# Sound Transmission Through Panels Using Element Free Galerkin Technique

Ying Yao

A Thesis

In

The Department

Of

Mechanical and Industrial Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science At

Concordia University

Montreal, Quebec, Canada

June, 2003

*Your file Votre référence*
*ISBN: 0-612-83889-7*
*Our file Notre référence*
*ISBN: 0-612-83889-7*

# Abstract

## Sound Transmission Through Panels Using Element Free Galerkin Technique

**Ying Yao**

Sound transmission through flexible panels is investigated using Element Free Galerkin (EFG) method. In contrast to Finite Element Method (FEM), EFG method uses a set of nodes scattered within the problem domain and its boundaries to model the structure. These sets of nodes do not form a mesh, and hence information on the relationship between the nodes is not required for field variable interpolation.

Moving Least Square (MLS) approach is employed to generate the displacement functions in EFG method for vibration analysis of elastic structures. Subdivisions similar to finite elements are used to provide a background mesh for numerical integration. The essential boundary conditions are enforced by Lagrange multipliers for static problems. For analysis of free vibration and forced vibrations, the essential boundary conditions are imposed using orthogonal transform techniques. To demonstrate the validity and versatility of the method, modal analysis of beams and thin plates with different boundaries have been carried out. In addition, the response of these structures under dynamic excitation has been analyzed. The results obtained are in good agreement with those obtained by other methods. Sound transmission loss through panels with all edges clamped is investigated and the results are presented and discussed. Results are compared with those obtained from simple application of mass law and the agreement is quite good.

## Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. R. B. Bhat, for his supervision, encouragement as well as financial support throughout the thesis work and looks forward to his guidance in the future in my PhD program.

I would like to express my appreciation for those who helped with my education at Concordia University. In particular, Dr. H. J. McQueen for the instructive courses he taught, the many directions he gave for my project and financial support in my first year of graduate study.

I would like to thank the great help from Concave.

Many thanks to my parents, who encouraged my mathematical and scientific interests at an early age.

Finally, I would also like to thank my husband and my lovely son for their support and understanding.

# Table of Contents

## List of Figures

## List of Tables

# List of Symbols

## Latin symbols

| | |
|---|---|
| $a(x)$ | Coefficients which are functions of the space coordinates $x$. The dimension of $a(x)$ is $m \times 1$ |
| $A(x)$ | Weighed moment matrix, the dimensions of $A(x)$ are $m \times m$. $m$ is the number of terms of polynomial basis |
| $B_i$ | Strain matrix. |
| $c_i$ | Distance between two neighboring nodes |
| $C_{,i}$ $C_{,j}$ $C_{,ij}$ | Partial derivatives of $C(x)$ with respect to spatial variables |
| $C(u)$ | Conditions that the approximated field function cannot satisfy |
| $C(x)$ | Matrix used for MLS approximation, the dimension is $m \times n$, $n$ is the number of nodes whose support domain includes point $x$ |
| $C_i$ | One Column of $C(x)$, the dimension is $m \times 1$ |
| $C$ | Damping matrix |
| $C_{diag}$ | Damping matrix |
| $d_{mi}$ | Radius of the domain of influence of the $I^{th}$ node |
| $d_{max}$ | Scaling factor |
| $D$ | Matrix of material |
| $H$ | Constraints matrix |
| $I_i$ | Incident intensity |

| | |
|---|---|
| $I_t$ | Transmitted intensity |
| $J(x)$ | Weighted, discrete error norm |
| $[J]$ | Jacobian matrix |
| $K$ | Stiffness matrix |
| $K_{ij}$ | Nodal stiffness matrix |
| $\tilde{K}$ | Condensed stiffness matrix |
| $K_{diag}$ | Stiffness matrix |
| $L$ | Lagrangian functional |
| $L$ | Differential operator matrix |
| $\tilde{L}$ | Modified Lagrangian |
| $m$ | Number of terms of polynomial basis |
| $M$ | Mass matrix |
| $M_{ij}$ | Nodal mass matrix |
| $M_{diag}$ | Diagonal matrices for mass |
| $\tilde{M}$ | Condensed stiffness matrix |
| $n$ | The number of nodes with domain of influence containing the point x |
| $n_t$ | The total number of nodes in the problem domain |
| $n_\lambda$ | The number of nodes used for interpolation Lagrange multiplier at node |
| $N_i(s)$ | Lagrange interpolant along the essential boundary |
| $N_i(\xi)$, $N_i(\xi,\eta)$ | Mapping function from physical coordinates to natural coordinates |
| $p(x)$ | Complete polynomial basis |
| $p^T(x)$ | Transpose of p(x), the dimension is $1\times m$ |

| | |
|---|---|
| $p_i$ | Incident pressure |
| $p_t$ | Transmitted pressure |
| q | Generalized displacement vector |
| s | Arc-length along the essential boundary |
| $\bar{t}$ | Traction forces |
| T | Kinetic energy |
| TL | Sound transmission loss |
| u | Displacement vector |
| $u^h(x)$ | Approximation of field variable u |
| $\bar{u}$ | Prescribed displacement vector on the essential boundaries (displacement) |
| U | Amplitude of vibration |
| w(r) | Weight function for support domain that can be written more compactly as a function of the normalized distance r |
| W | Modal matrix |
| $W_f$ | Work done by external forces |
| $\tilde{W}$ | Condensed modal matrix |
| $x_h$ | Homogeneous solution |
| $x_p$ | Particular solution |

**Greek symbols**

| | |
|---|---|
| $\omega$ | Natural frequencies |
| $\xi, \eta$ | Natural coordinates |

| | |
|---|---|
| $\varphi$ | MLS shape function |
| $\varphi_{i,x}$, $\varphi_{i,y}$ | Derivatives of the MLS shape function with respect to x and y |
| $\psi_i$ | Matrix of MLS shape function, which depends on the type of boundary and the type of problem |
| $\lambda_i$ | Lagrange multiplier at node i on the essential boundary |
| $\Phi_{,i}, \Phi_{,ij}$ | Partial derivatives of $\Phi(x)$ with respect to the spatial variables |
| $\Phi(x)$ | Matrix of MLS shape functions corresponding to nodes whose influence of domain cover point x |
| $\gamma(x)$ | Polynomial base matrix multiplied by the inverse matrix of A(x) |
| $\gamma_{,i}$, $\gamma_{,ij}$ | Partial derivatives of $\gamma(x)$ with respect to x,y spatial variables |
| $\Pi_s$ | Strain energy |
| $\Gamma_t$ | Boundary of the solids |
| $\lambda$ | Vector of the Lagrange multipliers |
| $\Omega$ | Volume of the solid |
| $\rho c$ | Characteristic acoustic impedance of the medium |

**Abbreviations**

| | |
|---|---|
| MLS | Moving Least Square method |
| EFG | Element Free Galerkin method |
| FEM | Finite Element Mathod |
| SDOF | Single Degree of Freedom System |

# Chapter 1

# Introduction

## 1.1 Sound Transmission Through the Panel

Sound of high intensity in workplace is harmful to human health. Continuous exposure to high levels of sound may lead to hearing loss, headache and loss of concentration. Noise also interferes with speech and may cause safety problems. Noise is more and more perceived as an environmental pollutant, and hence the reduction of sound is very important in industries.

Sound level may be reduced either (i) at the source, (ii) along the path, or (iii) at the receiver. In order to reduce sound along the path, barriers such as panels can be placed. The noise reduction in such panels can be found by studying their transmission loss. Upon determination of sound transmission loss characteristics of a structure, noise transmission loss can be improved through either passive or active means. Its evaluation is very important in many noise control problems.

There are two ways to predict sound transmission loss: Experimental measurements and analytical predictions. Traditionally the sound transmission loss of panels has been measured experimentally [1] [2] [3] [4]. The results of experiments depend on the laboratory conditions, not only room parameters, but also the measurement techniques. In addition, special attention will have to be paid to the sound transmission paths [5].

Transmitted sound is strongly related to the flexible panel response, which clearly identifies the need to accurately analyze the plate response. Analytical methods for determining modes with different plate boundary conditions including beam characteristic functions in Rayleigh-Ritz method were used by McDonald et al.[6]. Since the beam characteristic functions do not represent the plate modes exactly, a set of plate characteristic functions were developed by R. B. Bhat et al [7] [8] [9] to accurately determine the plate modes with different boundary conditions. The use of plate characteristic functions to express plate modes will result in more accurate estimation of the plate response. The plate characteristic functions were determined through the exact solution of the reduced equation, using an iterative method. The plate partial differential equation was reduced to an ordinary differential equation by substituting an assumed approximate solution that satisfies the boundary conditions along one direction of the panel. The method provides a more accurate estimation of natural frequencies and plate response than those obtained by the beam characteristic functions in the Rayleigh-Ritz method.

With the rapid development of computer technology, computational simulation techniques are used to model and investigate acoustics problems. Many studies have been carried out to solve sound transmission problems by using FEM [11]. Finite element method and boundary element method have been used for modeling silencers, enclosures, duct and piping networks as well as barrier walls.

The study in this thesis aims at developing a new method – using one kind of meshfree method - to solve for sound transmission loss. The method is called Element Free Galerkin (EFG) method. Meshfree methods were discovered 20 years ago by

Belytschko, T. [14], however, it is only recently that they have captured the interest of researchers. The reason for developing meshfree method is that certain classes of problems present difficulties when solved using conventional FEM, including applications with large changes in geometry such as fracture mechanics [15] [16]. The difficulties arise due to the inherent structure of the finite element: the rigid connectivity defined by elements. Although many meshfree methods have found applications in static analyses including fracture problems, contact problems and large deformation [17], little work has been done to solve sound transmission problem by using Element Free Galerkin method (EFG)..

## 1.2 EFG-Element Free Galerkin Method

The Element Free Galerkin (EFG) method is a relatively recent approach, which was proposed by T. Belytschko et al. [18]. It is in many respects strikingly similar to the finite element method. The finite element method for the modeling of complex problems in applied mechanics and related fields is well established. It is a robust and thoroughly developed technique, but it has shortcomings. The reliance of the method on a mesh leads to complications for certain classes of problems. In the modeling of large deformation processes, considerable loss in accuracy arises when the elements in the mesh become extremely skewed or compressed. The traditional technique for handling these complications is to remesh the domain of the problem at every step during the evolution of the simulation. This prevents the severe distortion of elements. To ameliorate these difficulties, a new class of methods have recently been developed which do not require a mesh to discretize the problem. One of this kind of methods is Element Free Galerkin (EFG) method.

The principal attraction of meshfree methods is the possibility of simplifying adaptively. In sound transmission problems, nodes can be added around area that has high sound intensity with the desired accuracy. Adaptive meshing for a large variety of problems, including linear and nonlinear stress analysis, can be effectively treated by these methods in a simple manner.

The EFG method is a really new modeling and simulation technique that can solve many complex problems. However, in classical approaches, there are already some meshfree ideas existing there. These methods include Rayleigh-Ritz [19] [20] method, Galerkin method [21] and collocation method [22] [23]. Rayleigh-Ritz method is an energy method; it requires the choice of suitable independent trial functions that satisfy the geometrical boundary conditions. The solution is approximated by a sequence of trial functions each attached with arbitrary coefficients. The unknown constants are determined using a variational approach. Galerkin method and collocation method can be classified as the weighted residuals method. In contrast with the Rayleigh-Ritz method, the weighted residuals method work directly with the differential equation. The difference between Galerkin method and collocation method is that different weighting functions are chosen. In Galerkin method, the weighting functions coincide with the trial functions while Dirac delta functions are weighting functions in collocation method. To some extent, collocation method has some similarities with meshfree methods. It uses nodes but no mesh to model structures. The detailed information related to Rayleigh-Ritz method, Galerkin method and collocation method will be explained in detail in chapter 2.

The EFG method, first developed by T. Belytschko as an improved version of the diffuse element method introduced by Nayroles et al. [24], only requires a set of nodes scattered within the problem domain as well as sets of nodes scattered on the boundaries of the domain. The connectivity is defined according to nodal weight function that possess compact support, so that each node represents only the local region surrounding it, called its domain of influence. The shape functions are constructed using moving least square (MLS) approximation which have been developed for curve and surface fitting of random data [25]. These MLS approximants replace the usual FE interpolants as the test and trial functions in the Galerkin formulation.

From finite element background, one may understand easily the fact that the only fundamental difference between the finite element method and the EFG method is computation of the shape functions. The construction of the shape functions and their spatial derivatives involves relatively much effort when compared to the finite element method. Furthermore, the shape functions and their spatial derivatives have complicated forms[18] (compared to the polynomials that appear in finite element procedures). Therefore, it is necessary to use a relatively large number of integration points (T. Belytschko et al [26]) in order to obtain accurate results. On the other hand, the shape functions and their derivatives are continuous and shape function derivatives are computed exactly. From the flowchart shown in Figure 1.1, one can have a better understanding of EFG procedure.

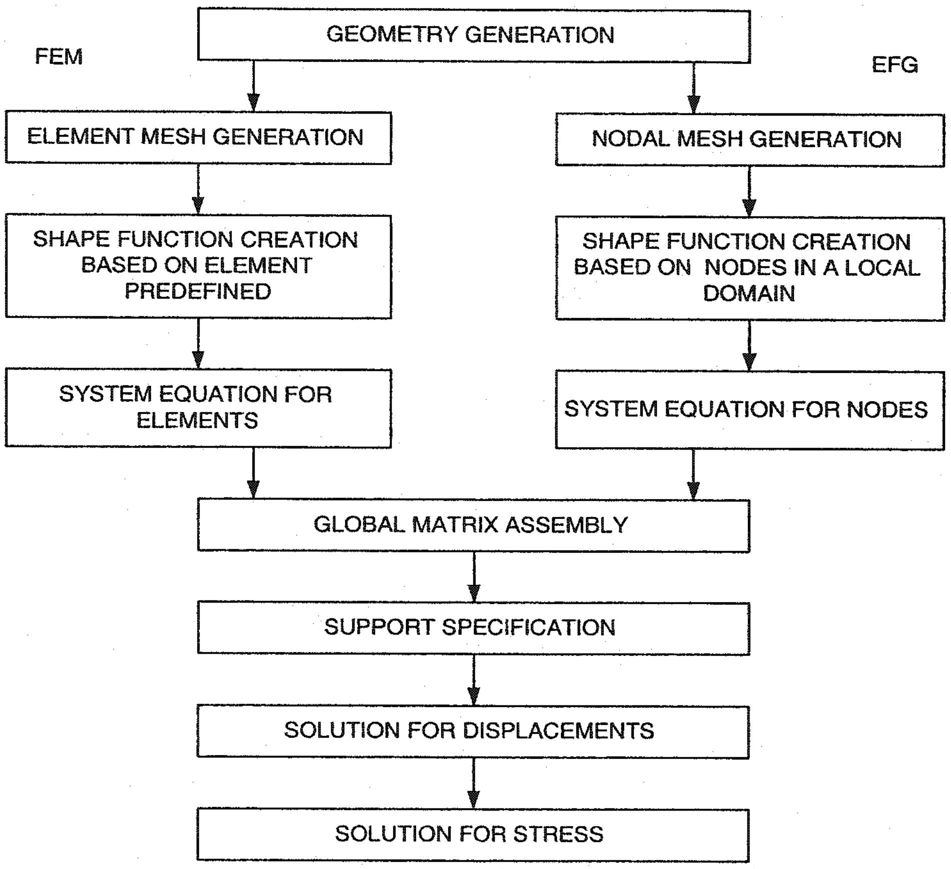
FEM
EFG
GEOMETRY GENERATION
ELEMENT MESH GENERATION
NODAL MESH GENERATION
SHAPE FUNCTION CREATION BASED ON ELEMENT PREDEFINED
SHAPE FUNCTION CREATION BASED ON NODES IN A LOCAL DOMAIN
SYSTEM EQUATION FOR ELEMENTS
SYSTEM EQUATION FOR NODES
GLOBAL MATRIX ASSEMBLY
SUPPORT SPECIFICATION
SOLUTION FOR DISPLACEMENTS
SOLUTION FOR STRESS

Figure 1.1 Flowchart for FEM and EFG method procedures

## 1.3 Thesis Organization

The description of this investigation starts with a literature survey on sound transmission and Element Free Galerkin Method (chapter 1). Chapter 2 covers different techniques to solve vibration problems. Chapter 3 deals with principles for Element Free Galerkin method and shape function construction. Chapter 4 deduces discretized equations of truss, beam, in-plane plate, and thin plate by Element Free Galerkin method for natural frequencies and mode shapes. Numerical results are presented in chapter 5. In chapter 6, Element Free Galerkin method and modal superposition method are employed to calculate the sound transmission loss through the panel. Conclusions are made in chapter 7 together with suggestions for future work.

# Chapter 2

# Different Numerical and Approximate Methods to Obtain Vibration Response of Panels

## 2.1 Introduction

This chapter covers different type of methods that can solve for the vibration response of elastic panel structures. The discussion starts with the finite element method, followed by the Rayleigh-Ritz and Galerkin method. Collocation method will be discussed in detail in the end since it is one kind of simple meshfree methods. The natural frequencies and system matrices by different techniques will be presented in chapter 5.

## 2.2 Finite Element Method

In general, modeling and simulation of engineering systems require solving the complex differential or partial differential equations that govern the problem. The finite element method is particularly useful for solving differential equations, together with their boundary conditions, over a domain of complex shape. The process, therefore, represents the problem domain by a large number of finite elements of simpler shapes connected together at a set of points called the nodes. In each of these elements, the structural behavior is considered individually, and then the over-all structural equilibrium equations are assembled from the individual components. The analysis is based on the principle of virtual displacements which is similar to the application of the principle of minimum potential energy. The general finite element analysis process, however, should be understood as a numerical procedure to obtain approximate solution to problems in continuum mechanics. The approximation achieved depends on the characteristics and the number of elements that are used to idealize the structure. The elements have to satisfy convergence requirements, but to obtain accurate results a large number of finite elements are also needed.

By using a properly predefined mesh and by applying a proper principle, complex differential or partial differential governing equations can be approximated by a set of algebraic equations for the mesh. The system of algebraic equations of the whole problem domain can be formed by assembling sets of algebraic equations for all the mesh.

In FEM, the shape functions are constructed using elements and the shape functions will be the same for all the elements of the same type. In fact, if the natural coordinate systems are used, the form of shape functions in the natural coordinates are identical for elements of the same type. These shape functions are usually predetermined for different types of elements before the finite element analysis starts.

The finite element method defines elements between the nodes. It must be known that each node is connected in order to compute the shape functions. As long as elements are used, the problem mentioned in chapter 1, such as, large deformation, crack growth and so on, will not be easy to solve. The concept of element free or mesh free methods has been proposed, in which the domain of the problem is represented by a set of arbitrarily distributed nodes.

There are a number of methods that can be regarded as simple meshfree methods, such as the Rayleigh-Ritz method, Galerkin method, and collocation method. By reviewing these methods, it is possible to comprehend easily the complex meshfree method – Element Free Galerkin (EFG) Method that will be discussed in chapter 3.

## 2.3 The Rayleigh-Ritz Method

The Rayleigh-Ritz method involves determination of the maximum kinetic and potential energies of the structural system and approximates the solution with a finite expansion of the form

$$W(\mathbf{x}) = a_1\phi_1(\mathbf{x}) + a_2\phi_2(\mathbf{x}) + \cdots + a_n\phi_n(\mathbf{x}) \tag{2.1}$$

where $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_n(\mathbf{x})$ are trial functions of $\mathbf{x}$, which individually must satisfy the geometrical boundary conditions. Satisfaction of the differential equation of motion is not required. The unknown coefficients $(a_1, a_2, \cdots a_n)$ in Equation (2.1) are obtained from the condition

$$\frac{\partial \Pi}{\partial a_i} = 0 \qquad (i = 1,2,3,\cdots,n) \tag{2.2}$$

where

$$\Pi = U - T \tag{2.3}$$

In the above, U is maximum strain energy, and T is maximum kinetic energy.

This procedure yields a set of homogeneous linear equations in $a_i$. In this way, the problem is reduced to an eigenvalue problem.

## 2.4 Weighted Residual Method

In contrast with the Rayleigh-Ritz method, the weighted residuals method works directly with the differential equation [23]. The differential equation of elastic body free vibration has the following form.

$$Lw(x) = \lambda m(x)w(x) \tag{2.4}$$

where L is differential operator of order 2p and m is the mass density. The solution w(x) is subjected to given boundary conditions. The following assumption is not a closed-form solution. $w^{(n)}(x)$ is an approximation of deflection w(x), and $w^{(n)}(x)$ can be written in the form

$$w(x) \cong w^{(n)}(x) = \sum_{i=1}^{n} a_i\phi_i(x) \tag{2.5}$$

in which $\phi_1, \phi_2, \ldots, \phi_n$ are n independent trial functions or shape functions.

The error is referred to as a residual and denoted by

$$R(w^{(n)}, x) = Lw^{(n)}(x) - \lambda^{(n)}mw^{(n)}(x) \tag{2.6}$$

At the same time, the n independent functions $\psi_1(x), \psi_2(x), \ldots \psi_n(x)$ are chosen as weight functions and the weighted residual is defined as

$$\psi_i R = \psi_i(Lw^{(n)} - \lambda^{(n)}mw^{(n)}) \qquad i = 1,2,\ldots,n \tag{2.7}$$

The objective is to obtain the unknown parameters $a_i$ by reducing the error to the largest extent possible. The coefficients $a_i$ in Equation (2.5) (i=1,2,...,n) be such that the integral of the weight residual be zero for every i.

$$\int_0^1 \psi_i R dx = \int_0^1 \psi_i(Lw^{(n)} - \lambda^{(n)}mw^{(n)})dx = 0 \qquad i = 1,2,\ldots,n \tag{2.8}$$

Inserting Equation (2.5) into Equation (2.8), we obtain the algebraic eigenvalue problem

$$(\psi_i, R) = \int_0^L \psi_i \left( \sum_{j=1}^{n} a_j L\phi_j - \lambda^{(n)} \sum_{j=1}^{n} a_j m\phi_j \right) dx$$

$$= \sum_{j=1}^{n} \left( k_{ij} - \lambda^{(n)} m_{ij} \right) a_j = 0 \qquad i = 1,2,\cdots,n \tag{2.9}$$

where

$$k_{ij} = \int_0^L \psi_i L\phi_j dx \qquad i,j = 1,2,\cdots,n \tag{2.10}$$

$$m_{ij} = \int_0^L \psi_i m\phi_j dx \qquad i,j = 1,2,\cdots,n \tag{2.11}$$

Rewriting Equation (2.10) and (2.11) in matrix form

$$K = \int_0^L \psi L \varphi^T dx \tag{2.12}$$

$$M = \int_0^L m \psi \varphi^T dx \tag{2.13}$$

Equation (2.8) states that the residual is orthogonal to every one of the weighting functions. The trial functions or shape functions (usually polynomials) are chosen to satisfy both the geometry boundary conditions and the natural boundary conditions.

### 2.4.1 Galerkin method

Galerkin method is the most widely used of the weighted residual methods. In Galerkin method, the weighted functions are same as the trial functions. In vector form, we have

$$\psi = \varphi$$

Therefore, Equation (2.12) and (2.13) become

$$K = \int_0^L \varphi L \varphi^T dx \tag{2.14}$$

$$M = \int_0^L m\varphi\varphi^T dx \tag{2.15}$$

### 2.4.2 Collocation Method

The collocation method requires that the boundary-value problem be satisfied exactly at n nodes in the problem domain. The parameters of polynomials are then found by forcing the approximation to satisfy the differential equation at a given set of n points, i.e. at these collocation points where the residual vanishes.

The collocation method requires that the strong form (the partial differential system equation) be satisfied exactly at n nodes in the problem domain; this is accomplished by choosing Dirac delta functions located at various preselected nodes of the system as weight functions.

$$w_i(x) = \delta(x - x_i), \qquad i = 1,2,\ldots,n \tag{2.16}$$

$$\int_0^l w_i R dx = \int_0^l \delta(x - x_i)(Lw^{(n)}(x) - \lambda^{(n)} m w^{(n)}(x))dx = 0$$

$$= Lw^{(n)}(x_i) - \lambda^{(n)} m(x_i) w^{(n)}(x_i) = 0 \qquad i = 1,2,\ldots,n \tag{2.17}$$

Introducing Equation (2.5) into Equation (2.17), yields the algebraic eigenvalue problem is given by

$$K^{(n)} a = \lambda^{(n)} M^{(n)} a \tag{2.18}$$

in which the matrices $K^{(n)}$ and $M^{(n)}$ have the elements, respectively,

$$k_{ij} = \int_0^l \delta(x - x_i) L\phi_j dx = L\phi_j(x_i) \qquad i,j = 1,2,\ldots,n \tag{2.19}$$

$$m_{ij} = \int_0^l \delta(x - x_i) m\phi_j dx = m(x_i)\phi_j(x_i) \qquad i, j = 1, 2, \ldots, n \qquad (2.20)$$

As an example, take five nodes in a cantilever beam. The deflection w(x) can be approximated in terms of the following polynomial

$$w(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_j x^j \qquad (2.21)$$

where w(x) must satisfy both the geometrical boundary and natural boundary conditions. The geometrical boundary conditions are:

$$w(0) = 0\,,\ w'(0) = 0\,. \qquad (2.22)$$

and the natural boundary conditions are:

$$w''(L) = 0, w'''(L) = 0\,. \qquad (2.23)$$

Substitution of Equation (2.21) into (2.22) and (2.23), and choosing three collocation points from the five nodes (leaving out the two boundary nodes), results in three trial functions, for j=4, 5 and 6, which satisfy both the geometrical and the natural boundary conditions. The three trial functions are:

$$\phi_1 = 6x^2 - 4x^3 + x^4$$

$$\phi_2 = 20x^2 - 10x^3 + x^5$$

$$\phi_3 = 45x^2 - 20x^3 + x^6$$

For cantilever beam, the differential operator is $L = \frac{d^4}{dx^4}$, and the fourth derivative of each trial function is given by:

$$\frac{d^4\phi_1}{dx^4} = 24$$

$$\frac{d^4\phi_2}{dx^4} = 120x$$

$$\frac{d^4\phi_3}{dx^4} = 360x^2$$

Using Equation (2.19), matrix K is obtained as

$$K = \begin{bmatrix} 24 & 30 & 22.5 \\ 24 & 60 & 90 \\ 24 & 90 & 202.5 \end{bmatrix}$$

Similarly, using Equation (2.20), matrix M is obtained as

$$M = \begin{bmatrix} 0.3164 & 1.0947 & 2.5002 \\ 1.0625 & 3.7813 & 8.7656 \\ 2.0039 & 7.2686 & 17.053 \end{bmatrix}$$

The resulting eigenvalue problem in Equation (2.18) can be solved to give natural frequencies and mode shapes. The disadvantage of collocation method is that it cannot be applied to complex, practical problems since the strong form (differential equation) of the problem must be known.

## 2.5 Element Free Galerkin (EFG) Method

From above discussions, each numerical approximate method has its advantages and disadvantages. Finite element method is based on energy principle and strong form equations are not needed in solving for engineering problems. The disadvantage of FEM is the rigid connection of nodes between elements that results in some evident limitations of FEM which are already mentioned in chapter 1. Classical methods such as the Rayleigh-Ritz, Galerkin, and collocation methods have been used with success in many areas of engineering. There are certain difficulties which prevent it from being more widely used for the solution of practical problems. One obvious problem involves the choice of trial functions. It is clear that for an irregular-shaped boundary, it would in general be impossible to find one function, let alone a sequence of functions, which satisfies any essential boundary conditions.

A new method, which is called meshfree method, has been proposed. It overcomes disadvantages of both FEM and the classical methods (Rayleigh-Ritz, Galerkin, and collocation methods). The problem domain (can be any shape) is represented by a set of arbitrarily distributed nodes. Also, since EFG uses Galerkin weak form, one need not know the strong form of the system equation (differential equations).

In the next chapter, one kind of mesh free method - EFG, is introduced in detail and the basic equations of solid mechanics along with their discrete forms for numerical implementation are discussed.

# Chapter 3

# Principles for Element Free Galerkin Method and Shape Function Construction

## 3.1 Introduction

In this chapter, the principles of Element Free Galerkin method are described. The static and dynamic equations in general form are obtained by using Element Free Galerkin techniques. Finally, the most important and difficult part of EFG method-shape function construction - is presented in detail. One can use the detailed information and formula to build shape functions and complete the very crucial step in EFG.

## 3.2 Strong Form and Weak Form

The partial differential equations developed from the equilibrium of forces on the block of elastic body are strong form system equations for the solids [27]. Obtaining the exact solution for strong form system equations is ideal for simple problems, but for complex problems, it is difficult and impractical to get answers. On the other hand, weak form requires weaker consistency and formulation based on weak form can produce a set of algebraic system equations and gives discretized system equations which can produce much more accurate results. Therefore weak form is preferred to obtain an approximate solution. This thesis uses weak form to get discretized system equations for Element Free method.

## 3.3 Hamilton's Principle

Hamilton's principle is one of the variational principles based on the energy principle. It states," Of all possible time histories of consistent displacement states which satisfy

(1) The compatibility conditions

(2) The essential displacement boundary conditions

(3) The conditions at initial time $t_1$ and final time $t_2$

the history corresponding to the actual solution makes the Lagrangian functional a minimum." Mathematically, Hamilton's principle states

$$\delta\int_{t_1}^{t_2} L dt = 0 \tag{3.1}$$

where L is the Lagrangian functional. For a system of solids and structures it can be defined as

$$L = T - \Pi_s + W_f \tag{3.2}$$

where T is the kinetic energy, $\Pi_s$ is the strain energy, and $W_f$ is the work done by external forces.

The kinetic energy is defined by

$$T = \frac{1}{2}\int_\Omega \rho \dot{\mathbf{u}}^T \dot{\mathbf{u}} d\Omega \tag{3.3}$$

where **u** is the displacement vector.

$$\mathbf{u} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{3.4}$$

u,v,w are the displacement components in x, y, and z direction respectively. $\Omega$ stands for the whole volume of the solid. For solids and structures of elastic materials, the strain energy of the system can be expressed as

$$\Pi_s = \frac{1}{2}\int_\Omega \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} d\Omega \tag{3.5}$$

where $\boldsymbol{\varepsilon}, \boldsymbol{\sigma}$ are the strain vector and stress vector, respectively.

$$\boldsymbol{\varepsilon}^T = \{\varepsilon_{xx} \quad \varepsilon_{yy} \quad \varepsilon_{zz} \quad \varepsilon_{yz} \quad \varepsilon_{xz} \quad \varepsilon_{xy}\} \tag{3.6}$$

$$\boldsymbol{\sigma}^T = \{\sigma_{xx} \quad \sigma_{yy} \quad \sigma_{zz} \quad \sigma_{yz} \quad \sigma_{xz} \quad \sigma_{xy}\} \tag{3.7}$$

The work done by the external forces can be obtained by

$$W_f = \int_\Omega \mathbf{u}^T \mathbf{b} d\Omega + \int_{\Gamma_t} \mathbf{u}^T \bar{\mathbf{t}} d\Gamma \tag{3.8}$$

where **b** is the vector of external body forces in x, y, and z directions

$$\mathbf{b} = \begin{Bmatrix} b_x \\ b_y \\ b_z \end{Bmatrix} \tag{3.9}$$

and $\Gamma_t$ stands for the boundary of the solids on which the traction forces $\bar{t}$ are prescribed. The advantage of using Hamilton's principle is that it is not necessary to know the strong form of the system equation.

## 3.4 Constrained Hamilton's Principle

There are cases when the approximate field function does not satisfy the conditions (1), (2), (3) in page 18 on parts of the problem domain including the boundaries. Hamilton's principle has to be modified or constrained for such situations. There are basically two methods often used to modify the functional: Lagrange multipliers and the penalty method. In this study, Lagrange multipliers method is employed to modify Equation (3.1). Consider the following k conditions that the approximated field function cannot satisfy:

$$\mathbf{C}(\mathbf{u}) = \begin{Bmatrix} C_1(\mathbf{u}) \\ C_2(\mathbf{u}) \\ \cdot \\ \cdot \\ \cdot \\ C_k(\mathbf{u}) \end{Bmatrix} = 0 \tag{3.10}$$

Our goal is to find the stationary point of the Lagrangian functional subjected to the constraint of Equation (3.10). The modified Lagrangian is written as follows:

$$\tilde{L} = L + \int_{\Omega} \boldsymbol{\lambda}^T \mathbf{C}(\mathbf{u}) d\Omega \tag{3.11}$$

where $\boldsymbol{\lambda}$ is a vector of the Lagrange multipliers

$$\boldsymbol{\lambda}^T = \{\lambda_1 \ \ \lambda_2 \ \ \lambda_3 \cdots \ \lambda_n\} \tag{3.12}$$

These Lagrange multipliers are unknown functions of independent coordinates in the domain $\Omega$. The modified Hamilton's principle is given by

$$\delta \int_{t_1}^{t_2} \tilde{L} dt = 0 \tag{3.13}$$

## 3.5 Galerkin Weak Form

The Galerkin weak form can be directly derived using Hamilton's principle for problems of solid mechanics. By using Equations (3.1) to (3.8), the Lagrangian L becomes

$$L = -\frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma} d\Omega + \int_{\Omega} \mathbf{u}^{T}\mathbf{b} d\Omega + \int_{\Gamma_t} \mathbf{u}^{T}\bar{\mathbf{t}} d\Gamma + \frac{1}{2}\int_{\Omega} \rho\dot{\mathbf{u}}^{T}\dot{\mathbf{u}} d\Omega \tag{3.14}$$

Substituting the above into Equation (3.1), we have

$$\delta\int_{t_1}^{t_2}\left[-\frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma} d\Omega + \int_{\Omega} \mathbf{u}^{T}\mathbf{b} d\Omega + \int_{\Gamma_t} \mathbf{u}^{T}\bar{\mathbf{t}} d\Gamma + \frac{1}{2}\int_{\Omega} \rho\dot{\mathbf{u}}^{T}\dot{\mathbf{u}} d\Omega\right]dt = 0 \tag{3.15}$$

Moving the variation operation into the integral operations, we obtain

$$\int_{t_1}^{t2}\left[-\frac{1}{2}\int_{\Omega} \delta(\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}) d\Omega + \int_{\Omega} \delta\mathbf{u}^{T}\mathbf{b} d\Omega + \int_{\Gamma_t} \delta\mathbf{u}^{T}\bar{\mathbf{t}} d\Gamma + \frac{1}{2}\int_{\Omega} \delta(\rho\dot{\mathbf{u}}^{T}\dot{\mathbf{u}}) d\Omega\right]dt = 0 \tag{3.16}$$

In Equation (3.16), the first term can be rewritten as

$$\delta\left(\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}\right) = \delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma} + \boldsymbol{\varepsilon}^{T}\delta\boldsymbol{\sigma} \tag{3.17}$$

The last term in Equation (3.17) can be changed as follows:

$$\boldsymbol{\varepsilon}^{T}\delta\boldsymbol{\sigma} = \left(\boldsymbol{\varepsilon}^{T}\delta\boldsymbol{\sigma}\right)^{T} = \delta\boldsymbol{\sigma}^{T}\boldsymbol{\varepsilon} \tag{3.18}$$

Using constitutive Equation and the symmetric property of the matrix of material constant, we arrive

$$\delta\boldsymbol{\sigma}^{T}\boldsymbol{\varepsilon} = \delta(D\varepsilon)^{T}\boldsymbol{\varepsilon} = \delta\boldsymbol{\varepsilon}^{T}D^{T}\boldsymbol{\varepsilon} = \delta\boldsymbol{\varepsilon}^{T}D\varepsilon = \delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma} \tag{3.19}$$

Equations (3.17) becomes:

$$\delta\left(\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}\right) = 2\delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma} \tag{3.20}$$

In order to investigate the last term in Equation (3.16), we move the time integration into spatial integration:

$$\int_{t1}^{t2}\left[\frac{1}{2}\int_{\Omega}\delta(\rho\dot{\mathbf{u}}^T\dot{\mathbf{u}})d\Omega\right]dt=\frac{1}{2}\int_{\Omega}\left[\int_{t1}^{t2}\delta(\rho\dot{\mathbf{u}}^T\dot{\mathbf{u}})dt\right]d\Omega \tag{3.21}$$

$$\int_{t1}^{t2}\delta(\rho\dot{\mathbf{u}}^T\dot{\mathbf{u}})dt=\rho\int_{t1}^{t2}[\delta\dot{\mathbf{u}}^T\dot{\mathbf{u}}+\dot{\mathbf{u}}^T\delta\dot{\mathbf{u}}]dt=2\rho\int_{t1}^{t2}[\delta\dot{\mathbf{u}}^T\dot{\mathbf{u}}]dt \tag{3.22}$$

$$\int_{t1}^{t2}[\delta\dot{\mathbf{u}}^T\dot{\mathbf{u}}]dt=\int_{t1}^{t2}\left[\frac{d\delta\mathbf{u}^T}{dt}\frac{d\mathbf{u}}{dt}\right]dt \tag{3.23}$$

$$\int_{t1}^{t2}\left[\frac{d\delta\mathbf{u}^T}{dt}\frac{d\mathbf{u}}{dt}\right]dt=\int_{t1}^{t2}\left[-\delta\mathbf{u}^T\frac{d^2\mathbf{u}}{dt^2}\right]dt+\delta\mathbf{u}^T\frac{d\mathbf{u}}{dt}\Big|_{t1}^{t2} \tag{3.24}$$

$\delta\mathbf{u}^T=0$ , Equation (3.24) becomes

$$\int_{t1}^{t2}\left[\frac{d\delta\mathbf{u}^T}{dt}\frac{d\mathbf{u}}{dt}\right]dt=\int_{t1}^{t2}\left[-\delta\mathbf{u}^T\frac{d^2\mathbf{u}}{dt^2}\right]dt \tag{3.25}$$

Therefore, Equation (3.21) can be rewritten

$$\int_{t1}^{t2}\left[\frac{1}{2}\int_{\Omega}\delta(\rho\dot{\mathbf{u}}^T\dot{\mathbf{u}})d\Omega\right]dt=\int_{\Omega}\left[\rho\int_{t1}^{t2}\left[-\delta\mathbf{u}^T\frac{d^2\mathbf{u}}{dt^2}\right]dt\right]d\Omega \tag{3.26}$$

Now change the order of the integration

$$\int_{t1}^{t2}\left[\frac{1}{2}\int_{\Omega}\delta(\rho\dot{\mathbf{u}}^T\dot{\mathbf{u}})d\Omega\right]dt=-\int_{t1}^{t2}\left[\rho\int_{\Omega}[\delta\mathbf{u}^T\ddot{\mathbf{u}}]d\Omega\right]dt \tag{3.27}$$

Substituting Equation (3.27) into Equation (3.16), we have

$$\int_{t1}^{t2}\left[-\int_{\Omega}\delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}d\Omega+\int_{\Omega}\delta\mathbf{u}^T\mathbf{b}d\Omega+\int_{\Gamma_t}\delta\mathbf{u}^T\bar{\mathbf{t}}d\Gamma-\int_{\Omega}\rho\delta\mathbf{u}^T\ddot{\mathbf{u}}d\Omega\right]dt=0 \tag{3.28}$$

In order to satisfy the above equation for all possible choices of u, Equation (3.28) now becomes

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}d\Omega-\int_{\Omega}\delta\mathbf{u}^T\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^T\bar{\mathbf{t}}d\Gamma+\int_{\Omega}\rho\delta\mathbf{u}^T\ddot{\mathbf{u}}d\Omega=0 \tag{3.29}$$

This is Galerkin weak form for dynamic problems. For static problems, the equation reduces to

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^{T}\bar{\mathbf{t}}d\Gamma=0 \tag{3.30}$$

By using the strain-displacement relationship Equation [28]

$$\boldsymbol{\varepsilon}=\mathbf{L}\mathbf{u} \tag{3.31}$$

and the strain-stress relationship

$$\boldsymbol{\sigma}=\mathrm{D}\boldsymbol{\varepsilon} \tag{3.32}$$

Equations (3.29) and (3.30) can be expressed as follows in terms of displacement vector **u**. For dynamic problem, the weak form equation is

$$\int_{\Omega}\delta(\mathbf{L}\mathbf{u})^{T}\mathrm{D}(\mathbf{L}\mathbf{u})d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^{T}\bar{\mathbf{t}}d\Gamma+\int_{\Omega}\rho\delta\mathbf{u}^{T}\ddot{\mathbf{u}}d\Omega=0 \tag{3.33}$$

For Static problem

$$\int_{\Omega}\delta(\mathbf{L}\mathbf{u})^{T}\mathrm{D}(\mathbf{L}\mathbf{u})d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^{T}\bar{\mathbf{t}}d\Gamma=0 \tag{3.34}$$

Equation (3.33) and (3.34) are the Galerkin weak form written in terms of displacement. The discretized system equation can be obtained using approximated displacement in FEM and meshfree method.

## 3.6 Constrained Galerkin Weak Form

In cases when the approximated field function does not satisfy the condition (1) or (2) on parts of the problem domain including the boundaries, we should use the constrained Hamilton's principle to derive the constrained Galerkin weak form. The procedure is the same as in section 3.4, except that the modified Lagrangian $\tilde{\mathrm{L}}$ is used for formulation. The following are equations for dynamic and static problems using Galerkin weak form with Lagrange multipliers.

For dynamic problems,

$$\int_{\Omega}\delta\varepsilon^{T}\sigma d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^{T}\bar{\mathbf{t}}d\Gamma-\int_{\Omega}\delta\lambda^{T}C(\mathbf{u})d\Omega-\int_{\Omega}\lambda^{T}\delta C(\mathbf{u})d\Omega+\int_{\Omega}\rho\delta\mathbf{u}^{T}\ddot{\mathbf{u}}d\Omega=0 \tag{3.35}$$

For Static problems,

$$\int_{\Omega}\delta\varepsilon^{T}\sigma d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma_t}\delta\mathbf{u}^{T}\bar{\mathbf{t}}d\Gamma-\int_{\Omega}\delta\lambda^{T}C(\mathbf{u})d\Omega-\int_{\Omega}\lambda^{T}\delta C(\mathbf{u})d\Omega=0 \tag{3.36}$$

Since Galerkin weak form has been presented in this chapter and it can be used to derive the discretized system equations in terms of displacement, the following sections will focus on the shape function construction-Moving Least Square method.

## 3.7 Introduction to EFG shape function construction

The Element Free Galerkin (EFG) method is considered a meshfree method because the method only requires a set of nodes and a description of the boundaries to construct an approximate solution. The connectivity between the data points and the shape functions are constructed by the method without recourse to elements. The EFG method employs moving least-square (MLS) approximations which are composed of three components: a weight function of compact support associated with each node, a polynomial basis, and a set of coefficients that depend on position. The support of the weight function defines a node's domain of influence, which is the subdomain over which a particular node contributes to the approximation. The overlap of the nodal domains of influence defines the nodal connectivity. One useful property of MLS approximations is that their continuity is equal to the continuity of the weight function; highly continuous approximations can be generated by an appropriate choice of the weight function.

In the following sections, we will describe the construction of MLS approximation and the resulting EFG shape functions. In the course of this description, the effect of different weight functions is illustrated.

## 3.8 Least Square method

Least square method shares the same character with moving least square (MLS) method. It can be regarded as a kind of moving least square method if weight function related to each node is constant. Therefore, before discussing the moving least square method, a review of the least square method will be done, in order to get a better understanding of the moving least square method.

The method of least squares assumes that the best-fit curve of a given type is the curve that has the minimal sum of the squared deviations (least square error) from a given set of data (Figure 3.1).

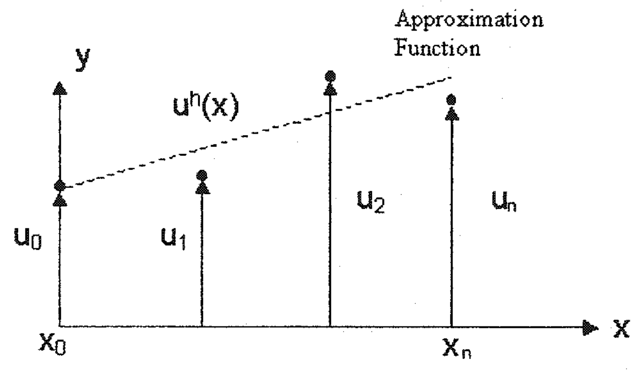

Figure 3.1 Least square method

Suppose that the data points are $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where x is the independent variable and y is the dependent variable. The fitting curve $u^h(x)$ has the deviation (error)

$d_i$ from each data point, i.e., $d_1 = y_1 - f(x_1)$, $d_2 = y_2 - f(x_2)$,..., $d_n = y_n - f(x_n)$. According to the method of least squares, the best fitting curve has the property

$$\Pi = d_1^2 + d_2^2 + \ldots + d_n^2 = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \text{min}. \tag{3.37}$$

## 3.9 Least Square Polynomials Fit

Polynomials are one of the most commonly used types of curves for least square curve fitting. The application of the method of least squares curve fitting using polynomials is briefly discussed in the following.

### 3.9.1 The Least Square line

The least-squares line uses a straight line

$$y = a_0 + a_1 x \tag{3.38}$$

to approximate the given set of data, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $n \geq 2$. The best fitting curve $u^h(x)$ has the least square error, i.e.,

$$\Pi = \sum_{i=1}^{n} [y_i - u^h(x_i)]^2 = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x)]^2 = \min \tag{3.39}$$

Here, $a_0$ and $a_1$ are unknown coefficients while all $x_i$ and $y_i$ are given. The least square error, will be minimum if its variation with respect to the unknown coefficients $a_0$ and $a_1$ is zero. Accordingly,

$$\begin{cases} \dfrac{\partial \Pi}{\partial a_0} = 2\sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_i)\right] = 0 \\ \dfrac{\partial \Pi}{\partial a_1} = 2\sum_{i=1}^{n} x_i \left[y_i - (a_0 + a_1 x_i)\right] = 0 \end{cases} \tag{3.40}$$

By solving these two equations, the unknown coefficients $a_0$ and $a_1$ can therefore be obtained.

### 3.9.2 The Least Square Parabola

The least square parabola uses a second degree curve

$$y = a_0 + a_1 x + a_2 x^2 \tag{3.41}$$

to approximate the given set of data, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $n \geq 3$. The best fitting curve $u^h(x)$ has the least square error, i.e.,

$$\Pi = \sum_{i=1}^{n} \left[y_i - u^h(x_i)\right]^2 = \sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_i + a_2 x_i^2)\right]^2 = \min. \tag{3.42}$$

Here, $a_0$, $a_1$, and $a_2$ are unknown coefficients while all $x_i$ and $y_i$ are given. Obtaining the variation of the least square error with respect to the unknown coefficients $a_0$, $a_1$, and $a_2$ and equating them to zero yields

$$\begin{cases} \dfrac{\partial \Pi}{\partial a_0} = 2\sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_i + a_2 x_i^2)\right] = 0 \\ \dfrac{\partial \Pi}{\partial a_1} = 2\sum_{i=1}^{n} x_i \left[y_i - (a_0 + a_1 x_i + a_2 x_i^2)\right] = 0 \\ \dfrac{\partial \Pi}{\partial a_2} = 2\sum_{i=1}^{n} x_i^2 \left[y_i - (a_0 + a_1 x_i + a_2 x_i^2)\right] = 0 \end{cases} \tag{3.43}$$

The unknown coefficients $a_0$, $a_1$, and $a_2$ can therefore be obtained by solving the above linear equations.

## 3.10 Moving Least Squares Approximations

Moving least square method (MLS), was first used by mathematicians for data fitting and surface construction. An excellent description of MLS method can be found in a paper by Lancaster and Salkamkas (1982) [25]. The MLS method is now a widely used alternative for construction of meshfree method shape functions. The MLS has two major features that make it popular: (1) the approximated field function is continuous and smooth in the entire problem domain; and (2) it is capable of producing an approximation with the desired order of consistency. Let u(x) be the function of the field variable defined in the domain $\Omega$. The approximation of u(x) at point **x** is denoted by $u^h(x)$ (Figure 3.2).

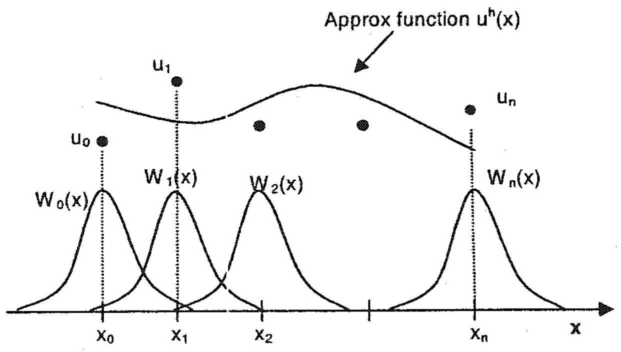

Figure 3.2 Moving Least square method

The MLS approximation $u^h(x)$ is given by

$$u^h(\mathbf{x}) = \sum_{j=1}^{m} p_j(\mathbf{x}) a_j(\mathbf{x}) \equiv p^T(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{3.44}$$

where p(**x**) is a complete polynomial basis of arbitrary order and a(**x**) are coefficients which are functions of the space coordinates x. Further, m is the number of terms of polynomial basis.

Examples of bases in one dimension are:

$$p^T(x) = p^T(x) = \{1, x, x^2, ..., x^n\} \tag{3.45}$$

$$p^T(x) = \{1, x\} \text{ (m=2, linear)} \tag{3.46}$$

$$p^T(x) = \{1, x, x^2\} \text{ (m=3, quadratic)} \tag{3.47}$$

Examples of bases in two dimensions are

$$p^T(x) = p^T(x, y) = \{1, x, y, xy, x^2, y^2, ..., x^n, y^n\} \tag{3.48}$$

$$p^T(x) = \{1, x, y\} \quad (m = 3, \ \text{linear}) \tag{3.49}$$

$$p^T(x) = \{1, x, y, xy, x^2, y^2\} \ (m = 6, \text{quadratic}) \tag{3.50}$$

The coefficients a(x) in Equation (3.44) are obtained at any point x by minimizing the following weighted, discrete error norm:

$$\begin{aligned} J &= \sum_{i}^{n} w(x - x_i)[u^h(x, x_i) - u(x_i)]^2 \\ &= \sum_{i}^{n} w(x - x_i)[p^T(x_i)a(x) - u(x_i)]^2 \end{aligned} \tag{3.51}$$

where $u(x_i)$ is the coefficient associated with node i at $x = x_i$, $w(x - x_i)$ is a weight function of compact support associated with node i, and n is the number of nodes with domains of influence containing the point x, i.e., $w(x - x_i) \neq 0$; see Figure (3.3, 3.4).

The weight function $w(x-x_i)$ provides the coefficients a(x) with their spatial dependence, and since the support of w is compact, the MLS approximation has local character.

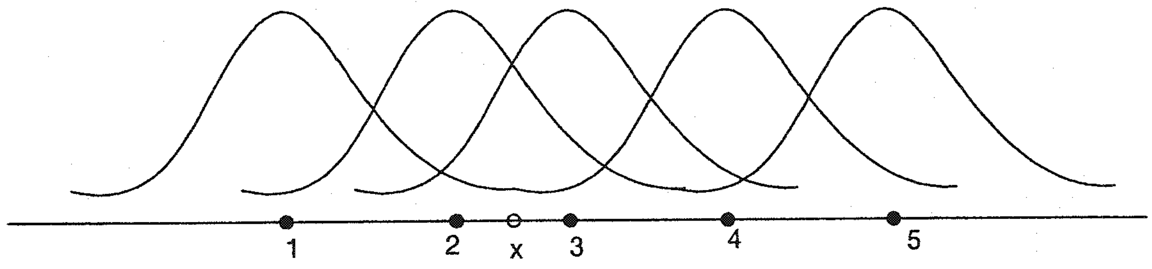

Figure 3.3 Overlapping domains of influence and local node numbering at point x.

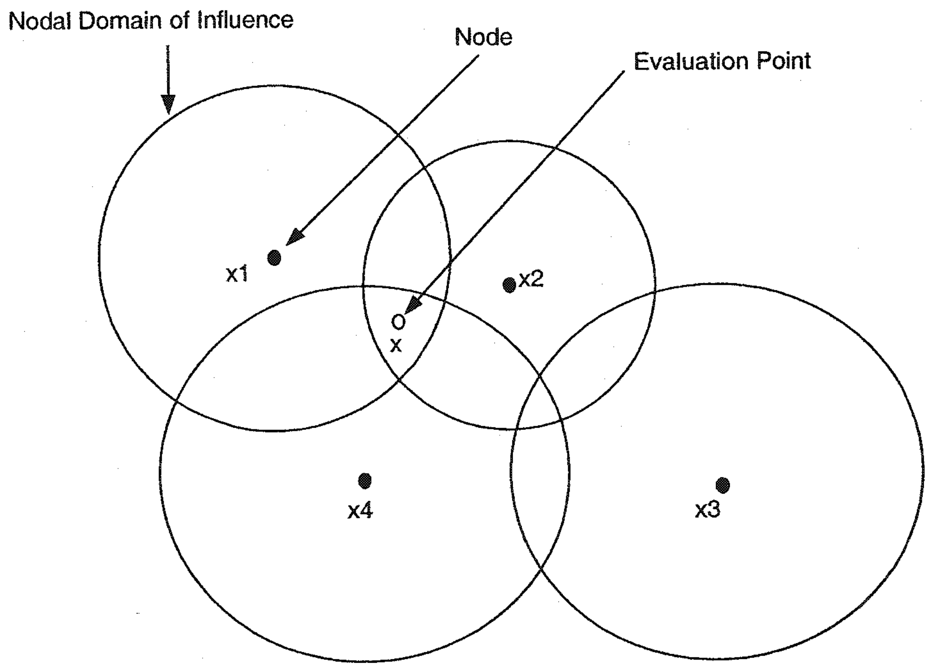

Figure 3.4 Illustration of nodal domains of influence in two dimensions. Nodes 1, 2, and 4 since their domains of influence contain point x, and they will be used in the approximation at x. Node 3 is excluded from the neighbor list for x.

## 3.11 Weight Functions

Weight functions play an important role in Element Free Galerkin (EFG) method. The weight function $w(x-x_i) \neq 0$ affects the approximation $u^h(\mathbf{x})$. As an illustration, consider the two cases shown in Figure 3.5 where a function u(**x**) in one dimension is approximated by $u^h(\mathbf{x})$ using five data points at x = 0, 1, 2, 3, 4. The MLS approximation $u^h(\mathbf{x})$ is constructed using a linear polynomial basis, $p^T = [1, x]$.

In the first example in Figure 3.5, the weight function associated with each node is constant over the entire domain. The minimization process at each point x involves all points in the domain (5 points) and J(x) in (3.51) reverts to the standard least squares norm, resulting in a linear fit through the data. In this case, the coefficients a(x) are no longer functions of the space variable x, but are constant everywhere in the domain; thus, the coefficients a(x) have spatial dependence only through the choice of the weight function. In fact, in this case, moving least square method is changed to least square method.

In the second example, Figure 3.6, the weight functions possess compact support, but now they are smooth functions that cover larger subdomains so that $n > m$, where $n$ is the number of points whose support domain includes x. The minimization of J(x) in (3.51) results in the type of MLS approximations that will be used in the EFG method; the approximation is smooth since the approximation inherits the continuity of the weight function. The approximation exhibits local character due to the limited support of the weight function.

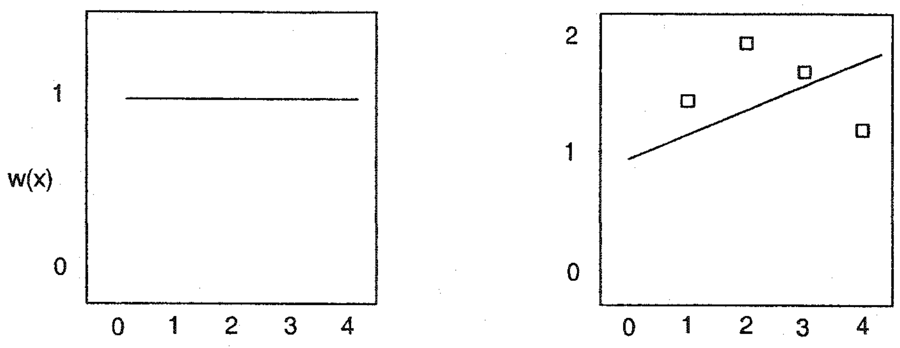

Figure 3.5 Constant weight function over entire domain

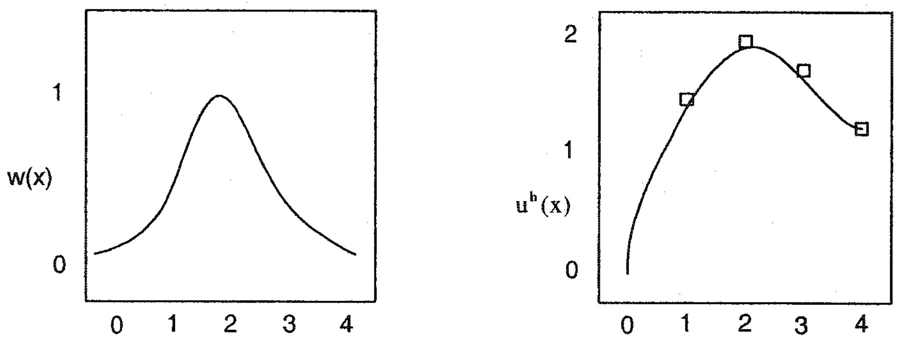

Figure 3.6 Smooth weight function with compact support.

For EFG, the weight function $w(x - x_i)$ is generally chosen as a monotonically decreasing function as $\|x - x_i\|$ increases. Defining $d_i = \|x - x_i\|$, and $r = d_i / d_{mi}$, where $d_{mi}$ is the radius of the domain of influence of the i$^{th}$ node, the weight can be written more compactly as a function of the normalized distance r. Some examples of weight functions are shown by using circle support domain.

$$\text{Gaussian: } w(r) = \begin{cases} e^{-(r/0.4)^2} & \text{for } r \le 1 \\ 0 & \text{for } r > 1 \end{cases} \tag{3.52}$$

$$\text{Cubic spline : } w(r) = \begin{cases} \frac{2}{3} - 4r^2 + 4r^3 & \text{for } r \le \frac{1}{2} \\ \frac{4}{3} - 4r + 4r^2 - \frac{4}{3}r^3 & \text{for } \frac{1}{2} < r \le 1 \\ 0 & \text{for } r > 1 \end{cases} \tag{3.53}$$

$$\text{Cubic polynomial: } w(r) = \begin{cases} 1 - 3r^2 + 2r^3 & \text{for } r \le 1 \\ 0 & \text{for } r > 1 \end{cases} \tag{3.54}$$

$$\text{Quartic polynomial: } w(r) = \begin{cases} 1 - 6r^2 + 8r^3 - 3r^4 & \text{for } r \le 1 \\ 0 & \text{for } r > 1 \end{cases} \tag{3.55}$$

The support radius or domain of influence at a node, $d_{mi}$ is computed by

$$d_{mi} = d_{max} c_i \tag{3.56}$$

where $d_{max}$ is a scaling factor, and $c_i$ is the maximum distance between node i and other nodes in the cell. If the nodes are uniformly distributed, $c_i$ is simply the distance between two neighboring nodes. The domain of influence multiplier $d_{max}$ is typically 2.0-- 4.0 for dynamic problems. Typical one dimensional examples of the weight functions in Equations (3.52-3.55) and their derivatives are shown in Figure 3.7.

The support domain above is a circle, however, square domains have also been used most frequently. In the present study, both kinds of support domains are investigated by solving cantilever thin plate free vibration problem. Both results are correct by comparing with exact solution. The results are given in chapter 5. Square support domain can be achieved by tensor product weights.

In this research, tensor product weights will be used with quartic weight function at any given point and is given by

$$w(x - x_i) = w(r_x) \cdot w(r_y) = w_x \cdot w_y \tag{3.57}$$

where $w(r_x)$ and $w(r_y)$ are given by Equation (3.52) to (3.55) with r replaced by $r_x$ and $r_y$ respectively; $r_x$ and $r_y$ are given by

$$r_x = \frac{\|x - x_i\|}{d_{mx}} \tag{3.58}$$

$$r_y = \frac{\|y - y_i\|}{d_{my}} \tag{3.59}$$

where

$$d_{mx} = d_{max} \cdot c_{xi} \tag{3.60}$$

$$d_{my} = d_{max} \cdot c_{yi} \tag{3.61}$$

and $c_{xi}$ and $c_{yi}$ are determined at a particular node by searching for enough neighbor nodes to satisfy the basis in both directions. The requirement is that the **A** matrix (weighted moment matrix will be discussed in section 3.12) be non-singular everywhere in the domain, and thus invertible, which is necessary to compute the shape function. If the nodes are uniformly spaced, the values $c_{xi}$ and $c_{yi}$ correspond to the distance between the nodes in the x and y directions, respectively.

The derivative of the weight function in (3.57) is calculated by

$$w_{,x} = \frac{dw_x}{dx} w_y \tag{3.62}$$

$$w_{,y} = \frac{dw_y}{dx} w_x \tag{3.63}$$

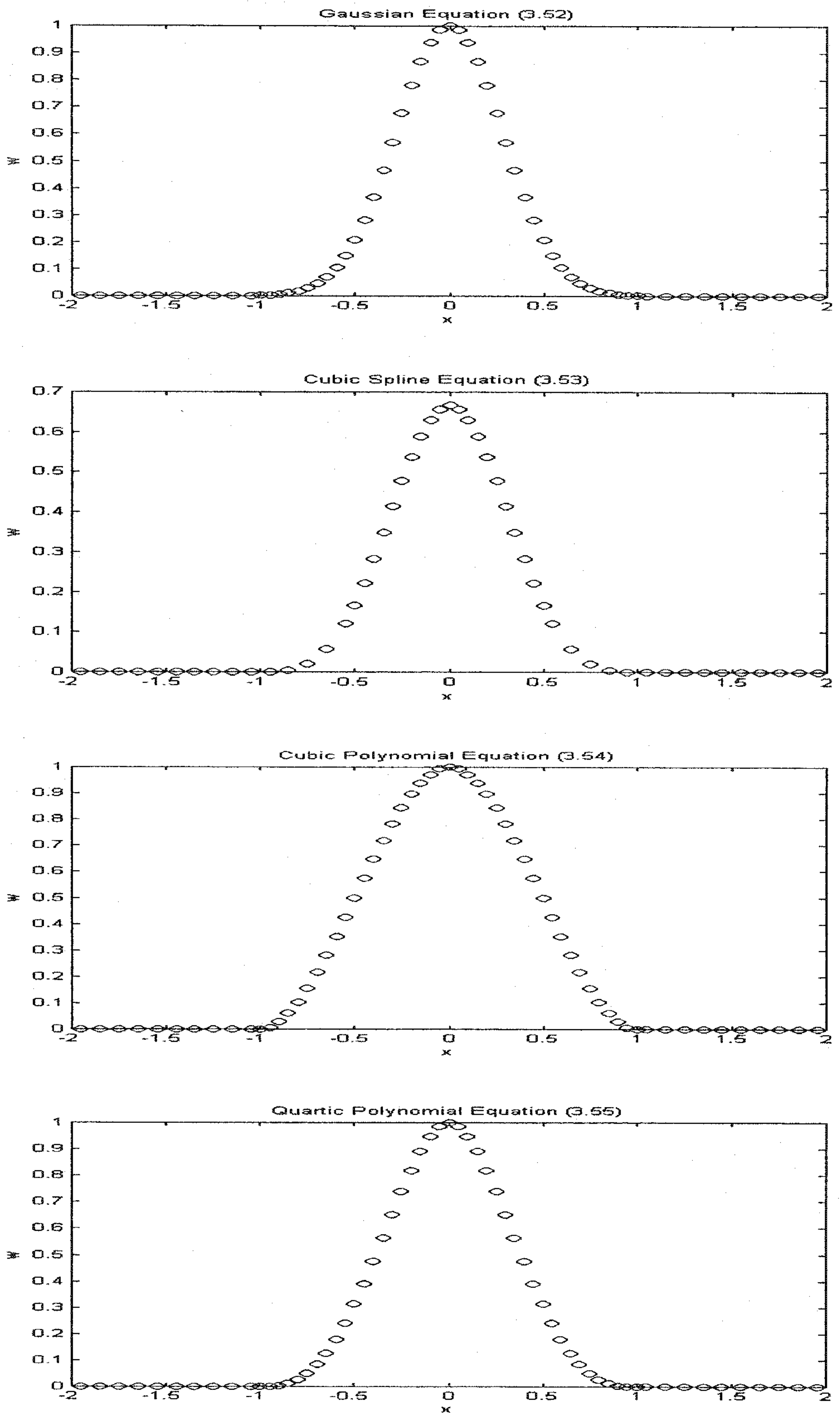

Figure 3.7: Typical weight functions

## 3.12 EFG Shape Functions

The EFG shape functions are constructed by minimizing J(**x**). Returning to the weighted discrete error norm in Equation (3.51), the stationary value of J(**x**) with respect to **a**(**x**) leads to

$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})} = 0 \tag{3.64}$$

which results in the following linear equation system:

$$A(\mathbf{x})\mathbf{a}(\mathbf{x}) = C(\mathbf{x})\mathbf{U}_s \tag{3.65}$$

where A is called the weighted moment matrix given by

$$A(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x}-\mathbf{x}_i)p(\mathbf{x}_i)p^T(\mathbf{x}_i) \tag{3.66}$$

In the Equation (3.65), matrix C has the form of

$$C(\mathbf{x}) = [w(\mathbf{x}-\mathbf{x}_1)p(\mathbf{x}_1), w(\mathbf{x}-\mathbf{x}_2)p(\mathbf{x}_2),..., w(\mathbf{x}-\mathbf{x}_n)p(\mathbf{x}_n)] \tag{3.67}$$

$$C_i = w(\mathbf{x}-\mathbf{x}_i)p(\mathbf{x}_i) \tag{3.68}$$

Noting that $\mathbf{U}_s$ is the vector that collects the nodal parameters of the field variable for all nodes whose support domain includes **x**

$$\mathbf{U}_s = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n]^T \tag{3.69}$$

and solving the Equation (3.65) for **a**(**x**), results in

$$\mathbf{a}(\mathbf{x}) = A^{-1}(\mathbf{x})C(\mathbf{x})\mathbf{U}_s \tag{3.70}$$

By substituting (3.70) into (3.44), the MLS approximants are

$$u^h(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_j(\mathbf{x})(A^{-1}(\mathbf{x})C(\mathbf{x}))_{ji}u_i \tag{3.71}$$

Equation (3.71) can be rewritten:

$$u^h(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) u_i \tag{3.72}$$

where the EFG shape functions $\varphi$ are defined as

$$\varphi_i(\mathbf{x}) = \sum_{j=1}^{m} p_j(x)(A^{-1}(\mathbf{x})C(\mathbf{x}))_{ji} = p^T(\mathbf{x})A^{-1}(\mathbf{x})C_i \tag{3.73}$$

Note that m is the number of terms of polynomials basis p(x), which is usually much smaller than n, which is the number of nodes in the neighborhood of point x used for constructing the shape function. The requirement of n>>m prevents the singularity of the matrix A, so that $A^{-1}$ exists.

Equation (3.72) can be rewritten as:

$$u^h(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x})\mathbf{U}_s \tag{3.74}$$

where $\mathbf{\Phi}(\mathbf{x})$ is the matrix of MLS shape functions corresponding to nodes whose influence of domain covers point x. It should be noted here that x is usually Gauss points in spatial coordinates, which can be obtained by mapping Gauss points in natural coordinates to Gauss points in physical coordinates. Further discussion can be found in chapter 4.

$$\Phi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \cdots, \varphi_n(\mathbf{x})] \tag{3.75}$$

In order to determine the spatial derivatives of the function of the field variable, which are required for deriving the discretized system equations, it is necessary to obtain the derivatives of the MLS shape functions. To obtain the partial derivatives of shape functions, Equation (3.75) is written as follows using Equations (3.73):

$$\mathbf{\Phi}(\mathbf{x}) = \gamma^T(\mathbf{x})C(\mathbf{x}) \tag{3.76}$$

where $\gamma(\mathbf{x})$ is determined by

$$A(\mathbf{x}) \cdot \gamma(\mathbf{x}) = p(\mathbf{x}) \tag{3.77}$$

The partial derivatives of $\gamma(\mathbf{x})$ can be obtained as follows:

$$A\gamma_{,i} = p_{,i} - A_{,i}\gamma \tag{3.78}$$

$$A\gamma_{,ij} = p_{,ij} - (A_{,i}\gamma_{,j} + A_{,j}\gamma_{,i} + A_{,ij}\gamma) \tag{3.79}$$

where i, j denote coordinates x and y. A comma designates a partial derivative with respect to the indicated spatial variable. The partial derivatives of shape function can be given as

$$\mathbf{\Phi}_{,i} = \gamma_{,i}C + \gamma C_{,i} \tag{3.80}$$

$$\mathbf{\Phi}_{,ij} = \gamma_{,ij}C + \gamma_{,i}C_{,j} + \gamma_{,j}C_{,i} + \gamma C_{,ij} \tag{3.81}$$

From Figure 3.7, one can observe that EFG approximants do not satisfy the Kronecker delta criterion, i.e., $\varphi_i(x_j) \neq \delta_{ij}$. The feature of moving least square (MLS) shape function results in

$$u^h(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) u_i \neq u_i \tag{3.82}$$

which means that the displacement obtained by solving the EFG system of equations is not the actual displacement at the nodes. Consequently, the imposition of essential boundary conditions is more complicated than for the standard FEM. Several methods have been developed, including Lagrange multipliers (Belytschko, Lu, and Gu, 1994)[18], penalty method [29], and coupling finite element [30]. In the following chapters of the thesis, Lagrange multipliers method and orthogonal transform techniques [31] will be introduced. For Static problems, Lagrange multipliers method is used to impose essential boundary conditions. For analysis of free vibration, the essential boundary conditions are enforced by orthogonal transform techniques.

This chapter presented the theory of Element Free Galerkin method. Constrained Galerkin weak form is used to form system equations. Very detailed information related to uniqueness shape functions construction by using moving least square method is given. Once the shape function is achieved, that will be more than half way toward Element Free Galerkin (EFG) method. In the next chapter, the method is applied to different kinds of weak forms to produce discretized system equations for the solution of free vibration problems.

# Chapter 4

# Discretized Equations by Element Free Galerkin Method for Natural Frequency Analysis

## 4.1 Element Free Galerkin Method (EFG)

Based on the moving least squares (MLS) approach, Element Free Galerkin Method (EFG) is employed to generate the displacement functions for vibration analysis of elastic bodies. The major characteristics of the EFG are the following:

1. Galerkin weak form is employed to develop the discretized system equation, which is described in chapter 3.
2. Moving least square (MLS) approximation is used for the construction of the shape function, which is discussed in chapter 3.
3. Cells of the background mesh for integration are required to carry out the integration to calculate system matrices, which will be explained in this chapter.

Although EFG can be considered meshfree with respect to shape function construction or function approximation, a mesh will be required for system equations calculation by the Galerkin approximation procedure. This is because evaluation of the integrals in the weak form requires a subdivision of the domain.

This chapter provides a very detailed procedure that leads to the EFG method. The equation of motion is established for both one-dimensional problems and two-dimensional problems by following the standard procedure and boundary conditions are imposed by orthogonal transform techniques.

## 4.2 Flow Chart of Element Free Galerkin Method (EFG)

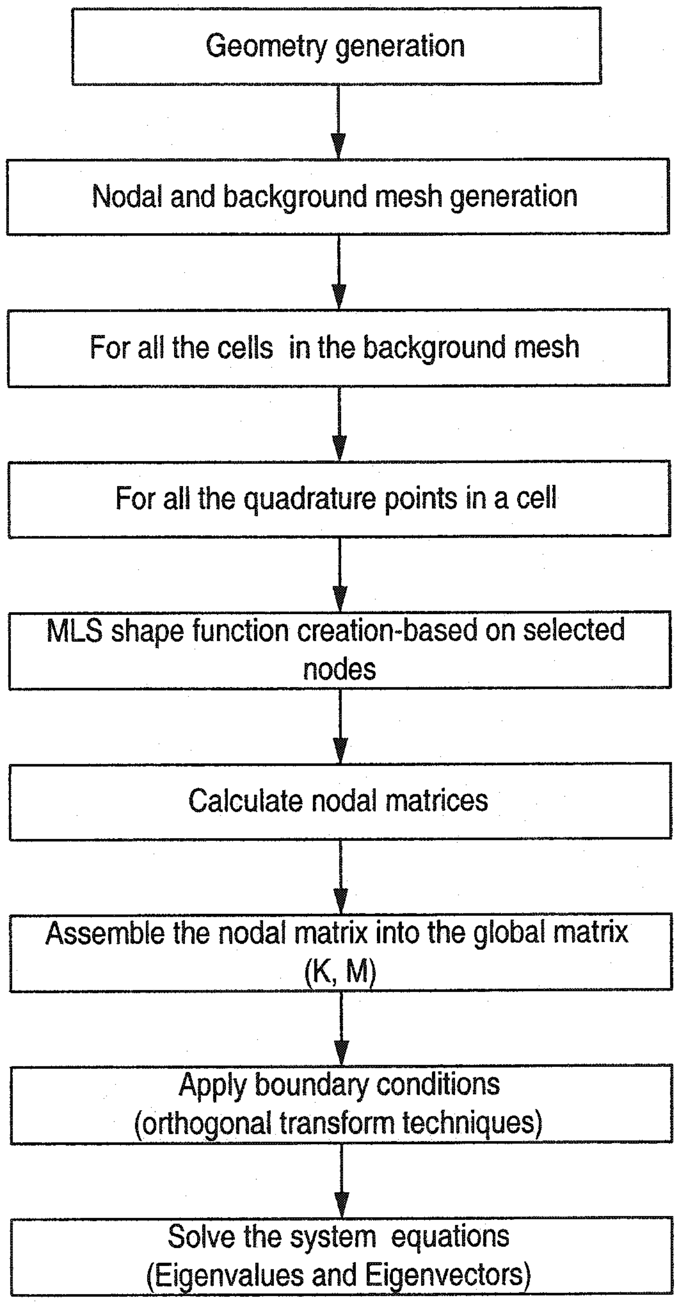
Geometry generation
Nodal and background mesh generation
For all the cells in the background mesh
For all the quadrature points in a cell
MLS shape function creation-based on selected nodes
Calculate nodal matrices
Assemble the nodal matrix into the global matrix (K, M)
Apply boundary conditions (orthogonal transform techniques)
Solve the system equations (Eigenvalues and Eigenvectors)

Figure 4.1 Flowchart of EFG

## 4.3 Constrained Galerkin Weak Form for Free Vibration Analysis

Since there are no forces applied on the elastic body for free vibration analysis, the second and third terms can be removed from Equation (3.33) in chapter 3. The next task is to impose essential boundary conditions. G. R. Liu [32] gives very detailed information on the imposition of boundary conditions. The enforcement of essential boundary conditions by Lagrange multipliers increases the number of unknowns and leads to awkward system equations. The use of Lagrange multipliers is extremely unwieldy for dynamic problems. In order to obtain a set of well-behaved system equations, it is more practical to formulate separately the system equations with the original Lagrangian and the constraint equations using their own weak form to obtain two separate discretized sets of equations. The discretized constrained equations are decomposed to obtain a set of orthogonal vectors which are used to produce a condensed set of system equations using orthogonal transformation techniques, which at the same time ensures the satisfaction of the constraints. Therefore, for free vibration analysis, the Galerkin weak form of the elastodynamic undamped equilibrium equations (3.33) can be rewritten by using two separate equations. One is for free vibration equilibrium equation, and the other is for essential boundary conditions.

## 4.4 Free Vibration Equilibrium Equation

$$\int_{\Omega} \delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}\,d\Omega + \int_{\Omega} \rho\,\delta\mathbf{u}^{T}\ddot{\mathbf{u}}\,d\Omega = 0 \tag{4.1}$$

The strain-displacement relationship is given by

$$\boldsymbol{\varepsilon} = \mathbf{Lu} \tag{4.2}$$

Stress-strain relationship is given by

$$\boldsymbol{\sigma} = \mathrm{D}\boldsymbol{\varepsilon} \tag{4.3}$$

Substituting Equation (4.2), (4.3) into Equation (4.1), yields

$$\int_{\Omega} \delta(\mathbf{Lu})^{T}(\mathrm{D}\mathbf{Lu})d\Omega + \int_{\Omega} \delta\mathbf{u}^{T}\rho\ddot{\mathbf{u}}d\Omega = 0 \tag{4.4}$$

The final dynamic equation for free vibration is obtained as follows:

$$\mathrm{K}\mathbf{u} + \mathrm{M}\ddot{\mathbf{u}} = 0 \tag{4.5}$$

where K is the global stiffness matrix, M is the global mass matrix, and **u** is deflection vector.

Consider that the elastic body is undergoing a harmonic vibration. The deflection **u** can be expressed in the form

$$\mathbf{u} = \mathbf{U}\exp(i\omega t) \tag{4.6}$$

where i is the imaginary unit, $\omega$ is the angular frequency, and U is the amplitude of the vibration. Substitution of Equation (4.6) into (4.5) leads to the following

$$(\mathrm{K} - \omega^{2}\mathrm{M})\mathbf{U} = 0 \tag{4.7}$$

where **U** is an eigenvector.

## 4.5 Equations for Essential Boundary Conditions

There are two kinds of boundary conditions: displacement boundary conditions (essential) and force boundary conditions (natural). The essential boundary conditions are the conditions that have to be satisfied by the trial functions before they are substituted into the weak form. In the stage of constructing the trial function, one does not have to consider any of the natural boundary conditions, because they will come out later naturally.

Imposing essential boundary conditions is very important in Element Free Galerkin (EFG) Method. The essential boundary conditions cannot be enforced as easily as in finite element method because moving least square interpolants do not pass through the data. This is a disadvantage in Element Free Galerkin (EFG) method for it complicates the imposition of essential boundary conditions.

In this study, Lagrange multipliers will be used to enforce the essential boundary conditions because it is the simplest technique which gives acceptably accurate results.

The weak form of the essential boundary conditions with Lagrange multipliers is employed to produce the discretized essential boundary conditions as given below:

$$\int_{\Gamma_u} \delta\lambda^T (\mathbf{u} - \bar{\mathbf{u}}) d\Gamma = 0 \tag{4.8}$$

where $\mathbf{u}$ is the displacement vector, $\bar{\mathbf{u}}$ is the prescribed displacement vector on the essential boundaries (displacement), and $\lambda$ is a vector of Lagrange multipliers, which can be interpolated as follows:

$$\lambda(\mathbf{x}) = \sum_{i=1}^{n_\lambda} N_i(s)\lambda_i \qquad x \in \Gamma_u \tag{4.9}$$

where $n_\lambda$ is the number of nodes used for this interpolation, s and $N_i(s)$ are arc-length and Lagrange interpolant along the essential boundary, $\lambda_i$ is the Lagrange multiplier at node i on the essential boundary. The Lagrange interpolant of the order n can be given in a general form

$$N_k^n(s) = \frac{(s - s_0)(s - s_1)\cdots(s - s_{k-1})(s - s_{k+1})\cdots(s - s_n)}{(s_k - s_0)(s_k - s_1)\cdots(s_k - s_{k-1})(s_k - s_{k+1})\cdots(s_k - s_n)} \tag{4.10}$$

The variation of the Lagrange multiplier can be written as

$$\delta\lambda(x) = \sum_{i=1}^{n\lambda} N_i(s)\delta s_i \qquad x \in \Gamma_u \tag{4.11}$$

From chapter 3, the approximation $u^h(x)$ of the displacement function u(x) has the following form:

$$u^h(x) = \sum_{i=1}^{n} \varphi_i(x)u_i \tag{4.12}$$

where n is the number of nodes $x_i$ whose support includes point x, and $u_i$ are nodal displacement parameters. The method of construction of $\varphi_i(x)$ is given in detail in chapter 3. Substitution of Equation (4.12) into the boundary condition weak form Equation (4.8) leads to a set of linear algebraic constraint equations [32]:

$$HU = q \tag{4.13}$$

For free-vibration analysis, if $\bar{u} = 0$ on the essential boundary (clamped or simply supported), and hence Equation (4.13) becomes

$$HU = 0 \tag{4.14}$$

where H is called the constraints matrix, and u is the displacement parameter vector.

$$H_{ki} = \int_{\Gamma_u} N_k \psi_i d\Gamma \tag{4.15}$$

This is the discretized essential boundary condition for free vibration analysis. $H_{ki}$ are the nonzero entries in the matrix H. $N_k$ is the Lagrange interpolation function for node k on the essential boundary. $\psi_i$ is matrix of MLS shape functions, which depend on the type of boundary and the type of problem. This will be discussed in the following section.

It should be noted that the matrix H in Equation (4.14) is formed using the weak form of the constraint equation, which requires integration on the boundary and ensures the satisfaction of the essential boundary conditions on the entire essential boundary. We can obtain the discrete constraint equations directly using MLS approximation and obtain the H matrix. After H matrix is formed, we use orthogonal transform techniques to impose the constraints, as shown by Ouatouati and Johnson [31]. Such direct formulation ensures the constraints at the nodes on the essential boundary. In the following sections, the system equations of rod, beam, in-plane plate and thin plate vibration are described. First, the nodal matrices are calculated and then the assembly of the discrete equations are formed. These system equations include the global stiffness k, the global mass matrix and the constraints matrix H.

## 4.6 Axial Vibration

To illustrate the complete procedure of the Element Free Galerkin (EFG) method the simplest problem of one-dimensional axial vibration of a rod is investigated. Once the basic concepts are understood, the method is easily extended to two-dimensional problem. There are some subtle differences between the two-dimensional and one-dimensional formulation, and these will be discussed. In order to get natural frequencies of axial vibration, first, it is necessary to calculate the nodal stiffness matrix of the rod, the nodal mass matrix of the rod, and the constraint matrix and then these matrices must be assembled.

## 4.6.1 Nodal Stiffness Matrix and Mass Matrix for Rod

For a rod, each node has one displacement parameter. Each term in Equation (4.4) is calculated and Equation (4.12) is substituted into (4.4), for rod, differential operator is $L = \frac{\partial}{\partial x}$, therefore

$$\mathbf{Lu} = \sum_{i=1}^{n} L\varphi_i(x)u_i = \sum_{i=1}^{n} L\varphi_i u_i = \sum_{i=1}^{n} \frac{\partial \varphi_i}{\partial x} u_i = \sum_{i=1}^{n} B_i u_i \tag{4.16}$$

Here, $B_i$ stands for the first derivatives of shape function $\varphi_i(x)$ and u is axial displacement.

$$B_i = \frac{\partial \varphi_i}{\partial x} \tag{4.17}$$

From Equation (3.73)

$$\varphi_i(x) = \sum_{j=1}^{m} p_j(x)(A^{-1}(x)C(x))_{ji} = P^T(x)A^{-1}(x)C_i \tag{4.18}$$

For a rod, choose polynomial basis function as follows:

$$p^T(x) = \{1, x\} \tag{4.19}$$

$$A(x) = \sum_{I=1}^{n_{sn}} w(x - x_i)P(x_i)P^T(x_i) \tag{4.20}$$

$$C_i = w(x - x_i)p(x_i) \tag{4.21}$$

Derivatives of the MLS shape function $\varphi$ can be obtained by Equation (3.80)

Substituting Equation (4.12), (4.16) into Equation (4.4), for truss member D = E,

$$\int_\Omega \delta\left(\sum_{i=1}^{n} B_i u_i\right)^T \left(E\sum_{j=1}^{n} B_j u_j\right) d\Omega + \int_\Omega \rho\delta\left(\sum_{i=1}^{n} \varphi_i u_i\right)^T \left(\sum_{j=1}^{n} \varphi_j \ddot{u}_j\right) d\Omega = 0 \tag{4.22}$$

Examining the first term in Equation (4.22)

$$\int_\Omega \delta\left(\sum_{i=1}^{n} B_i u_i\right)^T \left(E\sum_{j=1}^{n} B_j u_j\right) d\Omega = \int_\Omega \delta\left(\sum_{i=1}^{n} u_i^T B_i^T\right)\left(E\sum_{j=1}^{n} B_j u_j\right) d\Omega$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta u_i^T \left(\int_\Omega B_i^T E B_j d\Omega\right) u_j \qquad (4.23)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \delta u_i^T \left(\int_\Omega B_i^T E B_j d\Omega\right) u_j = \sum_{i=1}^{n}\sum_{j=1}^{n} \delta u_i^T K_{ij} u_j \qquad (4.24)$$

$$K_{ij} = \int_\Omega B_i^T E B_j d\Omega = \int_0^l B_i^T E B_j A dx = \int_0^l B_i^T E A B_j dx \qquad (4.25)$$

where $K_{ij}$ is called nodal stiffness matrix and the dimension of $K_{ij}$ is $1 \times 1$. The dimension of stiffness matrix K for the problem is $n_t \times n_t$.

Next, the second term in Equation (4.22) is computed

$$\int_\Omega \rho\delta\left(\sum_{i=1}^{n} \varphi_i u_i\right)^T \left(\sum_{j=1}^{n} \varphi_j \ddot{u}_j\right) d\Omega = \int_\Omega \delta\left(\sum_{i=1}^{n} u_i^T \varphi_i^T\right)\rho\left(\sum_{j=1}^{n} \varphi_j \ddot{u}_j\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta u_i^T \left(\int_\Omega \varphi_i^T \rho \varphi_j d\Omega\right) \ddot{u}_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta u_i^T M_{ij} \ddot{u}_j \qquad (4.26)$$

$$M_{ij} = \int_\Omega \varphi_i^T \rho \varphi_j d\Omega = \int_0^l \varphi_i^T \rho \varphi_j A dx = \int_0^l \rho A \varphi_i^T \varphi_j dx \qquad (4.27)$$

where $M_{ij}$ is called nodal mass matrix and the dimension of $M_{ij}$ is $1 \times 1$. The dimension of mass matrix M for the problem is $n_t \times n_t$.

### 4.6.2 Constraint Matrix for Rod

From Equation (4.14), each term in the constraint matrix H should be calculated. $N_k$ is only calculated for boundary node. Shape function $\varphi_i$ can be calculated for each node. The contribution of each node to the constraints matrix H is decided by nodal weight function which means that each node whose support domain includes the boundary node has influence on the constraints matrix H. The node whose support domain includes boundary node contributes non-zero terms to the constraints matrix H. Otherwise, the node whose support domain does not include boundary node contributes zero to the constraints matrix H.

A clamped free rod has only one node on the boundary. Therefore the dimension of H matrix is $1 \times n_t$. Only one Lagrange interpolant is used to form constraint matrix. For one dimensional problem, $N_i = 1$, constraint matrix in Equation (4.10) can be simplified as

$$H_{1\times n_t} = [\varphi_1, \varphi_2, \cdots, \varphi_{n_t}] \tag{4.28}$$

Here, i is the node ID, which means node i is on the boundary.

## 4.7 Beam Bending Vibration

This study deals with thin beam or Euler-bernoulli beam. It assumes that plane sections which are normal to the undeformed neutral axis remain plane after bending and are normal to the deformed axis. With this assumption, the axial displacement, u(x,y), at a distance y from the neutral axis is

$$u(x,y) = -y\frac{\partial v}{\partial x} \tag{4.29}$$

where v=v(x) is the displacement of the neutral axis in the y-direction at the position x. The strain components $\varepsilon_x$ and $\gamma_{xy}$ are therefore

$$\varepsilon_x = \frac{\partial u}{\partial x} = -y\frac{\partial^2 v}{\partial x^2} \tag{4.30}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0 \tag{4.31}$$

$$L = -y\frac{\partial^2}{\partial x^2} \tag{4.32}$$

The normal stress is given by

$$\sigma_x = E\varepsilon_x \tag{4.33}$$

### 4.7.1 Nodal Stiffness Matrix and Mass Matrix for Beam

For a beam, polynomial basis functions are chosen as follows. Since the governing weak form contains second-order derivatives, a quadratic polynomial must be employed in Equation (3.44), for the purpose of consistency.

$$p^T(x) = \{1, x, x^2\} \tag{4.34}$$

From the Equation (4.4)

$$\int_\Omega \delta(\mathbf{Lu})^T(\mathbf{DLu})d\Omega + \int_\Omega \delta\mathbf{u}^T\rho\ddot{\mathbf{u}}d\Omega = 0 \tag{4.35}$$

For a beam, the nodal variable in the EFG method is also only one-deflection compared to the two in the element based FEM formulation (one deflection and one rotation). The deflection is denoted by v(x)

$$v^h(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) v_i \tag{4.36}$$

From the above equation,

$$\mathbf{Lu} = \mathrm{L}v = \sum_{i=1}^{n} \mathrm{L}\varphi_i(\mathbf{x}) v_i = \sum_{i=1}^{n} -y\frac{\partial^2 \varphi_i(\mathbf{x})}{\partial x^2} v_i = \sum_{i=1}^{n} -yB_i v_i \tag{4.37}$$

$B_i$ stands for the second derivative of the shape function

$$B_i = \frac{\partial^2 \varphi_i(\mathbf{x})}{\partial x^2} \tag{4.38}$$

For a beam D = E and hence

$$\int_\Omega \delta(\mathbf{Lu})^T(\mathrm{D}\mathbf{Lu})d\Omega = \int_\Omega \delta(\mathrm{L}v)^T(\mathrm{EL}v) = \int_\Omega \delta\left(\sum_{i=1}^{n} -yB_i v_i\right)^T \left(\mathrm{E}\sum_{j=1}^{n} -yB_j v_j\right) d\Omega$$

$$= \int_\Omega \delta\left(\sum_{i=1}^{n} -y v_i^T B_i^T\right)\left(\mathrm{E}\sum_{j=1}^{n} -yB_j v_j\right) d\Omega$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta v_i^T \left(\int_\Omega (-y)B_i^T \mathrm{E}(-y)B_j d\Omega\right) v_j \tag{4.39}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta v_i^T \left(\int_\Omega (-y)B_i^T \mathrm{E}(-y)B_j dA dx\right) v_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \delta v_i^T \left(\int B_i^T \mathrm{E} I_z B_j dx\right) v_j \tag{4.40}$$

$$I_Z = \int_A y^2 dA \tag{4.41}$$

$$K_{ij} = \int_0^l B_i^T \mathrm{E} I_z B_j dx \tag{4.42}$$

The dimension of nodal stiffness matrix $K_{ij}$ is 1x1. The dimension of stiffness matrix K for the problem is $n_t \times n_t$.

Next, the second term in Equation (4.22) is computed

$$\int_\Omega \rho\delta\left(\sum_{i=1}^{n}\varphi_i v_i\right)^T\left(\sum_{j=1}^{n}\varphi_j \ddot{v}_j\right)d\Omega = \int_\Omega \delta\left(\sum_{i=1}^{n} v_i^T\varphi_i^T\right)\rho\left(\sum_{j=1}^{n}\varphi_j\ddot{v}_j\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}\delta v_i^T\left(\int_\Omega \varphi_i^T\rho\varphi_j d\Omega\right)\ddot{v}_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}\delta v_i^T M_{ij}\ddot{v}_j \tag{4.43}$$

$$M_{ij} = \int_\Omega \varphi_i^T\rho\varphi_j d\Omega = \int_0^l \varphi_i^T\rho\varphi_j A dx = \int_0^l \rho A\varphi_i^T\varphi_j dx \tag{4.44}$$

$M_{ij}$ is called nodal mass matrix, the dimension being $1\times1$. The dimension of mass matrix M for the problem is $n_t \times n_t$.

### 4.7.2 Constraint Matrix for Beam

For cantilever beam, there is only one node on the essential boundary. The deflection and slope at the boundary must be constrained. Therefore, only one Lagrange interpolant is used to form the constraint matrix. For beam problem, $N_i = 1$, i is the node on the boundary, constraint matrix in Equation (4.10) can be simplified as

$$H_{2\times n_t} = \begin{Bmatrix} \varphi_1 & \varphi_2 \cdots \varphi_n \\ \varphi_{1,x} & \varphi_{2,x} \cdots \varphi_{n_t,x} \end{Bmatrix} \tag{4.45}$$

As for the rod, the contribution of each node to the constraints matrix H is decided by nodal weight function which means that each node whose support domain includes the boundary node has influence on the constraints matrix H. The node whose support domain includes the boundary node contributes non-zero terms to constraints matrix H. Otherwise, the node whose support domain does not include boundary node contribute zero to constraints matrix H.

## 4.8 In-Plane Plate Vibration

For this kind of plate, vibration takes place in its plane [33]. Each node has two degrees of freedom, displacements in x and y directions.

### 4.8.1 Nodal Stiffness Matrix

The polynomial base is chosen as follows:

$$p^T(x) = \{1, x, y\} \tag{4.46}$$

Displacement u in x direction and displacement v in y direction can be approximated by MLS method in the following forms:

$$u(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) u_i \tag{4.47}$$

$$v^h(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) v_i \tag{4.48}$$

Here x is a coordinate vector given as

$$\mathbf{x} = \begin{Bmatrix} x \\ y \end{Bmatrix} \tag{4.49}$$

Combining Equation (4.47) and (4.48), we obtain

$$\mathbf{u} = \begin{Bmatrix} u \\ v \end{Bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} \varphi_i & 0 \\ 0 & \varphi_i \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} = \sum_{i=1}^{n} \Phi_i \mathbf{u}_i \tag{4.50}$$

$$\mathbf{Lu} = \mathbf{L}\sum_{i=1}^{n} \mathbf{\Phi}_i \mathbf{u}_i = \sum_{i=1}^{n} \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} \varphi_i & 0 \\ 0 & \varphi_i \end{bmatrix} \mathbf{u}_i = \sum_{i=1}^{n} \begin{bmatrix} \varphi_{i,x} & 0 \\ 0 & \varphi_{i,y} \\ \varphi_{i,y} & \varphi_{i,x} \end{bmatrix} \mathbf{u}_i = \sum_{i=1}^{n} B_i \mathbf{u}_i \tag{4.51}$$

$$B_i = \begin{bmatrix} \varphi_{i,x} & 0 \\ 0 & \varphi_{i,y} \\ \varphi_{i,y} & \varphi_{i,x} \end{bmatrix} \tag{4.52}$$

For in-plane plate or 2 dimensional problems, the elastic matrix for the plane stress problem is

$$D = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \tag{4.53}$$

Substituting Equation (4.50) and (4.51) into (4.4) and following the same procedure as for the rod and beam, the nodal stiffness matrix for 2 dimensional problem is given by

$$K_{ij} = \int_{\Omega} B_i^T D B_j d\Omega = \int_{A} B_i^T D B_j t dx dy \tag{4.54}$$

The dimension of $K_{ij}$ is $2 \times 2$. The dimension of stiffness matrix K for the problem is $2n_t \times 2n_t$.

The nodal mass matrix is given by

$$M_{ij} = \int_{\Omega} \varphi_i^T \rho \varphi_j d\Omega = \int_{A} \varphi_i^T \rho \varphi_j t dx dy \tag{4.55}$$

and the dimension of $M_{ij}$ is $2 \times 2$. The dimension of mass matrix M for the problem is $2n_t \times 2n_t$.

### 4.8.2 Constraint Matrix for In-plane Plate

In this case, first-order Lagrange interpolant is chosen for the 2D problem. The Lagrange multiplier at s is interpolated by using two nodes that are located before and after s.

$$N_1(s) = \frac{s - s_2}{s_1 - s_2} \qquad N_2(s) = \frac{s - s_1}{s_2 - s_1} \tag{4.56}$$

Equation (4.9) is used for in-plane problem. Since there are two degrees of freedom at each node, there are two Lagrange multipliers $\lambda_i$ for each node along the boundary.

$$\boldsymbol{\lambda} = \sum_{i=1}^{n_\lambda} \begin{bmatrix} N_i & 0 \\ 0 & N_i \end{bmatrix} \begin{Bmatrix} \lambda_{ui} \\ \lambda_{vi} \end{Bmatrix} = \sum_{i=1}^{n_\lambda} \mathbf{N}_i \boldsymbol{\lambda}_i \tag{4.57}$$

$$\mathbf{N}_k = \begin{bmatrix} N_k & 0 \\ 0 & N_k \end{bmatrix} \tag{4.58}$$

where $n_\lambda$ is the number of nodes in each boundary cell used for interpolation. In 2D problem, cells for integration stiffness matrix and mass matrix are always square or rectangle. Comparing these cells with those for integration constraint matrix, the boundary cells are one-dimensional. Two nodes in each boundary cell are taken for Lagrange interpolation, and hence $n_\lambda = 2$ in this study.

For clamped boundary, the displacements of x,y direction must be imposed.

$$\psi_i = \Phi_i = \begin{bmatrix} \varphi_i & 0 \\ 0 & \varphi_i \end{bmatrix} \tag{4.59}$$

By using Equation (4.15) and Gauss points for integration, we can calculate $H_{ki}$ and obtain constraint matrix H. The dimension of H is $2n_t \times 2n_t$. $n_t$ is the total number of nodes of the problem.

## 4.9 Bending Vibration of Thin Plate

This section presents an EFG method for the free-vibration analysis of thin plates. Formulations of discretized system equations are based on Kirchhoff's thin plate theory [34].

### 4.9.1 Nodal Stiffness Matrix

Based on Kirchhoff's thin plate assumption, the deflection w(x) of its neutral plane at $\mathbf{x} = [x, y]^T$ can be taken as an independent variable, and the other two displacement components u(x) and v(x) can be expressed in terms of w(x). Moving least square (MLS) approximation is employed to approximate w(x) using nodes whose support domain includes x, and hence the two rotations are also approximated in relation to the deflection [35]. For thin plate, we should use more terms of polynomial basis functions. This is because the rotation that relates to the derivative of deflection are also field variables, and they depend on the deflection. Therefore, a higher order of consistency is required. In this work, complete second order (m=6) of polynomial basis functions are used.

$$p^T(x) = \{1, x, y, x^2xy, y^2\} \tag{4.60}$$

Using MLS approximation, the deflection of the plates can be approximated using parameters of nodal deflection, $w_i$, in the following form:

$$w(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i(\mathbf{x}) w_i \tag{4.61}$$

where n is the number of nodes whose support domain includes x. $\varphi_i(x)$ is the MLS shape function obtained in the same way as described in chapter 3.

The relationship between strain and deflection is given by

$$\varepsilon_{xx} = \frac{\partial v}{\partial x} = -z\frac{\partial^2 w}{\partial x^2} \tag{4.62}$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y} = -z\frac{\partial^2 w}{\partial y^2} \tag{4.63}$$

$$\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -2z\frac{\partial^2 w}{\partial x \partial y} \tag{4.64}$$

or in matrix form

$$\boldsymbol{\varepsilon} = \mathbf{L}w$$

where **L** is the following differential operator.

$$\mathbf{L} = \begin{Bmatrix} -z\dfrac{\partial^2}{\partial x^2} \\ -z\dfrac{\partial^2}{\partial y^2} \\ -2z\dfrac{\partial^2}{\partial x \partial y} \end{Bmatrix} \tag{4.65}$$

Substituting Equation (4.61) and (4.65) in (4.4)

$$\mathbf{L}\mathrm{w} = \mathbf{L}\sum_{i=1}^{n}\varphi_i(\mathbf{x})\mathrm{w}_i = \sum_{i=1}^{n}\mathbf{L}\varphi_i(\mathbf{x})\mathrm{w}_i = \sum_{i=1}^{n}\begin{bmatrix} -z\dfrac{\partial^2}{\partial x^2} \\ -z\dfrac{\partial^2}{\partial y^2} \\ -2z\dfrac{\partial^2}{\partial x\partial y}\end{bmatrix}\varphi_i(\mathbf{x})\mathrm{w}_i = \sum_{i=1}^{n} -zB_i\mathrm{w}_i \tag{4.66}$$

where,
$$B_i = \begin{bmatrix} -z\dfrac{\partial^2\varphi_i}{\partial x^2} \\ -z\dfrac{\partial^2\varphi_i}{\partial y^2} \\ -2z\dfrac{\partial^2\varphi_i}{\partial x\partial y}\end{bmatrix} \tag{4.67}$$

Substitution of Equation (4.65) in Equation (4.4) results in

$$\int_\Omega \delta(\mathbf{Lu})^T(\mathbf{DLu})d\Omega = \int_\Omega \delta(\mathrm{Lw})^T(\mathrm{DLw})d\Omega = \int_\Omega \delta\left(\sum_{i=1}^{n} -zB_i\mathrm{w}_i\right)^T\left(D\sum_{i=1}^{n} -zB_j\mathrm{w}_j\right)d\Omega$$

$$= \int_\Omega \delta\left(\sum_{i=1}^{n} -z\mathrm{w}_i^TB_i^T\right)\left(D\sum_{i=1}^{n} -zB_j\mathrm{w}_j\right)d\Omega$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}\delta\mathrm{w}_i\left(\int_\Omega z^2B_i^TDB_jd\Omega\right)\mathrm{w}_j \tag{4.68}$$

Nodal stiffness matrix is given by

$$K_{ij} = \int_\Omega z^2B_i^TDB_jd\Omega = \int_\Omega z^2B_i^TDB_jdAdz \tag{4.69}$$

Integrating with respect to z gives the nodal stiffness matrix as

$$K_{ij} = \int_\Omega z^2B_i^TDB_jdAdz = \int_A \frac{t^3}{12}B_i^TDB_jdA = \int_A \frac{t^3}{12}B_i^TDB_jdxdy \tag{4.70}$$

where D is defined by Equation (4.53). The dimension of $K_{ij}$ is $1\times1$. The dimension of stiffness matrix K for the problem is $n_t \times n_t$.

The nodal mass matrix is given by

$$M_{ij} = \int_{\Omega} \varphi_i^T \rho \varphi_j d\Omega = \int_{A} \varphi_i^T \rho \varphi_j t dx dy \tag{4.71}$$

The dimension of $M_{ij}$ is $1 \times 1$. The dimension of mass matrix M for the problem is $n_t \times n_t$.

## 4.9.2 Constraint Matrix for Thin Plate

As in in-plane or 2D problem, first-order Lagrange interpolant is chosen for thin plate vibration. The Lagrange multiplier at s is interpolated by using two nodes that are located before and after s.

$$N_1(s) = \frac{s - s_2}{s_1 - s_2} \qquad N_2(s) = \frac{s - s_1}{s_2 - s_1} \tag{4.72}$$

Applying Equation (4.9) for thin plate problem, there are two Lagrange multipliers $\lambda_i$ for each node along the boundary since there are two degree of freedom at each node ( the deflection and the rotation on the boundary about the boundary line).

$$\lambda = \sum_{i=1}^{n_\lambda} \begin{bmatrix} N_i & 0 \\ 0 & N_i \end{bmatrix} \begin{Bmatrix} \lambda_{ui} \\ \lambda_{vi} \end{Bmatrix} = \sum_{i=1}^{n_\lambda} \mathbf{N}_i \lambda_i \tag{4.73}$$

$$\mathbf{N}_k = \begin{bmatrix} N_k & 0 \\ 0 & N_k \end{bmatrix} \tag{4.74}$$

where $n_\lambda$ is the number of nodes in each boundary cell used for interpolation. In thin plate problem, as in 2D problem, cells for integration stiffness matrix and mass matrix are always square or rectangle. Comparing these cells with the cells for integration constraint matrix, the boundary cells are one-dimensional. Two nodes are taken in each boundary cell for Lagrange interpolation, choosing $n_\lambda = 2$ in this study.

For clamped boundary, the displacements of the deflection and the rotation about boundary line must be imposed.

$$\psi_i = \Phi_i = \begin{bmatrix} \varphi_i \\ \varphi_{i,n} \end{bmatrix} \tag{4.75}$$

For simply supported boundary

$$\psi_i = \Phi_i = \begin{bmatrix} \varphi_i \\ \dfrac{\partial^2 \varphi_i}{\partial n^2} \end{bmatrix} \tag{4.76}$$

where n denotes the normal of the boundary of the problem domain.

By using Equation (4.15) and Gauss points for integration, $H_{ki}$ can be calculated to obtain constraint matrix H. The dimension of H is $2n_t \times n_t$, $n_t$ is the total number of nodes of the problem.

## 4.10 Numerical Integration

In a Galerkin formulation, numerical integration is used to evaluate the integrals in the weak form. The reason is the integrals cannot be evaluated analytically. Several different techniques have been proposed to numerically integrate the Galerkin weak form in meshfree method. In the present study Gaussian quadrature is employed.

### 4.10.1 Gauss Points and Weight Coefficients for Each Cell

In order to evaluate the integrals in Equation (4.25), (4.27), (4.42), (4.44), (4.54), (4.55), (4.70), and (4.71), it is necessary to define integration cells over the domain of problem. Background cells are needed in order to find quadrature points in each cell for integration. Cells are similar to the mesh of elements in FEM and no overlap or gap is permitted. In contrast to the mesh in FEM, the background cell in EFG is used only for the integration of the system matrices, and not for shape function construction. In principle, the background cells can be totally independent of the arrangement of nodes. The only consideration in design cells of the background mesh is to ensure the accuracy of integration for the system matrices.

The nodal stiffness matrix and mass matrix for different type elastic bodies are deduced in different sections 4.6, 4.7, 4.8, 4.9. The shape functions of EFG method constructed by MLS method are rational functions of the spatial coordinates and hence the intervals of physical coordinates should be changed to natural coordinates [-1,+1] if Gaussian quadrature method is employed for integration.

Transformation of integrals

One-dimensional integration:

$$\int f(x)dx = \int_{-1}^{1} f(x(\xi))Jd\xi = \int_{-1}^{1} f(\xi)Jd\xi = \sum_{i=1}^{n} W_i f(\xi_i)\det[J] \tag{4.77}$$

where $W_i$ is weight coefficients, $\xi_i$ is Gauss points, J is Jacobian matrix.

Two-dimensional integration:

$$\int_A f(x,y)dxdy = \int_{-1}^{1}\int_{-1}^{1} f(x(\xi(\eta)), y(\xi(\eta))Jd\xi d\xi = \sum_{i=1}^{n}\sum_{j=1}^{n} W_i W_j f(\xi_i, \eta_j)\det[J] \tag{4.78}$$

where $W_i$, $W_j$ are weight coefficients, $(\xi_i, \eta_i)$ are Gauss points

Table 4.1

Integration points and weight coefficients for the Gauss integration formula

| Number of Gauss Points in the interval [-1,1] | $\pm\xi_i$ | $W_i$ |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 0.57735026918963 | 1 |
| 3 | 0.77459666924148 | 5/9 |
| | 0 | 8/9 |
| 4 | 0.86113631159405 | 0.34785484513745 |
| | 0.33998104358486 | 0.65214515486255 |
| 6 | 0.93246951420315 | 0.17132449237917 |
| | 0.66120938646626 | 0.36076157304814 |
| | 0.23861918608319 | 0.46791393457269 |
| 9 | 0.96816023950763 | 0.08127438836157 |
| | 0.83603110732664 | 0.18064816069486 |
| | 0.61337143270059 | 0.26061069640294 |
| | 0.32425342340381 | 0.31234707704001 |
| | 0 | 0.33023935500126 |

### 4.10.2 Jacobian Matrix for Each Cell

Gaussian quadrature method is used to calculate integrals, and hence it is necessary to transform the coordinates from physical coordinates into natural coordinates, because Gauss points can be found only in the natural coordinates. For one-dimensional problem, the relationships of physical coordinates and natural coordinates are given as follows [36]:

$$x(\xi) = \sum_{i=1}^{2} N_i(\xi) x_i \tag{4.79}$$

where $x_i$ are the physical coordinates of vertices of cell i. Note that the field nodes and the integration cell vertices need not coincide. For the sake of simplicity, sometimes the integration cells are chosen to coincide with the intervals between the field nodes. For one-dimensional problem, cells are shown in the following graph, where each has two vertices.

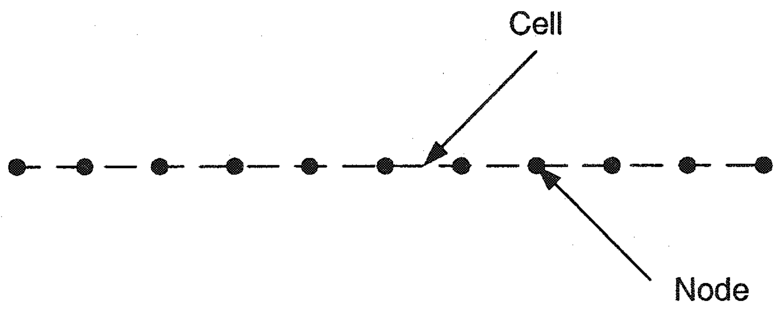

Figure 4.2 One-dimensional nodes and cells

Function $N_i(\xi)$ are defined

$$N_1 = \frac{1}{2}(1-\xi) \tag{4.80}$$

$$N_2 = \frac{1}{2}(1+\xi) \tag{4.81}$$

and the Jacobian matrix is given by

$$J = (x_2 - x_1)/2 \tag{4.82}$$

For two-dimensional problem, the relationships of physical coordinates and natural coordinates are given as follows:

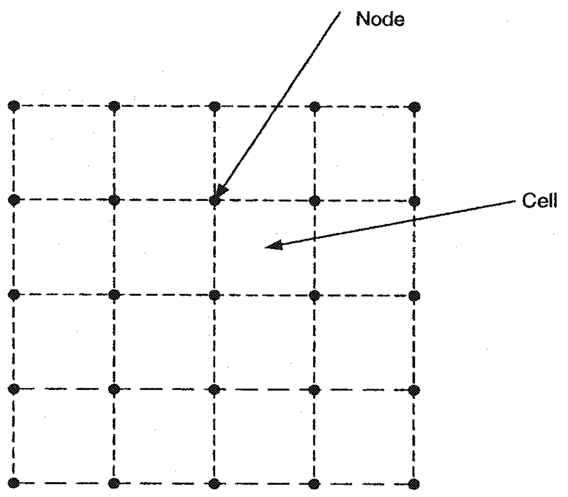

Figure 4.3 Cells coincide with the intervals between the field nodes

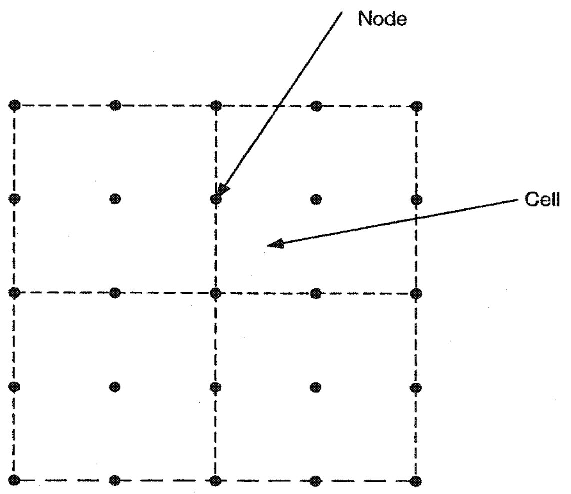

Figure 4.4 Cells do not coincide with the intervals between the field nodes

From Figure 4.3 and Figure 4.4, each cell has four vertices, and the relationships of physical coordinates and natural coordinates are given as follows [36]:

$$x(\xi,\eta) = \sum_{i=1}^{4} N_i(\xi,\eta)x_i \tag{4.83}$$

$$y(\xi,\eta) = \sum_{i=1}^{4} N_i(\xi,\eta)y_i \tag{4.84}$$

where $(x_i, y_i)$ are the physical coordinates of vertices of cell i. In each cell, mapping functions $N_i(\xi,\eta)$ are defined as

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta) \tag{4.85}$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta) \tag{4.86}$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta) \tag{4.87}$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta) \tag{4.88}$$

and the Jacobian matrix is given by

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{4.89}$$

Substituting Equation (4.83), (4.84) into (4.89) gives

$$[J] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{4.90}$$

Note, that unlike FEM, the same functions have been used to define both the integration and shape function. In EFG method, $N_i$ are used only for mapping cell from physical coordinates to natural coordinates in order to get Gauss points for integration, and MLS method is used for shape function construction. The shape functions built by MLS are the function of spatial coordinates (Equation (4.91), (4.92)). When Gauss points are obtained, transformation or mapping must be made by changing the Gauss points (in natural coordinates) to spatial coordinates (physical coordinates) using mapping function (Equation (4.79) for one-dimensional, Equation (4.83), (4.84) for two-dimensional), and then shape function can be built based on the nodes whose influence domain includes the gauss point (physical coordinate).

$$K_{ij} = \int_l B^T DB dx = \int_l B(x)^T DB(x) dx = \int_{-1}^{1} B(x(\xi))^T DB(x(\xi)) \det J d\xi \tag{4.91}$$

$$K_{ij} = \int_A B^T DB dA = \int_A B(x)^T DB(x) dxdy$$

$$= \int_{-1}^{1}\int_{-1}^{1} B(x(\xi,\eta))^T DB(x(\xi,\eta)) \det J d\xi d\eta \tag{4.92}$$

## 4.11 Assembly

In the assembly of the nodal stiffness matrices equation for rod, beam, in plane plate, and thin plate, the procedure are almost the same.

Rewriting the Equation (4.24), because the integration is over the entire problem domain, and all the nodes can be involved. Therefore, the summation limits n has to be changed to $n_t$, which is the total number of nodes in the entire problem domain. The contribution of each node to the system matrices is determined by weight function that controls the size of support domain at node.

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\delta u_i^T K_{ij} u_j = \sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\delta u_i^T K_{ij} u_j = \delta u_1^T K_{11} u_1 + \delta u_1^T K_{12} u_2 + \cdots + \delta u_1^T K_{1n_t} u_{n_t}$$

$$+ \delta u_2^T K_{11} u_1 + \delta u_2^T K_{12} u_2 + \cdots + \delta u_2^T K_{1n_t} u_{n_t}$$

$$\vdots$$

$$+ \delta u_n^T K_{11} u_1 + \delta u_n^T K_{12} u_2 + \cdots + \delta u_n^T K_{1n_t} u_{n_t}$$

$$= \delta U^T K U \tag{4.93}$$

where K is the global stiffness matrix assembled using the nodal stiffness matrix in the form

$$K = \begin{bmatrix} K_{11} & K_{12} & \cdots K_{1n_t} \\ K_{21} & K_{22} & \cdots K_{2n_t} \\ \vdots & \vdots & \ddots \vdots \\ K_{n_t1} & K_{n_t2} & \cdots K_{n_tn_t} \end{bmatrix} \tag{4.94}$$

The dimension of K matrix for rod, beam, and thin plate should be $n_t \times n_t$, for in plane plate the dimension of K matrix is $2n_t \times 2n_t$. and $n_t$ is the total number of nodes in the problem domain.

M is the global stiffness matrix assembled using the nodal stiffness matrix in the form

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots M_{1n_t} \\ M_{21} & M_{22} & \cdots M_{2n_t} \\ \vdots & \vdots & \ddots \vdots \\ M_{n_t1} & M_{n_t2} & \cdots M_{n_tn_t} \end{bmatrix} \tag{4.95}$$

The dimension of M matrix for rod, beam, and thin plate should be $n_t \times n_t$, for in plane plate the dimension of M matrix is $2n_t \times 2n_t$ and $n_t$ is the total number of nodes in the problem domain.

## 4.12 Flowchart of 1D EFG Program and 2D EFG Program

Table 4.2 Flowchart of 1D EFG Program

| | |
|---|---|
| 1 | Set up nodal coordinates and integration cells. |
| 2 | Set parameters for weight function (support domain), material properties. |
| 3 | Set up integration points, weights, and Jacobian for each cell. |
| 4 | Loop over integration points. |
| 4.a | Calculate weight at each node for given integration point $x_Q$. |
| 4.b | Calculate shape functions and derivatives at point $x_Q$. |
| 4.c | Assemble contributions to K matrix and M matrix. |
| 4.d | Form constraints matrix H at first integration point. |
| 5. | Orthogonal transformation techniques (Singular Value Decomposition H). |
| 6. | Solve for natural frequencies and mode shapes. |

Table 4.3 Flowchart of 2D EFG Program

1. Define the physical dimensions and material properties.
2. Define the plane stress D matrix
3. Set up the nodal coordinates
4. Determine the domain of influence for each node
5. Set up quadrature cells in the domain.
6. Set up Gauss points, weight, and Jacobian for each cell.
7. Loop over Gauss points.
    - 7.a Determine nodes in the neighborhood of the Gauss point.
    - 7.b Determine weights (support domain), shape functions and shape function derivatives for nodes 1 to n.
    - 7.c Add contributions to K and M matrices.
8. Determine nodes on essential boundary.
9. Set up Gauss points along the essential boundary.
10. Integrate Lagrange multipliers along essential boundary to form H matrix.
11. Enforce essential boundary conditions using orthpgonal techniques.
12. Solve for natural frequencies and mode shapes.

# Chapter 5

# Numerical Results

In this chapter, the natural frequencies of rod, beam, in-plane plate, and thin plate vibrations by EFG method are presented. The results show good agreements with analytical or FEM results. The natural frequencies of cantilever beam and fully clamped thin plate by different techniques are also given. The techniques include Rayleigh-Ritz method, Collocation method, Galerkin method (only for cantilever beam), Element Free Galerkin method (EFG), and Finite Element Method (FEM). The comparisons of global stiffness matrix and global mass matrix are made between Element Free Galerkin (EFG) and Finite Element Method (FEM). Five nodes clamped-free rod and cantilever beam model are used to illustrate the system matrices.

## 5.1 Axial Vibration

The natural frequencies of the clamped-free rod are calculated using Element Free Galerkin (EFG) method. The parameters of the rod used are shown in Table 5.1.

Table 5.1 The parameters of the rod

| | |
|---|---|
| Young's modulus | $E = 2.1e{+}11 N/m^2$ |
| Cross-section area | $A = 0.001\, m^2$ |
| Mass density | $\rho = 7800 kg/m^3$ |
| Length | $L = 1m$ |

The non-dimensional $\omega(\rho L^2 / E)^{\frac{1}{2}}$ natural frequencies are shown in the Table 5.2. The results are presented in Table 5.2 and compared with the exact solutions [36]. It is seen that the EFG solution is very close to the exact solution. As the number of nodes are increased, the first mode natural frequency deteriorates slightly, however, second natural frequency improves.

Table 5.2 Comparison of approximate frequencies with exact solution for a rod

| EFG Solution | | | | Exact |
|---|---|---|---|---|
| Mode | 11nodes | 21nodes | 101nodes | Solution |
| 1 | 1.570944884085 | 1.570812509147 | 1.5707959211 | 1.571 |
| 2 | 4.717615059605 | 4.712956817113 | 4.7123914588 | 4.712 |

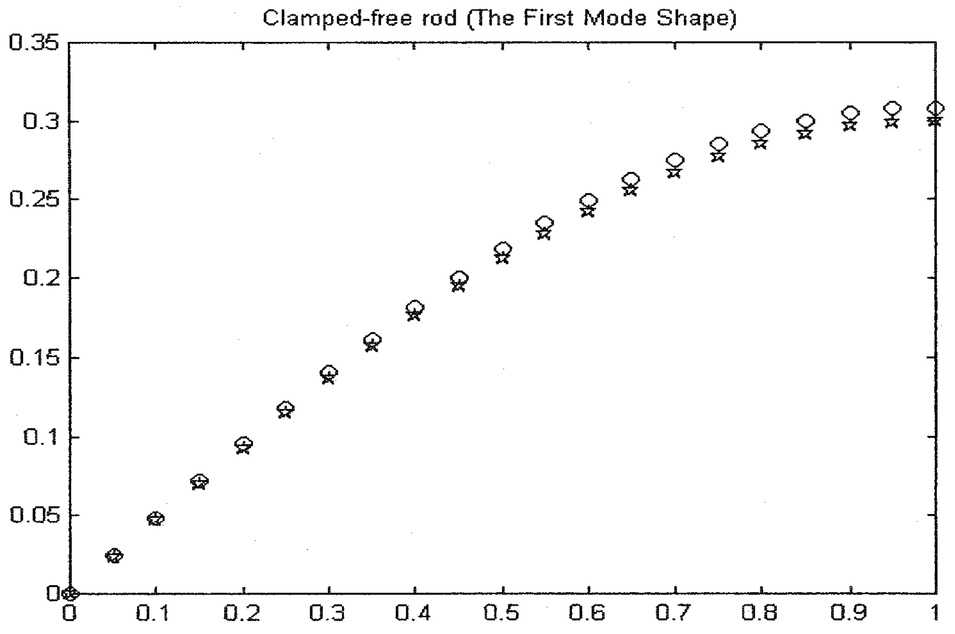

Figure 5.1 The first mode shape of clamped-free rod

(○ - EFG (21 nodes), ☆ - Analytical )

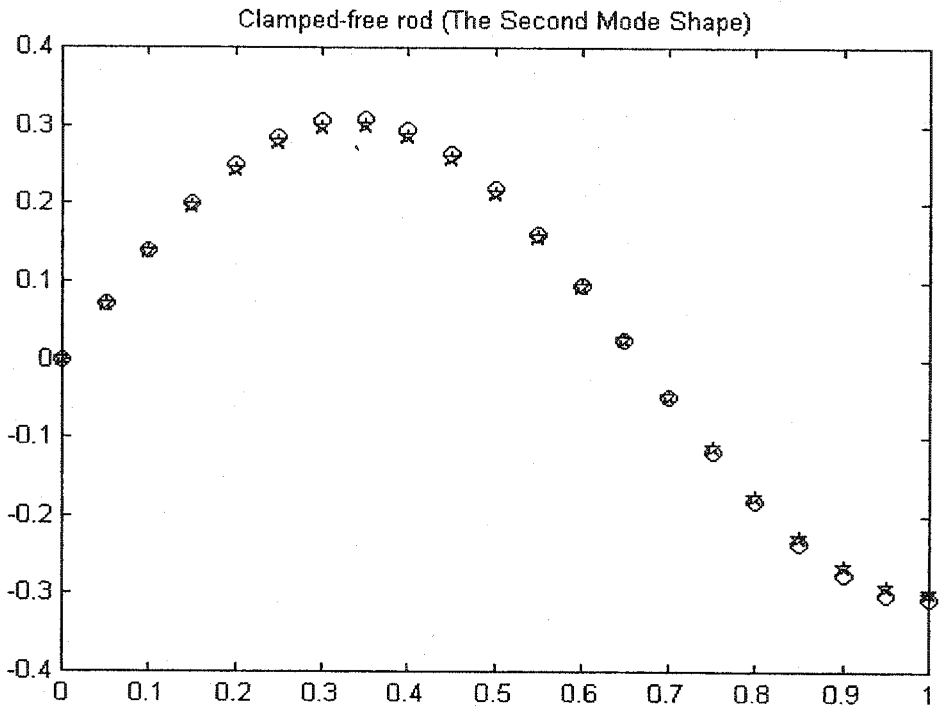

Figure 5.2 The second mode shape of clamped-free rod

(○ - EFG (21 nodes), ☆ - Analytical )

The first and second mode shapes obtained by the EFG method and analytical method are respectively shown in Figure 5.1 and Figure 5.2. In the above figures, the small circles denote the solution by EFG method and the small stars denote the solution by analytical method. Both methods agree well with the exact solutions.

## 5.2 Bending Vibration

The natural frequencies of the cantilever beam are calculated using Element Free Galerkin (EFG) method. The results are compared with the exact solution [29].

Comparing mode shapes with those obtained by other methods [37], good agreements have been noted. The parameters of the beam are shown in Table 5.3. The non-dimensional $\omega\left(\rho A L^4 / EI\right)^{\frac{1}{2}}$ natural frequencies are shown in the Table 5.4.

Table 5.3 The parameters of the beam

| | |
|---|---|
| Young's modulus | $E = 2.1e+11\ N/m^2$ |
| Mass Density | $\rho = 7800\ kg/m^3$ |
| Thickness | $b = 0.01$ m |
| Height | $h = 0.1$ m |
| Length | $L = 1$ m |

Table 5.4

Comparison of approximate frequencies with exact solution for a cantilever beam

| EFG Solution (one Gauss point in each cell) | | | | Exact |
|---|---|---|---|---|
| Mode | 11nodes | 21nodes | 101nodes | Solution |
| 1 | 3.36082381988 | 3.4761724364 | 3.517057712 | 3.516 |
| 2 | 20.70925433576 | 21.5734902809 | 22.029824631 | 22.035 |
| 3 | 55.71067926664 | 59.3750974434 | 61.631311291 | 61.695 |
| 4 | 102.63237012362 | 113.4030732297 | 120.616844033 | 120.903 |
| 5 | 157.80732854596 | 181.3289603744 | 199.052089196 | 199.859 |
| 6 | 221.73440699330 | 260.1449748330 | 296.730459349 | 298.556 |

Table 5.5

Comparison of approximate frequencies with exact solution for a cantilever beam

| EFG Solution (four Gauss points in each cell) | | | | Exact |
|---|---|---|---|---|
| Mode | 11nodes | 21nodes | 101nodes | Solution |
| 1 | 3.426542879540 | 3.49223557515 | 3.5163347157 | 3.516 |
| 2 | 21.673001236786 | 21.85105627288 | 22.1250283237 | 22.035 |
| 3 | 61.093533982430 | 60.98062148087 | 61.7235271928 | 61.695 |
| 4 | 120.369177663624 | 118.88675681855 | 120.8555720551 | 120.903 |
| 5 | 199.423537674733 | 195.22004936751 | 199.6505463814 | 199.859 |
| 6 | 296.386780227024 | 289.27306272927 | 298.0227146553 | 298.556 |

Table 5.6

Comparison of approximate frequencies with exact solution for a cantilever beam

(one, two, and four Gauss points in each cell)

| EFG Solution (change the number Gauss points in each cell) | | | | Exact |
|---|---|---|---|---|
| Mode | 11nodes(1GP) | 11nodes(2GP) | 11nodes(4GP) | Solution |
| 1 | 3.36082381988 | 3.422311359625 | 3.426542879540 | 3.516 |
| 2 | 20.70925433576 | 21.622209848823 | 21.673001236786 | 22.035 |
| 3 | 55.71067926664 | 60.771146115879 | 61.093533982430 | 61.695 |
| 4 | 102.63237012362 | 118.979696150096 | 120.369177663624 | 120.903 |
| 5 | 157.80732854596 | 194.791798611303 | 199.423537674733 | 199.859 |
| 6 | 221.73440699330 | 2.83755585408958 | 296.386780227024 | 298.556 |

From Table 5.4, it is shown that with one Gauss point in each background cell, the results improve with the number of nodes. From Table 5.6, with 11 nodes in a cantilever beam, the results improve considerably when the Gauss points in each cell are increased from 1, 2 to 4. From Table 5.5, the results also improve considerably when the Gauss points in each cell are increased from 1 to 4. The reason is that the background cells are not the best local domains for the spatial integration. The accuracy does not always improve with increasing the number of Gauss points.

The mode shapes by EFG for the first, second, third, fourth and fifth modes are shown respectively in Figure 5.3, 5.4, 5.5, 5.6 and 5.7. They agree well with the exact results plotted alongside (characteristic functions representing normal modes of vibration of a beam). Small circles represent EFG and small stars denote exact solutions.

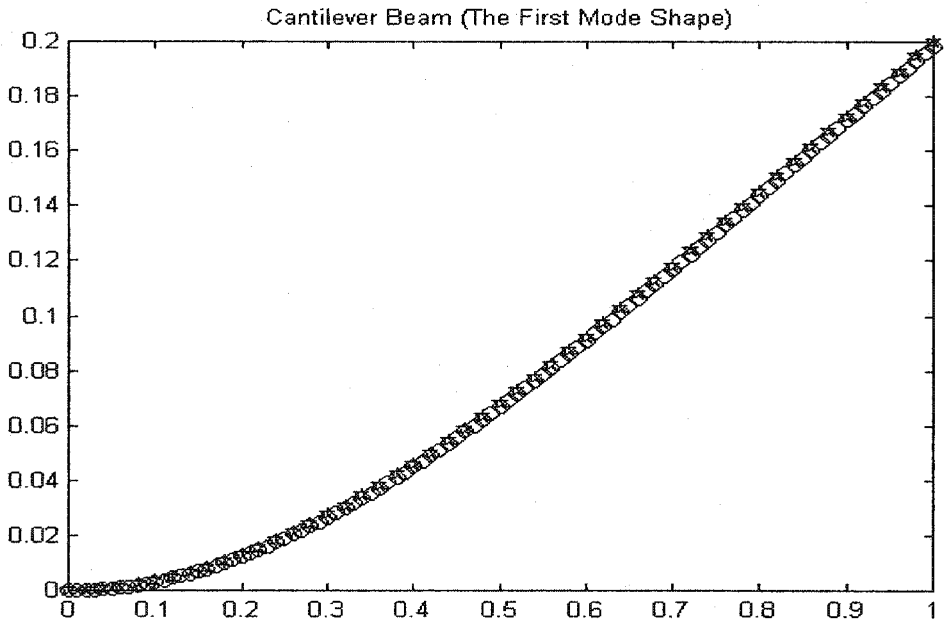

Figure 5.3 The first mode shape of cantilever beam

(○ - EFG (101 nodes), ☆ - Analytical)

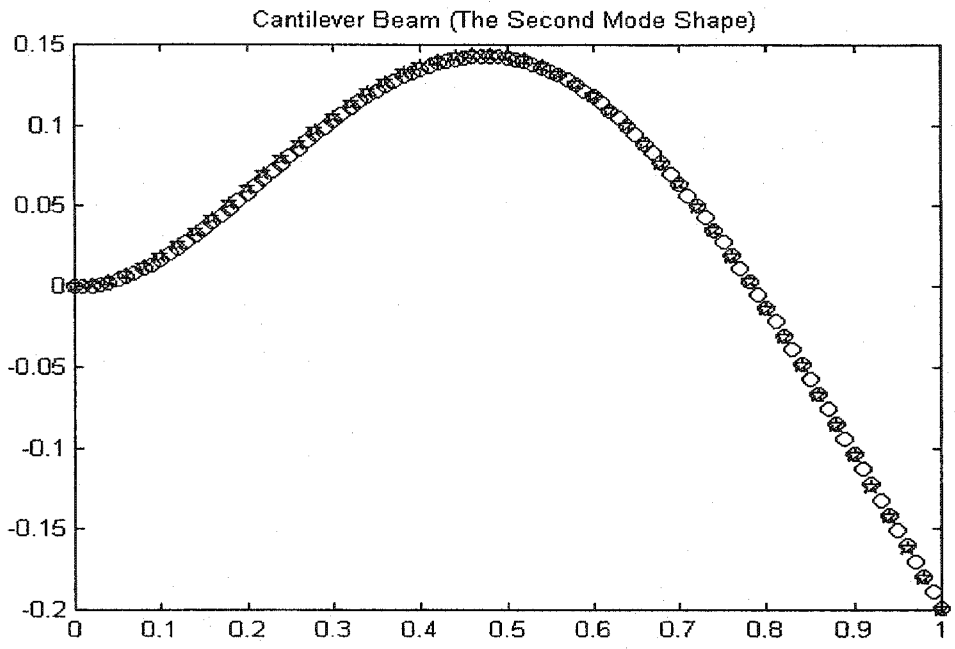

Figure 5.4 The second mode shape cantilever beam

(○ - EFG (101 nodes), ☆ - Analytical)

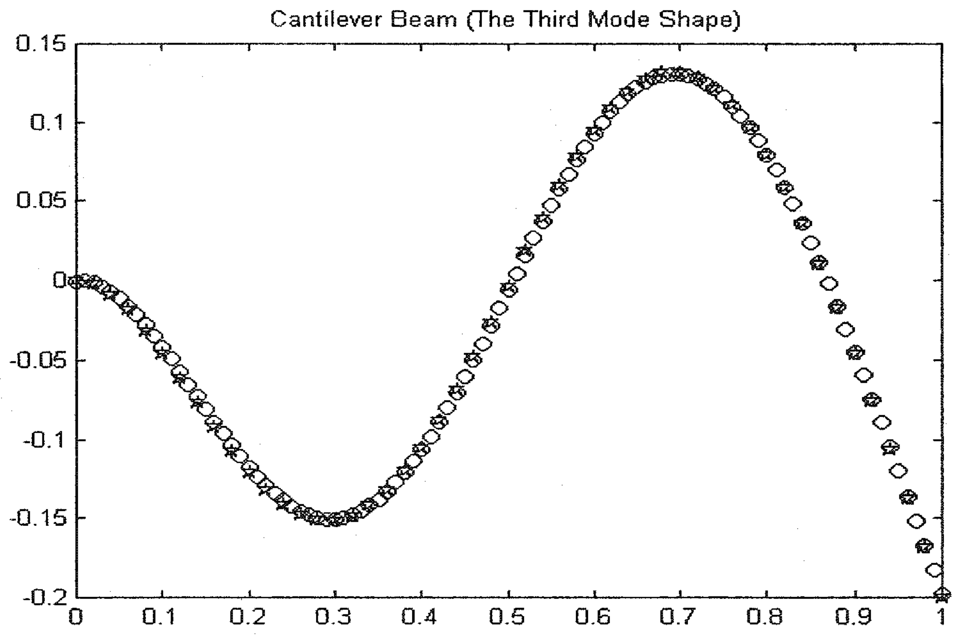

Figure 5.5 The third mode shape of cantilever beam

(○ - EFG (101 nodes), ☆ - Analytical)

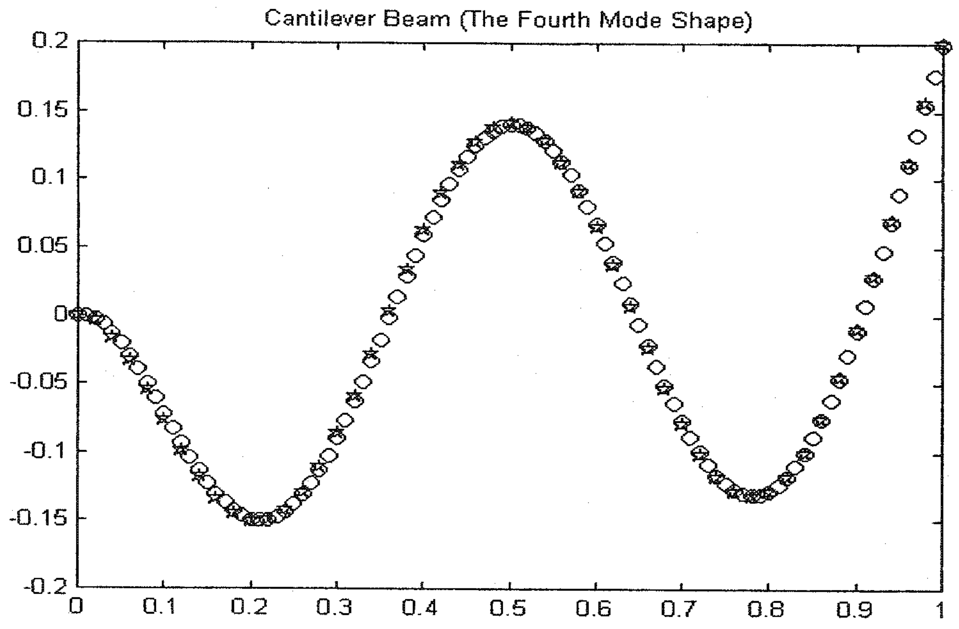

Figure 5.6 The fourth mode shape of cantilever beam

(○ - EFG (101 nodes), ☆ - Analytical)

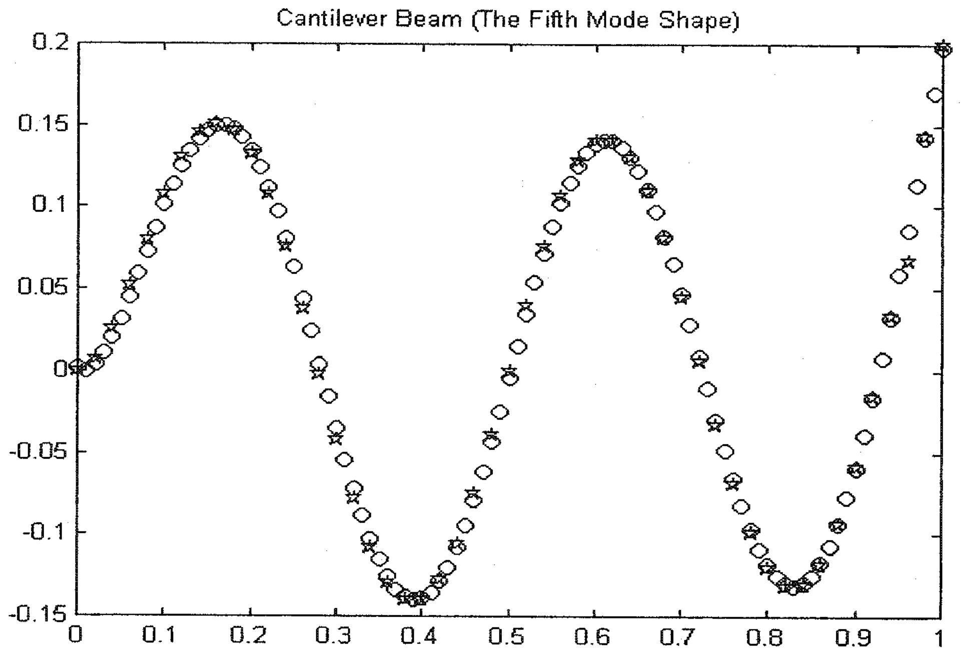

Figure 5.7 The fifth mode shape of cantilever beam

(○ - EFG (101 nodes), ☆ - Analytical)

## 5.3 In-plane vibration of plate

The first five natural frequencies for in-plane vibration of the plate are calculated using EFG method. The results are compared with analytical solution obtained by treating the plate as a deep beam. The results are obtained for $E = 2.1\text{x}10^{11}\,\text{N/m}^2$, $\nu = 0.3$, $\rho = 8000\text{kg/m}^3$, thickness of 0.001m, length of plate L=0.1m, and height of the plate=0.01m. Natural frequency results obtained by EFG are listed in Table 5.5. The results [32] [38] obtained by FEM software ABAQUS, the NBNM method, and MLPG method are also listed in the table 5.7.

Table 5.7

Natural Frequencies (Hz) of cantilever plate (in plane) by using EFG

(18 nodes)

| Mode | EFG(18nodes) | MLPG(63nodes) | FEM(ABAQUS) (63nodes) | NBNM(63nodes) |
|---|---|---|---|---|
| 1 | 829.582790048 | 919.47 | 870 | 926.10 |
| 2 | 5011.042989557 | 5732.42 | 5199 | 5484.11 |
| 3 | 12855.620073909 | 12983.25 | 12830 | 12831.88 |
| 4 | 13905.421595458 | 14808.64 | 13640 | 14201.32 |
| 5 | 37216.715201917 | 26681.81 | 24685 | 25290.04 |
| 6 | 38549.291809766 | 38961.74 | 37477 | 37350.18 |

Table 5.8

Natural Frequencies (Hz) of cantilever plate (in plane) by using EFG (55 nodes)

| Mode | EFG(55nodes) | MLPG(306nodes) | FEM(ABAQUS) (306nodes) | NBNM(306nodes) |
|---|---|---|---|---|
| 1 | 824.58645569 | 824.44 | 830 | 844.19 |
| 2 | 4947.71358351 | 5070.32 | 4979 | 5051.21 |
| 3 | 12833.96496038 | 12894.73 | 12826 | 12827.60 |
| 4 | 13034.93824201 | 13188.12 | 13111 | 13258.21 |
| 5 | 23693.43803877 | 24044.43 | 23818 | 23992.82 |
| 6 | 36230.55808267 | 36596.15 | 36308 | 36432.15 |

From Table 5.7 and Table 5.8, one can observe that the results by EFG method are in good agreement with those obtained using FEM, MLPG and NBNM method. The convergence of the EFG method is also demonstrated in these two tables. As the number of nodes increases, the results obtained by EFG approach the FEM results. The lowest five vibration modes by EFG are plotted in Figure 5.8 to Figure 5.12. Asterisk denotes the deformed shape while small circle denotes the node original position.

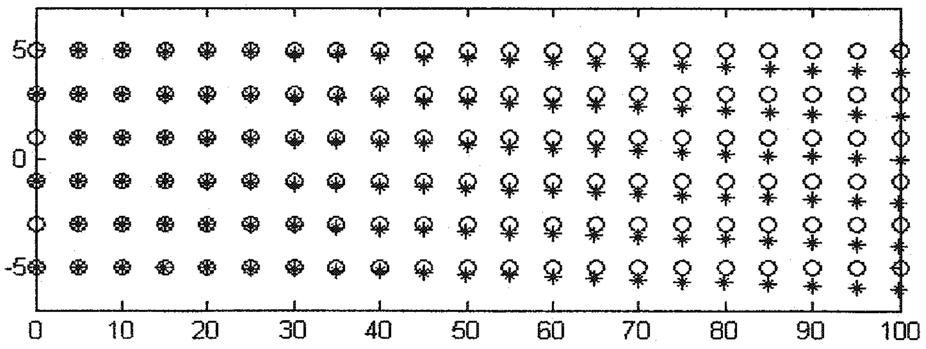

Figure 5.8 In-plane plate vibration mode shape 1 (○ - Undeformed, ∗ - Deformed)

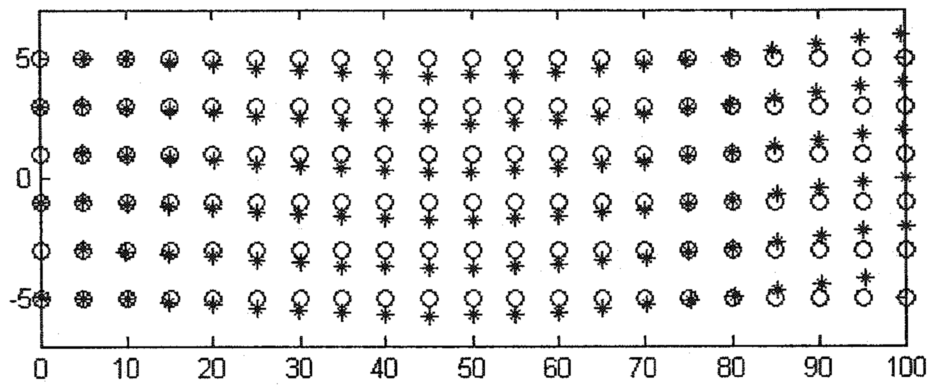

Figure 5.9 In-plane plate vibration mode shape 2 (○ - Undeformed, * - Deformed)

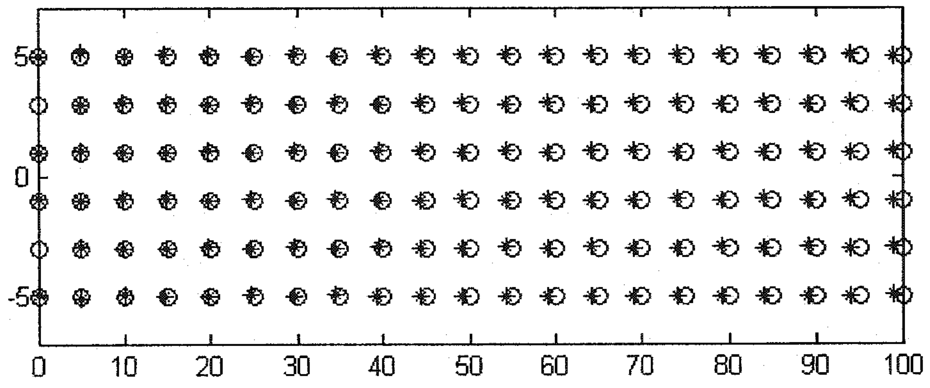

Figure 5.10 In-plane plate vibration mode shape 3 (○ - Undeformed, * - Deformed)

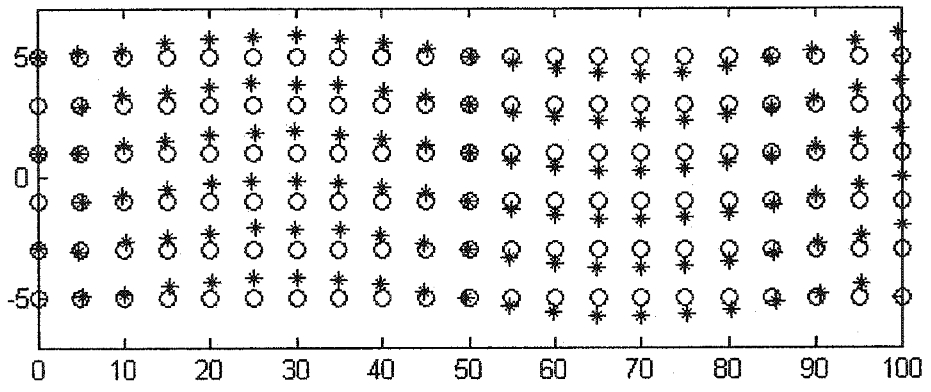

Figure 5.11 In-plane plate vibration mode shape 4 (○ - Undeformed, * - Deformed)

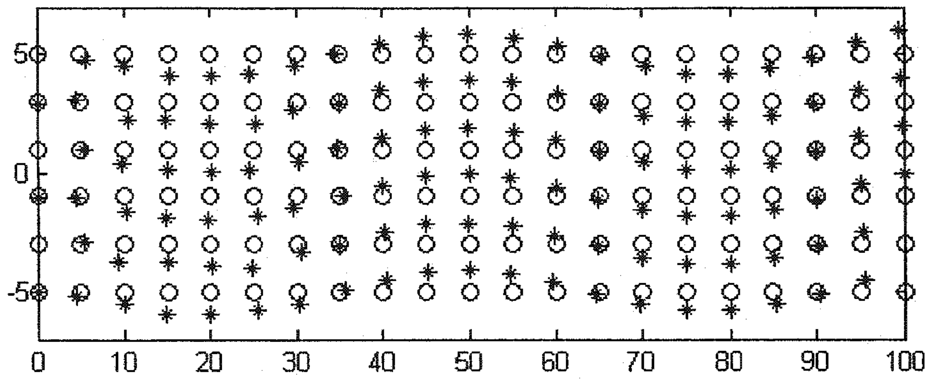

Figure 5.12 In-plane plate vibration mode shape 5 (○ - Undeformed, ✱ - Deformed)

Comparison of the EFG results with G. R. Liu (2001) results by MLPG (Meshless Local Petrov Galerkin Method) [32] reveals that they are almost identical. The correct results confirm that EFG method is effective not only for one-dimensional but also for two-dimensional problems. Based on the research on 2D problems by EFG, some modifications are made and the thin plate vibration problems by EFG are obtained successfully. The following section presents the natural frequencies of thin plate by EFG.

## 5.4 Natural Frequency Analysis of Thin Square Plates

Table 5.9 Parameters of a square plate

| | |
|---|---|
| Young's modulus | $E = 20x10^9 N/m^2$ |
| Mass density | $\rho = 3000 kg/m^3$ |
| Length | $a = b = 4m$ |
| Thickness | $t = 0.2m$ |
| Poisson's ratio | $\nu = 0.3$ |

The first six natural frequencies (non-dimensional values $\omega(\rho a^4 h / D_0)^{\frac{1}{2}}$ of thin plate with different boundary conditions are shown in Table 5.10, 5.11, 5.12, and 5.13. Analytical solution can be found in paper [29].

Table 5.10 The natural frequencies for thin plate (CFFF)

| Mode | EFG(Different Nodes) | | | | Analytical Solution |
|---|---|---|---|---|---|
| | 7x7 | 9x9 | 12x12 | 13x13 | |
| 1 | 3.482522761601 | 3.47929020449 | 3.47755348208 | 3.47609613970 | 3.474 |
| 2 | 8.527519654411 | 8.52117372675 | 8.51732996433 | 8.51501859204 | 8.591 |
| 3 | 21.415469873286 | 21.42294277873 | 21.40480355088 | 21.38587809326 | 21.298 |
| 4 | 27.290139042041 | 27.29607289008 | 27.27478680332 | 27.26142830142 | 27.154 |
| 5 | 31.113034432349 | 31.08944571774 | 31.06457781301 | 31.04610463331 | 31.036 |
| 6 | 54.490230339288 | 54.35599358785 | 54.31266528282 | 54.29580122770 | 54.178 |

Table 5.11 The natural frequencies for thin plate (CCCC)

| Mode | EFG(Different Nodes) | | | | Analytical Solution |
|---|---|---|---|---|---|
| | 7x7 | 9x9 | 12x12 | 13x13 | |
| 1 | 36.140038214255 | 36.159494482140 | 36.16784727556 | 36.15197952872 | 35.988 |
| 2 | 74.618450698129 | 73.986092976231 | 73.94734673677 | 73.90841692780 | 73.393 |
| 3 | 109.336707286560 | 108.781842711669 | 108.78648203872 | 108.75271815669 | 108.521 |
| 4 | 150.571009515510 | 133.480991100657 | 133.24820332677 | 132.81037266975 | 132.25 |
| 5 | | 166.266996688369 | 165.74083636327 | 165.90843144610 | 165.569 |

Table 5.12 The natural frequencies for thin plate (SSSS)

| Mode | EFG(Different Nodes) | | | | Analytical Solution |
|---|---|---|---|---|---|
| | 7x7 | 9x9 | 12x12 | 13x13 | |
| 1 | 19.750569360225 | 19.750138130429 | 19.75096736009 | 19.75012407532 | 19.739 |
| 2 | 107.203364750700 | 51.042495557439 | 49.56182804727 | 49.51949587999 | 49.348 |
| 3 | | 80.179908944879 | 79.18396061848 | 79.14410658730 | 78.957 |
| 4 | | 100.117005755493 | 99.48956217831 | 99.44426288847 | 98.696 |
| 5 | | 131.716574038813 | 128.94057591875 | 128.86604240841 | 128.369 |

Table 5.13 The natural frequencies for (CSCS) plate

| Mode | EFG(Different Nodes) | | | | Analytical Solution |
|---|---|---|---|---|---|
| | 7x7 | 9x9 | 12x12 | 13x13 | |
| 1 | 31.22615209646727 | 29.081168211969 | 29.07164910821 | 29.06036180980 | 28.944 |
| 2 | 72.60530909152656 | 56.375939575699 | 54.98930236680 | 54.94032042084 | 54.745 |
| 3 | | 69.932062125918 | 69.88817520132 | 69.84408610680 | 69.322 |
| 4 | | 98.076323726579 | 95.06189181168 | 94.97926516530 | 94.576 |
| 5 | | 103.844935402379 | 103.00000486831 | 102.94801976047 | 102.212 |

From Table 5.10 to Table 5.13, conclusion can be drawn (i) the natural frequencies by EFG are reasonably good by comparing with analytical solutions, and (ii) the accuracy improves with the number of nodes.

Figure 5.13 to Figure 5.18 are the mode shapes of cantilever plate and Figure 5.19 to Figure 5.24 are the mode shapes of fully clamped plate. Small circles denote the nodes at the original place. These mode shapes are compared with nodal patterns of cantilever plate and fully clamped plate [19] and the agreement is quite good.

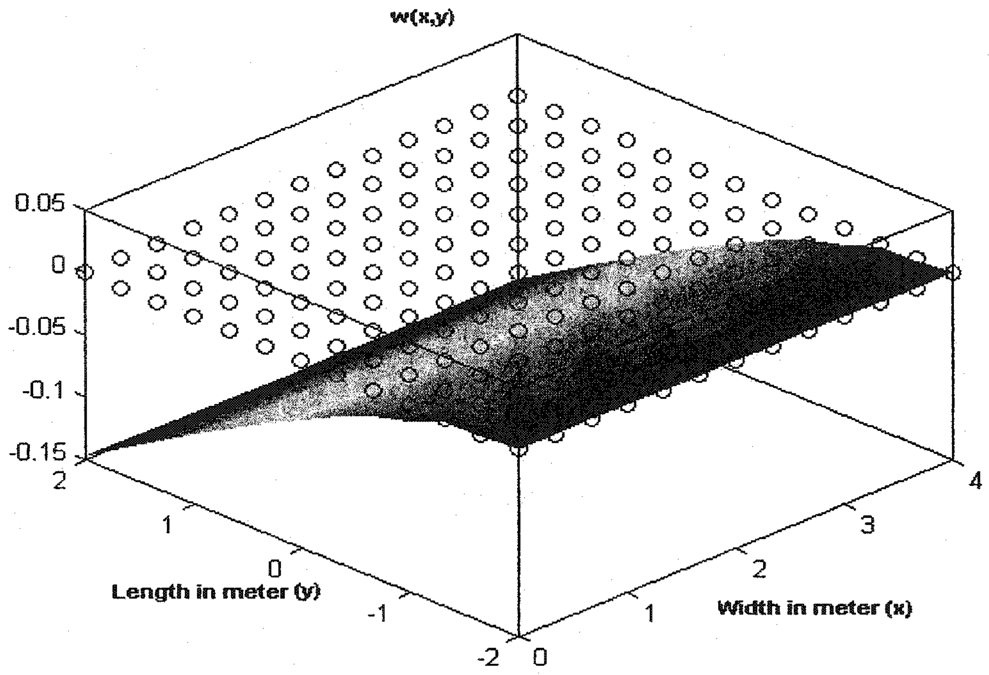

Figure 5.13 The first mode shape for cantilever plate (○ - Undeformed)

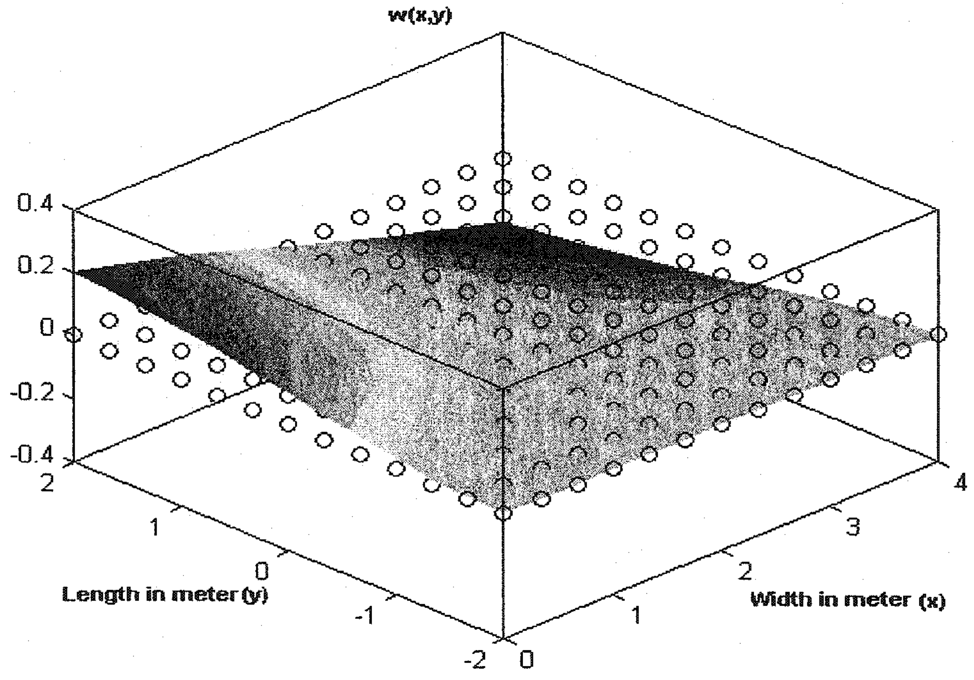

Figure 5.14 The second mode shape for cantilever plate (○ - Undeformed)

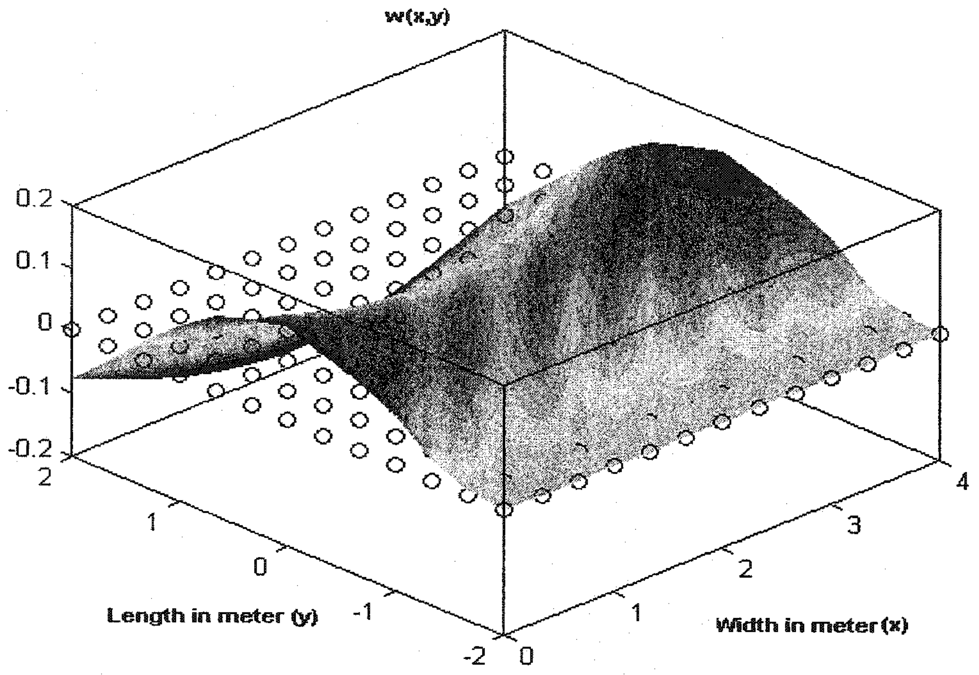

Figure 5.15 The third mode shape for cantilever plate (○ - Undeformed)

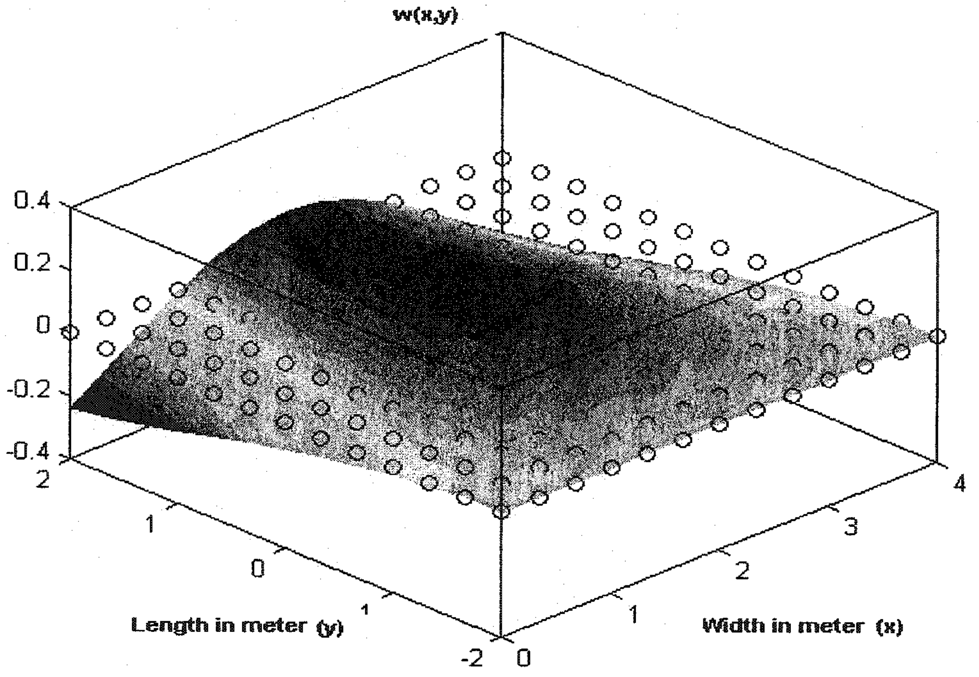

Figure 5.16 The fourth mode shape for cantilever plate (○ - Undeformed)

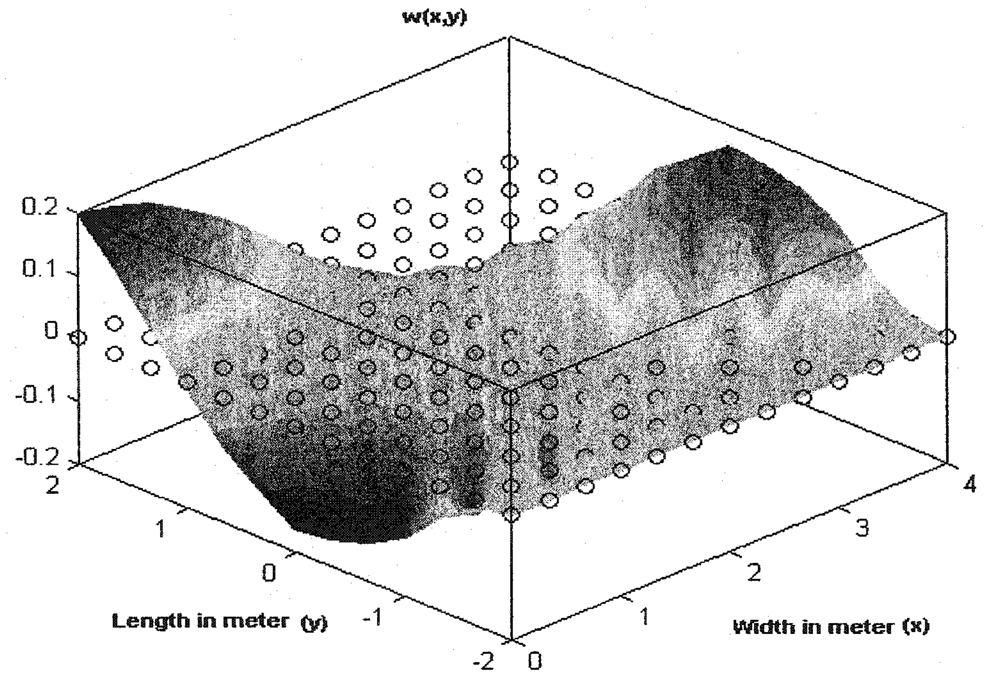

Figure 5.17 The fifth mode shape for cantilever plate (○ - Undeformed)

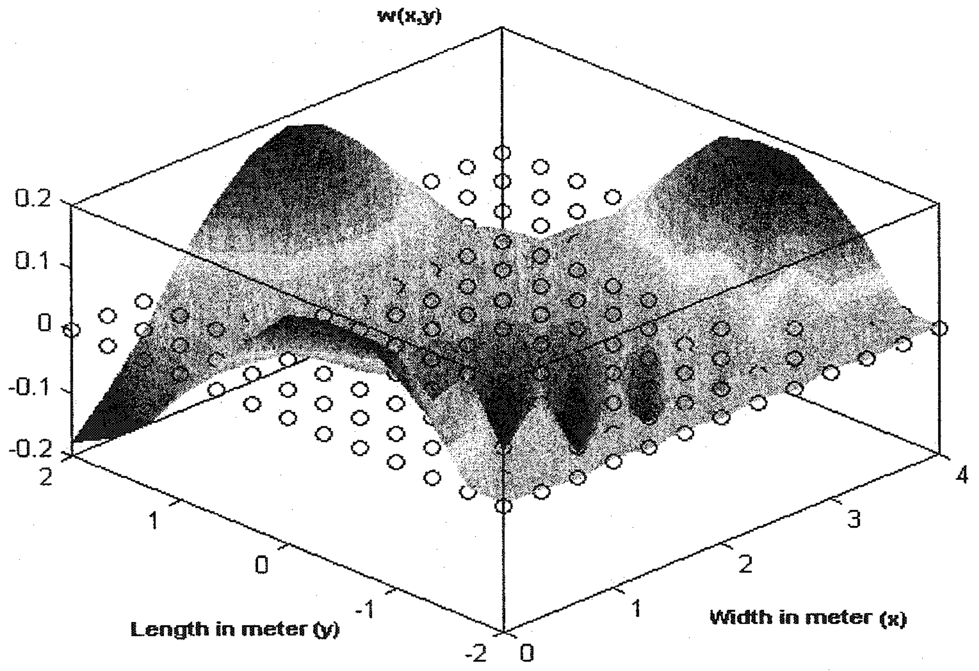

Figure 5.18 The sixth mode shape for cantilever plate (○ - Undeformed)

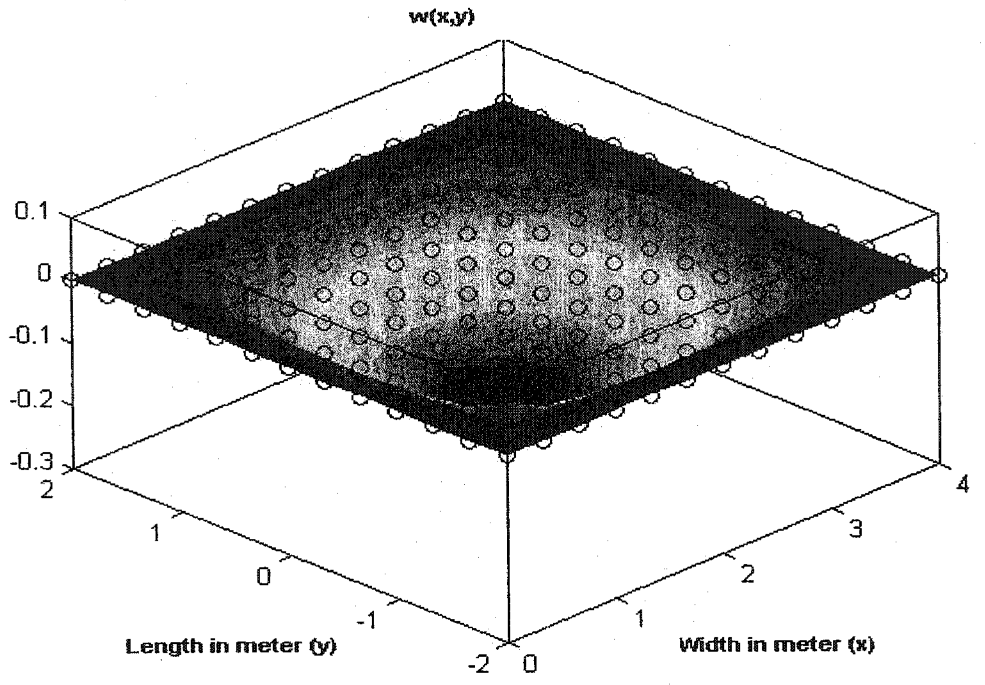

Figure 5.19 The first mode shape for fully clamped plate (○ - Undeformed)

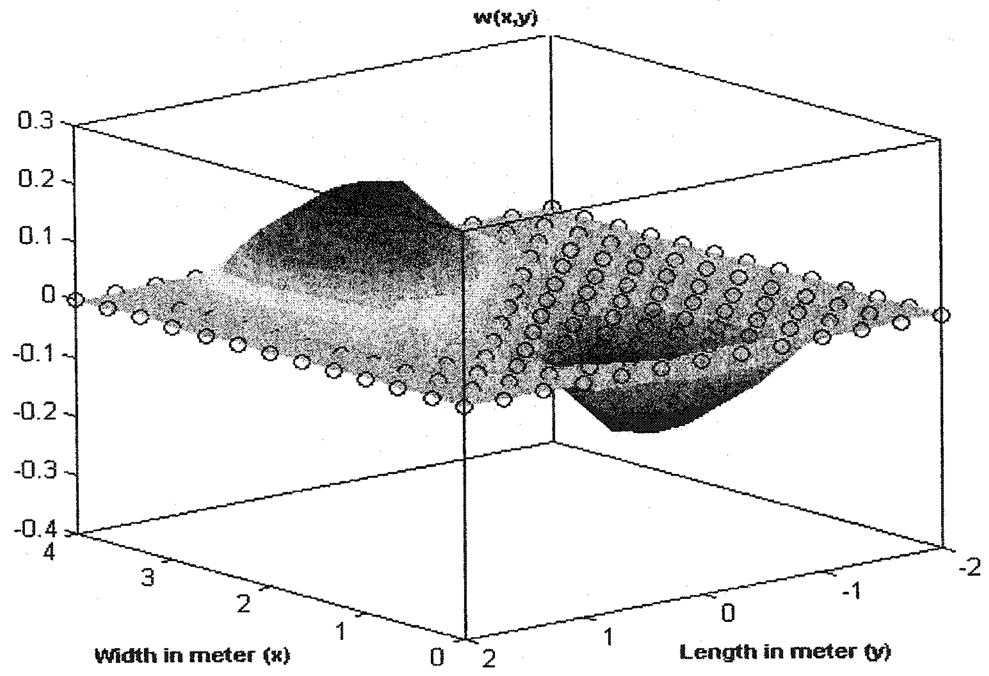

Figure 5.20 The second mode shape for fully clamped plate (○ - Undeformed)

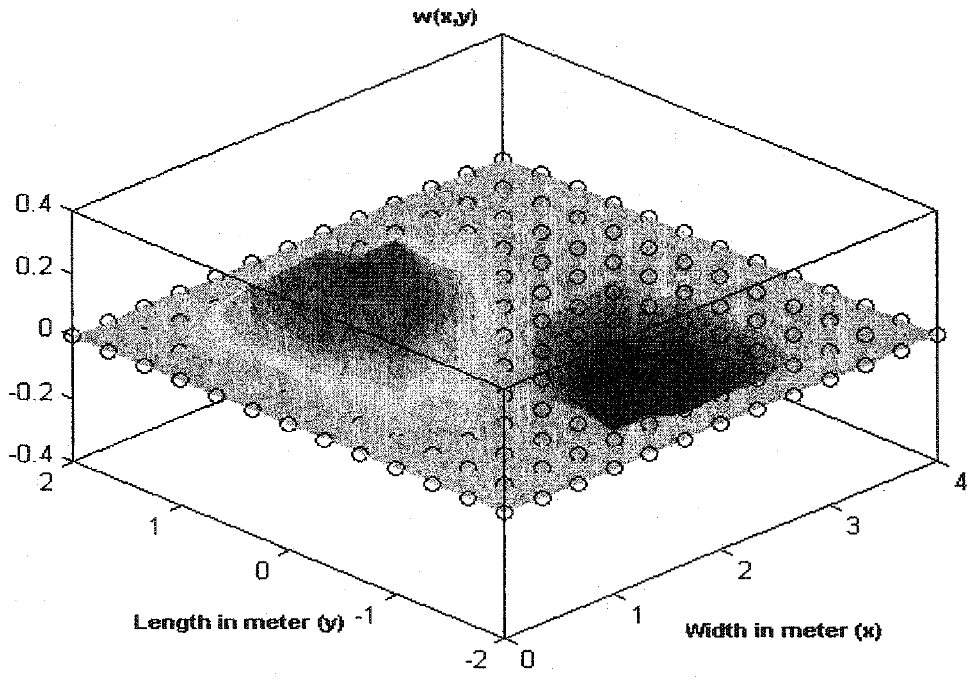

Figure 5.21 The third mode shape for fully clamped plate (○ - Undeformed)

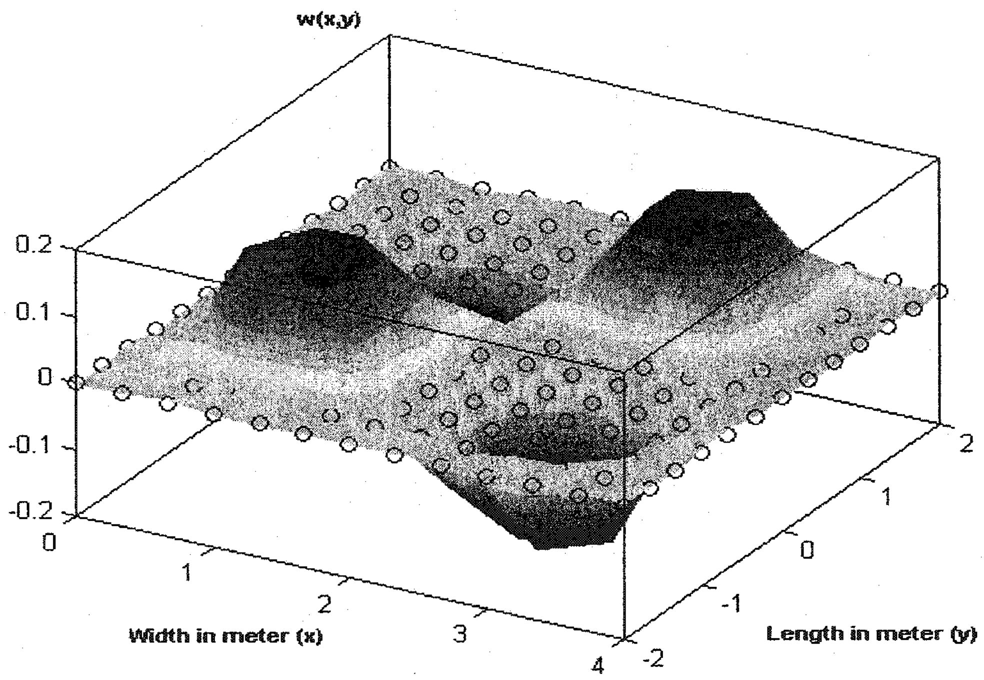

Figure 5.22 The fourth mode shape for fully clamped plate (○ - Undeformed)

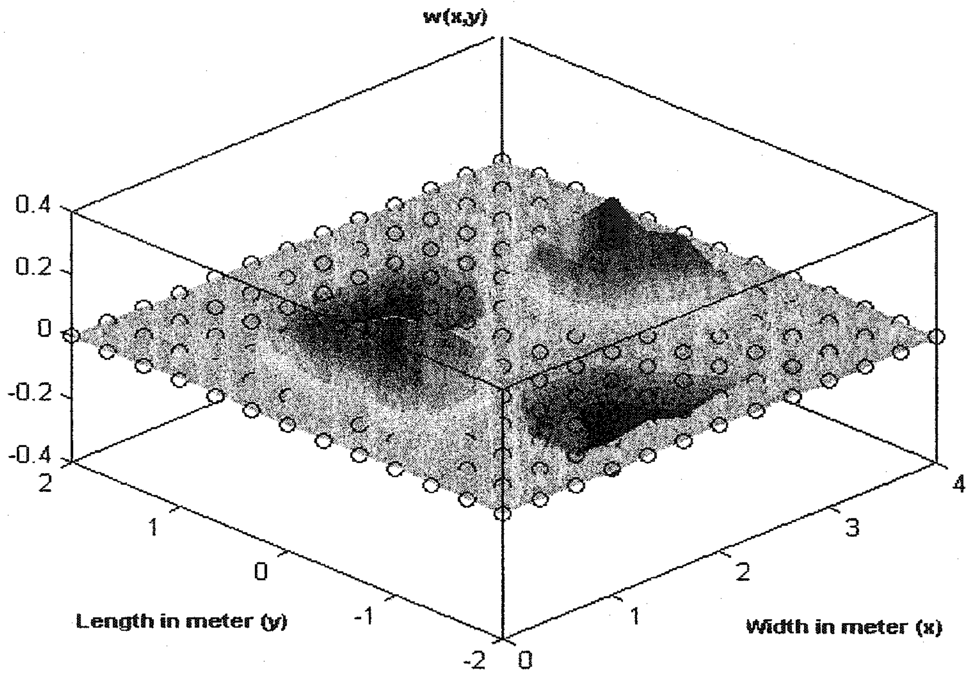

Figure 5.23 The fifth mode shape for fully clamped plate (○ - Undeformed)

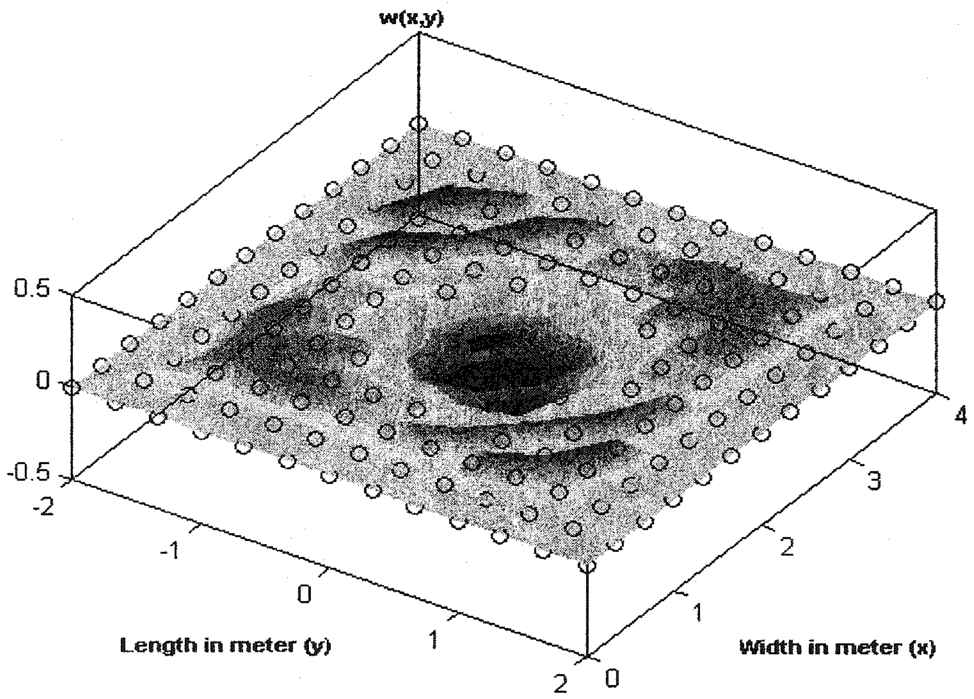

Figure 5.24 The sixth mode shape for fully clamped plate (○ - Undeformed)

## 5.5 Comparison of Natural Frequencies of the Thin Plate by Using Different Type of Support Domains

The natural frequencies of free vibration of a free thin square plate are calculated using different type of support domains. The parameters of a thin square plate are shown in Table 5.14. The non-dimensional natural frequencies are presented in Table 5.15. The first three frequencies corresponding to the rigid displacement are zero [32], and therefore are not listed in Table 5.15.

Table 5.14 The parameters of the panel

| | |
|---|---|
| Young's modulus | $E = 200\text{x}10^9 \, N/m^2$ |
| Mass density | $\rho = 8000 kg/m^3$ |
| Poisson's ratio | $\nu = 0.3$ |
| Length | $a = b = 10$ m |
| Thickness | $t = 0.05$ m |

Table 5.15

Comparison of natural frequencies of thin plate without constraints with other solution

| Mode | Present study ( EFG square domain ) | | | | EFG by G. R. Liu | | | |
|---|---|---|---|---|---|---|---|---|
| | 5x5 | 9x9 | 13x13 | 17x17 | 5x5 | 9x9 | 13x13 | 17x17 |
| 4 | 3.7013 | 3.6706 | 3.6704 | 3.6703 | 3.700 | 3.670 | 3.670 | 3.567 |
| 5 | 4.4678 | 4.4327 | 4.4307 | 4.4296 | 4.468 | 4.434 | 4.430 | 4.429 |
| 6 | 4.9993 | 4.9372 | 4.9334 | 4.9316 | 5.000 | 4.939 | 4.933 | 4.930 |
| 7 | 6.0129 | 5.9068 | 5.9035 | 5.9024 | 6.010 | 5.907 | 5.903 | 5.901 |
| 8 | 6.0129 | 5.9068 | 5.9035 | 5.9024 | 6.010 | 5.907 | 5.903 | 5.901 |
| 9 | 8.1889 | 7.8646 | 7.8462 | 7.8375 | 8.189 | 7.855 | 7.840 | 7.832 |

Table 5.16

Comparison of natural frequencies of thin plate without constraints by using different support domains – square and circle domains

| Mode | EFG ( square domain ) | | | | EFG ( circle domain ) | | | |
|---|---|---|---|---|---|---|---|---|
| | 5x5 | 9x9 | 13x13 | 17x17 | 5x5 | 9x9 | 13x13 | 17x17 |
| 4 | 3.7013 | 3.6706 | 3.6704 | 3.6703 | 3.7241 | 3.7132 | 3.7122 | 3.7131 |
| 5 | 4.4678 | 4.4327 | 4.4307 | 4.4296 | 4.5746 | 4.5584 | 4.5535 | 4.5493 |
| 6 | 4.9993 | 4.9372 | 4.9334 | 4.9316 | 5.1161 | 5.0668 | 5.0981 | 5.1107 |
| 7 | 6.0129 | 5.9068 | 5.9035 | 5.9024 | 6.2612 | 6.2098 | 6.2573 | 6.2981 |
| 8 | 6.0129 | 5.9068 | 5.9035 | 5.9024 | 6.2612 | 6.2098 | 6.2573 | 6.2981 |
| 9 | 8.1889 | 7.8646 | 7.8462 | 7.8375 | 9.2957 | 8.5443 | 8.5750 | 8.6304 |

From Table 5.15, the results of the present study have a good agreement with those by G. R. Liu [32]. From Table 5.16, it can be concluded that the square support domain has better accuracy than that of circle domain for rectangle or square thin plate problems.

## 5.6 Comparison of Natural Frequencies of Cantilever Beam by Different Weight Functions

The parameters of a cantilever beam are same as those in Table 5.3. The beam is represented by 101 nodes. Each cell has one Gauss point and 100 cells are used for integration. Shape functions are constructed by using four different types of weight functions. The results are present in Table 5.17.

Table 5.17

Comparison of natural frequencies of cantilever beam by using different weight functions

| Mode | Gaussian | Cubic Spline | Cubic Polynomial | Quartic Polynomial | Exact Solution |
|---|---|---|---|---|---|
| 1 | 3.5488758175 | 3.517057712 | 3.556126111 | 3.5297963387 | 3.516 |
| 2 | 22.2333195156 | 22.029824631 | 22.365636314 | 22.1031918346 | 22.035 |
| 3 | 62.2190107092 | 61.631311291 | 62.957563849 | 61.8309048709 | 61.695 |
| 4 | 121.8215210823 | 120.616844033 | 124.290318633 | 1210012639959 | 120.903 |
| 5 | 201.1556484889 | 199.052089196 | 207.382865384 | 1996814405171 | 199.859 |
| 6 | 300.0765711862 | 296.730459349 | 313.251765934 | 2976689201932 | 298.556 |

From Table 5.17, one can observe that weight functions affect the accuracy of the results. Among these weight functions, for this problem, the cubic spline weight function gives the best results by comparing with the exact solutions.

## 5.7 Comparison of Natural Frequencies of Cantilever Beam by Different Techniques

Table 5.18 Natural Frequencies of Cantilever Beam by Different Techniques

| | Rayleigh-Ritz nterm=3 | Galerkin nterm=5 | Collocation n=5 | EFG n=5 | FEM n=5 | Analytical |
|---|---|---|---|---|---|---|
| Mode1 | 3.5171 | 3.5160 | 3.48625954842374 | 2.926260 | 3.516130 | 3.516 |
| Mode2 | 22.2334 | 22.0354 | 21.42462970395474 | 26.75467 | 22.06016 | 22.039 |
| Mode3 | 118.1444 | 66.2562 | 66.98915445796999 | 190.9766 | 62.17489 | 61.695 |

nterm- the number of polynomial terms. n- the number of nodes.

From Table 5.18, one can observe that the Rayleigh-Ritz method, Collocation method, Galerkin method (only for cantilever beam), and Finite Element Method (FEM) gives very good results while Element Free Galerkin method (EFG) gives poor results. To overcome this disadvantage, more nodes are needed in EFG to increase the accuracy. The improvement can be found in the Table 5.4 and Table 5.5. When the number of nodes is increased from 11 to 101, the natural frequencies approach the exact solution.

## 5.8 Comparison of Natural Frequencies of Fully Clamped Plate by Different Techniques

Table 5.19 Natural Frequencies of Fully Clamped Plate by Different Techniques

| | Rayleigh-Ritz<br>Orthogonal polynomials<br>mtern=ntern=5 | Collocation<br>nx=ny=5<br>total nodes 49 | EFG<br>nx=ny=7<br>total nodes 49 | Analytical<br>solution |
|---|---|---|---|---|
| Mode 1 | 35.9855 | 36.0043 | 36.1400 | 35.988 |
| Mode2 | 73.4121 | 74.3339 | 74.6185 | 73.393 |
| Mode 3 | 73.4121 | 74.3339 | 74.6185 | 73.393 |
| Mode 4 | 108.2574 | 109.8099 | 109.3367 | 108.521 |

Table 5.19 shows that each method gives very accurate results by comparing it with analytical solutions.

## 5.9 System Matrices by Collocation Method

The following matrices are obtained by collocation method. The advantage of collocation method is that the evaluation of terms $k_{ij}$ and $m_{ij}$ do not involve integration. The method is simple, accurate, and uses less CPU time compared with other methods. The shortcoming of collocation method is that $K^{(n)}$ and $M^{(n)}$ are non-symmetric matrices. The Matlab code by Collocation method for cantilever beam free vibration is given in Appendix 7.

The stiffness and mass matrices are given below:

```
k(Collocation)=

1.0e+002 *

  0.24000000000000  0.24000000000000  0.14400000000000  0.06720000000000

  0.24000000000000  0.48000000000000  0.57600000000000  0.53760000000000

  0.24000000000000  0.72000000000000  1.29600000000000  1.81440000000000

  0.24000000000000  0.96000000000000  2.30400000000000  4.30080000000000


m(Collocation) =

  0.20960000000000  0.72032000000000  1.64006400000000  3.08001280000000

  0.72960000000000  2.57024000000000  5.92409600000000 11.20163840000000

  1.42560000000000  5.11776000000000 11.92665600000000 22.70799360000000

  2.0             8.00768000000000 18.82214400000000 36.04971520000000
```

## 5.10 Comparison of System Matrices by EFG and FEM Method

In EFG, each node of the cantilever beam has one degree of freedom (deflection) since slope can be expressed in terms of deflection. For the sake of this discussion, five nodes are considered on the beam, therefore, the dimension of the global stiffness matrix and the global mass matrix is $5\times5$ before the constraints are taken into account. The following printed matrices can be obtained by running Matlab code (Appendix 2).

```
k(EFG) =

  1.0e+008 *

  Columns 1 through 5

   0.11201493826796  -0.22533829306479   0.11463177132570  -0.00130841652887    0
  -0.22533829306479   0.56791025224505  -0.46111404182475   0.11985049917355   -0.00130841652904
   0.11463177132570  -0.46111404182475   0.80755149752301  -0.69028795487413    0.22921872785075
  -0.00130841652887   0.11985049917355  -0.69028795487413   1.02625807834223   -0.45451220611393
         0           -0.00130841652904   0.22921872785075  -0.45451220611393    0.22660189479280


m(EFG) =

  Columns 1 through 5

   0.28183593750000   0.47988281250000  -0.15996093750001   0.00761718750000    0
   0.47988281250000   1.72148437500000   0.36562500000002  -0.13710937499999    0.00761718750000
  -0.15996093750001   0.36562500000002   1.29492187499991   0.36562500000065   -0.15996093749972
   0.00761718750000  -0.13710937499999   0.36562500000065   1.72148437499722    0.47988281249889
         0            0.00761718750000  -0.15996093749972   0.47988281249889    0.28183593749957
```

One can observe that the global stiffness matrix k(EFG) and the global mass matrix m(EFG) are not banded but symmetric matrices. Nodal matrix $k_{ij}$ contains the stiffness coefficients between node i and node j evaluated at a point (usually quadrature point) in the problem domain. It is a function of coordinates, and needs to be integrated over the entire problem domain. It has to be assembled to the global matrix as long as the support domains of nodes i and j include the quadrature point. If nodes i and j are far apart and if they do not share any quadrature point determined by weight function, $k_{ij}$ vanishes. Therefore, sometimes if the support domain is very compact and does not cover too wide a problem domain, many $k_{ij}$ terms will be zero, and the global system matrix will be sparse and banded.

In FEM, the highest derivative appearing in the energy expressions for a beam is the second derivative. Therefore, it requires the shape functions and their field derivative to be continuous, hence it is necessary to take deflection v and slope $\partial v / \partial x$ as degrees of freedom at each node of beam element. Therefore, if five nodes are used to represent the cantilever beam, the dimension of the global stiffness matrix and the mass matrix is $10 \times 10$ before the constraints are applied. The following printed matrix can be obtained by running Matlab code (Appendix 10).

```
k(FEM) =

  1.0e+008 *

  Columns 1 through 4

  1.34400000000000   0.16800000000000  -1.34400000000000   0.16800000000000
  0.16800000000000   0.02800000000000  -0.16800000000000   0.01400000000000
 -1.34400000000000  -0.16800000000000   2.68800000000000   0
  0.16800000000000   0.01400000000000          0           0.05600000000000
  0                  0                 -1.34400000000000  -0.16800000000000
  0                  0                  0.16800000000000   0.01400000000000
  0                  0                  0                  0
  0                  0                  0                  0
  0                  0                  0                  0
  0                  0                  0                  0

  Columns 5 through 8

  0                  0                  0                  0
  0                  0                  0                  0
 -1.34400000000000   0.16800000000000   0                  0
 -0.16800000000000   0.01400000000000   0                  0
```

```
2.68800000000000    0                  -1.34400000000000   0.16800000000000

0                   0.05600000000000   -0.16800000000000   0.01400000000000

-1.34400000000000   -0.16800000000000   2.68800000000000   0

0.16800000000000    0.01400000000000    0                  0.05600000000000

0                   0                  -1.34400000000000   -0.16800000000000

0                   0                   0.16800000000000   0.01400000000000

Columns 9 through 10

0                   0

0                   0

0                   0

0                   0

0                   0

0                   0

-1.34400000000000   0.16800000000000

-0.16800000000000   0.01400000000000

1.34400000000000    -0.16800000000000

-0.16800000000000   0.02800000000000

m(FEM) =

Columns 1 through 4

0.72428571428571    0.02553571428571    0.25071428571429    -0.01508928571429

0.02553571428571    0.00116071428571    0.01508928571429    -0.00087053571429

0.25071428571429    0.01508928571429    1.44857142857143    0

-0.01508928571429   -0.00087053571429   0                   0.00232142857143

0                   0                   0.25071428571429    0.01508928571429

0                   0                   -0.01508928571429   -0.00087053571429

0                   0                   0                   0

0                   0                   0                   0

0                   0                   0                   0

0                   0                   0                   0

Columns 5 through 8

0                   0                   0                   0

0                   0                   0                   0

0.25071428571429    -0.01508928571429   0                   0

0.01508928571429    -0.00087053571429   0                   0

1.44857142857143    0                   0.25071428571429    -0.01508928571429
```

```
 0                   0.00232142857143    0.01508928571429   -0.00087053571429
 0.25071428571429    0.01508928571429    1.44857142857143    0
-0.01508928571429   -0.00087053571429    0                   0.00232142857143
 0                   0                   0.25071428571429    0.01508928571429
 0                   0                  -0.01508928571429   -0.00087053571429

Columns 9 through 10

0                   0
0                   0
0                   0
0                   0
0                   0
0                   0
0.25071428571429    -0.01508928571429
0.01508928571429    -0.00087053571429
0.72428571428571    -0.02553571428571
-0.02553571428571    0.00116071428571
```

From the printed result k(FEM) and m(FEM), it can be noted that they are banded, sparse, and symmetric matrices. This is a character of FEM system matrix. Because the dimension of the system matrices of FEM are larger than those obtained by EFG, terms related to deflection are selected from k(FEM) and m(FEM), and then k(select) and m(select) are compared with k(EFG) and m(EFG), respectively. One can find that the terms in k(EFG) and the terms in k(select) are different, m(EFG) and m(select) are also different. But both methods can achieve valid natural frequencies.

Select terms which are related to deflection from stiffness matrix k(FEM)

```
k(select)=

1.0e+008 *

 Columns 1 through 5

  1.34400000000000    -1.34400000000000    0                   0                   0
 -1.34400000000000     2.68800000000000   -1.34400000000000    0                   0
```

```
0                  -1.34400000000000   2.68800000000000   -1.34400000000000   0
0                  0                   -1.34400000000000   2.68800000000000   -1.34400000000000
0                  0                   0                   -1.34400000000000   1.34400000000000
```

Select terms which related deflection from mass matrix m(FEM)

```
m(select) =

 Columns 1 through 4

  0.72428571428571   0.25071428571429   0                  0
  0.25071428571429   1.44857142857143   0.25071428571429   0
  0                  0.25071428571429   1.44857142857143   0.25071428571429
  0                  0                  0.25071428571429   1.44857142857143
  0                  0                  0                  0.25071428571429

 Column 5

  0
  0
  0
  0.25071428571429
  0.72428571428571
```

From above comparison, the dimension of system matrices of cantilever beam by EFG and FEM methods are not same. It is better for us to compare system matrices with same size. Clamped-free rod can meet our requirement. For both methods EFG and FEM, the system matrices of five nodes clamped-free rod are all five by five matrices.

```
k (EFG) =

  1.0e+008 *

  Columns 1 through 4

   6.52180371541001  -4.79814935628586  -1.59240435912415  -0.13125000000000
  -4.79814935628586   8.06788256350339  -1.82598320721754  -1.31250000000000
  -1.59240435912415  -1.82598320721754   6.83677513268333  -1.82598320721752
```

```
-0.13125000000000  -1.31250000000000  -1.82598320721752   8.06788256350356

          0         -0.13125000000000  -1.59240435912414  -4.79814935628587

Column 5

          0

-0.13125000000000

-1.59240435912414

-4.79814935628587

 6.52180371540988


m (EFG)  =

 Columns 1 through 4

  0.52378959396258  0.48831180909864  0.03749867134354  0.00084635416667

  0.48831180909864  0.86891209608844  0.48335459183673  0.03893229166667

  0.03749867134354  0.48335459183673  0.89668633078231  0.48335459183673

  0.00084635416667  0.03893229166667  0.48335459183673  0.86891209608844

          0         0.00084635416667  0.03749867134354  0.48831180909864

 Column 5

          0

  0.00084635416667

  0.03749867134354

  0.48831180909864

  0.52378959396259

k (FEM) =

1.0e+009 *

 Columns 1 through 4

  0.84000000000000  -0.84000000000000    0                  0

 -0.84000000000000   1.68000000000000   -0.84000000000000   0

          0         -0.84000000000000    1.68000000000000  -0.84000000000000

          0          0                  -0.84000000000000   1.68000000000000

          0          0                   0                 -0.84000000000000

 Column 5

          0

          0

          0

 -0.84000000000000
 0.84000000000000

m (FEM) =

Columns 1 through 4

 0.65000000000000  0.32500000000000  0                 0

 0.32500000000000  1.30000000000000  0.32500000000000  0

         0         0.32500000000000  1.30000000000000  0.32500000000000

         0         0                 0.32500000000000  1.30000000000000

         0         0                 0                 0.32500000000000

 Column 5

         0

         0

         0

 0.32500000000000

 0.65000000000000
```

The elements in k(EFG) and k(FEM); m(EFG) and m(FEM) matrices of clamped-free rod are not same but at the same figure level. Both methods approach correct natural frequencies. Exact solutions can be found in Maurice Petyt [36].

Table 5.20 Natural Frequencies of Clamped-Free Rod by Different Techniques

| | EFG | FEM | Exact Solution |
|---|---|---|---|
| Mode 1 | 1.573308 | 1.580908 | 1.571 |
| Mode 2 | 4.837105 | 4.987196 | 4.712 |

In this chapter, Element Free Galerkin technique is employed to calculate the natural frequencies for truss, beam, in-plane plate, thin plate free vibration. Very good results have been achieved comparing with those by analytical method and published books and papers. Results from different techniques have been compared with each other.

The disadvantage of the classical methods is that they work directly with strong form equations. Hence, these methods are not practical for complex engineering problems. In the following chapter, the Element free Galerkin method and modal superposition method will be used to compute the sound transmission loss through panel.

# Chapter 6

# Sound Transmission Through the Panel by Element Free Galerkin (EFG) Techniques

## 6.1 Introduction

Previous chapters have described the detailed formulation and procedures of Element Free Galerkin method. Natural frequencies of rod, beam, in-plane and thin plate in bending have been presented in chapter 5. In this chapter, the natural frequencies and mode shapes of thin plate which we obtained from chapter 5 will be used to calculate the response and sound transmission loss when sound strikes a panel barrier.

In many practical situations, acoustic waves strike a structural panel and some of the acoustic energy is transmitted through the panel to another acoustic domain. Sound transmission loss can be found either by experiments or by analytical or finite element techniques. In this chapter, the sound transmission loss of panels by element free Galerkin technique are presented.. It is very challenging and interesting since little work has been done related to acoustics by EFG. The results show that this new modeling and simulation technique is very attractive.

When normal incidence sound hits a panel, the sound pressure waves cause the panel to vibrate. Consider a panel that is acted upon by a harmonic pressure. The pressure is uniformly distributed over one side of panel only. Therefore, sound transmission through a panel can be considered as a thin plate forced vibration problem. Hence, the response of a plate to a forced harmonic vibration will yield information on the sound transmission.

## 6.2 Modal Analysis –Eigenvalue Equation for Thin Plate

In chapter 5, natural frequencies and eigenvectors (mode shapes) have been calculated for a thin plate by EFG method. The mode shapes and the natural frequencies are required to solve more general dynamics problems such as transient response or response to other forms of excitation. The governing equation [39] for forced vibration analysis in its general form is

$$[M]\{\ddot{U}\}+[C]\{\dot{U}\}+[K]\{U\}=\{F\} \tag{6.1}$$

where $[M]$ is the mass matrix of the structure.

$[C]$ is the damping matrix of the structure.

$[K]$ is the stiffness matrix of the structure

$\{U\}$ is the vector of displacements (The first and second derivatives of $U$ with respect to time are the vectors of velocity and acceleration, respectively.)

$\{F\}$ is the vector of applied forces.

The force vector in Equation (6.1) is a function of time in the present study, where F(t) is applied as a harmonic wave at a specific frequency. The natural frequencies and corresponding mode shapes of the structure are obtained by solving the undamped homogeneous form of Equation (6.1)

$$[M]\{\ddot{U}\}+[K]\{U\}=0 \tag{6.2}$$

Since the plate is undergoing a harmonic vibration, the deflection U can be expressed in the form

$$U = W\exp(i\omega t) \tag{6.3}$$

where W is the amplitude of the vibration. Substitution of Equation (6.3) into (6.2) leads to the following equation

$$(K - \omega^2 M)W = 0 \tag{6.4}$$

For a thin plate with $n_t$ nodes, $n_t$ is the total number of nodes which represent the panel. The size of matrices K and M should be $n_t \times n_t$.

## 6.3 The Boundary Constraints- Orthogonal Transform Techniques

Using Singular Value Decomposition [32], constraints matrix H for the thin plate obtained in chapter 4 can be decomposed as follows:

$$H_{2n_t x n_t} = R_{2n_t x 2n_t} \begin{bmatrix} \sum_{rxr} & 0 \\ 0 & 0 \end{bmatrix}_{2n_t x n_t} V^T_{n_t x n_t} \tag{6.5}$$

where R and V are orthogonal matrices, $\sum_{rxr}$ which is diagonal matrix, and r is the rank of H, which is the same as the number of independent constraints.

The orthogonal matrix V can be partitioned as follows:

$$V^T = \{V_{n_t x r},\ V_{n_t \times (n_t - r)}\}^T \tag{6.6}$$

Post multiplying Equation (6.5) by $V_{n_t \times n_t}$

$$HV = H[V_{n_t x r} \quad V_{n_t \times (n_t - r)}] = [HV_{n_t x r} \quad HV_{n_t x (n_t - r)}]$$

$$= R_{2n_t x 2n_t} \begin{bmatrix} \sum_{rxr} & 0 \\ 0 & 0 \end{bmatrix}_{2n_t \times n_t} V^T_{n_t \times n_t} V_{n_t \times n_t} \tag{6.7}$$

Since V is an orthogonal matrix,

$$V^T_{n_t \times n_t} V_{n_t \times n_t} = I \tag{6.8}$$

Equation (6.7) becomes

$$HV = H[V_{n_t \times r} \quad V_{n_t \times (n_t - r)}] = [HV_{n_t \times r} \quad HV_{n_t \times (n_t - r)}]$$

$$= R_{2n_t x 2n_t} \begin{bmatrix} \sum_{rxr} & 0 \\ 0 & 0 \end{bmatrix}_{2n_t x n_t} \tag{6.9}$$

This implies

$$HV_{n_t \times (n_t - r)} = 0 \tag{6.10}$$

Therefore, the following orthogonal matrix transformation satisfies Equation (4.14)

$$W = V_{n_t \times (n_t - r)} \tilde{W} \tag{6.11}$$

Substitution of Equation (6.11) into Equation (6.4) results in

$$(K - \omega^2 M) V_{n_t \times (n_t - r)} \tilde{W} = 0 \tag{6.12}$$

$$V^T_{(n_t - r) \times n_t} (K - \omega^2 M) V_{n_t \times (n_t - r)} \tilde{W} = 0 \tag{6.13}$$

$$(\tilde{K} - \omega^2 \tilde{M}) \tilde{W} = 0 \tag{6.14}$$

Equation (6.14) is the condensed form of eigenvalue equations, where

$$\tilde{K} = V_{(n_t - r) \times n_t}{}^T \, K \, V_{n_t \times (n_t - r)} \tag{6.15}$$

is the condensed stiffness matrix and

$$\tilde{M} = V_{(n_t - r) \times n_t}{}^T \, M \, V_{nx(n-r)} \tag{6.16}$$

is the condensed mass matrix. Solving the standard eigenvalue problem in Equation (6.14) yields the natural frequencies and mode shapes of the free vibration of thin plate.

The dimensions of the two condensed matrices $\tilde{K}$, $\tilde{M}$ is $(n_t - r)\times(n_t - r)$, where r is the number of constraints. Therefore the number of eigenvalues $\omega_i$ is $(n_t - r)$, which correspond to $(n_t - r)$ natural frequencies and $(n_t - r)$ corresponding eigenvectors $\tilde{W}$. The size of $\tilde{W}$ is $(n_t - r)\times(n_t - r)$. The actual eigenvectors W of Equation (6.4) are obtained by the following transformation using Equation (6.11),

$$W = V_{n_t x(n_t - r)} \tilde{W}_{(n_t - r)\times(n_t - r)} \tag{6.17}$$

where W is called modal matrix, the size of which is $n_t \times (n_t - r)$.

## 6.4 Modal Superposition Method

In general, there are two ways to solve forced vibration analysis: mode superposition and direct integration. In the present study, the former approach is adopted for calculation. In a forced harmonic response analysis using modal superposition [40], the first step is to compute the natural frequencies and mode shapes with the structural damping and forces set to zero. The response of each mode is then computed for each time interval of the enforced time history, with the user-defined damping and forces included. Modal superposition is then applied to compute the overall system response.

Each column of W represents a mode shape corresponding to a particular natural frequency. The mode shapes represent relative amplitudes of vibration rather than absolute displacements, and they are orthogonal to each other. Since each mode shape represents an independent motion of the structure, superimposing the independent motions of the individual modes gives the complete motion of the structure.

The eigenvector matrix $W_{n_t \times (n_t - r)}$ has the following form:

$$W = [W_1, W_2, \cdots, W_{(n_t - r)}] \tag{6.18}$$

where $W_1$, $W_2$, ... , $W_{(n_t - r)}$ are column vectors which represent mode shapes for their corresponding natural frequencies. The size of each column vector is $n_t \times 1$.

Assume the solution of Equation (6.1) in the following form:

$$\{U\} = [W]\{q\} \tag{6.19}$$

where $\{q\}$ is generalized displacement vector, and the dimension of $\{q\}$ is $(n_t - r) \times 1$

$$\{q\} = \begin{Bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{Bmatrix}_{(n_t - r) \times 1} \tag{6.20}$$

Further, $\{U\}$ is displacement vector given by

$$\{U\} = \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{n_t} \end{Bmatrix}_{n_t \times 1} \tag{6.21}$$

Substituting Equation (6.19) into Equation (6.1) results in

$$MW\ddot{q} + CW\dot{q} + KWq = F \tag{6.22}$$

Premultiplying Equation (6.22) by $W^T$

$$W^T MW\ddot{q} + W^T CW\dot{q} + W^T KWq = W^T F \tag{6.23}$$

The damping matrix C is taken to be a linear combination of mass matrix M and stiffness matrix K, as Rayleigh damping, in the form

$$C = \alpha M + \beta K \tag{6.24}$$

where $\alpha$ and $\beta$ are scalar variables.

The products $W^T M W$, $W^T C W$, and $W^T K W$, will become diagonal matrices because of the orthogonality relations among the eigenvectors. Equation (6.12) can be written as follows:

$$M_{diag}\ddot{q} + C_{diag}\dot{q} + K_{diag}q = P \tag{6.25}$$

where $M_{diag}$, $C_{diag}$, $K_{diag}$ are diagonal matrices for mass, damping, and stiffness matrix respectively, the dimension being $(n_t - r) \times (n_t - r)$. Equations (6.25) are written in the form

$$\begin{aligned}
&m_1\ddot{q}_1 + c_1\dot{q}_1 + k_1 q_1 = p_1 \\
&m_2\ddot{q}_2 + c_2\dot{q}_2 + k_2 q_2 = p_2 \\
&\vdots \\
&m_{(n_t-r)}\ddot{q}_{(n_t-r)} + c_{(n_t-r)}\dot{q}_{(n_t-r)} + k_{(n_t-r)}q_{(n_t-r)} = p_{(n_t-r)}
\end{aligned} \tag{6.26}$$

The above equations are a system of decoupled equations. The solution of each of these simple ordinary differential equations can be found easily and the solution to the original problem can be recovered by Equation (6.8).

## 6.5 Solution for Forced Vibration Response of Single –DOF System

In the above section, Equation (6.1) has been decoupled to form a system of single degree of freedom equation (Equation (6.26)). The next step is to solve these single-DOF equations one by one. Each single degree of freedom system motion equation in Equation (6.26) has the following form

$$m\ddot{x} + c\dot{x} + kx = F_0 \cos\omega t \tag{6.27}$$

The solution of this equation has two parts. For F(t) = 0, the homogeneous differential equation whose solution corresponds physically to that of free-damped vibration is obtained. With $F(t) \neq 0$, the particular solution that is due to the excitation in addition to the homogeneous solution can be determined [41].

Any damping can be expressed in terms of the critical damping by a non-dimensional number $\zeta$, called damping ratio.

$$\zeta = \frac{c}{c_r} \tag{6.28}$$

where $c_r$ is the critical damping given by

$$c_r = 2m\sqrt{\frac{k}{m}} = 2m\omega_n = 2\sqrt{km} \tag{6.29}$$

If $\zeta<1$ (underdamped case), the homogeneous solution is:

$$\begin{gathered} x_h = e^{-\zeta\omega_n t}(c_1 \sin\omega_d t + c_2 \cos\omega_d t) \\ \omega_d = \omega_n\sqrt{1-\zeta^2} \end{gathered} \tag{6.30}$$

The constants $c_1$, $c_2$ are determined from initial conditions $x(0)$ and $\dot{x}(0)$ as

$$x_h = e^{-\zeta\omega_n t}\left(\frac{\dot{x}(0) + \zeta\omega_n x(0)}{\omega_d}\sin\omega_d t + x(0)\cos\omega_d t\right) \tag{6.31}$$

The particular solution to the preceding equation is a steady-state oscillation of the same frequency $\omega$ as that of the excitation and has the form

$$x_p = X\cos(\omega t - \phi) \tag{6.32}$$

where X is the amplitude of oscillation and $\phi$ is the phase of the displacement with respect to the exciting force.

$$X = \frac{F_0}{\sqrt{(k - m\omega^2)^2 + (c\omega)^2}} \tag{6.33}$$

$$\phi = \tan^{-1}\frac{c\omega}{k - m\omega^2} \tag{6.34}$$

X can be rewritten:

$$X = \frac{F_0 / k}{\sqrt{\left[1 - (\frac{\omega}{\omega_n})^2\right]^2 + \left[2\zeta(\frac{\omega}{\omega_n})\right]^2}} \tag{6.35}$$

The solution of Equation (6.26) is the combination of the homogeneous solution and particular solution given by.

$$x = x_h + x_p \tag{6.36}$$

## 6.6 The Flowchart of Modal Superposition Method

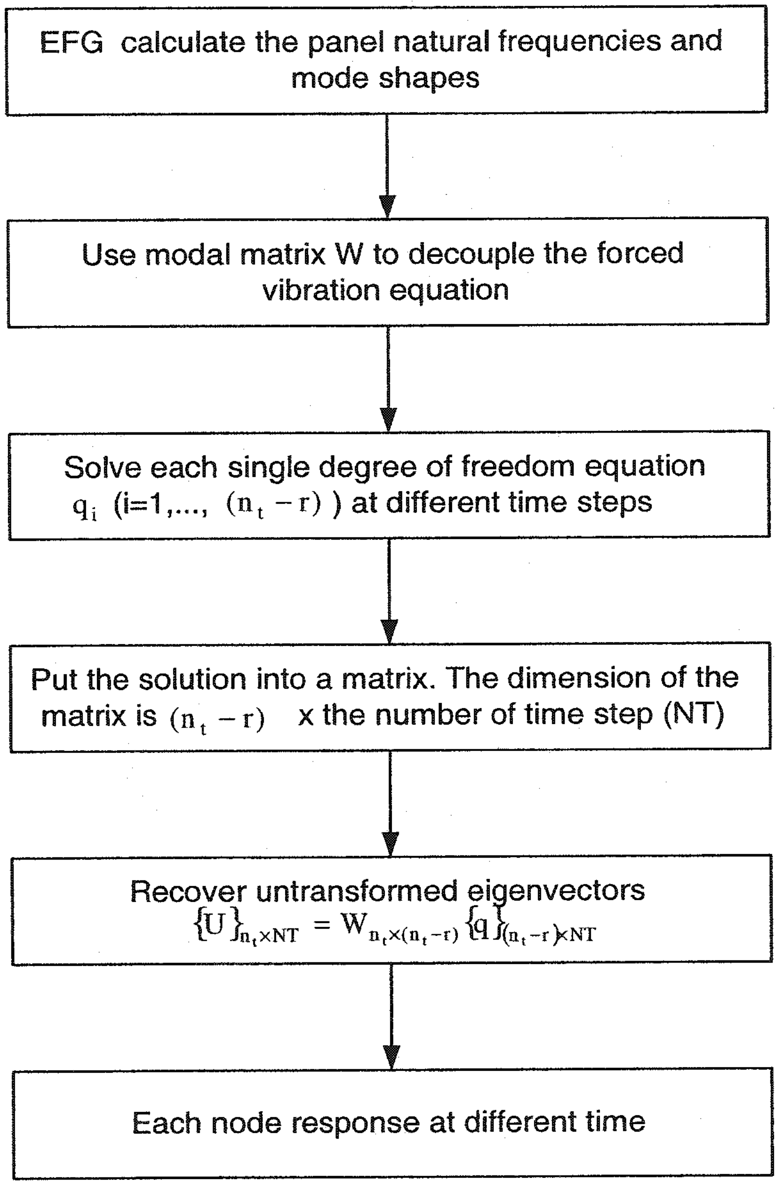
EFG calculate the panel natural frequencies and mode shapes
Use modal matrix W to decouple the forced vibration equation
Solve each single degree of freedom equation $q_i$ (i=1,..., $(n_t - r)$ ) at different time steps
Put the solution into a matrix. The dimension of the matrix is $(n_t - r)$ x the number of time step (NT)
Recover untransformed eigenvectors $\{U\}_{n_t \times NT} = W_{n_t \times (n_t - r)} \{q\}_{(n_t - r) \times NT}$
Each node response at different time

Figure 6.1 The Flow of Modal Superposition Method

## 6.7 The calculation of Sound Transmission Loss Through Panel

Sound Transmission Loss is a measure of the ability of a material to block sound from passing through it. Mathematically it is defined as the ratio of the sound energy incident on the panel to the sound energy transmitted through a panel [42] [43].

$$TL = 10\log\left(\frac{I_i}{I_t}\right) \tag{6.37}$$

where TL represents sound transmission loss. The unit of TL is dB. $I_i$ is the incident intensity and $I_t$ is the transmitted intensity. Since intensity and sound pressure are related by equation

$$I = \frac{p^2}{\rho c} \tag{6.38}$$

where $\rho$ is the density of medium, and c is the velocity of sound in the medium. The product $\rho c$ is termed the characteristic acoustic impedance of the medium. p is the root-mean-square pressure. Substitution of Equation (6.38) into Equation (6.37) gives

$$TL = 20\log\left(\frac{p_i}{p_t}\right) \tag{6.39}$$

where $p_i$ is the incident pressure and $p_t$ is the transmitted pressure.

When sound wave acts on the panel, the incident pressure $p_i$ is known. The panel sets the air on the other side into oscillations. The velocity of air particles on the other side of the panel will be equal to the panel velocity. The particle velocity and pressure are related as

$$u = \frac{p}{\rho c} \tag{6.40}$$

Hence the pressure is given by

$$p = \rho c u_{\text{mole}} \tag{6.41}$$

where $\rho$ is air density $1.21\ \text{kg/m}^3$, c is sound velocity 343 m/s. The displacement and velocity of the panel will be the same as molecular velocity of the air, and hence $u_{\text{mole}}$ can be calculated by the first derivatives displacement respect to time. The displacement has been found in previous section by EFG and modal superposition techniques.

## 6.8 Example – Sound Transmission Through the Clamped Panel

A fully clamped panel is excited by normal incidence sound pressure. Consider the panel which is acted upon by the harmonic pressure, $p = p_{\max}\cos\omega t$, where $p_{\max}$ is the amplitude of the pressure. The parameters of the panel are shown in Table 6.1. $7\times7$ nodes are used to model the fully clamped panel.

Table 6.1 The parameters of the panel

| | |
|---|---|
| Young's modulus | $E = 200\text{x}10^9\ \text{N/m}^2$ |
| Mass density | $\rho = 8000\text{kg/m}^3$ |
| Poisson's ratio | $\nu = 0.3$ |
| Length | $a = b = 10$ m |
| Thickness | $t = 0.05$ m |

The natural frequencies (rad/s) for fully clamped panel are shown in Table 6.2. The responses of the panel under different kinds of conditions are presented from Figure 6.2 to Figure 6.7

Table 6.2 Natural frequency of fully clamped plate in rad/s.

| Mode | Natural Frequency (rad/s) |
|---|---|
| 1 | 27.325654757253 |
| 2 | 55.701036181278 |
| 3 | 56.065452937563 |
| 4 | 82.542105264401 |
| 5 | 109.80258324775 |
| 6 | 114.75830429609 |
| 7 | 127.07135635406 |
| 8 | 146.43920911744 |
| 9 | 188.82813885533 |

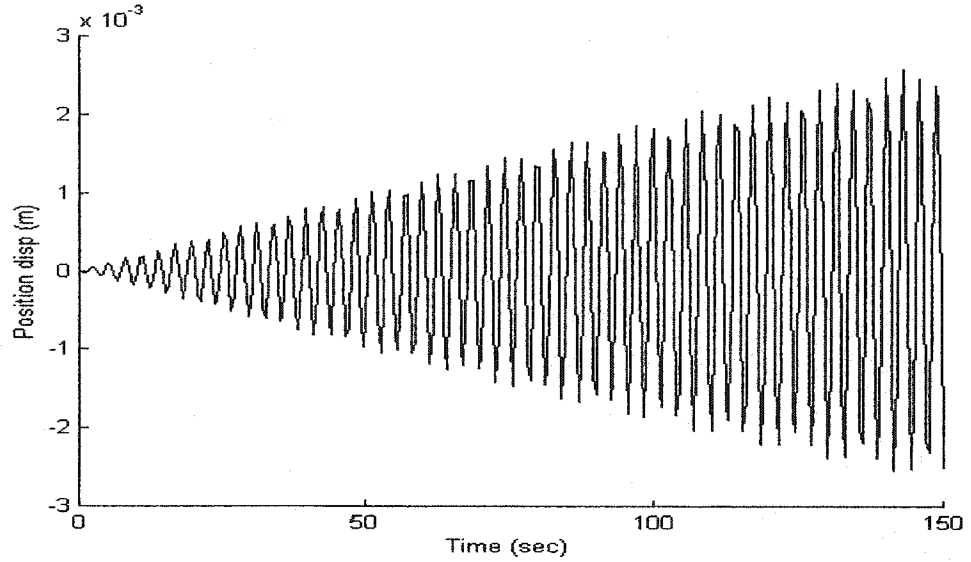

Figure 6.2 The response of node at the centre of the panel under normal incidence cosine sound pressure, no damping, the frequency of excitation is equal to the first natural frequency.

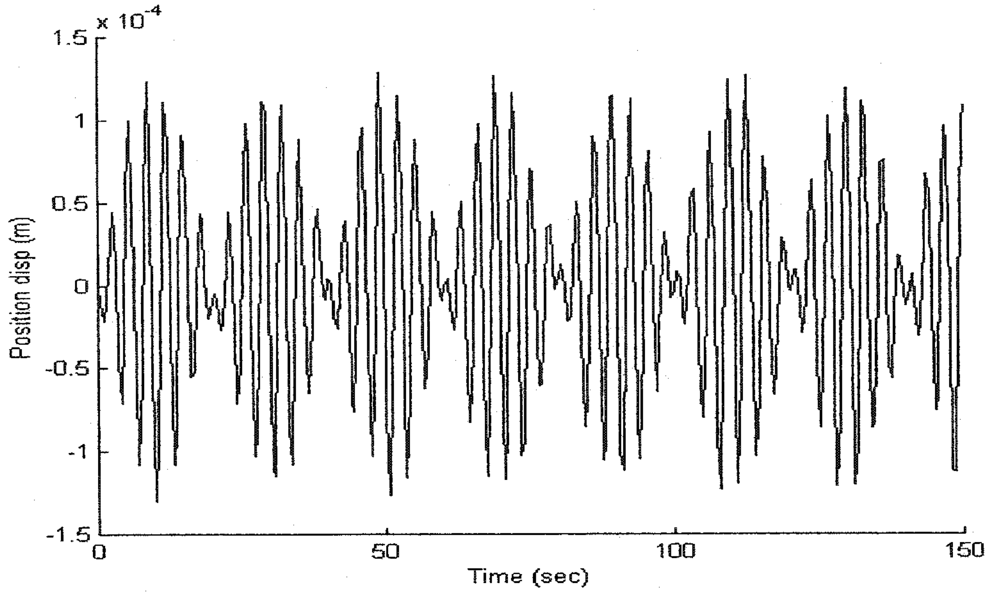

Figure 6.3 The response of node at the centre of the panel under normal incidence cosine sound pressure, no damping, the frequency of excitation is close to the first natural frequency.

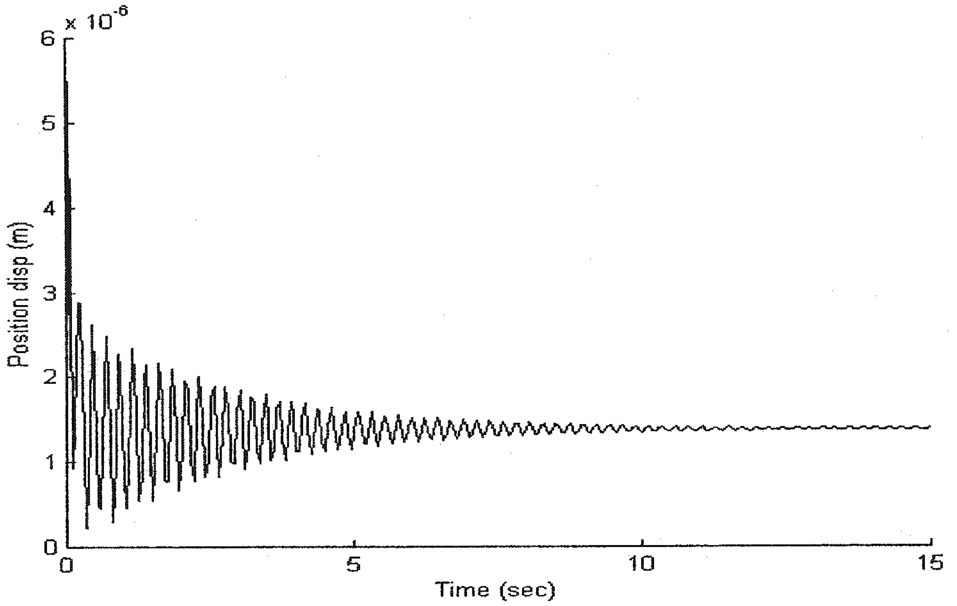

Figure 6.4 The response of node at the centre of the panel under normal incidence cosine sound pressure, damping exists, 'Rayleigh damping' coefficients $\alpha$=0.001, $\beta$=0.001. The frequency of excitation equals zero.

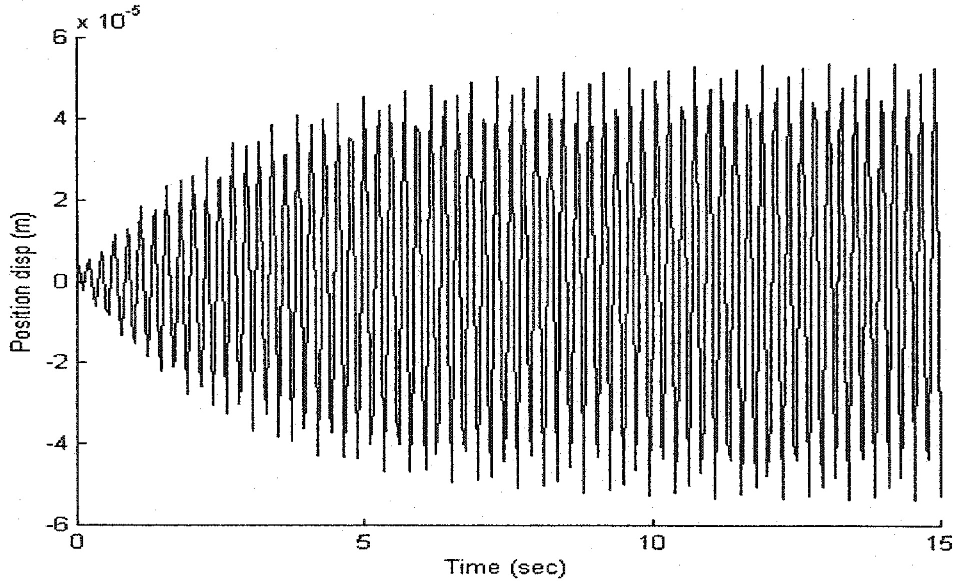

Figure 6.5 The response of node at the centre of the panel under normal incidence cosine sound pressure, damping exists, 'Rayleigh damping' coefficients $\alpha$=0.001. The frequency of excitation equals to the first natural frequency.

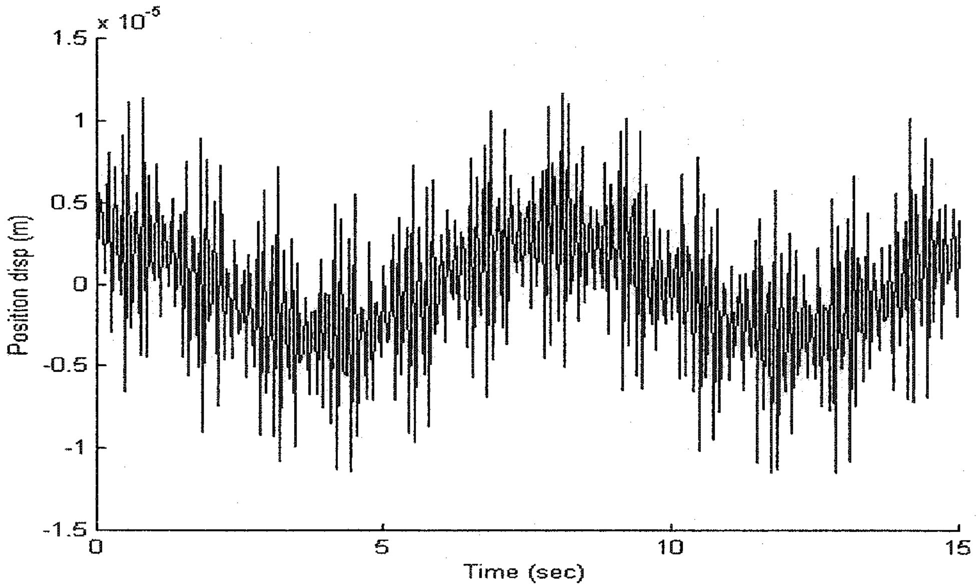

Figure 6.6 The response of node at the centre of the panel under normal incidence cosine sound pressure, no damping, the frequency of excitation is equal to the second natural frequency.

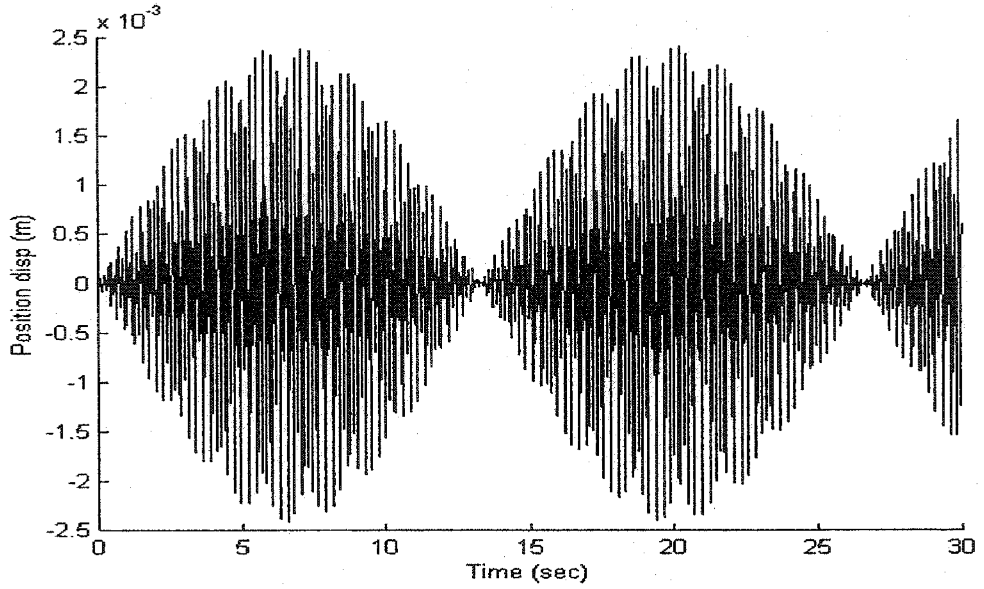

Figure 6.7 The response of node at the centre of the panel under normal incidence cosine sound pressure, no damping, the frequency of excitation is close to the eighth natural frequency.

Figure 6.2 to Figure 6.7 are the responses of node at the centre of the panel under different conditions. In Figure 6.2, the response will increase without bounds when the frequency of excitation equals to the first natural frequency. The vibration approaches the resonance condition. The phenomenon of Figure 6.3 and Figure 6.8 is called "beat", which occurs whenever the frequency of the excitation is close to the natural frequency of the panel. Figure 6.4 states that once the motion is initiated, because the sign of the damping force is always opposite to that of the velocity, the motion will cease. Amplitudes decay with time. In Figure 6.5, although the frequency of excitation equals to the first natural frequency, the response cannot reach to infinity because of damping. Since the second mode does not receive any contribution from the uniform excitation on the panel, the response in Figure 6.6 does not grow with time.

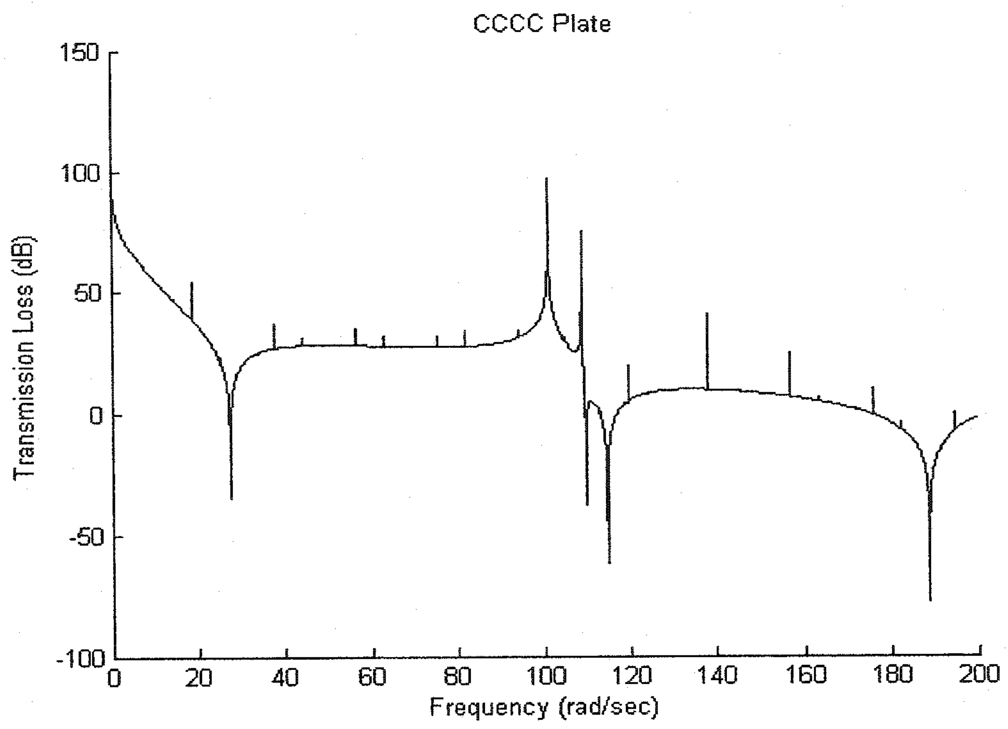

Figure 6.8 Sound transmission loss through clamped panel

From the plot in Fig.6.8, one can observe that sound transmission loss varies with the frequency of excitation. The maximum sound transmission through the panel takes place at the resonance frequencies. The sound reduction between two spaces is dependant on all of the elements of the structure separating them. The sound reduction of a panel is frequency-selective which can be seen from the diagram above. At low frequencies the stiffness of the material is the main controlling factor. At the resonances frequencies major variation in sound transmission occur. At about an octave above the lowest resonance, the mass of the wall takes over and dominates the sound reduction performance. Here the sound transmission loss depends on the surface density of the panel. Above the first natural frequency and except at the higher resonant frequencies, the sound transmission loss is similar to the mass law shown in Figure 6.9.

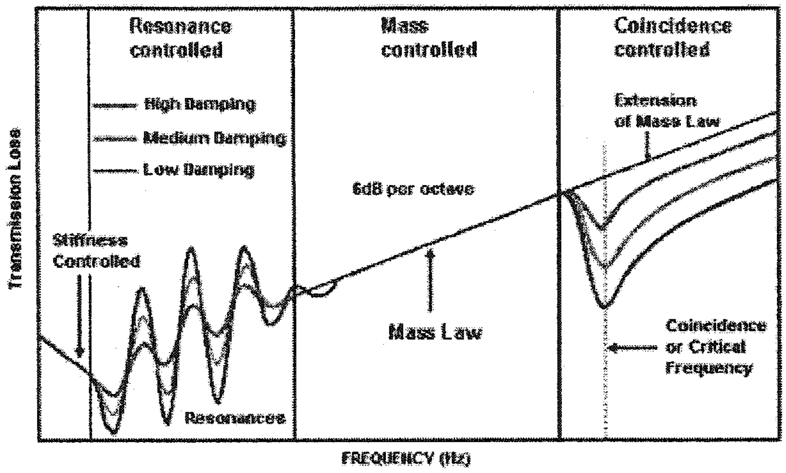

Figure 6.9 General Behaviour of a Panel [44]

## 6.9 Mass Law

Bascially, a panel attenuates the transmission of sound by the inertia of its mass. If the wall is massive, the transmitted wave will be of the same frequency as the incident wave, but the amplitude of the transmitted wave will be severely "damped." An approximate relation for the transmission loss through a panel as a function of its mass can be obtained by equating inertial forces per unit area of the wall to the difference of the sound pressures on its two sides. Let $p_i, p_t$ and $p_r$ be the acoustic pressure of the incident, transmitted, and reflected waves, respectively. m is the mass of the panel per unit area, v is the velocity amplitude of the vibrating wall, $\omega$ is the radian frequency of vibration. A balance of inertial forces gives:

$$p_i + p_r - p_t = im\omega v \qquad (6.42)$$

However, the transmitted energy will always be small, so that

$$p_i \approx \frac{im\omega v}{2} \tag{6.43}$$

On the side of the transmitted sound wave, the velocity of the panel will be the same as molecules of the air, and the pressure of the transmitted wave will be

$$p_t \approx \rho c v \tag{6.44}$$

where $\rho$ and c represent the mass density and speed of sound for air, respectively. The sound reduction or transmission loss, in dB is given by Equation (6.39). Substitution Equation (6.43) and (6.44), we have the equation:

$$TL \approx 20\log_{10}\frac{m\omega}{2\rho c} \tag{6.45}$$

This is the so-called mass law. It states that sound transmission loss increases logarithmically with (1) the mass of the wall per unit area and (2) the frequency of the sound wave. The mass law (Figure 6.10) predicts that the transmission loss will increase by approximately 6 dB by doubling of the surface mass or doubling of the frequency .

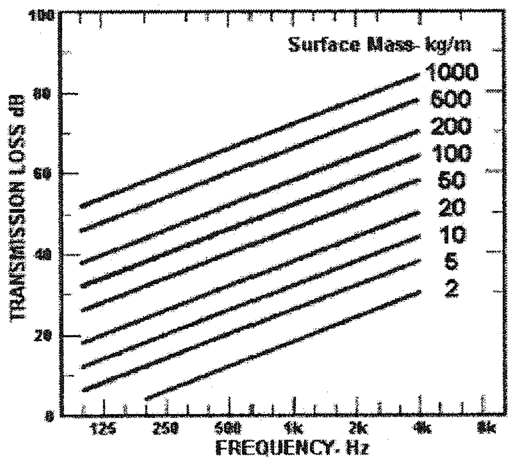

Figure 6.10 Mass Law [44]

## 6.10 Comparison with Mass Law

In figure 6.11, mass density is $8000 \mathrm{kg/m^3}$, thickness of the thinner panel is 0.05m, thickness of the thicker panel is 0.1m. As would be expected, the greater mass of the panel exhibits much higher transmission loss than a lighter panel. The present results are good agreement with mass law.

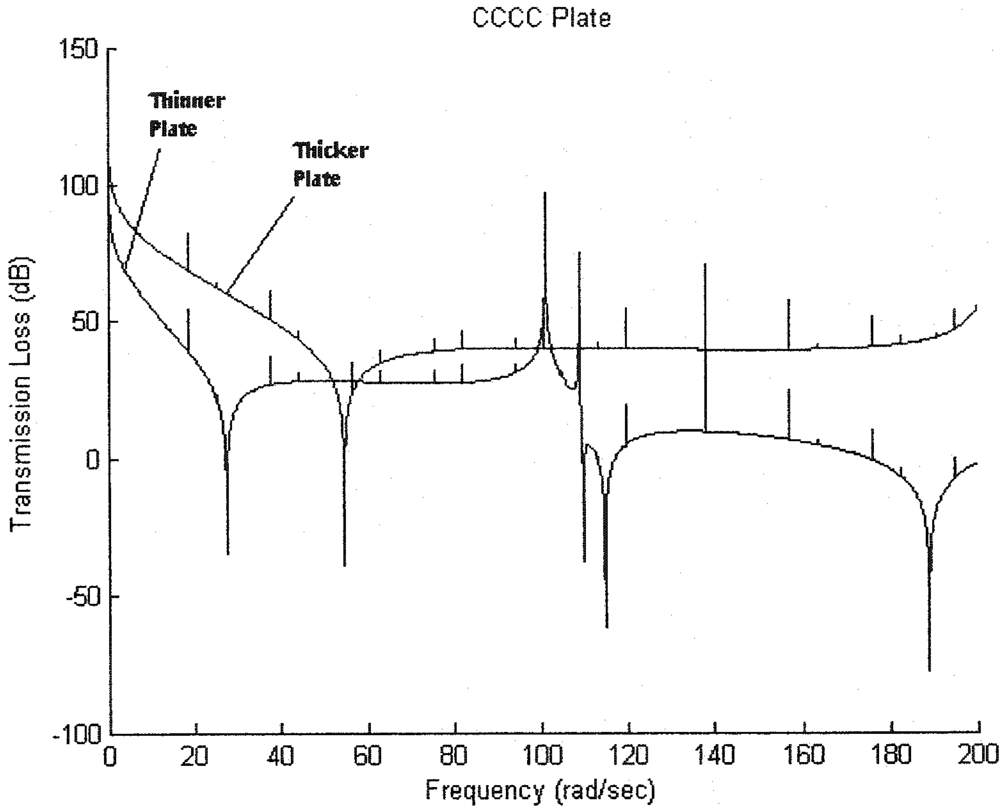

Figure 6.11 Transmission Loss for different thickness of panels

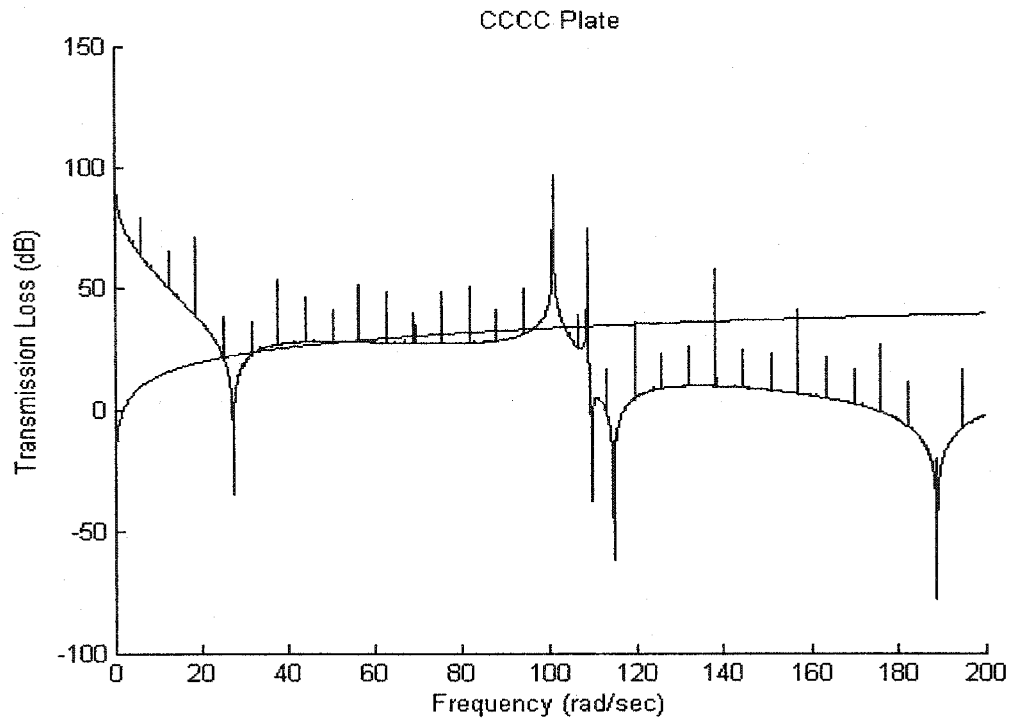

Figure 6.12 Comparison Mass law with sound transmission loss through panel by EFG

In Figure 6.12, the mass law curve is plotted by using the formula (6.45). From this plot, the calculated result differs from that determined theoretically from the Mass Law. But above the fundamental frequency, in the mass controlled region, the calculated results agree very well with mass law.

This chapter is devoted the calculation of response of panel and sound transmission loss through panel by Element Free Galerkin (EFG) method. The results confirm that EFG method provide a new way to sound transmission loss prediction. In the following chapter, the results are discussed, conclusions are drawn and some improvements for future work are suggested.

# Chapter 7

# Conclusion and Recommendations for Future Work

## 7.1 Conclusions

The study presents an application of the Element Free Galerkin Method in the prediction sound transmission loss through panels.

In order to demonstrate the benefits of the EFG method, modal analysis and harmonic forced vibration of one-dimensional and two-dimensional elastic bodies with different boundaries have been carried out. Matlab programs are developed for rod, beam, in-plane plate, thin plate free vibration. The results obtained are in good agreement with closed-form solutions and FEM solution. Based on the mode shapes of thin plate by Element Free Method (EFG) and Modal Superposition Method, Matlab program for Sound transmission loss through panels with fully clamped boundary conditions are also developed. The results also confirm that the computed frequencies agree well with the Mass Law.

The clear advantages of mesh free EFG method over the FEM are: (i). no elements are needed in shape function construction. In FEM, shape functions are created based on elements, therefore, much work is involved in the formulation of all different types of elements. As seen, in chapter 4, when problem is changed from rod to beam bending, we just increase the order of polynomial base and compute the second derivatives of shape function to finish natural frequency and mode shape calculation. Each node only has one variable for beam bending in EFG while in FEM each node has two variables (deflection and slope).

(ii). Shape functions are constructed in terms of field nodes which represent the real structure in EFG method. There is no need for node connection, and hence no non-conforming issues in EFG, which exists on the interface between the finite elements, as there is no element boundary in the meshfree method.

(iii). The discretized system equations by EFG are smaller than those generated by FEM. For thin plate problem, the nodal variable in the EFG methods is only one (deflection) compared to three in the element-based formulation (one deflection and two rotations). The dimension of the system equations is therefore one third of that generated using FEM.

Although the meshfree methods mentioned above offer considerable versatility, they are inherently more computationally expensive than finite element methods because shape function formulation by MLS is very complex and time consuming. Some shortcomings are found in the procedure of the study of EFG method.

1. Background cells are required for the integration of system matrices. The method is not truly mesh free.

2. Moving Least square methods interpolants do not pass through the data because the interpolation functions are not equal to unity at the nodes. It complicates the imposition of essential boundary conditions.

3. In the Element Free Galerkin (EFG) method, the shape functions are non-polynomial, they are rational functions of the spatial coordinates. More importantly, the shape

functions in EFG method are integrated without consideration of their support boundaries, as shown in Figure 7.1. The background cells are not the best local domains for the spatial integration [45] [46] [47] [48]. The accuracy does not always improve with increasing Gauss order.

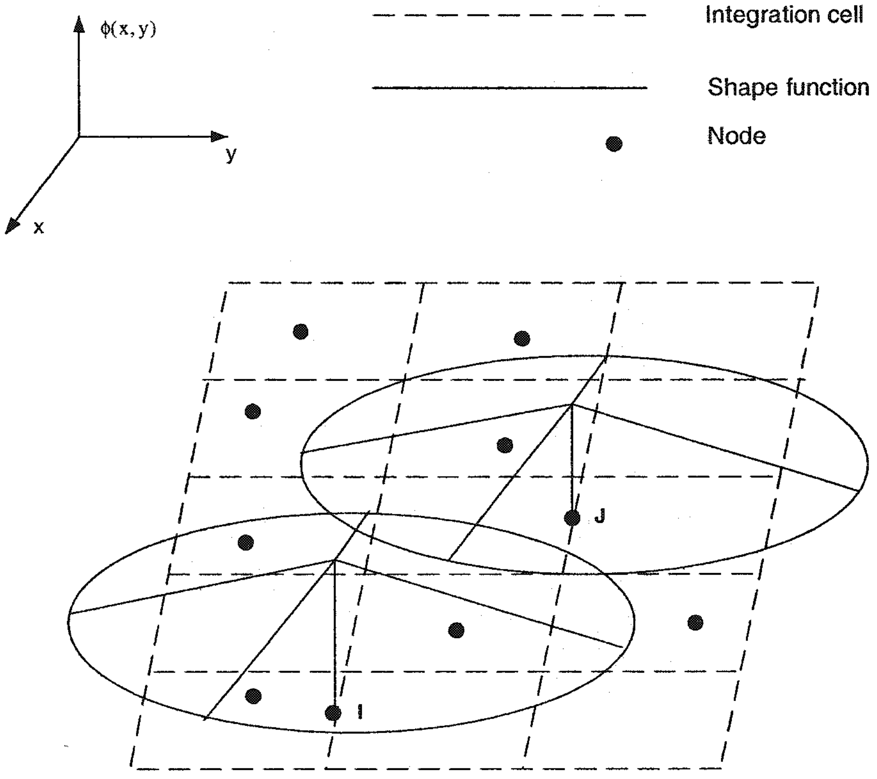

Figure 7.1 Spatial relationship between integration cell and support boundaries using radial weights. Supports are shown for nodes I and J only

However, these disadvantages are so heavily outweighed by the potential savings which are brought about by the absence of elements and the consequent avoidance of element meshing and hence the method looks very promising indeed. Some of the advantages of the Element Free Galerkin (EFG) technique are compared to the finite element method and have been brought forward.

Specific conclusions based on the present study are:

(1) The panel natural frequencies and mode shapes agree quite closely with the closed-form analytical solution. Sometimes, the accuracy is even higher than that by FEM. Modal analysis can be done by EFG. It is a very important first step because the results will be used later by modal superposition method to calculate the response of the panel.

(2) The responses of the panel under sound pressure excitation are conveniently computed by EFG and modal superposition method.

(3) Sound Transmission Loss through panel by EFG has a good agreement with mass law. Maximum sound transmission through the panel takes place at the resonance. An increase in transmission loss is expected with increasing mass because the heavier the panel the less it vibrates in response to the sound waves and hence the less sound energy it radiates on the other side. These conclusions agree very well with mass law, and hence the validation of the calculation of sound transmission loss based on the Element Free Galerkin method has been confirmed. It is a new, attractive modeling and simulation method which can predict sound transmission loss through panel reasonably well.

## 7.2 Recommendations for Future Work

From the above summary, improvements and future work is intended in the following directions:

1) In the present study, only simple geometry elastic body is considered. In order to realize the advantages of EFG method for real practical applications, it is necessary to apply it to more complex geometries.

2) When sound transmission loss is calculated, it was assumed that the air and panel are uncoupled. In the future work, the acoustics can be coupled with elastic panel by Element Free Galerkin (EFG) method.

3) Coupling Element Free Galerkin (EFG) method and finite element method and taking advantages of both method to apply boundaries conditions. EFG shape functions $\phi^{EFG}_{i(x_j)}$, however, in general do not have this property. A result of the EFG shape function failing the selective property is that essential boundary conditions cannot be imposed directly. In the present study, Lagrange Multipliers technique has been proposed to overcome this inconvenience [49]. This method will enlarge the dimension of system equations for static problem and therefore it will cost more computer time. For dynamic problems, another way to apply boundary conditions may be explored

4) Other possible work involves the study of sound transmission through more than one flexible panel, and also sound transmission into enclosures.

The Meshfree method is a very new exciting area of research. There exist many problems, that offer ample opportunities for research to develop the next generation of numerical method. Meshfree methods still require considerable improvement before they are equal in prominence to finite element method to solve practical engineering problem.

# References

[1] M. J. Crocker, P. K. Raju and B. Forssen, July-August (1981), "Measurement of Transmission Loss of Panel by the Direct Determination of Transmitted Acoustic Intensity", Noise Control Engineering, 6-11.

[2] A. cops and M. Minten (1984), "comparative Study Between the Sound Intensity Method and the Conventional Two-room Method to Calculate the Sound Transmission Loss of Wall Constructions", Noise Control Engineering, 104-111.

[3] R. E. Halliwell and A. C. C. Warnock (1985), "Sound Transmission Loss: Comparison of Conventional Techniques with Sound Intensity Techniques", Journal of the Acoustical Society of America, Vol 77 (6), 2094-2103.

[4] A. De Mey (1985), "Sound Transmission Loss Measurements by the Sound Intensity Techniques", Concordia University, Montreal, Canada.

[5] X. Vruvides (1981), "Effect of Room Geometry on the Transmission Loss of Panels", Concordia University, Montreal, Canada.

[6] W. B. Mcdonald, R. Vaicatis, and M. K. Myers, (1978)," Noise Transmission Through Plates into An Enclosure", NASA TP-1173.

[7] R. B. Bhat and G. Mundkur (1992), " Plate Characteristic Functions to Study Sound Transmission Loss Through Panels", Second International Congress on Recent Developments in Air and Structure-Borne Sound and Vibration, Auburn University, March 4-6, 1992.

[8] R. B. Bhat, G. Mundkur and J. Singh (1993), "Plate Characteristic Functions and Natural Frequencies of Vibration of Plates by Iterative Reduction of Partial Differential Equation", Journal of Vibration and Acoustics, volume 115.

[9] R. B. Bhat and G. Mundkur (1993), " Vibration of Plates Using Plate Characteristic Functions Obtained by Reduction of Partial Differential Equation", Journal of Sound and Vibration, No. 1, volume 161.

[10] M. Balike (1993), "Evaluation and Control of Noise Transmission Through a Cavity Backed Flexible Plate", Concordia University, Montreal, Canada.

[11] C. T. F. Ross, "Finite Element Methods in Engineering Science", Ellis Horwood Limited,1990.

[12] http://www.mece.ualberta.ca/staff/fyfe/acousticsilencers

[13] http://www.ansys.com/customer_stories/conference_papers/2000

[14] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming and P. Krysl (1996). Mesh-less Methods: An Overview and Recent Developments. Computer Methods in Applied Mechanics and Engineering 139, 3-47.

[15] T. Belytschko, Y. Y. Lu and L. Gu (1995), "Crack propagation By Element free Galerkin method", International Journal for Numerical Methods in Engineering 32, 2547-2570.

[16] P. Krysl and T. Belytschko (1999), "The Element Free Galerkin method for dynamic propagation of arbitary 3-D cracks", International Journal for Numerical Methods in Engineering 44,767-800.

[17] S. Jun and W. K. Liu (1998), "Explicit Reproducing Kernel Particle Methods for Large Deformation Problems", International Journal for Numerical Methods in Engineering 41, 137-166.

[18] T. Belytschko, Y. Y. Lu, and L. Gu (1994), "Element Free Galerkin Methods", International Journal for Numerical Methods in Engineering 37, 229-256.

[19] R. Szilard, "Theory and Analysis of Plates Classical and Numerical Methods", Prentice-Hall, 1974.

[20] S. Rao, "The Finite Element method in Engineering", Second Edition, Pergamon Press, 1989.

[21] M. Bata and V. Plachy, "Analysis of Dynamic Effects on Engineering Structure", Elsevier Science Publishers, 1987.

[22] A. J. Davies, "The Finite Element Method A First Approach", Clarendon Press, 1980.

[23] L. Meirovitch, "Principles and Techniques of Vibrations", Prentice Hall,1997.

[24] B. Nayroles, G. Touzot, and P. Villon (1992), "Generalizing the finite element method: diffuse approximation and diffuse elements", Computational Mechanics, 10, 307-318.

[25] P. Lancaster, and K. Salkauskas (1981), "Surfaces Generated by Moving Least Squares Methods", Mathematics of Computation 37, 141-158.

[26] P. Krysl and T. Belytschko (1996), " Analysis of Thin Plate by the Element Free Galerkin method", Computational Mechanics 17,26-35.

[27] Y. Chen, "Vibrations: Theoretical Methods", Addison-Wesley Publishing Company, 1966.

[28] L. G. Jaeger, "Elementary Theory of Elastic Plates", Pergamon Press 1964.

[29] Y. H. Wang and W. D. Li (2002), " Parametric Study for An Efficient Meshless Method in Vibration Analysis", Journal of Sound and Vibration 255(2), 261-279.

[30] T. Belytschko, D. Organ, and Y. Krongauz (1995), "A Coupled Finite Element-Element free Galerkin Method", Computational Mechanics 17, 186-195.

[31] A. E. Ouatouati and D. A. Johnson (1999), "New Approach for Numerical Modal Analysis Using the Element Free Method", International Journal for Numerical Methods in Engineering 46,1-27.

[32] G. R. Liu, "Mesh Free Methods – Moving beyond finite element method", CRC Press LLC, ~ 800 pages, July, 2002.

[33] K. Hyde, J. Y. Chang, C. Bacca and J.A.Wickert (2001)," Parameter Studies for Plane Stress in-plane Vibration of Rectangular Plates", Journal of Sound and Vibration 247(3),471-487.

[34] M. H. Jawad, "Theory and Design of Plate and Shell Structures", Chapman & Hall 1994.

[35] G. R. Liu, and X. L. Chen (2001), "A mesh free method for static and free vibration analysis of thin plates of arbitrary shape", Journal of Sound and Vibration. 241 (5), 839-855.

[36] M. Petyt, "Introduction to Finite Element Vibration Analysis", Cambridge Uniersity Press 1990.

[37] D. Schiff, "Dynamic Analysis and Failure Models of Simple Structures", John Wiley & Sons,1990.

[38] Y. T. Gu, G. R. Liu (2001), "A Meshless Local Petrov-Galerkin (MLPG) Method for Free and Forced Vibration Analyses for Solids", Computational Mechanics 27(3), 188-198.

[39] A. Dimarogonas, "Vibrations for Engineers", Second Edition, Prentice Hall, 1996

[40] I. M. Smith, Programming the finite element method with application to geomechanics, John Wiley & Sons ,1982. Simon Engineering Laboratories University of Manchester Hall, 1994.

[41] W. T. Thomson, "Theory of Vibration with Application", Fifth Edition, Prentice Hall, 1998.

[42] Lawrence E.Kinsler and Austin R. Frey, "Fundamentals of Acoustics", Second Edition, John Wiley & Sons, Inc, 1962.

[43] F. A. White, "Our Acoustic Environment", John Wiley & Sons, 1975.

[44] http://www.domesticsoundproofing.co.uk/tloss.htm

[45] J. Dolbow, T. Belytschko (1999), "Numerical integration of Galerkin weak form in meshfree methods", Comput. Mech. 23 219-230.

[46] G. R. Liu and Z. H. Tu (2002), "An adaptive procedure based on background cells for meshless methods", Computer methods in applied mechanics and Engineering, 191, 1923-1943.

[47] H. J. Chung, and T. Belytschko (1998), " An error estimation in the EFG method", Computational Mechanics 21, 91-100.

[48] Z. H. Tu and G. R. Liu (2000), " An Error Estimation Based on Background Cells for Meshless Methods", The First International Conference on Structural Stability and Dynamics December 7-9, 2000, Taipei, Taiwan.

[49] Y. Y. Lu, T. Belytschko, L. Gu (1994), "A new implementation of the element free Galerkin method", Computer Methods in Applied Mechanics and Engineering 113,397-414.

# Appendix

## A. Matlab code for rod axial vibration

```
% ONE DIMENSIONAL EFG PROGRAM FOR SOLVING NATRUAL FREQUENCY OF AXIAL
%VIBRATION OF ROD
%AXIAL VIBRATION
%
% SET UP NODAL COORDINATES ALONG BAR, DETERMINE NUMBER OF CELLS
%
%x = [0.0:.5:1.0]  %THREE NODES
%x = [0.0:.25:1.0];  %FIVE NODES
%x = [0.0:.1:1.0]    % 11 NODES
x = [0.0:0.05:1.0];    % 21 NODES
%x = [0.0:0.01:1.0];    % 101 NODES
nnodes = length(x)
%print('nnodes');
ncells = nnodes-1;
%
% SET PARAMETERS FOR WEIGHT FUNCTION, MATERIAL PROPERITES
%
dmax = 2.0; % RATIO OF DMI TO CI
E=2.1e+11;area=0.001;rho=7800;
L=1;
noncons=sqrt(rho*(L^2)/(E));
%
% DETERMINE DMI FOR EACH NODE
%
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
%
%SET UP GAUSS POINTS, WEIGHTS, AND JACOBIAN FOR EACH CELL
%
gg = zeros(1,ncells);
jac = (x(2)-x(1))/2;
weight = 2;
gg = -.25:0.5:0.75 ;  %THREE NODES
%gg = -.125:0.25:0.875;  %FIVE NODES
%gg = -.05:.1:0.95  ;     % 11 NODES
gg = -.025:.05:0.975;       % 21 NODES
%gg = -.005:.01:0.995 ;      % 101 NODES
gg(1) = 0.0;
%
% INITIALIZE MATRICES
%
k = zeros(nnodes);
m = zeros(nnodes);
f = zeros(nnodes,1);
GG = zeros(nnodes,1);
H=zeros(nnodes,2);
%
% LOOP OVER GAUSS POINTS
%
for j = 1:length(gg)
   xg = gg(j);
%
```

```
% DETERMINE DISTANCE BETWEEN NODES AND GAUSS POINT
%
dif = xg*ones(1,nnodes)-x;
%
% SET UP WEIGHTS W AND DW FOR EACH NODE
%
clear w dw
for i=1:nnodes
   drdx = sign(dif(i))/dm(i);
   r = abs(dif(i))/dm(i);
if r<=0.5
w(i) = (2/3) - 4*r*r + 4*r^3;
dw(i) = (-8*r + 12*r^2)*drdx;
elseif r<=1.0 & r>0.5
w(i) = (4/3)-4*r+4*r*r -(4/3)*r^3;
dw(i) = (-4 + 8*r-4*r^2)*drdx;
%elseif r>1.0
else r>1.0;
w(i) = 0.0;
dw(i) = 0.0;
end
end
%
%SET UP SHAPE FUNCTIONS AND DERIVATIVES
%
won = ones(1,nnodes);
nul=zeros(1,nnodes-1);
p = [won;x];
B = p.*[w;w];
pp = zeros(2);
A = zeros(2);
dA = zeros(2);
for i=1:nnodes
   pp = p(1:2,i)*p(1:2,i)'; %p(1:2,i) 2x1 loop 11 times;p(1:2,i)' 1x2 11
loop 11 times;pp 2x2
A = A+w(1,i)*pp ;      %A 2x2
dA = dA+dw(1,i)*pp;    %w(1,i) 1x1
end
Ainv = inv(A);
pg = [1 xg];
phi = pg*Ainv*B;
db = p.*[dw;dw];
da = -Ainv*(dA*Ainv);
dphi = [0 1]*Ainv*B+pg*(da*B+Ainv*db);
%
% ASSEMBLE DISCRETE EQUATIONS
%
if j == 1
GG = -phi';
 H=phi;
elseif j>1
 k = k+(weight*E*area*jac)*(dphi'*dphi);
 m = m+(weight*rho*area*jac)*(phi'*phi);
end
end
k;
m;
```

```
% ENFORCE BOUNDARY CONDITION USING ORTHOGONAL TRANSFORM TECHNIQUES
% SINGULAR VALUE DECOMPOSITION
% H(nbxnt)=R(nbxnb)*D(nbxnt)*V(ntxnt)
% PARTITION V
% CONDENSE k m
% kmat=V'((nt-r)*nt)*k*V(nt*(nt-r)),mmat=V'((nt-r)*nt)*m*V(nt*(nt-r))
[R,D,V] = svd(H);
%R, D, V
%R*D*V'  % show that decomposition works
Vpart=V(:,(2+0):nnodes);
kmat=Vpart'*k*Vpart;
mmat=Vpart'*m*Vpart;
%
%SOLVE FOR EIGENVALUE
%
[v,d] = eig(kmat,mmat);
omiga=(sqrt(d));
diag(omiga);
sort(diag(omiga));
%NON DIMENSIONAL FREQUENCY
%sort(diag(omiga))*noncons
(diag(omiga))*noncons
%
%PLOT MODE SHAPE
%
W=Vpart*v;
%def=W(:,20);%1mode 21node
%def=W(:,19);%2mode 21node
def=W(:,18);%3mode 21node
plot(x,def,'o','color',[0,0,0])
hold on
%plot(x,def,'color',[0,0,0])
ycor=zeros(1,nnodes);
hold on
%plot(x,ycor,'o','color',[0,0,0]);
hold on
%plot(x,ycor);
title('Clamped-free rod (The First Mode Shape)')
```

## B. Matlab code for beam bending vibration

```
% ONE DIMENSIONAL EFG PROGRAM FOR SOLVING NATRUAL FREQUENCY OF
CANTILEVER BEAM
%BENDING VIBRATION
function main()
% SET UP NODAL COORDINATES ALONG BAR, DETERMINE NUMBER OF CELLS
%x = [0.0:.5:1.0]  %THREE NODES
%x = [0.0:.25:1.0];  %FIVE NODES
%x = [0.0:.1:1.0];    % 11 NODES
%x = [0.0:0.05:1.0];    % 21 NODES
%x = [0.0:0.025:1.0];    % 41 NODES
x = [0.0:0.01:1.0];    % 101 NODES
%x = [0.0:(1/120):1.0];    % 121 NODES
%x = [0.0:(1/21):1.0];    % 22 NODES
%x = [0.0:(1/11):1.0]    % 12 NODES
%x = [0.0:(1/5):1.0];    % 6 NODES
L=1;
E=2.1e+11;
b=0.01; % thickness
h=0.1; % height
AREA=b*h; % cross sectional area in [m^2]
jp=h*h*h*b/12; % planar mom. of inertia in [m^4]
ro=7800; % density in [kg/m^3]
noncons=sqrt(ro*AREA*(L^4)/(E*jp));
xsquare=x.*x;
xpower3=x.*xsquare;
nnodes = length(x);
ncells = nnodes-1;
%
% SET PARAMETERS FOR WEIGHT FUNCTION (SUPPORT DOMAIN)
%
dmax = 2; % RATIO OF DMI TO CI(max distance between two nodes in one
%support domain)
jac = (x(2)-x(1))/2;
weight = 2;
%
% DETERMINE DMI FOR EACH NODE
%
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
%SET UP INTEGRATION POINTS, WEIGHTS, AND JACOBIAN FOR EACH CELL
%gg = zeros(1,ncells);
%gg = -.25:0.5:0.75   %THREE NODES
%gg = -.125:0.25:0.875  %FIVE NODES
%gg = -.05:.1:0.95;       % 11 NODES
%gg = -.025:.05:0.975;       % 21 NODES
%gg = -.0125:.025:0.9875;       % 41 NODES
gg = -.005:.01:0.995 ;      % 101 NODES
%gg = -(1/240):(1/120):(1-1/240);        % 121 NODES
%gg = -(1/42):(1/21):(1-1/42);       % 22 NODES
%gg = -(1/22):(1/11):(1-1/22);      % 12 NODES
%gg = -(1/10):(1/5):(1-1/10) ;   % 6 NODES
% INITIALIZE MATRICES
k = zeros(nnodes);
m = zeros(nnodes);
```

```
f = zeros(nnodes,1);
H=zeros(nnodes,2);
%
% LOOP OVER INTEGRATION POINTS
%
for j = 1:ncells
      [xq,jacobi]=mapgauss(x,j);
      for jg=1:2
    xg = xq(jg);
   xgsquare=xg^2;
   xgpower3=xg^3;
%
% DETERMINE DISTANCE BETWEEN NODES AND GAUSS POINT
%
dif = xg*ones(1,nnodes)-x;
%
% SET UP WEIGHTS W AND DW FOR EACH NODE
%
clear w
clear dw
clear d2w
for i=1:nnodes
 drdx = sign(dif(i))/dm(i);
%drdx = 1

   r = abs(dif(i))/dm(i);
if r<=0.5
w(i) = (2/3) - 4*r*r + 4*r^3;
dw(i) = (-8*r + 12*r^2)*drdx;
d2w(i)=(-8+24*r)*drdx;
%w(i) = 1- 6*r*r + 8*r^3-3*r^4;
%dw(i) = (-12*r + 24*r^2-12*r^3)*drdx;
%d2w(i)=(-12+48*r-36*r^2)*drdx;

elseif r<=1.0 & r>0.5
w(i) = (4/3)-4*r+4*r*r -(4/3)*r^3;
dw(i) = (-4 + 8*r-4*r^2)*drdx;
d2w(i)=(8-8*r)*drdx;
%w(i) = 1- 6*r*r + 8*r^3-3*r^4;
%dw(i) = (-12*r + 24*r^2-12*r^3)*drdx;
%d2w(i)=(-12+48*r-36*r^2)*drdx;

%elseif r>1.0
else r>1.0;

w(i) = 0.0;
dw(i) = 0.0;
d2w(i) = 0.0;

end
end
%
%SET UP SHAPE FUNCTIONS AND DERIVATIVES
%
won = ones(1,nnodes);
nul=zeros(1,nnodes-1);
%p = [won;x];
```

```
p = [won;x;xsquare];
%p = [won;x;xsquare;xpower3];
%B = p.*[w;w];
B = p.*[w;w;w];
%B = p.*[w;w;w;w];
%pp = zeros(2);
%A = zeros(2);
%dA = zeros(2);
%d2A = zeros(2);
pp = zeros(3);
A = zeros(3);
dA = zeros(3);
d2A = zeros(3);
%pp = zeros(4);
%A = zeros(4);
%dA = zeros(4);
%d2A = zeros(4);
for i=1:nnodes
   %pp = p(1:2,i)*p(1:2,i)';
   pp = p(1:3,i)*p(1:3,i)';
   %pp = p(1:4,i)*p(1:4,i)'; %
A = A+w(1,i)*pp ;      %A 2x2
dA = dA+dw(1,i)*pp;    %dw(1,i) 1x1
d2A = d2A+d2w(1,i)*pp;    %d2w(1,i) 1x1
end
Ainv = inv(A);
%pg = [1 xg];
%pg = [1 xg xgsquare];
pg = [1 xg xgsquare ];
%pg = [1 xg xgsquare xgpower3];
phi = pg*Ainv*B;
%db = p.*[dw;dw];  %B'
db = p.*[dw;dw;dw];  %B'
%db = p.*[dw;dw;dw;dw];  %B'
%d2b= p.*[d2w;d2w]; %B''
d2b= p.*[d2w;d2w;d2w]; %B''
%d2b= p.*[d2w;d2w;d2w;d2w]; %B''
da = -Ainv*(dA*Ainv);
%dphi = [0 1]*Ainv*B+pg*(da*B+Ainv*db)
dphi = [0 1 2*xg]*Ainv*B+pg*(da*B+Ainv*db);
%dphi = [0 1 2*xg 3*xg^2]*Ainv*B+pg*(da*B+Ainv*db)
%SECOND DERIVATIVE OF PHI
gama0=Ainv*pg';
%gama1=Ainv*([0 1]'-dA*Ainv*pg');
gama1=Ainv*([0 1 2*xg]'-dA*Ainv*pg');
%gama1=Ainv*([0 1 2*xg 3*xg^2]'-dA*Ainv*pg');
gama2=Ainv*([0 0 2 ]'-2*dA*gama1-d2A*gama0);
%gama2=Ainv*([0 0 2 6*xg]'-2*dA*gama1-d2A*gama0);
%gama2=Ainv*([0 0]'-2*dA*gama1-d2A*gama0);
d2phi=(gama2')*B+2*(gama1')*db+(gama0')*d2b;
%
% ASSEMBLE DISCRETE EQUATIONS
%
if j == 1

 H=[phi;dphi];
elseif j>1
```

```
     k = k+E*jp*weight*jac*(d2phi'*d2phi);%bending
     m = m+ro*AREA*weight*jac*(phi'*phi);

  end  % if elseif
  end %jg
end   %j
% ENFORCE BOUNDARY CONDITION USING ORTHOGONAL TRANSFORM TECHNIQUES
% SINGULAR VALUE DECOMPOSITION
% H(2nbxnt)=R(2nbx2nb)*D(2nbxnt)*V(ntxnt)
% PARTITION V
% CONDENSE k m
% kmat=V'((nt-r)*nt)*k*V(nt*(nt-r)),mmat=V'((nt-r)*nt)*m*V(nt*(nt-r))
[R,D,V] = svd(H);
%R, D, V
R*D*V' ; % show that decomposition works
Vpart=V(:,3:nnodes);
kmat=Vpart'*k*Vpart;
mmat=Vpart'*m*Vpart;
%
%SOLVE FOR EIGENVALUE
%
[v,d] = eig(kmat,mmat);
%v
omiga=(sqrt(d));
sort(diag(omiga));
sort(diag(omiga))*noncons
%
%PLOT MODE SHAPE
%
W=Vpart*v;
%def=W(:,9)%1mode 11node
%def=W(:,8)%2mode 11node
%def=W(:,7)%3mode 11node
%def=W(:,6)%4mode 11node
%def=W(:,5)%5mode 11node

%def=W(:,99);%1mode 101node
%def=W(:,98)%2mode 101node
%def=W(:,97)%3mode 101node
%def=W(:,96)%4mode 101node
def=W(:,95)%5mode 101node

plot(x,def,'o','color',[0,0,0])
hold on
%plot(x,def,'color',[0,0,0])
ycor=zeros(1,nnodes);
hold on
%plot(x,ycor,'o','color',[0,0,0]);
hold on
%plot(x,ycor);
title('Cantilever Beam (The Fifth Mode Shape)')

function [xq,jacobi]=mapgauss(x,j)
index=0;
ncg(1)=-1/sqrt(3);
ncg(2)=1/sqrt(3);
%WEIGHTS
```

```
ncw(1)=1;
ncw(2)=1;
one=ones(1,2);
ksai=[-1,1];
xq = zeros(1,2);
  for i=1:2
    index=index+1;
    ksaig=ncg(i);
    N=0.5*(one+ksaig*ksai);
    jacobi=(x(j+1)-x(j))/2;
         xtwo=[x(j),x(j+1)];
         xq(i)=N*xtwo';
      end
```

## C. Matlab code for in-plane plate vibration

```
%FREE VIBRATION-DYNAMIC PROBLEM
%IN -PLANE VIBRATIONOF PLATES
% 19/12/2002
%CANTILEVER PLATE
%SQUARE DOMAIN
function main()
clear;
%
% DEFINE BOUNDARIES/PARAMETERS
%
Lplate = 100;
Dplate=   10;
young =2.1e4;
nu=0.3;
density=8e-10;
thickness=1;
%Lplate = 60.96;
%Dplate=  15.24;
%young =34.474e9;
%nu=0.11;
%density=568.2;
%thickness=0.2289;
%
% PLANE STRESS DMATRIX
%
Dmat = (young/(1-nu^2))*[1 nu 0;nu 1 0;0 0 (1-nu)/2];
%
% SET UP NODAL COORDINATES
%
ndivl =50
ndivw = 5
[x,conn,numcell,numnod] = mesh(Lplate,Dplate,ndivl,ndivw);
%x
numnod
%
% DETERMINE DOMAINS OF INFLUENCE - UNIFORM NODAL SPACING
%
dmax=3.5
xindiv = Lplate/ndivl;
yindiv = Dplate/ndivw;
dm(1,1:numnod)=dmax*xindiv*ones(1,numnod);
dm(2,1:numnod)=dmax*yindiv*ones(1,numnod);
%
% SET UP QUADRATURE CELLS
%
ndivlq =50;
ndivwq = 5;
[xc,conn,numcell,numq] = mesh(Lplate,Dplate,ndivlq,ndivwq);
%xc

% SET UP GAUSS POINTS, WEIGHTS, AND JACOBIAN FOR EACH CELL

intpoint = 4
```

```
% intpoint= 2;
%intpoint = 1;
[gauss] = gauss2(intpoint);
tnumint = numcell*intpoint^2
gs = zeros(4,tnumint);
[gs] = mapgauss(xc,conn,gauss,numcell);
yy=0;
k = zeros(numnod*2);
m = zeros(numnod*2);
% LOOP OVER GAUSS POINTS TO ASSEMBLE DISCRETE EQUATIONS
for gg=gs
   yy=yy+1;
   gg;
gpos = gg(1:2);
weight = gg(3);
jac = gg(4);
index = zeros(1,2*numnod);
[phi,dphix,dphiy] = shape(gpos,dmax,x,numnod,dm);
Nmat=zeros(2,2*numnod);
Bmat=zeros(3,2*numnod);
for j=1:numnod
Nmat(1:2,(2*j-1):2*j) = [phi(j) 0;0 phi(j)];
Bmat(1:3,(2*j-1):2*j) = [dphix(j) 0;0 dphiy(j);dphiy(j) dphix(j)];
end
for i=1:numnod
   index(2*i-1) = 2*i-1;
index(2*i) = 2*i;
end
k(index,index) =
k(index,index)+(thickness*(weight*jac)*Bmat'*Dmat*Bmat);
m(index,index) =
m(index,index)+(thickness*density*(weight*jac)*Nmat'*Nmat);
end
%
% DETERMINE NODES ON BOUNDARY
%
ind1 = 0;ind2 = 0;
for j=1:numnod
if(x(1,j)==0.0)
ind1=ind1+1;
nnu(1,ind1) = x(1,j);
nnu(2,ind1) = x(2,j);
end
if(x(1,j)==Lplate)
ind2=ind2+1;
nt(1,ind2) = x(1,j);
nt(2,ind2) = x(2,j);
end
end
lthu = length(nnu);
ltht = length(nt);
f = zeros(numnod*2,1);
%
%SET UP GAUSS POINTS ALONG TRACTION BOUNDARY
%
ind=0;
gauss=gauss2(intpoint);
```

```
for i=1:(ltht-1)
jcob = abs((nt(2,i+1)-nt(2,i))/2);
%yqb Y DRIECTION QUADRATURE POINT ON BOUNDARY MAPPING FROM NATRUAL
COORDINATRES
%THE NUMBER OF POINTS IS quado
[yqb]=mapgptbdry(nt,i,intpoint);
for j=1:intpoint
   ind = ind+1;
%X
gst(1,ind)=nt(1,i);
%Y
gst(2,ind)=yqb(j);
%WEIGHTS
gst(3,ind)=gauss(2,j);
%JACOBI
gst(4,ind)=jcob;
end
end
gst;
%
%SET UP GAUSS POINTS ALONG DISPLACEMENT BOUNDARY
%
gsu=gst;
gsu;
gsu(1,1:ind)=zeros(1,ind);
qk = zeros(1,2*lthu);
%
% INTEGRATE G MATRIX  DISPLACEMENT BOUNDARY
%
GG = zeros(numnod*2,lthu*2);
ind1=0;ind2=0;
for i=1:(lthu-1)
ind1=ind1+1;
m1 = ind1; m2 = m1+1;
y1 = nnu(2,m1); y2 = nnu(2,m2);
len = y1-y2;
for j=1:intpoint
ind2=ind2+1;
gpos = gsu(1:2,ind2);
weight = gsu(3,j);
jac = gsu(4,j);
xp1 = gpos(1,1);
yp1 = gpos(2,1);
%LAGRANGE MULTIPLIERS INTERPOLATION FUNCTION
N1 = (gpos(2,1)-y2)/len; N2 = 1-N1;
%THE NUMBER OF NODES IN THE SUPPORT DOMAIN OF THE QUADRATURE ON THE
BOUNDARY
%v = domain(gpos,x,dm,numnod);
%NODE SHAPE FUNCTION IN ONE SUPPORT DOMAIN
[phi,dphix,dphiy] = shape(gpos,dmax,x,numnod,dm);
%L = length(v);
for n=1:numnod
G1 = -weight*jac*phi(n)*[N1 0;0 N1];
G2 = -weight*jac*phi(n)*[N2 0;0 N2];
c1=2*n-1;c2=2*n;c3=2*m1-1;c4=2*m1;
c5=2*m2-1;c6=2*m2;
GG(c1:c2,c3:c4)=GG(c1:c2,c3:c4)+ G1;
GG(c1:c2,c5:c6)=GG(c1:c2,c5:c6)+ G2;
end
end
end
H=GG';
%
% ENFORCE BOUNDARY CONDITION USING ORTHOGONAL TRANSFORM TECHNIQUES
% SINGULAR VALUE DECOMPOSITION
% H(2nbx2nt)=R(2nbx2nb)*D(2nbx2nt)*V(2ntx2nt)
% PARTITION V
% CONDENSE k m
% kmat=V'((2nt-r)*2nt)*k*V(2nt*(2nt-r)),mmat=V'((2nt-
r)*2nt)*m*V(2nt*(2nt-r))
%
[RR,Diag,VV] = svd(H);
%RR, Diag, VV ;
%Diag;
%RR*Diag*VV' ; % show that decomposition works
%Vpart=VV(:,11:numnod*2);
Vpart=VV(:,(2*lthu+1):numnod*2);
kmat=Vpart'*k*Vpart;
mmat=Vpart'*m*Vpart;
%
%SOLVE FOR EIGENVALUE
%
%[evec,eval] = eig(kmat,mmat,'qz');
[evec,eval] = eig(kmat,mmat);
omiga=(sqrt(eval));
selfre=sort(diag(omiga));
sort(diag(omiga))/(2*pi);
(diag(omiga))/(2*pi)
%
%MODE SHAPE POSSESS ORTHOGONALITY
%
W=Vpart*evec;   %EIGENVECTOR
mdiag=W'*m*W;
selmdiag=(diag(mdiag));
smdiag=sort(diag(mdiag));
kdiag=W'*k*W;
selkdiag=(diag(kdiag));
skdiag=sort(diag(kdiag));
rankm=rank(mdiag);
%
%VERIFY IF THE NATRUAL FREQUENCIES IS CORRECTLY COMPUTED
%
for i=1:rankm
   omigayy(i)=sqrt(selkdiag(i)/selmdiag(i));
end
colomigayy=omigayy';
%
%SORT NATRUAL FREQUENCIES AND REMEMBER THEIR ORIGINAL INDEX
%
ocolomigayy=colomigayy; %ocolomigayy store colomigayy since the member
of colomigayy will change sequence
for last=rankm:-1:1
[largest]=indexoflargest(colomigayy,last);
   temp=colomigayy(largest);
   colomigayy(largest)=colomigayy(last);
   colomigayy(last)=temp;
end

for i=1:rankm
   for j=1:rankm
      if (colomigayy(i) == ocolomigayy(j))
         oriindex(i)=j;
      end
   end
end
colomigayy;
oriindex;
%
%RE-ARRANGE W MATCH TO ASCENDING NATURAL FREQUENCIES
%
for i=1:rankm
   sortW(:,i)=W(:,oriindex(i));
end
W;
sortW;
%
%POST-PROCESSING  Cell 3x2
%
xpost=0:(100/3):100;
ypost=-5:(10/2):5;
index=ndivl+1;
z=sortW(:,1)'
%
%select x displacement, y displacement
%
for i=1:((ndivl+1)*(ndivw+1))
   xdisp(i)=z(2*i-1);
   ydisp(i)=z(2*i);
end
xdisp
ydisp
xcor=x(1,:)
ycor=x(2,:)
xplus=xdisp+xcor
yplus=ydisp+ycor
zzzz=zeros(1,(ndivl+1)*(ndivw+1));
for i=1:(ndivl+1)*(ndivw+1)
plot3(xplus(i),yplus(i),zzzz(i),'*')
end
%view(45,45)
%view(-45,45)
%shading interp
%omiga=(sqrt(d));
%sort(diag(omiga))
%NO CONSTRAINED ,FREE EDGES
%[v,d] = eig(k,m);
%omiga=(sqrt(d));
%sort(diag(omiga))

% MESH GENERATION PROGRAM FOR 2D BEAM IN PLANE BENDING
function[x,conn,numcell,numq] = mesh(length,height,ndivl,ndivw)
```

```
numcell= ndivw*ndivl;
numq = (ndivl+1)*(ndivw+1);
zzzz=zeros(1,(ndivl+1)*(ndivw+1));
% SET UP NODAL COORDINATES
for i = 1:(ndivl+1)
for j = 1:(ndivw+1)
x(1,((ndivw+1)*(i-1) +j))= (length/ndivl)*(i-1);
x(2,((ndivw+1)*(i-1) +j))= -(height/ndivw)*(j-1)+height/2;
plot(x(1,((ndivw+1)*(i-1) +j)),x(2,((ndivw+1)*(i-1)
+j)),'o','color',[0,0,0])
hold on
xs(j)=x(1,((ndivw+1)*(i-1) +j));
ys(j)=x(2,((ndivw+1)*(i-1) +j));
end
hold on
%plot(xs,ys,'color',[0,0,0])
end
for j = 1:(ndivw+1)
for i = 1:(ndivl+1)
xplot(1,((ndivw+1)*(i-1) +j))= (length/ndivl)*(i-1);
xplot(2,((ndivw+1)*(i-1) +j))= -(height/ndivw)*(j-1)+height/2;
xplots(i)=x(1,((ndivw+1)*(i-1) +j));
yplots(i)=x(2,((ndivw+1)*(i-1) +j));
end
hold on
%plot(xplots,yplots,'color',[0,0,0])
end
% SET UP CONNECTIVITY ARRAY
for j = 1:ndivl
for i = 1:ndivw
elemn = (j-1)*ndivw + i;
nodet(elemn,1) = elemn + (j-1);
nodet(elemn,2) = nodet(elemn,1) + 1;
nodet(elemn,3) = nodet(elemn,2)+ndivw+1;
nodet(elemn,4) = nodet(elemn,3)-1;
end
end
conn = nodet';


% This function returns a matrix with 4 gauss points and their weights
% in natural coordinates
function v = gauss2(k)
if k==4
v(1,1) =-.861136311594052575224;
v(1,2) =-.339981043584856264803;
v(1,3) = -v(1,2);
v(1,4) = -v(1,1);
v(2,1) =.347854845137453857373;
v(2,2) =.652145154862546142627;
v(2,3) = v(2,2);
v(2,4) = v(2,1);
elseif k==2
v(1,1) =-1/sqrt(3);
v(1,2) =1/sqrt(3);
v(2,1) =1;
v(2,2) =1;
```

```
else
v(1,1) =0;
v(2,1) =2;
end


%MAP GAUSS POINTS FROM NATURAL COORDINATES TO PHYSICAL COORDINATES
function [gs] = mapgauss(xc,conn,gauss,numcell)
index=0;
one = ones(1,4);
psiJ = [-1,+1,+1,-1]; etaJ = [-1,-1,+1,+1];
l = size(gauss);
l = l(2);
for e=1:numcell
% DETERMINE NODES IN EACH CELL
for j = 1:4
   je=conn(j,e);xe(j)=xc(1,je);ye(j)=xc(2,je);
   xe;
   ye;
end
for i=1:l
for j=1:l
index = index+1;
eta=gauss(1,i);psi=gauss(1,j);
%eta
%psi
N = .25*(one+psi*psiJ).*(one+eta*etaJ);
NJpsi=.25*psiJ.*(one+eta*etaJ);
NJeta=.25*etaJ.*(one+psi*psiJ);
xpsi=NJpsi*xe';ypsi=NJpsi*ye';xeta=NJeta*xe';yeta=NJeta*ye';
jcob=xpsi*yeta-xeta*ypsi;
xq = N*xe';yq = N*ye';
gs(1,index) = xq;
gs(2,index) = yq;
gs(3,index) = gauss(2,i)*gauss(2,j);
gs(4,index) = jcob;
%plot(xq,yq,'*','color',[0,0,0])
hold on
end
end
%clear xe  % yy
%clear ye  % yy
end


%MOVING LEAST SQUARE METHOD TO CONSTRUCT SHAPE FUNCTION
function [phi,dphix,dphiy] = shape(gpos,dmax,x,numnod,dm)
won = ones(1,numnod);
p = [won;x];
dif = gpos*won-x;
t = dm/dmax;
[w,dwdx,dwdy] = wgt(dif,t,x,dmax,dm);
B = p.*[w;w;w];
pp = zeros(3);
A = zeros(3);
dAx = zeros(3);
dAy = zeros(3);
```

```
for i=1:numnod
pp = p(1:3,i)*p(1:3,i)';
A = A+w(1,i)*pp;
dAx = dAx+dwdx(1,i)*pp;
dAy = dAy+dwdy(1,i)*pp;
end
Ainv=inv(A);
pg=[1 gpos'];
phi=pg*Ainv*B;
dBx=p.*[dwdx;dwdx;dwdx];
dBy=p.*[dwdy;dwdy;dwdy];
consx=-Ainv*(dAx*Ainv);
consy=-Ainv*(dAy*Ainv);
dphix = [0 1 0]*Ainv*B+pg*(consx*B+Ainv*dBx);
dphiy = [0 0 1]*Ainv*B+pg*(consy*B+Ainv*dBy);


%WEIGHT FUNCTION
function [w,dwdx,dwdy] = wgt(dif,t,v,dmax,dm)
l = length(v);
for i=1:l
drdx = sign(dif(1,i))/dm(1,i);
drdy = sign(dif(2,i))/dm(2,i);
rx = abs(dif(1,i))/dm(1,i);
ry = abs(dif(2,i))/dm(2,i);
if 1>rx>0.5
%wx = (4/3)-4*rx+4*rx*rx -(4/3)*rx^3;
%dwx = (-4 + 8*rx-4*rx^2)*drdx;

wx = 1- 6*rx^2 + 8*rx^3-3*rx^4;
dwx = (-12*rx + 24*rx^2-12*rx^3)*drdx;

elseif rx<=0.5
%wx = (2/3) - 4*rx*rx + 4*rx^3;
%dwx = (-8*rx + 12*rx^2)*drdx;

wx = 1- 6*rx^2 + 8*rx^3-3*rx^4;
dwx = (-12*rx + 24*rx^2-12*rx^3)*drdx;

else
wx=0;
dwx=0;
end
if 1>ry>0.5
%wy = (4/3)-4*ry+4*ry*ry -(4/3)*ry^3;
%dwy = (-4 + 8*ry-4*ry^2)*drdy;

wy = 1- 6*ry^2 + 8*ry^3-3*ry^4;
dwy = (-12*ry + 24*ry^2-12*ry^3)*drdy;

elseif ry<=0.5
%wy = (2/3) - 4*ry*ry + 4*ry^3;
%dwy = (-8*ry + 12*ry^2)*drdy;

wy = 1- 6*ry^2 + 8*ry^3-3*ry^4;
dwy = (-12*ry + 24*ry^2-12*ry^3)*drdy;
```

```
else
wy=0;
dwy=0;
end
w(i) = wx*wy;
dwdx(i) = wy*dwx;
dwdy(i) = wx*dwy;
end


%MAP GAUSS POINTS ON BOUNDARY FROM NATURAL COORDINATES TO PHYSICAL
COORDINATES
function [yqb]=mapgptbdry(nt,j,quadpt)
ncg(1)=-0.861136311594052;
ncg(2)=-0.339981043584856;
ncg(3)=0.339981043584856;
ncg(4)=0.861136311594052;
nt;
index=0;
one=ones(1,2);
ksai=[-1,1];
yqb = zeros(1,quadpt);
for i=1:quadpt
index=index+1;
ksaig=ncg(i);
N=0.5*(one+ksaig*ksai);
jacobi=(nt(2,j+1)-nt(2,j))/2;
xtwo=[nt(2,j),nt(2,j+1)];
yqb(i)=N*xtwo';
end

%SORT
function [index]=indexoflargest(a,size)
indexsofar=1;
for currentindex =2:size
   if(a(currentindex)>a(indexsofar))
  indexsofar=currentindex;
   end
end
index=indexsofar;
```

## D. Matlab code for sound transmission loss through the panel

```
% 05/03/2003
%
%Forced vibration to Harmonic Excitation F(t)=Fo*cos(Omega*t)
%
%FULLY CLAMPED PLATE
%
%FLEXURAL VIBRATIONOF OF PLATES
%
%SQUARE DOMAIN
%
%WEIGHT FUNCTION W=1-6r^2+8r^3-3r^4 1>r>0;0 r>1
%
%GAUSS POINT 1, 2, 4, 6, 9 IN EACH CELL(2 DIMENSIONAL CELL)
%
%BOUNDARY EACH CELL(1 DIMENSIONAL CELL) 4 GAUSS POINTS
%

function main()
clear all;
%
% DEFINE BOUNDARIES/PARAMETERS
%
%DATA FROM PAPER :JOURNAL OF SOUND AND VIBRATION (2002) 255(2),261-279
%Lplate = 4;
%Wplate=  4;
%young =20.0e9;
%nu=0.3;
%density=3000;
%thickness=0.2;
%DATA FROM Dr.LIU BOOK MESHFREE METHOD
Lplate = 10;
Wplate=  10;
young =2.0e11;
nu=0.3;
density=8000;
thickness=0.05;
%
%LOAD
%
q=1;
%
%TIME STEP
%
steps=100;
dtime=0.5;
%
%RAYLEIGH DAMPING COEFFICIENTS
%
alfa=0.00;%k
beta=0;%m
%
%FREQUENCY OF FORCE
%
% Omega=1
```

```
fprintf('THE SHAPE OF SUPPORT IS SQUARE ')
%
% PLANE STRESS DMATRIX
%
D0=young*(thickness^3)/(12*(1-nu^2));
Dmat = (young/(1-nu^2))*[1 nu 0;nu 1 0;0 0 (1-nu)/2];
%
% SET UP NODAL COORDINATES
%
ndivl =6;ndivw =6;
[x,conn,numcell,numnod] = mesh(Lplate,Wplate,ndivl,ndivw);
fprintf('THE NUMBER OF TOTAL NODES')
numnod
xsquare=x.*x;
xy=x(1,:).*x(2,:);
%
%6 POLYNOMIAL TERMS 1 X Y X^2 y^2 XY
%
x;
numnod;
%
% DETERMINE DOMAINS OF INFLUENCE - UNIFORM NODAL SPACING
%
fprintf('dmax')
dmax=4
xindiv = Lplate/ndivl;
yindiv = Wplate/ndivw;
dm(1,1:numnod)=dmax*xindiv*ones(1,numnod);
dm(2,1:numnod)=dmax*yindiv*ones(1,numnod);
%
% SET UP QUADRATURE CELLS
%
ndivlq =6;ndivwq =6 ;
%fprintf('THE NUMBER OF CELLS')
%numcell
[xc,conn,numcell,numq] = mesh(Lplate,Wplate,ndivlq,ndivwq);
%
% SET UP GAUSS POINTS, WEIGHTS, AND JACOBIAN FOR EACH CELL
%
%intpoint = 9;
%intpoint = 6;
intpoint = 4;
% intpoint= 2
%intpoint = 1;
[gauss] = gauss2(intpoint);
tnumint = numcell*intpoint^2;
gs = zeros(4,tnumint);
[gs] = mapgauss(xc,conn,gauss,numcell);
yy=0;
k = zeros(numnod);
m = zeros(numnod);
force=zeros(numnod,1);
%
% LOOP OVER GAUSS POINTS TO ASSEMBLE DISCRETE EQUATIONS
%
for gg=gs
yy=yy+1;
```

```
gg;
gpos = gg(1:2);
gposquare=gg(1:2).*gg(1:2);
gposxy=gg(1)*gg(2);
weight = gg(3);
jac = gg(4);
index = zeros(1,2*numnod);
[phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(gpos,gposquare,gposxy,dmax,x,xsquare,xy,numnod,dm);
Nmat=zeros(1,numnod);
Bmat=zeros(3,numnod);
for j=1:numnod
Nmat(1,j) = [phi(j)];
Bmat(1:3,j) = [d2phix(j);d2phiy(j);2*d2phixy(j)];
end
k=k+((thickness^3/12)*(weight*jac)*Bmat'*Dmat*Bmat);
m=m+(thickness*density*(weight*jac)*Nmat'*Nmat);
force=force+(weight*jac)*Nmat'*q;
end
force;
%
% DETERMINE NODES ON BOUNDARY
%
%LEFT AND RIGHT
%
%LEFT EDGE
ind1 = 0;ind2 = 0;
for j=1:numnod
if(x(1,j)==0.0)
ind1=ind1+1;
nleft(1,ind1) = x(1,j);
nleft(2,ind1) = x(2,j);
end
%RIGHT EDGE
if(x(1,j)==Lplate)
ind2=ind2+1;
nright(1,ind2) = x(1,j);
nright(2,ind2) = x(2,j);
end
end
lleft  = length(nleft);
lright = length(nright);
%
%TOP AND BOTTOM
%
%TOP EDGE
ind3 = 0;ind4 = 0;
for j=1:numnod
if(x(2,j)==Wplate/2)
ind3=ind3+1;
top(1,ind3) = x(1,j);
top(2,ind3) = x(2,j);
end
%BOTTOM EDGE
if(x(2,j)==-Wplate/2)
ind4=ind4+1;
bottom(1,ind4) = x(1,j);
bottom(2,ind4) = x(2,j);
end
end
ntop=top(:,3:(ndivl-1));
nbottom=bottom(:,3:(ndivl-1));
%ntop=top;
%nbottom=bottom;
ltop  = length(ntop);
lbottom = length(nbottom);
%
%SET UP GAUSS POINTS ALONG RIGHT BOUNDARY
%
direction=2; %y
intpointb=4;
ind=0;
gauss=gauss2(intpointb);
for i=1:(lright-1)
jcob = abs((nright(2,i+1)-nright(2,i))/2);
%yqb Y DRIECTION QUADRATURE POINT ON BOUNDARY MAPPING FROM NATRUAL
COORDINATRES
%THE NUMBER OF POINTS IS quado
[yqb]=mapgptbdry(nright,i,intpointb,direction);
for j=1:intpointb
ind = ind+1;
%X
gsright(1,ind)=nright(1,i);
%Y
gsright(2,ind)=yqb(j);
%WEIGHTS
gsright(3,ind)=gauss(2,j);
%JACOBI
gsright(4,ind)=jcob;
end
end
gsright;
%
%SET UP GAUSS POINTS ALONG LEFT BOUNDARY
%
gsleft=gsright;
gsleft;
gsleft(1,1:ind)=zeros(1,ind);
%
%SET UP GAUSS POINTS ALONG TOP BOUNDARY
%
direction=1;
intpointb=4;
ind=0;
gauss=gauss2(intpointb);
for i=1:(ltop-1)
jcob = abs((ntop(1,i+1)-ntop(1,i))/2);
%yqb Y DRIECTION QUADRATURE POINT ON BOUNDARY MAPPING FROM NATRUAL
COORDINATRES
%THE NUMBER OF POINTS IS quado
[xqb]=mapgptbdry(ntop,i,intpointb,direction);
xqb;
for j=1:intpointb
   ind = ind+1;
```

```
%X
gstop(1,ind)=xqb(j);
%Y
gstop(2,ind)=ntop(2,i);
%WEIGHTS
gstop(3,ind)=gauss(2,j);
%JACOBI
gstop(4,ind)=jcob;
end
end
gstop;
%
%SET UP GAUSS POINTS ALONG BOTTOM BOUNDARY
%
gsbottom=gstop;
gsbottom;
ind;
gsbottom(2,1:ind)=-gstop(2,1:ind);
%
% INTEGRATE G MATRIX  DISPLACEMENT BOUNDARY
GG = zeros(numnod,2*numnod);
GGL = zeros(numnod,2*numnod);
GGR = zeros(numnod,2*numnod);
GGT = zeros(numnod,2*numnod);
GGB = zeros(numnod,2*numnod);
%
%CONSTRAIN THE LEFT EDGE
%
ind1=0;ind2=0;
for i=1:(lleft-1)%LOOP FOR BOUNDARY CELL
ind1=ind1+1;
m1 = ind1; m2 = m1+1;
y1 = nleft(2,m1); y2 = nleft(2,m2);
len = y1-y2;
for j=1:intpointb %LOOP FOR GAUSS POINT IN EACH CELL
ind2=ind2+1;
gposb = gsleft(1:2,ind2);
gposbsquare=gsleft(1:2,ind2).*gsleft(1:2,ind2);
gposbxy= gsleft(1,ind2)* gsleft(2,ind2);
weight = gsleft(3,j);
jac = gsleft(4,j);
%LAGRANGE MULTIPLIERS INTERPOLATION FUNCTION
N1 = (gposb(2,1)-y2)/len; N2 = 1-N1;
[phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(gposb,gposbsquare,gposbxy,dmax,x,xsquare,xy,numnod,dm);
for n=1:numnod
G1 = -weight*jac*N1*[phi(n),dphix(n)]';
G2 = -weight*jac*N2*[phi(n),dphix(n)]';
c1=n;
c3=2*m1-1;
c4=2*m1;
c5=2*m2-1;
c6=2*m2;
%GG(c1,c3:c4)=GG(c1,c3:c4)+ G1';
%GG(c1,c5:c6)=GG(c1,c5:c6)+ G2';
GGL(c1,c3:c4)=GGL(c1,c3:c4)+ G1';
GGL(c1,c5:c6)=GGL(c1,c5:c6)+ G2';
```

```
end
end
end
GGL;
%
%CONSTRAIN THE RIGHT EDGE
%
ind1r=0;ind2r=0;
for i=1:(lright-1)
ind1r=ind1r+1;
m1 = ind1r; m2 = m1+1;
nright;
nleft;
y1 = nright(2,m1);
y2 = nright(2,m2);
len = y1-y2;
gsright;
for j=1:intpointb
ind2r=ind2r+1;
rgposb = gsright(1:2,ind2r);
rgposbsquare=gsright(1:2,ind2r).*gsright(1:2,ind2r);
rgposbxy= gsright(1,ind2r)* gsright(2,ind2r);
rweight = gsright(3,j);
rjac = gsright(4,j);
%LAGRANGE MULTIPLIERS INTERPOLATION FUNCTION
N1r = (rgposb(2,1)-y2)/len; N2r = 1-N1r;
[phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(rgposb,rgposbsquare,rgposbxy,dmax,x,xsquare,xy,numnod,dm);
for n=1:numnod
G1r = -rweight*rjac*N1r*[phi(n),dphix(n)]';
G2r = -rweight*rjac*N2r*[phi(n),dphix(n)]';
c1r=n;
c3r=2*(m1+ndivl*(ndivw+1))-1;
c4r=2*(m1+ndivl*(ndivw+1));
c5r=2*(m2+ndivl*(ndivw+1))-1;
c6r=2*(m2+ndivl*(ndivw+1));
%GG(c1,c3:c4)=GG(c1,c3:c4)+ G1';
%GG(c1,c5:c6)=GG(c1,c5:c6)+ G2';
GGR(c1r,c3r:c4r)=GGR(c1r,c3r:c4r)+ G1r';
GGR(c1r,c5r:c6r)=GGR(c1r,c5r:c6r)+ G2r';
end
end
end
GGR;
%
%CONSTRAIN THE TOP EDGE
%
ind1=0;ind2=0;
mt1=2;
%mt1=0;
for i=1:(ltop-1)
   ind1=ind1+1;
   mt1=mt1+1;
mt2=mt1+1;

m1 = ind1; m2 = m1+1;
x1 = ntop(1,m1);
```

```
x2 = ntop(1,m2);
len = x1-x2;
ntop;
gstop;
for j=1:intpointb
ind2=ind2+1;
gposb = gstop(1:2,ind2);
gposbsquare=gstop(1:2,ind2).*gstop(1:2,ind2);
gposbxy= gstop(1,ind2)* gstop(2,ind2);
weight = gstop(3,j);
jac = gstop(4,j);
%LAGRANGE MULTIPLIERS INTERPOLATION FUNCTION
N1 = (gposb(1,1)-x2)/len; N2 = 1-N1;
[phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(gposb,gposbsquare,gposbxy,dmax,x,xsquare,xy,numnod,dm);
for n=1:numnod
G1 = -weight*jac*N1*[phi(n),dphiy(n)]';
G2 = -weight*jac*N2*[phi(n),dphiy(n)]';
c1=n;%ROW INDEX
c3=2*((ndivw+1)*mt1-ndivl)-1;%COLUMN INDEX
c4=2*((ndivw+1)*mt1-ndivl);%COLUMN INDEX
c5=2*((ndivw+1)*mt2-ndivl)-1;%COLUMN INDEX
c6=2*((ndivw+1)*mt2-ndivl);%COLUMN INDEX
%GG(c1,c3:c4)=GG(c1,c3:c4)+ G1';
%GG(c1,c5:c6)=GG(c1,c5:c6)+ G2';
GGT(c1,c3:c4)=GGT(c1,c3:c4)+ G1';
GGT(c1,c5:c6)=GGT(c1,c5:c6)+ G2';
end
end
end
GGT;
%
%CONSTRAIN THE BOTTOM EDGE
%
ind1=0;ind2=0;
mb1=2;
%mb1=0;

for i=1:(lbottom-1)
   ind1=ind1+1;
   mb1=mb1+1;
mb2=mb1+1;
m1 = ind1; m2 = m1+1;
x1 = nbottom(1,m1); x2 = nbottom(1,m2);
len = x1-x2;
for j=1:intpointb
ind2=ind2+1;
gposb = gsbottom(1:2,ind2);
gposbsquare=gsbottom(1:2,ind2).*gsbottom(1:2,ind2);
gposbxy= gsbottom(1,ind2)* gsbottom(2,ind2);
weight = gsbottom(3,j);
jac = gsbottom(4,j);
%LAGRANGE MULTIPLIERS INTERPOLATION FUNCTION
N1 = (gposb(1,1)-x2)/len; N2 = 1-N1;
[phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(gposb,gposbsquare,gposbxy,dmax,x,xsquare,xy,numnod,dm);
for n=1:numnod
```

```
G1 = -weight*jac*N1*[phi(n),dphiy(n)]';
G2 = -weight*jac*N2*[phi(n),dphiy(n)]';
c1=n;
c3=2*((ndivw+1)*mb1)-1;
c4=2*((ndivw+1)*mb1);
c5=2*((ndivw+1)*mb2)-1;
c6=2*((ndivw+1)*mb2);
%GG(c1,c3:c4)=GG(c1,c3:c4)+ G1';
%GG(c1,c5:c6)=GG(c1,c5:c6)+ G2';
GGB(c1,c3:c4)=GGB(c1,c3:c4)+ G1';
GGB(c1,c5:c6)=GGB(c1,c5:c6)+ G2';
end
end
end
GGB;
GG=GGL+GGR+GGT+GGB;
%GG=GGL;
H=GG';
%
% ENFORCE BOUNDARY CONDITION USING ORTHOGONAL TRANSFORM TECHNIQUES
% SINGULAR VALUE DECOMPOSITION
% H(2ntxnt)=R(2ntx2nt)*D(2ntxnt)*V(ntxnt)
% PARTITION V
% CONDENSE k m
% kmat=V'((nt-r)*nt)*k*V(nt*(nt-r)),mmat=V'((nt-r)*nt)*m*V(nt*(nt-r))
%
[RR,Diag,VV] = svd(H);
%RR, Diag, VV ;
RR;
Diag;
rank(Diag)
%RR*Diag*VV' ; % show that decomposition works
Vpart=VV(:,(rank(Diag)+1):numnod);%L
kmat=Vpart'*k*Vpart;
mmat=Vpart'*m*Vpart;
%
%SOLVE FOR EIGENVALUE
%
[evec,eval] = eig(kmat,mmat);%evec IS NOT EIGENVECTOR
deval=diag(eval);
omiga=(sqrt(eval));
(diag(omiga));
sort(diag(omiga))
dym1=diag(evec'*kmat*evec);
dym2=diag(evec'*mmat*evec);
rankmbar=rank(evec'*mmat*evec);
for i=1:rankmbar
   dymomiga(i)=sqrt(dym1(i)/dym2(i));
end
dymomiga;

%
%NON DIMENSIONAL NATRUAL FREQUENCIES
%
selfreq=sort(diag(omiga))*((density*thickness*(Lplate^4)/D0)^0.5)
selfreq=(diag(omiga))*((density*thickness*(Lplate^4)/D0)^0.5);
%
```

```
%NO CONSTRAINED ,FREE EDGES
%
%[v,d] = eig(k,m);
%omiga=(sqrt(d));
%sort(diag(omiga));
%
%NON DIMENSIONAL NATRUAL FREQUENCIES
%
%coeff=(sort(diag(d))*density*thickness*(Lplate^4)/D0).^0.25;
%coeff(4:9,:);
%
%MODE SHAPE POSSESS ORTHOGONALITY
%
W=Vpart*evec;
mdiag=W'*m*W;
selmdiag=(diag(mdiag));
smdiag=sort(diag(mdiag));
kdiag=W'*k*W;
selkdiag=(diag(kdiag));
skdiag=sort(diag(kdiag));
rankm=rank(mdiag);
%
%VERIFY IF THE NATRUAL FREQUENCIES IS CORRECTLY COMPUTED
%
for i=1:rankm
   omigayy(i)=((sqrt(selkdiag(i)/selmdiag(i))));
end
omigayy;
colomigayy=omigayy';
sortwn=sort(omigayy);
%
%SORT NATRUAL FREQUENCIES AND REMEMBER THEIR ORIGINAL INDEX
%
ocolomigayy=colomigayy; %ocolomigayy store colomigayy since the member
of colomigayy will change sequence
for last=rankm:-1:1
[largest]=indexoflargest(colomigayy,last);
   temp=colomigayy(largest);
   colomigayy(largest)=colomigayy(last);
   colomigayy(last)=temp;
end

for i=1:rankm
   for j=1:rankm
      if (colomigayy(i) == ocolomigayy(j))
         oriindex(i)=j;
      end
   end
end
colomigayy;
oriindex;
%
%RE-ARRANGE selmdiag and selkdiag IN ORDER TO COMPUTE CRITICAL DAMPING
%
for i=1:rankm
   sortmdiag(i)=selmdiag(oriindex(i));
     sortkdiag(i)=selkdiag(oriindex(i));
```

```
end
sortmdiag;
sortkdiag;
yym=sortmdiag(1);
yyk=sortkdiag(1);
%
%RESPONSE CHOOSE NATURAL FREQUENCY
%
%FREQUENCY OF FORCE
%
force;
p=W'*force;
iiyy=0;
 for Omega=0:0.01:200
iiyy=iiyy+1;
%
%LOOP FOR EACH EQUATION
%

domiga=diag(omiga);
for choosefre=1:rankm
 wn=domiga(choosefre);
%
%CRITICAL DAMPING
%
ccr=2*sqrt(selkdiag(choosefre)*selmdiag(choosefre));
%
%DAMPING  C=ALFA*K+BETA*M
%
c=alfa*selkdiag(choosefre)+beta*selmdiag(choosefre);
zeta=c/ccr;
wd=sqrt(1-zeta^2)*wn;
%
%FORCE IN EACH EQUATION
%
force;
p=W'*force;
      x0normod(choosefre)=p(choosefre)/selkdiag(choosefre);
%x0orimod=W*x0normod';
x0= x0normod(choosefre);
%
%TIME STEP
%
t=[0:dtime:dtime*steps];
% xp(t)  Particular solution
r=Omega/wn;
Xpo=x0/sqrt((1-r^2)^2+(2*zeta*r)^2);
alpha=atan2(2*zeta*r,(1-r^2));
xp=Xpo*cos(Omega*t-alpha);%displacement
xpdot=-Omega*Xpo*sin(Omega*t-alpha);%velocity
%Homogenous solution xh(t)
xAB=inv([1,0;-zeta*wn,wd])*[0-Xpo*cos(-alpha);0+Omega*Xpo*sin(-alpha)];
Ap=xAB(1);
Bp=xAB(2);
xh=exp(-zeta*wn*t).*(Ap*cos(wd*t)+Bp*sin(wd*t));%displacement
```

```
xhdot=exp(-zeta*wn*t).*(Ap*cos(wd*t)+Bp*sin(wd*t))*(-zeta*wn)+exp(-
zeta*wn*t).*(-Ap*wd*sin(wd*t)+Bp*wd*cos(wd*t));%velocity
%disp=xp+xh;%response
%vel=xpdot+xhdot;%response
disp=xp+xh;%transmission loss calculation
vel=xpdot;% transmission loss calculation

for i=1:steps+1
   xxxxd(choosefre,i)=disp(i);
      xxxxv(choosefre,i)=vel(i);

   end
end
phydisp=W*xxxxd;
phyvel =W*xxxxv;
%
%Choose point
%
nodeid=25;
phydisp;
nodedisp=phydisp(nodeid,:);
nodevel=phyvel(nodeid,:);
%subplot(2,1,1)
%plot(t,nodedisp,'color',[0,0,0]);
%title('                          Forced Vibration of a Second-Order
System to Harmonic Excitation node 26');
%xlabel('Time (sec)');
%ylabel('Position disp (m)');
hold on
%subplot(2,1,2)
%plot(t,nodevel,'color',[0,0,0]);
%xlabel('Time (sec)');
%ylabel('Velocity (m/s)');
%
%
%
sortdisp=sort(abs(nodedisp));
sortvel=sort(abs(nodevel));
maxdisp=sortdisp(:,steps+1);
maxvel=sortvel(:,steps+1);
%
%dB OUT
%
pref=0.00002 ; %N/m^2
airdensity=1.21; %kg/m^3
souvel=343;%m/s
pin=force(nodeid,:);
pout=maxvel*airdensity*souvel;
%tl=20*log10(pin/pout);
tl=20*log10(pout/pin);
collecttl(iiyy)=tl;
collectomega(iiyy)=Omega/(2*pi);
collectomega(iiyy)=Omega;% rad/s
hold on
end
plot(collectomega,collecttl,'color',[0,0,0])
title('  CCCC Plate ');
```

```
xlabel('Frequency (rad/sec)');
ylabel('Transmission Loss (dB)');
hold on
%MASS LAW
density=8000;
thickness=0.05;
mass=density*thickness;
frequency=[0:0.01:200];
cons=2*343*1.21,
tl=20*log10(mass*frequency/cons)
plot(frequency,tl,'color',[0,0,0])
```

```
% MESH GENERATION PROGRAM FOR 2D BEAM IN PLANE BENDING
function[x,conn,numcell,numq] = mesh(length,height,ndivl,ndivw)
numcell= ndivw*ndivl;
numq = (ndivl+1)*(ndivw+1);
% SET UP NODAL COORDINATES
for i = 1:(ndivl+1)
for j = 1:(ndivw+1)
x(1,((ndivw+1)*(i-1) +j))= (length/ndivl)*(i-1);
x(2,((ndivw+1)*(i-1) +j))= -(height/ndivw)*(j-1)+height/2;
%plot(x(1,((ndivw+1)*(i-1) +j)),x(2,((ndivw+1)*(i-1) +j)),'o')
hold on
xs(j)=x(1,((ndivw+1)*(i-1) +j));
ys(j)=x(2,((ndivw+1)*(i-1) +j));
end
hold on
%plot(xs,ys)
end
for j = 1:(ndivw+1)
for i = 1:(ndivl+1)
xplot(1,((ndivw+1)*(i-1) +j))= (length/ndivl)*(i-1);
xplot(2,((ndivw+1)*(i-1) +j))= -(height/ndivw)*(j-1)+height/2;
xplots(i)=x(1,((ndivw+1)*(i-1) +j));
yplots(i)=x(2,((ndivw+1)*(i-1) +j));
end
hold on
%plot(xplots,yplots)
end
% SET UP CONNECTIVITY ARRAY
for j = 1:ndivl
for i = 1:ndivw
elemn = (j-1)*ndivw + i;
nodet(elemn,1) = elemn + (j-1);
nodet(elemn,2) = nodet(elemn,1) + 1;
nodet(elemn,3) = nodet(elemn,2)+ndivw+1;
nodet(elemn,4) = nodet(elemn,3)-1;
end
end
conn = nodet';
```

```
% This function returns a matrix with 4 gauss points and their weights
% in natural coordinates
```

```
function v = gauss2(k)
if k==4
   fprintf('THE NUMBER OF GAUSS POINTS')
k
v(1,1) =-.86113631159405257524;
v(1,2) =-.33998104358485626480З;
v(1,3) = -v(1,2);
v(1,4) = -v(1,1);
v(2,1) =.34785484513745385737З;
v(2,2) =.652145154862546142627;
v(2,3) = v(2,2);
v(2,4) = v(2,1);
elseif k==2
   fprintf('THE NUMBER OF GAUSS POINTS')
k
v(1,1) =-1/sqrt(3);
v(1,2) =1/sqrt(3);
v(2,1) =1;
v(2,2) =1;
elseif k==6
   fprintf('THE NUMBER OF GAUSS POINTS')
k
v(1,1) =-.93246951420315202781230 2;
v(1,2) =-.661209386466264513661400;
v(1,3) =-.238619186083196908630502;
v(1,4) = -v(1,3);
v(1,5) = -v(1,2);
v(1,6) = -v(1,1);
v(2,1) =.171324492379170345040З0;
v(2,2) =.360761573048138607569 83;
v(2,3) =.467913934572691047389 87;
v(2,4) = v(2,3);
v(2,5) = v(2,2);
v(2,6) = v(2,1);
elseif k==9
   fprintf('THE NUMBER OF GAUSS POINTS')
k
v(1,1) =-.968160239507626089835576;
v(1,2) =-.836031107326635794299430;
v(1,3) =-.613371432700590397308702;
v(1,4) =-.324253423403808929038538;
v(1,5) =0;
v(1,6) = -v(1,4);
v(1,7) = -v(1,3);
v(1,8) = -v(1,2);
v(1,9) = -v(1,1);
v(2,1) =.0812743883615744119718 9;
v(2,2) =.180648160694857404058 47;
v(2,3) =.260610696402935462318 74;
v(2,4) =.312347077040002840068 63;
v(2,5) =.330239355001259763164 53;
v(2,6) = v(2,4);
v(2,7) = v(2,3);
v(2,8) = v(2,2);
v(2,9) = v(2,1);
else
   fprintf('THE NUMBER OF GAUSS POINTS')
```

```
k
v(1,1) =0;
v(2,1) =2;
end


%MAP GAUSS POINTS FROM NATURAL COORDINATES TO PHYSICAL COORDINATES
function [gs] = mapgauss(xc,conn,gauss,numcell)
index=0;
one = ones(1,4);
psiJ = [-1,+1,+1,-1]; etaJ = [-1,-1,+1,+1];
l = size(gauss);
l = l(2);
for e=1:numcell
% DETERMINE NODES IN EACH CELL
for j = 1:4
   je=conn(j,e);xe(j)=xc(1,je);ye(j)=xc(2,je);
   xe;
   ye;
end
for i=1:l
for j=1:l
index = index+1;
eta=gauss(1,i);psi=gauss(1,j);
%eta
%psi
N = .25*(one+psi*psiJ).*(one+eta*etaJ);
NJpsi=.25*psiJ.*(one+eta*etaJ);
NJeta=.25*etaJ.*(one+psi*psiJ);
xpsi=NJpsi*xe';ypsi=NJpsi*ye';xeta=NJeta*xe';yeta=NJeta*ye';
jcob=xpsi*yeta-xeta*ypsi;
xq = N*xe';yq = N*ye';
gs(1,index) = xq;
gs(2,index) = yq;
gs(3,index) = gauss(2,i)*gauss(2,j);
gs(4,index) = jcob;
%plot(xq,yq,'*')
hold on
end
end
%clear xe  % yy
%clear ye  % yy
end


%MOVING LEAST SQUARE METHOD TO CONSTRUCT SHAPE FUNCTION
function [phi,dphix,dphiy,d2phix,d2phiy,d2phixy] =
shape(gpos,gposquare,gposxy,dmax,x,xsquare,xy,numnod,dm)
won = ones(1,numnod);
%p = [won;x;xsquare;xy];
p = [won;x;xy;xsquare];
dif = gpos*won-x;
t = dm/dmax;
[w,dwx,dwy,dwdx,dwdy,d2wdx,d2wdy] = wgt(dif,t,x,dmax,dm);
B = p.*[w;w;w;w;w;w];
pp = zeros(6);
A = zeros(6);
```

```
dAx = zeros(6);
d2Ax = zeros(6);
dAy = zeros(6);
d2Ay = zeros(6);
d2Axy = zeros(6);
for i=1:numnod
pp = p(1:6,i)*p(1:6,i)';
A = A+w(1,i)*pp;
dAx = dAx+dwdx(1,i)*pp;
d2Ax = d2Ax+d2wdx(1,i)*pp;
dAy = dAy+dwdy(1,i)*pp;
d2Ay = d2Ay+d2wdy(1,i)*pp;
d2Axy =d2Axy+dwx(1,i)*dwy(1,i)*pp;
end
Ainv=inv(A);
%pg=[1 gpos' gposquare' gposxy'];
pg=[1 gpos' gposxy' gposquare' ];
phi=pg*Ainv*B;
dBx=p.*[dwdx;dwdx;dwdx;dwdx;dwdx;dwdx];
d2Bx=p.*[d2wdx;d2wdx;d2wdx;d2wdx;d2wdx;d2wdx];
dBy=p.*[dwdy;dwdy;dwdy;dwdy;dwdy;dwdy];
d2By=p.*[d2wdy;d2wdy;d2wdy;d2wdy;d2wdy;d2wdy];
d2Bxy=p.*[dwx.*dwy;dwx.*dwy;dwx.*dwy;dwx.*dwy;dwx.*dwy;dwx.*dwy];
consx=-Ainv*(dAx*Ainv);
consy=-Ainv*(dAy*Ainv);
%
%FIRST DERIVATIVE OF PHI
%
dphix = [0 1 0 gpos(2) 2*gpos(1) 0 ]*Ainv*B+pg*(consx*B+Ainv*dBx);
dphiy = [0 0 1 gpos(1) 0 2*gpos(2) ]*Ainv*B+pg*(consy*B+Ainv*dBy);
%
%SECOND DERIVATIVE OF PHI
%
gama0=Ainv*pg';
%x second derivative
gama1x=Ainv*([0,1,0,gpos(2),2*gpos(1),0]'-dAx*Ainv*pg');
gama2x=Ainv*([0,0,0,0,2,0]'-2*dAx*gama1x-d2Ax*gama0);
d2phix=(gama2x')*B+2*(gama1x')*dBx+(gama0')*d2Bx;
%
%y second derivative
%
gama1y=Ainv*([0,0,1,gpos(1),0,2*gpos(2)]'-dAy*Ainv*pg');
gama2y=Ainv*([0,0,0,0,0,2]'-2*dAy*gama1y-d2Ay*gama0);
d2phiy=(gama2y')*B+2*(gama1y')*dBy+(gama0')*d2By;
%
%x,y
%
gamaxy=Ainv*([0 0 0 1 0 0]'-dAx*gama1y-dAy*gama1x-d2Axy*gama0);
d2phixy=gamaxy'*B+gama1x'*dBy+gama1y'*dBx+gama0'*d2Bxy;



%WEIGHT FUNCTION
function [w,dwxt,dwyt,dwdx,dwdy,d2wdx,d2wdy] = wgt(dif,t,v,dmax,dm)
l = length(v);
for i=1:l
```

```
drdx = sign(dif(1,i))/dm(1,i);
drdy = sign(dif(2,i))/dm(2,i);
rx = abs(dif(1,i))/dm(1,i);
ry = abs(dif(2,i))/dm(2,i);
if 1>rx>0.5
%wx = (4/3)-4*rx+4*rx*rx -(4/3)*rx^3;
%dwx = (-4 + 8*rx-4*rx^2)*drdx;
%d2wx=(8-8*rx)*drdx*drdx;
wx= 1- 6*rx^2 + 8*rx^3-3*rx^4;
dwx= (-12*rx + 24*rx^2-12*rx^3)*drdx;
d2wx=(-12+48*rx-36*rx^2)*drdx*drdx;
elseif rx<=0.5
%wx = (2/3) - 4*rx*rx + 4*rx^3;
%dwx = (-8*rx + 12*rx^2)*drdx;
%d2wx=(-8+24*rx)*drdx*drdx;
wx = 1- 6*rx^2 + 8*rx^3-3*rx^4;
dwx = (-12*rx + 24*rx^2-12*rx^3)*drdx;
d2wx=(-12+48*rx-36*rx^2)*drdx*drdx;
else
wx=0;
dwx=0;
d2wx=0;
end
if 1>ry>0.5
%wy = (4/3)-4*ry+4*ry*ry -(4/3)*ry^3;
%dwy = (-4 + 8*ry-4*ry^2)*drdy;
%d2wy=(8-8*ry)*drdy*drdy;
wy = 1- 6*ry^2 + 8*ry^3-3*ry^4;
dwy = (-12*ry + 24*ry^2-12*ry^3)*drdy;
d2wy=(-12+48*ry-36*ry^2)*drdy*drdy;
elseif ry<=0.5
%wy = (2/3) - 4*ry*ry + 4*ry^3;
%dwy = (-8*ry + 12*ry^2)*drdy;
%d2wy=(-8+24*ry)*drdy*drdy;
wy = 1- 6*ry^2 + 8*ry^3-3*ry^4;
dwy = (-12*ry + 24*ry^2-12*ry^3)*drdy;
d2wy=(-12+48*ry-36*ry^2)*drdy*drdy;
else
wy=0;
dwy=0;
d2wy=0;
end
w(i) = wx*wy;
dwxt(i)=dwx;
dwyt(i)=dwy;
dwdx(i) = wy*dwx;
dwdy(i) = wx*dwy;
d2wdx(i)= wy*d2wx;
d2wdy(i)= wx*d2wy;
end


%MAP GAUSS POINTS ON BOUNDARY FROM NATURAL COORDINATES TO PHYSICAL
COORDINATES
function [yqb]=mapgptbdry(nt,j,quadpt,direction)
ncg(1)=-0.861136311594052;
ncg(2)=-0.339981043584856;
ncg(3)=0.339981043584856;
ncg(4)=0.861136311594052;
nt;
index=0;
one=ones(1,2);
ksai=[-1,1];
yqb = zeros(1,quadpt);
for i=1:quadpt
index=index+1;
ksaig=ncg(i);
N=0.5*(one+ksaig*ksai);
jacobi=(nt(direction,j+1)-nt(direction,j))/2;
two=[nt(direction,j),nt(direction,j+1)];
yqb(i)=N*two';
end

%SORT
function [index]=indexoflargest(a,size)
indexsofar=1;
for currentindex =2:size
   if(a(currentindex)>a(indexsofar))
  indexsofar=currentindex;
   end
end
index=indexsofar;
```

## E. Rayleigh-Ritz method for cantilever beam

```
%Rayleigh-Ritz method
%w=a1*x^2+a2*x^3+......+a10*x^11+   an*x^n+1
%Cantilever beam one dimensional
clear all
fprintf('Rayleigh-Ritz method terms')
n=3
for i=1:n
   for j=1:n
      m(i,j)=1/(i+j+3)
      k(i,j)=(i+1)*i*(j+1)*j/(i+j-1)
     end
  end

  [v,d] = eig(k,m);
  omiga=sqrt(d)
```

## F. Rayleigh-Ritz method (orthogonal polynomial) for fully clamped plate

```
%Rayleigh-Ritz method   02/09/2002
%plate free vibration bending
clear all
fprintf('Rayleigh-Ritz method terms')
fprintf('x direction')
m=2
fprintf('y direction')
n=2
%shape function
%w=(a1*x^2+a2*x^3+......+a10*x^11+
...+am*x^n+1)*(b1*x^2+b2*x^3+......+b10*x^11+ ...+bn*x^n+1)
% a   lengh of plate
% b   width of plate
a=1
b=1
nu=0.3
% k1 k2 k3 k4 four part of k
% k1p 6x6   part of k1
% k2p 6x6   part of k2
% k3p 6x6   part of k3
% k4p 6x6   part of k4
% Form k matrix
for ii=1:m*n
   jj=1
%
    if ii == 1
      l=1;
      k=1;
   elseif mod(ii-1,n)== 0
      l=l+1
      k=1
   else
      l=1
      k=k+1
   end
%
    for i=1:m
       for j=1:n
k1p(i,j)=(l+1)*l*(i+1)*i/(l+i-1)/(k+j+3)*(a^(l+i-
1))*(b^(k+j+3))+(l+1)*l*(j+1)*j/(l+i+1)/(k+j+1)*(a^(l+i+1))*(b^(k+j+1))+
(k+1)*k*(i+1)*i/(l+i+1)/(k+j+1)*(a^(l+i+1))*(b^(k+j+1))+(k+1)*k*(j+1)*j/
(l+i+3)/(k+j-1)*(a^(l+i+3))*(b^(k+j-1))

k2p(i,j)=(l+1)*(k+1)*(i+1)*(j+1)/(l+i+1)/(k+j+1)*(a^(l+i+1))*(b^(k+j+1))

k3p(i,j)=(l+1)*l*(j+1)*j/(l+i+1)/(k+j+1)*(a^(l+i+1))*(b^(k+j+1))

k4p(i,j)=(k+1)*k*(i+1)*i/(l+i+1)/(k+j+1)*(a^(l+i+1))*(b^(k+j+1))

k1(ii,jj)=k1p(i,j)
k2(ii,jj)=k2p(i,j)
k3(ii,jj)=k3p(i,j)
k4(ii,jj)=k4p(i,j)
```

```
         jj=jj+1

       end
    end

end
kmat=2*k1+4*(1-nu)*k2-2*(1-nu)*k3-2*(1-nu)*k4

% Form m matrix
for ii=1:m*n
   jj=1;
%
   if ii == 1
      l=1;
      k=1;
   elseif mod(ii-1,n)== 0
      l=l+1
      k=1
   else
      l=l
      k=k+1
   end
 %
    for i=1:m
       for j=1:n
          mp(i,j)=(1/(l+i+3))*(1/(k+j+3))*(a^(l+i+3))*(b^(k+j+3))
          mmat(ii,jj)=2*mp(i,j)
          jj=jj+1

       end
    end

end

 [v,d] = eig(kmat,mmat)
  omiga=sqrt(d)
```

## G. Collocation method for cantilever beam

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%collocation method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% one dimensional cantilever beam
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%satisfy:
%
%(1).The trial function is chosen to satisfy the boundary conditions.
%
%(2).The parameters are then found by forcing phi to satisfy the
differential
%
%    equation at a given set of n points.
%
%(3).At these points the residual vanishes.
%
%06/10/2002
%
%Form K matrix
%
%phi4x is four derivatives of deflection shape function
% n the number of nodes in cantilever beam not include boundary points.
clear all
n=20
for i=1:n
for   j=1:n
   phi4x(i,j) =(i+2+1)*(i+2)*(i+1)*i*(j/(n+1)).^(i-1);
   k(j,i)=phi4x(i,j);
end
end
%
%Form M matrix
%
%phi is deflection shape function
for i=1:n
for   j=1:n
   phi(i,j)=(((i+2+1)*(i+2)*(i+1)-(i+2+1)*(i+2))/2)*(j/(n+1)).^2-
((i+2+1)*(i+2)*(i+1)/6)*(j/(n+1)).^3+(j/(n+1)).^(i+2+1);
   m(j,i)=phi(i,j);
end
end
%
% Solve natural Frequency
%
[v d]=eig(k,m);
omiga=sqrt(d);
diag(omiga)
```

## H. Collocation method for fully clamped plate

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%collocation method
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Two dimensional fixed-fixed-fixed-fixed plate
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%satisfy:
%
%(1).The trial function is chosen to satisfy the boundary conditions.
%
%(2).The parameters are then found by forcing phi to satisfy the
differential
%
%    equation at a given set of n points.
%
%(3).At these points the residual vanishes.
%
%06/10/2002-07/10/2002
%
%Form K  and M matrix
%
clear all
nx=4;
ny=4;
%yk=0;
r=1;
c=1;
for l=1:nx
   for k=1:ny
      clear k1
      clear k2
      clear k3
      clear ksum
      for i=1:nx
         for j=1:ny
%            yk=yk+1;
%
%k1 4th derivatives respect to x
k1(i,j)=((l+3)*(l+2)*(l+1)*l*(i/(nx+1)).^(l-1))*(k*(j/(ny+1)).^2-
(k+1)*(j/(ny+1)).^3+(j/(ny+1)).^(k+3));
%
%k2 4th derivatives respect to y
k2(i,j)=((k+3)*(k+2)*(k+1)*k*(j/(ny+1)).^(k-1))*(l*(i/(nx+1)).^2-
(l+1)*(i/(nx+1)).^3+(i/(nx+1)).^(l+3));
%
%k3 4th derivatives respect to x and y
k3(i,j)=2*(2*l-6*(l+1)*(i/(nx+1))+(l+3)*(l+2)*(i/(nx+1)).^(l+1))*(2*k-
6*(k+1)*(j/(ny+1))+(k+3)*(k+2)*(j/(ny+1)).^(k+1));
%
ksum(i,j)=k1(i,j)+k2(i,j)+k3(i,j);
```

```
m(i,j)=(l*(i/(nx+1)).^2-
(l+1)*(i/(nx+1)).^3+(i/(nx+1)).^(l+3))*(k*(j/(ny+1)).^2-
(k+1)*(j/(ny+1)).^3+(j/(ny+1)).^(k+3));

%put ksum into kmat. ksum(2x2) is one line of kmat(4x4).
kmat(r,c)=ksum(i,j);
%put m into mmat. m(2x2) is one line of mmat(4x4)
mmat(r,c)=m(i,j);
c=c+1;
          end % j  y coordinates
       end % i x coordinates
c=1;
r=r+1;
   end % k  psai
end % l  phi
%
% Solve natural Frequency
%
[v d]=eig(kmat,mmat);
omiga=sqrt(d);
sort(diag(omiga))
```

## I. Matlab code for cantilever beam free vibration by Galerkin method

```
%Galerkin method
%w=a1*x^2+a2*x^3+......+a10*x^11+   an*x^n+1
fprintf('galerkin method terms')
clear all
n=3
for i=3:n
   for j=3:n
      k(i-2,j-2)=(i+1)*(i-1)*(i-2)*(j+1)*j*(j-2)/2-i*(i-1)*(i-
2)*(j+1)*j*(j-1)/6+(i+1)*i*(i-1)*(i-2)/(i+j-1)

      m(i-2,j-2)=((i+1)*i*(i-2)/2)*((j+1)*j*(j-2)/10-(j+1)*j*(j-
1)/36+1/(j+4))+(-(i+1)*i*(i-1)/6)*((j+1)*j*(j-2)/12-(j+1)*j*(j-
1)/42+1/(j+5))+(j+1)*j*(j-2)/(2*(i+4))-(j+1)*j*(j-1)/(6*(i+5))+1/(i+j+3)

   end
end

[v,d] = eig(k,m)
omiga=sqrt(d)
for i=3:n
   a1coefficient(i-2)=(i+1)*i*(i-2)/2
end

for i=3:n
   a2coefficient(i-2)=-(i+1)*i*(i-1)/6
end
a1=(v)'*(a1coefficient)'
a2=(v)'*(a2coefficient)'

for i=0:0.1:1
   x(i*10+1)=i
   y(i*10+1)=6*x(i*10+1)^2-4*x(i*10+1)^3+x(i*10+1)^4
      end
plot(x,y)
title('Galerkin Method 3 terms')
xlabel('Point')
ylabel('Deflection W')
text(0.1,2,['6*x(i*10+1)^2-4*x(i*10+1)^3+x(i*10+1)^4'])
```

## J. Matlab code for cantilever beam free vibration by FEM method

```
function main()
clear
% calculate eigenfreuencies of a thin cantilever beam
% composed of kmax beam elements
% input data
kmax=4; % number of elements
imax=2*kmax+2; % number of global dof's
L=1; % beam's length in [m]
l1=L/kmax; % element's length
ey=2.1e11; % Young's modulus in [N/m^2]
% a rectangular cross section of a beam
b=0.01; % thickness
h=0.1; % height
A=b*h; % cross sectional area in [m^2]
jp=h*h*h*b/12; % planar mom. of inertia in [m^4]
ro=7800; % density in [kg/m^3]
noncons=sqrt(ro*A*(L^4)/(ey*jp));
% assemble local stiffness matrix
[xke]=VBErigbeam(ey,l1,jp);
% assemble local mass matrix
[xme]=VBEmasbeam(ro,l1,A);
% assemble global matrices
[xk,xm]=VBEglobeam(kmax,imax,xke,xme);
xk
xm
kkk=xk(1:2:9,1:2:9)
mmm=xm(1:2:9,1:2:9)
% boundary conditions - the right hand side is clamped
% delete the last two rows and columns of global matrices
% pointer to deleted dof's
bound = [imax-1 imax];
xk(bound,:)=[]; xk(:,bound)=[];
xm(bound,:)=[]; xm(:,bound)=[];
xk
xm
%
% calculate eigenvalues by the finite element formulation
ei=eig(xk,xm); % eigenvalues
% sorted natural angular frequencies [rad/s]
ef=sort(real(sqrt(ei)));
ef*noncons
function [xme]=VBEmasbeam(ro,l1,A);
% assemble local mass matrix of a beam
% element with 4 dof's
% ro ... density
% l1 ... element's lenght
% A .... cross sectional area
xme=zeros(4);
l12=l1*l1;
konst=ro*l1*A/420;
%konst=l1/420;

xme(1,1)=156; xme(1,2)=22*l1; xme(1,3)=54; xme(1,4)=-13*l1;
xme(2,2)=4*l12; xme(2,3)=13*l1; xme(2,4)=-3*l12;
xme(3,3)=156; xme(3,4)=-22*l1;
xme(4,4)= 4*l12;
% symmetry
for i=1:4,
for j=i:4,
xme(j,i)=xme(i,j);
end;
end;
% constant multiplication
xme=konst*xme;
% end of VBEmasbeam.m

function [xke]=VBErigbeam(ey,l1,jp);
% assemble a local stiffness matrix
% of a beam element with 4 dof's
% ey ... Young's modulus

% l1 ... element's length
% jp ... planar moment of inertia of the cross section
xke=zeros(4);
l12=l1*l1; l13=l12*l1;
konst=2*ey*jp/l13;
%konst=2/l13;

xke(1,1)=6; xke(1,2)=3*l1; xke(1,3)=-6; xke(1,4)= 3*l1;
xke(2,2)=2*l12; xke(2,3)=-3*l1; xke(2,4)= l12;
xke(3,3)= 6; xke(3,4)=-3*l1;
xke(4,4)= 2*l12;
% symmetry
for i=1:4,
for j=i:4,
xke(j,i)=xke(i,j);
end;
end;
% constant multiplication
xke=konst*xke;

function [xk,xm]=VBEglobeam(kmax,imax,xke,xme);
% assemble global stiffness and mass matrices
% for a thin cantilever beam assembled of kmax
% identical beam elements with 4 DOF
% clear arrays
xk=zeros(imax); xm=zeros(imax);
% loop over elements
for k=1:kmax
% code numbers of k-the element
k2=2*k;
ic=[k2-1 k2 k2+1 k2+2];
% assembling
xm(ic,ic)=xm(ic,ic)+xme;
xk(ic,ic)=xk(ic,ic)+xke;
end
```

## K. Matlab code for rod axial vibration by FEM method

```
%axial vibration of rod
function main()
clear
% input data
kmax=20; % number of elements
imax=kmax+1; % number of global dof's
L=1; % rod's length in [m]
l1=L/kmax; % element's length
ey=2.1e11; % Young's modulus in [N/m^2]
% a rectangular cross section of a rod
b=0.01; % thickness
h=0.1; % height
A=b*h; % cross sectional area in [m^2]
ro=7800; % density in [kg/m^3]
noncons=sqrt(ro*(L^2)/(ey));
%
% assemble local stiffness matrix
[xke]=VBErigrod(ey,l1,A);
% assemble local mass matrix
[xme]=VBEmasrod(ro,l1,A);
% assemble global matrices
[xk,xm]=VBEglorod(kmax,imax,xke,xme);
xk;
xm;
%
% boundary conditions - the right hand side is clamped
%
% delete the last rows and columns of global matrices
% pointer to deleted dof's
bound = [1];
xk(bound,:)=[]; xk(:,bound)=[];
xm(bound,:)=[]; xm(:,bound)=[];
xk;
xm;
%
% calculate eigenvalues by the finite element formulation
[evector,evalue]=eig(xk,xm); % eigenvalues
% sorted natural angular frequencies [rad/s]
%ef=sort(real(sqrt(diag(evalue))));
ef=(real(sqrt(diag(evalue))));
ef*noncons
% plot mode shape
x=L/kmax:L/kmax:L;
def=-evector(:,19);
plot(x,def,'P','color',[0,0,0])

function [xme]=VBEmasrod(ro,l1,A);
% assemble local mass matrix of a rod
% ro ... density
% l1 ... element's lenght
% A .... cross sectional area
xme=zeros(2);
konst=ro*l1*A/6;
xme(1,1)=2;
xme(1,2)=1;
xme(2,1)=1;
xme(2,2)=2;
xme=konst*xme;
% end of VBEmasbeam.m

function [xke]=VBErigrod(ey,l1,A);
% assemble a local stiffness matrix
% of a rod element with 2 dof's
% ey ... Young's modulus
% l1 ... element's length
xke=zeros(2);
konst=ey*A/l1;
xke(1,1)=1;
xke(1,2)=-1;
xke(2,1)=-1;
xke(2,2)=1;
% constant multiplication
xke=konst*xke;
% end of VBErigbeam

function [xk,xm]=VBEglorod(kmax,imax,xke,xme);
% assemble global stiffness and mass matrices
% for a thin cantilever beam assembled of kmax
% clear arrays
xk=zeros(imax); xm=zeros(imax);
% loop over elements
for k=1:kmax
% code numbers of k-the element
k2=k;
ic=[k2 k2+1];
% assembling
xm(ic,ic)=xm(ic,ic)+xme;
xk(ic,ic)=xk(ic,ic)+xke;
end
% end of VBEglorod.m
```

## L. Matlab code for rod axial vibration mode shapes by analytical method

```
%Analytical rod mode shape
%first mode shape
L=1;
x=0:0.05:1;
%y=0.3*sin((pi/2)*x/L);
%plot(x,y,'P','color',[0,0,0]);
%second mode shape
y=0.3*sin((pi+pi/2)*x/L);
plot(x,y,'P','color',[0,0,0]);
```

## M. Matlab code for cantilever beam vibration mode shapes by analytical method

```
x=0:0.02:1
%First mode shape

%phi=0.1*[0 0.00139 0.00552 0.01231 0.02168 0.03355 0.04784 0.06449
0.08340 0.10452 0.12774 0.15301 0.18024 0.20936 0.24030 0.27297 0.30730
0.34322 0.38065 0.41952 0.45977 0.50131 0.54408 0.58800 0.63301 0.67905
0.72603 0.77392 0.82262 0.87209 0.92227 0.97309 1.02451 1.07646 1.12889
1.18175 1.23500 1.28859 1.34247 1.39660 1.45096 1.50549 1.56016 1.61496
1.66985 1.72480 1.77980 1.83483 1.88988 1.94494 2]
%plot(x,phi,'P','color',[0,0,0])

%Second mode shape

%phi=0.1*[0.00000 0.00853 0.03301 0.07174 0.12305 0.18526 0.25670
0.33573 0.42070 0.51002 0.60211 0.69544 0.78852 0.87992 0.96827 1.05227
1.13068 1.20236 1.26626 1.32141 1.36694 1.40209 1.42619 1.43871 1.43920
1.42733 1.40289 1.36578 1.31600 1.25365 1.17895 1.09222 0.99384 0.88431
0.76419 0.63410 0.49475 0.34687 0.19123 0.02865 -0.14007 -0.31409 -
0.49261 -0.67484 -0.86004 -1.04750 -1.23660 -1.42680 -1.61764 -1.80877 -
2.00000]
%plot(x,phi,'P','color',[0,0,0])

%Third mode shape

%phi=-0.1*[0.00000 0.02339 0.08839 0.18727 0.31238 0.45614 0.61120
0.77049 0.92728 1.07535 1.20901 1.32324 1.41376 1.47707 1.51056 1.51248
1.48203 1.41931 1.32534 1.20196 1.05185 0.87841 0.68568 0.47822 0.26103
0.03937 -0.18130 -0.39555 -0.59802 -0.78359 -0.94753 -1.08556 -1.19398 -
1.26974 -1.31055 -1.31485 -1.28189 -1.21172 -1.10515 -0.96375 -0.78975 -
0.58594 -0.35563 -0.10245 0.16974 0.45702 0.75558 1.06189 1.37287
1.68610 2.00000]
%plot(x,phi,'P','color',[0,0,0])

%Fourth mode shape

%phi=-0.1*[0.00000 0.04482 0.16510 0.33974 0.54801 0.77002 0.98714
1.18256 1.34177 1.45299 1.50758 1.50027 1.42928 1.29634 1.10648 0.86774
0.59073 0.28808 -0.02621 -0.33748 -0.63112 -0.89330 -1.11166 -1.27592 -
1.37836 -1.41424 -1.38199 -1.28366 -1.12327 -0.90964 -0.65299 -0.36594 -
0.06264 0.24191 0.53258 0.79478 1.01518 1.18266 1.28688 1.32262 1.28608
1.17687 0.99762 0.75368 0.45270 0.10407 -0.28179 -0.69420 -1.12317 -
1.56035 -2.00000]
%plot(x,phi,'P','color',[0,0,0])


%Fifth mode shape

phi=0.1*[0.00000 0.07241 0.25958 0.51697 0.80177 1.07449 1.30078 1.45309
1.51209 1.46767 1.31925 1.07553 0.75353 0.37706 -0.02529 -0.42257 -
0.78399 -1.08140 -1.29126 -1.39826 -1.39310 -1.27670 -1.05846 -0.75579 -
0.39278 0.00170 0.39632 0.75976 1.06317 1.28253 1.40051 1.40786 1.30418
1.09793 0.80582 0.45146 0.06355 -0.32634 -0.68626 -0.98631 -1.20090 -
1.31066 -1.30378 -1.17672 -0.93411 -0.58801 -0.15633 0.33937 0.67658
1.43502 2.00000]
plot(x,phi,'P','color',[0,0,0])
```

## N. Matlab code for plotting Gaussian weight function

```
% Plot weight function (Gaussian Equation 3.52)
x = [0.0:0.05:1.0];     % 21 NODES
nnodes = length(x);
ncells = nnodes-1;
dmax=10;
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
gg = zeros(1,ncells);
gg = -.025:.05:0.975;        % 21 NODES
gg(1) = 0.0;
% LOOP OVER GAUSS POINTS
for j = 1:length(gg)
   xg = gg(j);
dif = xg*ones(1,nnodes)-x;
% SET UP WEIGHTS W AND DW FOR EACH NODE
clear w dw r
for i=1:nnodes
   drdx = sign(dif(i))/dm(i);
   r(i) = abs(dif(i))/dm(i);
if r<=1

   w(i) = 2.71828^(-r(i)*r(i)/0.16);
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
elseif r>1.0
w(i) = 0.0;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
xlabel('x')
ylabel('w')
title('Gaussian Equation (3.52)')
end
end
end
```

## O. Matlab code for plotting cubic spline weight function

```
% Plot weight function (Cubic spline 3.53)
x = [0.0:0.05:1.0];     % 21 NODES
nnodes = length(x);
ncells = nnodes-1;
dmax=10;
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
gg = zeros(1,ncells);
gg = -.025:.05:0.975;        % 21 NODES
gg(1) = 0.0;
% LOOP OVER GAUSS POINTS
for j = 1:length(gg)
   xg = gg(j);

% DETERMINE DISTANCE BETWEEN NODES AND GAUSS POINT
dif = xg*ones(1,nnodes)-x;

% SET UP WEIGHTS W AND DW FOR EACH NODE
clear w dw r
for i=1:nnodes
   drdx = sign(dif(i))/dm(i);
   r(i) = abs(dif(i))/dm(i);
if r<=0.5
w(i) = (2/3) - 4*r(i)*r(i) + 4*r(i)^3;
dw(i) = (-8*r(i) + 12*r(i)^2)*drdx;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
elseif r<=1.0 & r>0.5
w(i) = (4/3)-4*r(i)+4*r(i)*r(i) -(4/3)*r(i)^3;
dw(i) = (-4 + 8*r(i)-4*r(i)^2)*drdx;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
elseif r>1.0
w(i) = 0.0;
dw(i) = 0.0;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
xlabel('x')
ylabel('w')
title('Cubic Polynomial Equation (3.53)')

end

end

end
```

## P. Matlab code for plotting cubic polynomial weight function

```
% Plot weight function (Cubic polynomial 3.54)
x = [0.0:0.05:1.0];     % 21 NODES
nnodes = length(x);
ncells = nnodes-1;
dmax=10;
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
gg = zeros(1,ncells);
gg = -.025:.05:0.975;        % 21 NODES
gg(1) = 0.0;
% LOOP OVER GAUSS POINTS
for j = 1:length(gg)
   xg = gg(j);
% DETERMINE DISTANCE BETWEEN NODES AND GAUSS POINT
dif = xg*ones(1,nnodes)-x;
% SET UP WEIGHTS W AND DW FOR EACH NODE
clear w dw r
for i=1:nnodes
   drdx = sign(dif(i))/dm(i);
   r(i) = abs(dif(i))/dm(i);
if r<=1
w(i) = 1 - 3*r(i)*r(i) + 2*r(i)^3;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
elseif r>1.0
w(i) = 0.0;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
xlabel('x')
ylabel('w')
title('Cubic Polynomial Equation (3.54)')
end
end
end
```

## Q. Matlab code for plotting quartic polynomial weight function

```
% Plot weight function (Quartic polynomial 3.55)

x = [0.0:0.05:1.0];     % 21 NODES
nnodes = length(x);
ncells = nnodes-1;
dmax=10;
dm = dmax*(x(2)-x(1))*ones(1,nnodes);
gg = zeros(1,ncells);
gg = -.025:.05:0.975;        % 21 NODES
gg(1) = 0.0;
% LOOP OVER GAUSS POINTS
for j = 1:length(gg)
   xg = gg(j);

% DETERMINE DISTANCE BETWEEN NODES AND GAUSS POINT
dif = xg*ones(1,nnodes)-x;

% SET UP WEIGHTS W AND DW FOR EACH NODE
clear w dw r
for i=1:nnodes
   drdx = sign(dif(i))/dm(i);
   r(i) = abs(dif(i))/dm(i);
if r<=1
w(i) = 1 - 6*r(i)*r(i)  + 8*r(i)^3-3*r(i)^4;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
elseif r>1.0
w(i) = 0.0;
plot(-r(i),w(i),'o','color',[0,0,0])
hold on
plot(r(i),w(i),'o','color',[0,0,0])
hold on
xlabel('x')
ylabel('w')
title('Quartic Polynomial Equation (3.55)')

end

end

end
```