1) = -2.8; kxy(10, 2) = -0.2; kxy(10, 3) = -0.2; kxy(10, 4) = 2.8; kxy(10, 5) = -0.2; kxy(10, 6) = 0.2;
kxy(10, 7) = -2.8; kxy(10, 8) = 0.2; kxy(10, 9) = 0.2; kxy(10, 10) = 2.8; kxy(10, 11) = 0.2; kxy(10, 12) = -0.2;
kxy(11, 1) = -0.2; kxy(11, 2) = -0.26666667; kxy(11, 3) = 0; kxy(11, 4) = 0.2; kxy(11, 5) = 0.06666667; kxy(11, 6)
= 0; kxy(11, 7) = -0.2; kxy(11, 8) = -0.06666667; kxy(11, 9) = 0; kxy(11, 10) = 0.2; kxy(11, 11) = 0.26666667;
kxy(11, 12) = 0;
kxy(12, 1) = 0.2; kxy(12, 2) = 0; kxy(12, 3) = -0.06666667; kxy(12, 4) = -0.2; kxy(12, 5) = 0; kxy(12, 6) =
0.06666667; kxy(12, 7) = 0.2; kxy(12, 8) = 0; kxy(12, 9) = -0.26666667; kxy(12, 10) = -0.2; kxy(12, 11) = 0;
kxy(12, 12) = 0.26666667;

%' Inertia matrix
ki(1, 1) = 0.13706; ki(1, 2) = 0.01829; ki(1, 3) = 0.01829; ki(1, 4) = 0.04865; ki(1, 5) = -0.01087; ki(1, 6) =
0.0079; ki(1, 7) = 0.01563; ki(1, 8) = -0.0046; ki(1, 9) = -0.0046; ki(1, 10) = 0.04865; ki(1, 11) = 0.0079; ki(1,
12) = -0.01087;
ki(2, 1) = 0.01829; ki(2, 2) = 0.00317; ki(2, 3) = 0.0025; ki(2, 4) = 0.01087; ki(2, 5) = -0.00238; ki(2, 6) =
0.00167; ki(2, 7) = 0.0046; ki(2, 8) = -0.00119; ki(2, 9) = -0.00111; ki(2, 10) = 0.0079; ki(2, 11) = 0.00159;
ki(2, 12) = -0.00167;
ki(3, 1) = 0.01829; ki(3, 2) = 0.0025; ki(3, 3) = 0.00317; ki(3, 4) = 0.0079; ki(3, 5) = -0.00167; ki(3, 6) =
0.00159; ki(3, 7) = 0.0046; ki(3, 8) = -0.00111; ki(3, 9) = -0.00119; ki(3, 10) = 0.01087; ki(3, 11) = 0.00167;
ki(3, 12) = -0.00238;
ki(4, 1) = 0.04865; ki(4, 2) = 0.01087; ki(4, 3) = 0.0079; ki(4, 4) = 0.13706; ki(4, 5) = -0.01829; ki(4, 6) =
0.01829; ki(4, 7) = 0.04865; ki(4, 8) = -0.0079; ki(4, 9) = -0.01087; ki(4, 10) = 0.01563; ki(4, 11) = 0.0046;
ki(4, 12) = -0.0046;

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

ki(5, 1) = -0.01087; ki(5, 2) = -0.00238; ki(5, 3) = -0.00167; ki(5, 4) = -0.01829; ki(5, 5) = 0.00317; ki(5, 6) = -
0.0025; ki(5, 7) = -0.0079; ki(5, 8) = 0.00159; ki(5, 9) = 0.00167; ki(5, 10) = -0.0046; ki(5, 11) = -0.00119; ki(5,
12) = 0.00111;
ki(6, 1) = 0.0079; ki(6, 2) = 0.00167; ki(6, 3) = 0.00159; ki(6, 4) = 0.01829; ki(6, 5) = -0.0025; ki(6, 6) =
0.00317; ki(6, 7) = 0.01087; ki(6, 8) = -0.00167; ki(6, 9) = -0.00238; ki(6, 10) = 0.0046; ki(6, 11) = 0.00111;
ki(6, 12) = -0.00119;
ki(7, 1) = 0.01563; ki(7, 2) = 0.0046; ki(7, 3) = 0.0046; ki(7, 4) = 0.04865; ki(7, 5) = -0.0079; ki(7, 6) =
0.01087; ki(7, 7) = 0.13706; ki(7, 8) = -0.01829; ki(7, 9) = -0.01829; ki(7, 10) = 0.04865; ki(7, 11) = 0.01087;
ki(7, 12) = -0.0079;
ki(8, 1) = -0.0046; ki(8, 2) = -0.00119; ki(8, 3) = -0.00111; ki(8, 4) = -0.0079; ki(8, 5) = 0.00159; ki(8, 6) = -
0.00167; ki(8, 7) = -0.01829; ki(8, 8) = 0.00317; ki(8, 9) = 0.0025; ki(8, 10) = -0.01087; ki(8, 11) = -0.00238;
ki(8, 12) = 0.00167;
ki(9, 1) = -0.0046; ki(9, 2) = -0.00111; ki(9, 3) = -0.00119; ki(9, 4) = -0.01087; ki(9, 5) = 0.00167; ki(9, 6) = -
0.00238; ki(9, 7) = -0.01829; ki(9, 8) = 0.0025; ki(9, 9) = 0.00317; ki(9, 10) = -0.0079; ki(9, 11) = -0.00167;
ki(9, 12) = 0.00159;
ki(10, 1) = 0.04865; ki(10, 2) = 0.0079; ki(10, 3) = 0.01087; ki(10, 4) = 0.01563; ki(10, 5) = -0.0046; ki(10, 6) =
0.0046; ki(10, 7) = 0.04865; ki(10, 8) = -0.01087; ki(10, 9) = -0.0079; ki(10, 10) = 0.13706; ki(10, 11) =
0.01829; ki(10, 12) = -0.01829;
ki(11, 1) = 0.0079; ki(11, 2) = 0.00159; ki(11, 3) = 0.00167; ki(11, 4) = 0.0046; ki(11, 5) = -0.00119; ki(11, 6) =
0.00111; ki(11, 7) = 0.01087; ki(11, 8) = -0.00238; ki(11, 9) = -0.00167; ki(11, 10) = 0.01829; ki(11, 11) =
0.00317; ki(11, 12) = -0.0025;
ki(12, 1) = -0.01087; ki(12, 2) = -0.00167; ki(12, 3) = -0.00238; ki(12, 4) = -0.0046; ki(12, 5) = 0.00111; ki(12,
6) = -0.00119; ki(12, 7) = -0.0079; ki(12, 8) = 0.00167; ki(12, 9) = 0.00159; ki(12, 10) = -0.01829; ki(12, 11) =
-0.0025; ki(12, 12) = 0.00317;

```

```

for x = 1 : 12
for y = 1 : 12
k(x, y) = k(x, y) + kx(x, y) + kxy(x, y) / af ^ 2 + ky(x, y) / af ^ 4;
end
end

```

```

%TYP = InputBox("WHAT TYPE OF PLATE? 1 for SSSS, 2 for CCCC, 3 for CSCS, 4 for CCSS, 5 for
CCCC, 6 for CSSS"); TYP = TYP * 1

```

```

g = input("WHAT IS THE SIZE OF THE GRID 3,5,7 etc?"); g = g * 1;
n = 3 * (g ^ 2); NN = (g + 2 + g) * 2; m = 3; mm = 3 * g; nm = n + 3 * g;
%ReDim kk(nm, nm), q(nm), kki(nm, nm), kii(nm, nm)

```

```

for x = 1 : nm
for y = 1 : nm
kk(x, y) = 0; kii(x, y) = 0; kki(x, y) = 0;
end
end

```

```

% STIFFNESS for PURE W one node

```

```

x = 1;
%for x = 1 : n - 2 Step 3
while x < n - 1
kk(x, x) = 4 * k(1, 1); kii(x, x) = 4 * ki(1, 1);
x = x + 3;
end

```

```

% Pure Tx one node

```

```

x = 2;

```

APPENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

% for x = 2 : n - 1 Step 3
while x < n
    kk(x, x) = 4 * k(2, 2); kii(x, x) = 4 * ki(2, 2);
    x = x + 3;
end

% Pure Ty one node
x = 3;
% for x = 3 : n Step 3
while x < n + 1
    kk(x, x) = 4 * k(3, 3); kii(x, x) = 4 * ki(3, 3);
    x = x + 3;
end

% Pure W-Tx; W-Ty one node
x = 3;
%for x = 3 : n - 2 Step 3
while x < n - 1
    kk(x, x + 1) = 0; kii(x, x + 1) = 0;
    kk(x, x + 2) = 0; kii(x, x + 2) = 0;
    x = x + 3;
end

% Pure W two nodes
t = 1; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
        kk(t, t2) = k(1, 10) + k(4, 7); kii(t, t2) = ki(1, 10) + ki(4, 7);
        kk(t, t3) = k(1, 7); kii(t, t3) = ki(1, 7);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 2; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(1, 10) + k(4, 7); kii(t, t1) = ki(1, 10) + ki(4, 7);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 1; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
    t = t1; t1 = t1 + 3;
end

% Pure Tx two nodes
t = 2; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    end
end

```


APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

kk(t, t2) = k(2, 11) + k(5, 8); kii(t, t2) = ki(2, 11) + ki(5, 8);
kk(t, t3) = k(2, 8); kii(t, t3) = ki(2, 8);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g - 1; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(2, 11) + k(5, 8); kii(t, t1) = ki(2, 11) + ki(5, 8);
    t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 2; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    t = t1; t1 = t1 + 3;
end

```

```

% Pure Ty two nodes
t = 3; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
        kk(t, t2) = k(3, 12) + k(6, 9); kii(t, t2) = ki(3, 12) + ki(6, 9);
        kk(t, t3) = k(3, 9); kii(t, t3) = ki(3, 9);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(3, 12) + k(6, 9); kii(t, t1) = ki(3, 12) + ki(6, 9);
    t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 3; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    t = t1; t1 = t1 + 3;
end

```

```

% Pure W two nodes back down
t = 4; t1 = t + 3 * (g - 1); t2 = t1 + 1; t3 = t1 + 2;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(4, 10); kii(t, t1) = ki(4, 10);
    end
end

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

kk(t + 1, t1) = k(5, 10); kii(t + 1, t1) = ki(5, 10);
kk(t + 2, t1) = k(6, 10); kii(t + 2, t1) = ki(6, 10);

kk(t + 1, t2) = k(5, 11); kii(t + 1, t2) = ki(5, 11);
kk(t + 2, t3) = k(6, 12); kii(t + 2, t3) = ki(6, 12);

kk(t, t2) = k(4, 11); kii(t, t2) = ki(4, 11);
kk(t, t3) = k(4, 12); kii(t, t3) = ki(4, 12);
t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

% W-Tx W-Ty two nodes
t = 1; tx1 = t + 4; ty1 = t + 5; tx2 = t + 3 * g + 1; ty2 = t + 3 * g + 2;
tx3 = tx2 + 3; ty3 = ty2 + 3;
tt = t; ttx1 = tx1; tty1 = ty1; ttx2 = tx2; tty2 = ty2;
ttx3 = tx3; tty3 = ty3;

for x = 1 : g - 1
for y = 1 : g - 1
kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
kk(t + 1, tx1 - 1) = k(2, 4) + k(11, 7); kii(t + 1, tx1 - 1) = ki(2, 4) + ki(11, 7);
kk(t + 2, tx1 - 1) = k(3, 4) + k(12, 7); kii(t + 2, tx1 - 1) = ki(3, 4) + ki(12, 7);

kk(t, tx2) = k(1, 11) + k(4, 8); kii(t, tx2) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx2 - 1) = k(2, 10) + k(5, 7); kii(t + 1, tx2 - 1) = ki(2, 10) + ki(5, 7);
kk(t + 2, tx2 - 1) = k(3, 10) + k(6, 7); kii(t + 2, tx2 - 1) = ki(3, 10) + ki(6, 7);

kk(t, tx3) = k(1, 8); kii(t, tx3) = ki(1, 8);
kk(t + 1, tx3 - 1) = k(2, 7); kii(t + 1, tx3 - 1) = ki(2, 7);
kk(t + 2, tx3 - 1) = k(3, 7); kii(t + 2, tx3 - 1) = ki(3, 7);

kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
kk(t, ty2) = k(1, 12) + k(4, 9); kii(t, ty2) = ki(1, 12) + ki(4, 9);
kk(t, ty3) = k(1, 9); kii(t, ty3) = ki(1, 9);
t = t + 3; tx1 = tx1 + 3; tx2 = tx2 + 3; tx3 = tx3 + 3;
ty1 = ty1 + 3; ty2 = ty2 + 3; ty3 = ty3 + 3;
end
t = tt + 3 * g; tx1 = ttx1 + 3 * g; tx2 = ttx2 + 3 * g; tx3 = ttx3 + 3 * g;
ty1 = tty1 + 3 * g; ty2 = tty2 + 3 * g; ty3 = tty3 + 3 * g;
tt = t; ttx1 = tx1; ttx2 = tx2; ttx3 = tx3;
tty1 = ty1; tty2 = ty2; tty3 = ty3;
end

t = 3 * g - 2; tx1 = t + 3 * g + 1; ty1 = t + 3 * g + 2;
for x = 1 : g - 1
kk(t, tx1) = k(1, 11) + k(4, 8); kii(t, tx1) = ki(1, 11) + ki(4, 8);
kk(t + 1, tx1 - 1) = k(5, 7) + k(2, 10); kii(t + 1, tx1 - 1) = ki(5, 7) + ki(2, 10);
kk(t + 2, tx1 - 1) = k(6, 7) + k(3, 10); kii(t + 2, tx1 - 1) = ki(6, 7) + ki(3, 10);

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

kk(t, ty1) = k(1, 12) + k(4, 9); kii(t, ty1) = ki(1, 12) + ki(4, 9);
t = tx1 - 1; tx1 = tx1 + 3 * g; ty1 = ty1 + 3 * g;
end

t = n - 3 * g + 1; tx1 = t + 4; ty1 = t + 5;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(11, 7) + k(2, 4); kii(t + 1, tx1 - 1) = ki(11, 7) + ki(2, 4);
    kk(t + 2, tx1 - 1) = k(12, 7) + k(3, 4); kii(t + 2, tx1 - 1) = ki(12, 7) + ki(3, 4);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    t = tx1 - 1; tx1 = tx1 + 3; ty1 = ty1 + 3;
end

% Boundary conditions
t1 = n + 1; t2 = n + g; t3 = n + g + 1; t4 = n + 2 * g; t5 = n + 2 * g + 1;
t6 = n + 3 * g; t7 = n + 3 * g + 1; t8 = n + 4 * g;
f1 = 1; f2 = n - 3 * g + 1; f3 = 3 * g - 2; f4 = n - 2;
%left boundary
x = 1; tt1 = t1;
while x < f2 + 1
%for x = 1 : f2 Step 3 * g
    kk(x, t1) = k(4, 2) + k(7, 11); kii(x, t1) = ki(4, 2) + ki(7, 11);
    kk(x + 1, t1) = k(5, 2) + k(8, 11); kii(x + 1, t1) = ki(5, 2) + ki(8, 11);
    kk(x + 2, t1) = k(6, 2) + k(9, 11); kii(x + 2, t1) = ki(6, 2) + ki(9, 11);
    x = x + 3 * g;
    t1 = t1 + 1;
%end
end
x = 3 * g + 1; t1 = tt1;
while x < f2 + 1
%for x = 3 * g + 1 : f2 Step 3 * g
    kk(x, t1) = k(7, 2); kii(x, t1) = ki(7, 2);
    kk(x + 1, t1) = k(8, 2); kii(x + 1, t1) = ki(8, 2);
    kk(x + 2, t1) = k(9, 2); kii(x + 2, t1) = ki(9, 2);
    %kk(t1, t1 + 1) = k(2, 11);
    x = x + 3 * g;
    t1 = t1 + 1;
%end
end

for x = n + 1 : n + g - 1
kk(x, x + 1) = k(2, 11); kii(x, x + 1) = ki(2, 11);
end

x = 1; t1 = tt1 + 1;
while x < f2 - 3 * g + 1
%for x = 1 : f2 - 3 * g Step 3 * g
    kk(x, t1) = k(4, 11); kii(x, t1) = ki(4, 11);
    kk(x + 1, t1) = k(5, 11); kii(x + 1, t1) = ki(5, 11);
    kk(x + 2, t1) = k(6, 11); kii(x + 2, t1) = ki(6, 11);

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

    x = x + 3 * g;
t1 = t1 + 1;
%end
end
x = n + 1 ;
while x < n + g + 1
%for x = n + 1 : n + g
%%for x = n + 1; n + g
    kk(x, x) = k(2, 2) + k(11, 11); kii(x, x) = ki(2, 2) + ki(11, 11);
    x = x + 1;
%end
end

%%Right boundary
x = f3; tt3 = t3;
while x < f4 + 1
%for x = f3 : f4 Step 3 * g
    kk(x, t3) = k(1, 5) + k(10, 8); kii(x, t3) = ki(1, 5) + ki(10, 8);
    kk(x + 1, t3) = k(2, 5) + k(11, 8); kii(x + 1, t3) = ki(2, 5) + ki(11, 8);
    kk(x + 2, t3) = k(3, 5) + k(12, 8); kii(x + 2, t3) = ki(3, 5) + ki(12, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end
x = f3 + 3 * g; t3 = tt3;
while x < f4 + 1
%for x = f3 + 3 * g : f4 Step 3 * g
    kk(x, t3) = k(10, 5); kii(x, t3) = ki(10, 5);
    kk(x + 1, t3) = k(11, 5); kii(x + 1, t3) = ki(11, 5);
    kk(x + 2, t3) = k(12, 5); kii(x + 2, t3) = ki(12, 5);
    % kk(t3, t3 + 1) = k(5, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end

for x = n + g + 1 : n + 2 * g - 1
kk(x, x + 1) = k(5, 8); kii(x, x + 1) = ki(5, 8);
end

x = f3; t3 = tt3 + 1;
while x < f4 - 3 * g + 1
%for x = f3 : f4 - 3 * g Step 3 * g
    kk(x, t3) = k(1, 8); kii(x, t3) = ki(1, 8);
    kk(x + 1, t3) = k(2, 8); kii(x + 1, t3) = ki(2, 8);
    kk(x + 2, t3) = k(3, 8); kii(x + 2, t3) = ki(3, 8);
    x = x + 3 * g;
t3 = t3 + 1;
%end
end

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

% while x < n + 2 * g + 1
for x = n + g + 1 : n + 2 * g
%for x = n + 1; n + g
    kk(x, x) = k(5, 5) + k(8, 8); kii(x, x) = ki(5, 5) + ki(8, 8);
    %x = x + 1;
%end
end

%% Bot:m boundary
x = f2; t5 = t5;
while x < f4 + 1
%for x = f2 : f4 Step 3
    kk(x, t5) = k(1, 12) + k(4, 9); kii(x, t5) = ki(1, 12) + ki(4, 9);
    kk(x + 1, t5) = k(2, 12) + k(5, 9); kii(x + 1, t5) = ki(2, 12) + ki(5, 9);
    kk(x + 2, t5) = k(3, 12) + k(6, 9); kii(x + 2, t5) = ki(3, 12) + ki(6, 9);
    x = x + 3;
t5 = t5 + 1;
%end
end

    x = f2 + 3; t5 = t5;
while x < f4 + 1
%for x = f2 + 3 : f4 Step 3

    kk(x, t5) = k(4, 12); kii(x, t5) = ki(4, 12);
    kk(x + 1, t5) = k(5, 12); kii(x + 1, t5) = ki(5, 12);
    kk(x + 2, t5) = k(6, 12); kii(x + 2, t5) = ki(6, 12);
    % kk(t7, t7 + 1) = k(12, 9);
    x = x + 3;
t5 = t5 + 1;
%end
end

for x = n + 2 * g + 1 : nm - 1
kk(x, x + 1) = k(9, 12); kii(x, x + 1) = ki(9, 12);
end

x = f2; t5 = t5 + 1;
while x < f4 - 2
%for x = f2 : f4 - 3 Step 3
    kk(x, t5) = k(1, 9); kii(x, t5) = ki(1, 9);
    kk(x + 1, t5) = k(2, 9); kii(x + 1, t5) = ki(2, 9);
    kk(x + 2, t5) = k(3, 9); kii(x + 2, t5) = ki(3, 9);
    x = x + 3;
t5 = t5 + 1;
%end
end

% while x < nm + 1
for x = n + 2 * g + 1 : nm
%%for x = n + 1; n + g

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

kk(x, x) = k(9, 9) + k(12, 12); kii(x, x) = ki(9, 9) + ki(12, 12);
% x = x + 1;
%end
end

```

```

%Complete the symmetry
for x = 1 : nm
    for y = 1 : nm
        kk(y, x) = kk(x, y); kii(y, x) = kii(x, y);
    end
end

```

%Load Vector

```

x = 1;
while x < n + 1
%for x = 1 : n Step 3
    %%for x = 1 : n - 2 Step 3
        q(x) = 1;
        q(x + 1) = 0;
        q(x + 2) = 0;
        x = x + 3;
    %end
end

```

% left load

```

for x = n + 1 : n + g
    q(x) = 0.0833333333333333;
end

```

%right load

```

for x = n + g + 1 : n + 2 * g
    q(x) = -0.0833333333333333;
end

```

% bottom load

```

for x = n + 2 * g + 1 : n + 3 * g
    q(x) = -0.0833333333333333;
end

```

```

ncd = (n - 1) / 2;
kgv = inv(kk);
dd = kgv * transpose(q);
dc = dd(ncd) * 1000 / (1 + g) ^ 4;
kki = inv(kii) * kk;
ld = eig(kki) * (1 + g) ^ 4;
% end
% end

```

```

% dd(ncd) = dd(ncd) / (1 + g) ^ 4
% ReDim kgv(nm, 2 * nm), kgvr(nm, 2 * nm)
% for x = 1 : nm
% for y = 1 : 2 * nm

```

APENDIX D

MATLAB Program Formulated for CSSS Boundary Condition

```

% kgv(x, y) = 0; kgvr(x, y) = 0;
% end
% end

%for x = 1 : nm
% kgv(x, x + nm) = 1
% end

%for x = 1 : nm
% for y = 1 : nm
% kgv(x, y) = kk(x, y);
%end
% end

% for i = 1 : nm
% owuss = kgv(i, i)
% for j = 1 : 2 * nm
% kgv(i, j) = kgv(i, j) / owuss
% end
% for j = 1 : nm
% If (j = i) Then Go: 1460
%owuss = kgv(j, i)
% for z = 1 : 2 * nm
% kgv(j, z) = kgv(j, z) - owuss * kgv(i, z)
% Next z
%1460 end
% end

% for x = 1 : nm
% for y = 1 : nm
% kgv(x, y) = kgv(x, y + nm)
% end
% end

%this is the end of inversion

% displacem
% *** calculate global nodal displacements for the unrestrained nodes *****
% ncd = (n - 1) / 2 % number of central node
%ReDim dd(nm + 1)
% for x = 1 : nm
% dd(x) = 0
% end
% for x = 1 : nm
% for y = 1 : nm
% dd(x) = dd(x) + kgv(x, y) * q(y)
% end
% end

% dd(ncd) = dd(ncd) / (1 + g) ^ 4

% Text1.Text = Text1.Text + ("central deflection ") & vbCrLf

```

```

% Text1.Text = Text1.Text + CStr(format(dd(ncd), "0.000000#")) & vbCrLf
% Text1.Text = Text1.Text + CStr(format(dd(13), "0.000000#")) & vbCrLf

% Resonating matrix
% for i = 1 : nm
% for j = 1 : nm
% for l = 1 : nm
% kki(i, j) = kki(i, j) + kgv(i, l) * kii(l, j)
% Next l
% end
% end

% for x = 1 : nm
% for y = 1 : nm
% Text1.Text = Text1.Text + CStr(format(kki(x, y), "0.000000#")) & vbTab
% end
% Text1.Text = Text1.Text + (" ") & vbCrLf
% end

```


APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

%ReDim kx(12, 12), kxy(12, 12), ky(12, 12), k(12, 12), ki(12, 12)
for x = 1 : 12
for y = 1 : 12
k(x, y) = 0; kx(x, y) = 0; kxy(x, y) = 0; ky(x, y) = 0;
end
end
af = input('WHAT IS THE aspect ratio, b/a?'); af = af * 1;
kx(1, 1) = 4; kx(1, 2) = 2; kx(1, 3) = 0; kx(1, 4) = -4; kx(1, 5) = 2; kx(1, 6) = 0; kx(1, 7) = -2; kx(1, 8) = 1; kx(1, 9) = 0; kx(1, 10) = 2; kx(1, 11) = 1; kx(1, 12) = 0;
kx(2, 1) = 2; kx(2, 2) = 1.3333333; kx(2, 3) = 0; kx(2, 4) = -2; kx(2, 5) = 0.66666667; kx(2, 6) = 0; kx(2, 7) = -1; kx(2, 8) = 0.3333333; kx(2, 9) = 0; kx(2, 10) = 1; kx(2, 11) = 0.66666667; kx(2, 12) = 0;
kx(3, 1) = 0; kx(3, 2) = 0; kx(3, 3) = 0; kx(3, 4) = 0; kx(3, 5) = 0; kx(3, 6) = 0; kx(3, 7) = 0; kx(3, 8) = 0; kx(3, 9) = 0; kx(3, 10) = 0; kx(3, 11) = 0; kx(3, 12) = 0;
kx(4, 1) = -4; kx(4, 2) = -2; kx(4, 3) = 0; kx(4, 4) = 4; kx(4, 5) = -2; kx(4, 6) = 0; kx(4, 7) = 2; kx(4, 8) = -1; kx(4, 9) = 0; kx(4, 10) = -2; kx(4, 11) = -1; kx(4, 12) = 0;
kx(5, 1) = 2; kx(5, 2) = 0.66666667; kx(5, 3) = 0; kx(5, 4) = -2; kx(5, 5) = 1.3333333; kx(5, 6) = 0; kx(5, 7) = -1; kx(5, 8) = 0.66666667; kx(5, 9) = 0; kx(5, 10) = 1; kx(5, 11) = 0.3333333; kx(5, 12) = 0;
kx(6, 1) = 0; kx(6, 2) = 0; kx(6, 3) = 0; kx(6, 4) = 0; kx(6, 5) = 0; kx(6, 6) = 0; kx(6, 7) = 0; kx(6, 8) = 0; kx(6, 9) = 0; kx(6, 10) = 0; kx(6, 11) = 0; kx(6, 12) = 0;
kx(7, 1) = -2; kx(7, 2) = -1; kx(7, 3) = 0; kx(7, 4) = 2; kx(7, 5) = -1; kx(7, 6) = 0; kx(7, 7) = 4; kx(7, 8) = -2; kx(7, 9) = 0; kx(7, 10) = -4; kx(7, 11) = -2; kx(7, 12) = 0;
kx(8, 1) = 1; kx(8, 2) = 0.3333333; kx(8, 3) = 0; kx(8, 4) = -1; kx(8, 5) = 0.66666667; kx(8, 6) = 0; kx(8, 7) = -2; kx(8, 8) = 1.3333333; kx(8, 9) = 0; kx(8, 10) = 2; kx(8, 11) = 0.66666667; kx(8, 12) = 0;
kx(9, 1) = 0; kx(9, 2) = 0; kx(9, 3) = 0; kx(9, 4) = 0; kx(9, 5) = 0; kx(9, 6) = 0; kx(9, 7) = 0; kx(9, 8) = 0; kx(9, 9) = 0; kx(9, 10) = 0; kx(9, 11) = 0; kx(9, 12) = 0;
kx(10, 1) = 2; kx(10, 2) = 1; kx(10, 3) = 0; kx(10, 4) = -2; kx(10, 5) = 1; kx(10, 6) = 0; kx(10, 7) = -4; kx(10, 8) = 2; kx(10, 9) = 0; kx(10, 10) = 4; kx(10, 11) = 2; kx(10, 12) = 0;
kx(11, 1) = 1; kx(11, 2) = 0.66666667; kx(11, 3) = 0; kx(11, 4) = -1; kx(11, 5) = 0.3333333; kx(11, 6) = 0; kx(11, 7) = -2; kx(11, 8) = 0.66666667; kx(11, 9) = 0; kx(11, 10) = 2; kx(11, 11) = 1.3333333; kx(11, 12) = 0;
kx(12, 1) = 0; kx(12, 2) = 0; kx(12, 3) = 0; kx(12, 4) = 0; kx(12, 5) = 0; kx(12, 6) = 0; kx(12, 7) = 0; kx(12, 8) = 0; kx(12, 9) = 0; kx(12, 10) = 0; kx(12, 11) = 0; kx(12, 12) = 0;

ky(1, 1) = 4; ky(1, 2) = 0; ky(1, 3) = 2; ky(1, 4) = 2; ky(1, 5) = 0; ky(1, 6) = 1; ky(1, 7) = -2; ky(1, 8) = 0; ky(1, 9) = 1; ky(1, 10) = -4; ky(1, 11) = 0; ky(1, 12) = 2;
ky(2, 1) = 0; ky(2, 2) = 0; ky(2, 3) = 0; ky(2, 4) = 0; ky(2, 5) = 0; ky(2, 6) = 0; ky(2, 7) = 0; ky(2, 8) = 0; ky(2, 9) = 0; ky(2, 10) = 0; ky(2, 11) = 0; ky(2, 12) = 0;
ky(3, 1) = 2; ky(3, 2) = 0; ky(3, 3) = 1.3333333; ky(3, 4) = 1; ky(3, 5) = 0; ky(3, 6) = 0.66666667; ky(3, 7) = -1; ky(3, 8) = 0; ky(3, 9) = 0.3333333; ky(3, 10) = -2; ky(3, 11) = 0; ky(3, 12) = 0.66666667;
ky(4, 1) = 2; ky(4, 2) = 0; ky(4, 3) = 1; ky(4, 4) = 4; ky(4, 5) = 0; ky(4, 6) = 2; ky(4, 7) = -4; ky(4, 8) = 0; ky(4, 9) = 2; ky(4, 10) = -2; ky(4, 11) = 0; ky(4, 12) = 1;
ky(5, 1) = 0; ky(5, 2) = 0; ky(5, 3) = 0; ky(5, 4) = 0; ky(5, 5) = 0; ky(5, 6) = 0; ky(5, 7) = 0; ky(5, 8) = 0; ky(5, 9) = 0; ky(5, 10) = 0; ky(5, 11) = 0; ky(5, 12) = 0;
ky(6, 1) = 1; ky(6, 2) = 0; ky(6, 3) = 0.66666667; ky(6, 4) = 2; ky(6, 5) = 0; ky(6, 6) = 1.3333333; ky(6, 7) = -2; ky(6, 8) = 0; ky(6, 9) = 0.66666667; ky(6, 10) = -1; ky(6, 11) = 0; ky(6, 12) = 0.3333333;
ky(7, 1) = -2; ky(7, 2) = 0; ky(7, 3) = -1; ky(7, 4) = -4; ky(7, 5) = 0; ky(7, 6) = -2; ky(7, 7) = 4; ky(7, 8) = 0; ky(7, 9) = -2; ky(7, 10) = 2; ky(7, 11) = 0; ky(7, 12) = -1;
ky(8, 1) = 0; ky(8, 2) = 0; ky(8, 3) = 0; ky(8, 4) = 0; ky(8, 5) = 0; ky(8, 6) = 0; ky(8, 7) = 0; ky(8, 8) = 0; ky(8, 9) = 0; ky(8, 10) = 0; ky(8, 11) = 0; ky(8, 12) = 0;
ky(9, 1) = 1; ky(9, 2) = 0; ky(9, 3) = 0.3333333; ky(9, 4) = 2; ky(9, 5) = 0; ky(9, 6) = 0.66666667; ky(9, 7) = -2; ky(9, 8) = 0; ky(9, 9) = 1.3333333; ky(9, 10) = -1; ky(9, 11) = 0; ky(9, 12) = 0.66666667;
ky(10, 1) = -4; ky(10, 2) = 0; ky(10, 3) = -2; ky(10, 4) = -2; ky(10, 5) = 0; ky(10, 6) = -1; ky(10, 7) = 2; ky(10, 8) = 0; ky(10, 9) = -1; ky(10, 10) = 4; ky(10, 11) = 0; ky(10, 12) = -2;

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

ky(11, 1) = 0; ky(11, 2) = 0; ky(11, 3) = 0; ky(11, 4) = 0; ky(11, 5) = 0; ky(11, 6) = 0; ky(11, 7) = 0; ky(11, 8) =
0; ky(11, 9) = 0; ky(11, 10) = 0; ky(11, 11) = 0; ky(11, 12) = 0;
ky(12, 1) = 2; ky(12, 2) = 0; ky(12, 3) = 0.6666667; ky(12, 4) = 1; ky(12, 5) = 0; ky(12, 6) = 0.3333333; ky(12,
7) = -1; ky(12, 8) = 0; ky(12, 9) = 0.6666667; ky(12, 10) = -2; ky(12, 11) = 0; ky(12, 12) = 1.3333333;
kxy(1, 1) = 2.8; kxy(1, 2) = 0.2; kxy(1, 3) = 0.2; kxy(1, 4) = -2.8; kxy(1, 5) = 0.2; kxy(1, 6) = -0.2; kxy(1, 7) =
2.8; kxy(1, 8) = -0.2; kxy(1, 9) = -0.2; kxy(1, 10) = -2.8; kxy(1, 11) = -0.2; kxy(1, 12) = 0.2;
kxy(2, 1) = 0.2; kxy(2, 2) = 0.2666667; kxy(2, 3) = 0; kxy(2, 4) = -0.2; kxy(2, 5) = -0.0666667; kxy(2, 6) = 0;
kxy(2, 7) = 0.2; kxy(2, 8) = 0.0666667; kxy(2, 9) = 0; kxy(2, 10) = -0.2; kxy(2, 11) = -0.2666667; kxy(2, 12) =
0;
kxy(3, 1) = 0.2; kxy(3, 2) = 0; kxy(3, 3) = 0.2666667; kxy(3, 4) = -0.2; kxy(3, 5) = 0; kxy(3, 6) = -0.2666667;
kxy(3, 7) = 0.2; kxy(3, 8) = 0; kxy(3, 9) = 0.0666667; kxy(3, 10) = -0.2; kxy(3, 11) = 0; kxy(3, 12) = -
0.0666667;
kxy(4, 1) = -2.8; kxy(4, 2) = -0.2; kxy(4, 3) = -0.2; kxy(4, 4) = 2.8; kxy(4, 5) = -0.2; kxy(4, 6) = 0.2; kxy(4, 7) =
-2.8; kxy(4, 8) = 0.2; kxy(4, 9) = 0.2; kxy(4, 10) = 2.8; kxy(4, 11) = 0.2; kxy(4, 12) = -0.2;
kxy(5, 1) = 0.2; kxy(5, 2) = -0.0666667; kxy(5, 3) = 0; kxy(5, 4) = -0.2; kxy(5, 5) = 0.2666667; kxy(5, 6) = 0;
kxy(5, 7) = 0.2; kxy(5, 8) = -0.2666667; kxy(5, 9) = 0; kxy(5, 10) = -0.2; kxy(5, 11) = 0.0666667; kxy(5, 12) =
0;
kxy(6, 1) = -0.2; kxy(6, 2) = 0; kxy(6, 3) = -0.2666667; kxy(6, 4) = 0.2; kxy(6, 5) = 0; kxy(6, 6) = 0.2666667;
kxy(6, 7) = -0.2; kxy(6, 8) = 0; kxy(6, 9) = -0.0666667; kxy(6, 10) = 0.2; kxy(6, 11) = 0; kxy(6, 12) =
0.0666667;
kxy(7, 1) = 2.8; kxy(7, 2) = 0.2; kxy(7, 3) = 0.2; kxy(7, 4) = -2.8; kxy(7, 5) = 0.2; kxy(7, 6) = -0.2; kxy(7, 7) =
2.8; kxy(7, 8) = -0.2; kxy(7, 9) = -0.2; kxy(7, 10) = -2.8; kxy(7, 11) = -0.2; kxy(7, 12) = 0.2;
kxy(8, 1) = -0.2; kxy(8, 2) = 0.0666667; kxy(8, 3) = 0; kxy(8, 4) = 0.2; kxy(8, 5) = -0.2666667; kxy(8, 6) = 0;
kxy(8, 7) = -0.2; kxy(8, 8) = 0.2666667; kxy(8, 9) = 0; kxy(8, 10) = 0.2; kxy(8, 11) = -0.0666667; kxy(8, 12) =
0;
kxy(9, 1) = -0.2; kxy(9, 2) = 0; kxy(9, 3) = 0.0666667; kxy(9, 4) = 0.2; kxy(9, 5) = 0; kxy(9, 6) = -0.0666667;
kxy(9, 7) = -0.2; kxy(9, 8) = 0; kxy(9, 9) = 0.2666667; kxy(9, 10) = 0.2; kxy(9, 11) = 0; kxy(9, 12) = -
0.2666667;
kxy(10, 1) = -2.8; kxy(10, 2) = -0.2; kxy(10, 3) = -0.2; kxy(10, 4) = 2.8; kxy(10, 5) = -0.2; kxy(10, 6) = 0.2;
kxy(10, 7) = -2.8; kxy(10, 8) = 0.2; kxy(10, 9) = 0.2; kxy(10, 10) = 2.8; kxy(10, 11) = 0.2; kxy(10, 12) = -0.2;
kxy(11, 1) = -0.2; kxy(11, 2) = -0.2666667; kxy(11, 3) = 0; kxy(11, 4) = 0.2; kxy(11, 5) = 0.0666667; kxy(11, 6)
= 0; kxy(11, 7) = -0.2; kxy(11, 8) = -0.0666667; kxy(11, 9) = 0; kxy(11, 10) = 0.2; kxy(11, 11) = 0.2666667;
kxy(11, 12) = 0;
kxy(12, 1) = 0.2; kxy(12, 2) = 0; kxy(12, 3) = -0.0666667; kxy(12, 4) = -0.2; kxy(12, 5) = 0; kxy(12, 6) =
0.0666667; kxy(12, 7) = 0.2; kxy(12, 8) = 0; kxy(12, 9) = -0.2666667; kxy(12, 10) = -0.2; kxy(12, 11) = 0;
kxy(12, 12) = 0.2666667;

```

%' Inertia matrix

```

ki(1, 1) = 0.13706; ki(1, 2) = 0.01829; ki(1, 3) = 0.01829; ki(1, 4) = 0.04865; ki(1, 5) = -0.01087; ki(1, 6) =
0.0079; ki(1, 7) = 0.01563; ki(1, 8) = -0.0046; ki(1, 9) = -0.0046; ki(1, 10) = 0.04865; ki(1, 11) = 0.0079; ki(1,
12) = -0.01087;
ki(2, 1) = 0.01829; ki(2, 2) = 0.00317; ki(2, 3) = 0.0025; ki(2, 4) = 0.01087; ki(2, 5) = -0.00238; ki(2, 6) =
0.00167; ki(2, 7) = 0.0046; ki(2, 8) = -0.00119; ki(2, 9) = -0.00111; ki(2, 10) = 0.0079; ki(2, 11) = 0.00159;
ki(2, 12) = -0.00167;
ki(3, 1) = 0.01829; ki(3, 2) = 0.0025; ki(3, 3) = 0.00317; ki(3, 4) = 0.0079; ki(3, 5) = -0.00167; ki(3, 6) =
0.00159; ki(3, 7) = 0.0046; ki(3, 8) = -0.00111; ki(3, 9) = -0.00119; ki(3, 10) = 0.01087; ki(3, 11) = 0.00167;
ki(3, 12) = -0.00238;
ki(4, 1) = 0.04865; ki(4, 2) = 0.01087; ki(4, 3) = 0.0079; ki(4, 4) = 0.13706; ki(4, 5) = -0.01829; ki(4, 6) =
0.01829; ki(4, 7) = 0.04865; ki(4, 8) = -0.0079; ki(4, 9) = -0.01087; ki(4, 10) = 0.01563; ki(4, 11) = 0.0046;
ki(4, 12) = -0.0046;

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

ki(5, 1) = -0.01087; ki(5, 2) = -0.00238; ki(5, 3) = -0.00167; ki(5, 4) = -0.01829; ki(5, 5) = 0.00317; ki(5, 6) = -
0.0025; ki(5, 7) = -0.0079; ki(5, 8) = 0.00159; ki(5, 9) = 0.00167; ki(5, 10) = -0.0046; ki(5, 11) = -0.00119; ki(5,
12) = 0.00111;
ki(6, 1) = 0.0079; ki(6, 2) = 0.00167; ki(6, 3) = 0.00159; ki(6, 4) = 0.01829; ki(6, 5) = -0.0025; ki(6, 6) =
0.00317; ki(6, 7) = 0.01087; ki(6, 8) = -0.00167; ki(6, 9) = -0.00238; ki(6, 10) = 0.0046; ki(6, 11) = 0.00111;
ki(6, 12) = -0.00119;
ki(7, 1) = 0.01563; ki(7, 2) = 0.0046; ki(7, 3) = 0.0046; ki(7, 4) = 0.04865; ki(7, 5) = -0.0079; ki(7, 6) =
0.01087; ki(7, 7) = 0.13706; ki(7, 8) = -0.01829; ki(7, 9) = -0.01829; ki(7, 10) = 0.04865; ki(7, 11) = 0.01087;
ki(7, 12) = -0.0079;
ki(8, 1) = -0.0046; ki(8, 2) = -0.00119; ki(8, 3) = -0.00111; ki(8, 4) = -0.0079; ki(8, 5) = 0.00159; ki(8, 6) = -
0.00167; ki(8, 7) = -0.01829; ki(8, 8) = 0.00317; ki(8, 9) = 0.0025; ki(8, 10) = -0.01087; ki(8, 11) = -0.00238;
ki(8, 12) = 0.00167;
ki(9, 1) = -0.0046; ki(9, 2) = -0.00111; ki(9, 3) = -0.00119; ki(9, 4) = -0.01087; ki(9, 5) = 0.00167; ki(9, 6) = -
0.00238; ki(9, 7) = -0.01829; ki(9, 8) = 0.0025; ki(9, 9) = 0.00317; ki(9, 10) = -0.0079; ki(9, 11) = -0.00167;
ki(9, 12) = 0.00159;
ki(10, 1) = 0.04865; ki(10, 2) = 0.0079; ki(10, 3) = 0.01087; ki(10, 4) = 0.01563; ki(10, 5) = -0.0046; ki(10, 6) =
0.0046; ki(10, 7) = 0.04865; ki(10, 8) = -0.01087; ki(10, 9) = -0.0079; ki(10, 10) = 0.13706; ki(10, 11) =
0.01829; ki(10, 12) = -0.01829;
ki(11, 1) = 0.0079; ki(11, 2) = 0.00159; ki(11, 3) = 0.00167; ki(11, 4) = 0.0046; ki(11, 5) = -0.00119; ki(11, 6) =
0.00111; ki(11, 7) = 0.01087; ki(11, 8) = -0.00238; ki(11, 9) = -0.00167; ki(11, 10) = 0.01829; ki(11, 11) =
0.00317; ki(11, 12) = -0.0025;
ki(12, 1) = -0.01087; ki(12, 2) = -0.00167; ki(12, 3) = -0.00238; ki(12, 4) = -0.0046; ki(12, 5) = 0.00111; ki(12,
6) = -0.00119; ki(12, 7) = -0.0079; ki(12, 8) = 0.00167; ki(12, 9) = 0.00159; ki(12, 10) = -0.01829; ki(12, 11) =
-0.0025; ki(12, 12) = 0.00317;

```

```

for x = 1 : 12
for y = 1 : 12
k(x, y) = k(x, y) + kx(x, y) + kxy(x, y) / af ^ 2 + ky(x, y) / af ^ 4;
end
end

```

```

% TYP = InputBox("WHAT TYPE OF PLATE? 1 for SSSS, 2 for CCCC, 3 for CSCS, 4 for CCSS, 5 for
CCCC, 6 for CSSS"); TYP = TYP * 1
g = input("WHAT IS THE SIZE OF THE GRID 3,5,7 etc?"); g = g * 1;
n = 3 * (g ^ 2); NN = (g + 2 + g) * 2; m = 3; mm = 3 * g; nm = n + g;
% ReDim kk(nm, nm), q(nm), kki(nm, nm), kii(nm, nm)
for x = 1 : nm
for y = 1 : nm
kk(x, y) = 0; kii(x, y) = 0; kki(x, y) = 0;
end
end

```

```

% STIFFNESS for PURE W one node
x = 1;
%for x = 1 : n - 2 Step 3
while x < n - 1
kk(x, x) = 4 * k(1, 1); kii(x, x) = 4 * ki(1, 1);
x = x + 3;
end
% Pure Tx one node

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

x = 2;
% for x = 2 : n - 1 Step 3
while x < n
    kk(x, x) = 4 * k(2, 2); kii(x, x) = 4 * ki(2, 2);
    x = x + 3;
end

% Pure Ty one node
x = 3;
% for x = 3 : n Step 3
while x < n + 1
    kk(x, x) = 4 * k(3, 3); kii(x, x) = 4 * ki(3, 3);
    x = x + 3;
end

% Pure W-Tx; W-Ty one node
x = 3;
% for x = 3 : n - 2 Step 3
while x < n - 1
    kk(x, x + 1) = 0; kii(x, x + 1) = 0;
    kk(x, x + 2) = 0; kii(x, x + 2) = 0;
    x = x + 3;
end

% Pure W two nodes
t = 1; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
    for y = 1 : g - 1
        kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
        kk(t, t2) = k(1, 10) + k(4, 7); kii(t, t2) = ki(1, 10) + ki(4, 7);
        kk(t, t3) = k(1, 7); kii(t, t3) = ki(1, 7);
        t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
    end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

t = 3 * g - 2; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(1, 10) + k(4, 7); kii(t, t1) = ki(1, 10) + ki(4, 7);
    t = t1; t1 = t1 + 3 * g;
end

t = n - 3 * g + 1; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(1, 4) + k(10, 7); kii(t, t1) = ki(1, 4) + ki(10, 7);
    t = t1; t1 = t1 + 3;
end

% Pure Tx two nodes
t = 2; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    kk(t, t2) = k(2, 11) + k(5, 8); kii(t, t2) = ki(2, 11) + ki(5, 8);
    kk(t, t3) = k(2, 8); kii(t, t3) = ki(2, 8);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g - 1; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(2, 11) + k(5, 8); kii(t, t1) = ki(2, 11) + ki(5, 8);

```

```

t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 2; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(2, 5) + k(11, 8); kii(t, t1) = ki(2, 5) + ki(11, 8);
    t = t1; t1 = t1 + 3;
end

```

% Pure Ty two nodes

```

t = 3; t1 = t + 3; t2 = t + 3 * g; t3 = t2 + 3;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    kk(t, t2) = k(3, 12) + k(6, 9); kii(t, t2) = ki(3, 12) + ki(6, 9);
    kk(t, t3) = k(3, 9); kii(t, t3) = ki(3, 9);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

```

```

t = 3 * g; t1 = t + 3 * g;
for x = 1 : g - 1
    kk(t, t1) = k(3, 12) + k(6, 9); kii(t, t1) = ki(3, 12) + ki(6, 9);
    t = t1; t1 = t1 + 3 * g;
end

```

```

t = n - 3 * g + 3; t1 = t + 3;
for x = 1 : g - 1
    kk(t, t1) = k(3, 6) + k(12, 9); kii(t, t1) = ki(3, 6) + ki(12, 9);
    t = t1; t1 = t1 + 3;
end

```

% Pure W two nodes back down

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

t = 4; t1 = t + 3 * (g - 1); t2 = t1 + 1; t3 = t1 + 2;
tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, t1) = k(4, 10); kii(t, t1) = ki(4, 10);
    kk(t + 1, t1) = k(5, 10); kii(t + 1, t1) = ki(5, 10);
    kk(t + 2, t1) = k(6, 10); kii(t + 2, t1) = ki(6, 10);

    kk(t + 1, t2) = k(5, 11); kii(t + 1, t2) = ki(5, 11);
    kk(t + 2, t3) = k(6, 12); kii(t + 2, t3) = ki(6, 12);

    kk(t, t2) = k(4, 11); kii(t, t2) = ki(4, 11);
    kk(t, t3) = k(4, 12); kii(t, t3) = ki(4, 12);
    t = t + 3; t1 = t1 + 3; t2 = t2 + 3; t3 = t3 + 3;
end
    t = tt + 3 * g; t1 = tt1 + 3 * g; t2 = tt2 + 3 * g; t3 = tt3 + 3 * g;
    tt = t; tt1 = t1; tt2 = t2; tt3 = t3;
end

% W-Tx W-Ty two nodes
t = 1; tx1 = t + 4; ty1 = t + 5; tx2 = t + 3 * g + 1; ty2 = t + 3 * g + 2;
tx3 = tx2 + 3; ty3 = ty2 + 3;
tt = t; ttx1 = tx1; tty1 = ty1; ttx2 = tx2; tty2 = ty2;
ttx3 = tx3; tty3 = ty3;

for x = 1 : g - 1
for y = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(2, 4) + k(11, 7); kii(t + 1, tx1 - 1) = ki(2, 4) + ki(11, 7);
    kk(t + 2, tx1 - 1) = k(3, 4) + k(12, 7); kii(t + 2, tx1 - 1) = ki(3, 4) + ki(12, 7);

    kk(t, tx2) = k(1, 11) + k(4, 8); kii(t, tx2) = ki(1, 11) + ki(4, 8);
    kk(t + 1, tx2 - 1) = k(2, 10) + k(5, 7); kii(t + 1, tx2 - 1) = ki(2, 10) + ki(5, 7);
    kk(t + 2, tx2 - 1) = k(3, 10) + k(6, 7); kii(t + 2, tx2 - 1) = ki(3, 10) + ki(6, 7);

    kk(t, tx3) = k(1, 8); kii(t, tx3) = ki(1, 8);
    kk(t + 1, tx3 - 1) = k(2, 7); kii(t + 1, tx3 - 1) = ki(2, 7);
    kk(t + 2, tx3 - 1) = k(3, 7); kii(t + 2, tx3 - 1) = ki(3, 7);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    kk(t, ty2) = k(1, 12) + k(4, 9); kii(t, ty2) = ki(1, 12) + ki(4, 9);
    kk(t, ty3) = k(1, 9); kii(t, ty3) = ki(1, 9);
    t = t + 3; tx1 = tx1 + 3; tx2 = tx2 + 3; tx3 = tx3 + 3;
    ty1 = ty1 + 3; ty2 = ty2 + 3; ty3 = ty3 + 3;
end
    t = tt + 3 * g; tx1 = ttx1 + 3 * g; tx2 = ttx2 + 3 * g; tx3 = ttx3 + 3 * g;
    ty1 = tty1 + 3 * g; ty2 = tty2 + 3 * g; ty3 = tty3 + 3 * g;
    tt = t; ttx1 = tx1; ttx2 = tx2; ttx3 = tx3;
    tty1 = ty1; tty2 = ty2; tty3 = ty3;
end

t = 3 * g - 2; tx1 = t + 3 * g + 1; ty1 = t + 3 * g + 2;

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

for x = 1 : g - 1
    kk(t, tx1) = k(1, 11) + k(4, 8); kii(t, tx1) = ki(1, 11) + ki(4, 8);
    kk(t + 1, tx1 - 1) = k(5, 7) + k(2, 10); kii(t + 1, tx1 - 1) = ki(5, 7) + ki(2, 10);
    kk(t + 2, tx1 - 1) = k(6, 7) + k(3, 10); kii(t + 2, tx1 - 1) = ki(6, 7) + ki(3, 10);

    kk(t, ty1) = k(1, 12) + k(4, 9); kii(t, ty1) = ki(1, 12) + ki(4, 9);
    t = tx1 - 1; tx1 = tx1 + 3 * g; ty1 = ty1 + 3 * g;
end

t = n - 3 * g + 1; tx1 = t + 4; ty1 = t + 5;
for x = 1 : g - 1
    kk(t, tx1) = k(1, 5) + k(10, 8); kii(t, tx1) = ki(1, 5) + ki(10, 8);
    kk(t + 1, tx1 - 1) = k(11, 7) + k(2, 4); kii(t + 1, tx1 - 1) = ki(11, 7) + ki(2, 4);
    kk(t + 2, tx1 - 1) = k(12, 7) + k(3, 4); kii(t + 2, tx1 - 1) = ki(12, 7) + ki(3, 4);

    kk(t, ty1) = k(1, 6) + k(10, 9); kii(t, ty1) = ki(1, 6) + ki(10, 9);
    t = tx1 - 1; tx1 = tx1 + 3; ty1 = ty1 + 3;
end

% Boundary conditions

t1 = n + 1; t2 = n + g; t3 = n + g + 1; t4 = n + 2 * g; t5 = n + 2 * g + 1;
t6 = n + 3 * g; t7 = n + 3 * g + 1; t8 = n + 4 * g;
f1 = 1; f2 = n - 3 * g + 1; f3 = 3 * g - 2; f4 = n - 2;

%%Right boundary
x = f3; tt1 = t1;
while x < f4 + 1
%For x = f3 To f4 Step 3 * g
    kk(x, t1) = k(1, 5) + k(10, 8); kii(x, t1) = ki(1, 5) + ki(10, 8);
    kk(x + 1, t1) = k(2, 5) + k(11, 8); kii(x + 1, t1) = ki(2, 5) + ki(11, 8);
    kk(x + 2, t1) = k(3, 5) + k(12, 8); kii(x + 2, t1) = ki(3, 5) + ki(12, 8);
    x = x + 3 * g;
t1 = t1 + 1;
%end
end
x = f3 + 3 * g; t1 = tt1;
while x < f4 + 1
%For x = f3 + 3 * g To f4 Step 3 * g
    kk(x, t1) = k(10, 5); kii(x, t1) = ki(10, 5);
    kk(x + 1, t1) = k(11, 5); kii(x + 1, t1) = ki(11, 5);
    kk(x + 2, t1) = k(12, 5); kii(x + 2, t1) = ki(12, 5);
    % kk(t3, t3 + 1) = k(5, 8);
    x = x + 3 * g;
t1 = t1 + 1;
%end
end

for x = n + 1 : n + g - 1
kk(x, x + 1) = k(5, 8); kii(x, x + 1) = ki(5, 8);
end

```

APENDIX E

MATLAB Program Formulated for CCCS Boundary Condition

```

x = f3; t1 = tt1 + 1;
while x < f4 - 3 * g + 1
%For x = f3 To f4 - 3 * g Step 3 * g
    kk(x, t1) = k(1, 8);kii(x, t1) = ki(1, 8);
    kk(x + 1, t1) = k(2, 8);kii(x + 1, t1) = ki(2, 8);
    kk(x + 2, t1) = k(3, 8);kii(x + 2, t1) = ki(3, 8);
    x = x + 3 * g;
t1 = t1 + 1;
%end
end

% while x < n + 2 * g + 1
for x = n + 1 : n + g
%for x = n + 1; n + g
    kk(x, x) = k(5, 5) + k(8, 8);kii(x, x) = ki(5, 5) + ki(8, 8);
    %x = x + 1;
%end
end

%Complete the symmetry
for x = 1 : nm

    for y = 1 : nm
        kk(y, x) = kk(x, y); kii(y, x) = kii(x, y);
    end
end

%Load Vector

x = 1;
while x < n + 1
%for x = 1 ; n Step 3
    %%for x = 1 ; n - 2 Step 3
        q(x) = 1;
        q(x + 1) = 0;
        q(x + 2) = 0;
        x = x + 3;
%end
end

%right load
for x = n + 1 : n + g
    q(x) = -0.0833333333333333;
end

ncd = (n - 1) / 2;
kgv = inv(kk);
dd = kgv * transpose(q);
dc = dd(ncd) * 1000 / (1 + g) ^ 4;
kki = inv(kii) * kk;
ld = eig(kki) * (1 + g) ^ 4;

```