Winter 2008

Nonlinear Random Response of Large-Scale Sparse Finite Element Plate Bending Problems

Swati Chokshi
Old Dominion University

Recommended Citation

Chokshi, Swati. "Nonlinear Random Response of Large-Scale Sparse Finite Element Plate Bending Problems" (2008). Doctor of Philosophy (PhD), Dissertation, Civil & Environmental Engineering, Old Dominion University, DOI: 10.25777/nn8n-c574
https://digitalcommons.odu.edu/cee_etds/36

NONLINEAR RANDOM RESPONSE OF LARGE-SCALE SPARSE FINITE ELEMENT PLATE BENDING PROBLEMS

by

Swati Chokshi
B.E., August 1996, North Gujarat University, India
M.E., May 2002, Maharaja Sayajirao University, India

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

CIVIL ENGINEERING

OLD DOMINION UNIVERSITY
December 2008

Approved by:

Duc T. Nguyen (Director)

Zia Razzaq (Member)

Gene Hou (Member)

Laura Harrell (Member)

ABSTRACT

Nonlinear Random Response of
Large-Scale Sparse
Finite Element Plate Bending Problems

Swati Chokshi
Old Dominion University, 2008
Director: Dr. Duc T. Nguyen

Acoustic fatigue is one of the major design considerations for skin panels exposed to high levels of random pressure at subsonic/supersonic/hypersonic speeds. The nonlinear large deflection random response of the single-bay panels aerospace structures subjected to random excitations at various sound pressure levels (SPLs) is investigated. The nonlinear responses of plate analyses are limited to determine the root-mean-square displacement under uniformly distributed pressure random loads. Efficient computational technologies like sparse storage schemes and parallel computation are proposed and incorporated to solve large-scale, nonlinear large deflection random vibration problems for both types of loading cases: 1) synchronized in time and 2) unsynchronized and statistically uncorrelated in time. For the first time, large scale plate bending problems subjected to unsynchronized load are solved using parallel computing capabilities to account for computational burden due to the simulation of the unsynchronized random pressure fluctuations.

The main focus of the research work is placed upon computational issues involved in the nonlinear modal methodologies. A nonlinear FEM method in time domain is incorporated with the Monte Carlo simulation and sparse computational technologies, including the efficient sparse Subspace Eigen-solutions are presented and applied to accurately determine the random response with a refined, large finite element mesh for the first time. Sparse equation solver and sparse matrix operations embedded inside the subspace Eigen-solution algorithms are also exploited. The approach uses the von-Karman nonlinear strain-displacement relations and the classical plate theory. In the proposed methodologies, the solution for a small number (say less than 100) of lowest linear, sparse Eigen-pairs need to be solved for only once, in order to transform nonlinear large displacements from the conventional structural degree-of-freedom (dof) into the modal dof. Moreover, the linear and nonlinear matrices are stored using sparse storage schemes in order to save computational time and memory. In case of unsynchronized load case, the time history needs to be generated and also rescaled separately for each finite element. For problems with large mesh size, the numbers of elements are high and the generation of time histories makes the problem unsolvable (in terms of computational time and/or memory requirements) for all practical purposes. By implementing parallel processing techniques, large scale structural analysis problems are solved without resorting to the use of expensive computing equipment or incurring an inordinately high computational cost that leads to a feasible solution. The reduced and coupled nonlinear equations in modal dof are inexpensively solved by the familiar Runge Kutta numerical integration scheme. Accurate responses are ensured with modal convergence, mesh convergence, and time step studies. The obtained numerical results (for synchronized load case) have also been compared favorably with results obtained from commercialized F.E. code such as Abaqus. Small, medium and large-scale single bay panel models are used to validate and evaluate the numerical performance of the present formulation and its associated computer software.

ACKNOWLEDGMENTS

I would like to express gratitude and appreciation to my advisor Dr. Duc T. Nguyen for his invaluable guidance, encouragement and advice throughout the entire course of this study. I also want to extend my deepest thanks to my dissertation committee members Dr. Zia Razzaq, Dr. Gene Hou and Dr. Laura Harrell for their helpful suggestions on the dissertation research, and their patience on reviewing the draft.

Special thanks to Dr. Zia Razzaq and Dr. Chuh Mei for many priceless discussions, and allowing me to benefit from their in-depth knowledge in the area of structural engineering. I really appreciate the help of Amit Kumar and Ruben Igloria from Office of Computing and Communications Services (OCCS) department at Old Dominion University.

I am grateful to my parents whose support and encouragement have always been a source of inspiration for me. The best way to pay back their love is to succeed in my life.

Last but not least, I am out of words to thank my best friend and soul mate, Milind, for his deepest love, extremely important co-operation, support and encouragement. I feel lucky to have him in my life and without him it would have been nearer to impossible to finish this journey, especially with the responsibilities of two kids, Swara and Aarav. I always found him to ease me when I encountered difficulties and losses.

Swati Chokshi
October 2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$\{a\}, \{b\}$	generalized coordinates
A	element area
$[A]$	membrane stiffness matrix
a	element length
b	element width
$[B]$	coupling stiffness matrix
$[B_b]$	bending strain interpolation function matrix
$[B_m]$	in-plane strain interpolation function matrix
$[B_\theta]$	large deflection interpolation function matrix
$[C]$	interpolation function matrix
$[D]$	bending stiffness matrix
E_1, E_2	Young's modulus in material major and minor axis
f_c	upper cut-off frequency
$f_{\max}$	maximum frequency
G_{12}, G_{13}, G_{23}	shear modulus
G_p	cross-spectral density function
$[H]$	displacement function matrix
h	plate thickness
$[k], [K]$	element and system linear stiffness matrices
$[k_1], [K1]$	element and system first-order nonlinear stiffness matrices
$[k_2], [K2]$	element and system second-order nonlinear stiffness matrices
$[\overline{K}]$	modal linear stiffness matrix
$[K_q], [K_{qq}]$	modal nonlinear stiffness matrices
L	length of the element
$[m], [M]$	element and system mass matrices
$[\overline{M}]$	modal mass matrix

$\{M\}$	bending moment stress resultants
NPT	total time history points
$\{N\}$	inplane force stress resultants
$[n_1],[N_1]$	element and system first-order nonlinear incremental stiffness matrices
$[n_2],[N_2]$	element and system second-order nonlinear incremental stiffness matrices
p_0	reference pressure
$\{p\},\{P\}$	element and system force vector
P_{random}	random load intensity
$\{\overline{P}\}$	modal force vector
$[Q]$	lamina reduced stiffness matrix
$[\overline{Q}]$	transformed lamina reduced stiffness matrix
$\{q\}$	modal coordinate vector
S_0	spectral density
t	time
$[T_b],[T_m]$	transformation matrices
$[T_\varepsilon],[T_\sigma]$	strain and stress transformation matrices
U	strain energy
u, v	inplane displacements
W	work
$\{w\},\{W\}$	element and system nodal degree-of-freedom vectors
w	transverse displacement
x, y, z	Cartesian coordinates

GREEK SYMBOLS

$\{\varepsilon\}$	total strain vector
$\{\varepsilon^0\}$	in-plane strain vector
$\{\varepsilon_m^0\}$	membrane strain vector
$\{\varepsilon_\theta^0\}$	von-Karman strain vector
θ	fiber orientation angle
$\{\phi\}, \{\Phi\}$	element and system eigen vectors
$[\Phi]$	system eigenvector matrix
$\{\kappa\}$	bending curvature vector
$\nu, \nu_{12,} \nu_{21,}$	poisson's ratios
$[\theta]$	slope matrix
ρ	mass density
σ_{12}	stress in 1-2 direction
$\{\sigma\}$	stress vector
τ	shear stress
ω	frequency
ξ	structural modal damping ratio
δW	work done by the forces
ω	natural frequency
Δ	increase value
λ	eigen-value

SUBSCRIPTS

b	bending
cr	critical
ext	external
int	internal
m	membrane
max	maximum

θ	large deflection
u, v, w	in-plane and transverse displacements
ε	strain
σ	stress

SUPERSCRIPTS

(r)	rth normal mode of linear vibration problem
T	matrix and vector transpose
e	element level

CHAPTER I

INTRODUCTION

Efficient numerical procedures including modal method for solving large-scale, sparse, parallel, nonlinear large deflection random vibration problems are proposed. The solution for the small number (say less than 100) of lowest linear Eigen-pairs need to be solved for only once, in order to transform nonlinear large displacements from the conventional, large structural degree-of-freedom (dof) into the smaller modal dof. The reduced coupled nonlinear equations of motion in modal dof can be inexpensively solved by the popular Runge-Kutta (RK) or any other time integrating method. The main focus is placed upon computational issues involved in the nonlinear modal methodologies. Major time consuming portions of the nonlinear modal method are firstly identified. Then, efficient sparse and dense matrix technologies are proposed and incorporated into the developed procedures. Small, medium, and large-scale single panel models are used to validate and evaluate their numerical performance. Whenever possible, comparison in terms of numerical accuracy and computational time between the developed codes with existing solution including popular commercialized finite element software such as Abaqus is included. Results obtained to this date indicate that the developed algorithms and software are accurate and highly efficient.

A new spacecraft, Crew Exploration Vehicle, will be developed and eventually conduct the Space Exploration Mission. Recently, the NASA X-43A scramjet airplane made a successful flight with a new speed record of Mach 10. A new Air Force initiative, Hypersonic and Space Access Program, demands emerging technologies on effective hypersonic vehicle design. Acoustic fatigue is one of the major design considerations for skin panels exposed to high levels of random pressure and elevated temperature at subsonic/supersonic/hypersonic speeds. The severe flight environment leads to various loads[48] on the surface panel, including aerodynamic pressure, acoustic excitation, and thermal load as shown in Fig. 1.1.

The Journal model used for this work is the *AIAA Journal*

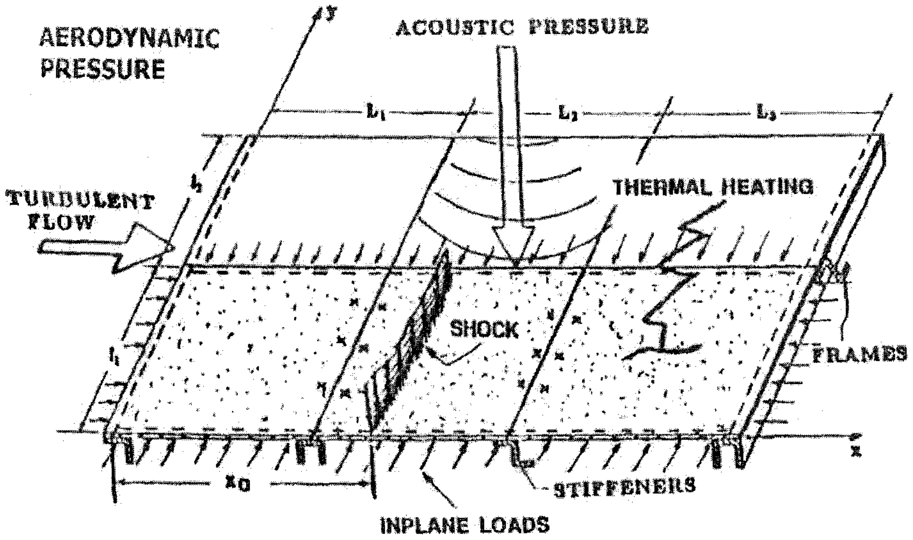


Fig. 1.1: Various loads on surface panels of supersonic/hypersonic flight vehicles[48]

Acoustic fatigue is a long-cycle fatigue failure induced by forced random vibration from outside sources such as jet engine noise, turbulent boundary layer pressure fluctuations, and unsteady aerodynamic forces due to flow separation. The sonic fatigue design guide[1] was based on semi-empirical data for isotropic metallic aircraft structures.

1.1 PROBLEM DEFINITION

The nonlinear large deflection random response of aerospace structures like single-bay panels subjected to random excitations at various sound pressure levels (SPLs) has been investigated. A nonlinear finite element modal (FEM) method incorporated with the Monte Carlo simulation and sparse computational techniques is presented and applied to determine the dynamic response accurately with a refined FE mesh for the first time. The proposed method is applicable to complex and highly efficient aerospace and civil structures by incorporating sparse computational techniques and parallel computation especially in case of unsynchronized loading.

1.2 ACOUSTIC FATIGUE TESTING

Acoustic fatigue tests have been conducted to determine the response and fatigue life of aircraft panels. One of the first tests in 1957[2] showed that a single aluminum panel subjected to acoustic pressure had a fatigue life of 840 sec at 154 dB, 180 sec at 160 dB, and 30-40 sec at 166 dB overall SPL. Both experimental and theoretical studies in 1968[3] indicated that the response of a single-bay panel was very much sensitive to the boundary conditions and small variations of spacing. The effect of stiffeners and stringers must be addressed. It was found in 1972[4] that the rear part of fuselage must be modeled with multi-bays in both stream-wise and span-wise directions under turbulent boundary layers. Fatigue tests on multi-bay composite aircraft structures were conducted by Holehouse[5] in 1980. A 3 x 3 bay panel (configuration “a” in ref.[5]) is shown here in Fig. 1.2. It was found that the measured RMS strains from a total of 20 panel configurations were all much lower than those of linear Nastran analysis results, indicating a highly nonlinear response of panels and un-synchronization of the random pressure in time. It would be excessively conservative if acoustic fatigue design was based on linear structural and random loads synchronized in time analysis. Other experimental work on multi-bay panels with stringers[6] in 1989 showed that the fatigue cracks often occurred in the stiffeners or areas close to them.

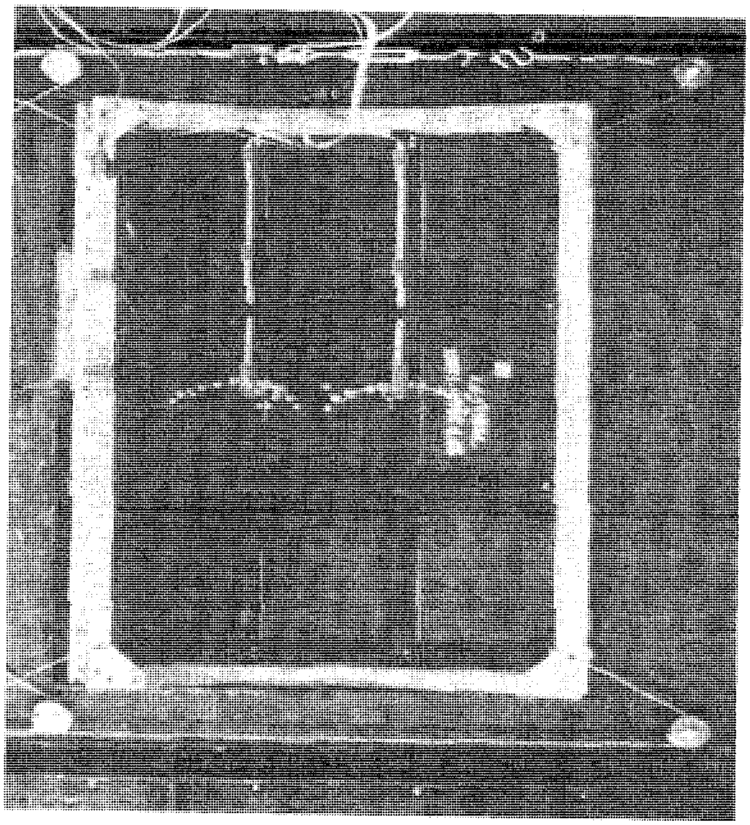


Fig. 1.2: A 3 x 3-bay panel[5]

1.3 ANALYSIS METHODS FOR ACOUSTIC FATIGUE

It is also well known that panel flutter occurs resulting from airflow acting on only one side of surface panel. There exists a critical non-dimensional dynamic pressure λ_{cr} as shown in Fig. 1.3.[52] Below λ_{cr} the panel undergoes linear random vibration with small amplitudes. The dominant frequency is observed near the lowest natural frequency of the panel. Neglecting structural nonlinearity, linear theory indicates that beyond λ_{cr} the panel motion becomes unstable and grows exponentially with time. Therefore, nonlinear effect must be considered in vibration analysis beyond λ_{cr}.

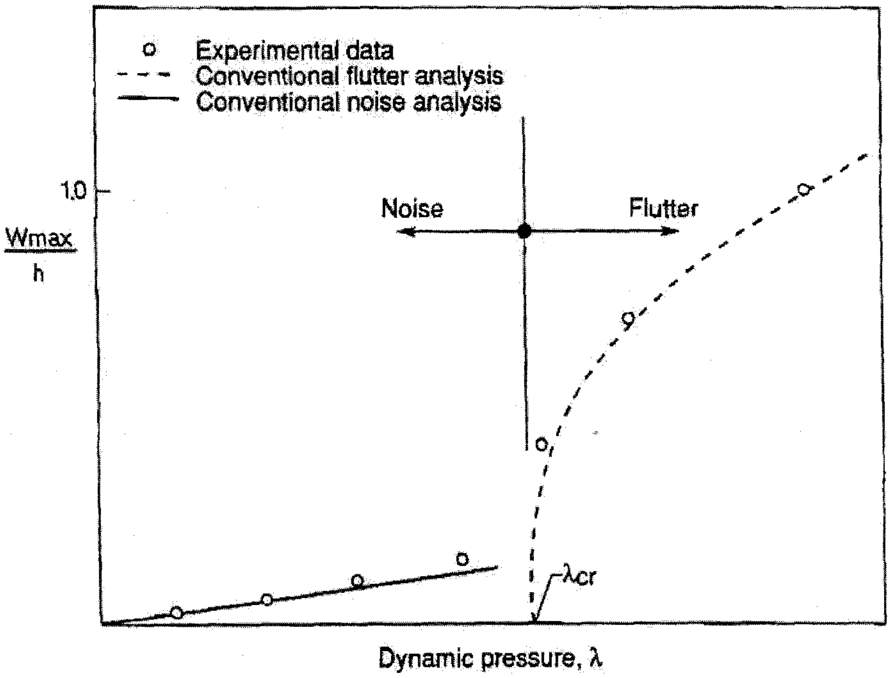


Fig. 1.3: Panel flutter: theory and experiment[52]

Accurate prediction of nonlinear random response of panels is critical for fatigue life estimation and design of aerospace structures. The Fokker-Plank-Kolmogorov (FPK) equation approaches[7] give exact solutions to a single-mode and some special 2-mode nonlinear systems subjected to white noise excitations.

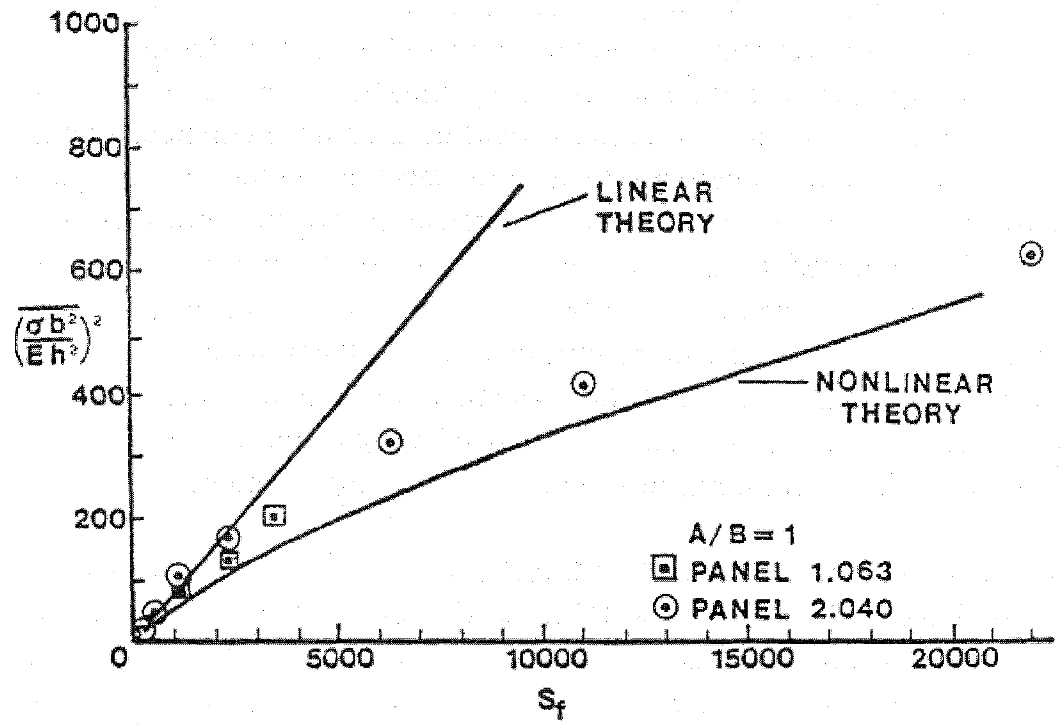


Fig. 1.4: Comparison of mean-square strains (Mean square center deflection Vs. Pressure spectral density for a clamped square plate) [8, 10]

The equivalent linearization (EL) technique incorporates PDE/Galerkin[8] or FEM[9] method to transform nonlinear ODE to a set of equivalent linear equations with the assumption that the response is Gaussian. Mei and Paul[8] compared analytical solutions with testing data[10] for two single-bay aluminum square panels as shown in Fig. 1.4. The nonlinear response of the aluminum panel is characterized by the broad peaks and frequency shifts in the power spectral density (PSD) plot as shown in Fig. 1.5.

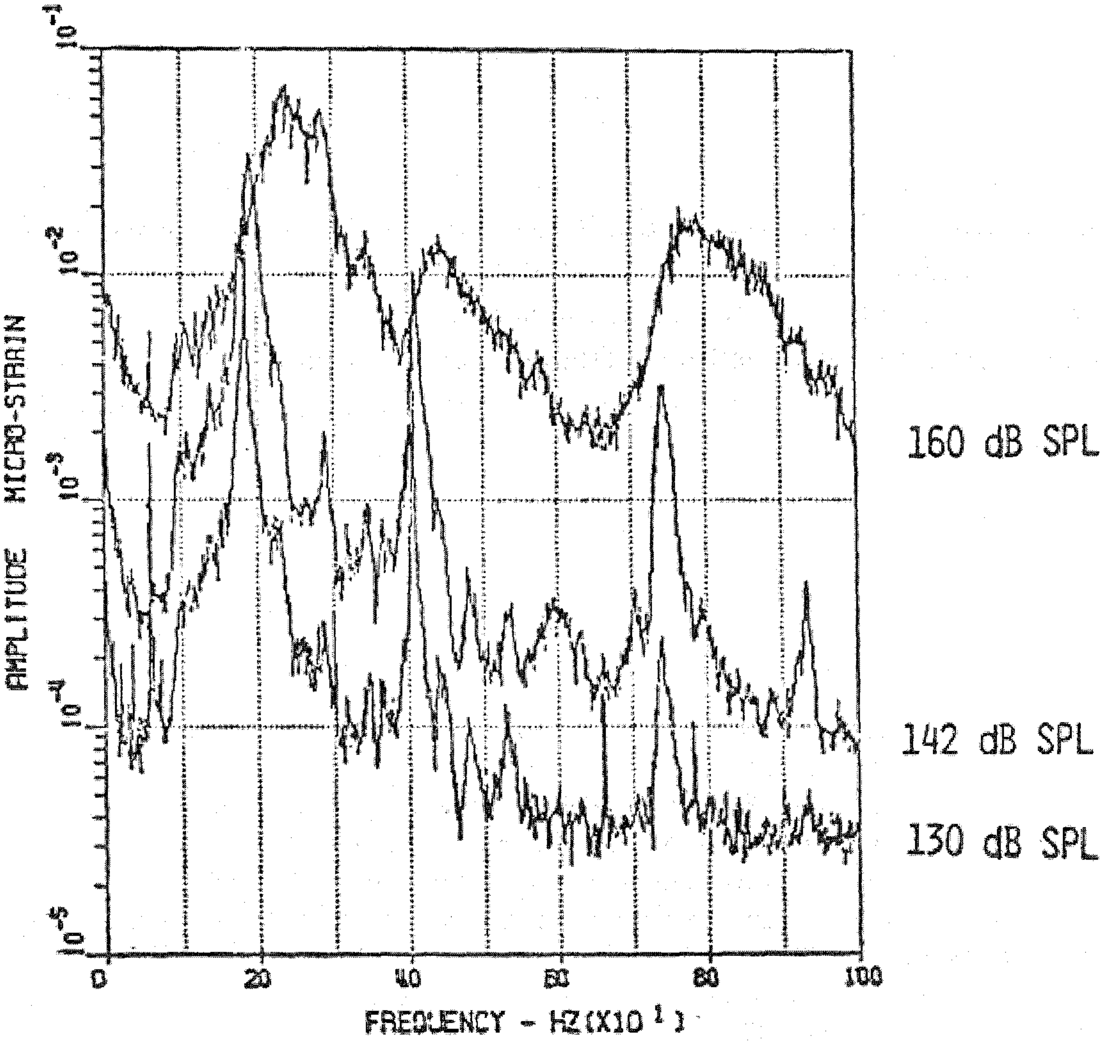


Fig. 1.5: Experiment strain PSD of a square aluminum panel at three overall SPL[10]

The Monte Carlo numerical simulation based on PDE/Galerkin[11] is a time domain method suitable for simple panel geometries and boundary conditions. Another time domain method is the Monte Carlo simulation based on the traditional finite element (FE) method. The computational cost is the main concern because the FE model often includes a very large number of structural dof, and the nonlinear terms have to be updated and reassembled at each time step of integration. Mei et al. [13] developed a FEM method to reduce the large number of FE structural dof to a very small number of modal dof. In another study, [14] the modal method is implemented using a regression analysis in which a series of nonlinear static test cases are used to identify the nonlinear modal model. Hollkamp et al. [15] verified recently that the two modal methods[13, 14] and the experiments agree very well for clamped beams in terms of strain PSD function.

Reviews on acoustic fatigue of aircraft and spacecraft structures were conducted by Clarkson[16] in 1994, and Mei and Wolfe[17] in 1986, respectively. The random response should be considered with the nonlinear large deflection effects, the influence of in-plane boundary conditions, and the appropriate analysis methods. The approaches to the estimation of damage accumulation and fatigue life were reviewed. They also gave the directions of future research and the factors to provide a reliable estimate of acoustic fatigue life of aerospace structures.

1.4 SINGLE-BAY AEROSPACE STRUCTURES

For the past twenty years researchers have been focusing on the random vibration and acoustic fatigue of isotropic/composite single panels under random loads synchronized in time. Nonlinear FEM methods in time domain have been developed to determine the time history of the random response.[13] The approach uses the von-Karman nonlinear strain-displacement relations and the laminated classical plate theory for fatigue life estimation. The nonlinear modal equations of motion are solved by Runge-Kutta numerical scheme to obtain maximum deflection. Monte Carlo simulation is adopted and the ensemble would take 10 or more samples. Accurate responses are ensured with modal convergence, mesh convergence, and time step studies.

Many of today's structures are subjected to excitations which are random in nature. Examples range all the way from aircraft and missile structures subjected to aero-elastic and aerodynamic loads to civil engineering structures like high rise buildings and bridges acted upon by earthquake and wind loads. In some cases, the response statistics of such structure will be strongly time dependent or non-stationary, but in many applications, the response may be considered stationary. In this study, only stationary excitations are considered. In stochastic structural dynamics, the majority of analyses have dealt with linear structures under stationary, Gaussian, and band-limited white noise excitations. Although these simplifying assumptions may be justified in many processes, experimental data have shown the non-stationary and non-Gaussian characteristics of the loads quite frequently.

1.5 ANALYTICAL APPROACHES FOR RANDOM VIBRATION ANALYSIS

In 1983, Crandall and Zhu[39] published a review article on the progress in random process and random fields, source of excitations, prediction of random responses, and reliability.

There are five major analysis methods for the prediction of nonlinear random response of the structural panel:

(1) Perturbation

(2) Fokker-Plank-Kolmogorov (FPK equation)

(3) Monte Carlo

(4) Equivalent linearization

(5) Finite element numerical Integration

The perturbation method[40] has been limited to very weak geometric nonlinear problems. So, it is not suitable for large nonlinear random vibration. The FPK method[7] can lead to exact solutions only for single degree of freedom systems. Heuer et al.[41] extended the application of the FPK approach to multi dof by utilizing a multi-modal projection method. They also investigated the nonlinear random vibration of thermally buckled skew plates. The probability of first occurrence of snap-through was determined. The implementation of the method is very tedious.

The equivalent linearization method is extensively used because of its ability to accurately capture the response statistics over a wide range of problems while maintaining a relatively low computational burden.[42, 43] Ng[44] presented a single-mode method for the analysis of snap-through. He divided the random response with the compressive load larger than the critical value into three regions: no snap-through, intermittent snap-through, and persistent snap-through. Lee[45] investigated the effects of thermal variation and thermal moment on the panel response. Locke and Mei,[46] and Mei and Chen[9] extended the finite element method to nonlinear random vibration analysis. The equivalent linearization method was adapted to the nonlinear finite element modal equations to determine RMS deflections and strains at different sound pressure levels. The application of the equivalent linearization method depends on the assumption of Gaussian distribution over the response. Thus it can not predict occurrence of snap-through since snap-through is non-Gaussian in nature.

Monte Carlo simulation[36, 37] was employed by Arnold and Vaicaitis[47], Vaicaitis[27], Vaicaitis and Kavallieratos[48] to study the nonlinear panel response and fatigue life subjected to acoustic excitation. The PDE/Galerkin method was employed and numerical integration was used to obtain time history of the panel response. Green and Killey[12] studied a similar problem but narrow-band acoustic loads were used and initial imperfections were also considered in the model. The PDE/Galerkin approach limits its applicability to rather simple structures.[27, 47]

The finite element numerical integration approach combines the finite element and Monte Carlo simulation method.[12, 49] The main disadvantage of the method is its computational cost, because the finite element model often includes hundreds, if not thousands, number of physical structural node dofs, and the nonlinear terms are updated and reassembled at each time step. Abdel-Motagaly et al.[50] used finite element numerical integration to study nonlinear panel response under combined aerodynamic and acoustic loads. Finite element system equations of motion were transferred to modal coordinates to reduce the large number of structural node dof. Dhainaut et al.[51] adopted the same approach, and studied the random response to the acoustic loads at elevated temperatures environment.

1.6 OBJECTIVE AND SCOPE

FEA code has been developed for nonlinear modal finite element analysis of the structures modeled using rectangular plate elements. The code is developed in Fortran language with the capability of linking with the Matlab environment. Developed FEA code is capable of solving large scale, synchronized as well as unsynchronized loading problems. By adding features like geometry transformation, various element types, different kind of loadings and accounting for composite materials, the code can be extended to solve generalized problems for analysis of large scale complex structures.

1.7 LIMITATIONS

Developed FEA code has the following limitations:

1. Code is applicable for nonlinear analysis of the structure modeled using rectangular plate elements only.
2. Structural material shall be isotropic only.
3. The plate element considered in this work is assumed to be aligned with the global coordinate axis. Thus, coordinate transformation from element to global axis is not necessary.
4. Simulated random load is considered truncated band limited white noise for the sound waves. Code is capable to perform nonlinear analysis for other types of random loads for which input load needs to be supplied.
5. Thermal loading has not been accounted in the code.
6. The environment should have access to Matlab in order to run the code with rescaling feature.

1.8 MOTIVATION AND DISSERTATION ORGANIZATION

The random excitations such as earthquake type motions, pressure waves of explosion, jet noise, and continuous atmospheric turbulence must take into consideration while designing structures like bridges, tall buildings that house nuclear reactors, and naval and aerospace structures, for the safety and reliability purpose. The investigation of dynamic response to random excitation started in 1905 with Einstein's pioneering study of Brownian motion. But it has acquired a special prominence with the advent of jet engines. In vibration analysis, the important task is simulation of random loads as close to real life cases as possible. This is because as far as random data is concerned, it can be obtained from various sources such as data recorded from an earthquake or acoustic load data measured from flight testing on an aircraft. But, it is extremely important to apply the random loads correctly to a structure in order to achieve results as close to experimental values as possible. All these concerns go into the design of structures under random loads. Because a structure's vibration characteristics determine how it responds to any type of dynamic load, modal analysis should always be performed first before trying any other dynamic analysis.

It was reported by Green and Killey[12] that only running a half-second time for nonlinear time domain Monte Carlo simulation of 5000-element for a single-bay panel took approximately 10 hours on a Cray C94 computer. To solve such problems with a large number of elements for longer time history, an efficient method must be used. Application of sparse technology combined with non linear modal finite element analysis makes it possible to solve unsolved problems because of the time and memory limitations.

Most real life cases are subjected to unsynchronized random loads, simple examples of which are long period of rain drops steadily falling on the roof top or about 30,000 marathon runners running on a suspension bridge. And as such it is important to study the behavior of structures to such loads. But given the complexity of the problem, it is detrimental to understand the fundamental aspects. In light of this fact, the problem is being studied and parallel computation has been involved along with sparse techniques to solve such problems with very large mesh size and/or with long random load time history within the limitation of time and/or memory.

The current commercial finite element codes, such as Nastran and ANSYS etc., could not study the linear random response under the unsynchronized loading case since they are solved in the frequency domain. To the best knowledge of the author there is no efficient analytical or numerical solution to the nonlinear response of plates, with large mesh size under unsynchronized random pressure loads available in the literature.

Following a discussion on the reasons leading to the motivation to study the panel response under synchronized and unsynchronized random loads and literature survey related to such problems, the flow of this thesis will include all the details about the solution procedure used. In Chapter 2, the finite element system governing equations are derived based upon von-Karman nonlinear strain-displacement relations and virtual work principle. In Chapter 3, the concepts and importance of un-synchronized load cases are discussed. Also, the generation of synchronized as well as un-synchronized random load is discussed along with the requirement of using parallel computation in case of unsynchronized load case. In Chapter 4, the solution procedures are developed. A modal equation of motion is derived in order to save time and memory. Runge-Kutta time integration scheme is employed for solving equation of motion. Random responses are characterized by Monte Carlo numerical simulation using a modal approach. Chapter 5 provides detailed description of sparse technology including storage scheme, Eigen-solution by Sub-space method, and equation solver usage to inverse the sparse matrix. Step-by-step sparse algorithm applied to the solution procedure is also addressed. Chapter 6 discusses basics of parallel computation and the reasons for usage of parallel computation in case of un-synchronized load case along with step-by--step solution procedure. In Chapter 7, numerical examples are presented with results and discussions for synchronized and un-synchronized dynamic pressure subjected. Finally, the concluding remarks and recommendation for future work are presented in Chapter 8.

CHAPTER II

FINITE ELEMENT FORMULATION

2.1 INTRODUCTION

Bogner-Fox-Schmit (BFS)[55] C^1 conforming rectangular element[59, 60] is adopted in the study. The finite element governing equations for random vibration to a BFS plate are derived in this chapter. The load subjected is assumed either to be a band-limited white or non-white Gaussian random pressure and uniformly distributed over the structural surface. The finite element formulation[56, 57, 64] is based on the von-Karman large deflection theory with the small strain assumption and the classic plate theory. The following assumptions are made throughout the derivation:

1. The panel is thin. Which means the length to thickness ratio, L/h > 40.
2. In-plane inertia, rotary inertia, and transverse shear deformation effects are negligible.
3. Von-Karman strain-displacement relations are valid.
4. Proportional damping $\xi_r\omega_r = \xi_s\omega_s$, is used. Where, coefficient ς_r is modal damping ratio for the rth mode and ω_r is the rth modal natural frequency.
5. Straight lines perpendicular to the mid surface before deformation remain straight and perpendicular after deformation.
6. The transverse normals do not experience elongation, i.e., they are inextensible.

2.2 ELEMENT DISPLACEMENT FUNCTIONS

In the derivation, C^1 conforming BFS rectangular plate elements are adopted. A C^1 conforming element provides inter-element continuity of the displacement field $w(x, y)$ in the z-direction, and its first derivatives $w_{,x}$ and $w_{,y}$ but it does not provide inter-element continuity of all second derivatives of $w(x, y)$.

As shown in Fig. 2.1, BFS rectangular plate element of length a, width b, and thickness h consists of four nodes and each node has 6 dof. Thus, each element has total 24 dof, which includes 16 bending dof, $\{w_b\}_{16\,\mathrm{X}\,1}$, and 8 in-plane dof, $\{w_m\}_{8\,\mathrm{X}\,1}$. They are expressed as:

$$\{w\} = \{\{w_b\} \quad \{w_m\}\}^{\mathrm{T}} \tag{2.1}$$

$$\{w_b\} = \{w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_{,x1} \quad w_{,x2} \quad w_{,x3} \quad w_{,x4} \quad w_{,y1} \quad w_{,y2} \quad w_{,y3} \quad w_{,y4} \quad w_{,xy1} \quad w_{,xy2} \quad w_{,xy3} \quad w_{,xy4}\}^{\mathrm{T}} \tag{2.2}$$

$$\{w_m\} = \{u_1 \quad u_2 \quad u_3 \quad u_4 \quad v_1 \quad v_2 \quad v_3 \quad v_4\}^{\mathrm{T}} \tag{2.3}$$

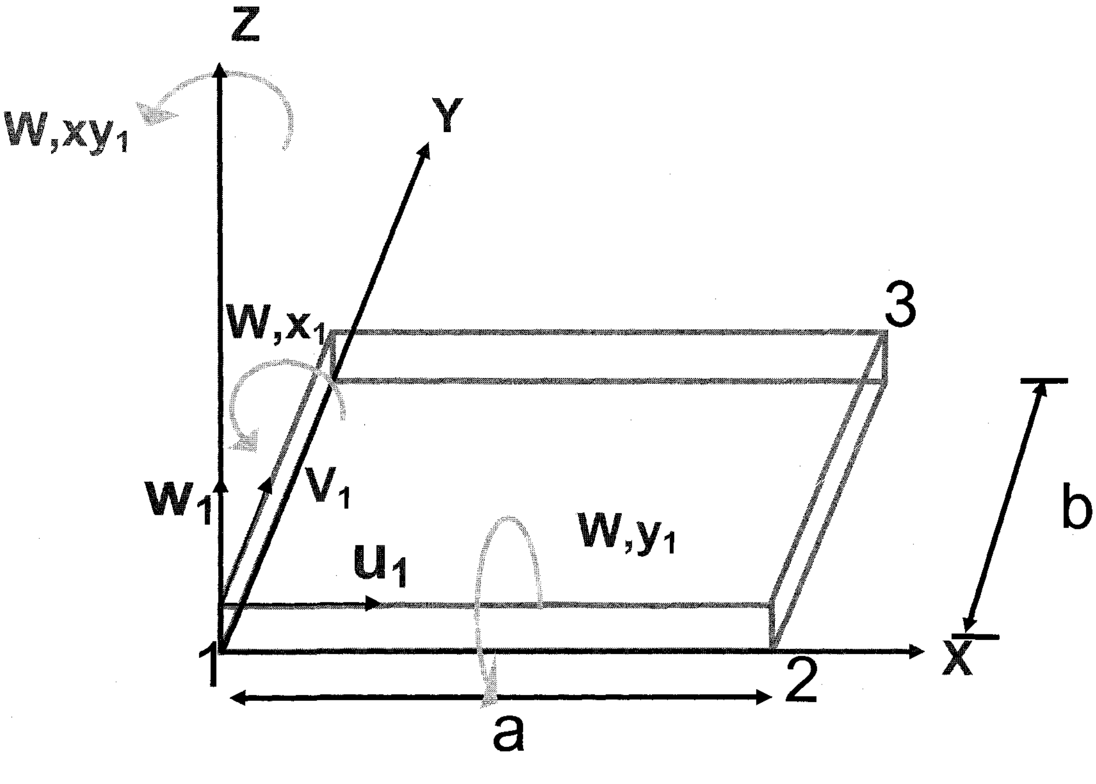


Fig. 2.1: Nodal degrees of freedom of a BFS C^1 conforming rectangular element

The element transverse displacement w and the in-plane displacements u and v can be approximated as a bi-cubic and bilinear polynomial function of x and y, which can be expressed[56, 62, 63] as:

$$w(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + a_8x^2y + a_9xy^2 \\ + a_{10}y^3 + a_{11}x^3y + a_{12}x^2y^2 + a_{13}xy^3 + a_{14}x^3y^2 + a_{15}x^2y^3 + a_{16}x^3y^3 \tag{2.4}$$

$$= [H_w(x,y)]\{a\} \tag{2.5}$$

where,

$$[H_w(x,y)] = \lfloor 1 \quad x \quad y \quad x^2 \quad xy \quad y^2 \quad x^3 \quad x^2y \quad xy^2 \quad y^3 \quad x^3y \quad x^2y^2 \quad xy^3 \quad x^3y^2 \quad x^2y^3 \quad x^3y^3 \rfloor \tag{2.6}$$

$$\{a\} = \{a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11} \quad a_{12} \quad a_{13} \quad a_{14} \quad a_{15} \quad a_{16}\} \tag{2.7}$$

and

$$u(x,y) = b_1 + b_2x + b_3y + b_4xy \tag{2.8}$$

$$= [H_u(x,y)]\{b\} \tag{2.9}$$

where,

$$[H_u(x,y)] = [1 \quad x \quad y \quad xy \quad 0 \quad 0 \quad 0 \quad 0] \tag{2.10}$$

$$\{b\} = \{b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8\}^T \tag{2.11}$$

$$v(x,y) = b_5 + b_6x + b_7y + b_8xy \tag{2.12}$$

$$= [H_v(x,y)]\{b\} \tag{2.13}$$

where,

$$[H_v(x,y)] = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad x \quad y \quad xy] \tag{2.14}$$

Here, $\{a\}$ and $\{b\}$ are called generalized coordinates and they are related to the nodal dof vectors by their transformation matrices as shown below:

$$\{a\} = [T_b]\{w_b\} \tag{2.15}$$

$$\{b\} = [T_m]\{w_m\} \tag{2.16}$$

The detailed derivation of bending and in-plane transformation matrices $[T_b]$ and $[T_m]$ is given in Appendix A. In terms of nodal displacement vectors, the element displacement functions can be expressed as:

$$\begin{aligned} w &= [H_w(x,y)]\{a\} \\ &= [H_w(x,y)][T_b]\{w_b\} \end{aligned} \tag{2.17}$$

$$\begin{aligned} u &= [H_u(x,y)]\{b\} \\ &= [H_u(x,y)][T_m]\{w_m\} \end{aligned} \tag{2.18}$$

$$\begin{aligned} v &= [H_v(x,y)]\{b\} \\ &= [H_v(x,y)][T_m]\{w_m\} \end{aligned} \tag{2.19}$$

2.3 NON-LINEAR STRAIN DISPLACEMENT RELATIONS

Using the Von-Karman large deformation strain-displacement relations, the total strain vector $\{\varepsilon\}$, in terms of in-plane strain and curvature, can be written as follows:

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \{\varepsilon^0\} + z\{\kappa\} \tag{2.20}$$

where in-plane strain vector, $\{\varepsilon^0\}$, consists of two components and can be written as:

$$\{\varepsilon^0\} = \{\varepsilon_m^0\} + \{\varepsilon_\theta^0\} \tag{2.21}$$

In the above equation,

$\{\varepsilon_m^0\}$ = In-plane strain vector

$\{\varepsilon_\theta^0\}$ = Non-linear von-Karman strain vector

where in-plane strain can be expressed in terms of finite element displacement functions as:

$$\{\varepsilon_m^0\} = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{Bmatrix} \tag{2.22}$$

The non-linear Von-Karman strain can be expressed in terms of finite element displacement functions as:

$$\{\varepsilon_\theta^0\} = \begin{Bmatrix} \dfrac{1}{2}\left(\dfrac{\partial w}{\partial x}\right)^2 \\ \dfrac{1}{2}\left(\dfrac{\partial w}{\partial y}\right)^2 \\ \dfrac{\partial w}{\partial x}\dfrac{\partial w}{\partial y} \end{Bmatrix} \tag{2.23}$$

Substituting the in-plane strain vector from Eq. (2.22) and non-linear von-Karman strain vector from eq. (2.23) into Eq. (2.21):

$$\left\{\varepsilon^0\right\} = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{Bmatrix} + \frac{1}{2} \begin{Bmatrix} \left(\dfrac{\partial w}{\partial x}\right)^2 \\ \left(\dfrac{\partial w}{\partial y}\right)^2 \\ 2\dfrac{\partial w}{\partial x}\dfrac{\partial w}{\partial y} \end{Bmatrix} \tag{2.24}$$

As per Eq. (2.18),

$$u = [H_u(x,y)]\{b\}$$

So,

$$\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}[H_u(x,y)]\{b\} \tag{2.25}$$

And,

$$\frac{\partial u}{\partial y} = \frac{\partial}{\partial y}[H_u(x,y)]\{b\} \tag{2.26}$$

As per Eq. (2.19),

$$v = [H_v(x,y)]\{b\}$$

So,

$$\frac{\partial v}{\partial x} = \frac{\partial}{\partial x}[H_v(x,y)]\{b\} \tag{2.27}$$

And,

$$\frac{\partial v}{\partial y} = \frac{\partial}{\partial y}[H_v(x,y)]\{b\} \tag{2.28}$$

As per Eq. (2.16),

$$\{b\} = [T_m]\{w_m\}$$

Thus,

$$\varepsilon_m^0 = [C_m][T_m]\{w_m\} \tag{2.29}$$

$$= [B_m]\{w_m\} \tag{2.30}$$

where,

$$[B_m] = [C_m][T_m] \tag{2.31}$$

In which,

$$[C_m] = \begin{Bmatrix} \frac{\partial}{\partial x}[H_u(x,y)] \\ \frac{\partial}{\partial y}[H_v(x,y)] \\ \frac{\partial}{\partial x}[H_u(x,y)] + \frac{\partial}{\partial y}[H_v(x,y)] \end{Bmatrix} \tag{2.32}$$

$$= \begin{bmatrix} 0 & 1 & 0 & y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x \\ 0 & 0 & 1 & x & 0 & 1 & 0 & y \end{bmatrix} \tag{2.33}$$

Now,

$$\{\varepsilon_\theta^0\} = \frac{1}{2}[\theta]\begin{Bmatrix} \frac{\partial w}{\partial x} \\ \frac{\partial w}{\partial y} \end{Bmatrix} \tag{2.34}$$

where,

$$[\theta] = \begin{bmatrix} \frac{\partial w}{\partial x} & 0 \\ 0 & \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial y} & \frac{\partial w}{\partial x} \end{bmatrix} \tag{2.35}$$

As per Eq. (2.17),

$$w = [H_w(x,y)]\{a\}$$

So,

$$\frac{\partial w}{\partial x} = \frac{\partial}{\partial x}[H_w(x,y)]\{a\}$$

And,

$$\frac{\partial w}{\partial y} = \frac{\partial}{\partial y}[H_w(x,y)]\{a\}$$

Now,

$$[C_\theta] = \left\{\begin{matrix} \frac{\partial}{\partial x}[H_w(x,y)] \\ \frac{\partial}{\partial y}[H_w(x,y)] \end{matrix}\right\} \tag{2.36}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 2x & y & 0 & 3x^2 & 2xy & y^2 & 0 & 3x^2y & 2xy^2 & y^3 & 3x^2y^2 & 2xy^3 & 3x^2y^3 \\ 0 & 0 & 1 & 0 & x & 2y & 0 & x^2 & 2xy & 3y^2 & x^3 & 2x^2y & 3xy^2 & 2x^3y & 3x^2y^2 & 3x^3y^2 \end{bmatrix} \tag{2.37}$$

As per Eq. (2.15),

$$\{a\} = [T_b]\{w_b\}$$

Also,

$$\{\varepsilon_\theta^0\} = \frac{1}{2}[\theta][C_\theta][T_b]\{w_b\} \tag{2.38}$$

$$= \frac{1}{2}[\theta][B_\theta]\{w_b\} \tag{2.39}$$

where,

$$[B_\theta] = [C_\theta][T_b] \tag{2.40}$$

The bending curvature vector $\{\kappa\}$is defined as:

$$\{\kappa\} = \begin{Bmatrix} -\dfrac{\partial^2 w}{\partial x^2} \\ -\dfrac{\partial^2 w}{\partial y^2} \\ -2\dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} \tag{2.41}$$

So, the curvature vector component,

$$\{\kappa\} = [C_b][T_b]\{w_b\} \tag{2.42}$$

$$=[B_b]\{w_b\}$$

$$=[C_b]\{a\} \tag{2.43}$$

where,

$$[B_b] = [C_b][T_b] \tag{2.44}$$

And,

$$[C_b] = \begin{Bmatrix} -\dfrac{\partial^2}{\partial x^2}[H_w(x,y)] \\ -\dfrac{\partial^2}{\partial y^2}[H_w(x,y)] \\ -2\dfrac{\partial^2}{\partial x \partial y}[H_w(x,y)] \end{Bmatrix} \tag{2.45}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 6x & 2y & 0 & 0 & 6xy & 2y^2 & 0 & 6xy^2 & 2y^3 & 6xy^3 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2x & 6y & 0 & 2x^2 & 6xy & 2x^3 & 6x^2y & 6x^3y \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 4x & 4y & 0 & 6x^2 & 8xy & 6y^2 & 12x^2y & 12xy^2 & 18x^2y^2 \end{bmatrix} \tag{2.46}$$

The matrices $[B_m],[B_\theta]$and$[B_b]$expressed through Eq. (2.31), (2.40), and (2.44) are the strain interpolation matrices corresponding to in-plane, large deflection, and bending strain components, respectively. Similarly, the subscripts m, θ , and b denote that the strain components are due to membrane, large deflection, and bending, respectively.

2.4 RESULTANT FORCE AND MOMENT VECTOR

To include composite material for future extension of current work, the equations are derived for composite plate, from which the equations for isotropic plate material can easily obtained. As shown in Fig. 2.2, consider the plate of overall thickness h composed of many layers of lamina with an arbitrary orientation angle θ.

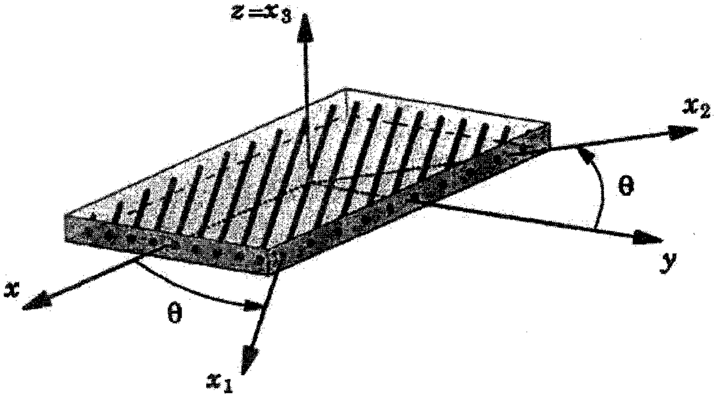


Fig. 2.2: A fiber-reinforced lamina with global and material coordinate systems

The linear constitutive relations [49, 50] for the k^{th} layer in the principal material coordinates (x_1, x_2) can be written as:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix}_k = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{21} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix}_k \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} \tag{2.47}$$

where,

$[Q]_k$ = the reduced stiffness matrix of the composite lamina

$$Q_{11} = \frac{E_1}{1-\mu_{12}\mu_{21}}$$

$$Q_{12} = \frac{\mu_{12}E_2}{1-\mu_{12}\mu_{21}}$$

$$Q_{21} = \frac{\mu_{21}E_1}{1-\mu_{12}\mu_{21}}$$

Note that, $Q_{21} = Q_{12}$

$$Q_{22} = \frac{E_2}{1-\mu_{12}\mu_{21}}$$

$$Q_{66} = G_{12}$$

For the isotropic plate,

$$E_1 = E_2 = E$$

$$\nu_{12} = \nu_{21} = \nu$$

$$G_{12} = \frac{E}{2(1-\nu)}$$

Considering the composite lamina shown in Fig. 2.2, the stress and strain transformation relations from the principal directions $x_1, x_2 \ to \ x, y$ directions are:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} = [T_\sigma(\theta)] \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{2.48}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} = [T_\varepsilon(\theta)] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{2.49}$$

where, defining $c = \cos\theta$ and $s = \sin\theta$,

$$[T_\sigma(\theta)] = \begin{bmatrix} c^2 & s^2 & 2sc \\ s^2 & c^2 & -2sc \\ -sc & sc & c^2-s^2 \end{bmatrix} \qquad (2.50)$$

$$[T_\varepsilon(\theta)] = \begin{bmatrix} c^2 & s^2 & sc \\ s^2 & c^2 & -sc \\ -2sc & 2sc & c^2-s^2 \end{bmatrix} \qquad (2.51)$$

Thus, the stress-strain relations for a generalized k^{th} lamina becomes

$$\{\sigma\}_k = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix}_k = \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{21} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{61} & \overline{Q}_{62} & \overline{Q}_{66} \end{bmatrix}_k \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}$$

$$\{\sigma\}_k = [\overline{Q}]_k \{\varepsilon\} \qquad (2.52)$$

where, $[\overline{Q}]_k$, the transformed reduced stiffness matrix, is given by

$$[\overline{Q}]_k = [T_\sigma(\theta)]^{-1}[Q][T_\varepsilon(\theta)] \qquad (2.53)$$

The resultant forces and moments per unit length are:

$$(\{N\}, \{M\}) = \int_{-h/2}^{h/2} \{\sigma\}_k (1, z)\, dz \qquad (2.54)$$

where z = layer thickness

And the constitutive equations for a laminate can be written as:

$$\begin{Bmatrix} N \\ M \end{Bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} \begin{Bmatrix} \varepsilon^0 \\ \kappa \end{Bmatrix} \qquad (2.55)$$

Thus,

$$\varepsilon_m^0 = [C_m][T_m]\{w_m\} \qquad (2.29)$$

$$= [B_m]\{w_m\} \qquad (2.30)$$

where,

$$[B_m] = [C_m][T_m] \qquad (2.31)$$

In which,

$$[C_m] = \begin{Bmatrix} \dfrac{\partial}{\partial x}[H_u(x,y)] \\ \dfrac{\partial}{\partial y}[H_v(x,y)] \\ \dfrac{\partial}{\partial y}[H_u(x,y)] + \dfrac{\partial}{\partial x}[H_v(x,y)] \end{Bmatrix} \qquad (2.32)$$

$$= \begin{bmatrix} 0 & 1 & 0 & y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x \\ 0 & 0 & 1 & x & 0 & 1 & 0 & y \end{bmatrix} \qquad (2.33)$$

Now,

$$\{\varepsilon_\theta^0\} = \frac{1}{2}[\theta] \begin{Bmatrix} \dfrac{\partial w}{\partial x} \\ \dfrac{\partial w}{\partial y} \end{Bmatrix} \qquad (2.34)$$

where,

$$[\theta] = \begin{bmatrix} \dfrac{\partial w}{\partial x} & 0 \\ 0 & \dfrac{\partial w}{\partial y} \\ \dfrac{\partial w}{\partial y} & \dfrac{\partial w}{\partial x} \end{bmatrix} \qquad (2.35)$$

where $[A],[B]$ and $[D]$ are the laminate extensional, extension-bending, and bending stiffness matrices, respectively, and are given by,

$$A_{ij} = \int_{-h/2}^{h/2} \left(\overline{Q}_{ij}\right)_k dz$$

i, j=1,2,6

$$= \sum_{k=1}^{L} \left(\overline{Q}_{ij}\right)_k \left(z_{k+1} - z_k\right) \tag{2.56}$$

$$B_{ij} = \int_{-h/2}^{h/2} \left(\overline{Q}_{ij}\right)_k zdz$$

i, j=1,2,6

$$= \frac{1}{2}\sum_{k=1}^{L} \left(\overline{Q}_{ij}\right)_k \left(z_{k+1}^2 - z_k^2\right) \tag{2.57}$$

$$D_{ij} = \int_{-h/2}^{h/2} \left(\overline{Q}_{ij}\right)_k z^2 dz$$

i, j=1,2,6

$$= \frac{1}{3}\sum_{k=1}^{L} \left(\overline{Q}_{ij}\right)_k \left(z_{k+1}^3 - z_k^3\right) \tag{2.58}$$

While deriving the equation of motion, it is assumed that the plate is thin and it means the ratio of length or width over thickness is greater than 40. Thus, the rotary inertia and shear deformation effects are considered negligible.

$$\{N\} = [A]\{\varepsilon^0\} + [B]\{\kappa\} \tag{2.59}$$

$$\{M\} = [B]\{\varepsilon^0\} + [D]\{\kappa\} \tag{2.60}$$

The isotropic plate is adopted for present study for which [B] = 0.

2.5 DERIVATION OF ELEMENT MATRICES USING PRINCIPAL OF VIRTUAL WORK

According to virtual work theory, the total work done by internal and external forces on an infinitesimal virtual displacement is null. Here, the governing equation is derived [8,46, 56, 63] using the same principle:

$$\delta W = \delta W_{\text{int}} - \delta W_{ext} = 0 \tag{2.61}$$

On the plate element, work done by the internal forces is:

$$\delta W_{\text{int}} = \int_A \left(\{\delta\varepsilon^0\}^T \{N\} + \{\delta\kappa\}^T \{M\} \right) dA \tag{2.62}$$

where,

A = Area of the element

$\{N\}$ = Resultant force vector

$\{M\}$ = Moment vector

The virtual in-plane strain vector can be expressed as:

$$\{\delta\varepsilon^0\} = \{\delta(\varepsilon_m^0 + \varepsilon_\theta^0)\}$$

$$= \delta[[B_m]]\{w_m\} + \frac{1}{2}[\theta][B_\theta]\{w_b\} \tag{2.63}$$

where,

$$\delta([B_m]\{w_m\}) = [B_m]\{\delta w_m\} \tag{2.64}$$

$$\delta\left(\frac{1}{2}[\theta][B_\theta]\{w_b\}\right) = \frac{1}{2}[\delta\theta][B_\theta]\{w_b\} + \frac{1}{2}[\theta][B_\theta]\{\delta w_b\}$$

$$= \frac{1}{2}[\theta][B_\theta]\{\delta w_b\} + \frac{1}{2}[\theta][B_\theta]\{\delta w_b\} \tag{2.65}$$

$$= [\theta][B_\theta]\{\delta w_b\}$$

Therefore Eq. (2.63) can be written as:

$$\{\delta\varepsilon^0\} = [B_m]\{\delta w_m\} + [\theta][B_\theta]\{\delta w_b\} \tag{2.66}$$

Now,

$$\{\delta\kappa\} = \delta([B_b]\{w_b\}) = [B_b]\{\delta w_b\} \tag{2.67}$$

Substituting Eq. (2.66) and Eq. (2.67) into Eq. (2.62) gives:

$$\begin{aligned} \delta W_{\text{int}} &= \int_A \left(([B_m]\{\delta w_m\} + [\theta][B_\theta]\{\delta w_b\})^T([A]\{\varepsilon^0\})\right)dA \\ &+ \int_A \left(([B_b]\{\delta w_b\})^T([D]\{\kappa\})\right)dA \end{aligned} \tag{2.68}$$

$$\begin{aligned} \delta W_{\text{int}} &= \int_A \left(([B_m]\{\delta w_m\} + [\theta][B_\theta]\{\delta w_b\})^T\left([A]\left([B_m]\{w_m\} + \frac{1}{2}[\theta][B_\theta]\{w_b\}\right)\right)\right)dA \\ &+ \int_A \left(([B_b]\{\delta w_b\})^T([D]([B_b]\{w_b\}))\right)dA \end{aligned} \tag{2.69}$$

$$\begin{aligned} \delta W_{\text{int}} &= \int_A \left(([B_m]\{\delta w_m\} + [\theta][B_\theta]\{\delta w_b\})^T\left([A][B_m]\{w_m\} + \frac{1}{2}[A][\theta][B_\theta]\{w_b\}\right)\right)dA \\ &+ \int_A \left(([B_b]\{\delta w_b\})^T([D][B_b]\{w_b\})\right)dA \end{aligned} \tag{2.70}$$

$$= \int_A \left(\{\delta w_m\}^T[B_m]^T[A][B_m]\{w_m\}\right)dA \tag{2.70-1}$$

$$+ \int_A \left(\frac{1}{2}\{\delta w_m\}^T[B_m]^T[A][\theta][B_\theta]\{w_b\}\right)dA \tag{2.70-2}$$

$$+ \int_A \left(\{\delta w_b\}^T[\theta]^T[B_\theta]^T[A][B_m]\{w_m\}\right)dA \tag{2.70-3}$$

$$+ \int_A \left(\frac{1}{2}\{\delta w_b\}^T[\theta]^T[B_\theta]^T[A][\theta][B_\theta]\{w_b\}\right)dA \tag{2.70-4}$$

$$+ \int_A \left(\{\delta w_b\}^T[B_b]^T[D][B_b]\{w_b\}\right)dA \tag{2.70-5}$$

The digit after the equation number 2.70-x indicates the term number. For instance, term 2 is the same as equation 2.70-2. Expressions for the linear stiffness matrices will be given first. Next, expressions for the first-order nonlinear stiffness matrices depending linearly on $\{w_b\}$or$\{w_m\}$ will be expressed. Finally, expressions for the second-order nonlinear stiffness matrix, depending quadratically on $\{w_b\}$, will be addressed.

2.5.1 Linear Stiffness Matrix

Equations (2.70-1) and (2.70-5) can be expressed[55] as:

$$\begin{Bmatrix} \delta w_b \\ \delta w_m \end{Bmatrix}^T \begin{bmatrix} k_b & 0 \\ 0 & k_m \end{bmatrix} \begin{Bmatrix} w_b \\ w_m \end{Bmatrix} \tag{2.71}$$

where, linear stiffness matrices $[k_b]$ and $[k_m]$ are defined as:

$$[k_b] = \int_A [B_b]^T [D][B_b] dA \tag{2.72}$$

$$[k_m] = \int_A [B_m]^T [A][B_m] dA \tag{2.73}$$

2.5.2 First-order Non-linear Stiffness Matrices

Rewriting Eq. (2.34):

$$\{\varepsilon_\theta^0\} = \frac{1}{2}[\theta]\begin{Bmatrix} \dfrac{\partial w}{\partial x} \\ \dfrac{\partial w}{\partial y} \end{Bmatrix}$$

$$= \frac{1}{2}[\theta][C_\theta][T_b]\{w_b\} \tag{2.74}$$

$$= \frac{1}{2}[\theta][B_\theta]\{w_b\} \tag{2.75}$$

Eq. (2.70-3) can also be written as:

$$\int_A \left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA$$

$$= \int_A \frac{1}{2}\left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA$$

$$+ \int_A \frac{1}{2}\left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA \tag{2.76}$$

Say, $[A][B_b]\{w_m\} = [N_m]$

Therefore,

$$[\theta]^T [A][B_m]\{w_m\} = [\theta]^T [N_m] = [N_m][B_\theta]\{w_b\} \tag{2.77}$$

Substituting the terms of Eq. (2.77) into Eq. (2.76),

$$\int_A \left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA$$

$$= \int_A \frac{1}{2}\left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA$$

$$+ \int_A \frac{1}{2}\left(\{\delta w_b\}^T [B_\theta]^T [N_m][B_\theta]\{w_b\}\right) dA \tag{2.78}$$

Combining Eq (2.70-2) and Eq. (2.78):

$$\int_A \left(\frac{1}{2}\{\delta w_m\}^T [B_m]^T [A][\theta][B_\theta]\{w_b\}\right) dA$$

$$+ \int_A \frac{1}{2}\left(\{\delta w_b\}^T [\theta]^T [B_\theta]^T [A][B_m]\{w_m\}\right) dA$$

$$+ \int_A \frac{1}{2}\left(\{\delta w_b\}^T [B_\theta]^T [N_m][B_\theta]\{w_b\}\right) dA$$

It can be expressed as

$$\frac{1}{2}\begin{Bmatrix}\delta w_b \\ \delta w_m\end{Bmatrix}^T \begin{bmatrix} n1_{nm} & n1_{bm} \\ n1_{mb} & 0 \end{bmatrix} \begin{Bmatrix} w_b \\ w_m \end{Bmatrix} \tag{2.79}$$

where $n1_{nm}, n1_{bm}, n1_{mb}$ are first-order nonlinear incremental stiffness matrices and they are linearly dependent on $\{w\}$ and can be expressed[61] as:

$$[n1_{nm}] = \int_A [B_\theta]^T [N_m(\{w_m\})][B_\theta] dA \tag{2.80}$$

$$[n1_{mb}] = [n1_{bm}]^T = \int_A [B_m]^T [A][\theta(\{w_b\})][B_\theta] dA \tag{2.81}$$

2.5.3 Second-order Non-linear Stiffness Matrix

Eq. (2.70-4) can be written as:

$$\frac{1}{3}\begin{Bmatrix}\delta w_b \\ \delta w_m\end{Bmatrix}^T \begin{bmatrix} n2_b & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} w_b \\ w_m \end{Bmatrix} \tag{2.82}$$

where $n2_b$ is second-order nonlinear incremental stiffness matrix and it is a quadratic function of $\{w_b\}$ and can be expressed as:

$$[n2_b] = \frac{3}{2}\int_A [B_\theta]^T [\theta(\{w_b\})]^T [A][\theta(\{w_b\})][B_\theta]dA \tag{2.83}$$

2.6 EQUATION OF MOTION IN STRUCTURAL DEGREES OF FREEDOM

Based on the discussion in the previous section, the virtual work of the internal forces on a plate becomes:

$$\delta W_{\text{int}} = \begin{Bmatrix} \delta w_b \\ \delta w_m \end{Bmatrix}^T \left(\begin{bmatrix} k_b & 0 \\ 0 & k_m \end{bmatrix} + \frac{1}{2}\begin{bmatrix} n1_{nm} & n1_{bm} \\ n1_{mb} & 0 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} n2_b & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{Bmatrix} w_b \\ w_m \end{Bmatrix} \tag{2.84}$$

Considering inertia and random pressure excitation, the virtual work of the external forces can be expressed[63] as:

$$\delta W_{ext} = \int_A [\delta w(-\rho h\ddot{w} + P_{random}(t)) + \delta u(-\rho h\ddot{u}) + \delta v(-\rho h\ddot{v})]dA \tag{2.85}$$

Re- writing Eq. (2.17),

$$w = [H_w][T_b]\{w_b\}$$

The above equation leads to,

$$\delta w = [H_w][T_b]\{\delta w_b\} \tag{2.86}$$

$$\dot{w} = [H_w][T_b]\{\dot{w}_b\} \tag{2.87}$$

$$\ddot{w} = [H_w][T_b]\{\ddot{w}_b\} \tag{2.88}$$

Re- writing Eq. (2.18),

$$u = [H_u][T_m]\{w_m\}$$

The above equation leads to,

$$\delta u = [H_u][T_m]\{\delta w_m\} \tag{2.89}$$

$$\dot{u} = [H_u][T_m]\{\dot{w}_m\} \tag{2.90}$$

$$\ddot{u} = [H_u][T_m]\{\ddot{w}_m\} \tag{2.91}$$

Re- writing Eq. (2.19),

$$v = [H_v][T_m]\{w_m\}$$

$$\delta v = [H_v][T_m]\{\delta w_m\} \tag{2.92}$$

$$\dot{v} = [H_v][T_m]\{\dot{w}_m\} \tag{2.93}$$

$$\ddot{v} = [H_v][T_m]\{\ddot{w}_m\} \tag{2.94}$$

Substituting Eq. (2.86, 2.89 and 2.92) into Eq. (2.85), the finite element form of the virtual work done by external forces on the plate can be written as:

$$\delta W_{ext} = \int_A \begin{bmatrix} \{\delta w_b\}^T [T_b]^T [H_w]^T \left(-\rho h [H_w][T_b]\{\ddot{w}_b\} + p\right) \\ + \{\delta w_m\}^T [T_m]^T [H_u]^T \left(-\rho h [H_u][T_m]\{\ddot{w}_m\}\right) \\ + \{\delta w_m\}^T [T_m]^T [H_v]^T \left(-\rho h [H_v][T_m]\{\ddot{w}_m\}\right) \end{bmatrix} dA \tag{2.95}$$

$$= -\begin{Bmatrix} \delta w_b \\ \delta w_m \end{Bmatrix}^T \begin{bmatrix} m_b & 0 \\ 0 & m_m \end{bmatrix} \begin{Bmatrix} \ddot{w}_b \\ \ddot{w}_m \end{Bmatrix} + \begin{Bmatrix} \delta w_b \\ \delta w_m \end{Bmatrix}^T \begin{Bmatrix} p_b \\ p_m \end{Bmatrix} \tag{2.96}$$

where $[m_b]$ and $[m_m]$ are mass matrices and they are defined[55] as:

$$[m_b] = \int_A \rho h [T_b]^T [H_w]^T [H_w][T_b] dA \tag{2.97}$$

$$[m_m] = \int_A \rho h \left([T_m]^T [H_u]^T [H_u][T_m] + [T_m]^T [H_v]^T [H_v][T_m]\right) dA \tag{2.98}$$

$$\{p_b\} = \int_A [T_b]^T [H_w]^T P_{random} \quad dA \tag{2.99}$$

where, P_{random} = Random load intensity

By equating internal and external work expressed by equations (2.84) and (2.96), the element equation of motion can be expressed as:

$$\begin{bmatrix} m_b & 0 \\ 0 & m_m \end{bmatrix} \begin{Bmatrix} \ddot{w}_b \\ \ddot{w}_m \end{Bmatrix} + \left(\begin{bmatrix} k_b & 0 \\ 0 & k_m \end{bmatrix} + \frac{1}{2} \begin{bmatrix} n1_{nm} & n1_{bm} \\ n1_{mb} & 0 \end{bmatrix} + \frac{1}{3} \begin{bmatrix} n2_b & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{Bmatrix} w_b \\ w_m \end{Bmatrix} = \begin{Bmatrix} p_b \\ p_m \end{Bmatrix} \tag{2.100}$$

In other words, the above equation can be written as:

$$[m]\{\ddot{w}\} + \left([k] + \frac{1}{2}[n1] + \frac{1}{3}[n2] \right)\{w\} = \{p\} \tag{2.101}$$

where,

$\{w\}$ = Element nodal displacement vector

$[m]$ = Element mass matrix

$[k]$ = Element linear stiffness matrix

$[n1]$ = Element first-order nonlinear incremental stiffness matrix

$[n2]$ = Element second-order nonlinear incremental stiffness matrix

$\{p\}$ = Element force vector generated because of random excitation

2.7 SYSTEM EQUATION OF MOTION

The system equation of motion is achieved by assembling the element equations of motion to system level by summing up the contributions from all elements and, then, applying the boundary conditions. It can be written as:

$$\begin{bmatrix} M_b & 0 \\ 0 & M_m \end{bmatrix}\begin{Bmatrix} \ddot{W}_b \\ \ddot{W}_m \end{Bmatrix}+\left(\begin{bmatrix} K_b & 0 \\ 0 & K_m \end{bmatrix}+\frac{1}{2}\begin{bmatrix} N1_{nm} & N1_{bm} \\ N1_{mb} & 0 \end{bmatrix}+\frac{1}{3}\begin{bmatrix} N2_b & 0 \\ 0 & 0 \end{bmatrix}\right)\begin{Bmatrix} W_b \\ W_m \end{Bmatrix}=\begin{Bmatrix} P_b \\ 0 \end{Bmatrix} \quad (2.102)$$

The above equation can also be written as follows:

$$[M]\{\ddot{W}\}+\left([K]+\frac{1}{2}[N1(\{W\})]+\frac{1}{3}[N2(\{W\},\{W\})]\right)\{W\}=\{P\} \quad (2.103)$$

Eq.(2.103) is a set of nonlinear equations that describes the motion of the structure made of plate elements due to random loads. Generally, the problems associated with equation of motion can be categorized as static or dynamic problems. If the inertial and damping term is dropped from Eq. (2.103), then it becomes the equation associated with static problems. In other words, defining static load vector as $\{P_{static}\}$, the equation of motion can be written as:

$$\left([K]+\frac{1}{2}[N1(\{W\})]+\frac{1}{3}[N2(\{W\},\{W\})]\right)\{W\}=\{P_{static}\} \quad (2.104)$$

2.8 CONDENSED SYSTEM EQUATION OF MOTION

Eq. (2.84) can be written in condensed form through separating $\{W_b\}$ and $\{W_m\}$ as follows:

$$\begin{bmatrix} M_b & 0 \\ 0 & M_m \end{bmatrix}\begin{Bmatrix} \ddot{W}_b \\ \ddot{W}_m \end{Bmatrix}+\left(\begin{bmatrix} K_b & 0 \\ 0 & K_m \end{bmatrix}+\begin{bmatrix} K1_{nm} & K1_{bm} \\ K1_{mb} & 0 \end{bmatrix}+\begin{bmatrix} K2_b & 0 \\ 0 & 0 \end{bmatrix}\right)\begin{Bmatrix} W_b \\ W_m \end{Bmatrix}=\begin{Bmatrix} P_b \\ P_m \end{Bmatrix} \quad (2.105)$$

where,

$$[K1_{nm}]=\frac{1}{2}[N1_{nm}] \quad (2.106)$$

$$[K1_{bm}]=\frac{1}{2}[N1_{bm}] \quad (2.107)$$

$$[K1_{mb}] = \frac{1}{2}[N1_{mb}] \tag{2.108}$$

$$[K2_b] = \frac{1}{3}[N2_b] \tag{2.109}$$

Separating Eq. (2.105) into two different equations,

$$[M_b]\{\ddot{W}_b\} + (([K_b] + [K1_{nm}] + [K2_b]))\{W_b\} + [K1_{bm}]\{W_m\} = \{P_b\} \tag{2.110}$$

$$[M_m]\{\ddot{W}_m\} + [K1_{mb}]\{W_b\} + [K_m]\{W_m\} = \{P_m\} \tag{2.111}$$

For thin plates, in-plane natural frequencies are usually 2 to 3 order higher than bending frequencies. So, neglecting the in-plane inertia term, $[M_m]\{\ddot{W}_m\}$ of Eq. (2.111) will not bring significant error. By neglecting the in-plane inertia term, the in-plane displacement vector $\{W_m\}$ can be easily expressed in terms of bending displacement vector $\{W_b\}$ as:

$$\{W_m\} = [K_m]^{-1}(\{P_m\} - [K1_{mb}]\{W_b\}) \tag{2.112}$$

$$= [K_m]^{-1}\{P_m\} - [K_m]^{-1}[K1_{mb}]\{W_b\} \tag{2.113}$$

$$= \{W_m\}_0 - \{W_m\}_2 \tag{2.114}$$

In Eq. (2.113), the first term is constant whereas the second term is quadratically dependent on $\{W_b\}$ as $[K1_{mb}]$ is a linear function of $\{W_b\}$ too. Thus, the matrix $[K1_{nm}(\{W_m\})]$ is evaluated by algebraic sum of two components $[K1_{nm}(\{W_m\}_0)]$ and $[K1_{nm}(\{W_m\}_2)]$, which are independent of each other and quadratically dependent on $\{W_b\}$.

According to Eq. (2.77),

$$[K1_{bm}]\{W_m\} = [K1_{nm}]\{W_b\} \tag{2.115}$$

Similarly,

$$[K1_{bm}]\{W_m\}_0 = [K1_{nm}(\{W_m\}_0)]\{W_b\} \tag{2.116}$$

Substituting Eq. (2.116) into Eq. (2.110),

$$[M_b]\{\ddot{W}_b\} + (([K_b] + [K1_{nm}] + [K2_b]))\{W_b\} + [K1_{nm}]\{W_b\} = \{P_b\}$$

Reorganizing the above equation,

$$[M_b]\{\ddot{W}_b\}+(([K_b]+2[K1_{nm}]+[K2_b]))\{W_b\}=\{P_b\} \tag{2.117}$$

In short, in terms of bending displacements, the equation of motion can be written as:

$$[M_b]\{\ddot{W}_b\}+([K]+[K_2])\{W_b\}=\{P_b\} \tag{2.118}$$

where

$$[K]=[K_b]+2[K1_{nm}(\{W_m\}_0)] \tag{2.119}$$

$$[K_2]=[K2_b]-[K1_{bm}][K_m]^{-1}[K1_{mb}]-[K1_{nm}(\{W_m\}_2)] \tag{2.120}$$

The matrices $[K]$and $[K_2]$ are independent and quadaratically dependent on bending displacement vector $\{W_b\}$, respectively.

CHAPTER III

SYNCHRONIZED AND UNSYNCHRONIZED RANDOM LOAD

The synchronized and unsynchronized random loads will be discussed in detail. This involves explanation about the generation of random load with computational aspect and also the basic concepts. The essence of the ISEED number will be clearly explained. One more important aspect that will be covered is of rescaling the generated random load vector to give the exact value of power spectrum density (PSD). This is important for dealing with unsynchronized loads. If rescaling is not dealt with initially, the difference on each load vector adds up. A simple iterative procedure for such rescaling and Matlab command for accurate reliability calculations are highlighted at last.

3.1 GENERATION OF RANDOM LOAD TIME HISTORY

Currently the random acoustic pressure for linear as well as non-linear analysis of beam, plate,[11,13,14,17,30,31,32,33] and shallow shell[34] structures is often considered uniformly distributed over the surface of the structure and synchronized in time. In addition, the random loading is generally assumed as a truncated Gaussian white noise. Truncation means that the white noise is selected for a bandwidth by choosing an appropriate value of maximum frequency $f_{\max}$. This value of maximum frequency should be chosen carefully so that all the modes that are required for modal convergence should be covered within the truncated value. It is noted here that a truncated white noise is an ideal situation, and most real life cases will have non-white spectral plot. But for most design purposes, it is convenient to use an ideal broad-band white noise as far as analytical solutions are concerned. It is customary to use the highest PSD value from a non-white noise and use it for design purposes, which provides a safe designed structure.

The use of nonwhite PSD lies in the fact that the recorded B-1B flight acoustic pressure fluctuations were available, and the nonwhite PSD does affect panel response and fatigue life. As said earlier, the highest measured or estimated PSD level should be used for beam or panel design in practice and it may also be reasonable to assume that the PSD is a band-limited white noise since the contribution from the high frequency modes is usually small.

3.2 BASIC CONCEPTS: SYNCHRONIZED LOAD VS. UNSYNCHRONIZED LOAD

Unfortunately, until now, the random load is considered as synchronized in pattern for most of the experimental and analytical work. It means that the load varies with time only, which is not always true. In other words, when a structure is subjected to random loads, it is not possible to use the load in the form $p(t)$. A simple practical example to prove the previous statement is a long period of rain drops steadily falling on the structure. Even though random in nature, and more or less uniform, the rain drops are certainly not synchronized in time. The intensity of the rain drops varies at different locations. This loading is unsynchronized in time, and it is also space dependent.

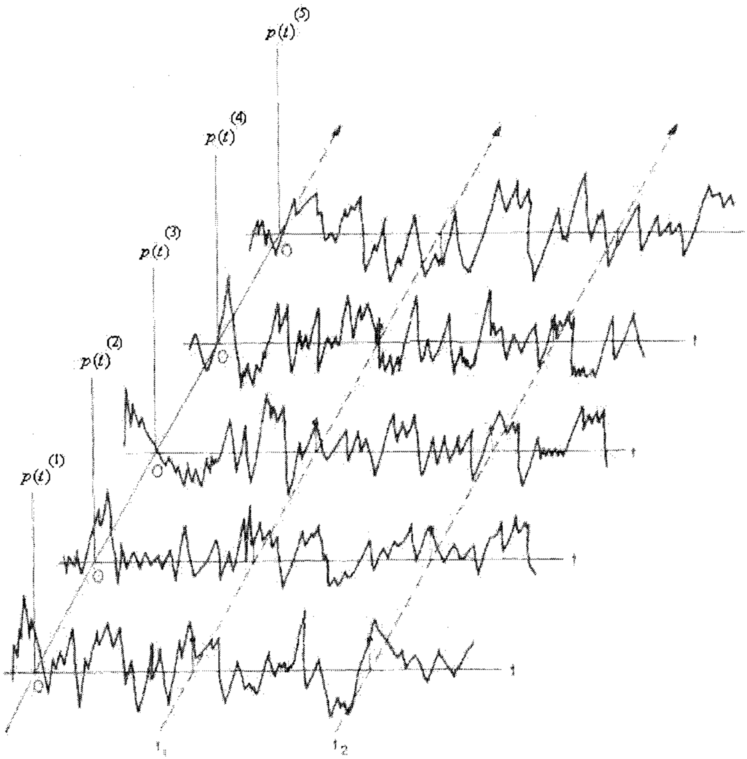


Fig. 3.1: Schematic representation of a random process $p(t)$. Each $p^{(j)}(t)$ is a sample function of the ensemble.[28]

The concept of unsynchronized load can be explained with a reference to Fig. 3.1. Consider a plate under the uniform load $p(t)$. However, it is possible that at time t_1 or t_2, the load at one point is $p^{(1)}(t)$ whereas at the next point it is $p^{(3)}(t)$. Thus, at each time, the loading intensity is different at different locations. It is also noticed that, at the same location, the load keeps changing at different times. In other words, at any given point of time, the load is varying not only in time but also in space. This is the essence of an unsynchronized load. It is apparent that a true random load should always be expressed as $p(x, y, t)$ units and not as $p(t)$ units. And it should be expressed in terms of its spectral density, S_0 (units)^2/Hz.

3.3 WHITE RANDOM PRESSURE SIMULATION

Consider random pressure $p(x,y,t)$ acting on the surface of a high-speed flight vehicle. The pressure acting normal to the panel surface varies randomly in time and space along the surface coordinates x and y. The pressure $p(x,y,t)$ is characterized by a cross-spectral density function $G_p(\xi,\eta,\omega)$, where $\xi = x_1 - x_2$ and $\eta = y_1 - y_2$ are the spatial separations and ω is the frequency in rad/sec. The simplest form of the cross-spectral density is the truncated Gaussian white noise pressure that is uniformly distributed with spatial coordinates x and y.

$$G_p(\xi,\eta,f) = \begin{cases} G_0 & if \quad 0 \le f \le f_c \\ 0 & if \quad f < 0 \; or \; f > f_c \end{cases} \tag{3.1}$$

where, G_0 is constant and f_c is the upper cut-off frequency in Hertz (Hz). The expression for G_0 can be written as[27]

$$G_0 = p_0^2 \; 10^{SPL/10} lb^2 / Hz \tag{3.2}$$

where, p_0 = reference pressure

$$= 2.90075X10^{-9} \; psi \; (20 \; \mu Pa) \tag{3.3}$$

Here, SPL is Sound Pressure Level, and it is expressed in decibels (dB).

Fig 3.2 provides enough details about a typical simulated random load at 120 dB SPL.

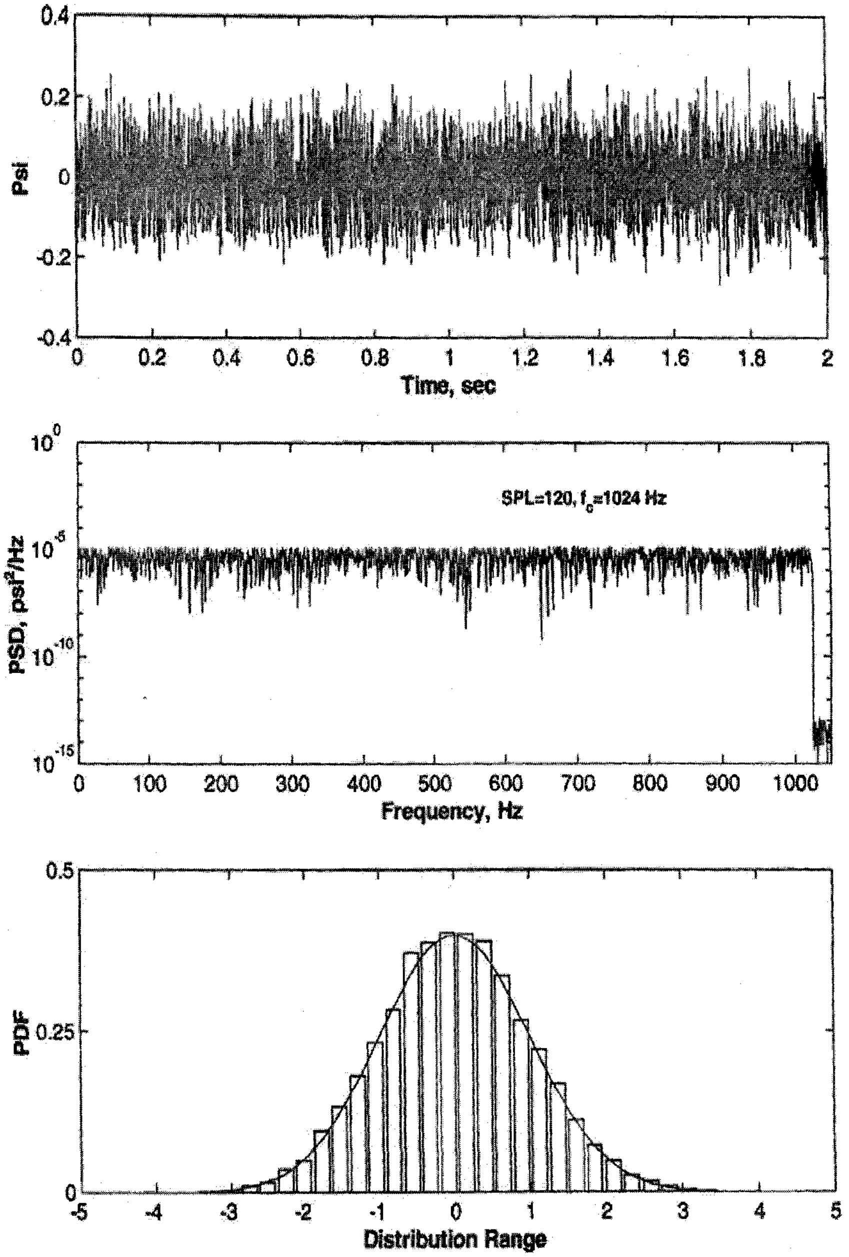


Fig. 3.2: Random white noise at SPL=120 dB and f_c =1024 Hz

The band limited white noise is generated by a Fortran code shown through Appendix B that stimulates a random pressure using complex numbers with independent random phase angles uniformly distributed between 0 and 2π . The PSD value of the random process is obtained by taking the ensemble average of the Fourier transform of the random load. The PSD value is then compared to the exact one given by equation

(3.2). The analyses presented are obtained for a cut-off frequency of 1024 Hz. The default selected frequency bandwidth in this work is $\Delta\omega = 0$ rad/sec with the random load prescribed in decibels.

The random input $p(x, y, t)$ was simulated using the FORTRAN code given in Appendix B and generated by Vaicaitis[11] with a total number of 16,384 points. The length of the simulated process is nothing but time-step multiplying the number of points. To compute the power spectrum of the responses, FFT is selected, which is a numerically suitable technique when the total number of points is expressible as a power of two. The FFT is a complicated algorithm that becomes computationally lengthy when the input numbers of points are not expressed as a power of two. For instance, note that the FORTRAN code for the white random pressure fluctuation simulation uses a similar FFT base. The total number of input points is 16,384, which corresponds to 2 to the 14^{th} power.

3.4 COMPUTATIONAL ASPECTS OF GENERATION OF UNSYNCHRONIZED TIME HISTORY

In FORTRAN language, to generate random phase angles between 0 and 2π, inbuilt subroutines called RANDOM_SEED ( ) and RANDOM_NUMBER ( ) are used. These subroutines need to use the parameter known as "ISEED" number. Each ISEED number creates its own time history of random load. By a synchronized load, it is meant that the simulated random pressure time history is generated from one seed. The different ISEED numbers guarantee each random pressure time history to be statistically uncorrelated in time.[53] This is shown through Fig. 3.3.

The choice of different ISEED numbers for each element along the plate gives a different sample function in the same ensemble. Using a different ISEED number, a different time history of random load can be obtained using the random load generator suggested by Shinozuka[35, 36, 37]. This random load can be uniform, concentrated, or non-uniform, but to simulate a real random load, the ISEED number should be different along the space for each element.

In case of unsynchronized loading, the load is also space dependent. It can not be simulated with one seed number (ISEED number) but it needs different seed numbers for each element. All the input time history of the random loads has the same power spectral density (PSD). However, when considering a large surface, the load itself can be assumed to be uniform. This is conceptually the case of application of unsynchronized random loads on a structure.

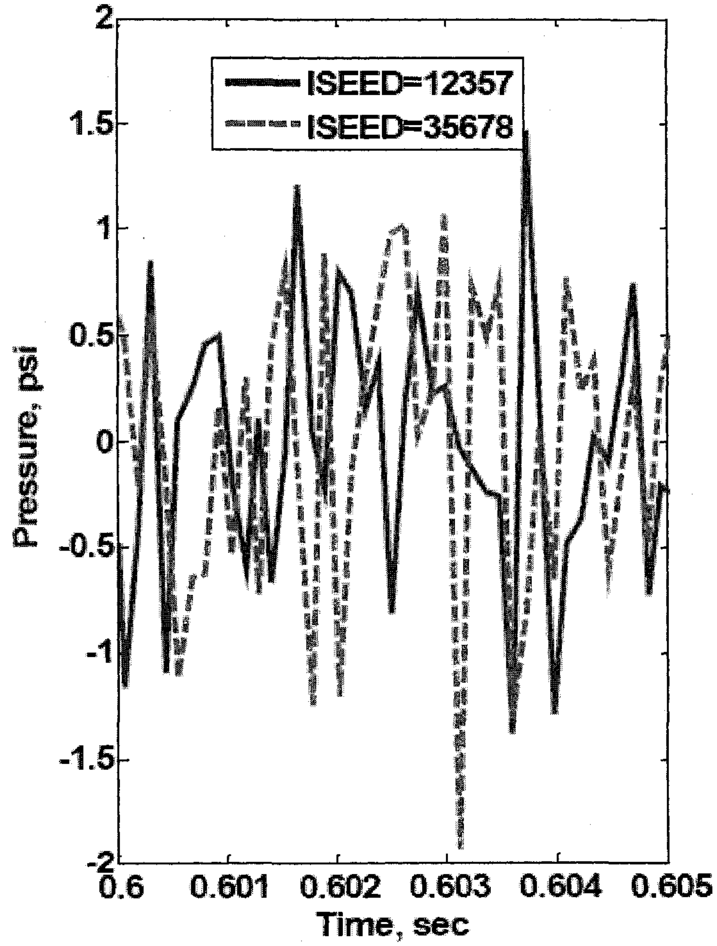
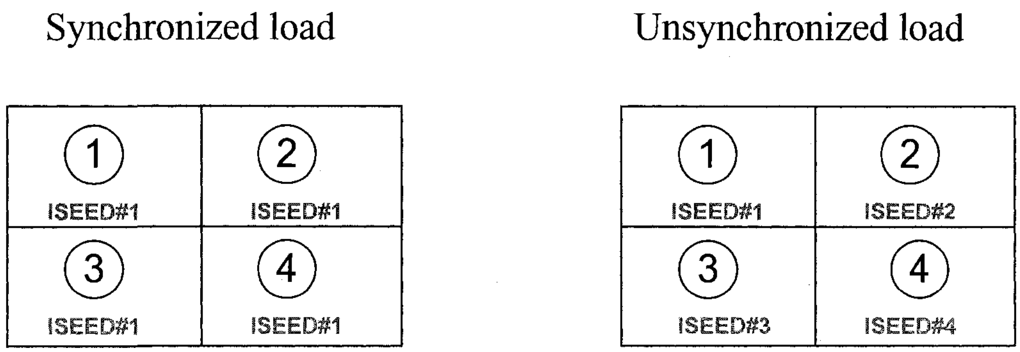


Fig. 3.3: Two pressure time histories of the same PSD from different ISEED numbers[53]

From a computational view point, when the loading considered is unsynchronized in pattern, analysis of large scale problems is extremely expensive in terms of time and memory. This statement can be easily explained through a simple example. Consider a simple structure divided in four plate elements as shown in Fig. 3.4. As shown in left hand side of the Fig. 3.4, when the loading is synchronized, the same ISEED number is used for all the four plate elements to generate time histories. Whereas, when the loading pattern is unsynchronized in nature, ISEED numbers used for each of the four plate elements are different and it means different time history is generated for each element as expressed through the right hand side of Fig. 3.4. For a large scale problem with very large number of elements, the problem becomes computationally complex as each element owns individual time history. The solution leads to application of parallel computation.


Synchronized load

Unsynchronized load

Fig 3.4: Computational basics: Synchronized vs. Unsynchronized load case

3.5 MONTE CARLO SIMULATION (MCS)

For the Monte Carlo Simulation (MCS), an ensemble of ten or more time histories is generated by specifying different seeds (ISEED) to the random number generator in the FORTRAN code described in Appendix B. The response statistics are generated from an ensemble of p=16 time histories at each load level. Estimates of the RMS displacement serve as a basic comparison with response of the two flight data sets (NWs), which essentially have the RMS as their basic unknown. Additionally, confidence intervals for the mean value of the RMS estimate are generated to quantify the degree of uncertainty in the results. For an input quantity x_i, the value is estimated from p independent observations $x_{i,k}$ of x_i. The input estimate is the sample mean and can be expressed as:

$$x_i = \bar{x}_i = \frac{1}{p}\sum_{k=1}^{p} x_{i,k} \tag{3.4}$$

And the standard uncertainty $u(x_i)$ to be associated with x_i is the estimated standard deviation of the mean.[38]

$$u(x_i) = \sigma(\bar{x}_i) = \left[\frac{1}{n(n-1)}\sum_{k=1}^{p}(x_{i,k} - \bar{x}_i)^2\right]^{1/2} \tag{3.5}$$

3.6 TIME STEP CONSIDERATIONS

The time step of integration depends on the scheme selected means. Whether the scheme is explicit or implicit, the element size and the order of nonlinearity need to be studied. If an explicit integration scheme is selected, the system is conditionally stable and stability is achieved as soon as a solution is obtained. Conversely, the explicit integration schemes will diverge, showing instability in the system. For an implicit scheme a solution is always obtained, i.e., the system is always unconditionally stable. It is widely recognized that an implicit scheme is faster than explicit schemes because a larger time step can be used for a converged solution. However, for an equal time step the explicit scheme is much faster than the implicit scheme because of its simplicity and ease in programming. In practical structural problems, engineers first try the implicit integration scheme because lower integrating time steps can be used. However, as soon as the time step becomes the order of 10^{-4} for converged solutions, engineers switch to explicit schemes as they are more suitable for the computation.

Depending on the nonlinearity of the system, more or less refined mesh would be necessary to catch the response characteristics. The more nonlinear the system, the more refined mesh and smaller integrating time step is required. A good amount of literature is available on numerical approaches that give empirical relations to estimate the maximum usable time steps for explicit and implicit schemes. For instance, Zienkiewics and Taylor[29] report empirical relations for the time step of integration as a function of the element size. After this brief discussion, it becomes obvious that modal truncation reduces the step integration time by reducing the dof. The mesh size remains the same for accuracy purposes. Computational time is also saved because the nonlinear matrices do not need to be assembled and updated at each time step.

One should also keep in mind the Nyquist-Shannon sampling theorem, which basically states that it is necessary to sample a time sequence at least two times faster than the highest frequency present in the waveform to uniquely resolve that frequency from the lower frequency

$$\Delta t_s \leq \frac{1}{2\ f_c} \tag{3.6}$$

where, f_c is the cut-off upper frequency of the uniformly generated random load

Taking into consideration the above remarks, an appropriate time step was selected as follows. Knowing the highest frequency of the panel, Δt_s is evaluated and used as the time integration step-size for a given loading. Then, the step-size of integration is cut into one-half until the time histories of the response are found identical. For simplicity, in the modal FEA code the time step Δt, the explicit integration scheme such as Runge-Kutta scheme is selected when the total number of points is expressible as a power of two such that the specified loading at each Δt is maintained. As mentioned previously, a radix-2 number of time history samples are chosen to facilitate use of the FFT algorithm employed in the subsequent analysis. Note that for linear problems, the Nyquist time-step Δt_s is generally sufficient for the explicit scheme. However, for nonlinear problems, the identical verification of the responses for two decreasing consecutive time steps is required and that yields a much smaller integration time step.

3.7 RESCALING OF RANDOM LOAD VECTOR

As mentioned before, the number of points and number of ensembles have an important role for the PSD calculation. The evaluation of the PSD using the Matlab command called "PWELCH" is defined as follows:

$$[Pxx, F] = pwelch(x, NFFT, Fs, Window, Noverlap) \tag{3.7}$$

where,

x = Discrete-time signal

$NFFT$ = Integer indicating the length of the FFT, which is equal to number of time history points in most cases

F_s = Sampling frequency in Hz

$Window$ = Length of the segments windowed with a *Hanning window*

$Noverlap$ = Number of overlapping sections

P_{xx} = PSD in powers/Hz

F = Frequency range in Hz

As already discussed, the random load is generated using subroutine SIMLOAD given in Appendix B where one of the input parameters is S_p. It is observed that there is a small difference between the input S_P and the S_P from the generated random load time history. In the unsynchronized case, for each load the ISEED number is different. Therefore, the values of the generated S$_P$ will be different too. This leads to an error, and convergence is delayed. Thus, it is suggested to rescale the random load generated by SIMLOAD. The following procedure gives a fairly good readjustment in the value of S_P.

1. Generate random pressure vector, $\{P_{random}\}_{NPTX1}$ using different *ISEED* numbers for all time history points (NPT)
2. Compute Mean: $[PSD(P_{random})] = S_{pOLD}$

3. Compute the ratio: $ratioS_p = \dfrac{S_{pOLD}}{S_p}$,where S_p is the input value

4. Scale the random pressure vector: $\{P_{random\_scaled}\}_{NPTX1} = \dfrac{\{P_{random}\}_{NPTX1}}{\sqrt{ratioS_p}}$

5. Recalculate Mean: $\lfloor PSD(P_{random\_scaled})\rfloor = S_{pNEW}$

6. % Error = $\dfrac{S_{pNEW} - S_{pOLD}}{100}$

7. Use the $\{P_{random\_scaled}\}$ in numerical integration

Thus, once the random pressure time history is generated the average value of the auto spectral density is calculated and compared with S_0 for a given SPL for verification purposes. The FORTRAN code shown in Appendix D follows the same procedure exactly as per the above discussion. To verify the code, random load time history for 2 seconds is generated using cut-off frequency, $f_c = 4096\ Hz.$ and ISEED=14407. Fig. 3.5 shows PSD and Probability Density Function (PDF) plotting. Fig 3.6 confirms accuracy of the code generated and the difference between the input S_p and calculated S_p based on rescaled load vector is clearly visible in the plot.

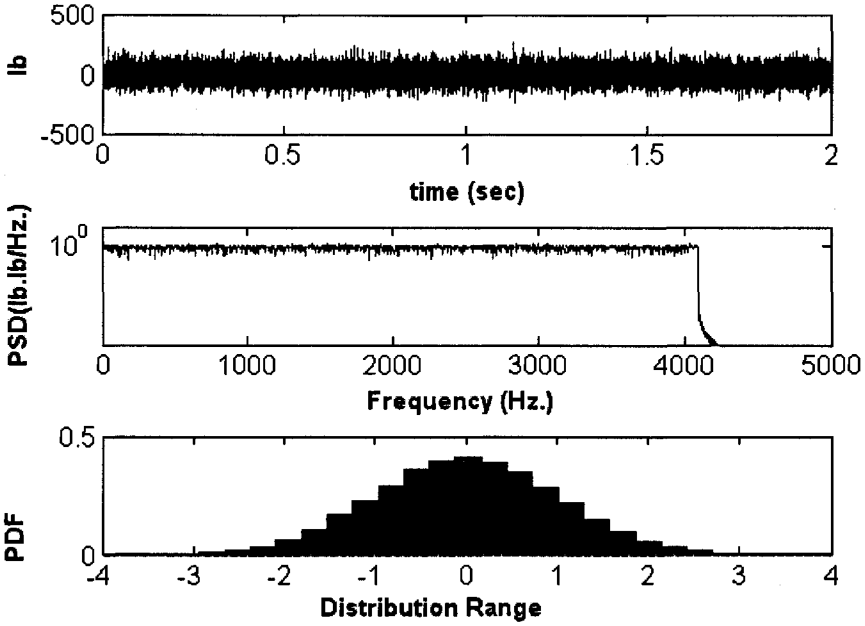


Fig 3.5: Random load time history, PSD and PDF (Probability Density Function); ISEED=14407

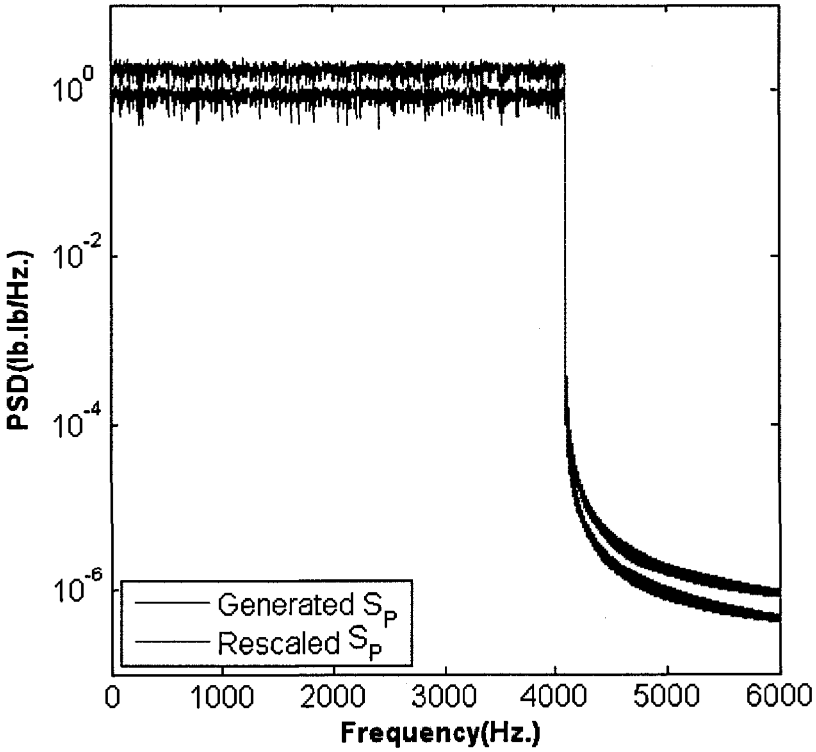


Fig. 3.6: Effect of rescaling of random load vector

CHAPTER IV

SOLUTION PROCEDURE USING MODAL FORMULATION

4.1 INTRODUCTION

In this chapter, detailed solution procedures are presented for solving all the physical problems described in Chapter 2. In order to proceed with specific problems, various preliminary tasks need to be performed. These include solving linear Eigen-problems to obtain frequencies and mode shapes for the modal transformation, and generation of accurate time histories of random pressure fluctuations with flat power spectral densities as discussed in detail in Chapter 3. For the structure subjected to random vibration, the system equations of motion are first transferred into modal equations using normal modes followed by time domain numerical method. The advantages of using modal approach and time integration scheme are also listed. Numerical considerations like the integration scheme, convergence criteria, and removing the transient response to ensure accurate response statistics are also addressed. Finally, importance of the usage of a modal participation factor is discussed in detail.

4.2 ADVANTAGES OF THE MODAL APPROACH

Rewriting the condensed system of equation of motion defined by Eq. (2.118)

$$[M_b]\{\ddot{W}_b\}+([K]+[K_2])\{W_b\}=\{P_b\} \tag{4.1}$$

where

$$[K]=[K_b]+2[K1_{nm}(\{W_m\}_0)] \tag{4.2}$$

$$[K_2]=[K2_b]-[K1_{bm}][K_m]^{-1}[K1_{mb}]-[K1_{nm}(\{W_m\}_2)] \tag{4.3}$$

Eq. (4.1) can be solved by numerical integration in the structural node dof. This approach turns out to be computationally expensive because of following:

1. At each time step, the element nonlinear stiffness matrices have to be evaluated, and the system nonlinear stiffness matrices have to be assembled and updated.
2. The number of structural bending dof $\{W_b\}$ is usually very large.
3. The time step of the integration scheme should be extremely small in order to make the solution accurate and stable.

An alternative and effective solution procedure is to transform the equation of motion from structural degrees of freedom into modal coordinates. The main advantage of using the modal approach is computational saving. As nonlinear stiffness matrices are constant, they do not need to be reassembled at each time step of integration. Moreover, the number of equations remained in the solution is usually 2-3 orders lower compared to structural dof approach. For most of the cases, the number of modes needed to obtain modal convergence is less than twenty and the time step when performing numerical integration is larger. Another advantage of the FE modal approach is that the in plane inertia does not need to be neglected in order to obtain the solution. It is not the same case for the Galerkin/PDE procedure. The procedure for the modal formulation, using reduced system normal modes, is described in the next section.

4.3 LINEAR VIBRATION PROBLEM

Re-writing equation of motion in structural dof expressed through Eq. (2.105),

$$\begin{bmatrix} M_b & 0 \\ 0 & M_m \end{bmatrix} \begin{Bmatrix} \ddot{W}_b \\ \ddot{W}_m \end{Bmatrix} + \left(\begin{bmatrix} K_b & 0 \\ 0 & K_m \end{bmatrix} + \begin{bmatrix} K1_{nm} & K1_{bm} \\ K1_{mb} & 0 \end{bmatrix} + \begin{bmatrix} K2_b & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{Bmatrix} W_b \\ W_m \end{Bmatrix} = \begin{Bmatrix} P_b \\ P_m \end{Bmatrix} \qquad (4.4)$$

In order to attempt modal transformation of above equation, the linear Eigen problem expressed by following equation needs to be solved:

$$\omega_r^2 \begin{bmatrix} M_b & 0 \\ 0 & M_m \end{bmatrix} \begin{Bmatrix} \phi_b \\ \phi_m \end{Bmatrix}^{(r)} = \begin{bmatrix} K_b & 0 \\ 0 & K_m \end{bmatrix} \begin{Bmatrix} \phi_b \\ \phi_m \end{Bmatrix}^{(r)} \qquad (4.5)$$

where, $\{\phi_b\}^{(r)}$ and $\{\phi_m\}^{(r)}$ are r$^{\text{th}}$ normal modes of the linear vibration problem related to bending and in-plane dof, respectively. Normal mode $\{\phi_b\}^{(r)}$ and corresponding linear frequency ω_r can be obtained by solving part of the equation of linear vibration which can be written as:

$$\omega_r^2\ [M_b]\ \{\phi_b\}^{(r)} = [K_b]\ \{\phi_b\}^{(r)} \tag{4.6}$$

For isotropic plate, there is no coupling between bending $\{\phi_b\}^{(r)}$ and in-plane $\{\phi_m\}^{(r)}$ modes. As a result, the in-plane displacement $\{W_m\}$ will be expressed as a function of the bending displacement $\{W_b\}$.

4.4 DYNAMIC RESPONSE USING MODAL EQUATIONS IN NORMAL MODES

As discussed earlier in Chapter 2, by neglecting the inertia term, the membrane displacement vector can be expressed in terms of the bending displacement vector as:

$$\{W_m\} = [K_m]^{-1}(\{P_m\} - [K1_{mb}]\{W_b\}) \tag{4.7}$$

Re-writing equation of motion expressed through Eq. (2.110),

$$[M_b]\{\ddot{W}_b\} + (([K_b] + [K1_{nm}] + [K2_b]))\{W_b\} + [K1_{bm}]\{W_m\} = \{P_b\} \tag{4.8}$$

Substituting Eq. (4.5) into Eq. (4.8),

$$\begin{aligned}&[M_b]\{\ddot{W}_b\} + (([K_b] + [K1_{nm}] + [K2_b]))\{W_b\} \\ &+ ([K1_{bm}][K_m]^{-1}\{P_m\} - [K1_{bm}][K1_{mb}])\{W_b\} = \{P_b\}\end{aligned} \tag{4.9}$$

In the above equation, system bending displacement $\{W_b\}$ can be expressed as a linear combination of some known base functions called mode shapes as:

$$\{W_b\} = \sum_{r=1}^{n} q_r(t)\ \{\phi_b\}^{(r)} = [\Phi]\{q\} \tag{4.10}$$

where,

q = Modal amplitude

$\{\phi_b\}^{(r)}$ = rth normal mode of the linear vibration problem

$[\Phi]$ = Eigen vector matrix

$\{q\}$ = Modal displacement vector

To convert the equation of motion into modal co-ordinate system, it is necessary to transform all the matrices in Eq. (4.1) into modal coordinates. First of all, element nonlinear stiffness matrices are evaluated with the corresponding element components $\{w_b\}$ and which in turn is obtained from the known system linear mode $\{\phi_b\}^{(r)}$. The nonlinear stiffness matrices, which are directly related to $\{w_b\}$, can be expressed as the summation of the products of normal modes amplitudes q_r (where r =1 to number of modes, n).Thus, nonlinear modal stiffness matrices become:

$$[K1_{bm}] = \sum_{r=1}^{n} q_r(t)\ [K1_{bm}(\phi_b)]^{(r)} \tag{4.11}$$

$$[K1_{mb}] = \sum_{r=1}^{n} q_r(t)\ [K1_{mb}(\phi_b)]^{(r)} \tag{4.12}$$

$$[K2_b] = \sum_{r=1}^{n}\ \sum_{s=1}^{n} q_r(t)\ q_s(t)\ \left([K2_b(\phi_b)(\phi_b)]^{(rs)}\right) \tag{4.13}$$

In Eq. (4.11), (4.12) and (4.13), the super indices of the nonlinear modal stiffness matrices assembled from the corresponding element nonlinear stiffness matrices.

As shown through Eq. (4.7), the in-plane displacement $\{W_m\}$, is expressed as:

$$\{W_m\} = \left([K_m]^{-1}\{P_m\} - [K_m]^{-1}[K1_{mb}]\right)\{W_b\}$$

$$= \{W_m\}_0 - [K_m]^{-1}\left(\sum_{r=1}^{n} q_r(t)[K1_{mb}(\phi_b)]^{(r)}\right)[\Phi]\{q\} \tag{4.14}$$

$$= \{W_m\}_0 - \sum_{r=1}^{n}\ \sum_{s=1}^{n} q_r(t)\ q_s(t)\ \{\phi_{rs}\}_m \tag{4.15}$$

$$= \{W_m\}_0 - \{W_m\}_2 \tag{4.16}$$

where, the in-plane mode corresponding to the bending mode is:

$$\{\phi_{rs}\}_m = [K_m]^{-1}[K1_{mb}]^{(r)}\{\phi_b\}^{(s)} \tag{4.17}$$

Therefore, $[K1_{nm}(\{W_m\}_2)]$can be expressed as:

$$[K1_{nm}(\{W_m\}_2)] = \sum_{r=1}^{n}\sum_{s=1}^{n} q_r q_s [K1_{nm}(\{\phi_{rs}\}_m)] = \sum_{r=1}^{n}\ \sum_{s=1}^{n} q_r q_s [K2_{nm}]^{(rs)} \tag{4.18}$$

$\{W_m\}_0$ term of Eq. (4.16) has been considered in Eq. (4.2) that defines matrix $[K]$. It is important to notice that above defined matrices $[K1_{nm}]$, $[K1_{bm}]$, $[K1_{mb}]$ and $[K2_b]$ are constant. Once these matrices are evaluated then the system dynamic equation is transformed into the modal co-ordinates. Introducing structural modal damping term, the modal equation of motion can be written as:

$$[\overline{M}_b]\{\ddot{q}\} + 2\varsigma_r\omega_r[\overline{M}_b]\{\dot{q}\} + ([\overline{K}] + [\overline{K}_{qq}])\{q\} = \{\overline{P}_b\} \tag{4.19}$$

where

$[\overline{M}_b]$ = Modal mass matrix related to bending dof only

$$= [\Phi]^T [M_b][\Phi] \tag{4.20}$$

$[\overline{K}]$ = Modal linear stiffness matrix

$$= [\Phi]^T [K][\Phi] \tag{4.21}$$

$[\overline{K}_{qq}]$ = Second order nonlinear modal stiffness matrix

$$= [\Phi]^T \sum_{r=1}^{n}\sum_{s=1}^{n} q_r q_s \left([K2_b]^{(rs)} - [K2_{nm,}]^{(rs)} - [K1_{bm}]^{(r)}[K_m]^{-1}[K1_{mb}]^{(s)}\right) [\Phi] \tag{4.22}$$

$\{\overline{P}_b\}$ = Modal load vector related to bending dof only

$$= [\Phi]^T \{P_b\} \tag{4.23}$$

$2\varsigma_r\omega_r[\overline{M}_b]$ = Modal structural damping matrix

Here, the coefficient ς_r is the modal damping ratio for the rth mode, and it can be determined experimentally or pre-selected from a similar structure, whereas, ω_r is the rth modal natural frequency.

4.5 FOURTH ORDER RUNGE-KUTTA INTEGRATION SCHEME

The Runge-Kutta method[25] is an explicit step-by-step process to obtain approximation q_{k+1} from q_k in such a way that the power series expansion of the approximation would coincide up to terms of a certain h^N in the spacing, $h = t_{k+1} - t_k$, with the actual Taylor series development of $q(t_k + h)$in powers of h.

The fourth-order accuracy Runge-Kutta scheme, $O(h^4)$, is given by

$$q_{k+1} \cong q_k + \frac{1}{6}(b_1 + 2b_2 + 2b_3 + b_4) \tag{4.24}$$

where, the coefficients b_1, b_2, b_3 and b_4 are defined as follows:

$$b_1 = hF(t_k,\ q_k) \tag{4.25}$$

$$b_2 = hF\left(t_k + \frac{1}{2}h,\ q_k + \frac{1}{2}b_1\right) \tag{4.26}$$

$$b_3 = hF\left(t_k + \frac{1}{2}h,\ q_k + \frac{1}{2}b_2\right) \tag{4.27}$$

$$b_4 = hF\left(t_k + h,\ q_k + b_3\right) \tag{4.28}$$

4.5.1 Solving Second Order Differential Equations Using Runge-Kutta Time Integration Scheme

The method described in the previous section is often sufficient to approximate first order differential equations, but it may not be obvious about how to apply it to the approximation of differential equations of higher order. The trick here is to break down the higher order differential equation into several first order differential equations. The following example explains the technique in detail.

<u>Example 4.1</u>:

Considering second order differential equation (4.1)

$$[M_b]\{\ddot{W}_b\} + ([K] + [K_2])\{W_b\} = \{P_b\} \tag{4.29}$$

Considering initial conditions given are zero. i.e. When time t=0, displacement $\{W_b\}$=0 and velocity $\{\dot{W}_b\}$=0 . In order to recast the above equation, introducing two new variables:

$$q_1(t) = \{W_b\} \tag{4.30}$$

and

$$q_2(t) = \frac{dq_1(t)}{dt} \tag{4.31}$$

So, $\dfrac{dq_2(t)}{dt} = \{\ddot{W}_b\}$

Eq. (4.29) can be written as:

$$\{\ddot{W}_b\} = [M_b]^{-1}\{P_b\} - [M_b]^{-1}([K]+[K_2])\{W_b\} \tag{4.32}$$

Substituting Eq. (4.30) and (4.31) into Eq. (4.32),

$$\frac{dq_2(t)}{dt} = [M_b]^{-1}\{P_b\} - [M_b]^{-1}([K]+[K_2])q_1(t) \tag{4.33}$$

Here, Eq. (4.31) and Eq. (4.33) are two coupled first order equations and they represent Eq. (4.29). These equations can also be expressed as first order matrix equations:

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{4.34}$$

$$\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -[M_b]^{-1}([K]+[K_2]) & 0 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ [M_b]^{-1}\{P_b\} \end{Bmatrix} \tag{4.35}$$

Using the following steps, both first order equations expressed through Eq. (4.31) and Eq. (4.33) can be solved simultaneously.

Step: 1. Starting at time t_0, choose a value for *h*, and find initial conditions for all state variables $q_1(t_0), q_2(t_0)$...etc.

Step: 2. From the values of$q_i(t_0)$, calculate derivatives for each $q_i(t)$ at $t = t_0$

$$y_{1i} = \frac{dq_i(t)}{dt} \text{at } t = t_0$$

So, $y_{11} = \dfrac{dq_1(t)}{dt}, \ldots$etc.

Step: 3.Using the value of y_{1i}, find an approximate value for each $\overline{q}_i(t_0 + h) = q_i(t_0) + y_{1i}h$

Step: 4. Substitute $t_0 = t_o + h$, and for each q_i, let $q_i(t_0) = \overline{q}_i(t_0 + h)$

Step: 5. Repeat steps 2 through 4 until the solution is converged.

4.5.2 Advantages of using the Runge-Kutta Time Integration Scheme

Because of the nature of the problem to be analyzed, the explicit integration scheme was selected over an implicit integration scheme. The advantages of using the Runge-Kutta time integration scheme can be listed as follows:

1. The computational ease of the Runge-Kutta method makes it quite simple to program and implement.
2. Compared with methods in structural node dof, the computational cost is reduced dramatically.
3. No preliminary differentiation is needed.
4. No initial values are needed beyond the prescribed values. Instead of using values of the N derivatives at y at one point, only the values of the first derivatives at N suitably chosen points are required.
5. It is more efficient particularly for the kind of problems for which accuracy of the response frequency contents becomes critical for the evaluation of the displacements.
6. For a nonlinear random vibration problem, it does not need to assume that the random response distribution is Gaussian as using the equivalent linearization method.

4.6 SOLUTION IN STRUCTURAL DOF

As discussed in the previous section, using the Runge-Kutta integration scheme, the initial values of modal coordinates $\{q\}$ and $\{\dot{q}\}$ given in Eq. (4.19) can be solved to obtain the numerical value of modal co-ordinate vector $\{q\}$. The dof of $\{q\}$depends on the number of modes that have to be considered in order to accurately capture the desired response. Based on which nodal displacement vector associated with bending degrees of freedom, $\{W_b\}$can be calculated using the following equation:

$$\{W_b\} = [\Phi]\{q\} \tag{4.36}$$

Once $\{W_b\}$ is evaluated, the nodal displacement vector related to in-plane dof, $\{W_m\}$ can be calculated using the equation:

$$\{W_m\} = [K_m]^{-1}\{P_m\} - [K_m]^{-1}[K1_{mb}]\{W_b\} \tag{4.37}$$

Finally, Root-Mean-Square (RMS) maximum deflection is calculated as,

$$RMS(W_{\max}) = \sqrt{E\left[(W_{\max})^2\right]} \tag{4.38}$$

where, E is as discussed in Appendix C.

4.7 MODE SELECTION

In case of synchronized loads, analytical solutions are available, from which we can see that the even-modes are non-existent, and they can be removed all together in simulation to save computation time.

However, in case of unsynchronized loads, it is expected that both even and odd modes participate, but it is difficult to say which mode is dominant. Such a conclusion can only be made after obtaining the plot of power spectral density vs. the frequencies. It is expected that all the modes contribute and modes participation need to be studied in detail. Also, the location of the maximum deflection point can be determined.

4.8 DATA MANIPULATION

Initially the structure is at rest; therefore, an initial transient response is induced before the response becomes fully developed. The transient response must be eliminated to ensure that the accurate response statistics are covered. For each input loading of time history, the first 20% is omitted out of the total run. It is already discussed that in order to improve the FFT algorithm it was convenient to use a total number of points that will be a power of two. Consequently, for each displacement, the data was linearly interpolated in order to produce 2^n points, where n is an integer.

As per the technique of Monte Carlo Simulation discussed in Section 3.5, each of these *ISEED* numbers needs to be changed for each sample and 16 different samples are used for statistical averages.

4.8.1 Convergence Considerations

The accuracy of the solution discussed in previous sections is directly related to the mesh size. Under those circumstances, a convergence test for modes and mesh sizes that will give a set of modal equations for accurate response must be performed prior to any further calculations.

Two types of the convergence of the solution must be addressed. Firstly, while attempting the solution of the linear vibration problem, the natural frequency convergence must be reached. To investigate this type of convergence the finite model discretization is refined and the change in the fundamental frequency is calculated. It is worth noticing that since the forcing function is assumed uniform over the plate, the advantage of the symmetry is used and the response of a rectangular plates are calculated based on the modeling of a quarter plate only.

Secondly, when performing modal transformation the important issue is to predict how many modes should be retained in the analysis. In order to resolve this issue modal convergence is sought. The nonlinear response of the panel is the linear combination of certain modes and each of them has a certain contribution to the total response. Since the mode contribution in the total response varies with the random vibration, the estimation of the modal convergence should be performed over the entire range of the response under investigation. Generally, more number of modes are required for larger values of $W_{\max} / h$ or $RMS(W_{\max} / h)$. There are two ways to predict the mode contribution from a particular mode:

1. Plotting the graph of power spectral density (PSD) vs. frequency
2. Calculating Modal Participation Factor

Both of these methods are discussed in detail in the next sections. Both mesh and modal convergence criteria are a compromise between the accuracy and the computational cost and can be adjusted by user according to one's objective and computational possibilities.

As discussed in Section 3.6, time-step convergence is also sought. Firstly, a time step $\Delta t = 1/4096 = 2.4414 \times 10^{-4}$ sec was selected, and then the time step was cut by one-half with $\Delta t = 1/(2 \times 4096)$ sec, followed by $\Delta t = 1/(2 \times 2 \times 4096)$. The maximum deflection time histories for the last two integration time steps were compared and found to be exactly identical, establishing the time-step convergence.

4.9 POWER SPECTRAL DENSITY (PSD) VS. FREQUENCY GRAPH

When loading is considered synchronized in time and uniformly distributed, the asymmetric natural bending modes of the panel need not to be considered in the analysis. Whereas, for unsynchronized load cases, the asymmetric as well as the symmetric modes would be excited and should be considered in the analysis. The classical solution is misleading in the sense of non-participation of asymmetric modes. It gives an impression that these modes do not have an influence on the response of a random load. The above statement can be proved easily by spectral density plots as they show peaks at those modes, showing their participation.

The power spectral density vs. frequency graph helps to figure out which modes are contributing to the dynamic response and also the most dominant mode numbers. For example, Fig. 4.1 shows one of the sixteen time histories using seven modes in case of synchronized load. By plotting a PSD graph as shown in Fig. 4.2 one can easily visualize that the peaks are at modes 1, 3 and 5, showing maximum contribution from those modes. At the same time, it makes it clear that only symmetric modes have contribution in the response. It provides clear guidance about which modes should be omitted and which should be retained in a particular case.

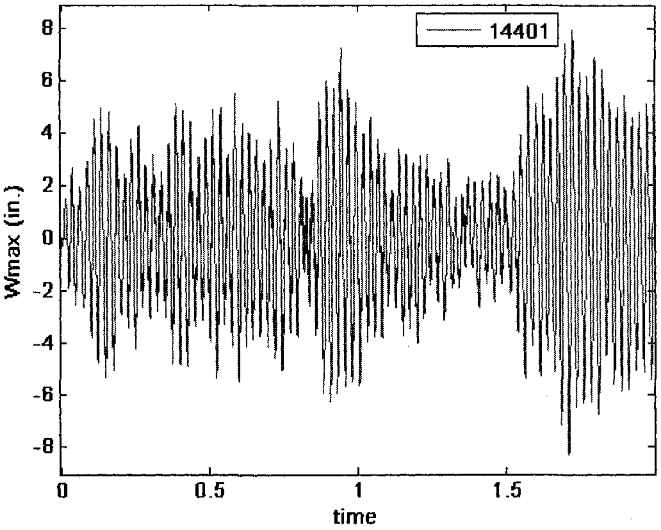


Fig. 4.1: Maximum deflection time history under synchronized loads

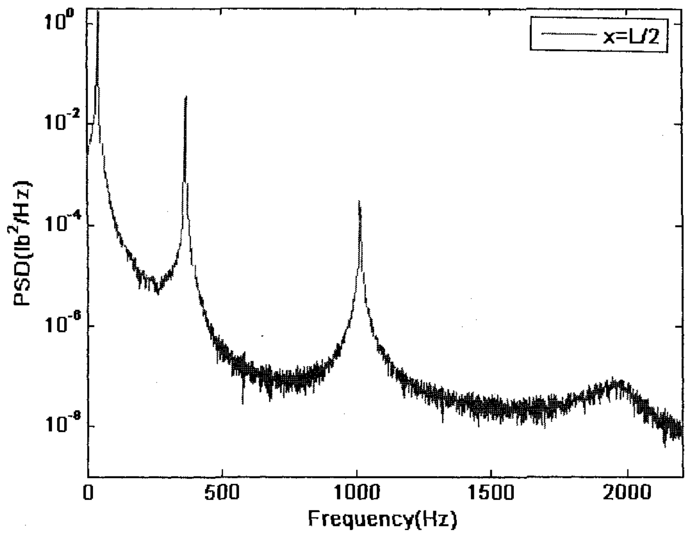


Fig. 4.2: PSD under synchronized load

4.10 MODAL PARTICIPATION FACTOR

While solving Eq. (4.19), it is extremely important to know which modes contribute to the total response. And a factor called "modal participation factor" is useful for this purpose. It can be evaluated based on the numerical values of modal co-ordinate vector $\{q\}$. The equation to calculate modal participation factor in order to know a small number of most contributing modes to the total response can be written as:

$$Partcipation\ of\ r^{th}\ \text{mode} = \begin{cases} \dfrac{RMS(q_r)}{\sum\limits_{s=1}^{n} RMS(q_s)} & if\ random\ load\ case \\ \dfrac{\max|q_r|}{\sum\limits_{s=1}^{n} \max|q_s|} & otherwise \end{cases} \tag{4.39}$$

where, RMS stands for root-mean-square value.

Those modes with significant participation values can be identified using Eq. (4.39) and they should be retained in the analysis.

CHAPTER V

SPARSE COMPUTATION METHODOLOGIES

Sparse technology is used efficiently throughout the analysis in order to save computational time and memory. The algorithm used for Eigen-solution is discussed in detail. The sparse storage scheme is explained in detail along with numerical examples whenever necessary. Some other techniques, like symbolic and numerical factorization and LDL^T equation solver are also detailed in this chapter.

5.1. EIGEN SOLUTION USING SUB SPACE ALGORITHM

The Sub-space iteration method[23] is adopted in the study as it is the effective method to find few lowest Eigen pairs of a large Eigen problem. The method incorporates inverse iteration and the generalized Jacobi iteration methods. The main steps of the algorithm can be described as follows:

1. Assuming the first "m" Eigen-pair solution of the linear vibration problem is sought. Rewriting the linear vibration equation (4.6),

   $$\omega_r^{\ 2}\ [M_b]\ \{\phi_b\}^{(r)} = [K_b]\ \{\phi_b\}^{(r)} \tag{5.1}$$

   In the above equation, the size of $[M_b]$ and $[K_b]$ is n x n.

   where, n is number of bending dof

   One can compute,

   $$L = \text{Minimum}(2*\text{m}, \text{m}+8) \tag{5.2}$$

   And $L \leq n$

   Guess the first L Eigen-vectors matrix $[X_1]_{nxL}$

   For k=1, 2, 3,...until convergence is achieved.

2. Solve for $\overline{X}_{k+1}$ from:

   $$[K_b][\overline{X}_{k+1}] = [M_b][X_k] \tag{5.3}$$

3. Find the reduced stiffness and mass matrices from:

$$\left[K^{R}_{b\,k+1}\right]_{LXL} = \left[\overline{X}^{T}_{k+1}\right]_{LXn}\left[K_{b}\right]_{nxn}\left[\overline{X}_{k+1}\right]_{nXL} \tag{5.4}$$

$$\left[M^{R}_{b\,k+1}\right]_{LXL} = \left[\overline{X}^{T}_{k+1}\right]\left[M_{b}\right]\left[\overline{X}_{k+1}\right] \tag{5.5}$$

4. Solve the reduced Eigen problem

$$\left[K^{R}_{b\,k+1}\right]_{LXL}\left[Q_{k+1}\right]_{LXL} = \left[\wedge_{k+1}\right]_{LXL}\left[M^{R}_{b\,k+1}\right]\left[Q_{k+1}\right]_{LXL} \tag{5.6}$$

5. Find the improved Eigen-vectors

$$\left[X_{k+1}\right]_{nXL} = \left[\overline{X}_{k+1}\right]_{nXL}\left[Q_{k+1}\right]_{LXL} \tag{5.7}$$

Then

$$\left[\wedge_{k+1}\right] \rightarrow [\lambda] \equiv eigen-values \;\; and \;\; \left[X_{k+1}\right] \rightarrow [\Phi] \equiv eigen-vectors$$
$$as \;\; k \rightarrow \infty$$

In Eq. (5.3), inverse iteration method is employed. A generalized Jacobi iteration method can be used to solve reduced Eigen equation (5.6). The initial guess Eigen-vector $[X_1]$ should contain independent columns. Generally, when extracting a small number of modes (< 40) in similar size models, the subspace method can be more suitable. It requires relatively less memory but large disk space.

5.2 SPARSE STORAGE SCHEME FOR SYSTEM LINEAR STIFFNESS AND MASS MATRICES

Generally, the stiffness and mass matrices generated for finite element analysis contains so many zeros. In such cases, it is computationally efficient to deal with the only non-zero terms rather than whole matrices. With the same concept, the non-zero terms of the matrices are stored using a technique known as sparse storage scheme.

Using the sub-space algorithm discussed in Section 5.1, linear vibration problem, expressed through Eq. (5.1) is solved for Eigen-solution. Re-writing the equation,

$$\omega_r^{\,2}\,[M_b]\,\{\phi_b\}^{(r)} = [K_b]\,\{\phi_b\}^{(r)}$$

In the above equation,

$[K_b]$ = Square, symmetrical, positive definite (non-singular) system bending stiffness matrix

$[M_b]$ = Square, symmetrical, positive definite (non-singular) system bending mass matrix

$\{\phi_b\}^{(r)}$ = r$^{\text{th}}$ normal mode of the linear vibration problem

The matrices $[K_b]$ and $[M_b]$ are stored using the sparse storage scheme[22] which is most efficient technology, especially for large-scale engineering applications. The following simple example explains the storage scheme effectively.

Example 5.1:

Consider a linear stiffness matrix, $[K_b]$ of size 6 x 6. It is a square, symmetrical and positive definite matrix and is defined as:

$$[K_b] = \begin{bmatrix} 11 & 0 & 0 & 41 & 0 & 52 \\ & 44 & 0 & 0 & 63 & 0 \\ & & 66 & 0 & 74 & 82 \\ & & & 88 & 85 & 0 \\ & SYM & & & 110 & 97 \\ & & & & & 112 \end{bmatrix} \tag{5.8}$$

Normally, the number of storage requirements to store matrix $[K_b]$ is 36. The storage space can be saved if zero terms of the matrix are omitted. The basic of sparse technology is to store only non-zero terms in order to save space and memory occupied by zero terms, which is accomplished by storing non-zero terms of the matrix in form of the vectors as discussed here in detail. The four vectors that store the matrix $[K_b]$ are defined as:

1. Vector IA:

The integer array IA is of size N + 1 x 1. Where, N is the size of the matrix $[K_b]$and IA is described as:

$$IA\begin{pmatrix}1\\2\\3\\4\\5\\6\\7\end{pmatrix}=\begin{Bmatrix}1\\3\\4\\6\\7\\8\\8\end{Bmatrix} \tag{5.9}$$

Here, IA indicates the starting location of the first non-zero and off-diagonal term in each row. It should be noticed that as the matrix is symmetrical, here only upper triangular terms are involved. The location number of the non-zero terms considered in array IA can be shown in matrix $[K_b]$ as:

$$[K_b]=\begin{bmatrix} K_{11} & 0 & 0 & \textcircled{1} & 0 & \textcircled{2} \\ & K_{22} & 0 & 0 & \textcircled{3} & 0 \\ & & K_{33} & 0 & \textcircled{4} & \textcircled{5} \\ & & & K_{44} & \textcircled{6} & 0 \\ & SYM & & & K_{55} & \textcircled{7} \\ & & & & & K_{66} \end{bmatrix} \tag{5.10}$$

Thus, array IA involves information about the number of non- zero off-diagonal terms each row contains, and it can be computed as follows:

The number of non-zero, off-diagonal terms in the 1st row = IA(2) – IA(1) = 3-1=2
The number of non-zero, off-diagonal terms in the 2nd row = IA(3) – IA(2) = 4-3=1
The number of non-zero, off-diagonal terms in the 3rd row = IA(4) – IA(3) = 6-4=2
The number of non-zero, off-diagonal terms in the 4th row = IA(5) – IA(4) = 7-6=1
The number of non-zero, off-diagonal terms in the 5th row = IA(6) – IA(5) = 8-7=1
The number of non-zero, off-diagonal terms in the 6th row = IA(7) – IA(6) = 8-8=0

2. Vector JA :

The integer array JA is of size $NCOEF$ x 1, where $NCOEF$ is the total number of non-zero and off-diagonal terms of matrix $[K_b]$ before factorization. Array JA is described as:

$$JA\begin{pmatrix}1\\2\\3\\4\\5\\6\\7\end{pmatrix}=\begin{Bmatrix}4\\6\\5\\5\\6\\5\\6\end{Bmatrix} \tag{5.11}$$

JA indicates the column number associated with non- zero off-diagonal terms for each row. The following equation shows the associated column number for each non-zero, off-diagonal term of matrix $[K_b]$ clearly.

$$\begin{array}{lc}
\text{COLUMN NO:} & \begin{array}{cccccc}1 & 2 & 3 & 4 & 5 & 6\end{array}\\
[K_b]= & \begin{bmatrix}
K_{11} & 0 & 0 & \textcircled{1} & 0 & \textcircled{2}\\
 & K_{22} & 0 & 0 & \textcircled{3} & 0\\
 & & K_{33} & 0 & \textcircled{4} & \textcircled{5}\\
 & & & K_{44} & \textcircled{6} & 0\\
 & SYM & & & K_{55} & \textcircled{7}\\
 & & & & & K_{66}
\end{bmatrix}
\end{array} \tag{5.12}$$

And,

$NCOEF = IA(\mathrm{N}+1) - IA(1) = 8 - 1 = 7$

3. Vector AD:

The real array AD is of size N x 1 and it is described as:

$$AD\begin{pmatrix}1\\2\\3\\4\\5\\6\end{pmatrix}=\begin{Bmatrix}11\\44\\66\\88\\110\\112\end{Bmatrix} \tag{5.13}$$

Vector AD involves numerical values of all the diagonal terms.

4. Vector AN :

The real array AN is of size $NCOEF$ x 1 and it is described as:

$$AN\begin{pmatrix}1\\2\\3\\4\\5\\6\\7\end{pmatrix}=\begin{Bmatrix}41\\52\\63\\74\\82\\85\\97\end{Bmatrix} \tag{5.14}$$

Vector AN contains all the numerical values of non-zero, off-diagonal terms of upper triangular.

It is obvious in above example 5.1 that number of storage requirement for matrix $[K_b]$ is reduced from 36 to 13 (= 6 for storing diagonal terms +7 for storing off-diagonal terms) by using the sparse storage scheme. Using the same methodology, all the matrices, including linear stiffness matrix, mass matrix, and all nonlinear stiffness matrices defined in Chapter 4, are stored. Moreover, this storage method provides computational ease because of the vector operations instead of matrix operations.

5.3 APPLICATION OF DIRICHLET BOUNDARY CONDITIONS

System stiffness and mass matrices generated by assembling the element level matrices are singular in nature. They become non-singular once the Dirichlet boundary conditions are applied.[22] The method of boundary condition application is explained by the following simple example.

Example 5.2:

Assuming $[K]\vec{w}=\vec{f}$, where size of matrix $[K]$ is N x N = 4 x 4.

And Dirichlet boundary conditions given are:

$w_2 = s_2$ and $w_3 = s_3$

Following equation explains the method to apply the boundary conditions.

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{Bmatrix} w_1 \\ w_2 = s_2 \\ w_3 = s_3 \\ w_4 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{Bmatrix} \tag{5.15}$$

$$\Updownarrow$$

$$\begin{bmatrix} K_{11} & 0 & 0 & K_{14} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ K_{41} & 0 & 0 & K_{44} \end{bmatrix} \begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix} = \begin{Bmatrix} f_1 - K_{12}.s_2 - K_{13}.s_3 \\ s_2 \\ s_3 \\ f_4 - K_{42}.s_2 - K_{43}.s_3 \end{Bmatrix} \tag{5.16}$$

5.4 L D L^T EQUATION SOLVER

The inverse of a matrix is performed using LDL^T equation solver.[22] This method is useful as the matrices are symmetrical and positive definite in nature. The following example explains the method clearly.

Example 5.3:

Consider, the most generalized Finite element equation shown by Eq. (5.17), needs to be solved in order to get displacement vector $\{w\}$

$$[K]\{w\} = \{p\} \tag{5.17}$$

where,

$[K]$ = Square, symmetrical, positive definite (non-singular) system stiffness matrix of size N x N and is known.

$\{w\}$ = System displacement vector of size N x 1 and is unknown

$\{p\}$ = System load vector of size N x 1 and is known

N = Number of dof = 6 (say, for this example)

The solution is sought in three sequential phases as follows:

1. Factorization Phase:

As matrix $[K]$ is square, symmetrical and positive definite, it can be written as:

$$[K]=[L][D][L^T] \tag{5.18}$$

The terms of the matrices of the above equation can be written as:

$$[K]=\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ L_{21} & 1 & 0 & 0 & 0 & 0 \\ L_{31} & L_{32} & 1 & 0 & 0 & 0 \\ L_{41} & L_{42} & L_{43} & 1 & 0 & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & 1 & 0 \\ L_{61} & L_{62} & L_{63} & L_{64} & L_{65} & 1 \end{bmatrix} \begin{bmatrix} D_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & D_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & D_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & D_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & D_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & D_{66} \end{bmatrix} \begin{bmatrix} 1 & L_{21} & L_{31} & L_{41} & L_{51} & L_{61} \\ 0 & 1 & L_{32} & L_{42} & L_{52} & L_{62} \\ 0 & 0 & 1 & L_{43} & L_{53} & L_{63} \\ 0 & 0 & 0 & 1 & L_{54} & L_{64} \\ 0 & 0 & 0 & 0 & 1 & L_{65} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

By equating the upper triangular portion terms of the LHS with the corresponding RHS terms of Eq. (5.19), one can get a sufficient number of equations (= 21) to solve for $[L]$ with 15 unknowns and $[D]$ with 6 unknowns.

If non-zero terms are indicated by X, for the data shown in Eq. (5.19), $[L^T]$ may look like this:

$$[L^T]=\begin{bmatrix} 1 & 0 & 0 & X & 0 & X \\ & 1 & 0 & 0 & X & 0 \\ & & 1 & 0 & X & X \\ & & & 1 & X & \boxed{F} \\ & & & & 1 & X \\ & & & & & 1 \end{bmatrix} \tag{5.20}$$

The symbol, $\boxed{F}$of Eq. (5.20) indicates “fill-in term.” This means during the numerical factorization phase, the zero-term of matrix $[K]$ at a particular location became non-zero. The number of fill-in terms should be kept as low as possible.

2. Forward Substitution:

Substituting Eq. (5.18) into Eq. (5.17),

$$[L][D]\lfloor L^T \rfloor\{w\}=\{p\} \tag{5.21}$$

Say,

$$[D]\lfloor L^T \rfloor\{w\}=\{y\} \tag{5.22}$$

Using Eq. (5.22), Eq. (5.21) can be written as:

$$[L]\{y\}=\{p\} \tag{5.23}$$

As the terms of $[L]$ and $\{p\}$ are known, using Eq. (5.23), $\{y\}$can be evaluated. Schematically, the procedure can be described by following equation (5.24). It is obvious that first of all $y_1$is evaluated and then terms y_2 to y_6 are evaluated by substituting the value of the previous term calculated. That is why the procedure is called “forward substitution.”

$$\begin{bmatrix} 1 & & & & & \\ X & 1 & & & & \\ X & 0 & 1 & & & \\ X & X & 0 & 1 & & \\ X & X & X & X & 1 & \\ X & 0 & X & 0 & X & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{Bmatrix} = \begin{Bmatrix} X \\ X \\ X \\ X \\ X \\ X \end{Bmatrix} \tag{5.24}$$

3. Backward Substitution:

After the forward substitution phase $\{y\}$ is known, it can be substituted in Eq. (5.22) and $\{w\}$can be easily evaluated as $[D]$and $\lfloor L^T \rfloor$is already known. The procedure can be represented as:

$$\begin{bmatrix} X & & & & & \\ & X & & & 0 & \\ & & X & & & \\ & & & X & & \\ & 0 & & & X & \\ & & & & & X \end{bmatrix}\begin{bmatrix} 1 & X & X & X & X & X \\ & 1 & 0 & X & X & 0 \\ & & 1 & 0 & X & X \\ & & & 1 & X & 0 \\ & & & & 1 & X \\ & & & & & 1 \end{bmatrix}\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{Bmatrix} = \begin{Bmatrix} X \\ X \\ X \\ X \\ X \\ X \end{Bmatrix} \qquad (5.25)$$

It is clear from above equation that w_6 will be evaluated first and then terms w_5, w_4, w_3, w_2 and finally w_1 can be calculated by substituting previous term calculated.

5.5 STEP-BY-STEP PROCEDURE OF SPARSE COMPUTATION METHODOLOGY FOR NONLINEAR FINITE ELEMENT ANALYSIS

Step: 1. Input general information that includes element numbers and size, material properties, boundary conditions, nodal co-ordinates, and load applied at the joints.

Step: 2. Input node-element connectivity information.

Step: 3. To store connectivity matrix using sparse technique, generate arrays *IE* and *JE* based on node-element connectivity information. The following example explains the procedure clearly.

Example 5.4:

Consider a simple finite element structure divided in four plate elements as shown in fig.5.1. To simplify the discussion, assume each node has 1 DOF.

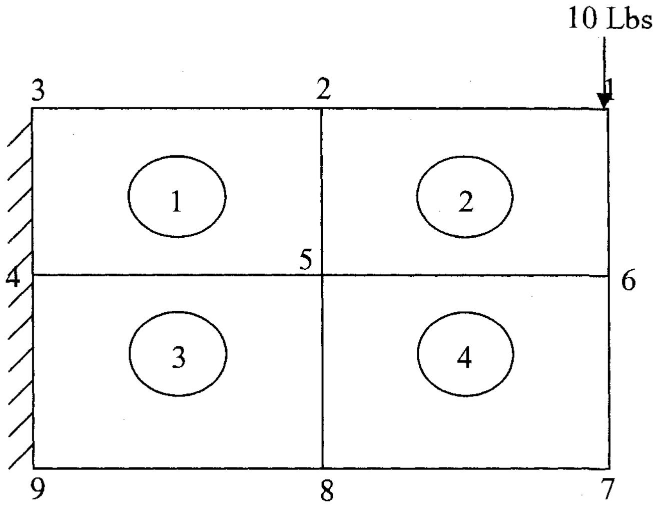


Fig. 5.1: a simple finite element mesh

The element-node connectivity matrix $[E]$ can be expressed as:

$$\begin{array}{cc} & \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ [E]= & \left[\begin{array}{ccccccccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right] \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \end{array} \end{array} \qquad (5.26)$$

It can be noticed that all the non-zero terms have a numerical value equal to 1. So, as discussed in Section 5.1, matrix $[E]$ can be stored using two integer arrays:

1. Array *IE* of size *NEL*+1 x 1

   where,

   NEL = Number of elements = 4

   $\vec{IE}$ = Locations of the first non-zero term of each row

   = 1st, 5th, 9th, 13th, 17th (5.27)

2. Array *JE* of size *NCOEF1* x 1

   where,

   NCOEF1 = NEL*NDOFPE

NDOFPE = Number of dof per element = 4

$\vec{JE}$ = Element-node column index numbers "unordered" for each row

= Global node numbers associated with each element

$$\vec{JE} = |2 \quad 3 \quad 4 \quad 5|;|1 \quad 2 \quad 5 \quad 6|;|5 \quad 4 \quad 9 \quad 8|;|6 \quad 5 \quad 8 \quad 7| \tag{5.28}$$

Step: 4.The coordination of local dof to global dof is stored by creating an array, called *lm*, the size of which is *NDOFPE x 1.*

For example 5.4, element number 1 is connected by nodes 2, 3, 4 and 5.

Hence,

$$lm^{(e=1)} = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 5 \end{pmatrix} \tag{5.29}$$

Step: 5. From the input data related to the load applied on the joints, create a vector $\{b\}$ of size *NDOF* x 1. where, *NDOF* = total number of dof. Initially, b is defined as zero vector. It stores the load intensity at corresponding global dof. For example 5.4, the array b can be written as:

$$b\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{5.30}$$

Step: 6. From the input boundary conditions, create an integer array "iboundc" of size *NDOF* x 1 . The value of the array is 1 where the boundary condition is defined, zero otherwise.

Step: 7. Using subroutine TRANSA2D, generate vectors $\vec{IET}$ and $\vec{JET}$ which are the vectors to store transpose of matrix $[E]$. The size of vector $\vec{IET}$ is $NDOF$ +1 x 1 and that of vector $\vec{JET}$ is $NEL * NDOFPE$ x 1. The requirement to generate the transpose of matrix $[E]$ arises in order to arrange the associated column number stored in vector $\vec{JE}$, in order. The transpose of matrix $[E]$ can be written as:

$$[E^T]=\begin{array}{cc} \begin{array}{cccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} \end{array} & \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \end{array} \tag{5.31}$$

= Node-element connectivity information

Hence,

$$\vec{IET} = 1^{st}, 2^{nd}, 4^{th}, 5^{th}, 7^{th}, 11^{th}, 13^{th}, 14^{th}, 16^{th}, 17^{th} \tag{5.32}$$

= Locations of the first nonzero of each row

$\vec{JET}$ = Node-element-node column index numbers (ORDER) for each row

= Global element numbers associated with each element

$$\vec{JET} = (2), (1, 2), (1), (1, 3), (1, 2, 3, 4), (2, 4), (4), (3, 4), (3) \tag{5.33}$$

It can be noticed from Eq. (5.28) that column index numbers stored by $\vec{JE}$ were unordered. But, Eq. (5.33) shows that column-index numbers are arranged in order by transposing the matrix $[E]$.

Step: 8. Perform symbolic assembly of the structural linear stiffness matrix and mass matrix. Symbolic assembly finds locations of all nonzero, off-diagonal terms of the assembled matrix which helps to predict the computer memory required for numerical assembly of the same matrix. That is why symbolic assembly is always done before numerical assembly is performed. It defines starting locations of the first non-zero off-diagonal terms for each row of structural stiffness matrix after applying boundary conditions, and also provides the column numbers that correspond to each non-zero, off-diagonal term of each row of structural stiffness matrix.

For example 5.4, the size of structural stiffness and mass matrices will be 9 x 9 as NDOF=9. With Dirichlet boundary conditions defined at dof corresponding to node number 3, 4 and 9, the system linear stiffness or mass matrix may be written as:

$$\begin{array}{r|ccccccccc|}
\text{COLUMN NO:} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
 & X & 0 & 0 & 0 & X & 0 & X & 0 & 0 \\
 & 0 & X & 0 & 0 & 0 & X & X & X & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
[K]\ OR\ [M]= & X & 0 & 0 & 0 & X & X & 0 & 0 & 0 \\
 & 0 & X & 0 & 0 & X & X & 0 & X & 0 \\
 & X & X & 0 & 0 & 0 & 0 & X & 0 & 0 \\
 & 0 & X & 0 & 0 & 0 & X & 0 & X & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
\end{array} \tag{5.34}$$

The above matrix can be stored using two arrays, $\vec{IA}$ of size $NDOF$ x 1 and $\vec{JA}$ of size $NCOEF$ x 1 as follows:

$\vec{IA}$ = Starting locations of the first non-zero, off-diagonal terms of each row

=1st, 3rd, 6th, 6th, 6th, 7th, 8th, 8th, 9th, 9th (5.35)

$\vec{JA}$ = Column numbers associated with each non-zero, off-diagonal terms for each row that could be unordered

= (5, 7), (6, 7 ,8), (6),(8) (5.36)

A subroutine called "symbass" performs the symbolic assembly discussed above.

Step: 9. Generate element linear stiffness matrix using Eq.(2.72) and (2.73), mass matrix using Eq. (2.97) and (2.98) and load vector using Eq.(2.99). Then apply numerical assembly to create system linear stiffness matrix, system mass matrix and system load vector using subroutine "numass," discussed in Appendix D. Assembling of linear stiffness matrix, mass matrix and load vector is performed simultaneously in this subroutine. Also, Diritchlet boundary conditions are taken care of inside this subroutine using procedure discussed in Section 5.3. As discussed in Section 5.2, the assembled matrices are stored using sparse storage scheme so the output of the numerical assembly is in the form of several vectors.

Step: 10. From the assembled system stiffness and mass matrix, the terms related to bending DOF and membrane DOF are separated using subroutine "split_sparse_bbmm_improved." This is required as during the Eigen-solution only part of the linear stiffness matrix and mass matrix are utilized, which are related to bending dof as shown by Eq.(4.6).

Step: 11. Depending upon whether the mass matrix is diagonal or lumped, the linear vibration equation, Eq.(4.6) is solved using subroutines "eigsolver011" or "eigsolver022," respectively. The sub-space algorithm discussed in detail in Section 5.1 is applied for Eigen-solution. The output is Eigen-vector matrix $[\Phi]$, the size of which is number of bending dof x number of modes used and Eigen-value vector $\{\lambda\}$ of size number of modes x 1.

Step: 12. Perform mass normalization of the Eigen-vector matrix $[\Phi]$. The mass matrix normalization of Eigen-vector matrix saves time and memory because of following advantages:

1. The modal mass matrix

$$[\Phi_N]^T[M_b][\Phi_N]=1 \tag{5.37}$$

where,

$[\Phi_N]$ =Mass matrix normalized Eigen-vector matrix

2. The modal linear stiffness matrix

$$[\Phi_N]^T[K_b][\Phi_N]=\{\lambda\} \tag{5.38}$$

= Eigen-value vector

The mass matrix normalization procedure, provided only 2 modes are considered, can be described as follows:

$$C_1=(\phi^1)^T[M_b](\phi^1) \tag{5.39}$$

$$C_2=(\phi^2)^T[M_b](\phi^2) \tag{5.40}$$

where, C_1 and C_2 are constants, $[M_b]$ is the mass matrix related to only bending dof, ϕ^1 and ϕ^2 are the Eigen-vectors corresponding to first and second Eigen- values respectively. For the first mode, the normalized Eigen-vector is calculated as:

$$\phi_N^1=\frac{\phi^1}{\sqrt{C_1}} \tag{5.41}$$

Similarly, for second mode, the normalized Eigen-vector is:

$$\phi_N^2=\frac{\phi^2}{\sqrt{C_2}} \tag{5.42}$$

Hence, the normalized Eigen-vector matrix can be written as:

$$[\Phi_N]=\lfloor\phi_N^1 \quad \phi_N^2\rfloor \tag{5.43}$$

Step: 13. Perform symbolic factorization for matrix $[K_m]$ by calling subroutine "symfactd." The purpose of symbolic factorization is to find the locations of all nonzero (including "fills-in" terms), off-diagonal terms of the factorized matrix $[U]$. Matrix $[U]$ has not been evaluated yet and will be generated during the numerical factorization procedure as discussed in Section 5.3. The main goal at this phase is to predict the required computer memory for subsequent numerical factorization. The output of symbolic factorization is stored using two integer arrays in similar manner as discussed during symbolic assembly procedure through step 8.

Step: 14. Matrix $[K_m]$ is transposed twice to put column numbers in order by calling subroutine "transad." The purpose and procedure for this step is already discussed in step 7.

Step: 15. Evaluate matrix $[K2_b]$ that is the first term of the equation for second order nonlinear stiffness matrix expressed through Eq. (4.3). Here, subroutine "BFS_K2_modal" is used.

Rewriting Eq. (4.3),

$$[K_2]=[K2_b]-[K1_{bm}][K_m]^{-1}[K1_{mb}]-[K1_{nm}(\{W_m\}_2)] \tag{5.44}$$

The following sub steps are used to generate matrix $[K2_b]$:

- Get the global dof associated with each element
- Extract only components of the r-th Eigen-vector $\lfloor\Phi^e\rfloor$ that is related to global dof associated with a particular element.
- Using element properties, evaluate element level matrix $\lfloor K2_b^e\rfloor$ using Eq. (2.83) and (2.109)
- Calculate triple product $[\Phi^e]^T[K2_b^e][\Phi^e]$

This is an extremely important and tricky procedure. Instead of computing the triple product $[\Phi]^T[K2_b][\Phi]$ which is at system level, $[\Phi^e]^T[K2_b^e][\Phi^e]$is evaluated at element level. This approach saves a lot of memory and computational time as well.

- Assemble the triple product $[\Phi^e]^T[K2_b^e][\Phi^e]$ to generate system level matrix $[K2_b]$

Step: 16. Evaluate $[K1_{bm}]$ and compute the second term of Eq. (5.44), which is: $[K1_{bm}][K_m]^{-1}[K1_{mb}]$

The following sub steps are used for the calculations:

- Get the global dof associated with each element
- Extract only components of the r-th Eigen-vector $\lfloor\Phi^e\rfloor$ that is related to global dof associated with a particular element
- Generate element level matrix $\lfloor K1_{bm}^e\rfloor$ based on Eq. (2.81) and Eq. (2.107)
- Convert element level matrix $\lfloor K1_{bm}^e\rfloor$ from 2-D array into 1-D array (column-wise)
- To save time and memory, as discussed in step 15, evaluate $[\Phi^e]^T[K1_{bm}^e][\Phi^e]$ by calling subroutine " mvsparse"
- Perform numerical assembly of the $\lfloor K1_{bm}^e\rfloor$
- Sparse "numerical" factorization of matrix $[K_m]$ using subroutine "numfa1d." The factorization method used is as discussed in Section 5.3.
- Sparse forward/backward solution phase as per Section 5.3, using subroutine "fbed"
- Perform the multiplication:

$$\text{K1\_modal} = [K1_{bm}][K_m]^{-1}[K1_{mb}] \tag{5.45}$$

Step: 17. Generate third term of Eq. (5.44), $[K1_{nm}(\{W_m\}_2)]$ by calling subroutine "BFS_K2nm_modal"

The sub steps for the computation are as follows:

- To Generate matrix $[K1_{nm}(\{W_m\}_2)]$, $\{W_m\}$ is generated based on W_b using the following equation:

$$[K_m]^{-1}[K1_{mb}]^s\{W_b\} \tag{5.46}$$

- In previous step 16, $[K_m]^{-1}[K1_{mb}]^r[\Phi]$ is already calculated. Using the same procedure, calculate $[K_m]^{-1}[K1_{mb}]^s\{W_b\}$

- From system level vector $\{W_m\}$,element level vector $\{W_m^e\}$ is evaluated
- Using $\{W_m^e\}$, evaluate element level $\lfloor K2_{nm}^e \rfloor$ using Eq. (2.80), Eq. (2.108), and Eq. (2.120)
- To save time and memory, as discussed in step 15, evaluate $[\Phi^e]^T[K2_{nm}^e][\Phi^e]$ by calling the subroutine "mvsparse"
- Assemble the triple product $[\Phi^e]^T[K2_{nm}^e][\Phi^e]$ to generate system level matrix $\lfloor K2_{nm} \rfloor$

Step: 18. Using the terms evaluated through steps 15, 16 and 17, compute the nonlinear stiffness matrix equation shown through Eq. (5.44).

Step: 19 Random pressure is generated using Shinozuka's method, explained in Chapter 3. The procedure is shown in Appendix B, and load vector is rescaled as expressed in Appendix D. Here, user defined cut-off frequency and Sound Pressure Level (SPL) is taken into account for random load generation. In case of un-synchronized loading the task of random load generation is distributed among various processors. Details about the difference in the procedure for different load cases are discussed in the next chapter.

Step: 20 Using the Runge-Kutta time Integration scheme, the modal equation of motion shown by Eq. (4.19) is solved to obtain the random responses, such as displacement, velocity and acceleration in modal coordinates. The modal responses are transformed back to the original structural dof through equation (4.36). Based on bending displacements obtained, the in-plane displacement vector is calculated using equation (4.37).

Step: 21 Calculate RMS of deflection as discussed in detail through Section 4.4, 4.5 and 4.6 of Chapter 4. Calculate the modal participation factor as per discussion in Section 4.11. When all convergence criteria discussed in Section 4.9 are satisfied, the RMS of maximum deflection is calculated. Here, data manipulation discussed in Section 4.9 is also accounted for.

CHAPTER VI

PARALLEL COMPUTATIONS

While dealing with large scale structural analysis and design problems, considerable computational effort is required. By implementing parallel processing techniques, such problems can be solved without resorting to the use of expensive computing equipment or incurring an inordinately high computational cost. Here, basics of parallel computation are discussed along with the needs and advantages to use for specific types of problems.

6.1. BASICS OF PARALLEL COMPUTATION

Modern High performance computers have multiple processing capabilities which become extremely useful while solving large-scale problems. The distribution of the computational task among the multiple processors saves huge amount of time and memory. Distributed memory computers, in general, consist of many processors or nodes. Each one has its own local memory and is strong in terms of speed and memory compared to the processor itself. As shown in fig 6.1, the communication among the processors can be done by message passing.

Typically, as numbers of processors are increased, the time consumed to perform the computation should decrease. In practice, this is true up to certain number of processors. Up to certain extent, if the numbers of processors are increased the computation process speeds up. After that even though the processors are increased, there is no significant contribution in time saving. Such performance of parallel computing will be shown in later chapters through practical implementation.

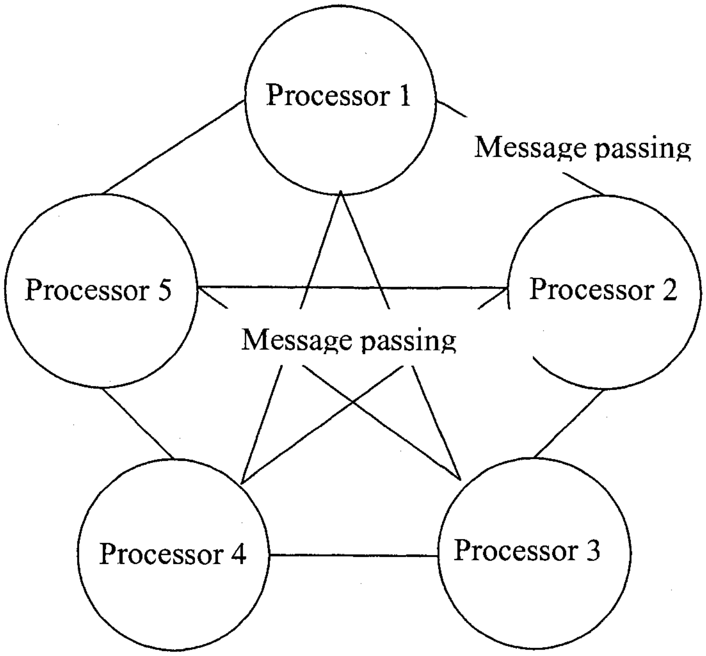


Fig 6.1: Message passing in parallel computers

6.2 APPLICATION OF PARALLEL COMPUTING IN CASE OF UNSYNCHRONIZED LOAD CASE

Computational burden due to the simulation of the unsynchronized random pressure fluctuations makes parallel computing an extremely important application.

6.2.1 Large Scale Problem Solving

As discussed earlier, in Section 3.4, in case of unsynchronized load case, the time history needs to be generated separately for each finite element. For problems with large mesh size, the numbers of elements are high and the generation of time histories makes the problem complex in terms of time and/or memory. And sometimes it is impossible to resolve. Also, as discussed in Section 3.7, once matrices are generated using the method by Shinozuka, the random load vector needs to be rescaled in order to adjust same SPP value using MATLAB function "pwelch." The rescaling procedure is quite time consuming as it involves using MATLAB inbuilt function called "pwelch" in the FORTRAN environment.

As already discussed, for unsynchronized loading simulation, each finite element is excited by a random load of the same S_0 or SPL generated from a different ISEED number. It becomes important to check that time histories are completely independent to one another. A more detailed analysis using the correlation coefficient command in MATLAB has indicated that the pressure time histories were statistically uncorrelated to each other as shown in Fig. 6.2.[53] The correlation coefficient values of 10 generated samples of pressure time history are plotted. They are generated using 10 different ISEED numbers and denoted by different line style or marker type. For instance, if the first pressure time history is compared with the other 9, the lowest correlation coefficient found was 0.0011 and the highest is 1. A coefficient of 1 gives total correlation, and the two loads are similar in every sense, whereas a coefficient of 0 means lack of any correlation at all. A low value of the correlation factor also indicates that all time histories are not periodic with each other.

6.2.2 Advantages

Computational burden due to the simulation of the unsynchronized random pressure fluctuations requires the use of parallel computing capabilities. Without parallel computation, it is almost impossible to solve some large-scale problems, when the load subjected is considered un-synchronized in pattern. The usage of parallel computing can be considered the only efficient option to solve such problem. Even for medium-scale sequential problems, usage of parallel computation becomes extremely important, as it reduces the time dramatically.

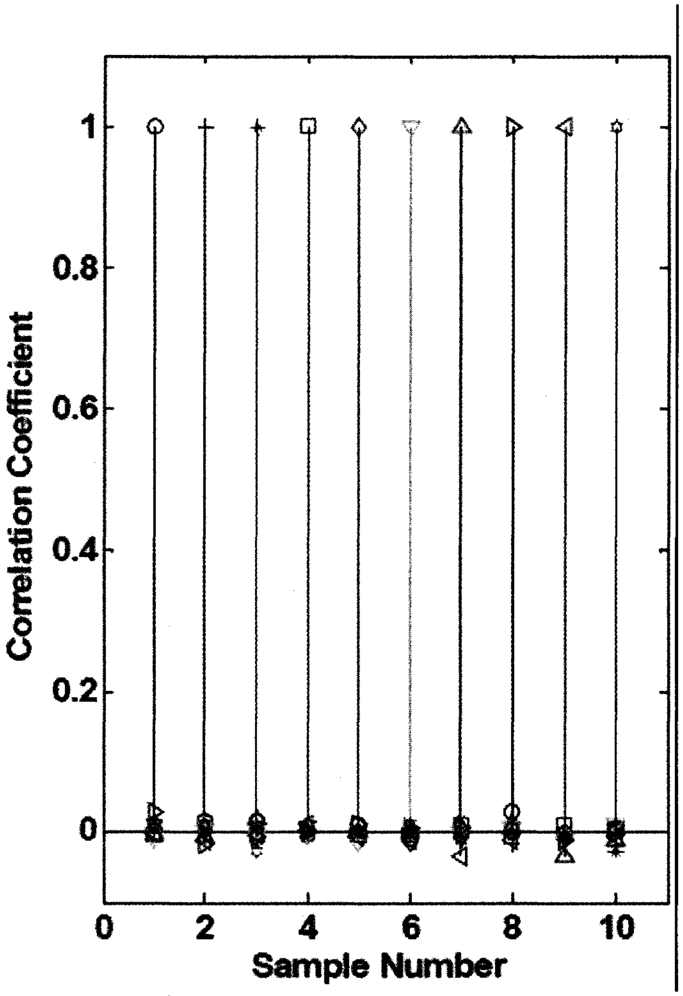


Fig. 6.2: Correlation coefficients among 10 different random time histories[53]

6.3 BASIC CONCEPT: DIFFERENCE IN RANDOM LOAD VECTOR GENERATION

The finite element analysis phenomena discussed in Chapters 2 and 4 is same whether the load subjected is synchronized or unsynchronized except the generation of load vector. From a computational aspect, the differences in the procedure when dealing with these two loading types are detailed here.

As shown by Eq. (4.1), the equation of motion in terms of only bending dof is

$$[M_b]\{\ddot{W}_b\}+([K]+[K_2])\{W_b\}=\{P_b\} \tag{6.1}$$

The random force vector, $\{P_b\}$ of equation (6.1) can be evaluated as:

$$\{P_b\} = \int_A [T_b]^T [H_w]^T p(x,y,t) dA \tag{6.2}$$

As defined in Chapter 2, $[T_b]$ and $[H_w]$ are transformation matrix and displacement function matrix, respectively. And $p(x,y,t)$ is generated random load intensity, which is a function of space and time.

As previously discussed, in case of synchronized load, $p(x,y,t)$ remains the same for each element. It needs to be generated only once, using one ISEED number. In such a case, the element load vector remains the same for all the elements, as $[T_b]*[H_w]$ indicates shape function, which is also the same for all the elements. Whereas, in case of unsynchronized load case, $p(x,y,t)$ is different for all the elements, which means element load vector is different for each element.

6.4 STEP BY STEP PROCEDURE OF LOAD VECTOR GENERATION USING PARALLEL COMPUTATION IN CASE OF SYNCHRONIZED LOAD CASE

In case of synchronized loading, parallel computation doesn't need to be involved, as the random pressure is generated only once during the entire analysis procedure. The step-by step procedure to generate random load vector $\{P_b\}$, when the load considered is synchronized in pattern is as follows:

Step: 1. Evaluate part of the element load vector using shape functions. As per Eq. (2.99), element load vector,

$$\{p_b\} = \int_A [T_b]^T [H_w]^T P_{random} \quad dA$$

Firstly, the element unit load vector is evaluated to distribute the load uniformly among all the nodes of the element. The unit load vector can be defined as:

$$\{p_u\} = \int_A [T_b]^T [H_w]^T dA \tag{6.3}$$

Step: 2. Assemble the element load vector at system level to generate $\{P_u\}$.This can be done simultaneously during numerical assembly procedure of linear stiffness and mass matrices.

Step: 3. Evaluate modal unit load vector,

$$\{\overline{P}_u\} = [\Phi]^T \{P_u\} \tag{6.4}$$

Step: 4. Generate the random load time history $\{P_{random}\}$ using ISEED number only once by following the code in appendix B.

Step: 5. Rescale the random load vector by following the step-by-step procedure discussed in Section 3.7. Appendix D contains the FORTRAN code used for rescaling purposes. It involves calling MATLAB in-built function "pwelch."

Step: 6. For nth time history point, the load considered is $\{\overline{P}_u\} * \{P_{random}(n)\}$. As $\{P_{random}(n)\}$ is a scalar quantity, the size of the random load vector will be number of modes x 1.

6.5 STEP-BY-STEP PROCEDURE OF RANDOM LOAD VECTOR GENERATION USING PARALLEL COMPUTATION IN CASE OF UN-SYNCHRONIZED LOAD CASE

For unsynchronized load case, as each element has its own time history, it is obvious that the procedure discussed in Section 6.3 can not be followed. The step-by-step process for generation of un-synchronized random load can be described as follows:

Step: 1. Generate the random load time history $\{P_{random}\}$ by following the code in appendix B for all the elements using different ISEED number each time.

Step: 2. Rescale the random load vectors generated for all the elements by following the step-by-step procedure discussed in Section 3.7. Appendix D contains the FORTRAN code used for rescaling purpose. It involves calling MATLAB in-built function "pwelch."

Step: 3. Calculate element load vector as,

$$\{P_b^e\}^n = \int_A [T_b]^T [H_w]^T \{P_{random}\}^n dA \qquad (6.5)$$

In the above equation, superscript n indicates corresponding element number.

Step: 4. Assemble the element load vector at system level to generate $\{P_b\}$. Here, the procedure involves taking care of random load intensity at a particular time history point for each element. So, the assembling is done separately and not with the numerical assembly of linear stiffness and mass matrices.

Step: 5. Evaluate modal load vector using Eq. (4.23),

$$\{\overline{P}_b\} = [\Phi]^T \{P_b\} \qquad (6.6)$$

The pseudo FORTRAN code to generate unsynchronized modal load vector can be described as follows:

```
Do $I=1, NPT$
    Do $J=1, NEL$
        Do $K=1, MAXDOF$
            $P_b^J(J) = p_u(J) + Random\_load(I,J)$
        Enddo
    Enddo
```

- Assemble vector $\{P_b^e\}$ to generate system unit load vector $\{P_b\}$

```
        • Evaluate $\{\overline{P}_b\} = [\Phi]^T \{P_b\}$
    Enddo
```

where,

NPT	= Number of time history points
NEL	= Total number of elements
$MAXDOF$	= Maximum dof per element
$\{P_b^e\}$	= Element load vector
$\{p_u\}$	= Element unit load vector
$[Random\_load]$	= Random load matrix to store random load vectors of all the elements
$[\Phi]$	=Eigen-vector matrix

Here, random load matrix $[Random\_load]$ stores time histories of all the elements. In other words, $\{P_{random}\}$ stores time history which is different for each element and $[Random\_load]$ stores $\{P_{random}\}$ for all the elements. Thus, the size of $[Random\_load]$ is number of points (which are generally large, 16384 in our case) X number of elements (which is a large number in the case of large scale problem). The computationally expensive task of generating as well as scaling random load vector for all the elements can be shared using different processors/nodes via parallel computing. If noticed carefully, even though random load time history vector $\{P_{random}\}$is of size NPT x 1, from matrix$[Random_load]$, only a scalar value needs for the calculation at a time. So, all the processors/nodes generate matrix $[Random\_load]$ first and then during calculations required data is pulled from the pre-generated matrix$[Random_load]$. As each processor works independently the communication time between the processors is zero, which saves a significant amount of time.

Message Passing Interface (MPI) FORTRAN language is used to accomplish the above task. MPI FORTRAN in-built subroutine mpi_wtime( ) is used to note wall clock time consumed by each processor as well as for different segments of the whole procedure.

CHAPTER VII

NUMERICAL EXAMPLES

Efficient computational technologies like sparse storage schemes and parallel computation are proposed and incorporated to solve large-scale, nonlinear large deflection random vibration problems for both type of loading cases: 1) synchronized in time and 2) un-synchronized and statistically uncorrelated in time. Finite element nonlinear modal formulation in conjunction with the time domain Monte Carlo simulation is used. Moreover, the linear and nonlinear matrices are stored using sparse storage schemes in order to save computational time and memory. In case of un-synchronized load case, the time history needs to be generated and also rescaled separately for each finite element. For problems with large mesh size, the numbers of elements are high and the generation of time histories makes the problem unsolvable (in terms of computational time and/or memory requirements), for all practical purposes. By implementing parallel processing techniques, large scale structural analysis problems are solved without resorting to the use of expensive computing equipment or incurring an inordinately high computational cost.

The FEM approach has been verified:

1. By comparing the nonlinear modal coefficients with those obtained using the PDE/Galerkin analytical solution[20] as shown in Table 7.1
2. By Experimental data of random nonlinear vibration of clamped beams[15] shown in Fig. 7.1.

Table 7.1 Comparison of nonlinear coefficients using classical PDE/Galerkin and FEM methods for a 14x10x0.04 in. simply supported panel[20]

	q_{11}^3	$q_{11}^2 q_{31}$	$q_{31}^2 q_{11}$	q_{31}^3
		Classical PDE/Galerkin		
1st Eq	1.2966E12	-8.2707E11	5.2128E12	0.0
2nd Eq	-2.7569E11	5.2128E12	0.0	2.1139E13
Mesh*		**FE Modal Method**		
8x8	1.3026E12	-8.2691E11	5.3487E12	1.6572
	-2.7569E11	5.3487E12	4.9357	2.2128E11
12x12	1.2992E12	-8.2699E11	5.2740E12	2.2294
	-2.7566E11	5.2740E12	6.6420	2.1593E13
16x16	1.2981E12	-8.2702E11	5.2473E12	54.8464
	-2.7567E11	5.2473E12	16.4525	2.1397E13

*Mesh sizes are in quarter of a plate

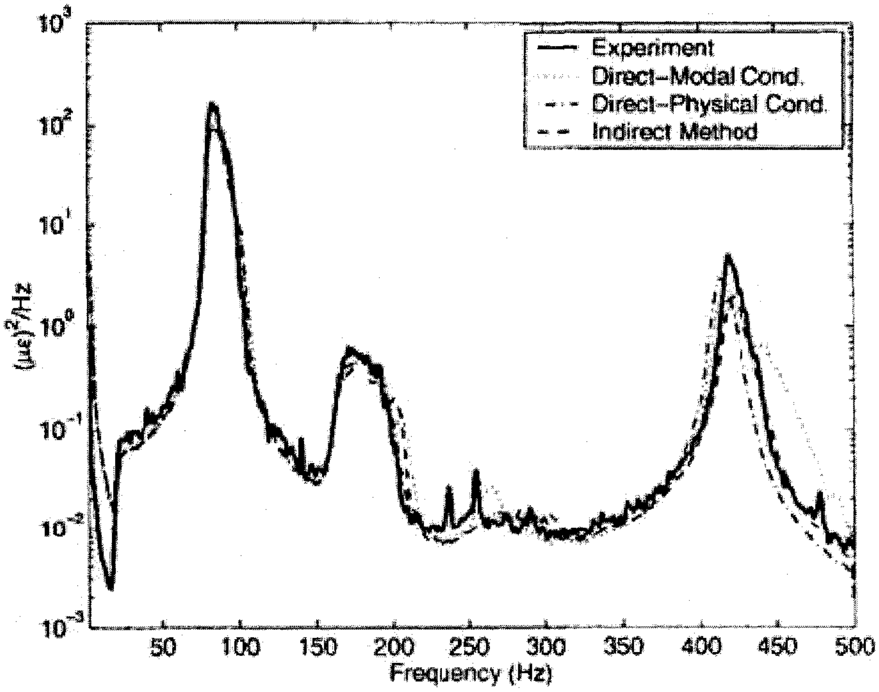


Fig. 7.1: Comparison of strain PSD among experiment aluminum panel at three overall SPL[10] and three FE methods for nonlinear random vibration of a clamped beam[15]

Based upon the nonlinear modal, sparse formulations discussed in the previous chapters, the following examples are used to validate the numerical accuracy and performance of the developed nonlinear time-dependent random response.

Problem Statements:

A simply supported isotropic plate with immovable in-plane conditions $(0, y) = u(a, y) = v(x,0) = v(x,b) = 0$ is studied in detail. The plate size is 14 in x 10 in x 0.04 in. Only a quarter of a plate is modeled using the extended BFS (Bogner-Fox-Schmit) elements. The material properties are: Elastic modulus E=10.587 psi (73 Gpa); Poisson's ratio v = 0.3; Density ρ = 2.588 x 10^{-4} lbf-sec^2/in^4 (2763 kg/m^3). A proportional damping ratio of $\xi_r\omega_r = \xi_s\omega_s$, with ξ_1=0.02 is used.

7.1 SYNCHRONIZED LOAD CASE

When load is synchronized in nature, random pressure history needs to be generated only once, as it remains the same for all the elements. Thus, the application of parallel computation is not competent to use for synchronized load case. For small and medium scale problems, Time step = 9.4 x 10^{-8} Sec is used. This time step has been automatically computed by Abaqus and therefore is also adopted in our proposed nonlinear sparse modal method for comparison purpose. Based on the number of dof, the plate is modeled in three categories: small scale, medium scale and large scale.

7.1.1. Synchronized Load Case: Small Scale Problem with 16 x 16 Mesh Size

In order to verify the results with the available data,[13] the small scale problem is solved first with the mesh size of 16 x 16. Data used to solve the small-scale problem is shown in table 7.2.

Table 7.2: Data used for small-scale problem (16 x 16) mesh size

Mesh size	= 16 x 16
No. of elements	= 256
No. of nodes	= 289
Total no. of dof	= 1734
No. of modes used	= 4
Total active dof	= 1534
Time history	= 2 Secs
Time step	= 9.4 x 10^{-8} Sec
Sound Pressure Level	= 120
Cut off Frequency	= 1024 Hertz

When seven lowest modes are taken in account, the lowest natural frequencies are given and compared with the exact values through Table 7.3.

Table 7.3: The lowest seven natural frequencies (in Htz) of simply supported plate

Mode	**(1,1)**	**(3,1)**	**(1,3)**	**(3,3)**	**(5,1)**	**(5,3)**	**(1,5)**
Exact	58.116	215.19	365.98	523.05	529.33	837.19	981.70
FEA	58.118	215.19	365.98	523.05	529.35	837.21	981.74

Using the Runge-Kutta time integration method, the modal coordinates $\{q\}$ for converged deflection are first calculated by solving, the modal equation of motion expressed through Eq. (4.19). Then, for each time history point, bending displacement vector $\{W_b\}$can be easily computed using Eq. (4.36). Table 7.4 shows the Root Mean Square (RMS) of maximum non-dimensional deflection at Sound Pressure Level, SPL= 120 and compares with data available in the literature. Here, the average RMS is calculated after solving the same problem for 16 different samples. Each sample has a different ISEED number to generate the random load. It has been observed that 4 modes are sufficient to get the converged deflection. The time history is generated for 2 seconds. Then, to avoid the effect of initial displacement (if any), the initial history of 0.4 seconds is neglected. Time taken by different segments of the developed FEA code is shown in Table 7.5.

Table 7.4: Verification of results for a simply supported isotropic plate

SPL	**RMS of Wmax/h for time history of 1.6 sec**	**RMS of time Wmax/h for history of 2 .0 Sec from**[13]
120	1.496925555	1.4039

Note: The small difference in the results is because of: 1) In Ref[13], the time history is considered for a full 2 sec, whereas our RMS calculations are based on 1.6 sec. 2) Random loading patterns are different in this work and in Ref[13].

Table 7.5: Time consumed by various segments of sparse FEA code for synchronized loading using 16 x 16 mesh size

Time Zone	**Function**	**Time(Minutes)***
1	Read input data	0.00010
2	Linear stiffness and mass matrix generation along with symbolic and numerical assembly	0.002595
3	Splitting into bending and membrane dof	0.000035
4	Sparse Sub-space eigen solution	0.007757
5	Modal calculations and mass normalization	0.000028
6	Factorization	0.000020
7	First and second order nonlinear stiffness matrices generation	0.340602
8	Random load generation and re-scaling	**0.076090**
9	Runge Kutta integration, RMS deflection and modal participation factor calculations	0.019640
	Total time :	**0.446867**

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

7.1.2 Synchronized Load Case: Medium Scale Problem with 192 x 192 Mesh Size

The mesh size of 192 x 192 is used here to compare the computational time used by proposed nonlinear sparse modal method with the time taken by Abaqus for the same problem. Tables 7.6 and 7.7 provide data used and timing results for this medium scale problem, respectively.

Table 7.6: Data used for medium-scale problem (192 x 192 mesh size)

Mesh size	= 192 x 192
No. of elements	= 36864
No. of nodes	= 37249
Total no. of dof	= 223494
No. of modes used	= 4
Total active dof	= 221182
Time history	= 2 Secs
Time step	= 9.4 x 10^{-8} Sec
Sound Pressure Level	= 120
Cut off Frequency	= 1024 Hertz

Table 7.7: Time consumed by various segments of sparse FEA code for synchronized loading using 192 x 192 mesh size

Time Zone #	**Function**	**Time Proposed Sparse Method (Minutes)***	**Time Abaqus (Minutes)****
1	Read input data	0.00287287	0.317
2	Linear stiffness and mass matrix generation along with symbolic and numerical assembly	0.29432437	-
3	Splitting into bending and membrane dof	0.00493162	-
4	Sparse Sub-space eigen solution	21.1467037	-
5	Modal calculations and mass normalization	0.00262425	-
6	Factorization	0.02210662	-

7	First and second order nonlinear stiffness matrices generation	41.6832000	-
8	Random load generation and re-scaling	0.021324	-
9	Runge Kutta integration, RMS deflection and modal participation factor calculations	929.235428	
	TOTAL TIME :	**993**	**70967**

* Wall-Clock time was reported.(ODU computer's model: Sun-Fire-280R,speed=1.2 GHz , Operating system: solaris OS system, sparc version 8.0)

**Wall clock time was reported (Dell Precision 370 Workstation. Intel Pentium 4, 3.2 GHz, 3 GB RAM)

7.1.3 Synchronized Load Case: Large Scale Problem with 256 x 256 Mesh Size

The mesh size of 256 x 256 is used here to evaluate the numerical performance of the code in terms of the computational time and memory requirements for solving the largest size problem using available computational resources. Tables 7.8 and 7.9 provide data used and timing results for this large scale problem, respectively.

Table 7.8: Data used for large-scale problem (256 x 256 mesh size)

Mesh size	= 256 x 256
No. of elements	= 65536
No. of nodes	= 66049
Total no. of dof	= 396294
No. of modes used	= 4
Total active dof	= 393214
Time history	= 2 Secs
Time step	= 1.2207×10^{-4} Sec
Sound Pressure Level	= 120
Cut off Frequency	= 1024 Hertz

Table 7.9: Time consumed by various segments of sparse FEA code for synchronized loading using 256 x 256 mesh size

Time Zone	**Function**	**Time(Minutes)***
1	Read input data	0.005248
2	Linear stiffness and mass matrix generation along with symbolic and numerical assembly	0.636123
3	Splitting into bending and membrane dof	0.008153
4	Sparse Sub-space eigen solution	33.098150
5	Modal calculations and mass normalization	0.006952
6	Factorization	0.047998
7	First and second order nonlinear stiffness matrices generation	134.798800
8	Random load generation and re-scaling	0.028494
9	Runge Kutta integration, RMS deflection and modal participation factor calculations	7.067228
	Total time :	**175.697147**

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

7.2 UNSYNCHRONIZED LOAD CASES

The same problem discussed in the previous section of this chapter is solved when the load subjected is unsynchronized in nature. Table 7.10 provides the details about time taken by various segments of the problem when only a single processor is used. To solve the same problem, parallel computation is adopted. Here, the load is simulated using different ISEED numbers for each element, which makes application of parallel computation beneficial. First of all, the major time consuming part is identified, and then parallel computation is adopted to distribute the task for that particular segment. Here, Time step = 1.2207 x 10^{-4} Sec is used as per the proposed code requirement. Comparison in timing shown by Table 7.5 and 7.10 confirms the necessity of parallel computation for the part of the code that generates and re-scales random load. Small, medium and large scale problems are solved to check efficiency of parallel computation for different scale problems.

Table 7.10: Time consumed by various segments of sparse FEA code for unsynchronized loading using 16 x 16 mesh size

Time Zone	**Function**	**Time(Minutes)***
1	Read input data	0.000107
2	Linear stiffness and mass matrix generation along with symbolic and numerical assembly	0.002437
3	Splitting into bending and membrane dof	0.000032
4	Sparse Sub-space eigen solution	0.006382
5	Modal calculations and mass normalization	0.000027
6	Factorization	0.000018
7	First and second order nonlinear stiffness matrices generation	0.311050
8	Random load generation and re-scaling	**5.062069**
9	Runge Kutta integration, RMS deflection and modal participation factor calculations	0.021730
	Total time :	5.403851

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

In case of un-synchronized loading the random load generation, along with rescaling, consumes an ample amount of time. This can be noticed by comparing time zone 8 of Table 7.5 and 7.10. One can easily visualize the increase in the time difference for rescaled random load generation in case of a large scale problem.

7.2.1 Un-synchronized Load Case: Small Scale Problem with 16 x 16 Mesh Size

In order to verify the results, the small scale problem is solved first with the mesh size of 16 x 16. The program is used to run for synchronized loading, i.e., by providing a similar time history for each element. Comparison of RMS deflection proves the accuracy of the code. Results show that the Root Mean Square deflections are reduced for the unsynchronized loading as compared with the traditional synchronized loading case. Table 7.11 shows time consumed by various numbers of processors to solve the problem.

Fig 7.2 helps in visualizing the data shown in table 7.11 and efficiency gained using 31 processors.

Table 7.11: Time consumed by different number of processors using sparse-parallel FEA code to solve small scale problem with un-synchronized loading using 16 x 16 mesh size

NP	Time to solve whole problem (Mins)*	Time for parallel segment (Mins)*	Speed Up**
1	7.678	7.340	1.00
2	4.013	3.893	1.89
3	3.038	2.700	2.72
5	2.058	1.737	4.23
7	1.708	1.370	5.36
11	1.397	1.059	6.93
15	1.177	0.840	8.74
27	1.048	0.71	10.34
31	0.988	0.65	11.29

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

** Speed up is calculated based on timing for parallel segment of the program (generation and re-scaling of un-synchronized load)

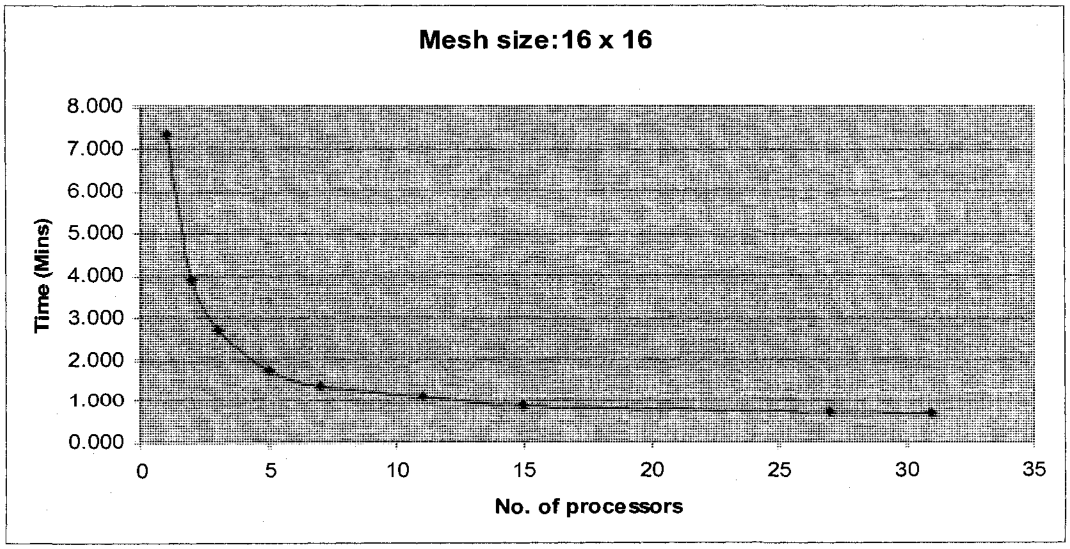


Fig. 7.2: Graph of No. of processors Vs. Time (minutes) for small scale problem with un-synchronized load using 16 x 16 mesh size

7.2.2 Unsynchronized Load Case: Medium Scale Problem (96 X 96 Mesh Size)

The medium size problem has been solved to compare the efficiency of parallel process for different size problems. Table 7.12 shows time consumed by various numbers of processors to solve the problem. Fig 7.3 helps in visualizing the data shown in table 7.12.

Table 7.12: Time consumed by different number of processors using sparse-parallel FEA code to solve medium scale problem with un-synchronized loading Using 96 x 96 mesh size

NP	Time to solve whole problem (Mins)*	Time for parallel segment (Mins)*	Speed-Up**
1	307.685	292.495	1.00
2	162.470	148.300	1.97
3	118.940	104.750	2.79
5	82.450	68.198	4.29
7	65.170	51.000	5.74
11	49.589	36.743	7.96
15	44.196	30.186	9.69
27	40.130	26.150	11.19
31	37.630	23.650	12.37

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

** Speed up is calculated based on timing for parallel segment of the program (generation and re-scaling of un-synchronized load).

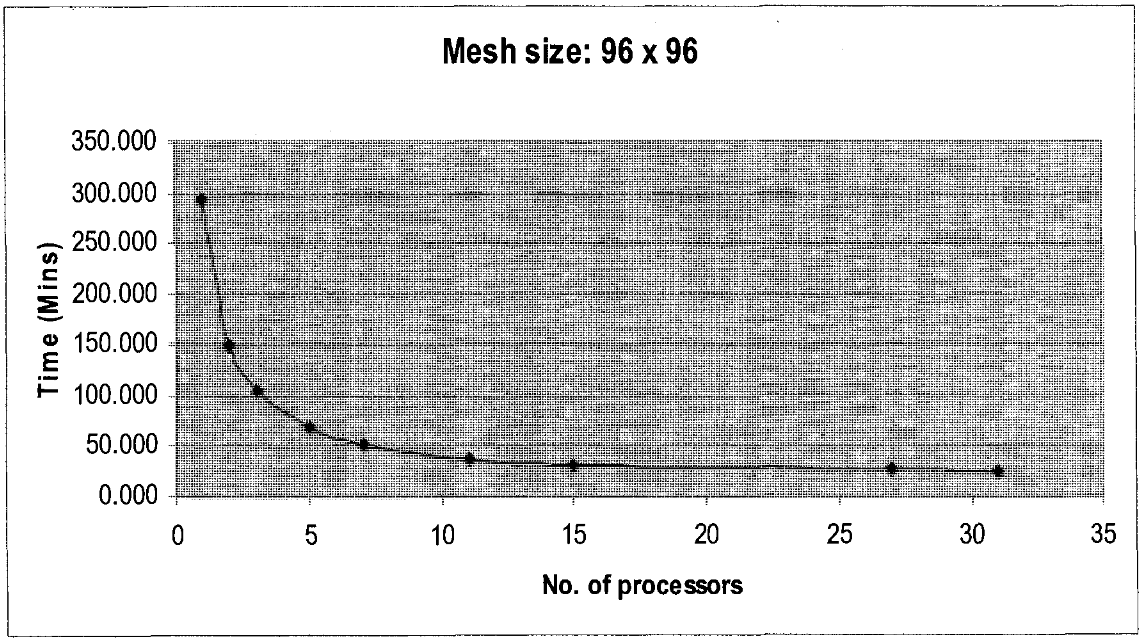


Fig. 7.3: Graph of No. of processors Vs. Time (minutes) for medium scale problem with un-synchronized load using 96 x 96 mesh size

7.2.3 Unsynchronized Load Case: Large Scale Problem (192 x 192 Mesh Size)

It is obvious that the bigger the problem size, the more benefit from parallel computation in terms of time saving. Using available resources, maximum size problem solved is of mesh size 192 x 192 with 223,494 dof. The large scale problem subjected to un-synchronized load is solved using the modal FEA code combined with parallel computation. Table 7.14 gives a time comparison using a different number of processors to solve the whole problem. Fig.7.4 helps in visualizing the data shown in table 7.14. It should be noticed that for this problem, memory restriction does not allow usage of fewer than 12 processors.

Table 7.13: Time consumed by different number of processors using sparse-parallel FEA code to solve large scale problem with un-synchronized loading using 192 x 192 mesh size

NP	Time to solve whole problem(Mins)*	Time for parallel segment (Minutes)*
12	340.042	269.042
16	286.513	213.813
31	189.233	116.333

* Wall-Clock time was reported. (Wright Brothers HPC Environment, small parallel environment: Sun Fire V20z cluster, 2.4 GHz, 4 GB RAM)

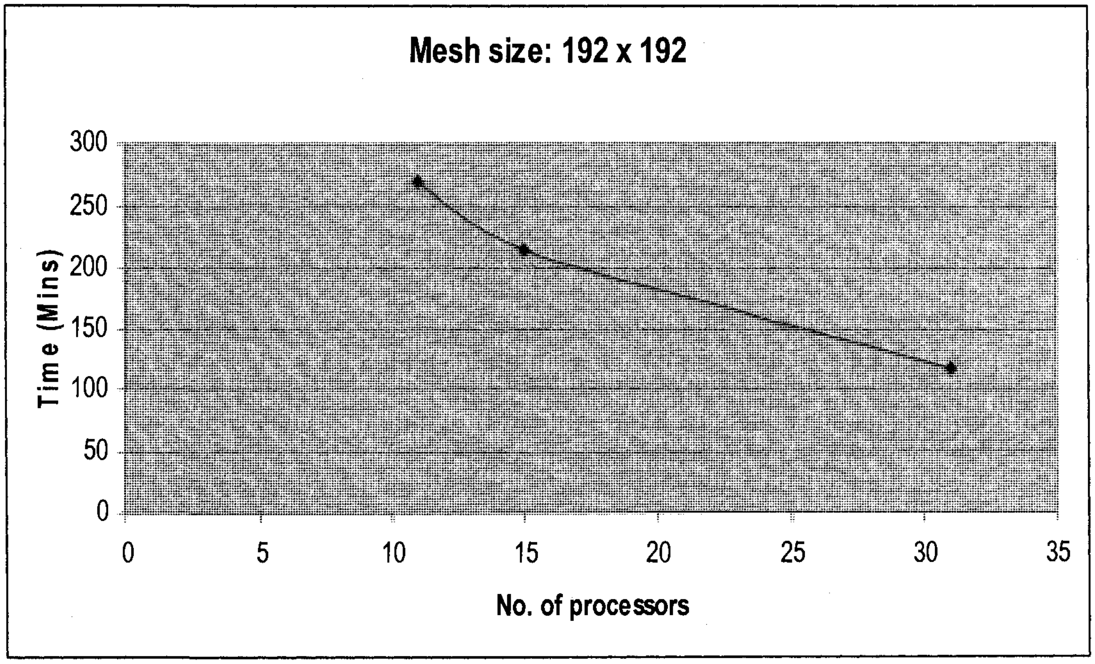


Fig.7.4: Graph of No. of processors Vs. Time (minutes) for medium scale problem with un-synchronized load using 192 x 192 mesh size

7.3 USER INPUT DATA

The input file is generated separately to provide user defined input data. Table 7.14 shows details about how required input data is defined for sparse FEA code.

Table 7.14: Input data requirement

Required Input Data:	Defined in the code as:
Total number of boundary conditions	Nboundc
Number of dof per node	Ndofpn
Total number of nodes	Numnodes
Number of dof on which external load is applied	Loadof
Number of materials	Nummat
Number of layers	Nlayers
Number of sectional properties	Nsect
Flag to indicate whether mass matrix is lumped or diagonal	Lumpmass
Flag to indicate Eigen solver	Ianal
Number of eigen values	Neig
Flag to indicate algorithm to be used for reordering	Nreord
Level of unrolling	Nunroll
Defined value to perform shift in eigen solver	Ishift
Flag to provide printing information on output of eigen solver	Iprint
Number of elements	Nel
Number of dof per element	Ndofpe
Young's Modulus	E
Area of the element surface	A
Nodal co-ordinates	joint# x-coord y-coord z-coord
Element-node connectivity	element# node-1
Load intensity subjected on nodes	Loaded-dof

The input file needs to be in sequential order as follows:

Table 7.15: Order required for input data

nboundc ndofpn numnodes loadof nummat nlayers nsect lumpmass ianal neig nreord nunroll ishift iprint
Nel (1,2,3,4,5,6)
ndofpe (1,2,3,4,5,6)
young-modulus + 10 more material properties
area + 11 more cross-sectional properties
joint# x-coord y-coord z-coord
element# node-1 node-2 node-3 node-4
loaded-dof force/moment intensity

7.4 MAIN PROGRAM

The main program file is named "sparse_nonlinear_modal.F." By calling different subroutines explained in Appendix E, the main program performs nonlinear modal finite element analysis. Parallel computation is used inside this program to distribute the load generation task among different processors. Also, modal displacements are converted into displacements in structural dof. Finally, Root Means Square (RMS) of maximum deflection is calculated. Modal participation factors are also calculated to predict contribution of different modes. The wall clock time for different segments of the whole analysis procedure is noted. For MPI procedures timing is measured using function mpi_wtime( ) whereas for sequencial procedure, function system_clock( ) is used for the same.

7.5 JOB SUBMISSION

The developed MPI FORTRAN source code for nonlinear finite element analysis is compiled and run in the UNIX environment. A script file is used to submit the job. A similar script file can be used for both synchronized as well as un-synchronized load case. But, parallel environment is not recommended in the case of synchronized load case. So, the information regarding number of processors/nodes should be different while dealing with such cases. Appendix E shows the script file used to submit the job containing MPI Fortran and MATLAB functions usage when 3 MPI processes are used.

Remarks:

1. The reported timing, shown in Tables 7.5, 7.7, 7.9, 7.10, 7.11, and 7.12 is based on one sample run.
2. All wall clock time reported in Table 7.9 has already been converted from ODU Sun-Fire-280R computer (slow) into Intel Pentium 4 (fast) computer platform.
3. For medium scale synchronized loading problem, the single precision Abaqus/Explicit wall clock time was estimated by running the simulation for (several) much shorter time durations than 2 seconds. The wall clock time was found to be almost proportional to the time duration. Hence, the timing values were extrapolated for the full 2.0 second duration. The time step was automatically computed by Abaqus and was estimated to be 9.4 $\times 10^{-8}$ seconds.

CHAPTER VIII
CONCLUSION

The study has been undertaken with an objective to solve large scale problems with synchronized or unsynchronized uniform random loads for the response of a plate. In this research work, computational issues in conjunction with the nonlinear modal methods have been discussed and implemented. The motive of this study started with the observation that because of the computational limitations many of the large scale non linear analysis problems for random responses take an enormous amount of time for the solution or remain unsolved. Also, real-life random loads are not deterministic, leading to studying simulation of unsynchronized load and solving the same for large scale problems.

Accuracy of the proposed sparse nonlinear modal algorithms has been validated through Table 7.4. Computational efficiency has been established through Table 7.9 and Table 7.14. The numerical results have indicated that the proposed sparse nonlinear method is accurate and highly efficient. It was reported by Green and Killey[12] that only running a half-second time for nonlinear time domain Monte Carlo simulation of 5000-element for a single-bay panel took approximately 10 hours on a Cray C94 computer. The developed FEA code consumes only ~176 mins (~ 3 hours) for running two second time history for nonlinear time domain Monte Carlo simulation of 65,536 elements. The comparison between the two programs itself proves the efficiency of the code. In fact, it can be noticed that for synchronized load cases, even though using double precision and without using the sparse re-ordering algorithms[22,23] in proposed sparse method, it is about 72 (~70967 mins/993mins) times faster than Abaqus, which uses single precision. For a synchronized load case problem, the maximum size problem solved is for about 400,000 dof. With capability of more RAM memory, the code is capable of solving even larger scale problems.

For the frst time, large scale un-synchronized load cases are solved using the proposed code. The proposed code provides ~ 90 % efficiency to solve a large scale problem using 31 processors. The same problem is unsolvable when fewer than 12 processors are used, which proves that parallel computation not only benefits by time savings but also helps solving memory related problems. The code could solve many unsolvable problems because of time and memory limitations. The code has the capability to link MATLAB and FORTRAN environment, and data exchange between the two.

Abaqus provides the nonlinear random response (NRR) in the structural dof (but not in the modal dof). However, these existing capabilities in current versions of commercialized FEA codes are highly inefficient. In Ref[21,] the 3x3 bay panel with approximately 96,000 dof took 24 hours for 0.1 sec nonlinear random response (NRR of RMS Max. deflection and RMS Max. Stress) using Abaqus/Explicit on the NASA Langley fast computer system (Itanium2). Since we need 2 sec NRR, it will take 20x24 hours= 20 days or less for each sample. For Monte Carlo simulation, we need 10 or more samples, that is almost 200 days for each sound pressure level. This can not be used as design tools. The reasons are: a) the nonlinear stiffness matrices $K_1(w)$ and $K_2(w^2)$ in structural dof are functions of the panel deflection w. They need to be updated at each time integration Δt (or every 5x Δt, 10x Δt, etc. this leads to poor accuracy), and (b) very small Δt compared to Δt in the modal dof.

Mex script is developed to submit the job for compilation and linking several FORTRAN source files into a shared library called a binary MEX-file from MATLAB software. At ODU'S Wright Brothers HPC Environment, only "small" environment is set up for Mex jobs. It has 8 slots and each slot can run a maximum of 4 processes. Especially for the problems which are both CPU and memory intensive, the best result is achieved through running 1 process per slot, which is a total of 8 parallel processes. For the un-synchronized load case, using 31 processors 36.43% efficiency is achieved for smallest scale problem which is increased up to 89.77% for large scale problem. The reason behind limiting efficiency up to ~90% is the usage of MATLAB function. Time consumed by each processor to call MATLAB and pass data from FORTRAN to MATLAB and vice versa remains the same, affecting the efficiency of the program. For un-synchronized load case, solution of a large scale problem using fewer than 12 processors was impossible due to memory limitations of the processors. When fewer than 12 processors are used, generation, re-scaling and storage of random load exceed the memory allotted to each processor.

For the problems with a large number of dof, the nonlinear finite element analysis becomes computationally challenging because large size matrices are involved. The best option to overcome such difficulty is usage of modal formulation. The method not only reduces the size of matrices drastically but also makes the nonlinear matrices constant and uses larger step size resulting in large time savings. In modal formulation, eigen solver needs to be employed initially, which restricts size of stiffness and mass matrices due to memory limitations. Moreover, conversion of linear and nonlinear matrices from structural dof to modal dof and vice versa makes the solution difficult while dealing with large dof. Application of sparse technology makes it possible to solve such problems, which are complex in terms of time and memory. For large scale problems considering unsynchronized loading, the solution becomes extremely difficult to resolve computationally and leads to application of parallel computation. It is extremely important to develop a software code to work as a design tool that combines nonlinear finite element analysis, modal formulation, sparse technology and parallel computation along with rescaling of the random load vector, with the capability of solving large scale problems. The research work presented in this dissertation fulfils the requirement and provides a versatile design tool.

8.1 FUTURE SCOPE

Unrolling techniques[23] and algorithms for sparse minimizing fill-in terms that occurred during the numerical factorization[22] have not yet been incorporated into the current version of the code. The sparse re-ordering technique is not implemented, either. Both of these techniques will further reduce the computational time of the proposed nonlinear sparse modal method.

Analysis of multi-bay aerospace structures can be extended from this work, by simply adding additional subroutines to transform an arbitrarily oriented rectangular plate element into the global co-ordinate references. The modeling of a 3x3-bay panel[21] includes a refined mesh for all individual separate panels and stiffeners. The stiffeners with various sections such as Z-, L- or hat sections will also be modeled with plate elements. Multi-bay structures are much more complicated than the single-bay panels, which could result in global (stiffeners) and local (separate panels) vibration modes. Therefore, a new challenge is to study the effects of two types of modes and to select the proper modes to be retained in the computational procedures.

The developed research code can be further extended for analysis of composite structures. Moreover, the software can be made generalized to perform even stress analysis for different kind of structures using curved shell elements and other types of elements. Also including the feature to facilitate aerodynamics loads (supersonic and hypersonic) and its coupling with thermal loads will help the design and behavior understanding of future high-speed flight vehicles.

APPENDIX A

TRANSFORMATION MATRICES $[T_b]$ AND $[T_m]$

The displacement vector of the BFS element can be written as:

$$\{w\} = \{\{w_b\} \quad \{w_m\}\}^{\mathrm{T}} \tag{A.1}$$

where transverse displacement vector is:

$$\{w_b\} = \{w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_{,x1} \quad w_{,x2} \quad w_{,x3} \quad w_{,x4} \quad w_{,y1} \quad w_{,y2} \quad w_{,y3} \quad w_{,y4} \quad w_{,xy1} \quad w_{,xy2} \quad w_{,xy3} \quad w_{,xy4}\}^{\mathrm{T}} \tag{A.2}$$

And membrane displacement vector is:

$$\{w_m\} = \{u_1 \quad u_2 \quad u_3 \quad u_4 \quad v_1 \quad v_2 \quad v_3 \quad v_4\} \tag{A.3}$$

The element transverse displacement function w and the in-plane displacement functions u and v are approximated as a bi-cubic and a bi-linear polynomial functions in x and y, which can be written as:

$$w(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + a_8x^2y + a_9xy^2$$
$$+ a_{10}y^3 + a_{11}x^3y + a_{12}x^2y^2 + a_{13}xy^3 + a_{14}x^3y^2 + a_{15}x^2y^3 + a_{16}x^3y^3$$

$$= [H_w(x,y)]_{1X16} \; \{a\}_{16X1} \tag{A.4}$$

$$u(x,y) = b_1 + b_2x + b_3y + b_4xy$$

$$= [H_u(x,y)]_{1x8} \; \{b\}_{8x1} \tag{A.5}$$

$$v(x,y) = b_5 + b_6x + b_7y + b_8xy$$

$$=\left[H_v(x,y)\right]_{1x8}\ \{b\}_{8x1} \tag{A.6}$$

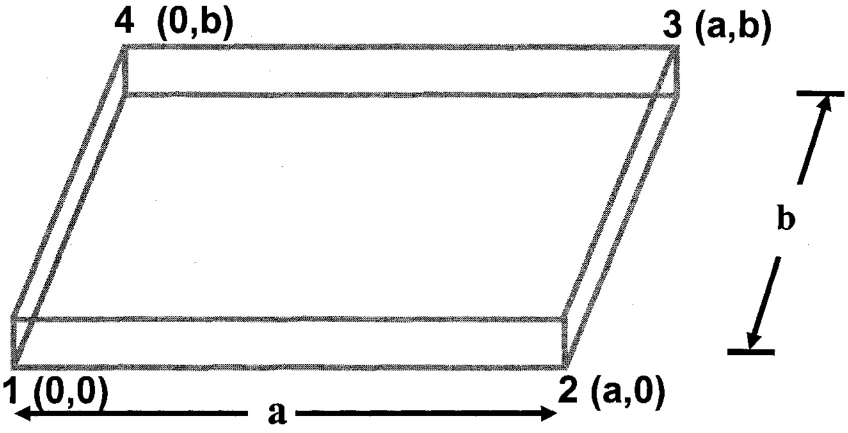
For the BFS C^1 –conforming element, the membrane displacement vector can be expressed as:

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} 1 & x & y & xy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & xy \end{bmatrix} \begin{Bmatrix} b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_8 \end{Bmatrix} \tag{A.7}$$

And the transverse displacement vector can be expressed as:

$$\begin{Bmatrix} w \\ w,_x \\ w,_y \\ w,_{xy} \end{Bmatrix} = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 & x^3y & x^2y^2 & xy^3 & x^3y^2 & x^2y^3 & x^3y^3 \\ 0 & 1 & 0 & 2x & y & 0 & 3x^2 & 2xy & y^2 & 0 & 3x^2y & 2xy^2 & y^3 & 3x^2y & 2xy^3 & 3x^2y^3 \\ 0 & 0 & 1 & 0 & x & 2y & 0 & x^2 & 2xy & 3y^2 & x^3 & 2x^2y & 3xy^2 & 2x^3y & 3x^2y^2 & 3x^3y^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2x & 2y & 0 & 3x^2 & 4xy & 3y^2 & 6x^2y & 6xy^2 & 9x^2y^2 \end{bmatrix} \begin{Bmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{15} \\ a_{16} \end{Bmatrix} \tag{A.8}$$

The nodal coordinates of the BFS plate elements are as shown in the fig:



Substituting the nodal coordinates into Eq. (A.7), the nodal membrane displacement $\{w_m\}$ can be written as:

$$\{w_m\}_{8x1} = [T_m]^{-1}_{8x8}\ \{b\}_{8x1} \qquad \text{(A.9)}$$

$$\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & a & b & ab & 0 & 0 & 0 & 0 \\ 1 & 0 & b & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & a & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & a & b & ab \\ 0 & 0 & 0 & 0 & 1 & 0 & b & 0 \end{bmatrix} \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{Bmatrix} \qquad \text{A.10)}$$

The in-plane transformation matrix $[T_m]$ is obtained by inverting the above matrix $[T_m]^{-1}$. Similarly, by substituting nodal coordinates into Eq. (A.8) , the nodal bending displacement $\{w_b\}$ can be written as:

$$\{w_b\}_{16x1} = [T_b]^{-1}_{16x16}\ \{a\}_{16x1} \qquad \text{(A.11)}$$

$$\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w,_{x1} \\ w,_{x2} \\ w,_{x3} \\ w,_{x4} \\ w,_{y1} \\ w,_{y2} \\ w,_{y3} \\ w,_{y4} \\ w,_{xy1} \\ w,_{xy2} \\ w,_{xy3} \\ w,_{xy4} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & a & 0 & a^2 & 0 & 0 & a^3 & 0 \\ 1 & a & b & a^2 & ab & b^2 & a^3 & a^2b \\ 1 & 0 & b & 0 & 0 & b^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2a & 0 & 0 & 3a^2 & 0 \\ 0 & 1 & 0 & 2a & b & 0 & 3a^2 & 2ab \\ 0 & 1 & 0 & 0 & b & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & a & 0 & 0 & a^2 \\ 0 & 0 & 1 & 0 & a & 2b & 0 & a^2 \\ 0 & 0 & 1 & 0 & 0 & 2b & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2a \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2a \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ab^2 & b^3 & a^3b & a^2b^2 & ab^3 & a^3b^2 & a^2b^3 & a^3b^3 \\ 0 & b^3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b^2 & 0 & 3a^2b & 2ab^2 & b^3 & 3a^2b^2 & 2ab^3 & 3a^2b^3 \\ b^2 & 0 & 0 & 0 & b^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a^3 & 0 & 0 & 0 & 0 & 0 \\ 2ab & 3b^2 & a^3 & 2a^2b & 3ab^2 & 2a^3b & 3a^2b^2 & 3a^3b^2 \\ 0 & 3b^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3a^2 & 0 & 0 & 0 & 0 & 0 \\ 2b & 0 & 3a^2 & 4ab & 3b^2 & 6a^2b & 6ab^2 & 9a^2b^2 \\ 2b & 0 & 0 & 0 & 3b^2 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{15} \\ a_{16} \end{Bmatrix} \tag{A.12}$$

The in-plane transformation matrix $[T_b]$ is obtained by inverting the above matrix $[T_b]^{-1}$.

APPENDIX B

FORTRAN CODE FOR GAUSSIAN-STATIONARY RANDOM LOAD GENERATION

```
c=====================================================================
c                              SIMLOAD
c=====================================================================
c    N           No. of intervals in the spectrum
c                N should be an integer power of two
c    NPT         No. of points for the time series
c                NPT should be an integer power of two, NPT > N
c    ISEED       Random number seed
c    TTOTAL = N/FMAX = Total Integration Time
c    DT        = N/(NPT*FMAX) = Integration Time Step Size
c=====================================================================
c   INSTRUCTIONS FOR SELECTING INPUT DATA
c=====================================================================
c  1. Take Highest Frequency, FMAX
c  2. Minimum Time Step is STEP_MIN=1/(2.5*FMAX)
c  3. N=FMAX*2
c  4. Pick Up Total Running Time (1sec, 2 sec, ...) T_total=N/FMAX
c  5. Select NPT to satisfy 2**(integer number)
c  6. Select NPT such that STEP=N/(NPT*FMAX)
c=====================================================================
      subroutine simload(spl, NPT, Fmax, x, y, dt)
      implicit real*8(a-h,o-z)
      parameter (N=Fmax*2)
      real*8 y(NPT),sp(N+1),w(N+1),rand(N),spl
      complex x(NPT),zimag
c=====================================================================
c  initial variables
```

```
c==================================================================
      spp=8.4144*10**(-18.+spl/10.) ! for distributed acoustic pressure
      pi=3.141592654
      pi2=pi*2.0
      np1=n+1
      zimag=cmplx(0.0,1.0)
      sppw=spp/pi2
      wu=fmax*pi2
      dw=wu/float(n)
      do 119 i=1,np1
            sp(i)=sppw
            w(i)=(i-1)*dw
 119  continue
      area=spp*fmax
      sq2dw=dsqrt(2.0*dw)
      ttotal=pi2/dw
      dt=ttotal/float(npt)

c==================================================================
c     Set x(1)=0. in order to obtain new mean zero time series
c==================================================================

      x(1)=cmplx(0.0,0.0)
      do 50 i=n+1,npt
            x(i)=cmplx(0.0,0.0)
 50         Continue

c==================================================================
c     Generate random phase angles uniformly distributed between
c     zero and 2.0*pi
c==================================================================
      iseed=12357
          call random_seed()
          call random_number(rand)
          do 60 i=2,n+1
                 phi=rand(i-1)*pi2
                 p1=sq2dw*dsqrt(sp(i))
                 x(i)=p1*cdexp(-zimag*phi)
 60            continue

c==========================================================
c...... perform forward transform
c==========================================================
      call fft(x,npt,1)
c==========================================================
c      get real part
c==========================================================
      do 70 i=1,npt
                 y(i)=real(x(i))                    ! for pressure loads
 70    continue
      return
      end
c==========================================================
c                         FFT
c==========================================================
      subroutine fft(x,n,k)
      implicit integer (a-z)
      real*4 gain,pi2,ang,re,im
      complex x(n),xtemp,t,u(16),v,w
      logical new
      data pi2,gain,n0,k0/6.283185307,1.0,0,0/
      new=n0.ne.n
      if(.not.new) go to 2
      l2n=0
      n0=1
1      l2n=l2n+1
      n0=n0+n0
      if(n0.lt.n) go to 1
      gain=1.0/n
      ang=pi2*gain
      re=cos(ang)
      im=sin(ang)
2      if(.not.new .and. k*k0.ge.1) go to 4
      u(1)=cmplx(re,-sign(im,float(k)))
      do 3 i=2,l2n
3                   u(i)=u(i-1)*u(i-1)
                    k0=k
4                   sby2=n
                    do 7 stage=1,l2n
                              v=u(stage)
                              w=(1.0,0.0)
                              s=sby2
                              sby2=s/2
                              do 6 L=1,sby2
                              do 5 i=1,n,s
                                        p=i+L-1
                                        q=p+sby2
                                        t=x(p)+x(q)
                                        x(q)=(x(p)-x(q))*w
5                                       x(p)=t
6                                       w=w*v
7         continue
          do 9 i=1,n
```

```
                    index=i-1
                    jndex=0
                    do 8 j=1,L2n
                     jndex=jndex+jndex
                     itemp=index/2
                     if (itemp+itemp.ne.index) jndex=jndex+1
                     index=itemp
8                   continue
                    j=jndex+1
                    if(j.LT.i) go to 9
                    xtemp=x(j)
                    x(j)=x(i)
                    x(i)=xtemp
9       continue
        if (k .gt. 0) return
        do 10 i=1,n
10           x(i)=x(i)*gain
        return
        end
```

APPENDIX C
LINEAR RANDOM VIBRATION

From PDE for an isotropic rectangular plate,

$$\rho h \frac{\partial^2 w}{\partial t^2} + D\nabla^4 w = p_0(t) \tag{C.1}$$

For a simply supported boundary condition, the plate deflection can be expressed as:

$$w(x,y,t) = \sum_m \sum_n q_{mn}(t)\phi_{mn}(x,y) \tag{C.2}$$

And for a simply supported boundary condition, the mode shapes are:

$$\phi_{mn}(x,y) = \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right) \tag{C.3}$$

Substituting Eq. (C.3) into Eq. (C.1) and applying the modal orthogonality condition, The modal equation becomes:

$$\ddot{q}_{mn} + \omega_{mn}^2 q_{mn} = \frac{p_0(t)}{m_{mn}}, \qquad where\ m,n = 1,3,5,..... \tag{C.4}$$

Adding a structural damping,

$$\ddot{q}_{mn} + 2\xi_{mn}\omega_{mn}\dot{q}_{mn} + \omega_{mn}^2 q_{mn} = \frac{p_0(t)}{m_{mn}} \tag{C.5}$$

$$\omega_{mn} = \pi^2 \sqrt{\frac{D}{\rho h}}\left[\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2\right] \quad rad/\sec \tag{C.6}$$

$$m_{mn} = \frac{mn\pi^2\rho h}{16} \tag{C.7}$$

where ω_{mn} and m_{mn} are the natural frequency and modal mass, respectively.

The response to Eq. (C.5) is given by Eq. (3-57) and (7-37) in reference[26]

$$E\left[q_{mn}^2\right] = \frac{\pi\ S_0(f)}{8m_{mn}^2\xi_{mn}\omega_{mn}^3} \tag{C.8}$$

Set mn=r and kl=s,

$$E[q_{mn}q_{kl}] = E[q_r q_s] = \frac{(\xi_r\omega_r + \xi_s\omega_s)S_0(f)}{m_r m_s\left[\left(\omega_r^2 - \omega_s^2\right)^2 + 4\omega_r\omega_s(\xi_r\omega_r + \xi_s\omega_s)(\xi_r\omega_s + \xi_s\omega_r)\right]}$$

The root mean square of maximum deflection obtained from Eq. (C.2) can be expressed as:

$$RMS(w_{\max}) = \sqrt{\left\{E\left[\left(\sum_{r=1}^{n} q_r\right)^2\right]\right\}} \tag{C.9}$$

$$= \sqrt{\left(E\left[q_1^2\right] + E\left[q_2^2\right] + ... + 2E[q_1q_2] + ...\right)} \tag{C.10}$$

APPENDIX D

FORTRAN CODE FOR LOAD VECTOR RE-SCALING

```
c=================================================================
c                             RESCALE
c=================================================================
c                            INPUT DATA
c=================================================================
c       spl          = Sound Pressure Level
c       y            = Random Load vector generated using SIMLOAD
c       NPT          = No. of points for the time series used in SIMLOAD
c       Fmax         = Frequency
c=================================================================
c                            OUTPUT DATA
c=================================================================
        temp_rdn     = Rescaled random load vector
c=================================================================
        subroutine rescale(spl, NPT, Fmax, y, temp_rdn)
        implicit real*8(a-h, o-z)
#include "fintrf.h"
        mwpointer engOpen, engGetVariable
        mwpointer mxCreateDoubleMatrix, mxGetPr
        mwpointer ep, x_m, w_m, p, x_m1, w_m1, p_m
        integer engPutVariable, engEvalString, engClose

        real*8 y(NPT)
        real*8 dt,spp
        real*8  freq(NPT), psd(NPT),temp_rdn(NPT)
        integer status
```

```
c=================================================================
c Initializing vectors
c=================================================================
      Wpp=spp
      do 5 i=1,NPT
      freq(i)=0.0
      psd(i)=0.0
5     continue
      ep = engOpen('matlab ')
      if (ep .eq. 0) then
      write(6,*) 'ERROR: MATLAB engine did not start'
      stop
      endif
c=================================================================
c     Put variable 'x' into MATLAB workspace
c     Create a vector of size NPT X 1 & initialize to zero:
c=================================================================
      x_m = mxCreateDoubleMatrix(NPT, 1, 0)
c=================================================================
c     copy  Fortran array y into MATLAB array called mxGetPr
c     x_m is no of elements to copy y(x_m)
c     x_m=NPT
c     copy array y of Fortran to MATLAB array using pointer mxGetPr
c=================================================================
      call mxCopyReal8ToPtr(y, mxGetPr(x_m), NPT)
c=================================================================
c   Put variable x_m into MATLAB
c=================================================================
      status = engPutVariable(ep, 'x_m', x_m)
      if (status .ne. 0) then
      write(6,*) 'ERROR: engPutVariable'
      stop
      endif
c=====================================================================
c  Evaluate PSD using pwelch
c=====================================================================
      if(engEvalString(ep,'[Pxx,w_m]=pwelch(x_m,[],[],16384,8192);')
      ne. 0)then
      write(6,*) 'ERROR: engEvalString'
      stop
      endif
c=====================================================================
c   Get the variable  w_m from MATLAB
c=====================================================================
      w_m = engGetVariable(ep, 'w_m')
c=====================================================================
c     Copy from MATLAB ARRAY to FORTRAN ARRAY:
c=====================================================================
      call mxCopyPtrToReal8(mxGetPr(w_m), freq, 1000)
c=====================================================================
c     Get the variable  Pxx from MATLAB which is PSD
c=====================================================================
      p = engGetVariable(ep, 'Pxx')
      call mxCopyPtrToReal8(mxGetPr(p), psd, 1000)
c=====================================================================
c     Calculate average of first 1000 PSD values which should be equal to
c     input spp value
c=====================================================================
      Wpp_sim1=0.0
      do 101 j=1,1000
            Wpp_sim1=Wpp_sim1 + psd(j)
 101        continue
```

```
        avg1=Wpp_sim1/1000
c======================================================================
c     Check the difference by calculating ratio between calculated spp and initial spp
c======================================================================
        ratio = avg1/Wpp
c======================================================================
c     Scale the load vector {y} using difference ratio
c======================================================================
        do k=1,NPT
                temp_rdn(k) =y(k)/sqrt(ratio)
                enddo
c======================================================================
c     Now using the same procedure check the spp values for re-scaled load vector
c======================================================================
        x_m1 = mxCreateDoubleMatrix(NPT, 1, 0)
        call mxCopyReal8ToPtr(temp_rdn, mxGetPr(x_m1),NPT )
        status = engPutVariable(ep, 'x_m1',x_m1)
        if (status .ne. 0) then
                write(6,*) 'ERROR: engPutVariable'
                stop
        endif

        if(engEvalString(ep,'[Pxx,w_m]=pwelch(x_m1,[],[],16384,8192);')
        .ne. 0)then
                write(6,*) 'ERROR: engEvalString'
        stop
        endif
        w_m = engGetVariable(ep,'w_m')
        call mxCopyPtrToReal8(mxGetPr(w_m),freq, 1000)
        p_m = engGetVariable(ep, 'Pxx')
        call mxCopyPtrToReal8(mxGetPr(p_m), psd, 1000)
```

```
      Wpp_sim2=0.0
      do 102 j=1,1000
              Wpp_sim2=Wpp_sim2 + psd(j)
102               continue
      avg2=Wpp_sim2/1000
c=====================================================================
c     Calculate % of error between calculated new spp & initial spp
c=====================================================================
       ERROR= (avg2-Wpp)*100/Wpp
c=====================================================================
c     Delete all the arrays created in MATLAB
c=====================================================================
      call mxDestroyArray(p)
      call mxDestroyArray(w_m)
      call mxDestroyArray(x_m)
c=====================================================================
c     Close MATLAB environment
c=====================================================================
      status = engClose(ep)
      if (status .ne. 0) then
      write(6,*) 'ERROR: engClose'
      stop
      endif
      stop
      end
```

APPENDIX E

DESCRIPTION OF SUBROUTINES

1. Subroutine generalip

Purpose: To read input data for general information regarding the structure

Output:

nel	= Number of elements; scalar
ndofpe	= Number of dof per element; scalar
neltype	= Element type; scalar
maxdofpe	= Maximum dof per element; scalar
nboundc	= Number of active boundary conditions; scalar
ndofpn	= Number of dof per node; scalar
numnodes	= Number of nodes; scalar
loadof	= Number of nodes at which external load is applied; scalar
nummat	= Number of materials; scalar
nlayers	= Number of layers; scalar
nsect	= Number of sections; scalar
lumpmass	= Indicator for lump or diagonal mass matrix; scalar
ndof	= Total number of dof; scalar
ieall	= Total number of different type of elements; scalar
jeall	= Total dof of all different type of elements; scalar
neig	= Number of eigen-solutions; scalar
nreord	= Indicator for reordering technique; scalar
Iflag	= Flag array, vector of 10 x 1
nunroll	= Level of unroll; scalar
nmode	= Number of calculated modes; scalar
modepick	= Number of selected modes; scalar

2. Subroutine materprop

Purpose: To input material properties, assuming maximum 10 materials and each material type has 11 properties

Input:

nummat = Number of materials; scalar

Output:

propmat = Properties of the material; vector of size nummat *11 x 1

3. Subroutine sectprop

Purpose: To input information regarding material cross-sectional properties, assuming maximum 10 cross sections and each cross sectional type has 12 properties

Input:

nsect = Number of sections; scalar

Output:

propsect = Cross-sectional properties of the material; vector of size nsect*12 x 1

4. Subroutine nodecoor

Purpose: To provide co-ordinates at each node

Input:

numnodes = Number of nodes; scalar

Output:

x,y,z = Nodal co-ordinates; vectors of size numnodes x 1

5. Subroutine elconect

Purpose: To provide information regarding element connectivity

Input:

neltype = Element type; scalar

nel = Number of elements; scalar

ndofpe = Number of dof per element; scalar

ndofpn = Number of dof per node; scalar

Output:

ie = Locations of the first non-zero term of each row,vector of size ieall +1 x 1

je = Global node numbers associated with each element,vector of size jeall x 1

lm = Indicator to coordinate local dof to global dof,vector of size maxdofpe x 1

lm_store = Stores information to coordinate local dof to global dof for all elements, matrix of size maxdofpe x nel

6. Subroutine loads

Purpose: To input applied loads at the joints

Input:

ndof = Number of dof; scalar

loadof = Number of nodes at which external load is applied; scalar

Output:

b = Load vector; vector of size ndof x 1

7. Subroutine supportdof

Purpose: To input support dof information

Input:

ndof = Number of dof; scalar

b = Load vector; vector of size ndof x 1

nboundc = Number of active boundary conditions; scalar

Output:

iboundc = Indicator to define location where boundary condition is defined; vector of size ndof x 1

ia = boundary flag array to use in sparse symbolic assembly; vector of size ndof + 1 x 1

8. Subroutine transa2d

Purpose: To transpose a sparse matrix

Input:

ndof = Number of dof; scalar

ieall = Total number of different type of elements; scalar

ie = Locations of the first non-zero term of each row, vector of size ieall +1 x 1

je = Global node numbers associated with each element, vector of size jeall x 1

Output:

iet = Transpose of vector ie; vector of size ndof + 1 x 1

jet = Transpose of vector je; vector of size jeall x 1

9. Subroutine symbass

Purpose: To perform symbolic assembly of a sparse matrix

Input:

ndof = Number of dof; scalar

ie = Locations of the first non-zero term of each row, vector of size ieall +1 x 1

je = Global node numbers associated with each element, vector of size jeall x 1

iet = Transpose of vector ie; vector of size ndof + 1 x 1

jet = Transpose of vector je; vector of size jeall x 1

Output:

ia = starting locations of the first non-zero off-diagonal terms for each row of structural stiffness matrix; vector of size ndof+1 x 1

ja = column numbers (unordered) corespond to each nonzero, off-diagonal term of each row of structural stiffness matrix; vector of size ncoefl x 1

ncoefl = Number to define size of vector ja; scalar

10. Subroutine assembly

Purpose: To assemble element stiffness matrices, element mass matrices and element load vectors (by calling subroutines 11 to 15)

11. Subroutine info_bfs

Purpose: To provide information regarding each element

Input:

iel = Element number, scalar

lm = Indicator to coordinate local dof to global dof; vector of size maxdofpe x 1

propmat = Properties of the material; vector of size nummat *11 x 1

x,y,z = Nodal co-ordinates; vectors of size numnodes x 1

Output:

x_bfs,y_bfs = Co-ordinates of the 4 nodes associated with particular element; vectors of size numnodes x 1

e11 = Young's modulus of the element material; scalar

xnu12 = Posision's ratio for the element material; scalar

rho = Density of the element material; scalar

t = Element thickness; scalar

A, D = Matrices to define material properties for composite as well as isotropic material; matrices of size 3 x 3

12. Subroutine bfs_ls

Purpose: To evaluate element stiffness matrix

Input:

maxdofpe = Maximum dof per element; scalar

x,y,z = Nodal co-ordinates; vectors of size numnodes x 1

A, D = Matrices to define material properties for composite as well as isotropic material; matrices of size 3 x 3

Output:

Ke = Element stiffness matrix; matrix of size maxdofpe x maxdofpe

13. Subroutine bfs_lm

Purpose: To evaluate element mass matrix

Input:

maxdofpe = Maximum dof per element; scalar

x,y,z = Nodal co-ordinates; vectors of size numnodes x 1

A, D = Matrices to define material properties for composite as well as isotropic material; matrices of size 3 x 3

rho = Density of the element material; scalar

t = Element thickness; scalar

Output:

Me =Element mass matrix; matrix of size maxdofpe x maxdofpe

14. Subroutine bfs_be

Purpose: To evaluate element load vector

Input:

maxdofpe = Maximum dof per element; scalar

x,y,z = Nodal co-ordinates; vectors of size numnodes x 1

A, D = Matrices to define material properties for composite as well as isotropic material; matrices of size 3 x 3

Output:

be = Element load vector; vector of size maxdofpe x 1

15. Subroutine numass

Purpose: To perform sparse numerical assembly

Input:

ia = starting locations of the first non-zero off-diagonal terms for each row of structural stiffness matrix; vector of size ndof+1 x 1

ja = Unordered column numbers correspond to each nonzero, off-diagonal term of each row of structural stiffness matrix; vector of size ncoefl x 1

idir = Flag which stores 1 in the positions correspond to Dirichlet boundary conditions, 0 elsewhere; vector of size ndof x 1

ae = 1-D array which stores element stiffness matrix; vector of size (ndofpe^2) x 1

be = Element load vector; vector of size maxdofpe x 1

lm = Indicator to coordinate local dof to global dof; vector of size maxdofpe x 1

maxdofpe = Maximum dof per element; scalar

b = Load vector; vector of size ndof x 1

Before using this sub-routine, values of $\{b\}$ should be initialized to values of prescribed Dirichlet bc at proper locations or values of applied nodal loads

Output:

an = Numerical values of nonzero, off-diagonal terms of structural stiffness matrix; vector of size ncoefl x 1

ad = Numerical values of diagonal terms of structural stiffness matrix; vector of size ndof x 1

b = Assembled load vector; vector of size ndof x 1

16. Subroutine split_sparse_bbmm_improved

Purpose: To split sparse assembled matrix into 2 sets of sparse matrices, one set is related to bending dof and other set is related to membrane dof only. This can be done when bending and membrane portions are completely uncoupled.

Input:

ndof = Total number of dof; scalar

an = Numerical values of nonzero, off-diagonal terms of structural stiffness matrix; vector of size ncoefl x 1

ad = Numerical values of diagonal terms of structural stiffness matrix; vector of size ndof x 1

ia = Starting locations of the first non-zero off-diagonal terms for each row of structural stiffness matrix; vector of size ndof+1 x 1

ja = Unordered column numbers correspond to each nonzero, off-diagonal term of each row of structural stiffness matrix; vector of size ncoefl x 1

Output:

iabb, jabb, adbb, anbb = Bending dof portion of the vectors ia, ja, ad and an, respectively

iamm, jamm, admm, anmm = Membrane dof portion of the vectors ia, ja, ad and an, respectively

ncoefl_bb, ncoefl_mm = Number which indicates the value of ncoefl corresponding to bending and membrane dof, repectively

17. Subroutine eigsolver011

Purpose: To perform eigen solution using Subspace eigen solver for real symmetric stiffness and mass matrices when lumped mass matrix is considered.

Input:

ndof_b = Total number of bending dof; scalar

ncoefl_bb = Number which indicates the value of ncoefl corresponding to bending dof only; scalar

neig = Number of eigen-solutions; scalar

lumpmass = Indicator for lump or diagonal mass matrix; scalar

mtot = Estimated total static memory; scalar

iabb, jabb, adbb, anbb = Bending dof portion of the vectors ia, ja, ad and an, respectively

dmbb = Array to store diagonal-mass matrix; vector of size ndof_b x 1

Output:

evalues = Computed eigenvalues; vector of size nc x 1 where nc is the number of eigen values required to compute

evectors = Computed eigenvectors; matrix of size ndof_b x nc

iperm_b = Permutation vector from the reordering; vector of size ndof_b x 1

18. Subroutine symfactd

Purpose: To perform symbolic factorization

Input:

n = Order of given matrix A; scalar

ia = Starting locations of the first non-zero off-diagonal terms for each row of given matrix; vector of size n+1 x 1

ja = Unordered column numbers correspond to each nonzero, off-diagonal term of each row of given matrix; vector of size ncoef1 x 1

Output:

iu,ju = Structure of resulting matrix U; vectors of size n + 1 x 1 and ncoef2 x 1, respectively

ncoef2 = Number to define the size of the vector ju; scalar

ip = Chained lists of rows associated with each column. Also used as the multiple switch array; vector of size n x 1

19. copy_int

Purpose: To save a copy of input array

Input:

iarray = Array to be copied;vector of size n x 1

Output:

Icopy = copy of the input array;vector of size n x 1

20. subroutine mvsparse

Purpose: To perform multiplication of sparse matrix and vector

Input:

n = Number to define size of vectors ia and ad; scalar

ia, ja, an, ad = Vectors to store given sparse matrix; vectors of size n + 1 x 1, ncoef1 x 1, ncoef1 x 1, n x 1, respectively

b = Given vector of size n x 1

Output:

c = Array resulting after multiplication of given sparse matrix and vector; vector of size n x 1

21. Subroutine transad

Purpose: After symbolic factorization, ja is just a merge list without ordering (i.e. the nonzero column numbers of a particular row are 12, 27, 14, 46, 22, 133). Upon completion, this routine will rearrange the nonzero column numbers with ordering, to be ready for numerical factorization phase (i.e. 12, 14, 22, 27, 46, 133)

Input:

ia, ja, an = Arrays to store sparse matrix

n = Number of rows of the sparse matrix; scalar

m = Number of columns of the sparse matrix; scalar

Output:

iat, jat, ant = Arrays to store transpose of sparse matrix;

22. Subroutine EvaTb

Purpose: To evaluate matrix $[T_b]$ for BFS plate element

Input:

a = BFS plate element length; scalar

b = BFS plate element width; scalar

Output:

$[T_b]$ = Transformation matrix related to bending dof; matrix of size ndofb_pe x ndofb_pe. where, ndofb_pe = number of bending dof per element

23. Subroutine EvaTm

Purpose: To evaluate matrix $[T_m]$ for BFS plate element

Input:

a = BFS plate element length; scalar

b = BFS plate element width; scalar

Output:

$[T_m]$ = Transformation matrix related to membrane dof; matrix of size ndofm_pe x ndofm_pe. where, ndofm_pe = number of membrane dof per element

24. Subroutine Trans

Purpose: To evaluate transpose of given matrix

Input:

$[A]$ = Given matrix of which transpose needs to be evaluated; matrix of size m x n

Output:

$[B]$ = Transpose of matrix $[A]$; matrix of size n x m

25. Subroutine EvaCm

Purpose: To evaluate matrix $[Cm]$ for BFS plate element. Also provides transpose of matrix $[Cm]$ by using subroutine Trans.

Input:

x = x co-ordinates of the nodes; scalar

y = y co-ordinates of the nodes; scalar

Output:

$[Cm]$ = Matrix needs for the stiffness matrices evaluation; matrix of size 3 x 8

$[Cmt]$ = Transpose of matrix $[Cm]$; matrix of size 8 x 3

26. Subroutine EvaCzi

Purpose: To evaluate matrix $[Czi]$ for BFS plate element. Also provides transpose of matrix $[Czi]$ by using subroutine Trans.

Input:

x = x co-ordinates of the nodes; scalar

y = y co-ordinates of the nodes; scalar

Output:

$[Czi]$ = Matrix needs for the stiffness matrices evaluation; matrix of size 2 x 16

$[Czit]$ = Transpose of matrix $[Czi]$; matrix of size 16 x 2

27. Subroutine VECTM

Purpose: To evaluate multiplication of given matrix $[B]$and a vector $\{C\}$

Input:

$[B]$ = Given matrix of size m x n

$\{C\}$ = Given vector of size n x 1

Output:

{A} = Multiplication of given matrix$[B]$ and vector$\{C\}$; vector of size m x 1

28. Subroutine multiply

Purpose: To evaluate multiplication of a matrix $[A]$and a matrix $[B]$

Input:

$[A]$ = Given matrix of size m x l

$[B]$ = Given matrix of size l x n

Output:

$[C]$ = Multiplication of given matrices $[A]$ and $[B]$; matrix of size m x n

29. Subroutine numfa1d

Purpose: To perform numerical factorization

Input:

ia, ja, an, ad = Vectors to store given sparse matrix; vectors of size n + 1 x 1, ncoef1 x 1, ncoef1 x 1, n x 1, respectively

n = Number to define size of vectors ia and ad; scalar

iu, ju = Structure of resulting matrix, U after symbolic factorization; vectors of size n + 1 x 1 and ncoef2 x 1,respectively

Output:

un = Numerical values of the non-zeros of matrix U di = Inverse of the diagonal matrix D

Working space:

ip = Chained lists of rows associated with each column; vector of size n x 1

iup = Auxiliary pointers to portions of rows

di = Array used as the expanded accumulator

30. Subroutine fbed

Purpose: To perform sparse forward/backward solution phase

Input:

iu, ju, un = Vectors to store given upper triangular matrix with unit diagonal matrix.

di = Inverses of the diagonal elements of the diagonal matrix D.

b = Right-hand side vector b.

n = Order of the system, n>1.

Output:

x = Vector of unknowns x.

31. Subroutine BFS_k1_element

Purpose: To evaluate element matrix $[k1_{bm}]$.

Input:

xx, yy, zz = Nodal co-ordinates of the element; vectors of size numnodespe x1. where, numnodespe is number of nodes per element.

$[A],[D]$ = Membrane stiffness matrix; bending stiffness matrix; matrices of size 3 x 3

$\{wb1\}$ = Eigen-vector values for specific element; vector of size maxdofpe x 1

Output:

$[K1e]$ = Element matrix $[k1_{bm}]$

32. Subroutine assembly_bfs_k1

Purpose: To perform sparse numerical assembly of element matrices $[k1_{bm}]$. It provides the output in sparse matrix form.

Input:

Same as input data required for subroutines "info_bfs", "BFS_k1_element" and "numass."

Output:

ia, ja, an, ad = Vectors to store given system sparse matrix $[K1_{bm}]$; vectors of size n + 1 x 1, ncoefl x 1, ncoefl x 1, n x 1, respectively.

33. Subroutine BFS_K1_modal

Purpose: To evaluate first order nonlinear modal stiffness matrix $[K1_{bm}]$ by calling subroutines "BFS_k1_element" and "assembly_bfs_k1." Also, calculates $[K1\_\mathrm{mod}al]$ $=[K1_{bm}][K_m]^{-1}[K1_{mb}]$. Here, $[K_m]^{-1}$ is evaluated using LDL^T equation solving method.

Input:

$[evect]$ = Eigen vector matrix; matrix of size ndof_b x nmode

ndofpe = Number of dof per element; scalar

Note: All other required input datas are same as required by subroutines "BFS_k1_element" and "assembly_bfs_k1."

Output:

$[K1\_\mathrm{mod}al]$ = Matrix to define triple product $[K1_{bm}][K_m]^{-1}[K1_{mb}]$; matrix of size nmode x nmode x nmode x nmode

store_ia, store_ja, store_an, store_ad = Vectors to store ia,ja,an and ad vectors which defines system sparse matrix $[K1_{bm}]$

store_ncoefl = Number to define size of vectors store_ja and store_an; scalar

34. Subroutine BFS_k2_element

Purpose: To evaluate element matrix $[k2_b]$.

Input:

xx, yy, zz = Nodal co-ordinates of the element; vectors of size numnodespe x1. where, numnodespe is number of nodes per element.

$[A]$, $[D]$ = Membrane stiffness matrix; bending stiffness matrix; matrices of size 3 x 3 .

$\{wb1\}, \{wb2\}$ = Eigen-vector values for specific element; vector of size maxdofpe x 1

Output:

$[K2e]$ = Element matrix $[k2_b]$

35. Subroutine BFS_K2_modal

Purpose: To evaluate second order nonlinear system modal stiffness matrix $[K2_b]$ by calling subroutines "BFS_k2_element" and "info_bfs." Also, calculates $[K2\_modal] = [\Phi]^T [K2_b][\Phi]$.For which, Firstly triple procduct, $[\phi]^T [k2_b][\phi]$ at element level is evaluated and then numerical assembly is done.

Input:

$[evect]$ = Eigen vector matrix; matrix of size ndof_b x nmode

ndofpe = Number of dof per element; scalar

Note: All other required input datas are same as required by subroutines "BFS_k2_element","MULTIPLY" and "trans."

Output:

$[K2\_modal]$ = Matrix to define triple product $[\Phi]^T [K2_b][\Phi]$; matrix of size nmode x nmode x nmode x nmode

36. Subroutine BFS_k2nm_element

Purpose: To evaluate element matrix $[k2_{nm}]$.

Input:

xx,yy,zz = Nodal co-ordinates of the element; vectors of size numnodespe x1. where, numnodespe is number of nodes per element.

$[A]$, $[D]$ = Membrane stiffness matrix; bending stiffness matrix; matrices of size 3 x 3.

$\{wm2\}$ = Element level vector calculated based on eigen-vector values for specific element; vector of size ndof_m x 1.

Output:

$[K2nme]$ = Element matrix $[k2_{nm}]$; matrix of size maxdofpe x maxdofpe

37. Subroutine BFS_K2nm_modal

Purpose: To evaluate second order nonlinear modal stiffness matrix $[K2nm\_modal]$ by calling subroutines “mvsparse,” ”numfal d,” “fbed,” “info_bfs,” “BFS_k2nm_element.” Firstly, it calculates triple product, $[\phi]^T[k2_{nm}][\phi]$ at element level and then numerical assembly is done.

Input:

$[evect]$ = Eigen vector matrix; matrix of size ndof_b x nmode

store_ia, store_ja, store_an, store_ad = Vectors to store ia, ja, an and ad vectors which defines system sparse matrix $[K1_{bm}]$

store_ncoefl = Number to define size of vectors store_ja and store_an; scalar

Note: All other required input datas are same as required by subroutines “mvsparse”,”numfal d”, “fbed”, “info_bfs”, “BFS_k2nm_element”.

Output:

$[K2nm\_modal]$ = Matrix to define triple product $[\Phi^T][K2_{nm}][\Phi]$; matrix of size nmode x nmode x nmode x nmode

38. Subroutine simload

Purpose: To generate random load time history. It also performs rescaling of the generated random load vector by using MATLAB function pwelch. It uses subroutine FFT to perform the function.

Input:

spl = Sound Pressure Level; scalar

NPT = Number of time history points; scalar

N = Number of intervals in the spectrum; scalar

Fmax = Highest frequency; scalar

Output:

y = Random load vector; vector of size NPT x 1

dt = Integration time step size; scalar

39. Subroutine FFT

Purpose: To compute the power spectrum of the responses.

Input:

NPT = Number of time history points; scalar

k = Flag which is equal to 1 to perform forward transform and -1 for inverse transform; scalar

Output:

x = Complex number indicating spatial points; vector of size NPT x 1

40. Subroutine RK4

Purpose: To perform 4th order Runge-Kutta time integration to solve second order differential equation. It uses subroutine DERY to evaluate differentiation.

Input:

Step s= Integration time step size; scalar

work1, work2 = Temporary working arrays; vectors of size 2 * nmode + 1 x 1

Flag = Number to indicate whether the values are initial or not

bkesi = Modal damping coefficients; vector of size nmode x 1

Omega = Modal frequency

modalk = Modal linear stiffness matrix; matrix of size nmode x nmode

K1,K2 = Nonlinear modal first order stiffness matrix, nonlinear modal second order stiffness matrix; matrices of size nmode x nmode x nmode and nmode x nmode x nmode x nmode

P = Modal load vector

N = Number of modes used

MINV = Diagonal terms of inverse of normalized mass matrix

Modevalues

Output:

x = Modal displacement and modal velocity vector; vector of size 2*nmode + 1 x 1

dx = Modal velocity and modal acceleration; vector of size 2*nmode + 1 x 1

APPENDIX F

SCRIPT FILE FOR JOB SUBMISSION

```
#!/bin/tcsh
#$ -cwd
#$ -j y
#$ -S /bin/tcsh
#$ -pe large-impi 1
echo " "
echo "*** MPI    ***"
echo " "
set intel_mpi_env=/share/opt/intel/mpi/env
echo " "
source      ${intel_mpi_env}
echo " "
echo "*** MATLAB ***"
echo " "
set intel_mpi_opt=/share/opt/matlab/R2007b/opt/intel_mpi.sh
set mdir=/opt/matlab/R2007b/bin
echo " "
echo "*** PATH ***"
echo " "
setenv PATH ${mdir}:$PATH
setenv LD_LIBRARY_PATH ${mdir}/glnxa64:$LD_LIBRARY_PATH
echo " "
echo "*** MEX  ***"
echo " "

    mex -f ${intel_mpi_opt} -v  -fortran -I/opt/matlab/2007a/extern/include/
sparse_nonlinear_modal.F  sparse_subroutines.F no_source_code.o bfs_linear.o
bfs_nonlinear.o bfs_subroutines.o brick8_linear_k.o EigNormcheck1.o
EigSubspace022.o normcheckR.o reord002.o EigNormcheck2.o NewAM.o numfa1.o
reordAdj.o EigPrint001.o NewDiagM.o numfa2.o solver000.o EigPrint002.o ernorm.o
numfa8.o solver001.o EigPrint003.o fbe.o numfaR.o solver002.o EigPrint004.o gennd.o
pierrotime.o solver003.o EigPrint005.o jacobi2.o print001.o solver004.o EigRead001.o
matmat3.o print002.o solver1.o EigRead002.o print003.o supnode.o EigSolver001.o
metisreord.o print004.o symfact.o EigSolver002.o mmd.o print005.o transa.o
EigSolver011.o multspa.o print006.o transa2.o EigSolver022.o newAN.o read001.o
transaR.o EigSubspace001.o newDiagB.o read003.o EigSubspace002.o newIAJA.o
reord000.o EigSubspace011.o normcheck3.o reord001.o met01.o met02.o met03.o

echo "*** RUN  ***"
echo " "
```

```
@ NPROCS=$NSLOTS *  2 + 1
echo " "
    mpdtrace
echo " "
    mpiexec -n $NPROCS -env I_MPI_DEVICE ssm sparse_nonlinear_modal
echo " "
echo "*** END  ***"
echo " "
```

APPENDIX G
LISTINGS OF THE ENTIRE FORTRAN SOURCE CODE OF PROPOSED SPARSE-PARALLEL NONLINEAR FEA METHOD

Listing of the entire FORTRAN source code of proposed sparse-parallel nonlinear FEA method can be obtained by contacting either of the following persons:

1. Dr. Swati Chokshi, Email: swati.str.eng@gmail.com, Ph: 757 489 4422
2. Prof. Duc T. Nguyen, ODU, CEE Dept., 135 KAUF, Norfolk, VA 23529,Email: dnguyen@odu.edu, Ph: 757 683 3761

REFERENCES

1 Rudder, F.F., and Plumblee, H.E., "Sonic Fatigue Design Guide of Military Aircraft," *AFFDL-TR-74-112*, Wright-Patterson AFB, OH, May 1975, pp. 489.

2 Lassiter, L.W., Hess, R.W., and Hubbard, H.H., "An Experimental Study of the Response of Simple Panels to Intense Acoustic Loading," *Journal of Aeronautical Sciences*, Vol. 24, No. 1, 1957, pp. 19-24.

3 Clarkson, B.L., "Stresses in Skin Panels Subjected to Random Acoustic Loads," *Journal of Royal Aero Society,* Vol. 72, 1968, pp. 1000-1010.

4 Wilby, J.F., and Gloyna, F.L., "Vibration Measurement of an Airplane Fuselage Structure. Part I: Turbulent Boundary Excitation," *Journal of Sound and Vibration*, Vol. 23, 1972, pp. 205-210.

5 Holehouse, I., "Sonic Fatigue Design Techniques for Advanced Composite Aircraft Structures," *AFWAL TR 80-3019*, Wright-Patterson AFB, OH, April 1980, pp. 30-80.

6 Choi, S.T., and Vaicaitis, R., "Nonlinear Response and Fatigue of Stiffened Panels," *Probabilistic Engineering Mechanics*, Vol. 4, 1989, pp. 150-160.

7 Bolotin, V.V., "*Random Vibration of Elastic Systems*," Martinus Nijhoff Publishers, The Netherlands, 1984, pp. 290-292.

8 Mei, C., and Paul, D.B., "Nonlinear Multimode Response of Clamped Rectangular Plates to Acoustic Loading," *AIAA Journal*, Vol. 24, No. 4, 1986, pp. 643-648.

9 Mei, C., and Chen, R.R., "Finite Element Nonlinear Random Response of Composite Panels of Arbitrary Shape to Acoustic and Thermal Loads Applied Simultaneously," *WL-TR-97-3085*, Wright-Patterson AFB, OH, 1997.

10 Mei, C., and Wentz, K.R., "Analytical and Experimental Nonlinear Response of Rectangular Panels to Acoustic Excitation," AIAA/ASME/ASCE *23rd Structures, Structural Dynamics, and Materials Conference*, New Orleans, LA, May 1982, pp. 514-520.

11 Arnold, R.R., and Vaicaitis, R.R., "Nonlinear Response and Fatigue of Surface Panels by the Time Domain Monte Carlo Approach," *WRDC-TR-90-3081*, Wright-Patterson AFB, OH, May 1992.

12 Green, P.D., and Killey, A., "Time Domain Dynamic Finite Element Modeling in Acoustic Fatigue Design," *Proceedings 6th International Conference on Structural Dynamics*, Institute of Sound and Vibration Research, University of Southampton, UK, 1997, pp. 1007-1026.

13 Mei, C., Dhainaut, J.M., Duan, B., Spotswood, S.M., and Wolfe, H.F., "Nonlinear Random Response of Composite Panels in an Elevated Thermal Environment," *AFRL-VA-WP-TR-2000-3049*, Wright-Patterson AFB, OH, Oct. 2000.

14 McEvan, M.J., Wright, J.R., Copper, J.E., and Leung, A.Y.T., "A Combined Modal/Finite Element Analysis Technique for the Dynamic Response of a Non-linear Beam to Harmonic Excitation," *Journal of Sound and Vibration*, Vol. 243, No. 4, 2001, pp. 601-624.

15 Hollkamp, J.J., Gordon, R.W., and Spotswood S.M., "Nonlinear Sonic Fatigue Prediction From Finite Element Modal Models: a Comparison with Experiments," *44th Structures, Structural Dynamics, and Materials Conference*, AIAA-2003-1709, Norfolk, VA, April 2003.

16 Clarkson, B.L., "Review of Sonic Fatigue Technology," *NASA CR 4587*, April 1994, pp.1-75.

17 Mei, C., and Wolfe, H. F., "On Large Deflection Analysis in Acoustic Design," *Random Vibrations – Status and Recent Developments*. The Stephen H. Crandall Festschrift, Editors: I. Elishakoff and R. H. Lyon, Elsevier Applied Science Publishers, Amsterdam 1986, pp. 279-302.

18 Rychlik, I., "Rain-Flow Cycle Distribution for Ergodic Load Processes," *SIAM Journal of Applied Mathematics*, Vol. 48, 1988, pp. 662-679.

19 Dowling, N.E., "Fatigue Failure Predictions for Complicated Stress-Strain Histories," *Journal of Materials,* Vol. 7, 1972, pp. 71-87.

20 Dhainaut, J.M., and Mei, C., "Nonlinear Response and Fatigue Life of Isotropic Panels Subjected to Nonwhite Noise," *Journal of Aircraft*, Vol. 43, July-August 2006, pp. 975-979.

21 Przekop A., Rizzi S.A., and Groen D.S., "Nonlinear Acoustic Response of an Aircraft Fuselage Sidewall Structure by a Reduced-Order Analysis," *9th International Conference on Recent Advances in Structural Dynamics, Institute of*

Sound and Vibration Research, University of Southhampton, U.K., July 17-19, 2006.

22 Nguyen D. T., *Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions*, Springer, New York, 2006.

23 Nguyen D. T., *Parallel-Vector Equation Solvers for Finite Element Engineering Applications*, Plenum/Kluwer, New York, 2001.

24 Przekop Adam (private communication on the timing obtained from ABUQUS and MSC/NASTRAN), National Institute of Aerospace, NASA Larc, Mail stop 463, Hampton, VA, May 2005.

25 Hildebrand, F. B., "Advanced Calculus for Applications," Prentice-Hall, 2nd Edition, Englewood-Cliffs, New Jersey, 1976.

26 Lutes, D.L., and Sarkani, S., *Stochastic Analysis of Structural and Mechanical Vibrations*, Prentice-Hall, New Jersey, 1997.

27 Vaicaitis, R., "Recent Advances of Time Domain Approach for Nonlinear Response and Sonic Fatigue," *Proceedings 4th International Conference on Structural Dynamics*, ISVR, University of Southhampton, UK, July 1991, pp.84-103.

28 Crandall, S. H., and Mark, W. D., *Random Vibration in Mechanical Systems*, Academic Press, New York, 1963.

29 Zienkiewics O.C., and Taylor R.L., *The Finite Element Method*, Mc-Graw Hill, 4th Edition, Vol. 2, Barcelona, 1991.

30 Woinowsky-Kreiger, S., "The Effect of an Axial Force on the Vibration of Hinged Bars," *Journal of Applied Mechanics*, Vol. 17, 1950, pp. 35-37.

31 Rizzi, S. A., and Muravyov, A. A. "Improved Equivalent Linearization Implementations Using Nonlinear Stiffness Evaluation," *NASA TM-2001-210838*, March 2001.

32 Muravyov, A. A., and Rizzi, S. A., "Determination of Nonlinear Stiffness with Application to Random Vibration of Geometrically Nonlinear Structures," *Computers and Structures*, Vol. 81, No. 15, 2003, pp. 1513-1523.

33 Dhainaut, J. M., and Mei, C., "Nonlinear Response and Fatigue Life of Isotropic Panels Subjected to Nonwhite Pressure Fluctuations," AIAA 2002-1635, *43th Structures, Structural Dynamics and Materials Conference*, Denver, CO, April 2002

(CD-ROM). To appear in *Journal of Aircraft*.

34 Przekop, A., Guo, X., Azzouz, S., and Mei, C., "Reinvestigation of Nonlinear Random Response of Shallow Shells Using Finite Element Modal Formulation," AIAA 2004-1553, *45th Structures, Structural Dynamics and Materials Conference*, Palm Spring, CA, April 2004 (CD-ROM).

35 Shinozuka, M., "Monte Carlo Solution of Structural Dynamics," *International Journal of Computers and Structures*, Vol.2, 1972, pp. 855-874.

36 Shinozuka, M., and Wen, Y. K., "Monte Carlo Solution of Nonlinear Vibrations," *AIAA Journal*, Vol.10, No. 1, 1972, pp. 37-40.

37 Shinozuka, M., and Jan, D. M., "Digital Simulation of Random Processes and Its Applications," *Journal of Sound and Vibration*, Vol. 25, 1972, pp. 111-128.

38 Taylor, B. N., and Kuyatt, C. E., "Guidelines for Evaluating and Expressing the Uncertainity of NIST Measurement Results," NIST,TN 1297, 1994, p. 8.

39 Crandall, S., and Zhu, W., "Random Vibration: A survey of Recent Development," *Journal of Applied Mechanics*, Vol. 50, No. 5, 1983, pp.953-962.

40 Iwan, W. D., and Yang, M. I., "Application of Statistical Linearization Techniques to Nonlinear Multi-Degree of Freedom Systems," *Journal of Applied Mechanics*, Vol. 39, 1972, pp. 545-550.

41 Heuer, R., Irschik, H., and Ziegler, F., "Nonlinear Random Vibrations of Thermally Buckled Skew Plates," *Probabilistic Engineering Mechanics*, Vol.8, No. 3-4, 1993, pp. 265-271.

42 Elishakoff, I., and Zhang, X., "An Appraisal of Different Stochastic Linearization Techniques," *Journal of Sound and Vibration*, Vol. 153, No. 2, 1992, pp. 370-375.

43 Roberts, J. B., and Spanos, P. D.,"Random Vibration of Statistical Linearization," John Wiley & Sons, New York 1990.

44 Ng, C. F.," Nonlinear and Snap-Through Response of Curved Panels to Intense Acoustic Excitation," *Journal of Aircraft*, Vol.26, No.3, 1989, pp.281-288.

45 Lee, J., "Large Amplitude Plate Vibration in an Elevated Thermal Environment," *Applied Mechanics Reviews*, Vol. 46, part 2, No. 11, 1993, pp. 242-254.

46 Locke, J., and Mei, C., "Finite Element, Large Deflection Random Response of Thermally Buckled Beams," *AIAA Journal*, Vol. 28, No. 12, 1990, pp. 2125-2131.

47 Arnold, R. R., and Vaicaitis, R., "Nonlinear Response and Fatigue of Surface Panels by the Time Domain Monte Carlo Approach," WRDC-TR-90-3081, Wright-Patterson AFB, OH, 1990.

48 Vaicaitis, R., "Generalized Random Forces for Rectangular Panels," *AIAA Journal*, Vol. 11, No. 7, 1973, pp. 984-988.

49 Abdel-Motagaly, K., Chen, R., and Mei, C., "Nonlinear Flutter of Composite Panels Under Yawed Supersonic Flow Using Finite Elements," *AIAA Journal*, Vol. 37, No. 9, 1999, pp.1025-1032.

50 Abdel-Motagaly, K., Duan B., and Mei, C., "Nonlinear Response of Composite Panels under Combined Acoustic Excitation and Aerodynamic Pressure," *40th Structures, Structural Dynamics and Materials Conference*, St Louis, MO, 1999, pp.1963-1972. Also *AIAA Journal*, Vol. 38, No. 9, 2000, pp.1534-1542.

51 Dhainaut, J. M., Duan, B., Mei, C., Spotttswood, S. M., and Wolfe, H. F., "Nonlinear Response of Composites Panels to Random Excitations at Elevated Temperatures," *7th International Conference on Recent Advances in Structural Dynamics*, Southhampton, England, July 2000, pp. 769-784.

52 Dowell, E. H., "Aeroelasticity of Plates and Shells," Noordhoff International Publishing, The Netherlands, 1975.

53 Dhainaut, J. M., Cheng, G., and Mei, C., "Nonlinear Response of Plates under Uniform Random Loads Unsynchronized in Time," AIAA 2007-2111, *48th Structures, Structural Dynamics and Materials Conference,* Honolulu, Hawaii, April 2007.

54 Bathe K. J., *Finite Element Procedures*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

55 Bogner, F. K., Fox, R. L., and Schmit, L.A," The Generation of Inter-Element Compatible Stiffness and Mass Matrices by the Use of Interpolation Formulas," AFAFFDL-TR-66-80, Wright-Patterson AFB, OH, 1996, pp. 396-443.

56 Locke, J. E., "A Finite Element Formulation for the Large Deflection Random Response of Thermally Buckled Structures," Ph.D. Dissertation, Old Dominion University, Norfolk, VA, 1988.

57 Shi, Y., and Mei, C., "Coexisting Thermal Post Buckling of Composite Plates with

Initial Imperfections Using Finite Element Modal Methods," *Proceedings 37th Structures, Structural Dynamics, and Materials Conference*, Salt Lake City, UT, April 1996, pp. 1355-1362.

58 Shi, Y., Lee, R., and Mei, C., "A Finite Element Multimode Method to Nonlinear Free Vibrations of Composite Plates," *AIAA Journal*, 1997, Vol. 35, pp.159-166.

59 Pian, T.H.H., "Derivation of element stiffness matrices by assumed stress distribution," *AIAA Journal*, 1964, Vol. 2, pp. 1332-1336.

60 Pian, T.H.H., and Tong, P., "Basis of finite element methods for solid continua," *International Journal for Numerical Methods in Engineering*, 1969, Vol. 1, pp. 3-28.

61 Wood, R. D., and Schrefler, B., "Geometrically Non-Linear Analysis –A Correlation of Finite Element Notations," *International Journal for Numerical Methods in Engineering*, Vol. 12, 1978, pp. 635-642.

62 Barlow, J., "Optimal Stress Locations in Finite Element Models," *International Journal for Numerical Methods in Engineering*, Vol. 10, 1976, pp. 243-251.

63 Barlow, J., "Optimal Stress Locations in Finite Element Models," *International Journal for Numerical Methods in Engineering*, Vol. 10, 1976, pp. 243-251.

64 Cook, R. D., Malkus, D. S. and Plesha, M. E., "Concepts and Applications of Finite Element Analysis," 3rd Edition, John Wiley, New York, 1989.

65 Robinson, J. H., "Finite Element Formulation and Numerical Simulation of the Large Deflection Random Vibration of Laminated Composite Plates," MS Thesis, Old Dominion University, 1990.

CURRICULUM VITA
for
Swati Chokshi

DEGREES:

Doctor of Philosophy (Civil Engineering), Old Dominion University, Norfolk, Virginia, December 2008.

Master of Engineering (Civil Engineering), Maharaja Sayajirao University, Baroda, Gujarat, August 2002.

Bachelor of Engineering (Civil Engineering), North Gujarat University, Modasa, Gujarat, March 1998.

AWARDS AND HONORS:

1. Gold medal, for ranking first in the North Gujarat University, Bachelor of Engineering, INDIA, 1996.
2. Silver medal, for scoring highest marks in final year of engineering, Institute of Engineers, INDIA, 1996.
3. Certification of Graduate Aptitude Test in Engineering (GATE), 2000.
4. Best Teacher Award, Government Engineering College, INDIA, 2000.

PROFESSIONAL CHRONOLOGY:

NASA Langley Center, Hampton, Virginia

Research Assistant, September 2002 - August 2006.

Department of Civil and Environmental Engineering, Old Dominion University, Norfolk, Virginia

Teaching Assistant, September 2006 - April 2008.

EMPLOYMENT:

Halcrow, Inc., Virginia Beach, Virginia

Structural Engineer, August 2008 – Present

Department of Applied Mechanics and Civil Engineering, Government Engineering College, Modasa, India

Assistant Professor, July 1998 - April 2000.

SCHOLARLY ACTIVITIES COMPLETED:

1. Chokshi, S.M.; Mei, C.; Nguyen, D.T.; and Rajan. S.; "Nonlinear Random Response of Large-Scale Sparse Finite Element Structural Problems." *Journal of Computational and Applied Mechanics*, Vol. 9, No. 1, (2008), pp. 1–12.

2. Chokshi, S.M.; Mei, C.; Nguyen, D.T., "Nonlinear Random Response of Large-Scale Finite Element Structural Problems under Un-synchronized Loading with Parallel-Sparse Methodology." Submitted to *Journal of Finite Element Analysis and Design*, December 2008.

3. Wilson, J. W.; Korte, J. J.; Sobieszczanski-Sobieski, J.; Badavi, F. F., Chokshi, S. M.; Martinovic, Z. N.; Cerro, J. A.; and Qualls, G. D. "Radiation Shielding, MDO Processes, and RLV Design." *Journal of American Institute of Aeronautics and Astronautics (AIAA) SPACE 2003 Conference* and Exhibit, September 23-25, 2003, Long Beach, California In Proceedings, AIAA Paper No. 2003-6259.

4. Chokshi, S.M.; Mei, C.; Nguyen, D.T.; and Rajan. S., "Nonlinear Random Response of Large-Scale Sparse Finite Element Structural Problems." *Proceedings, American Institute of Aeronautics and Astronautics Structures, Structural Dynamics, and Materials (SDM) Conference*, April, 2007.