Development of the Velocity Transformation Function of Damped Flat Shell Finite Element for the Experimental Spatial Dynamics Modeling

By

Kyongchan Song

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

APPROVED:

____________________
Robert L. West, Chair

____________________  ____________________
Don Leo                                          Reginald G. Mitchiner

December 1, 2000
Blacksburg, Virginia
Development of the Velocity Transformation Function of Damped Flat Shell Finite Element for the Experimental Spatial Dynamics Modeling

Kyongchan Song

(Abstract)

Experimental Spatial Dynamics Modeling (ESDM) is the new process of constructing a three dimensional, complex-valued dynamic model of a harmonically vibrating structure using numerical models and laser-based experimental data obtained from a Scanning Laser Doppler Vibrometer (SLDV).

In ESDM process, a finite element formulation is used to construct a numerical model of a structure. A conventional finite element such as rod, beam, or plate element, can be used to construct the numerical model of a structure from its mid-plane. In this research, the damped flat shell element is developed to construct the numerical models of a cantilever beam and a simply supported flat plate.

The velocity transformation function developed in this research will make possible to use the FE model, constructed by the damped flat shell element, and the laser-based experimental data within a framework of ESDM in the consistent manner.
Acknowledgments

I acknowledge my advisor, Dr Robert L West for all his time and effort to help finish my research. Without his understanding and consideration, it would not possible to complete this work. I am also thankful for the willingness of my other committee member, Dr Mitchiner and Dr. Don Leo to serve on my committee.

I appreciate Mr. Ben Poe and Mr. Jemie Archaul for all their support and help.
Lastly I thank to my family and friends for their support and love.
Contents

Abstract ii
Acknowledgment iii
List of Figure vii
List of Tables x

1. Instruction 1
   1.1 General overview and statement of need .............................................. 1
   1.2 Research hypothesis and goals ............................................................ 2
   1.3 Scope of the research ........................................................................... 3
   1.4 Outline of the thesis ............................................................................. 4
       1.4.1 Chapter 1 ....................................................................................... 4
       1.4.2 Chapter 2 ....................................................................................... 4
       1.4.3 Chapter 3 ....................................................................................... 5
       1.4.4 Chapter 4 ....................................................................................... 5
       1.4.5 Chapter 5 ....................................................................................... 5
       1.4.6 Chapter 6 ....................................................................................... 5
   1.5 Literature review .................................................................................... 5

2. Concept of Experimental Spatial Dynamic Modeling 9
   2.1 Experimental spatial dynamic modeling .................................................. 9
   2.2 Response of a linear system to harmonic excitation .............................. 11
   2.3 Velocity measurement by the SLDV ....................................................... 13
   2.4 Error function of ESDM ........................................................................ 18

   3.1 Finite element method in vibration and dynamic problems .................. 20
       3.1.1 Equation of motion (Single Degree of Freedom System) ............... 21
       3.1.2 Equation of motion (Multiple Degree of Freedom System) .......... 24
       3.1.3 Displacement and shape function ................................................... 26
       3.1.4 The Mass Matrix ........................................................................... 28
3.1.5 The Stiffness Matrix ................................................................. 29
3.1.6 The Damping Matrix ............................................................... 32
3.2 Flat shell element ................................................................... 34
  3.2.1 Bi-linear plane stress element ............................................. 36
  3.2.2 Kirchoff plate element .......................................................... 41
3.3 Frequency response function and the solution of normal modes
  of the structure .......................................................................... 48
  3.3.1 Frequency response function ................................................. 48
  3.3.2 Natural frequencies and mode shapes ................................... 49
3.4 Summary .................................................................................. 50

4. Velocity Transformation Function of Flat Shell Element 51
  4.1 Velocity field on the mid plane of flat shell element .................. 51
  4.2 The velocity transfer function from the mid-plane to surface
    of the structure ......................................................................... 54
  4.3 Summary .................................................................................. 64

5. Test Cases 67
  5.1 Cantilever beam (Undamped system) ....................................... 67
    5.1.1 Closed form solution ......................................................... 68
    5.1.2 Finite element model .......................................................... 70
    5.1.3 Dynamic response of the FE model ..................................... 72
    5.1.4 Solution of normal modes .................................................. 76
  5.2 Cantilever beam (Damped system) .......................................... 77
    5.2.1 Dynamic response of the FE model ..................................... 77
    5.2.2 Solution of normal modes .................................................. 83
  5.3 Dynamic response model of the cantilever beam ...................... 85
    5.3.1 Dynamic response model for a undamped system ............... 85
    5.3.2 Dynamic response model for a damped system .................. 89
  5.4 Simply supported rectangular plate ........................................... 95
    5.4.1 Finite Element Model .......................................................... 96
5.4.2 Dynamic response of the FE model ........................................... 99
5.4.3 Solution of normal modes ....................................................... 104
5.4.4 Dynamic response model of the simply supported plate .......... 108
5.5 Summary .............................................................................. 114

6. Conclusions ................................................................. 115
  6.1 Summary of thesis ................................................................. 115
  6.2 Conclusions of this research .................................................. 117
  6.3 Recommendations for future work ......................................... 118

Reference ........................................................................... 120

Appendix A
  Matlab source files for Flat Shell Finite Element Program (Undamped System) 122

Appendix B
  Matlab source files for Flat Shell Finite Element Program (Damped System) 139

Appendix C
  Matlab sources files to determine the velocity on the mid-plane of a structure 151

Appendix D
  Matlab sources files for the velocity transformation function of the flat shell element 172

Vita ...................................................................................... 192
List of Figures

Figure 2.1 Experimental Spatial Dynamic Modeling.................................................10
Figure 2.2 Forcing and response function in the complex plane..............................12
Figure 2.3 Relationship between measured velocity, $V_L$, and modeled velocity, $V_s$, in the XY plane.................................................................14
Figure 2.4 Magnitude of measure velocity and forcing signal .................................17
Figure 3.1 Single degree of freedom system............................................................21
Figure 3.2 Vector relationship for forced vibration of the SDOF system with damping.................................................................22
Figure 3.3 Two degrees of freedom mass, spring, and damper system....................24
Figure 3.4 Rod element............................................................................................27
Figure 3.5 Flat-shell element with five degree of freedom per node.......................35
Figure 3.6 Sub matrices of the bi-linear plane element...........................................40
Figure 3.7 Sub matrices of the kirchoff plate element.............................................45
Figure 3.8 Consistent matrix for the bi-linear plane element..................................46
Figure 3.9 Consistent mass matrix for the kirchoff plate element..........................47
Figure 4.1 Geometry and displacements of differential slice of a flat shell element before and after loading in XZ plane..................................................55
Figure 4.2 Geometry and displacements of differential slice of a flat shell element before and after loading in YZ plane..................................................55
Figure 4.3 Close up of top half of flat shell element in XZ plane..............................58
Figure 4.4 Close up of top half of flat shell element in YZ plane..............................58
Figure 4.5 The velocity transfer function matrix.....................................................63
Figure 5.1 The Cantilever beam...............................................................................68
Figure 5.2 Convergence of displacement at the free end of the FE model...............71
Figure 5.3 The mesh of the converged FE model.....................................................71
Figure 5.4 The deformed shape of the cantilever beam..........................................72
Figure 5.5 Receptance of the cantilever beam (at node 22, undamped system).......75
Figure 5.6 Mobility of the cantilever beam (at node 22, undamped system)..........75
Figure 5.7  (a) First bending mode shape
(b) Second bending mode shape
(c) Third bending mode shape
(d) First twist mode shape

Figure 5.8 Real, imaginary, magnitude, and relative phase angle of transverse displacement

Figure 5.9 Undamped and damped transverse displacement of the cantilever beam

Figure 5.10 Receptance of the cantilever beam (at node 22, damped system)

Figure 5.11 Mobility of the cantilever beam (at node 22, damped system)

Figure 5.12 (a) Mobility of the cantilever beam (at node 22, damping ratio = 2%)
(b) Mobility of the cantilever beam (at node 22, damping ratio = 5%)

Figure 5.13 25 laser measurement locations within the flat shell element

Figure 5.14 Imaginary velocity field of on the mid-plane of the FE model (undamped system)

Figure 5.15 Imaginary velocity field of on the top surface of cantilever beam (undamped system)

Figure 5.16 Axial (x-axis) and lateral (y-axis) directional velocity of the beam on the mid-plane

Figure 5.17 Transverse velocity of the beam on the mid-plane

Figure 5.18 X directional (axial) velocity of the beam on the top surface

Figure 5.19 Y directional (lateral) velocity of the beam on the top surface

Figure 5.20 Transverse velocity of the beam on the top surface

Figure 5.21 The simply supported rectangular plate

Figure 5.22 Real component of the transverse velocity of the rectangular plate measured by SLDV (79.0 Hz)

Figure 5.23 Real component of the transverse velocity of the rectangular plate (79.0 Hz)

Figure 5.24 Convergence of the maximum magnitude of transverse velocity of the rectangular plate
Figure 5.25  Nodes and elements of 30 element Finite Element model……………………99
Figure 5.26  Transverse displacement of the rectangular plate…………………………100
Figure 5.27  Receptance of the rectangular plate (at node 14) .........................102
Figure 5.28  Mobility of the rectangular plate (at node 14)..............................102
Figure 5.29  (a) Mobility of the rectangular plate (at node 14, $\gamma = 0.5\%$)
(b) Mobility of the rectangular plate (at node 14, $\gamma = 5\%$)......................103
Figure 5.30  (a) The first mode shape (83.241 Hz) obtained by FSFE program
(b) The second mode shape (180.15 Hz) obtained by FSFE program
(c) The third mode shape (236.32 Hz) obtained by FSFE program ..........105
Figure 5.31  (a) The first mode shape (83.241 Hz) obtained by I-DEAS
(b) The second mode shape (180.15 Hz) obtained by I-DEAS.................106
(c) The third mode shape (236.32 Hz) obtained by I-DEAS...............107
Figure 5.32  Axial (x-axis) and lateral (y-axis) directional velocity of the plate
on the mid-plane.................................................................109
Figure 5.33  Transverse (z-axis) velocity of the plate on the mid-plane.............110
Figure 5.34  Axial (x-axis) velocity of the plate on the top surface.................111
Figure 5.35  Lateral (y-axis) velocity of the plate on the top surface...............112
Figure 5.36  Transverse (z-axis) velocity of the plate on the top surface...........113
List of Tables

Table 3.1    Forcing function and dynamic response of structure.................................25
Table 3.2    Definition of the frequency response function..........................................49
Table 5.1    Material properties, geometry, and load of the cantilevered beam..............68
Table 5.2    Natural frequencies of the cantilever beam.............................................69
Table 5.3    Difference between closed form solution and result
of the FE models.................................................................70
Table 5.4    The maximum displacement of the cantilevered beam
(umdamped system)...............................................................72
Table 5.5    Reaction forces of the cantilever beam (umdamped system).........................73
Table 5.6    Natural frequencies of the cantilever beam (umdamped system)....................76
Table 5.7    Displacement of the cantilever beam at node 12 (damped system).................78
Table 5.8    Reaction forces of cantilever beam (damped system)..................................79
Table 5.9    Natural frequencies of the cantilever beam (damped system).........................83
Table 5.10   Material properties, geometry , and load of the rectangular plate.................96
Table 5.11   Difference between the experimental result and output of FE models.............97
Table 5.12   Displacements of rectangular plate at node 21...........................................100
Table 5.13   Magnitude of transverse reaction forces of the rectangular plate
(unit of force is N).................................................................101
Table 5.14   Relative phase angle of transverse reaction forces of
the rectangular plate (unit of angle is degree)......................................101
Table 5.15   Frequencies of free oscillation and decay rate of the rectangular plate..........104
Table 5.16   Natural frequency and mode shape of the rectangular plate.........................104
Chapter 1
Introduction

1.1 General overview and Statement of Need

In the design process of a product, the use of numerical models to obtain the dynamic response of structures has been increased to save money and time and eventually to make a better product. Therefore, constructing an accurate numerical model becomes an essential part early in the design process. The development of the high speed digital computer has made it possible to develop and solve numerical models capable of accurate numerical approximation of the dynamic response of a complicated structure. However, the proper use of a numerical model in the design process requires some verification usually as an experimental test to validate or to upgrade the numerical model.

A common experimental test to find the dynamic response of structures is frequency response testing. Although this method can obtain dynamic response information of the structure over a wide frequency range, the number of spatial degrees of freedom of the structure is limited to the restriction on the number and location of accelerometers installed on the structure. Also, surface-mounted accelerometers add their own mass to the structure. Therefore, there is a need for developing accurate models of structures for the dynamic response with a large number of spatial degrees of freedom.

One of the new procedures for modeling the dynamic response of a structure that incorporates numerical approximation and the experimental testing of the structure is Experimental Spatial Dynamics Modeling (ESDM) (Montgomery, 1994). Experimental Spatial Dynamics Modeling (ESDM) is the process of constructing a three dimensional dynamic model using numerical models and laser-based experimental data obtained from a Scanning Laser Doppler Vibrometer (SLDV). There are several advantages obtaining experimental data by a SLDV. The experiment with a SLDV is a non-contacting dynamic measuring method. Therefore, there is no need for a large number of attached accelerometers and wires on a test
structure. Also the high spatial density measurement capability of a SLDV can be exploited to
developed model with a large number of spatial degrees of freedom of a structure.

In the ESDM process, a finite element formulation is used to develop a numerical
approximation of the dynamic response of a structure. The dynamic responses of the structure
obtained from the Finite Element model (FE model) consists of node displacements and
velocities of the FE model. Node displacements and velocities of the structural elements of FE
models, such as beam, plate, and shell, are developed from the mid plane of the modeled
structure. Because a SLDV measures the velocity field of the structure on the surface of the
structure, it is necessary to transfer the node displacement and velocities of the FE model from
the mid plane to the surface of the structure so that the result of the FE model can be consistent
with laser-based experimental data measured by a SLDV. Therefore, it is important to develop
the velocity transformation function within the framework of ESDM which is consistent with
conventional FE models for direct correlation of the dynamic response model with laser-based
experimental data.

1.2 Research Hypothesis and Goals

The hypothesis statement for this research is given that the Finite Element model
determines dynamic response on the mid-plane of the structure and laser-based experiment data
are obtained from the surface of the structure by a Scanning Laser Doppler Vibrometer (SLDV).
The development of a velocity transformation function for the Finite Element model, dynamic
response on the outer surface of the FE model can be obtained and can be made consistent with
laser-based experimental data measured by a SLDV.

The research goals of this thesis are
1. Develop a damped flat shell element which can be used to construct a FE model of a
structure and be capable of simulating laser-based experimental data obtained by a
Scanning Laser Doppler Vibrometer (SLDV).
2. Develop the velocity transformation function which will transfer dynamic responses of
the FE model from the mid plane to the outer surface of the FE model where laser-based
experimental data is acquired by a SLDV.
3. Determine the dynamic response on the surface of structures using the FE model
constructed by the damped flat shell FE and the velocity transformation function.

1.3 Scope of the research

The overall scope of this research is to rework the formulation or the framework of
ESDM developed by Montgomery (1994) to make the formulation consistent with a
conventional FE model. In the previous research, the error function of the numerical models of
the structure with the laser-based experimental data was developed from the relationship between
the dynamic response on the surface of the numerical models and the laser-based experimental
data obtained by a SLDV. The numerical models of structures will be used to find the dynamic
response and the solutions of the normal mode of structures. Also the laser-based experimental
data obtained by the SLDV are used to verify numerical models of the structure.

The main goal of this research is to develop the formulation in order to use the numerical
model and experimental data in a consistent manner within the framework of ESDM. A
conventional FE model of structures is used as the numerical model. To develop and
demonstrate the formulation, called the velocity transformation function, the candidate FE model
is needed. For this research, the flat shell element developed by Venter (1995) is used to
construct the candidate FE model of structures. Because the velocity transformation function
will be developed from the element chosen to construct the candidate FE model, it is important
to understand properties and the limitations of this flat shell element. In Venter’s research, the
use of the flat shell element was limited to consider the model as an undamped system only.
This means that only the real part of generalized complex dynamic response was used, and the
relative phase response was either 0 or 180 degrees relative to the reference force. However,
since the relative phase response is not always 0 or 180 degrees in real structures, the flat shell
element needs to be generalized to construct FE models of the damped system.
The damping formulation used in this research uses the structural damping factor and complex stiffness matrix. The flat shell element will be improved and be able to construct the FE model of damped systems. To validate the FE models developed in this research, the dynamic responses and the solution of normal mode of structures obtained by FE models will be compared with the closed form solution, experimental data, or output of I-DEAS, a commercial FE package. The velocity transformation function will be developed based on the properties of the damped flat shell element, a linear system and the deformation mechanics of the structure. To demonstrate the application of the velocity transformation function, the velocity transformation function and the result of the FE models will be used to product the velocities on the surface of structures.

1.4 Outline of the thesis

This thesis consists of six chapters. An outline of the thesis can be summarized as following.

1.4.1 Chapter 1

Chapter 1 covers the introduction and literature review.

1.4.2 Chapter 2

Chapter 2 discusses the general framework of Experimental Spatial Dynamics. The relationship between the numerical model, which is a conventional FE model, and laser-based experimental data obtained by a SLDV is explained. The formulation used to compare to a FE model with the laser-based experimental data taken at specific spatial locations is developed. This formulation will test the candidate FE model with experimental data and later can be used to update the FE model of the structure to provide accurate prediction of dynamic response.
1.4.3 Chapter 3

Chapter 3 describes the development of the flat shell element used in this research. First, the flat shell element is developed for an undamped system. Then using a structural damping factor and complex stiffness formulation, the flat shell element for a damped system is developed. Dynamic response and solution of normal modes of the FE model are discussed.

1.4.4 Chapter 4

The velocity transformation function of the flat shell element is developed in Chapter 4. This velocity transformation function makes the dynamic response obtained by FE models to be consistent with laser-based experimental data gathered by a SLDV.

1.4.5 Chapter 5

In Chapter 5, candidate FE models of test structures are constructed using flat shell elements. Dynamic response and solution of normal modes of candidate FE models are compared with the closed form solutions or results obtained from FE model constructed by I-DEAS. Using the velocity transformation function developed in Chapter 4 and velocities of test structures obtained from candidate FE models, the experimental data obtained by SLDV are simulated.

1.4.6 Chapter 6

Chapter 6 covers the conclusions of this research and suggests future works.

1.5 Literature Review

The previous research on ESDM, an early velocity transformation function, the Finite Element Method in structural dynamics and vibration, and the formulation of various structural Finite Elements were reviewed to set the direction of this research.

Montgomery (1994) developed the general framework of ESDM and demonstrated the validity of ESDM. In his work, he generated the mesh of the FE model not on the mid plane of a
structure but on the surface of the structure. Because the mesh of FE models was generated on the surface of the structure, the output of the FE models could be used directly with the laser-based experimental data.

In ESDM, Finite Element models are used to determine the dynamic response of the vibrating structure. Therefore, it is necessary to develop Finite Element model which can represent structures in ESDM. Cook (1995) explained general concepts of the FE model in dynamic and vibration analysis of structures. Cook showed detailed procedures to generating various kinds of elements and the application of them. The most popular elements in structural mechanics are base on the assumption that displacements within an element are adequately described by a simple polynomial. In this research, the flat shell element is used to construct the Finite Element models of structures. This flat shell element was developed by Venter (1995). In his research, he developed the flat shell element to construct numerical models of structures and to find the sensitivity of the error function of the non-parametric experimental model used in ESDM with respect to structure parameters. Venter also studied of the sensitivity error function with respect to error in data due to the laser position and orientation. Venter’s flat shell element is a rectangular element with four nodal points at each corner with three translation degrees of freedom and two rotation degrees of freedom at each node. The flat shell element is constructed by superposition of the components of the bi-linear plane stress and Kirchoff plate elements. Axial (x-axis) and lateral (y-axis) nodal displacements of a flat-shell element will be obtained from components of the bi-linear element. The transverse (z-axis) bending displacement of the flat shell element will be obtained from components of the Kirchoff plate element.

The bi-linear plane stress element is a rectangular element with four nodal points, one at each corner, and two translation degrees of freedom at each node. The bi-linear plane stress element is called bi-linear because coefficients in the matrix come from the product of two linear expressions with the sides remaining straight when the element deforms. The bi-linear plane stress element is based on an assumed bi-linear, in-plane displacement polynomial. Cook, Malkus, and Plesha (1989) developed the stiffness and mass matrices of the bi-linear plane stress element using the bi-linear Lagranian shape functions.

The Kirchoff plate element is a rectangular element with four nodal points, one at each corner, and three degrees of freedom at each node. This element is based on an assumed cubic, transverse displacement polynomial and thin plate theory. Haug, Choi, and Komkov (1986)
developed the stiffness and mass matrices of Kirchoff plate element using the Hermitian shape functions. Both stiffness and mass matrices of the Kirchoff plate element depends on material properties and plate thickness, which may be taken as design parameters.

Venter’s research was limited to the use of the flat shell elements representing structures considered as an undamped system. The flat shell element will be updated to account for damping in the system. To update the flat shell element, the damping of structures was reviewed.

The damping of structures can be described in terms of viscous damping only as a rough approximation. However, many structural materials, such as steel and aluminum, when subject to cyclic loading, show a behavior that can be described as structural damping (Genta, 1995). Using structural damping, the linear equation of motion of damped system can be rewritten as

\[
[M]\{\ddot{x}\} + [K][1 + i\gamma]\{x\} = \{F_o\}e^{i\omega t}
\]

(1.1)

where \([M]\) is mass matrix, \([K][1 + i\gamma]\) is the complex stiffness matrix, \(\gamma\) is the structural damping factor and \(\{F_o\}e^{i\omega t}\) is applied harmonic forces (Thomson, 1998). As shown in Equation 1.1, structural damping is a type of proportional damping. The damping matrix is proportional to the stiffness matrix and can be obtained using the structural damping factor and stiffness matrix. Hurty and Rubinstein (1964) gave more details about proportional damping. They explained properties and formulations of proportional damping.

When beam, plate, or shell elements are used to construct FE models to find the dynamic response of structures, the deformation of the neutral axis (for beam), mid plane (for plates), or mid surface (for shell) of structures are considered as a whole. The equation of equilibrium resulting from the variational process applied to the total potential and kinetic energy indicates that this simplification of deformation makes FE models which are stiffer system than actual structures. Dym and Shames (1985) showed the relationship between deformation on the mid plane of the structures and deformation on the surface of structures, and develop improved theories of beams and plates. In Dym and Shames’ improved theories of beams and plates, they showed that the displacement on the surface of beams and plate could be obtained using displacement of the neutral axis or the mid plane and deformation mechanics. Using Dym and
Shames’ improved theories of plates, the velocity transformation function will be developed in this research.

Siethoff (1998) developed a consistent method to update parameters in FE models which were used for vibration analysis using experimental data gathered with a SLDV. In order to use the result of the FE models in a consistent manner with laser-based experimental data, Siethoff introduced the velocity transformation function that transfers the velocities of the FE model from the mid plane to the surface where the laser measures. However, Siethoff’s velocity transformation function has two limitations. First, he used the beam element which is a combination of a rod element and a simple plane beam element to make FE models of structures. Therefore, velocities represented in the FE model were limited to transverse and axial velocities. Secondly, since the FE models of structures were constructed based only an undamped system, the velocity transformation function is also limited to undamped system.

This research will bring together the frameworks presented by Venter and Siethoff. By using the flat shell element developed by Venter and the velocity transformation presented by Siethoff and combining their works, it will be able to construct the generalized FE model that can be used in ESDM developed by Mongomery. This generalized FE model will also benefit to increasing the applications of ESDM.
Chapter 2
Concept of Experimental Spatial Dynamic Modeling

Experimental Spatial Dynamic Modeling (ESDM) is used to construct a complex valued continuous three dimensional velocity field of a vibrating structure by laser-based experimental data and numerical models. Numerical models used in this research are finite element models (FE model). In this chapter, the fundamental formulation of ESDM is discussed. The relationship between laser-based experimental data obtained by a Scanning Laser Doppler Vibrometer (SLDV) and structural velocity on the surface of structures generated by FE models will be derived. Using this relationship, the post processing formulation of the FE model that simulates the experimental data from the output of the FE model is developed.

2.1 Experimental spatial dynamic modeling

The first step of ESDM is to sample the experimental data of the surface shape and velocity of a vibrating structure using a SLDV. The resulting surface shape data and surface velocity information generate the actual 3-D experimental dynamic response model of the structure. The ESDM finite element model is developed for modeling velocity response as a statistically qualified complex valued continuous three dimensional velocity field. The experimental dynamic response model will allow spatial analysis and visualization of experimental dynamics directly analogous to the Finite Element Method in computational modeling. Montgomery (1994) defined the goal of ESDM as “The goal of ESDM is to develop statistically qualified experimentally derived spatial dynamics models of actual test structure that are the experimental counterparts to analytical finite element models”. Figure 2.1 shows the outline of ESDM. In this research, FE models of vibrating structures are constructed by damped flat shell elements. These FE models of vibrating structures are used for modeling dynamic response as a statistically qualified complex valued continuous three dimensional velocity field. Using the velocity transformation function, this approximate dynamic response model of the structure constructed by the FE model will be used in consistent manner with the experimental
model in the frame work of ESDM. In this research, the dynamic responses on the surface of the structure from the dynamic response obtained by the FE model and the velocity transformation function are constructed. The application of the velocity transformation function will be demonstrated.

![Flowchart](image)

**Figure 2.1 Experimental Spatial Dynamic Modeling**
2.2 Response of a linear system to harmonic excitation

In this research, structures are assumed to behave linearly. Assuming a structure is a linear system, the response of the structure can be expressed as a second-order linear system and the equation of motion of the structure can be written as

\[ M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = F(t) \]

Where \( M, C, \) and \( K \) are constants for mass, damping, and stiffness respectively.

\( F(t) \) is a forcing function

\( x(t) \) is the response

(2.1)

Assuming the excitation forcing function, \( F(t) \), is a harmonic function in the form

\[ F(t) = \tilde{f} e^{i\omega t} \]

Where \( \tilde{f} \) is the amplitude of the forcing function, \( \omega \) is the circular forcing frequency. The response of the structure is found in the form (Meirovitch, 1997)

\[ x(t) = \bar{x} e^{i(\omega t + \phi)} \]

(2.3)

Where \( \bar{x} \) is the amplitude of the response, \( \omega \) is the same circular forcing frequency, and \( \phi \) is the relative phase angle of the response to the reference forcing function. The response of the structure obtained by the FE model is the displacement of the structure. As shown in Equation 2.2 and 2.3, an important property of the linear system is that the response function of structure is also a harmonic function at the same frequency as the forcing frequency, \( \omega \). Figure 2.2 shows the forcing function and response function in the complex plane.
Another important property of a linear system is that the response due to multiple forcing functions is simply the sum of the response of each individual forcing function. For example, the forcing function is a linear combination of harmonic functions at different frequencies

$$F(t) = \sum_{j=1}^{K} F_j(t) = F_1(t) + F_2(t) + \cdots + F_K(t)$$

$$F_j(t) = \tilde{F}_j e^{i(\omega_j + \phi_j)}, j = 1, 2, 3, \ldots, k$$

(2.4)

Then the response of the structure is the sum of the response of each forcing frequency

$$x(t) = \sum_{j=1}^{K} x_j(t) = x_1(t) + x_2(t) + \cdots + x_K(t)$$

$$x_j(t) = \tilde{x}_j e^{i(\omega_j + \phi_j)}, j = 1, 2, 3, \ldots, k$$

(2.5)

where \(x_j(t)\) is the response of the structure at each frequency.

$$M\ddot{x}_j(t) + C\dot{x}_j(t) + Kx_j(t) = F_j(t)$$

$$j = 1, 2, 3, \ldots, k$$

(2.6)

These properties of the linear system are important to make assumptions about the structure’s dynamic response and will be used in gathering experimental data and modeling the dynamic response of structures analytically.
2.3 Velocity measurement by SLDV

When a structure is oscillating due to a harmonic function, \( F(t) = \bar{F} e^{i\omega t} \), a laser vibrometer measures the velocity sample at the location where the laser strikes the surface of the structure at an instant of time, \( t \) where \( \bar{F} \) is the force amplitude and \( \omega \) is forcing frequency. Each measurement sample is the magnitude of the structure’s velocity vector projected along with the laser beam’s line-of-sight. Using the position and orientation of the laser relative to the structure’s coordinate systems, the components of the measured velocity in structural coordinates can be determined. Therefore, positioning a laser in different locations about the structure, the 3D velocity of the vibrating structure can be obtained.

The modeled velocity on the surface of the oscillating structure can be obtained by taking the first derivative of the displacement of the FE model respect to time. The velocity vector at the any point on the surface of the structure can be written in terms of components of the structure coordinate system.

\[
\bar{V}_S = \bar{u} \hat{i} + \bar{v} \hat{j} + \bar{w} \hat{k}
\]  

(2.7)

A unit vector in the direction of the incident laser beam is written as

\[
\bar{V}_U = \cos(\theta_X) \hat{i} + \cos(\theta_Y) \hat{j} + \cos(\theta_Z) \hat{k}
\]  

(2.8)

Where \( \theta_X \), \( \theta_Y \), and \( \theta_Z \) are the angle between the positive x, y, and z coordinate directions in the structure’s coordinate system and the incident laser beam. The magnitude of the projection of the modeled structure’s velocity vector onto the laser beam direction will be the magnitude of velocity vector, \( |\bar{V}_L| \), which is measured by SLDV. Therefore, the dot product of the unit vector, \( \bar{V}_U \), in the direction of the laser beam’s line-of-sight and the magnitude of modeled structural velocity vector can generate the magnitude of measured velocity vector.

\[
|\bar{V}_L| = |\bar{V}_S \cdot \bar{V}_U|
\]  

(2.9)
Figure 2.3 shows the relationship between the measured velocity vector, $\vec{V}_L$, and the modeled structural velocity, $\vec{V}_S$, vector in the x-y plane.

Figure 2.3 Relationship between measured velocity, $V_L$, and modeled velocity, $V_S$, in the XY plane
Substituting Equations 2.7 and 2.8 into Equation 2.9, the magnitude of the measured velocity becomes

\[ |\vec{V}_L| = \hat{u}\cos(\theta_x) + \hat{v}\cos(\theta_y) + \hat{w}\cos(\theta_z) \]

(2.10)

Defining direction cosines \( \Psi_X, \Psi_Y, \) and \( \Psi_Z \) as follow

\[ \Psi_X = \cos(\theta_x) \]
\[ \Psi_Y = \cos(\theta_y) \]
\[ \Psi_Z = \cos(\theta_z) \]

(2.11)

Then the magnitude of the measured velocity becomes

\[ |\vec{V}_L| = \hat{u}\Psi_X + \hat{v}\Psi_Y + \hat{w}\Psi_Z \]

(2.12)

For the harmonic loading with forcing frequency, \( \omega \), the components of the modeled complex velocities are given

\[ \hat{u} = \bar{u}e^{i(\omega t + \phi)} \]
\[ \hat{v} = \bar{v}e^{i(\omega t + \phi)} \]
\[ \hat{w} = \bar{w}e^{i(\omega t + \phi)} \]

(2.13)

Where \( \bar{u}, \bar{v}, \) and \( \bar{w} \) are the magnitudes of the modeled structural velocity, and \( \phi \) is the relative phase angle of the modeled velocity on the surface of the oscillating structure, \( \vec{V}_S \), relative to the forcing function, \( F(t) = \tilde{f} e^{i\omega t} \).

Substituting Equation 2.13 into Equation 2.12, the magnitude of the measured velocity can be written

\[ |\bar{V}_L| = \bar{u}e^{i(\omega t + \phi)}\Psi_X + \bar{v}e^{i(\omega t + \phi)}\Psi_Y + \bar{w}e^{i(\omega t + \phi)}\Psi_Z \]

(2.14)

The function \( e^{i\omega t} \) is relative to the trigonometric function \( \cos \omega t \) and \( \sin \omega t \) and can be expressed as
Using Equation 2.15, Equation 2.13 can be rewritten as
\[
\begin{align*}
\ddot{u} &= \ddot{u}e^{i(\omega t + \phi)} = \ddot{u}e^{i\omega t}e^{i\phi} = \ddot{u}(\cos \phi + i \sin \phi)e^{i\omega t} \\
\ddot{v} &= \ddot{v}e^{i(\omega t + \phi)} = \ddot{v}e^{i\omega t}e^{i\phi} = \ddot{v}(\cos \phi + i \sin \phi)e^{i\omega t} \\
\ddot{w} &= \ddot{w}e^{i(\omega t + \phi)} = \ddot{w}e^{i\omega t}e^{i\phi} = \ddot{w}(\cos \phi + i \sin \phi)e^{i\omega t}
\end{align*}
\] (2.16)

Substituting Equation 2.16 into Equation 2.14, the magnitude of the measured velocity can be written as
\[
\begin{align*}
\left| \ddot{V}_L \right| &= \ddot{u}(\cos \phi + i \sin \phi)e^{i\omega t}\Psi_x + \ddot{v}(\cos \phi + i \sin \phi)e^{i\omega t}\Psi_y + \ddot{w}(\cos \phi + i \sin \phi)e^{i\omega t}\Psi_z
\end{align*}
\] (2.17)

Regrouping components of Equation 2.17 by real and imaginary components, Equation 2.17 can be expressed as
\[
\begin{align*}
\left| \ddot{V}_L \right| &= \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\cos \phi + i \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\sin \phi e^{i\omega t}
\end{align*}
\] (2.18)

However, when the components of the modeled complex velocities are obtained from the FE models, the harmonic function $e^{i\omega t}$ in Equation 2.16 will be canceled out with $e^{i\omega t}$ function in the harmonic forcing function, $F(t) = \tilde{f} e^{i\omega t}$. Therefore, $e^{i\omega t}$ function is removed from Equation 2.16 and Equation 2.18 is rewritten as
\[
\begin{align*}
\left| \ddot{V}_L \right| &= \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\cos \phi + i \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\sin \phi
\end{align*}
\] (2.19)

The Real and Imaginary components of Equation 2.19 are defined as
\[
\begin{align*}
\text{Real component: } \text{Re} &= \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\cos \phi \\
\text{Imaginary component: } \text{Imag} &= \left( \ddot{u}\Psi_x + \ddot{v}\Psi_y + \ddot{w}\Psi_z \right)\sin \phi
\end{align*}
\] (2.20)

Therefore the magnitude of measured velocity at instant time, $t$, is written as
\[
\left| \dot{V}_L \right| = \sqrt{\text{Re}^2 + \text{Imag}^2}
\] (2.21)
and the relative phase angle of the measured velocity at instant, \( t \), relative to the reference forcing function is written as

\[
\phi_{VL} = \tan^{-1} \frac{\text{Imag}}{\text{Re}}
\]

(2.22)

The SLDV will measure the result of the Equation 2.21 in instant time, \( t \). Figure 2.4 shows Equation 2.21 as a function of time, \( t \). Figure 2.4 is for the ideal condition such that no noise is present in the data. As shown in Figure 2.4 the magnitude of the measured velocity is a harmonic function with the relative phase angle, \( \phi_{VL} \), with respect to the forcing function.

**Figure 2.4 Magnitude of measure velocity and forcing signal**

(No noise and scale of the magnitude both force and velocity function are assumed)
2.4 Error function of ESDM

To evaluate the dynamic response on the surface of the structure obtained by the FE model, the modeled velocity on the surface of the structure and directional cosines of the laser beam will be used to simulate the magnitude of the measured velocity of the structure. As discussed earlier, the magnitude of the measured velocity of the structure can be constructed from the dot product of the modeled velocity on the surface of the structure and directional cosines of the laser beam (Equation 2.9). Using the modeled velocity on the surface of the structure and the magnitude of the measured velocity of the structure, the error function of ESDM is derived. The error function of ESDM can be written as

\[
\begin{align*}
\text{Error}_{\text{Real}} &= \mathbf{V}_{\text{L Real}} \cdot \mathbf{V}_{\text{S Real}} - \mathbf{V}_{\text{L Measured}} \\
&= (\mathbf{V}_U \cdot \mathbf{V}_S)_{\text{Real}} - |\mathbf{V}_L|_{\text{Real}} \\
&= (\mathbf{V}_X + \mathbf{V}_Y + \mathbf{V}_Z) \cos \phi - |\mathbf{V}_L|_{\text{Real}} \\
\text{Error}_{\text{Imag}} &= \mathbf{V}_{\text{L Imag}} \cdot \mathbf{V}_{\text{S Imag}} - \mathbf{V}_{\text{L Measured}} \\
&= (\mathbf{V}_U \cdot \mathbf{V}_S)_{\text{Imag}} - |\mathbf{V}_L|_{\text{Imag}} \\
&= (\mathbf{V}_X + \mathbf{V}_Y + \mathbf{V}_Z) \sin \phi - |\mathbf{V}_L|_{\text{Imag}}
\end{align*}
\]

(2.23)

Where \( |\mathbf{V}_L| \) is the magnitude of the measured velocity

\( \mathbf{V}_U \) is unit vector in the direction of the laser beam ’s line-of-sight

\( \mathbf{V}_S \) is structural velocity on the surface of FE model

Assuming the measured magnitude of velocity data also includes random noise, this noise factor, \( \varepsilon \), needs to be included in Equation. 2.23. Therefore Equation 2.23 can be rewritten as
\[
\text{Error}_{\text{Real}} = \mathbf{V}_{L,\text{Real-Predicted}} - \mathbf{V}_{L,\text{Real-Measured}} + \epsilon \\
= (\mathbf{V}_U \cdot \mathbf{V}_S)_{\text{Real}} - |\mathbf{V}_L|_{\text{Real}} + \epsilon \\
= (\mathbf{u}\Psi_x + \mathbf{v}\Psi_y + \mathbf{w}\Psi_z) \cos \phi - |\mathbf{V}_L|_{\text{Real}} + \epsilon \\
\text{Error}_{\text{Imag}} = \mathbf{V}_{L,\text{Imag-Predicted}} - \mathbf{V}_{L,\text{Imag-Measured}} + \epsilon \\
= (\mathbf{V}_U \cdot \mathbf{V}_S)_{\text{Imag}} - |\mathbf{V}_L|_{\text{Imag}} + \epsilon \\
= (\mathbf{u}\Psi_x + \mathbf{v}\Psi_y + \mathbf{w}\Psi_z) \sin \phi - |\mathbf{V}_L|_{\text{Imag}} + \epsilon
\]

(2.24)

For this research, the structural velocity on the surface \( \mathbf{V}_S \) will be generated by the FE model, which is constructed by the flat shell element, and the velocity transformation function. Using the given location and orientation of the laser and unit vector in the direction of the laser beam’s line-of-sight, \( \mathbf{V}_U \), will be obtained. The predicted magnitude of measured velocity of the structure is given by \( \mathbf{V}_S \cdot \mathbf{V}_U \). The application of the velocity transformation function will be demonstrated.
Chapter 3
Dynamic and Vibration Analysis of Structures Using Finite Element Method

Candidate finite element models, which are constructed by the flat shell element, will be used to construct the dynamic response models of vibrating structures and to simulate experimental data. The simulated experimental data is the magnitudes of structural velocities measured by a scanning laser doppler vibrometer (SLDV). In this chapter, the structural dynamics and vibration problems will be discussed using the Finite Element method (FE method). The flat shell element will be developed. Lastly, frequency response analysis and the extraction of natural frequencies and mode shapes will be discussed to help in understanding the dynamic response of structures.

3.1 Finite element method in vibration and dynamic problems

In this research the dynamic response models of structures, used in the ESDM formulation, will be constructed by the finite element (FE) models of structures and velocity transformation function. The dynamic response model of a vibrating structure is the complex valued continuous three dimensional velocity field on the surface of the structure. To make proper FE models of structures, it is necessary to develop a candidate finite element. Based on properties of the candidate finite element, the velocity transformation function of the FE model will be developed. To develop the candidate finite element models and velocity transformation function, it is important to understand how the FE method is used in problems of structural dynamics and vibrations. In this section the equation of motion (the governing equation) of the FE model is discussed for vibration and dynamic analysis of structures. The general formulation of mass, stiffness, damping matrices that are used in the equation of motion are developed. The physical meaning of these matrices is explained.
### 3.1.1 Equation of motion (Single Degree of Freedom System)

Using Newton’s second law of motion (Equation 3.1), the equation of motion of the single degree of freedom (SDOF) system, shown in Figure 3.1, can be written as Equation 3.2

\[ \sum F = M \cdot a = M \cdot \ddot{x} \]
\[ (3.1) \]

\[ M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = F(t) \]
\[ (3.2) \]

where \( \ddot{x}(t) = \) time dependent acceleration  ;  \( M = \) Equivalent mass
\( \dot{x}(t) = \) time dependent velocity  ;  \( C = \) Characteristic damping coefficient
\( x(t) = \) time dependent displacement  ;  \( K = \) Characteristic stiffness coefficient

![Figure 3.1 Single degree of freedom system](image)

As shown in Equation 3.2, the equation of motion of the SDOF system is a second order linear differential equation with constant coefficients, \( M, C, \) and \( K \). For this research, structures will be excited by a harmonic forcing function, shown in Equation 3.3.

\[ F(t) = \tilde{f} e^{i\omega t} \]
\[ (3.3) \]

where  \( \tilde{f} = \) amplitude of force
\( \omega = \) circular forcing frequency
Since the SDOF system is represented as a linear system with the harmonic forcing function, shown in Equation 3.3, the velocity and acceleration of the system can be obtained from the derivative of the displacement, $x(t)$, with respect to time. Assuming the displacement of the system is $\tilde{x}e^{i(\omega t + \phi)}$, where $\tilde{x}$ is the amplitude of displacement and $\phi$ is the relative phase angle of the displacement respect to the harmonic force, the velocity and acceleration can be written as

\[
\text{Displacement} = \tilde{x}e^{i(\omega t + \phi)} = \tilde{x}e^{i\omega t}e^{i\phi} = \hat{x}e^{i\omega t}
\]

(3.4)

\[
\text{Velocity } \dot{x} = \frac{\partial x}{\partial t} = i\omega \hat{x} e^{i\omega t}
\]

(3.5)

\[
\text{Acceleration } \ddot{x} = \frac{\partial \dot{x}}{\partial t} = -\omega^2 \hat{x} e^{i\omega t}
\]

(3.6)

Where $\hat{x} = \tilde{x}e^{i\phi} = \tilde{x}(\cos(\phi) + i\sin(\phi)) = \tilde{x}\cos(\phi) + i\tilde{x}\sin(\phi) = \hat{x}_{\text{Real}} + \hat{x}_{\text{Imag}}$

(3.7)

Figure 3.2 shows the vector relationship for forced vibration of the SDOF system with damping in the complex plain.

![Figure 3.2 Vector relationship for forced vibration of the SDOF system with damping](image)
Substituting Equation 3.4, 3.5 and 3.6 into Equation 3.2 and setting the characteristic damping coefficient is zero, the equation of motion for the undamped SDOF system can be written as

\[
- \omega^2 M \dot{x} e^{i \omega t} + K x e^{i \omega t} = \ddot{f} e^{i \omega t} \\
\left( - \omega^2 M + K \right) \ddot{x} = \ddot{f} \\
\left( - \omega^2 M + K \right) \dot{x} = \ddot{f}
\]

(3.8)

The equation of motion for damped SDOF system can be written as

\[
- \omega^2 M \dot{x} e^{i \omega t} + i \omega C e^{i \omega t} + K \dot{x} e^{i \omega t} = \ddot{f} e^{i \omega t} \\
\left( - \omega^2 M + i \omega C + K \right) \dot{x} = \ddot{f} \\
\left( - \omega^2 M + i \omega C + K \right) \ddot{x} = \ddot{f}
\]

(3.9)

For many structural metals such as steel or aluminum, the energy dissipated per cycle is independent of frequency over a wide frequency range and proportional to the square of the amplitude of vibration. Internal damping fitting this classification is called solid damping or structural damping (Thomson, 1998). For this research, assuming harmonic cyclic loading is applied to structures, and the material of structures has the property described above, the damping of structures is represented by structural damping. Using the structural damping factor and the complex stiffness matrix, the damping and stiffness term in Equation 3.2 can be expressed as

\[
C \ddot{x} + K x = (K + i \omega C) \dot{x} e^{i \omega t} = (K + i \gamma) \dot{x} e^{i \omega t} = K (1 + i \gamma) \dot{x} e^{i \omega t}
\]

(3.10)

Where \( C = K \frac{\alpha}{\pi \omega} \) and \( \gamma = \frac{\alpha}{\pi} \). \( \alpha \) is a constant with units of force/displacement. The structural damping factor is referred to as \( \gamma \). The structural damping factor, \( \gamma \), is the energy lost per cycle divided by \( 2\pi \) times the maximum potential energy. The quantity \( K (1 + i \gamma) \) is called the complex stiffness. The equation of motion for the SDOF system with structural damping (Equation 3.9) can be rewritten

\[
\left( - \omega^2 M + (1 + i \gamma) K \right) \ddot{x} = \ddot{f}
\]

(3.11)
3.1.2 Equation of motion (Multiple Degree of Freedom System)

When a system requires more than one coordinate to describe its motions, it is called a multiple degree of freedom (MDOF) system. Using Equation 3.2 and matrix notation, the equation of motion of the MDOF system can be written. For example, the two degree of freedom system, shown in Figure 3.3, has the following differential equations of motion.

\[
M_1 \ddot{x}_1(t) + (C_1 + C_2) \dot{x}_1(t) - C_2 \dot{x}_2(t) + (K_1 + K_2)x_1(t) - K_2x_2(t) = 0 \\
M_2 \ddot{x}_2(t) - C_2 \dot{x}_1(t) + C_2 \dot{x}_2(t) - K_2x_1(t) + K_2x_2(t) = F_2(t)
\]

(3.12)

Using matrix notation, Equation 3.12 becomes the differential equation in matrix form as Equation 3.13

\[
\begin{bmatrix}
M_1 & 0 \\
0 & M_2
\end{bmatrix}
\begin{bmatrix}
\ddot{x}_1(t) \\
\ddot{x}_2(t)
\end{bmatrix} +
\begin{bmatrix}
C_1 + C_2 & -C_2 \\
-C_2 & C_2
\end{bmatrix}
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t)
\end{bmatrix} +
\begin{bmatrix}
K_1 + K_2 & -K_2 \\
-K_2 & K_2
\end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix} =
\begin{bmatrix}
0 \\
F_2(t)
\end{bmatrix}
\]

(3.13)

Figure 3.3: Two degrees of freedom mass, spring, and damper system

In general, the MDOF system has the following equation of motion.

\[
[M]\ddot{x}(t) + [C]\dot{x}(t) + [K]x(t) = \{F(t)\}
\]
Where \([M]\) is the mass \(n \times n\) matrix
\([C]\) is the damping \(n \times n\) matrix
\([K]\) is the stiffness \(n \times n\) matrix
\(\{F(t)\}\) is forcing vector
\(\{\ddot{x}(t)\}\) is acceleration vector
\(\{\dot{x}(t)\}\) is velocity vector
\(\{x(t)\}\) is displacement vector

\[(3.14)\]

Like the SDOF system, the dynamic response of the MDOF system due to the harmonic forcing function, Equation 3.3, also can be expressed as a complex function in terms of forcing frequency and the amplitude of the dynamic response of the structure. Table 3.1 summarizes the forcing and dynamic response terms of an undamped and damped structure.

**Table 3.1 Forcing function and dynamic response of structure**

<table>
<thead>
<tr>
<th></th>
<th>Undamped</th>
<th>Damped</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F(t))</td>
<td>(f e^{i\omega t})</td>
<td>(f e^{i\omega t})</td>
</tr>
<tr>
<td>(x(t))</td>
<td>(\bar{x} e^{i\omega t})</td>
<td>(\hat{x} e^{i\omega t})</td>
</tr>
<tr>
<td>(\dot{x}(t))</td>
<td>(\omega i \bar{x} e^{i\omega t})</td>
<td>(\omega i \hat{x} e^{i\omega t})</td>
</tr>
<tr>
<td>(\ddot{x}(t))</td>
<td>(-\omega^2 i \bar{x} e^{i\omega t})</td>
<td>(-\omega^2 i \hat{x} e^{i\omega t})</td>
</tr>
</tbody>
</table>

\[\hat{x} = \bar{x} \cos(\phi) + i\bar{x} \sin(\phi)\]

\(\bar{x}\) = amplitude of the displacement
\(\phi\) = the relative phase angle of the displacement respect to the reference force

Assuming the damping matrix, \([C]\), is zero and substituting the harmonic forcing function and undamped dynamic response into Equation 3.14, the equation of motion of the undamped MDOF system can be written as
Substituting the harmonic forcing function and damped dynamic responses, the equation of the damped MDOF system is written as

\[- \omega^2 [M] \{ \ddot{x} \} e^{i\omega t} + [C] \{ \dot{x} \} e^{i\omega t} + [K] \{ x \} e^{i\omega t} = \{ f \} e^{i\omega t}, \]

\[- \omega^2 [M] + [C] \{ \dot{x} \} e^{i\omega t} = \{ f \} e^{i\omega t}, \]

\[- \omega^2 [M] + [K] \{ x \} e^{i\omega t} = \{ f \} \]

(3.15)

Since structural damping is used to represent the damping of the MDOF system, the equation of motion for the MDOF system with structural damping matrix can be written as

\[- \omega^2 [M] \{ \ddot{x} \} e^{i\omega t} + i\omega [C] \{ \dot{x} \} e^{i\omega t} + [K] \{ x \} e^{i\omega t} = \{ f \} e^{i\omega t}, \]

\[- \omega^2 [M] + i\omega [C] + [K] \{ \dot{x} \} e^{i\omega t} = \{ f \} e^{i\omega t}, \]

\[- \omega^2 [M] + i\omega [C] + [K] \{ x \} = \{ f \} \]

(3.16)

where \( (1 + i\gamma) [K] \) is the complex stiffness matrix

3.1.3 Displacement and shape Function

When using the FE method in structural dynamics and vibration problems, the solution for the unknown displacement is obtained at every finite element node (Charalambides, 1996). The displacement within each element is then approximated by shape (interpolation) functions and the nodal solution. Linear, quadratic, and higher order shape functions are used in the finite element approximation. The shape function is dependent on the element chosen for use in the finite element model. For example, the typical rod element, which has linear axial displacement only, uses linear shape functions and the typical beam element uses cubic shape functions. In solid mechanics, the displacement vector \( \{ u \} \) within an element is obtained as the product of the
shape function matrix $[N]$ and the nodal displacement vector $\{d\}$ as follows (Charalambides, 1996).

$$\{u\} = [N]^T \{d\}$$

(3.18)

The approximated displacement vector, $\{u\}$, is called the displacement field of the structure. The dimension of the shape function matrix, $[N]$, and displacement field, $\{u\}$, and nodal displacement vector, $\{d\}$, depend on whether the problem involves 1-D, 2-D, or 3-D solids and on the type of the element used. Figure 3.4 shows a rod element, which has only a linear axial displacement field. The length of the rod element is $L$. The displacement field within the rod element can be written in terms of nodal displacements and shape functions (Cook, Malkusm and Plesha 1989).

$$u(x) = N_1(x)d_1 + N_2(x)d_2$$

(3.19)

Where shape functions are defined as

$$[N] = \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} = \begin{bmatrix} \frac{L-x}{L} \\ \frac{x}{L} \end{bmatrix} \quad \text{with } 0 \leq x \leq L$$

(3.20)

$$u(x) = [N]^T \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

(3.21)

![Figure 3.4 Rod element](image.png)
The candidate FE models of test structures give the nodal displacement and velocity field. In this research, however, displacement and velocity field of the candidate FE model need to be constructed from the laser measurement locations not from the node locations. The shape function of candidate FE model will interpolate the displacement and velocity field of the candidate FE model based on the laser measurement locations.

3.1.4 The Mass Matrix

In Equation 3.14, the n x n matrix, \([M]\), is called the mass matrix. The mass matrix \([M]\) describes the coefficient of the inertial forces due to mass acceleration of structure. The simplest mass matrix is called a “diagonal” or “lumped” mass matrix. This mass matrix can be obtained by distributing the total element mass to each node of the element evenly. For example, if the rod element, shown in Figure 3.4, has cross-sectional area \(A\), length \(L\), and mass density \(\rho\), the total element mass is \(\rho A L\). The lumped mass matrix \([M]\) for the rod element can be written as

\[
[M]_{\text{lump}} = \begin{bmatrix} \frac{\rho A L}{2} & 0 \\ 0 & \frac{\rho A L}{2} \end{bmatrix} = \frac{\rho A L}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

(3.22)

An additional mass matrix can be developed by using inertia forces in a virtual work argument. This general result for this element mass matrix can be derived as follows

\[
[M]_{\text{consistent}} = \int_V [N]^T [N] \rho dV
\]

Where \(\rho\) is mass density

\(V\) is element volume

\([N]\) is element shape function matrix

(3.23)
This element mass matrix \([M]_{\text{consistent}}\) is called the consistent mass matrix. For example, the consistent mass of the rod element, shown in Figure 3.4, can be expressed as

\[
[M]_{\text{consistent}} = \begin{bmatrix}
\frac{\rho A L}{3} & \frac{\rho A L}{6} \\
\frac{\rho A L}{6} & \frac{\rho A L}{3}
\end{bmatrix} = \rho A L \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}
\]

(3.24)

The lumped mass matrix and the consistent mass matrix differ in how the angular acceleration is contributed to the inertial force. Whether a mass matrix of FE model is the lumped mass matrix or the consistent mass matrix, proper convergence with mesh refinement is assured if element mass matrices provide the correct inertia force in response to all possible translational accelerations of the element (Cook, 1994). A benefit of the consistent mass, however, is that computed natural frequencies of the FE model are guaranteed to be upper bound on natural frequencies of the mathematical modal. In other words, natural frequencies of the FE model guaranteed to converge from above with mesh refinement. For this research, a consistent mass matrix will be used.

3.1.5 The Stiffness Matrix

In Equation 3.14, the \(n \times n\) matrix, \([K]\), is called the stiffness matrix. General formulas of the stiffness matrix are obtained from the virtual work principle. The virtual work principle states that if a structure which is in equilibrium with its applied forces is subjected to a set of small compatible virtual displacements, the virtual work done by the applied forces is equal to the virtual work done by the internal forces. The virtual work done by the internal forces is the virtual strain energy from the internal stresses. Equation 3.25 shows the virtual work principle in equation form.

\[
\delta U_e = \delta W_e
\]

(3.25)

Where \(\delta U_e\) is the virtual strain energy from internal stresses, and \(\delta W_e\) is the virtual work from the external forces. While the virtual displacements, \(\{\delta d\}\), are small disturbances of the actual node
displacement, \( \{d\} \), produced by the applied loads when the element reaches equilibrium, the
virtual work from the external forces is defined as

\[
\delta W_e = \{\delta d\}^T \{f\}
\]

(3.26)

Where \( \{f\} \) are applied nodal forces. The virtual strain energy from internal stresses is defined as
the integral over the element volume of the sum of the products of each corresponding stress and
virtual strain component. Equation 3.27 shows the virtual strain energy.

\[
\delta U_e = \int \{\delta \varepsilon\}^T \{\sigma\} \, dV
\]

(3.27)

Where \( \{\delta \varepsilon\} \) are the virtual strain components by the virtual displacement, \( \{\delta d\} \), \( \{\sigma\} \) are the
stress components, and \( dV \) indicates the differential volume of the element.

Using the strain displacement relation, the virtual strains inside an element can be defined
by the displacement gradient and virtual displacements.

\[
\{\delta \varepsilon\} = [\partial][N][\delta d] = [B][\delta d], \ [B] = [\partial][N]
\]

(3.28)

Where \( [\partial] \) is the derivative operator matrix, \( [N] \) is the shape function matrix, \( \{\delta d\} \) is the virtual
displacement vector, and \( [B] \) is the strain-displacement matrix.

The stresses inside an element are found from the stress-strain relationship. The stresses can be
written as

\[
\{\sigma\} = [E][\varepsilon] = [E][B][d]
\]

(3.29)

Where the matrix, \( [E] \), is the material property matrix. Substituting Equation 3.28 and 3.29 into
Equation 3.27, the virtual strain energy is written as

\[
\delta U_e = \int \{\delta \varepsilon\}^T \{\sigma\} \, dV = \int \{\delta d\}^T [B]^T [E][B] \, dV \{d\}
\]

(3.30)

Substituting Equation 3.26 and 3.30 into Equation 3.25, the principle of virtual work can be
written as
\[
\left\{ \{ \delta d \} \right\}^T \left[ \frac{\partial}{\partial x} \right] \left[ \begin{bmatrix} B \end{bmatrix} \right] dV \{ d \} = \left\{ \{ \delta d \} \right\}^T \{ f \}
\] 

(3.31)

Since both the virtual and real displacement are independent of integral over the element volume, Equation 3.31 can be rewritten as
\[
\left\{ \{ \delta d \} \right\}^T \left[ \{ \delta d \} \right] dV \{ d \} = \left\{ \{ \delta d \} \right\}^T \{ f \}
\]

(3.32)

Equation 3.32 can be expressed as
\[
[K] [d] = \{ f \}
\]

(3.33)

Where the general formula for the stiffness matrix, \([K]\), is written as
\[
[K] = \left[ \left[ \frac{\partial}{\partial x} \right] \left[ \begin{bmatrix} B \end{bmatrix} \right] \right] dV
\]

(3.34)

For example, the rod element, shown in Figure 3.1, the element stiffness matrix, \([K]\), is defined in terms of the strain-displacement and stress-strain relationship of the element. The length of the rod element is \(L\) and the cross sectional area is \(A\). Using the shape function matrix (Equation 3.20), the strain-displacement matrix, \([B]\), for the rod element is obtained
\[
[B] = \frac{\partial [N]}{\partial x} = \left[ \begin{array}{c} \frac{1}{L} \\ 1 \\ \frac{1}{L} \end{array} \right]
\]

(3.35)

Substituting Equation 3.35 into Equation 3.34, the stiffness matrix for the rod element can be found as
\[
[K] = \int_0^L \left[ \begin{array}{c} \frac{1}{L} \\ 1 \\ \frac{1}{L} \end{array} \right] \left[ \begin{array}{c} \frac{1}{L} \\ 1 \\ \frac{1}{L} \end{array} \right] dx = \frac{AE}{L} \left[ \begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array} \right]
\]

(3.36)

In this research, the stiffness matrix of a candidate finite element will be constructed based on the general formula of the stiffness matrix (Equation 3.34).
3.1.6 The Damping Matrix

The damping of structures is represented by structural damping. Using the complex stiffness matrix, \((1 + i\gamma)[K]\), the damping of the structure is introduced in the equation of motion for the MDOF system which is defined as \((-\omega^2[M] + (1 + i\gamma)[K])\{\ddot{x}\} = \{\ddot{f}\}.\) The forcing function vector, \(\{\ddot{f}\}\), and dynamic response of a structure, \(\{\ddot{x}\} = \{\ddot{x}(\cos\phi + i\sin\phi)\}\), are defined in terms of complex quantities. For the MDOF system, which has \(n\) degrees of freedom, the real and complex components of the forcing function and dynamic response of a structure can be separated into the real and imaginary coefficients by substituting the complex stiffness matrix into Equation 3.14. Equation 3.14 can be partitioned in terms of real and imaginary components and written as

\[
\begin{bmatrix}
K - \omega^2[M] & -\gamma[K] \\
\gamma[K] & K - \omega^2[M]
\end{bmatrix}
\begin{bmatrix}
\{\ddot{x}_{\text{real}}\}_{j=1..n} \\
\{\ddot{x}_{\text{imaginary}}\}_{j=1..n}
\end{bmatrix}
=
\begin{bmatrix}
\{\ddot{f}_{\text{real}}\}_{j=1..n} \\
\{\ddot{f}_{\text{imaginary}}\}_{j=1..n}
\end{bmatrix}
\]

(3.37)

where dynamic stiffness matrix, \([D]\), is defined as

\[
\begin{bmatrix}
K - \omega^2[M] & -\gamma[K] \\
\gamma[K] & K - \omega^2[M]
\end{bmatrix}
\]

(3.38)

Equation 3.28 forms a system in which all coefficients are real and the dynamic response of the structure can be directly determined by pre multiplying inverse dynamic stiffness matrix \([D]\) with the force vector.

\[
\begin{bmatrix}
\{\ddot{x}_{\text{real}}\}_{j=1..n} \\
\{\ddot{x}_{\text{imaginary}}\}_{j=1..n}
\end{bmatrix}
= [D]^{-1}
\begin{bmatrix}
\{\ddot{f}_{\text{real}}\}_{j=1..n} \\
\{\ddot{f}_{\text{imaginary}}\}_{j=1..n}
\end{bmatrix}
\]

(3.39)

The results of Equation 3.39, then, can be interpreted as

\[
\begin{align*}
\hat{x}_{\text{real},j} &= \bar{x}_j \cos \phi_j \\
\hat{x}_{\text{imaginary},j} &= \bar{x}_j \sin \phi_j
\end{align*}
\]

(3.40)
where $\bar{x}_j$ is $\sqrt{x_{real j}^2 + x_{imaginary j}^2}$ and $\phi_j$ is $\tan^{-1}\frac{x_{imaginary j}}{x_{real j}}$, $j = 1, 2, \ldots, n$

(3.41)

For this research, general Equation 3.37 will be used for the governing equation of the finite element solution for direct frequency response. In summary, the general formulation of mass, stiffness, and damping matrices are developed and the physical meaning of these matrices is explained. The governing equation are developed for structures which are excited by the harmonic forcing function. In the next section the candidate finite element is developed based on the general formation of mass, stiffness, and damping matrices.
3.2 Flat shell element

For this research, flat-shell elements, shown in Figure 3.5, are used to build a FE model of structures. As shown in Figure 3.5, this element has four nodes (i=1,2,3,and 4) and five degrees of freedom per node \((u_i, v_i, w_i, \theta_{xi}, \theta_{yi})\) \(i=1,2,3,4\). This element consists of the components of the bi-linear plane stress element (Cook, Malkus, and Plesha 1989) and the Kirchoff plate element (Haug, Choi, and Komkov 1986). The mass and stiffness matrices of the flat shell element are constructed by matrix superposition of the mass and stiffness matrices of bi-linear plane stress and Kirchoff plate element. Some of the characteristic quantities of the element are as follows.

1. Sides of the element remain straight when the element deforms in-plane.
2. Thin-plate theory is used to develop the flat shell element. Thin-plate theory neglects transverse shear deformation, which can be significant if the plate is thick, that is, if thickness is more than roughly one-tenth the span of the plate (Cook, 1995). Therefore, thickness of the element is small enough so that transverse shear deformation is neglected.

The stiffness matrices of the bi-linear plane stress element and Kirchoff plate element will be discussed separately.
Figure 3.5: Flat-shell element with five degree of freedom per node
3.2.1 Bi-linear plane stress element

Axial (x-axis) and lateral (y-axis) nodal displacement of a flat-shell element will be obtained from a bi-linear element. The displacement field of a bi-linear plane stress element can be written in terms of the nodal displacements and shape functions. The shape functions and displacement field of the bi-linear plane stress element are defined in the normalized local coordinate, \( \xi \) and \( \eta \) (Figure 3.5). Assuming the bi-linear planes stress element is the rectangular shape element, the normalized local coordinates, \( \xi \) and \( \eta \), are defined as

\[
\xi = \frac{x}{\alpha}, \quad \eta = \frac{y}{\beta} \quad \text{with} \quad 0 \leq \xi, \eta \leq 1
\]

(3.42)

Where \( \alpha \) and \( \beta \) are the length and width of the element along with the \( x \)- and \( y \)-axis respectively.

The displacement field of the bi-linear stress element is written as

\[
\begin{bmatrix}
  u(x, y) \\
  v(x, y)
\end{bmatrix}
= \begin{bmatrix}
  u_1 \\
  v_1 \\
  \vdots \\
  u_4 \\
  v_4
\end{bmatrix}
\begin{bmatrix}
  N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\
  0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4
\end{bmatrix}
\]

(3.43)

Shape functions of the bi-linear plane stress element in terms of the normalized local coordinate, \( \xi \) and \( \eta \), are given as follows.

\[
\begin{align*}
N_1 &= (\xi - 1)(\eta - 1) \\
N_2 &= (1 - \xi)\eta \\
N_3 &= \xi \eta \\
N_4 &= (1 - \eta)\xi
\end{align*}
\quad \text{with} \quad 0 \leq \xi, \eta \leq 1
\]

(3.44)
As explained earlier, the stiffness matrix for the bi-linear plane stress element is obtained from the potential energy stored in the element in terms of the strain-displacement and stress-strain relationships. Using a plane stress condition, in which \( \sigma_z = \tau_{yz} = \tau_{zx} = 0 \), the plane stress matrix, \([E]\), is found as

\[
\begin{bmatrix}
\sigma_x \\
\sigma_y \\
\tau_{xy}
\end{bmatrix} = [E]
\begin{bmatrix}
\varepsilon_x \\
\varepsilon_y \\
\gamma_{xy}
\end{bmatrix}
\]

(3.45)

\[
[E] = \frac{E}{1 - \nu^2}
\begin{bmatrix}
1 & \nu & 0 \\
\nu & 1 & 0 \\
0 & 0 & \frac{(1 - \nu)}{2}
\end{bmatrix}
\]

(3.46)

Where \( E \) is the elastic modulus and \( \nu \) is Poisson’s ratio.

The strain-displacement relationships for the plane stress condition are defined as

\[
\varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}
\]

(3.47)

and strain of the element can be expressed in matrix form as

\[
[\varepsilon] = [\partial][N]\{d\} = [B]\{d\} =
\begin{bmatrix}
\frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \cdots \\
0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \cdots \\
\frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \cdots
\end{bmatrix}\{d\}
\]

(3.48)
\[
[B] = \begin{bmatrix}
\frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \cdots \\
0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \cdots \\
\frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \cdots \\
\end{bmatrix}
\]

(3.49)

Where \([\partial]\) is the derivative operator matrix, \([N]\) is the shape function matrix, and \(\{d\}\) is the nodal displacement vector. Matrix \([B]\) is called the strain-displacement matrix.

Substituting the plane stress matrix, \([E]\), strain-displacement matrix, \([B]\), determinant of Jacobian matrix, \(|J|\), and element thickness, \(t\), into Equation 3.34, the stiffness matrix of bi-linear plain stress element can be written in normalized local coordinate, \(\xi\) and \(\eta\), as

\[
K = \int \left( [B]^T [E] [B]^T \right) dV = \int \left( [B]^T [E] [B]^T \right) tdx dy = \int \int \left( [B]^T [E] [B]^T \right) t |J| \, d\xi \, d\eta
\]

(3.50)

Evaluating Equation 3.50 for the bi-linear plane stress element, the following stiffness matrix is obtained.

\[
[K]_{\text{bi-linear plain stress}} = \frac{E \, t}{24 \alpha \beta (\nu^2 - 1)} \begin{bmatrix}
K_{i,i} & K_{i,i,1} \\
K_{i,i,1} & K_{i,i,1}
\end{bmatrix}
\]

(3.51)

where \(\alpha\) and \(\beta\) are the length and width of the element along with \(x\) and \(y\)-axis respectively.

Sub-matrices \(K_{i,i}\), and \(K_{i,i,1}\) of the stiffness matrix of the bi-linear plane stress element are the coefficients of the axial (x-axis) and lateral (y-axis) nodal displacement of stiffness matrix of the flat shell element. The size of the stiffness matrix of the bi-linear plane stress element is \(8 \times 8\) and the size of sub-matrix \(K_{i,i}\) and \(K_{i,i,1}\) is a \(4 \times 4\) matrix. Since the stiffness matrix of the flat shell element is a \(20 \times 20\) matrix, the size of the stiffness matrix of the bi-linear plane stress element is expanded to a \(20 \times 20\) matrix by expanding sub matrices \(K_{i,i}\) and \(K_{i,i,1}\) to a \(10 \times 10\) matrices. Sub matrices \(K_{i,i}\) and \(K_{i,i,1}\) are expanded by filling components of \(K_{i,i}\) and \(K_{i,i,1}\) into axial and lateral nodal displacement components of the flat shell element and filling zeroes into
the transverse and two angular displacements components of the flat shell element. This expanded the stiffness matrix bi-linear plane stress element will be added to the expanded stiffness matrix of the Kirchoff plate element by matrix summation.

Expanded sub matrices $K_{II}$, and $K_{II}$ are shown in Figure 3.6
$$K_{II,I} = \begin{bmatrix}
4(-\alpha^2 - 2\beta^2 + \alpha^2\nu) & -3\alpha\beta(\nu + 1) & 4(-2\alpha^2 - \beta^2 + \beta^2\nu) \\
0 & 0 & 0 & \text{Symmetric} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
4(\alpha^2 - \beta^2 - \alpha^2\nu) & -3\alpha\beta(3\nu - 1) & 0 & 0 & 4(-\alpha^2 - 2\beta^2 + \alpha^2\nu) \\
3\alpha\beta(3\nu - 1) & 2(4\alpha^2 - \beta^2 + \beta^2\nu) & 0 & 0 & 3\alpha\beta(\nu + 1) & 4(-2\alpha^2 - \beta^2 + \beta^2\nu) \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}$$

$$K_{III,I} = \begin{bmatrix}
2(\alpha^2 + 2\beta^2 - \alpha^2\nu) & 3\alpha\beta(\nu + 1) & 0 & 0 & 0 & 2(-\alpha^2 + 4\beta^2 + \alpha^2\nu) & -3\alpha\beta(3\nu - 1) & 0 & 0 & 0 \\
3\alpha\beta(\nu + 1) & 2(2\alpha^2 + \beta^2 - \beta^2\nu) & 0 & 0 & 0 & 3\alpha\beta(3\nu - 1) & 4(-\alpha^2 + \beta^2 - \beta^2\nu) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2(-\alpha^2 + 4\beta^2 + \alpha^2\nu) & 3\alpha\beta(3\nu - 1) & 0 & 0 & 0 & 2(\alpha^2 + 2\beta^2 - \alpha^2\nu) & -3\alpha\beta(\nu + 1) & 0 & 0 & 0 \\
3\alpha\beta(3\nu - 1) & 4(-\alpha^2 + \beta^2 - \beta^2\nu) & 0 & 0 & 0 & -3\alpha\beta(\nu + 1) & 2(2\alpha^2 + \beta^2 - \beta^2\nu) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}$$

Figure 3.6 Sub matrices of the bi-linear plane element
3.2.2 Kirchoff Plate element

The transverse bending displacement of the flat shell element will be obtained from the Kirchoff plate element. The displacement field of the Kirchoff plate element can be written in terms of the nodal displacements and Hermitian shape functions. Like the bi-linear plane stress element, the shape functions and displacement field of the Kirchoff plate are defined in the normalized local coordinate, $\xi$ and $\eta$ (Equation 3.42). The displacement field of the Kirchoff plate element can be written as follows

$$w(\xi, \eta) = [N(\xi, \eta)] \begin{bmatrix} w_1 \\ \theta_{x1} \\ \theta_{y1} \\ w_4 \\ \theta_{x4} \\ \theta_{y4} \end{bmatrix}$$

(3.52)

Hermitian shape functions can be written in terms of the normalized local coordinates, $\xi$ and $\eta$ for the Kirchoff plate element. Equation 3.53 shows the Hermitian shape functions of the Kirchoff plate element.
\[
[N]^T = \begin{bmatrix}
1 - \xi \eta - (3 - 2\xi)\xi^2(1 - \eta) - (1 - \xi)(3 - 2\eta)\eta^2 \\
(1 - \xi)\eta(1 - \eta)^2 \beta \\
-\xi(1 - \xi)^2(1 - \eta)\alpha \\
(1 - \xi)(3 - 2\eta)\eta^2 + \xi(1 - \xi)(1 - 2\xi)\eta \\
-\xi(1 - \xi)(1 - \eta)\eta^2 \beta \\
-\xi(1 - \xi)^2 \eta \alpha \\
(3 - 2\xi)\xi^2\eta - \xi \eta (1 - \eta)(1 - 2\eta) \\
-\xi(1 - \eta)\eta^2 \beta \\
(1 - \xi)\xi^2 \eta \alpha \\
(3 - 2\xi)\xi^2(1 - \eta) + \xi \eta(1 - \eta)(1 - 2\eta) \\
\xi \eta(1 - \eta)^2 \beta \\
(1 - \xi)\xi^2(1 - \eta)\alpha 
\end{bmatrix}
\]
with \(0 \leq \xi, \eta \leq 1\)

(3.53)

Since deformation of plates can be characterized by the curvature of the plate along the \(x\)-axis, and \(y\)-axis and the twist, the strain of the plate can be written as

\[
\{\varepsilon\} = \begin{bmatrix}
-\frac{\partial^2 w}{\partial x^2} \\
-\frac{\partial^2 w}{\partial y^2} \\
-2\frac{\partial^2 w}{\partial x \partial y}
\end{bmatrix}
\]

(3.54)

By Hook’s law for the plate, stresses in the plate, given by (Shames and Dym 1985), can be expressed as

\[
\{\sigma\} = \begin{bmatrix} M_X \\
M_Y \\
M_{XY} \end{bmatrix} = [E]_M \{\varepsilon\}, \text{ where } [E]_M = \frac{E t^3}{12(1 - v^2)} \begin{bmatrix} 1 & v & 0 \\
v & 1 & 0 \\
0 & 0 & 1 - v \end{bmatrix}
\]

(3.55)
Matrix $[E]_M$ is called the moment-curvature matrix.

By the matrix operator $\{\partial\}$ (Shames and Dym 1985), the length and width of the element along the $x$-axis and $y$-axis respectively, $\alpha$ and $\beta$, the curvature-displacement matrix $[B]$ is expressed in terms of the normalized local coordinates, $\xi$ and $\eta$.

$$\{\partial\} = \begin{bmatrix} -\frac{\partial^2}{\alpha^2 \partial \xi^2} \\ -\frac{\partial^2}{\beta^2 \partial \eta^2} \\ \frac{2}{\alpha \beta} \frac{\partial}{\partial \xi} \frac{\partial}{\partial \eta} \end{bmatrix}, \text{ and } [B] = [\partial][N]$$

(3.56)

Finally the stiffness matrix for the Kirchoff plane element in terms of local coordinate, $\xi$ and $\eta$, is as follow

$$[K] = \int_0^1 \int_0^1 [(B)^T [E]_M [B]) dV = \int_0^1 [(B)^T [E]_M [B]\alpha \beta d\xi d\eta$$

(3.57)

Evaluating Equation 3.57 for the Kirchoff plate element, the following stiffness matrix is obtained.

$$[K]_{\text{Kirchoff plate}} = \frac{E t^3}{12 \alpha \beta (1 - \nu^2)} \begin{bmatrix} K_{I, I} & K_{I, II} \\ K_{II, I} & K_{II, II} \end{bmatrix}$$

(3.58)

Sub matrices $K_{I, I}$, $K_{II, I}$, and $K_{II, II}$ of the stiffness matrix of the Kirchoff plate element are coefficient of the transverse and two angular displacement components of the stiffness matrix of the flat shell element. The size of the stiffness matrix of the Kirchoff plate element is $12 \times 12$ and the size of sub-matrix $K_{I, I}$, $K_{II, I}$, and $K_{II, II}$ is a $6 \times 6$ matrix. Since the stiffness matrix of flat shell element is a $20 \times 20$ matrix and, the size of stiffness matrix of the Kirchoff plate element is expanded to a $20 \times 20$ matrix by expanding sub matrices $K_{I, I}$, $K_{II, I}$, and $K_{II, II}$ to $10 \times 10$ matrices. Sub-matrices $K_{I, I}$, $K_{II, I}$, and $K_{II, II}$ are expanded by filling components of $K_{I, I}$, $K_{II, I}$, and $K_{II, II}$ into the transverse and two angular displacements component of the flat shell element and filling
zeroes into axial and lateral nodal displacement component of the stiffness matrix of the flat shell element. This expanded stiffness matrix of Kirchoff plate element will be added to the expanded stiffness matrix of bi-linear plane stress element by matrix summation. Expanded sub matrices, $K_{I,I}$, $K_{II,I}$, and $K_{II,II}$ with aspect ratio $\gamma = \frac{\beta}{\alpha}$ are shown in Figure 3.7.

The complete stiffness matrix of the flat shell element can be written by the matrix summation of the expanded stiffness matrix of the bi-linear plain stress element and the Kirchoff plate element.

\[
[K]_{\text{Flat shell}} = [K]_{\text{bi-linear plain stress}} + [K]_{\text{Kirchoff plate}}
\]

(3.59)

For the flat shell element, a consistent mass matrix is used. Equation 3.16 shows the general formula for the consistent mass matrix. If Equation 3.16 is written in terms of the normalized local coordinate, $\xi$ and $\eta$, the consistent mass matrix can be expressed as follows.

\[
[M] = \int \rho [N]^T [N] dV = \int \int \rho [N]^T [N] t | J | d\xi d\eta
\]

(3.60)

The consistent mass matrix of the flat shell element is obtained by the superposition of the consistent mass matrices of each bi-linear plain stress and Kirchoff plate element.

\[
[M]_{\text{Flat shell}} = [M]_{\text{bi-linear plain stress}} + [M]_{\text{Kirchoff plate}}
\]

(3.61)

Figures 3.8 and 3.9 show the consistent mass matrix of the bi-linear plain stress element and the Kirchoff plate element.
Figure 3.7 Sub matrices of the Kirchoff plate element
Figure 3.8 Consistent mass matrix for the bi-linear element
$M_{\text{Kirchoff}} = \frac{\alpha \cdot \beta \cdot \rho \cdot t}{176400}$

$$
\begin{pmatrix}
0 & 0 & 24178 & 0 & 3227\beta & 560\beta^2 \\
0 & 0 & -3227\alpha & -441\alpha\beta & 560\alpha^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 8582 & 1918\beta & -1393\alpha & 0 & 0 & 24178 \\
0 & 0 & -1918\beta & -420\beta^2 & 294\alpha\beta & 0 & 0 & -3227\beta & 560\beta^2 \\
0 & 0 & -1393\alpha & -294\alpha\beta & 280\alpha^2 & 0 & 0 & -3227\alpha & 444\alpha\beta & 560\alpha^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2758 & 812\beta & -812\alpha & 0 & 0 & 8582 & -1393\beta & -1918\alpha & 0 & 0 & 24178 \\
0 & 0 & -812\beta & -210\beta^2 & 196\alpha\beta & 0 & 0 & -1393\beta & 280\beta^2 & 294\alpha\beta & 0 & 0 & -3227\beta & 560\beta^2 \\
0 & 0 & -812\alpha & -196\alpha\beta & -210\alpha^2 & 0 & 0 & -1918\alpha & -294\alpha\beta & -420\alpha^2 & 0 & 0 & 3227\alpha & -444\alpha\beta & 560\alpha^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 8582 & 1393\beta & -1918\alpha & 0 & 0 & 2758 & -812\beta & -812\alpha & 0 & 0 & 8582 & -1918\beta & 1393\alpha & 0 & 0 & 24178 \\
0 & 0 & 1393\beta & 280\beta^2 & -294\alpha\beta & 0 & 0 & 812\beta & -210\beta^2 & -196\alpha\beta & 0 & 0 & 1918\beta & -420\beta^2 & 294\alpha\beta & 0 & 0 & 3227\beta & 560\beta^2 \\
0 & 0 & 1918\alpha & 294\alpha\beta & -420\alpha^2 & 0 & 0 & 812\alpha & -196\alpha\beta & -210\alpha^2 & 0 & 0 & 1393\alpha & -294\alpha\beta & 210\alpha^2 & 0 & 0 & 3227\alpha & 444\alpha\beta & 560\alpha^2 \\
\end{pmatrix}
$$

Figure 3.9 Consistent mass matrix for Kirchoff plate element
3.3 Frequency Response Function and the solution of normal modes of the structure

3.3.1 Frequency Response Function

Equation 3.9 is the equation of motion of a SDOF system. Solving Equation 3.9 for the response vector \( x(\omega) \) gives

\[
x(\omega) = \frac{F(\omega)}{(K - M\omega^2 + iC\omega)} = H(\omega)F(\omega)
\]

(3.62)

\[
H(\omega) = \frac{x(\omega)}{F(\omega)}
\]

(3.63)

Equation 3.62 is called the frequency response function (FRF) and it shows the relationship between the input, \( F(\omega) \), and the output, \( x(\omega) \). The FRF, \( H(\omega) \), is called the receptance when the output is a structural displacement and the input is an excitation force. However, the output of the FRF could be velocity or acceleration instead of displacement.

For velocity, \( v(\omega) = \dot{x}(\omega) = i \omega H(\omega)F(\omega) = \omega H(\omega)F(\omega)e^{i\phi} = Y(\omega)F(\omega) \)

(3.64)

and the relative phase angle is given by \( \theta = \phi + \pi/2 \), since \( i = e^{i\pi/2} \)

(3.65)

\( Y(\omega) \) is the FRF called the mobility. Equation (3.64) and (3.65) show that mobility is simply \( \omega \) times the magnitude of the receptance and a +90 degree phase shift.

Also acceleration is

\[
a(t) = \ddot{v}(\omega) = i\omega Y(\omega)F(\omega) = \omega Y(\omega)F(\omega)e^{i\phi} = -\omega^2 H(\omega)F(\omega) = A(\omega)F(\omega)
\]

(3.66)

and the phase angle is given by \( \Theta = \theta + \pi/2 = \phi + \pi \)

(3.67)
A(ω) is a FRF called the accelerance. Equation 3.53 and 3.54 show that accelerance is simply $ω^2$ times the magnitude of the receptance with a 180 degree phase shift. Table 3.1 shows the name and definition of the Frequency Response Functions.

**Table 3.2 Definition of the Frequency Response Function**

<table>
<thead>
<tr>
<th>Definition</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(ω)/F(ω) = H(ω)$</td>
<td>Receptance</td>
</tr>
<tr>
<td>$v(ω)/F(ω) = Y(ω) = iω H(ω)$</td>
<td>Mobility</td>
</tr>
<tr>
<td>$a(ω)/F(ω) = A(ω) = iω Y(ω)$</td>
<td>Accelerance</td>
</tr>
</tbody>
</table>

For this research, the SLDV will collect surface velocities of a structure. Therefore, the mobility FRF will be used for the dynamic response analysis of structures.

### 3.3.2 Natural Frequencies and Mode shapes

In vibration analysis, often the modal summation method is used to find the dynamic response of the structure instead of a direct frequency response analysis. The first step of the modal summation method is to find natural frequencies and mode shapes. The natural frequency is defined as the frequency of oscillation of free vibration. The mode shape is defined as the relative position of all nodes of structures at a given natural frequency. This research will use the mobility FRF to characterize the dynamic response of structures. Natural frequencies and mode shapes of the structure will also be discussed in this section.

For the undamped n degree of freedom system, the natural frequencies and mode shapes are found from free vibration, where the external excitation force is $F = 0$. Equation 3.68 shows the characteristic equation for the undamped MDOF system in free vibration.
\[
([K] - [M] \omega^2) \{x\} = 0
\]  

(3.68)

Equation 3.68 is called the characteristic equation. This equation for the MDOF of system has a trivial solution \( \{x\} = 0 \), and \( n \) nontrivial solutions. Nontrivial solutions are obtained by solving the characteristic equation, \( ([K] - [M] \omega^2) = \{0\} \), where each nontrivial solution consists of an eigenvalue and its associated eigenvector. The eigenvalue and eigenvector of \( n \)th nontrivial solution is the square value of natural frequency, \( \omega_n^2 \), and its associated mode shape. For undamped systems or proportionally damped systems, all eigenvalues and eigenvectors are real values so that the motion of the different coordinates is either in phase or 180 out of phase with each other. The smallest non zero natural frequency is called the fundamental frequency of vibration.

In this research, structural damping represents the damping of structures. As shown in Equation 3.17 the structural damping matrix \( [C] \) is proportional to \( [K] \). The system with proportional damping may be made to vibrate freely in a set of uncouples modes which resembles, in shape, the modes of the undamped system, with amplitude diminishing exponentially with time and uniformly over the of stationary nodal points or lines (Hurty and Robinstein, 1964). Therefore, the mode shape of the proportionally damped system is very much like the vibration of the undamped system except that the motion diminishes in amplitude until the system comes to rest.

### 3.4 Summary

In this chapter, using the stiffness and mass matrix of the bi-linear plane stress element and Kirchoff Plate element, the flat shell element is developed. This flat shell element is used to construct the FE model of the vibrating structures. Based on the properties of the flat shell element, the velocity transformation function is developed. The frequency response, natural frequencies and mode shapes were discussed to understand the dynamic response of the structures. In this research, the FE model of the structure will be used to calculate the frequency response, natural frequencies and mode shapes to characterize a dynamic response of the structure.
Chapter 4  
Velocity Transformation Function of Flat Shell Element

In the previous chapter, the flat shell element was developed for constructing the finite element model (FE model) of a structure. The Frequency Response Function (FRF) was discussed to represent the dynamic response of the structure when the structure was excited by a harmonic forcing function. The dynamic response model of the structure constructed by the FE model is a complex valued continuous three dimensional displacement field on the mid plane of the FE model. To use this dynamic response model of the structure in manner consistent with the experimental model in the frame work of ESDM, these following post-processing steps are needed after generating the displacement field of the FE model.

1. Generate the velocity field of the vibrating structure using the displacement field of the FE model.
2. Move the velocity field of the vibrating structure from the mid-plane of the structure to the top surface of the surface using the velocity transformation function.

In this chapter, these post processing steps will be discussed. The velocity field at the mid-plane of the flat shell element will be discussed. The velocity transformation function will be developed to transfer the velocity field from the mid-plane to top surface of flat shell element. Most importantly, the relationship between the numerical approximation of the surface velocities of structures, obtained from the FE model, and the experimental data will be discussed.

4.1 Velocity field on the mid-plain of flat shell element

In general, the displacement field of a structure obtained by the FE model can be expressed in terms of the spatial and temporal components as
\[ U(x, y, t) = S(x, y)G(t) = \overline{u}(x, y)e^{i(\omega t + \phi)} \]

where \( S(x, y) = \overline{u}(x, y) \) is the spatial term

\[ G(t) = e^{i(\omega t + \phi)} \] is the temporal term

(4.1)

The product of nodal displacements and shape functions of the flat shell element approximates the displacement field over the finite element. The approximated displacement field consists of the axial, \( u \), lateral, \( v \), and transverse, \( w \), displacement within the flat shell element. For these approximated displacements, shape functions become the spatial component and time dependent nodal displacements become the temporal component. In the last chapter, bi-linear strain and Hermetian shape functions were used to construct the shape functions of the flat shell element. Therefore, these shape functions become the spatial components and time dependent nodal displacements become the temporal components of the approximated displacements. Equation 4.2 shows the approximated displacements of the flat shell element in terms of the spatial components and the temporal components. The approximated displacements and the shape functions are written in terms of the normalized local coordinates, \( \xi \) and \( \eta \) (Equation 3.31).

\[
\{\tilde{U}(\xi, \eta, t)\} = \left[ N(\xi, \eta) \right] \{\tilde{t}(t)\}
\]

\[
\begin{bmatrix}
\{\tilde{U}(\xi, \eta, t)\} = \begin{bmatrix} u(\xi, \eta, t) \\ v(\xi, \eta, t) \\ w(\xi, \eta, t) \end{bmatrix}, & \{\tilde{t}(t)\} = \begin{bmatrix} \theta_{x1} \\ \theta_{y1} \\ \vdots \\ \theta_{y4} \end{bmatrix} e^{i(\omega t + \phi)}
\end{bmatrix}
\]
Where \( \{\tilde{\mathbf{u}}(\xi, \eta, t)\} \) is the approximated displacement field, \( [N(\xi, \eta)] \) is the shape function matrix of the flat shell element consists of bi-linear strain and Hermetian shape functions, and \( \{\tilde{\mathbf{u}}(t)\} \) are the nodal displacements.

When a harmonic forcing function is applied to the structure, the approximated velocities can be obtained from the first derivative of the approximated displacement with respect to time. Therefore the approximated velocities of the FE model can be written as
\[
\{\tilde{\mathbf{V}}(\xi, \eta, t)\} = [N(\xi, \eta)] \frac{\partial}{\partial t} \{\tilde{\mathbf{u}}(t)\}
\]

(4.3)

Therefore, using Equation 4.3 and matrix notation, the approximated velocity field within the element can be expressed in terms of the shape function matrix and nodal displacements as follow.
\[
\{\tilde{\mathbf{V}}(\xi, \eta, t)\} = \begin{bmatrix} \dot{u}(\xi, \eta, t) \\ \dot{v}(\xi, \eta, t) \\ \dot{w}(\xi, \eta, t) \end{bmatrix} = i\omega [N(\xi, \eta)] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \theta_{x1} \\ \theta_{y1} \\ \vdots \\ \theta_{y20} \end{bmatrix} e^{i(\omega t + \phi)}
\]

(4.4)

Considering \( i = e^{i\pi/2} \) and substituting this value into Equation 4.4, the temporal component of the approximated velocity field, \( e^{i(\omega t + \phi)} \), becomes \( e^{i(\omega t + \phi + \pi/2)} \).
Therefore, the approximated velocity field, \( \{ \tilde{V}(\xi, \eta, t) \} \), can be obtained by multiplying the forcing frequency, \( \omega \), and adding the phase angle \( \pi/2 \) to the approximated displacement vector, \( \{ N(\xi, \eta) \} \). Using Equation 4.5 the velocity field on the mid-plane of the FE model will be developed.

**4.2 The velocity transfer function from the mid-plane to surface of the structure**

Figures 4.1 and 4.2 show displacements and the geometry of a differential slice of a flat shell element before and after loading. The thickness of the flat shell element is \( t \), and the distance from the mid plane to the top or the bottom of surface is \( \frac{t}{2} \). Assuming the thickness of the flat shell element is small enough, the transverse shear deformation of the flat shell element can be neglected and right angles across the section in a xz and yz plane are preserved. Point O is at the mid-plane of an undeformed structure where typically the displacement field of the FE model is obtained. Point S is at the top surface of the undeformed structure where a laser takes a measurement. Point O' is at the mid-plane of the deformed structure. Point S' is at the top surface of the deformed structure. Lines normal to the mid-plane, such as the line between point O and S, \( \overline{OS} \), in the undeformed geometry remain normal to the mid-plane in the deformed geometry and these lines rotate as a result of bending. The line \( \overline{O'S'} \) shows the line \( \overline{OS} \) in the deformed structure with rotating angles, \( \theta_x \) and \( \theta_y \).
Figure 4.1 Geometry and displacements of differential slice of a flat shell element before and after loading in XZ plane

Figure 4.2 Geometry and displacements of differential slice of a flat shell element before and after loading in YZ plane
As shown in Figure 4.1 and 4.2, the displacements on the mid-plane have only $U_{na}, V_{na},$ and $W_{na}$ displacements. However, the displacement on the surface, $U_s, V_s,$ and $W_s,$ have not only $U_{na}, V_{na},$ and $W_{na}$ displacements but also the additional displacements, $U_b$ and $V_b,$ due to the bending action. Limiting the deflection angle of the flat shell element to be small, the transverse displacements, $W_{na}$ and $W_s$ will be same in the deformed geometry. The displacement on the top surface of the flat shell element can be written as

$$U_s = U_{na} + U_b$$
$$V_s = V_{na} + V_b$$
$$W_s = W_{na}$$

(4.6)

Figure 4.3 and 4.4 show an enlarged top half of the cross section in the xz and yz plane. Additional displacements, $U_b$ and $V_b,$ in the x and y directions a result of the bending action, are found from Figures 4.3 and 4.4.

$$U_b = -\frac{t}{2} \frac{\partial w}{\partial x} = -\frac{t}{2} \theta_y$$
$$V_b = \frac{t}{2} \frac{\partial w}{\partial y} = \frac{t}{2} \theta_x$$

(4.7)

Therefore, the surface displacements, $U_s, V_s,$ and $W_s$ can be written as follows

$$U_s = U_{na} + U_b = U_{na} - \frac{t}{2} \frac{\partial w}{\partial x} = U_{na} - \frac{t}{2} \theta_y$$
$$V_s = V_{na} + V_b = V_{na} + \frac{t}{2} \frac{\partial w}{\partial y} = V_{na} + \frac{t}{2} \theta_x$$
$$W_s = W_{na}$$

(4.8)

The velocities on the top surface of the flat shell element can be derived from displacements on the top surface of the flat shell element. The derivative of displacements on the top surface with respect to time is taken from Equation 4.8 will result in velocities on the top surface of the flat shell element. This result is shown in Equation 4.9. Again the small angle deflections of the flat shell element are assumed.
\[ \dot{U}_s = \dot{U}_{na} - \frac{t}{2} \dot{\theta}_y \]
\[ \dot{V}_s = \dot{V}_{na} + \frac{t}{2} \dot{\theta}_x \]
\[ \dot{W}_s = \dot{W}_{na} \]
\[
\frac{\partial w}{\partial x} = \frac{w_2 - w_1}{x_2 - x_1} = -\theta_y
\]

Figure 4.3 Close up of top half of the flat shell element in XZ plane

\[
\frac{\partial w}{\partial y} = \frac{w_2 - w_1}{y_2 - y_1} = -\theta_x
\]

Figure 4.4 Close up of top half of the flat shell element in YZ plane
Equation 4.9 can be written in the matrix form as

\[
\begin{bmatrix}
\ddot{v}_s \\
\ddot{u}_s \\
\end{bmatrix} = \begin{bmatrix}
\ddot{v}_{na} \\
\ddot{u}_{na} \\
\end{bmatrix} + \begin{bmatrix}
-t \frac{\dot{\theta}_y}{2} \\
t \frac{\dot{\theta}_x}{2} \\
0 \\
\end{bmatrix}
\]

where \( \{\ddot{v}_s\} = \{\ddot{v}_{na}\}, \{\ddot{u}_s\} = \{\ddot{u}_{na}\}, \{\ddot{v}_{na}\} = \{\ddot{v}_{na}\}\).

(4.10)

As shown in Equation 4.10, the velocity field on the top surface, \(\{\ddot{v}_s\}\), has translation velocity components, \(\{\ddot{v}_{na}\}\), and angular velocity components. The translation velocity components on the top surface are obtained from the velocities on the mid-plane defined as \(\{\ddot{v}_s\} = \{\ddot{v}_{na}\}\) in terms of shape functions \([N]\) and nodal velocities on the mid-plane, \(\{\ddot{v}_{na}\}\).

Angular velocities, \(t \frac{\dot{\theta}_x}{2}\) and \(-t \frac{\dot{\theta}_y}{2}\), can be obtained from derivative of the transverse velocity, \(\dot{W}_{na}\), with respect to the axial or lateral coordinate. The transverse velocity can be expressed in terms of the spatial shape functions and nodal velocities.

\[
\dot{W}_{na} = N_{H_1} \dot{w}_1 + N_{H_2} \dot{\theta}_1 X_1 + N_{H_3} \dot{\theta}_1 Y_1 + \cdots + N_{H_{10}} \dot{w}_4 + N_{H_{11}} \dot{\theta}_4 X_4 + N_{H_{12}} \dot{\theta}_4 Y_4
\]

where \(N_{H_{1, i=1,2,\ldots,10}}\) are Hermetian shape functions

(4.11)

Assuming a small deflection angle for the flat shell element, the derivative of the spatial shape functions of the transverse velocity, \(\dot{W}_{na}\), with respect to the axial or lateral coordinate will give the angular velocities, \(\dot{\theta}_x\) and \(\dot{\theta}_y\) as follows.
\[ \dot{\theta}_x = \frac{\partial \dot{W}_{na}}{\partial y} \]

\[ \dot{\theta}_y = \frac{\partial \dot{W}_{na}}{\partial x} \]

\[ \dot{\theta}_x = \frac{\partial N_{H1}^2}{\partial y} \dot{w}_1 + \frac{\partial N_{H2}^2}{\partial y} \dot{\theta}_{X1} + \frac{\partial N_{H3}^2}{\partial y} \dot{\theta}_{X1} + \cdots + \frac{\partial N_{H10}^2}{\partial y} \dot{w}_4 + \frac{\partial N_{H11}^2}{\partial y} \dot{\theta}_{X4} + \frac{\partial N_{H12}^2}{\partial y} \dot{\theta}_{Y4} \]

(4.12)

And

\[ \dot{\theta}_y = \frac{\partial \dot{W}_{na}}{\partial x} \]

\[ \dot{\theta}_y = \frac{\partial N_{H1}^2}{\partial x} \dot{w}_1 + \frac{\partial N_{H2}^2}{\partial x} \dot{\theta}_{X1} + \frac{\partial N_{H3}^2}{\partial x} \dot{\theta}_{X1} + \cdots + \frac{\partial N_{H10}^2}{\partial x} \dot{w}_4 + \frac{\partial N_{H11}^2}{\partial x} \dot{\theta}_{X4} + \frac{\partial N_{H12}^2}{\partial x} \dot{\theta}_{Y4} \]

(4.13)

If Equations 4.12 and 4.13 are written in matrix form, the angular velocities can be expressed in terms of the spatial shape functions and temporal nodal velocities of the flat shell element.

\[ \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & A_{X1} & A_{X2} & A_{X3} & 0 & 0 & A_{X4} & A_{X5} & A_{X6} & 0 & 0 & A_{X7} & A_{X8} & A_{X9} & 0 & 0 & A_{X10} & A_{X11} & A_{X12} \\ 0 & 0 & A_{Y1} & A_{Y2} & A_{Y3} & 0 & 0 & A_{Y4} & A_{Y5} & A_{Y6} & 0 & 0 & A_{Y7} & A_{Y8} & A_{Y9} & 0 & 0 & A_{Y10} & A_{Y11} & A_{Y12} \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{w}_1 \\ \dot{\theta}_{X1} \\ \vdots \\ \dot{\theta}_{Y20} \end{bmatrix} \]

where

\[ A_{X1} = \frac{\partial N_{H1}}{\partial x} \]
\[ A_{X2} = \frac{\partial N_{H2}}{\partial x} \]
\[ A_{X3} = \frac{\partial N_{H3}}{\partial x} \]
\[ A_{X4} = \frac{\partial N_{H4}}{\partial x} \]
\[ A_{X5} = \frac{\partial N_{H5}}{\partial x} \]
\[ A_{X6} = \frac{\partial N_{H6}}{\partial x} \]
\[ A_{X7} = \frac{\partial N_{H7}}{\partial x} \]
\[ A_{X8} = \frac{\partial N_{H8}}{\partial x} \]
\[ A_{X9} = \frac{\partial N_{H9}}{\partial x} \]
\[ A_{X10} = \frac{\partial N_{H10}}{\partial x} \]
\[ A_{X11} = \frac{\partial N_{H11}}{\partial x} \]
\[ A_{X12} = \frac{\partial N_{H12}}{\partial x} \]
\[ A_{Y1} = -\frac{\partial N_{H1}}{\partial y} \]
\[ A_{Y2} = -\frac{\partial N_{H2}}{\partial y} \]
\[ A_{Y3} = -\frac{\partial N_{H3}}{\partial y} \]
\[ A_{Y4} = -\frac{\partial N_{H4}}{\partial y} \]
\[ A_{Y5} = -\frac{\partial N_{H5}}{\partial y} \]
\[ A_{Y6} = -\frac{\partial N_{H6}}{\partial y} \]
\[ A_{Y7} = -\frac{\partial N_{H7}}{\partial y} \]
\[ A_{Y8} = -\frac{\partial N_{H8}}{\partial y} \]
\[ A_{Y9} = -\frac{\partial N_{H9}}{\partial y} \]
\[ A_{Y10} = -\frac{\partial N_{H10}}{\partial y} \]
\[ A_{Y11} = -\frac{\partial N_{H11}}{\partial y} \]
\[ A_{Y12} = -\frac{\partial N_{H12}}{\partial y} \]

and \( N_{H1,i=1,2,...,12} \) are Hermetian shape functions.

(4.14)
In Equation 4.10, the velocities on the top surface of the structure are defined in matrix form. To substitute the angular velocities defined in Equation 4.14 into Equation 4.10, the new matrix \([R]\) is defined as

\[
\begin{bmatrix}
0 & 0 & A_{y1} & A_{y2} & A_{y3} & 0 & 0 & A_{y5} & A_{y6} & 0 & 0 & A_{y7} & A_{y8} & A_{y9} & 0 & 0 & A_{y10} & A_{y11} & A_{y12} \\
0 & 0 & A_{x1} & A_{x2} & A_{x3} & 0 & 0 & A_{x5} & A_{x6} & 0 & 0 & A_{x7} & A_{x8} & A_{x9} & 0 & 0 & A_{x10} & A_{x11} & A_{x12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

(4.15)

Substituting Equation 4.15 and \([\dot{\tilde{r}}_{m}]= [N][\dot{\tilde{u}}]\) into Equation 4.10, Equation 4.10 can be rewritten as

\[
\{\ddot{\tilde{r}}\} = \{\dot{\tilde{u}}\} = [N][\dot{\tilde{u}}] + [R][\dot{\tilde{u}}]
\]

(4.16)

In Equation 4.16, the spatial shape function matrix, \([N]\), is written in the normalized local coordinates, \(\xi\) and \(\eta\). Therefore, the \([R]\) matrix has to be expressed in terms the normalized local coordinates, \(\xi\) and \(\eta\). By taking partial derivative of \(\frac{\partial N_{H_i}}{\partial x}\) and \(\frac{\partial N_{H_i}}{\partial y}\) (\(i = 1, 2, \ldots, 12\)) respect to \(\xi\) and \(\eta\), shown in Equation 4.17, the \([R]\) matrix can be written in the normalized local coordinates, \(\xi\) and \(\eta\) as

\[
\frac{\partial N_{H_i}}{\partial x} = \frac{\partial N_{H_i}}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_{H_i}}{\partial \eta} \frac{\partial \eta}{\partial x} = \frac{1}{\alpha} \frac{\partial N_{H_i}}{\partial \xi} \\
\frac{\partial N_{H_i}}{\partial y} = \frac{\partial N_{H_i}}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_{H_i}}{\partial \eta} \frac{\partial \eta}{\partial y} = \frac{1}{\beta} \frac{\partial N_{H_i}}{\partial \eta}
\]

where \(N_{H_i, i=1,2,\ldots,12}\) are Hermitian shape functions

(4.17)

Equation 4.18 shows the \([R]\) matrix, in terms of normalized local coordinate, \(\xi\) and \(\eta\).

\[
[R] = \frac{\ell}{2} \begin{bmatrix}
0 & 0 & T_{y1} & T_{y2} & T_{y3} & 0 & 0 & T_{y5} & T_{y6} & 0 & 0 & T_{y7} & T_{y8} & T_{y9} & 0 & 0 & T_{y10} & T_{y11} & T_{y12} \\
0 & 0 & T_{x1} & T_{x2} & T_{x3} & 0 & 0 & T_{x5} & T_{x6} & 0 & 0 & T_{x7} & T_{x8} & T_{x9} & 0 & 0 & T_{x10} & T_{x11} & T_{x12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Where

\[

g_{ij} = \frac{1}{\beta} \frac{\partial N_{Hj}}{\partial \eta} \quad \text{and} \quad h_{ij} = -\frac{1}{\beta} \frac{\partial N_{Hj}}{\partial \eta}
\]

Therefore, Equation 4.16 can be written as

\[

g_{ij} h_{ij} = \frac{\partial N_{Hj}}{\partial \eta}
\]

(4.18)

The nodal velocities on the mid-plane, \( \{\dot{u}\} \), can be factored out in Equation 4.16.

Therefore, Equation 4.16 can be written as

\[
\{\dot{u}_{i}\} = (\{N\} + [R])\{\dot{u}\}
\]

(4.19)

Considering the new angular velocities in the deformation of the flat shell element, the shape function of flat shell element can be updated to approximate velocity field on the surface of the structure. The updated transfer shape function is defined as \([N_t]\) and is called velocity transformation function matrix.

\[
[N_t] = [N] + [R]
\]

(4.20)

Where the \([N]\) matrix is the shape function matrix of the flat shell element and \([R]\) matrix is the angular velocity matrix which is used to find angular velocities.

Therefore, the surface velocity field can be obtained from

\[
\{\dot{v}_i\} = \{\dot{u}_{i}\} = [N_t]\{\dot{u}\}
\]

(4.21)

Where \(\{\dot{v}_i\}\) is the top surface velocity field and \(\{\dot{u}\}\) is the nodal velocity field on the mid-plane of flat shell.

Figure 4.5 shows the velocity transfer function matrix, \([N_t]\), for the flat shell element. The velocity transfer function matrix and displacement obtained by the FE model will be used to find a velocity field on the surface of structures in the next chapter.
\[
\begin{bmatrix}
N_{B1} & 0 & \frac{t}{2} T_{Y1} & \frac{t}{2} T_{X1} & N_{B2} & 0 & \frac{t}{2} T_{Y2} & \frac{t}{2} T_{X2} & N_{B3} & 0 & \frac{t}{2} T_{Y3} & \frac{t}{2} T_{X3} & N_{B4} & 0 & \frac{t}{2} T_{Y4} & \frac{t}{2} T_{X4} & N_{B5} & 0 & \frac{t}{2} T_{Y5} & \frac{t}{2} T_{X5} & N_{B6} & 0 & \frac{t}{2} T_{Y6} & \frac{t}{2} T_{X6} & N_{B7} & 0 & \frac{t}{2} T_{Y7} & \frac{t}{2} T_{X7} & N_{B8} & 0 & \frac{t}{2} T_{Y8} & \frac{t}{2} T_{X8} & N_{B9} & 0 & \frac{t}{2} T_{Y9} & \frac{t}{2} T_{X9} & N_{B10} & 0 & \frac{t}{2} T_{Y10} & \frac{t}{2} T_{X10} & N_{B11} & 0 & \frac{t}{2} T_{Y11} & \frac{t}{2} T_{X11} & N_{B12} & 0 & \frac{t}{2} T_{Y12} & \frac{t}{2} T_{X12}
\end{bmatrix}
\]

where

\[ N_{B_{i, i=1,2,...,4}} \] are bi-linear strain shape functions

\[ N_{H_{i, i=1,2,...,12}} \] are Hermetian shape functions

\[
\begin{align*}
T_{Y1} &= -\frac{1}{\beta} \frac{\partial N_{H1}}{\partial \eta} \\
T_{Y2} &= -\frac{1}{\beta} \frac{\partial N_{H2}}{\partial \eta} \\
T_{Y3} &= -\frac{1}{\beta} \frac{\partial N_{H3}}{\partial \eta} \\
T_{Y4} &= -\frac{1}{\beta} \frac{\partial N_{H4}}{\partial \eta} \\
T_{Y5} &= -\frac{1}{\beta} \frac{\partial N_{H5}}{\partial \eta} \\
T_{Y6} &= -\frac{1}{\beta} \frac{\partial N_{H6}}{\partial \eta} \\
T_{Y7} &= -\frac{1}{\beta} \frac{\partial N_{H7}}{\partial \eta} \\
T_{Y8} &= -\frac{1}{\beta} \frac{\partial N_{H8}}{\partial \eta} \\
T_{Y9} &= -\frac{1}{\beta} \frac{\partial N_{H9}}{\partial \eta} \\
T_{Y10} &= -\frac{1}{\beta} \frac{\partial N_{H10}}{\partial \eta} \\
T_{Y11} &= -\frac{1}{\beta} \frac{\partial N_{H11}}{\partial \eta} \\
T_{Y12} &= -\frac{1}{\beta} \frac{\partial N_{H12}}{\partial \eta}
\end{align*}
\]

\[
\begin{align*}
T_{X1} &= \frac{1}{\alpha} \frac{\partial N_{H1}}{\partial \xi} \\
T_{X2} &= \frac{1}{\alpha} \frac{\partial N_{H2}}{\partial \xi} \\
T_{X3} &= \frac{1}{\alpha} \frac{\partial N_{H3}}{\partial \xi} \\
T_{X4} &= \frac{1}{\alpha} \frac{\partial N_{H4}}{\partial \xi} \\
T_{X5} &= \frac{1}{\alpha} \frac{\partial N_{H5}}{\partial \xi} \\
T_{X6} &= \frac{1}{\alpha} \frac{\partial N_{H6}}{\partial \xi} \\
T_{X7} &= \frac{1}{\alpha} \frac{\partial N_{H7}}{\partial \xi} \\
T_{X8} &= \frac{1}{\alpha} \frac{\partial N_{H8}}{\partial \xi} \\
T_{X9} &= \frac{1}{\alpha} \frac{\partial N_{H9}}{\partial \xi} \\
T_{X10} &= \frac{1}{\alpha} \frac{\partial N_{H10}}{\partial \xi} \\
T_{X11} &= \frac{1}{\alpha} \frac{\partial N_{H11}}{\partial \xi} \\
T_{X12} &= \frac{1}{\alpha} \frac{\partial N_{H12}}{\partial \xi}
\end{align*}
\]

Figure 4.5 The velocity transfer function matrix
4.3 Summary

The FE model, constructed by flat shell elements, approximates the displacement field, \( \{d\} \), on the mid-plane of oscillating structures. By taking the first derivative of the displacements with respect to time, the velocity field on the mid-plane of the FE model, \( \{\dot{d}\} \), is calculated. The approximated velocity field at the laser measurement locations on the surface of the structure, \( \{\hat{V}\} \), is then recomputed using the velocity transformation function.

\[
\{\hat{V}\} = \{\dot{d}\} = [N] \{\ddot{d}\}
\]

Where the \([N]\) matrix is the velocity transformation function matrix.

For the damped system, the velocity field on the mid-plane of the FE model has the real and imaginary components. Therefore, the real and imaginary components of the approximated velocity field at the laser measurement locations on the surface of the structure can be calculated as

\[
\begin{align*}
\{\hat{V}_{s, \text{Re}}\} &= \{\dot{d}_{s, \text{Re}}\} = [N] \{\ddot{d}_{s, \text{Re}}\} \\
\{\hat{V}_{s, \text{Imag}}\} &= \{\dot{d}_{s, \text{Imag}}\} = [N] \{\ddot{d}_{s, \text{Imag}}\}
\end{align*}
\]

(4.22)

Where \( \{\dot{d}_{s, \text{Re}}\} \) and \( \{\dot{d}_{s, \text{Imag}}\} \) are the real and imaginary components of approximated velocity field on mid-plane of the FE model, and \( \{\ddot{d}_{s, \text{Re}}\} \) and \( \{\ddot{d}_{s, \text{Imag}}\} \) are the real and imaginary components of approximated velocity field of the laser measurement locations on the surface of the structure.

The approximated surface velocity field \( \{\hat{V}_{s, \text{Re}}\} \) and \( \{\hat{V}_{s, \text{Imag}}\} \) consists of the axial, \( \hat{u} \), lateral, \( \hat{v} \), and transverse, \( \hat{w} \), velocity components. The real and imaginary approximated surface velocity can be written in the vector form below.

**Real velocity:** \( \hat{V}_{s, \text{Re}} = \dot{u}_{s, \text{Re}} \hat{i} + \dot{v}_{s, \text{Re}} \hat{j} + \dot{w}_{s, \text{Re}} \hat{k} \)

**Imaginary velocity:** \( \hat{V}_{s, \text{Imag}} = \dot{u}_{s, \text{Imag}} \hat{i} + \dot{v}_{s, \text{Imag}} \hat{j} + \dot{w}_{s, \text{Imag}} \hat{k} \)

(4.23)
In Chapter 2, the real and imaginary component of experimental data was shown to be represented by the velocity on the surface of structure with direction cosines $\Psi_X$, $\Psi_Y$, and $\Psi_Z$ as follow

Real component: $\text{REAL} = (\hat{u}\Psi_X + \hat{v}\Psi_Y + \hat{w}\Psi_Z)\cos\phi$

Imaginary component: $\text{IMG} = (\hat{u}\Psi_X + \hat{v}\Psi_Y + \hat{w}\Psi_Z)\sin\phi$

(4.24)

$$\hat{u} = \text{magnitudes of velocity in x - direction} = \sqrt{\hat{u}_{Re}^2 + \hat{u}_{Imag}^2}$$

$$\hat{v} = \text{magnitudes of velocity in y - direction} = \sqrt{\hat{v}_{Re}^2 + \hat{v}_{Imag}^2}$$

$$\hat{w} = \text{magnitudes of velocity in z - direction} = \sqrt{\hat{w}_{Re}^2 + \hat{w}_{Imag}^2}$$

(4.25)

$$\Psi_X = \cos(\theta_X)$$

$$\Psi_Y = \cos(\theta_Y)$$

$$\Psi_Z = \cos(\theta_Z)$$

(4.26)

Where $\theta_X$, $\theta_Y$, and $\theta_Z$ are the angles between the positive x, y, and z coordinate directions in the structure’s coordinate system and the incident laser beam, and $\phi$ is the phase angles of the approximated velocity on the surface of the oscillating structure relative to the reference forcing function. Therefore, the experimental data which is the magnitude of the measured velocity at instant time, $t$, is written as

$$|V_L| = \sqrt{\text{REAL}^2 + \text{IMG}^2}$$

(4.27)

and relative phase angle of the measured velocity at instant time, $t$, to reference forcing function is written as

$$\phi_{\text{RL}} = \tan^{-1}\frac{\text{IMG}}{\text{REAL}}$$

(4.28)
Using the velocity transformation function matrix, \([N_r]\), a complex valued continuous three dimensional velocity field of the structure constructed by the FE model will be able to be used in a consistent manner with the experimental model in the frame work of ESDM. Substituting the surface velocity obtained by the FE model and direction cosines into Equations 4.26, 4.27 and 4.28, the experimental data can be obtained.

In the next chapter, FE models of simple structures are developed and tested to demonstrate the application of the velocity transformation function.
Chapter 5
Test Cases

The main goal of this research is to develop the flat shell element and the velocity transformation function that recomputes the velocities from the mid-plane to the surface of the finite element model. This velocity transformation function makes it possible to directly compare the finite element model (FE model) constructed by the flat shell element with the laser-based experimental data. For simple structures, such as a cantilever beam and a simply supported flat plate, the FE model represented by flat shell elements can be verified with the closed form solutions or the output of commercial software such as the I-DEAS. In this chapter, the flat shell element developed in the previous chapter is used to construct the FE models of a cantilever beam and a simply supported flat plate. The results obtained from the FE models are compared with the closed form solution or the output of I-DEAS to verify the accuracy of the FE model of the structure. Then using displacements on the mid-plane of the structure generated by the FE models and the velocity transformation function, the velocities on the surface of the structure will be determined later in this chapter. A cantilever beam will be considered as both an undamped and damped system. A simply supported rectangular plate will be considered as a damped system.

5.1 Cantilever Beam (Undamped system)

Figure 5.1 and Table 5.1 show material properties, geometry, and the loading condition of a cantilever beam. First the cantilever beam is considered to be an undamped system. The closed form solutions and results of the FE model of the cantilever beam are discussed in the following section.
5.1.1 Closed form solution

Venter (1995) explained the closed form solution for the transverse, steady-state vibrating cantilevered beam. Using the following fourth-order ordinary differential equation derived by Clough and Penzien (1975), with boundary conditions for the cantilever beam, the displacements of the cantilever beam can be calculated as

\[w''(x) - \frac{M(x)\omega^2}{EI}w(x) = 0\]

Where \(w\) is the transverse displacement

\(E\) is the elastic modulus

\(I\) is the area moment of inertia

\(\omega\) is the circular forcing frequency
**Boundary condition**

Displacement at $x = 0$ : $w(0) = 0$

Slope at $x = 0$ : $w'(0) = 0$

Moment at $x = L$ : $EIw''(L) = 0$

Shear force at $x = L$ : $w''(L) = -\frac{F}{EI}$

(5.1)

In this research, for any given forcing frequency, $\omega$, the solution of Equation 5.1 was obtained using Matlab’s (Version 5.3) ordinary differential equations solver. The solution for the transverse displacement anywhere along the length of the beam as a function of time and position is then

$$W(x,t) = w(x)\sin(\omega t)$$

(5.2)

Natural frequencies of the beam can be obtained using Euler’s equation for a beam (Thomson, 1998). Equation 5.3 shows the natural frequencies of vibration for the beam found from the solution of the characteristic equation for the transverse vibration of a beam.

$$\omega_n = \beta_n \sqrt{\frac{EI}{\rho}} = (\beta_n l)^2 \sqrt{\frac{EI}{\rho l^4}}$$

(5.3)

where $\rho$ is mass per unit length, $l$ is the length of the beam, and $(\beta_n l)^2$ is constant for typical end conditions of the beam. Table 5.2 shows numerical values of the constant $(\beta_n l)^2$ and the resulting natural frequencies of the cantilever beam.

**Table 5.2 Natural frequencies of the cantilever beam**

<table>
<thead>
<tr>
<th></th>
<th>$(\beta_n l)^2$</th>
<th>Natural Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fundamental</td>
<td>3.52</td>
<td>2.7194</td>
</tr>
<tr>
<td>Second Mode</td>
<td>22.0</td>
<td>16.9968</td>
</tr>
<tr>
<td>Third Mode</td>
<td>61.7</td>
<td>47.6678</td>
</tr>
</tbody>
</table>
5.1.2 Finite element model

Using the Flat Shell Finite Element Program (FSFE Program) developed for this research, FE models of the cantilever beam is constructed. To enforce the boundary conditions of the cantilever beam, all degrees of freedom \((u, v, w, \theta_x, \theta_z)\) of nodes at the left end of the beam are fixed. The harmonic force whose total magnitude is 5 N with forcing frequency 3 Hz is applied on the nodes at the free end of the beam. The fundamental frequency of the cantilever beam determined by Equation 5.3 is 2.72 Hz. The forcing frequency is chosen to excite the structure near the first mode of the cantilever beam shown in Figure 5.1. The forcing frequency is larger than the fundamental frequency. The total magnitude of the force is divided by number of nodes at the free end of the cantilever beam and applied to each node. In this FE model, there are two nodes at the end of the cantilever beam. Therefore, half of the total magnitude of force is applied on each of the right end nodes. Before obtaining the results from the FE model of the cantilever beam, the converged FE model needs to be found. To find the converged model, the mesh of the FE model is refined. Each time a FE model is refined, the transverse displacement at the free end of the refined FE models is compared with the closed form solution. The transverse displacement at the free end of the cantilever beam calculated using Equation 5.1 is 0.05889 m downward. Figure 5.2 shows the convergence of displacement at the free end of the FE model with respect to the number of elements used. The minus sign of the displacement in Figure 5.2 shows the direction of the displacement of the cantilever beam. The difference between closed form solution and results of the differently meshed FE models is defined as

\[
\text{Difference (\%)} = \frac{\text{FE result} - \text{Closed form solution}}{\text{Closed form solution}} \times 100
\]

Table 5.3 shows the difference between closed form solution and results of the differently meshed FE models

<table>
<thead>
<tr>
<th>Number of Element used</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference (%)</td>
<td>-66.99</td>
<td>-7.71</td>
<td>-2.03</td>
<td>-1.51</td>
<td>-1.03</td>
<td>-0.16</td>
</tr>
</tbody>
</table>
As shown in Table 5.3 and Figure 5.2, the FE model consisted of ten flat shell elements converges to within 0.5% of the closed form solution. Therefore, ten flat shell elements are used to construct the FE model of an undamped system.

![Figure 5.2 Convergence of displacement at the free end of the FE model](image)

Figure 5.2 shows the mesh of the converged FE model of the cantilever beam with node and element numbers.

![Figure 5.3 The mesh of the converged FE model](image)
5.1.3 Dynamic response of the FE model

The dynamic response of the cantilever beam can be obtained by post processing the
displacement results of the converged FE model. Figure 5.4 shows the deformed shape of the FE
model in steady-state oscillation with the forcing frequency, $f=3$ Hz.

![Deformed shape of cantilever beam](image)

**Figure 5.4 The deformed shape of the cantilever beam**

As shown in Figure 5.4, the maximum displacements are found from nodes at the free
end of the cantilever beam (nodes 23 & 24) where the forces were applied. Table 5.4 shows the
maximum displacements of the cantilever beam.

**Table 5.4 The maximum displacement of the cantilever beam (Undamped system)**

*(Sign of displacement indicates the direction of displacement)*

<table>
<thead>
<tr>
<th>Node 22</th>
<th>Displacement</th>
<th>Node 23</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>0.0 (m)</td>
<td>$u$</td>
<td>0.0 (m)</td>
</tr>
<tr>
<td>$V$</td>
<td>0.0 (m)</td>
<td>$v$</td>
<td>0.0 (m)</td>
</tr>
<tr>
<td>$W$</td>
<td>-0.0588 (m)</td>
<td>$W$</td>
<td>-0.0588(m)</td>
</tr>
<tr>
<td>$\theta_x$</td>
<td>2.1139e-05 (rad)</td>
<td>$\theta_x$</td>
<td>-2.1139e-05 (rad)</td>
</tr>
<tr>
<td>$\theta_y$</td>
<td>0.05879 (rad)</td>
<td>$\theta_y$</td>
<td>0.05879 (rad)</td>
</tr>
</tbody>
</table>
Reaction forces are found from nodes at the left end of the cantilever beam (nodes 1 & 2). Table 5.5 shows the reaction forces of the cantilevered beam.

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Reaction force</th>
<th>Node 2</th>
<th>Reaction force</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_X$</td>
<td>0</td>
<td>$F_X$</td>
<td>0</td>
</tr>
<tr>
<td>$F_Y$</td>
<td>0</td>
<td>$F_Y$</td>
<td>0</td>
</tr>
<tr>
<td>$M_X$</td>
<td>6.136 (N.m)</td>
<td>$M_X$</td>
<td>-6.136 (N.m)</td>
</tr>
<tr>
<td>$M_Y$</td>
<td>-18.243 (N.m)</td>
<td>$M_Y$</td>
<td>-18.243 (N.m)</td>
</tr>
</tbody>
</table>

Because the FE model is constructed using the flat shell element, as shown in Table 5.4 and 5.5, three translation displacements ($u$, $v$, and $w$) and reaction forces ($F_X$, $F_Y$, and $F_Z$) and two angular displacements ($\theta_X$ and $\theta_Y$) and moments ($M_X$ and $M_Y$) are found from the output of the FE model. If a FE model of a cantilever beam is constructed by a 2D beam element which is a combination of a rod element and a simple plane beam element, two translation displacements ($u$ and $w$) and reaction forces ($F_X$ and $F_Z$) and one rotational angle, $\theta_Y$, and moment, $M_Y$, can be determined from the results of the FE model. For a narrow beam, lateral outputs ($v$, $F_Y$, $\theta_Y$, and $M_Y$) of the FE model constructed by flat shell element are small enough to ignore and the 2D beam element is accurate enough to represent a FE model of a beam. However, as the width of a beam increases, these lateral outputs increase and become useful to explain the necking and other physical phenomena of a beam.

From Chapter 3, receptance and mobility of a structure were defined as

\[
\text{Receptance : } H(\omega) = \frac{d(\omega)}{F(\omega)} \\
\text{Mobility : } Y(\omega) = \frac{\dot{d}(\omega)}{F(\omega)} = i\omega H(\omega)
\]

Where $d(\omega)$ is a structural displacement

\[
\dot{d}(\omega) \text{ is a structural velocity} \\
F(\omega) \text{ is an excitation force}
\]

(5.5)
Receptance and mobility of the cantilever beam can be obtained by post processing the displacement results of the FE model. Receptance and mobility at the free end of the cantilever beam are obtained over the frequency range between 0 to 100 Hz where the frequency range included the first three natural frequencies. Figure 5.5 shows the magnitude and relative phase angle of receptance of the undamped FE model of the cantilever beam as function of frequency.

The condition where the frequency of the applied load equals the natural frequency is called resonance. The steady-state response of the cantilever beam tends toward infinity at resonance. As shown in Figure 5.5, resonance peaks near the natural frequencies of the cantilever beam. The relative phase angle, $\phi$, varies from zero to $-180$ degrees as the forcing frequency passes through resonance in Figure 5.5. Figure 5.6 shows the magnitude and the relative phase angle of the mobility function. The relative phase angle of the mobility of the beam was shifted by 90 degrees from the relative phase angle of the receptance function due to multiplying the receptance by $i\omega$. The relative phase angle of the mobility is zero degrees at the resonance.
Figure 5.5 Receptance of the cantilever beam (at node 22, undamped system)

Figure 5.6 Mobility of the cantilever beam (at node 22, undamped system)
5.1.4 Solution of Normal Modes

The fundamental, second, and third transverse bending modes, and first twist natural frequencies are obtained from the converged FE model. Table 5.6 shows natural frequencies from the FE model, the exact solution found earlier, and difference between results of the FE model and the exact solution. The difference between results of the FE model and exact solution is defined as

\[
\text{Difference (\%) = } \frac{\text{Result of the FE model} - \text{Exact solution}}{\text{Exact solution}} \times 100
\]

Table 5.6 Natural frequencies of the cantilever beam

<table>
<thead>
<tr>
<th>Mode Type</th>
<th>FE Model (Hz)</th>
<th>Exact solution (Hz)</th>
<th>Difference (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Bending (Fundamental)</td>
<td>2.7167</td>
<td>2.7163</td>
<td>0.01</td>
</tr>
<tr>
<td>2nd Bending (Second Mode)</td>
<td>17.4153</td>
<td>17.156</td>
<td>1.2</td>
</tr>
<tr>
<td>3rd Bending (Third Mode)</td>
<td>51.5079</td>
<td>50.723</td>
<td>1.5</td>
</tr>
<tr>
<td>First twist</td>
<td>96.3433</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.7 shows mode shapes of the cantilever beam FE model.
5.2 Cantilever Beam (Damped system)

In this section the cantilever beam is considered as a damped system. In this research, structural damping is used to represent the damping in the structure. Using the structural damping factor and complex stiffness matrix, the damped FE model of the cantilever beam is constructed. The cantilever beam is considered to be a very lightly damped system (Under-damped condition). The structural damping factor of the cantilevered beam is taken to be 1 %.

5.2.1 Dynamic response of the FE model

The damped FE model of the cantilever beam consists of ten flat shell elements excited at \( f = 3 \) Hz. In steady-state oscillation, the nodal displacements obtained from the FE model of damped system are complex values due to the complex stiffness matrix. Using the complex stiffness matrix, \((1 + i \gamma[K])\), the equation of motion for the damped MDOF system with structural damping is written as

\[
(-\omega^2[M] + (1 + i \gamma)[K])\{x\} = \{f\}
\]

(5.7)

Where \( \gamma \) is the structural damping factor. As described in Chapter 3.1.6, the real and imaginary components of displacement of the cantilever beam can be determined from

\[
\{x\} = [D]^{-1}\{f\}.
\]

(5.8)

The dynamic stiffness matrix is defined as

\[
[D] = (-\omega^2[M] + (1 + i \gamma)[K]) = \begin{bmatrix}
[K] - \omega^2[M] & -\gamma[K] \\
\gamma[K] & [K] - \omega^2[M]
\end{bmatrix}
\]

(5.9)

and the real and imaginary components of the displacement and force are

\[
\{x\} = \left\{ \begin{array}{l}
\{x_{\text{real}}\} \\
\{x_{\text{imaginary}}\}
\end{array} \right\}, \quad \{f\} = \left\{ \begin{array}{l}
\{f_{\text{real}}\} \\
\{f_{\text{imaginary}}\}
\end{array} \right\}
\]

(5.10)
Table 5.7 shows the real and imaginary values of the maximum transverse displacement of the damped FE model which are obtained from node 22 at the free end of the cantilever beam where the forces are applied. Figure 5.8 shows the transverse displacement of the damped FE model.

**Table 5.7 Displacement of the cantilever beam at node 12 (damped system)**

<table>
<thead>
<tr>
<th>Displacement</th>
<th>Real component</th>
<th>Imaginary component</th>
<th>Magnitude</th>
<th>Relative phase angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>0 (m)</td>
<td>0 (m)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$V$</td>
<td>0 (m)</td>
<td>0 (m)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W$</td>
<td>-5.868e-2 (m)</td>
<td>-2.6914e-3 (m)</td>
<td>5.907e-2(m)</td>
<td>-177.36 (degree)</td>
</tr>
<tr>
<td>$\theta_X$</td>
<td>-2.345e-5 (rad)</td>
<td>-9.5484e-7 (rad)</td>
<td>2.358e-5 (rad)</td>
<td>-177.36 (degree)</td>
</tr>
<tr>
<td>$\theta_Y$</td>
<td>5.7898e-2 (rad)</td>
<td>2.724e-3 (rad)</td>
<td>5.893-2 (rad)</td>
<td>-177.36 (degree)</td>
</tr>
</tbody>
</table>

**Figure 5.8 Real, imaginary, magnitude, and relative phase angle of transverse displacement**
Figure 5.9 shows the damped and undamped transverse displacement of the node 22 at the free end of the cantilevered beam in the complex plane.

![Figure 5.9 Undamped and damped transverse displacement of the cantilever beam](image)

Undamped transverse displacement has only the real value but the damped transverse displacement has both real and imaginary values. In general, structural damping has the effect of changing the relative phase angle between reference force and response. Since the applied reference force of the cantilevered beam has only the real value, as shown in Figure 5.9, the relative phase angle of the undamped system is 180 degrees out of phase but the relative phase angle of displacement of damped system is –177.34 degrees respect to the forcing function. Because between modes 1 and 2, structural damping is proportional damping, the relative phase angle of displacement of each node respect to the reference force is same. Table 5.8 shows the reaction forces obtained at node 1 of the damped FE model. The reaction forces are complex values. For the damped system, like the displacement, the relative phase angle of the reaction forces is no longer zero because of the imaginary component of reaction forces.

<table>
<thead>
<tr>
<th>Reaction force</th>
<th>Real component</th>
<th>Imaginary component</th>
<th>Magnitude</th>
<th>Relative phase angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_X$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_Y$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$F_Z$</td>
<td>19.089 (N)</td>
<td>0.978 (N)</td>
<td>19.091 (N)</td>
<td>-177.36 (degree)</td>
</tr>
<tr>
<td>$M_X$</td>
<td>6.1197 (N.m)</td>
<td>0.331 (N.m)</td>
<td>6.198(N.m)</td>
<td>-177.36 (degree)</td>
</tr>
<tr>
<td>$M_Z$</td>
<td>-18.194 (N.m)</td>
<td>-0.983 (N.m)</td>
<td>18.194(N.m)</td>
<td>-177.36 (degree)</td>
</tr>
</tbody>
</table>
Receptance and mobility of a damped system at the free end were obtained in the frequency range between 0 to 100 Hz. Figure 5.10 shows the magnitude and relative phase angle of receptance of the damped and undamped system as a function of frequency. Figure 5.11 shows the magnitude and relative phase angle of mobility of the damped and undamped system as a function of frequency.

In general structural damping presented in oscillatory systems causes the decay of the magnitude of free vibration. The effect of structural damping in frequency response functions of the structure is shown in Figure 5.10 and 5.11. As shown in Figure 5.10 and 5.11, peaks that appear near the natural frequencies of the undamped system become smaller and wider for the damped system. In the undamped system, the frequency response function exhibit instantaneous relative phase angle changes at the natural frequency. However, in the damped system these relative phase angle changes are smoother. Figure 5.12 shows the mobility of the undamped and damped system with different structural damping factors. Figure 5.12 shows the effect of varying the structural damping factor to 2 % and 5 %.
Figure 5.10 Receptance of the cantilever beam (at node 22, damped system)

Figure 5.11 Mobility of the cantilever beam (at node 22, damped system)
Figure 5.12 (a) Mobility of the cantilever beam (at node 22, damping ratio = 2%)
5.2.2 Solution of Normal Modes

For damped systems in general, the natural frequencies and mode shapes are complex values. In this research, the equation of motion for the damped MDOF systems with structural damping is defined as \((-\omega^2[M] + (1 + i\gamma)[K])\{x\} = \{f\}\). The natural frequencies and mode shape of the damped system are determined by the characteristic equation defined as

\[
\text{Det} \left( -\omega^2[M] + (1 + i\gamma)[K] \right) = 0
\]

(5.11)

Damped natural frequencies and decay rates are obtained from the damped FE model. Table 5.9 shows frequencies of damped oscillation and decay rate.

<table>
<thead>
<tr>
<th>Mode shape</th>
<th>Damped natural frequency (Hz)</th>
<th>Decay rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>First bending</td>
<td>2.7168</td>
<td>0.0136</td>
</tr>
<tr>
<td>Second bending</td>
<td>17.5156</td>
<td>0.0876</td>
</tr>
<tr>
<td>Third bending</td>
<td>51.1085</td>
<td>0.2555</td>
</tr>
<tr>
<td>First twist</td>
<td>96.3445</td>
<td>0.4817</td>
</tr>
</tbody>
</table>

For the fundamental frequency, Genta (1995) defined the damped natural frequency and decay rate of damped system with structural damping

Damped natural frequency: \(\omega_d = \omega_n \sqrt{1 + \sqrt{1 + \gamma^2}} / 2 = \omega_n\)

(5.12)

Decay rate: \(\delta = \omega_n \sqrt{-1 + \sqrt{1 + \gamma^2}} / 2 \approx \omega_n \gamma / 2\)

(5.13)

Where \(\omega_d\) is damped natural frequency, \(\omega_n\) is fundamental frequency of undamped system, \(\gamma\) is structural damping factor, and \(\delta\) is the decay rate. Therefore, the closed form solution of the damped natural frequency is 2.7194 Hz and decay rate is 0.0135. The differences of the closed form solution and damped FE model at the fundamental frequency are calculated as
Difference of Frequency of damped oscillation = \(\frac{\text{FE result} - \omega_d}{\omega_d} \times 100 = 0.1\%\)

Difference of Decay rate = \(\frac{\text{FE result} - \delta}{\delta} \times 100 = 0.1\%\)

Because structural damping, which is proportional damping, is used for the FE model of damped system, the mode shapes of the damped FE model are much like the mode shapes of undamped system as shown in Figure 5.8. However the mode shapes of the damped system have exponential decay in amplitude at all points in the damped system at the same rate. Therefore, the motion of mode shape of damped system is diminished in amplitude until the system comes to rest (Hurty, Rubinstein, 1964).

The solution for the dynamic response and the normal modes of the FE model is consistent with the closed form solution of the cantilever beam shown in Figure 5.1. Therefore, the flat shell element is proven to be valid for constructing the FE model of the cantilever beam. In the next section, using the FE model and velocity transformation function, the dynamic responses at the top surface of the cantilever beam are obtained.
5.3 Dynamic response model of the cantilever beam

The dynamic response model of the structure, used in ESDM formulation, is the complex-valued continuous three dimensional velocity field on the top surface of a structure. This velocity field on the top surface of a structure can be approximated by the FE model of the structure and the velocity transformation function of the FE model. The complex-valued continuous three dimensional velocity field on the mid-plane of a structure can be determined from the FE model of the structure. Using the velocity transformation function, this velocity field on the mid-plane of the FE model can be recomputed for the velocity field on the top surface of the FE model.

5.3.1 Dynamic response model for a undamped system

Since an undamped system does not have real component of the velocity field, the imaginary valued continuous three dimensional velocity field is constructed for a undamped system. Using nodal displacements of the FE model of the cantilever beam and shape functions of the flat shell element, the displacement fields of the FE model at laser measurement location on the mid-plane of the beam are determined. Figure 5.13 shows the laser measurement location within in the flat shell element.

Where \( \alpha \) is the length of the element and \( \beta \) is the width of the element.

![Figure 5.13 25 laser measurement locations within in the flat shell element](image-url)
Assuming a harmonic forcing function is applied to the cantilevered beam, the velocity field on the mid-plane of the FE model of the beam can be obtained by the first derivative of the displacement field of the FE model of the beam with respect to time. Figure 5.14 shows the velocity on the mid-plane of the FE model. As shown in Figure 5.14, since only transverse forces are applied on the cantilever beam, both axial (x-axis) and lateral (y-axis) velocity is zero and the transverse (z-axis) velocity on the mid-plane of the FE model of the beam is determined. Using the velocity transformation function of the FE model and the velocity on the mid-plane of the FE model, the velocity field on top surface of the FE model is constructed. Figure 5.15 shows the velocity on top surface of the FE model. As shown in Figure 5.15, on the top surface of the FE model, not only the transverse velocity but also axial (x-axis) and lateral (y-axis) velocity are obtained due to the bending action of the cantilever beam. The transverse (z-axis) velocity of the beam on the mid-plane and the top surface of the beam are the same because small deformation of the FE model of the beam. The axial velocity of the beam shows the tension on the surface of the beam. These velocities on the top surface of the FE model represent the velocities on the top surface of the cantilevered beam and are used in ESDM formulation.
Figure 5.14 Imaginary velocity field of on the mid-plane of the FE model (undamped system)
Figure 5.15 Imaginary velocity field of on the top surface of cantilever beam (undamped system)
5.3.2 Dynamic response model for a damped system

Using the damped dynamic stiffness matrix, defined as
\[
\begin{bmatrix}
[K] - \omega^2[M] & \gamma[K] \\
\gamma[K] & [K] - \omega^2[M]
\end{bmatrix},
\]
the real and imaginary displacement components of the FE model of the cantilever beam are obtained. Using the real and imaginary displacement components of the FE model and shape functions of the flat shell element, the real and imaginary displacement components of the FE model at laser measurement location, shown in Figure 5.13, on the mid-plane of the FE model are determined. Assuming a harmonic forcing function is applied to the cantilever beam, the real and imaginary velocity components on the mid-plane of the FE model of the beam can be obtained by the first derivative of the real and imaginary displacement components of the FE model of the beam with respect to time. Figure 5.16 and 5.17 show the real, imaginary, magnitude and relative phase angle of velocity on the mid-plane of the cantilever beam.

As shown in Figure 5.16 and 5.17, since only transverse forces are applied on the cantilever beam, the velocity components (real, imaginary, magnitude and relative phase angle) of axial (x-axis) and lateral (y-axis) velocity are zero and the velocity components of transverse (z-axis) velocity on the mid-plane of the FE model is determined. Relative phase angle of transverse velocity of the FE model is shifted by 90 degrees from the relative phase angle of transverse displacement of the FE model, shown in Figure 5.8. Figure 5.18, 5.19, and 5.20 show the real, imaginary, magnitude and relative phase angle of velocities on top surface of the FE model. As shown in Figure 5.18, 5.19 and 5.20, at the top surface of the FE model, not only the velocity components of the transverse (z-axis) velocity but also the velocity components of axial (x-axis) and lateral (y-axis) velocity are obtained due to the bending action of the cantilever beam. The transverse velocities of the beam on the mid-plane and the top surface of the beam are the same because of small deformations of the FE model of the beam. Relative phase angles of velocities on the top surface of the FE model are shifted by 90 degrees from the relative phase angle of displacements on the top surface of the FE model. These velocities on the top surface of the FE model are used in the ESDM formulation.
Figure 5.16 Axial (x axis) and lateral (y axis) directional velocity of the cantilever beam on the mid-plane
Figure 5.17 Transverse velocity of the cantilever beam on the mid-plane
Figure 5.18 X directional (axial) velocity of the cantilever beam on the top surface
Figure 5.19 Y directional (lateral) velocity of the cantilever beam on the top surface
Figure 5.20 transverse velocity of the cantilever beam on the top surface
5.4 Simply supported rectangular plate

Blotter (1995) conducted an experiment designed to measure the power flow of a plate using a three-stage experimental spatial power flow (ESPF) method. In this experiment, velocities on the top surface of the rectangular plate were obtained by a Scanning Laser Doppler Vibrometer. The rectangular plate was mounted to a rigid steel frame by thin steel shims and screws. The shims were also rigidly attached to a heavy support frame to simulate a simply supported plate. Two shakers were tuned to have the same magnitude and a phase difference of approximately 10 degrees. The rectangular plate was forced at 79.0 Hz which resulted in an operating shape with a similar form to the first natural mode shape of the rectangular plate. Figure 5.21 and Table 5.10 show the material properties, geometry, and the loading condition of the simply supported rectangular plate used in the experiment. The two shakers had a phase difference of approximately 10 degrees. The FE model of undamped system can represent only the structure which has multiple forces either exactly in-phase or out-of-phase. Therefore, for the structure, which has multiple exciting forces with phase difference, superposing the FE models of undamped system with separate loading condition is required. Since, it is easy to construct the FE model of a damped system for the tested rectangular plate, the rectangular plate was considered only to be a damped system. Outputs of the FE models are compared with experimental results and I-DEAS outputs.

Figure 5.21 The simply supported rectangular plate
Table 5.10 Material properties, geometry, and load of the rectangular plate

<table>
<thead>
<tr>
<th>Material</th>
<th>Geometry</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Modulus = 196 GPa</td>
<td>Length = 380 mm</td>
<td>Force 1 = 0.85 N</td>
</tr>
<tr>
<td>Poisson's Ratio = 0.27</td>
<td>Width = 300 mm</td>
<td>Force 2 = 0.85 N</td>
</tr>
<tr>
<td>Mass density = 7905 kg/m$^3$</td>
<td>Thickness = 2 mm</td>
<td>Forcing Frequency = 79 Hz</td>
</tr>
<tr>
<td>Reference Force = Force 1</td>
<td>Relative phase angle of Force 2 respect to Force 1 = -5 degrees</td>
<td></td>
</tr>
</tbody>
</table>

5.4.1 Finite Element Model

Using the Flat shell Finite Element Program, FE models of the rectangular plate are constructed. The rectangular plate is considered to be a very lightly damped system (Under damped condition). The structural damping factor of the rectangular plate is taken to be 1 %. To enforce boundary conditions of the rectangular plate used in this experiment, translation degrees of freedom ($u, v$ and $w$) at all four edges of the rectangular plate are fixed and rotation degrees of freedom ($\theta_x$ and $\theta_z$) at all four edges of the rectangular plate are set free. Force 1 and Force 2 were applied on nodes of the rectangular plate. The two forces had the same magnitude but Force 2 had a relative phase difference of approximately -5 degrees with respect to Force 1. To find a converged FE model the magnitude of maximum transverse velocity of the rectangular plates were compared with the experimental results. Given that Force 1 and 2 are harmonic forcing functions and the transverse displacement of the rectangular plate is $\bar{w}e^{i(\omega t+\phi)}$, where $\bar{w}$ is the magnitude of transverse displacement, $\phi$ is the relative phase angle and $\omega$ is the circular forcing frequency, the transverse velocity can be calculated as

Displacement = $\bar{w}e^{i(\omega t+\phi)} = \bar{w}e^{i\omega t}e^{i\phi} = \hat{w}e^{i\omega t}$

Velocity $\dot{w} = \frac{\partial w}{\partial t} = i\omega \hat{w} e^{i\omega t}$

Where $\hat{w} = \bar{w}e^{i\phi} = \bar{w}(\cos(\phi) + i\sin(\phi))$

\[ (5.14) \]

The magnitude of the maximum normal velocity can be obtained from

\[ |\hat{w}| = \sqrt{(i \omega \bar{w} \cos(\phi))^2 + (-\omega \bar{w} \sin(\phi))^2} \]

\[ (5.15) \]
Figure 5.22 shows the experimental result of the real component of the transverse velocity of the rectangular plate. The maximum magnitude of the transverse velocity obtained from the experiment was 7.953 (mm/s) (Blotter, 1995). Figure 5.23 shows the real component of the transverse velocity of the FE model. Table 5.11 shows the maximum of the magnitude of the transverse velocity of the FE modes and difference between results of experiment and FE models. The percent difference between experimental result and output of FE model is calculated as

$$\text{Difference} \% = \frac{\text{Output of FE model} - \text{Experimental result}}{\text{Experimental result}} \times 100$$

(5.16)

Figure 5.22 shows the convergence of the maximum magnitude of transverse velocity of the rectangular plate.

Table 5.11 Difference between the experimental result and output of FE models

<table>
<thead>
<tr>
<th>Number of element used</th>
<th>9</th>
<th>12</th>
<th>16</th>
<th>24</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum magnitude transverse velocity (mm/s)</td>
<td>16.9</td>
<td>7.143</td>
<td>7.1322</td>
<td>7.748</td>
<td>7.773</td>
</tr>
<tr>
<td>Difference (%)</td>
<td>112.498</td>
<td>10.185</td>
<td>10.321</td>
<td>2.578</td>
<td>2.26</td>
</tr>
</tbody>
</table>

Figure 5.22 Real component of the transverse velocity of the rectangular plate measured by SLDV (79.0 Hz)
Figure 5.23 Real component of the transverse velocity of the rectangular plate (79 Hz)

Figure 5.24 Convergence of the maximum magnitude of transverse velocity of the rectangular plate

As shown in Table 5.11 and Figure 5.24, the FE model consisted of 30 flat shell elements converge within 2.26% of the experimental result. Therefore, 30 flat shell elements are used to construct the FE model of the rectangular plate which oscillates with forcing frequency 79 Hz. Figure 5.25 shows the mesh of the converged FE model.
5.4.2 Dynamic response of the FE model

Figure 5.26 shows the displacements of the converged FE model in steady-state oscillation. The maximum displacements were found in the middle of the rectangular plate (node 21). Table 5.12 shows the maximum displacements of the rectangular plate in steady-state oscillation.
Table 5.12 Displacements of rectangular plate at node 21

<table>
<thead>
<tr>
<th>Displacement</th>
<th>Real Values</th>
<th>Imaginary Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$V$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W$</td>
<td>9.713e-5 (m)</td>
<td>-5.4850e-6 (m)</td>
</tr>
<tr>
<td>$\theta_x$</td>
<td>-9.8e-3 (rad/sec)</td>
<td>5.8662e-6 (rad/sec)</td>
</tr>
<tr>
<td>$\theta_y$</td>
<td>3.843e-4 (rad/sec)</td>
<td>-2.2507e-6 (rad/sec)</td>
</tr>
</tbody>
</table>

Transverse reaction forces can be found at the nodes along edges of the rectangular plate. Table 5.13 and 5.14 show the magnitude and relative phase of transverse reaction forces of the rectangular plate.

![Figure 5.26 Transverse displacement of the rectangular plate](image)
### Table 5.13 Magnitude of transverse reaction forces of the rectangular plate (unit of Force is N)

<table>
<thead>
<tr>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
<th>Node#</th>
<th>(F_z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.18</td>
<td>7</td>
<td>0.52</td>
<td>13</td>
<td>1.90</td>
<td>19</td>
<td>2.65</td>
<td>25</td>
<td>1.70</td>
<td>31</td>
<td>0.65</td>
<td>37</td>
<td>1.13</td>
</tr>
<tr>
<td>2</td>
<td>2.11</td>
<td>8</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>38</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>0.29</td>
<td>9</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>39</td>
<td>0.35</td>
</tr>
<tr>
<td>4</td>
<td>0.41</td>
<td>10</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>40</td>
<td>0.35</td>
</tr>
<tr>
<td>5</td>
<td>1.82</td>
<td>11</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>41</td>
<td>1.99</td>
</tr>
<tr>
<td>6</td>
<td>1.18</td>
<td>12</td>
<td>0.46</td>
<td>18</td>
<td>1.79</td>
<td>24</td>
<td>2.68</td>
<td>30</td>
<td>1.80</td>
<td>36</td>
<td>0.72</td>
<td>42</td>
<td>1.21</td>
</tr>
</tbody>
</table>

### Table 5.14 Relative phase angle of transverse reaction forces of the rectangular plate (unit of angle is degree)

<table>
<thead>
<tr>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
<th>Node#</th>
<th>(\theta_z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.7</td>
<td>7</td>
<td>177.3</td>
<td>13</td>
<td>177.3</td>
<td>19</td>
<td>177.3</td>
<td>25</td>
<td>177.3</td>
<td>31</td>
<td>177.3</td>
<td>37</td>
<td>-2.7</td>
</tr>
<tr>
<td>2</td>
<td>177.3</td>
<td>8</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>38</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>177.3</td>
<td>9</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>39</td>
<td>0.35</td>
</tr>
<tr>
<td>4</td>
<td>177.3</td>
<td>10</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>40</td>
<td>0.35</td>
</tr>
<tr>
<td>5</td>
<td>177.3</td>
<td>11</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>41</td>
<td>1.99</td>
</tr>
<tr>
<td>6</td>
<td>-2.7</td>
<td>12</td>
<td>177.3</td>
<td>18</td>
<td>177.3</td>
<td>24</td>
<td>177.3</td>
<td>30</td>
<td>177.3</td>
<td>36</td>
<td>177.3</td>
<td>42</td>
<td>-2.7</td>
</tr>
</tbody>
</table>

Figure 5.27 and 5.28 show the frequency response function of the rectangular plate. Receptance and mobility of the rectangular plate at node 14, where reference Force 1 is applied to the structure, are obtained in frequency range between 0 to 350 Hz. This frequency range includes the first three natural frequencies of the rectangular plate. Figure 5.27 shows the magnitude and relative phase angle of the receptance function of the rectangular plate as a function of frequency. As shown in Figure 5.27, in the given frequency range, the resonance peaks are shown at the first, second, and third natural frequencies. Figure 5.28 shows the magnitude and relative phase angle of the mobility function of the rectangular plate as a function of frequency. The relative phase of the mobility function is shifted by 90 degrees from the relative phase angle of the receptance function. Figure 5.29 shows the mobility of the rectangular plate with different structural damping factors. Figure 5.29 shows the effect of varying the structural damping factor from 2 % to 5 %.
Figure 5.27 Receptance of the rectangular plate (at node 14)

Figure 5.28 Mobility of the rectangular plate (at node 14)
Figure 5.29 (a) Mobility of the rectangular plate (at node 14, $\gamma = 0.5\%$)

Figure 5.29 (b) Mobility of the rectangular plate (at node 14, $\gamma = 5\%$)
5.4.3 Solution of Normal Modes

The first, second, and third natural mode shape of the rectangular plates are obtained by post processing the solution of the normal modes of the FE model constructed by the flat shell element. Also since the FE model is considered to be a lightly damped system, damped natural frequencies and decay rates are determined by the characteristic equation, defined in Equation 5.11, associated with its mode shapes. Figure 5.30 shows the mode shapes of the FE model. Table 5.15 shows damped natural frequency and decay rate of the FE model.

<table>
<thead>
<tr>
<th>Mode Shape</th>
<th>Damped natural frequency (Hz)</th>
<th>Decay rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>First mode</td>
<td>83.24</td>
<td>0.416</td>
</tr>
<tr>
<td>Second mode</td>
<td>180.15</td>
<td>0.901</td>
</tr>
<tr>
<td>Third mode</td>
<td>236.32</td>
<td>1.18</td>
</tr>
</tbody>
</table>

To verify the accuracy of the mode shape and damped natural frequency of the FE model constructed by the flat shell element, these results are compared with the mode shape and natural frequency of the rectangular plate obtained by commercial software, I-DEAS. Table 5.16 shows the natural frequency of the plate obtained by the FE model constructed by the flat shell element and the FE model generated by I-DEAS.

<table>
<thead>
<tr>
<th>Mode Shape</th>
<th>Damped natural frequency (Hz) (FSFE)</th>
<th>Damped natural frequency (Hz) (I-DEAS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First mode</td>
<td>83.24</td>
<td>81.63</td>
</tr>
<tr>
<td>Second mode</td>
<td>180.15</td>
<td>175.14</td>
</tr>
<tr>
<td>Third mode</td>
<td>236.32</td>
<td>236.32</td>
</tr>
</tbody>
</table>

Figure 5.30 (a), (b), and (c) show the mode shapes of the rectangular plate obtained by the Flat Shell Finite Element Program (FSFE Program). Figure 5.31 (a), (b), and (c) show the mode shapes of the rectangular plate obtained by I-DEAS.
Figure 5.30(a) The first mode shape obtained by FSFE Program (83.241 Hz)

Figure 5.30(b) The second mode shape obtained by FSFE Program (180.15 Hz)

Figure 5.30(c) The third mode shape obtained by FSFE Program (236.32 Hz)
Figure 5.31(a) The first mode shape of the rectangular plate obtained by I-DEAS

Figure 5.31(b) The second mode shape of the rectangular plate obtained by I-DEAS
Figure 5.31(c) The third mode shape of the rectangular plate obtained by I-DEAS

The solution of dynamic response and normal mode of the FE model are consistent with the experimental results and the output of FE model made by I-DEAS. Therefore, the flat shell element is proven to be valid for constructing the FE model of the rectangular plate shown in Figure 5.21. Given an accurate FE model of the rectangular plate, shown in Figure 5.21, the FE model and velocity transformation function will simulate the experimental result of the rectangular plate obtained by SLDV.
5.4.4 Dynamic response model of the simply supported plate

Using the damped dynamic stiffness matrix, defined as

\[
\begin{bmatrix}
[K] - \omega^2 [M] & -\gamma [K] \\
\gamma [K] & [K] - \omega^2 [M]
\end{bmatrix},
\]

the real and imaginary displacement components of the FE model of the simply supported plate are obtained. Using the real and imaginary displacement components of the FE model and shape functions of the flat shell element, the real and imaginary displacement components of the FE model at laser measurement location on the mid-plane of the FE model are obtained. Assuming the harmonic forcing function is applied to the simply supported plate, the real and imaginary velocity components on the mid-plane of the FE model of the plate can be obtained by the first derivative of the real and imaginary displacement components of the FE model of the plate with respect to time. Figures 5.32 and 5.33 show the real, imaginary, magnitude and relative phase angle of velocity on the mid-plane of the simply supported plate.

As shown in Figures 5.32 and 5.33, since only transverse forces are applied on the simply supported plate, the velocity components (real, imaginary, magnitude and relative phase angle) of axial (x-axis) and lateral (y-axis) velocity are zero and the velocity components of transverse (z-axis) velocity on the mid-plane of the FE model are determined. The relative phase angle of transverse velocity of the FE model is shifted by 90 degrees from the relative phase angle of transverse displacement of the FE model, shown in Figure 5.26. Figures 5.34, 5.35, and 5.36 show the real, imaginary, magnitude and relative phase angle of velocities on top surface of the FE model. As shown in Figures 5.34, 5.35, and 5.36, the top surface of the FE model, not only the velocity components of the transverse (z-axis) velocity but also the velocity components of axial (x-axis) and lateral (y-axis) velocity are obtained. The transverse velocities of the plate on the mid-plane and the top surface of the plate are the same. Relative phase angles of velocity on the top surface of the FE model are shifted by 90 degrees from the relative phase angle of displacements on the top surface of the FE model. These velocities on the top surface of the FE model represent the velocities on the top surface of the simply supported plate, and could be used in the ESDM formulation.
Figure 5.32 Axial (x-axis) and lateral (y-axis) directional velocity of the plate on the mid-plane
Figure 5.33 Transverse (z-axis) velocity of the plate on the mid-plane
**Figure 5.34 Axial (x-axis) velocity of the plate on the top surface**
Figure 5.35 Lateral (y-axis) velocity of the plate on the top surface
Figure 5.36 Transverse (z-axis) velocity of the plate on the top surface
5.5 Summary

In this chapter, the flat shell element was proven to be valid for constructing FE models for simple structures such as a beam and a plate. Therefore, the FE model constructed by the flat shell element can be used to determine the dynamic response and solution of normal modes of a beam and a plate. Using the velocity transformation function of the flat shell element, the FE model can represent the velocity on the surface of simple structures. This velocity on the surface of the FE model can be used in the ESDM formulation with experimental data. In chapter 2, the relationship between the velocity on the surface of the structure and the laser-based experimental data, which are magnitudes of measured velocities, is explained.

Real and Imaginary components of the magnitude of measured velocity are defined as

\[
\text{Real component: } \text{REAL} = \left( \hat{u} \Psi_x + \hat{v} \Psi_y + \hat{w} \Psi_z \right) \cos \phi
\]

\[
\text{Imaginary component: } \text{IMG} = i \left( \hat{u} \Psi_x + \hat{v} \Psi_y + \hat{w} \Psi_z \right) \sin \phi
\]

(5.17)

\(\hat{u}, \hat{v}, \text{ and } \hat{w}\) are magnitudes of the axial, lateral, and transverse velocities on the surface of the structure. \(\Psi_x, \Psi_y\), and \(\Psi_z\) are direction cosines of the laser beam, \(\phi\) is the relative phase angle of the velocity on the surface of the oscillating structure.

The magnitude of the measured velocity at instant time, \(t\), is written as

\[|V_L| = \sqrt{\text{REAL}^2 + \text{IMG}^2}\]

(5.18)

and relative phase angle of the measured velocity at instant time, \(t\), relative to reference forcing function is written as

\[\phi_{rz} = \tan^{-1} \frac{\text{IMG}}{\text{REAL}}\]

(5.19)

Using the velocity transformation function, the FE model can determine \(\hat{u}, \hat{v}, \hat{w}\), and \(\phi\), and the direction cosines of the laser beam can be obtained from the orientation of the laser. Therefore, using Equation 5.17, 5.18 and 5.19 the velocity obtained by the FE model can be consistent with the experimental data.
Chapter 6

Conclusion

A summary of thesis, conclusion of this research, and recommendations for future work are given in this chapter.

6.1 Summary of thesis

This thesis introduces the velocity transformation function within the framework of Experimental Spatial Dynamics Modeling (ESDM) which correlates the FE model with laser-based experimental data.

The main research goals of this thesis are given as

1. Develop a damped flat shell element which can be used to construct a FE model of a structure and be capable of simulating laser-based experimental data obtained by a Scanning Laser Doppler Vibrometer (SLDV).
2. Develop the velocity transformation function which will transfer dynamic responses of the FE model from the mid plane to the outer surface of the FE model where laser-based experimental data is acquired by a SLDV.
3. Determine the dynamic response on the surface of structures using the FE model constructed by the damped flat shell FE and the velocity transformation function.

To achieve these main research goals, this thesis began by showing basic formulation of ESDM. Using the relationship between the velocity on the top surface of a structure and the laser-based experimental data, the error formulation of ESDM was developed. Using this error function, the FE model of the structure can be verified with laser-based experimental data. In this research, the damped flat shell element was developed to construct the FE models of structures.
Development of the damped flat shell element began with reviewing the flat shell element developed by Venter (1995). In his research, Venter developed the flat shell element using the stiffness matrix and the mass matrix of the bi-linear plane stress element (Cook, Malkus, and Plesha 1989) and the Kirchoff plate element (Haug, Choi, and Komkov 1986). Since the use of Venter’s flat shell element was limited to an undamped system only, the flat shell was updated for a damped system. In this research, structural damping represented the proportional damping of a structure. Adding the structural damping factor and the complex stiffness matrix which combined the structural damping matrix and the stiffness matrix to Venter’s flat shell element, the damped flat shell element was developed and used for a damped system. The FE model, consisting of the damped flat shell element, determined dynamic response and the solution of normal modes of a structure. The dynamic response of structures obtained by the FE model was the nodal displacement on the mid-plane of the FE model of the structure.

Using nodal displacements of the FE model of the structure and the shape functions of the damped flat shell element, displacements of the FE model at each laser measurement location on the mid-plane of the structure was determined. Assuming the harmonic forcing function was applied to the structure, the velocity at each laser measurement location on the mid-plane of the FE model of structure was obtained by the first derivative of displacements of the FE model of structures with respect to time. The velocity transformation function makes the dynamic response obtained by the FE model consistent with laser-based experimental data gathered by a SLDV.

To test the damped flat shell element and the velocity transformation function, FE models of the cantilever beam and the simply supported plate were constructed. The dynamic response and the solution of normal modes of the test structures obtained from the FE models were compared with the closed form solution, the output of I-DEAS, or the experimental data to verify the accuracy of FE models of test structures. After comparing the dynamic response and the solution of normal modes obtained by FE models with the closed form solution, the output of I-DEAS, or the experimental data, the damped flat shell element was proven to be valid to construct FE models for test structures. Using the velocity on the mid plane of test structures obtained from the FE models and the velocity transformation function, the velocities on the surface of test structures were determined.
The velocity on the top surface of the cantilever beam or the simply supported plate was consistent and could be compared with laser-based experimental data obtained by a SLDV. The velocity on the top surface of the cantilever beam and the simply supported plate showed not only the displacement of the structures due to the applied force but also the displacement due to the deformed shape of the structure. Test cases of the cantilever beam and the simply supported plate proved that using the velocity transformation function, dynamic response of the FE model on the top surface can be obtained and the dynamic response of the FE model can be consistent with laser-based experimental data acquired by a SLDV.

6.2 Conclusions of this research

The hypothesis statement for this research, introduced in Chapter 1, is that the finite element model determines dynamic response on the mid-plane of the structure and laser-based experiment data are obtained from the surface of the structure by a Scanning Laser Doppler Vibrometer (SLDV). The development of a velocity transformation function for the Finite Element model, dynamic response on the outer surface of the FE model can be obtained and can be made consistent with laser-based experimental data measured by a SLDV.

This hypothesis statement for this research was satisfied by determined the velocity on the top surface of test structures by the FE models and the velocity transformation function. Two major conclusions of this research can be drawn from this research.

The first results of this research is the development of the damped flat shell element. In this research the damped flat shell element was developed to construct the FE model of a structure. The Finite Element modal of a structure constructed by the damped flat shell element is capable as below

1. Generate the FE models for undamped and damped system
2. Determine the dynamic response on the mid plane of a structure
3. Determine the Frequency response function of a structure
4. Determine the solution of normal mode of a structure
The second result of this research is the development of the velocity transformation function. The velocity transformation function of the FE model was developed based on the properties of the damped flat shell element, a linear system, and the deformation mechanics of a structure. Using the velocity transformation function, the velocity on the top surface of a structure can be determined from the FE model. The velocity on the top surface of the structure obtained by the FE model and the velocity transformation function is consistent with the laser-based experimental data obtained by a SLDV.

Therefore, the velocity transformation function developed in this research will make possible the use of the FE model, constructed by the damped flat shell element, and the laser-based experimental data within a framework of ESDM in a consistent manner.

6.3 Recommendations for future work

Some suggestions regarding future work of the ESDM and development of finite element are mentioned as the follows.

The damped flat shell element developed in this research has several limitations. First the damped flat shell element is a rectangular shell element. When the surface of a structure has an arbitrary curvature, or is difficult to model with the rectangular shape, the damped flat shell element can’t be used. Therefore, the generalized shell element, which can represent generalized curved surface, needs to be developed. Second, in this research, structural damping, which is proportional damping, represents the damping of structures. Although the proportional damping can represent the lightly damped (under-damped) and simple structures such as beams or plates, since a real structure has non-proportional damping in many cases, it is necessary to develop the finite element model which uses non-proportional damping. Also in this research the FE models of structures were used to find a dynamic response and solution of normal modes of structures. The application of the FE model constructed by the damped flat shell element can be expanded to find the sensitivity with respect to modal parameters, the power flow within a structure, and the stress analysis of a structure. Therefore, the post processing of the FE model can be developed for the future research need.
The velocity transformation function was developed based on the properties of the damped flat shell element, a linear system and the deformation mechanics of a flat shell structure. However in this research, assuming thickness of the element is small enough, transverse shear deformation of a structure was neglected, and sides of the damped flat shell element remained straight when the element deformed in-plane. When the thickness of the element is no longer small, this assumption is violated and the transverse shear deformation needs to be considered. Therefore, in the future work, the velocity transformation function needs to be developed to include the transverse shear deformation of a structure.

Lastly, the flat shell element and the velocity transformation function were developed by making use of the Matlab (Version 5.3) software package. This software is ideal to develop and apply the new procedures due to the fact that the Matlab is easy to use and to program the computer source code. However, the disadvantage of using the Matlab is that computation time of computer source code for large problems is relatively longer than other source code built by more efficient programming language the such as FORTRAN or C. Therefore, it is strongly recommended that use FORTRAN or C to develop the generalized shell element to increase the application of the FE model.
References


Appendix A

Matlab source files for the Flat Shell Finite Element Program
(Undamped System)

A.1 Dynamic response of a structure

plateFRF.m (main file)
Dofmap.m
PlateAssembleGlobalDyM.m
PlateStaticStiff.m
PlateElementLengths.m
mass.m
RodAssembleMatrix.m
displacementplot.m

A.2 Frequency response function (mobility)

Platemobility.m

A.3 Solution of normal modes of a structure

modalsummation.m (main file)
Dofmap.m*
PlateAssembleGlobalDyM.m*
PlateStaticStiff.m*
mass.m*
RodAssembleMatrix.m*
eigenvalues.m
normal.m
modeplot.m
A.1 Dynamic response of a structure

plateFRF.m (main file)

```matlab
function [u, f, node, k, m] = plateFRF(node, element, xforce, yforce, zforce,...
xmoment, zmoment, matrl)

[node, element, adofx, pdofx, udofx, adofy, pdofy, udofy, ...
adofz, pdofz, udofza,adofza, pdofta, pdofza, udofza, ...]

w = 3*2*pi;                % Forcing frequency (rad/sec) : w=3 Hz
Kd = (k.uu-(w^2)*m.uu);
u.u = Kd\(f.u - (k.up-(w^2)*m.up) * u.p);

% Solve the reaction eqn for unknown forces
f.p = (k.pp-(w^2)*m.pp)* u.p + (k.pu-(w^2)*m.pu) * u.u;
```

Dofmap.m

```matlab
function [node, element, adofx, pdofx, udofx, adofy, pdofy, udofy, ...
adofz, pdofz, udofza,adofza, pdofta, pdofza, udofza, ...]

null_id = -1;                             % set the null id
n_node_per_elmnt = 4;                     % 4 nodes per rod element
n_dof_per_node   = 5;                     % get # dof/node
n_node = length(node);                   % get ttl # of nodes
n_elmnt = length(element);               % get ttl # of elements
```

If you want to use this program or have any question, please contact to
kyongchan@yahoo.com
udofza.n_dof = 0;
for node_id = 1: n_node
% do case of constraint on dofx
switch node(node_id).spc(1)
% do case of unknown dofx
    case null_id
        % current dofx is free
        udfx.n_dof = udfx.n_dof + 1; % increment # of udfx
        udfx_id = udfx.n_dof;
        udfx.node(udfx_id) = node_id; % add node id to udfx map
        now update nodal data structure
        node(node_id).udof(1) = udfx_id; % assign udf id to
        node(node_id).pdof(1) = null_id; % assign NULL pdof id
    % do case of prescribed dofx map
    otherwise
        % current dofx is fixed
        % update prescribed dofx map
        pdofx.n_dof = pdofx.n_dof + 1; % increment # of pdofx
        pdofx_id = pdofx.n_dof;
        pdofx.node(pdofx_id) = node_id; % add node id to pdofx map
        now update nodal data structure
        node(node_id).pdof(1) = pdofx_id; % assign pdof id to
        node(node_id).udof(1) = null_id; % assign NULL udfx id
    end
end do case of constraint
end
for node_id = 1: n_node
% do case of constraint on dofy
switch node(node_id).spc(2)
% do case of unknown dof
    case null_id
        % current dofy is free
        udfy.n_dof = udfy.n_dof + 1; % increment # of udfy
        udfy_id = udfy.n_dof;
        udfy.node(udfy_id) = node_id; % add node id to udfy map
        now update nodal data structure
        node(node_id).udof(2) = udfy_id; % assign udf id to
        node(node_id).pdof(2) = null_id; % assign NULL pdof id
    % do case of prescribed do
    otherwise
        % current dofy is fixed
        % update prescribed do map
        pdofy.n_dof = pdofy.n_dof + 1; % increment # of pdofy
        pdofy_id = pdofy.n_dof;
        pdofy.node(pdofy_id) = node_id; % add node id to pdofy map
        now update nodal data structure
        node(node_id).pdof(2) = pdofy_id; % assign pdof id to
        node(node_id).udof(2) = null_id; % assign NULL udfy id
    end
end do case of constraint
end
for node_id = 1: n_node
% do case of constraint on dofz
switch node(node_id).spc(3)
% do case of unknown dof
    case null_id
        % current dofz is free
        udfz.n_dof = udfz.n_dof + 1; % increment # of udfz
        udfz_id = udfz.n_dof;
        udfz.node(udfz_id) = node_id; % add node id to udfz map
        now update nodal data structure
        node(node_id).udof(3) = udfz_id; % assign udf id to
        node(node_id).pdof(3) = null_id; % assign NULL pdof id
    % do case of prescribed dof
    otherwise
        % current dofz is fixed
        % update prescribed dof map
        pdofz.n_dof = pdofz.n_dof + 1; % increment # of pdofz
        pdofz_id = pdofz.n_dof;
        pdofz.node(pdofz_id) = node_id; % add node id to pdofz map
        now update nodal data structure
        node(node_id).pdof(3) = pdofz_id; % assign pdof id to
        node(node_id).udof(3) = null_id; % assign NULL udfz id
    end
end
for node_id = 1: n_node
    % do case of constraint on dof xa
    switch node(node_id).spc(4)
        % do case of unknown dof xa
        case null_id
            % current dof xa is free
            udofxa.n_dof = udofxa.n_dof + 1; % increment # of udofxa
            udofxa_id = udofxa.n_dof; % set current udofxa id
            udofxa.node(udofxa_id) = node_id; % add node id to udofxa map
            % now update nodal data structure
            node(node_id).udof(4) = udofxa_id; % assign udofxa id to
            node(node_id).pdof(4) = null_id; % assign NULL pdofxa id
        % do case of prescribed dof
        otherwise % current dof xa is fixed
            % update prescribed dof map
            pdofxa.n_dof = pdofxa.n_dof + 1; % increment # of pdofxa
            pdofxa_id = pdofxa.n_dof; % set current pdofxa id
            pdofxa.node(pdofxa_id) = node_id; % add node id to pdofxa map
            % now update nodal data structure
            node(node_id).pdof(4) = pdofxa_id; % assign pdofxa id to
            node(node_id).udof(4) = null_id; % assign NULL udofxa id
        end %end do case of constraint
    end %end switch
for node_id = 1: n_node
    % do case of constraint on dof za
    switch node(node_id).spc(5)
        % do case of unknown dof za
        case null_id
            % current dof za is free
            udofza.n_dof = udofza.n_dof + 1; % increment # of udofza
            udofza_id = udofza.n_dof; % set current udofza id
            udofza.node(udofza_id) = node_id; % add node id to udofza map
            % now update nodal data structure
            node(node_id).udof(5) = udofza_id; % assign udofza id to
            node(node_id).pdof(5) = null_id; % assign NULL pdofza id
        % do case of prescribed dof
        otherwise % current dof za is fixed
            % update prescribed dof map
            pdofza.n_dof = pdofza.n_dof + 1; % increment # of pdofza
            pdofza_id = pdofza.n_dof; % set current pdofza id
            pdofza.node(pdofza_id) = node_id; % add node id to pdofza map
            % now update nodal data structure
            node(node_id).pdof(5) = pdofza_id; % assign pdofza id to
            node(node_id).udof(5) = null_id; % assign NULL udofza id
        end %end do case of constraint
    end %end switch
for id = 1: n_node
    switch node(id).pdof(1)
        case null_id
            adof.n_dof = adof.n_dof; % increment the # of assembled dof
            adof_id = adof.n_dof;
        otherwise % current dof za is fixed
            adof.n_dof = adof.n_dof + 1; % increment the # of assembled dof
            pdof.n_dof = pdof.n_dof + 1;
            adof_id = adof.n_dof;
            pid = node(id).pdof(1);
            node_id = pdofx.node(pid);
            adofx.node(node_id) = node_id;
            node(id).adof(1) = adof_id;
            node(id).pdofx = pdof.n_dof;
        end %end switch
    end %end switch
for id = 1: n_node
    switch node(id).pdof(2)
        case null_id
            adof.n_dof = adof.n_dof; % increment the # of assembled dof
            adof_id = adof.n_dof;
        otherwise % current dof za is fixed
            adof.n_dof = adof.n_dof + 1; % increment the # of assembled dof
            pdof.n_dof = pdof.n_dof + 1;
            adof_id = adof.n_dof;
        end %end switch
pid = node(id).pdof(2);
node_id = pdofy.node(pid);
node(id).adof(2) = adof_id;
node(id).pdofy = pdof.n_dof;
end
switch node(id).pdof(3)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;                       % increment the # of assembled dof
end
switch node(id).pdof(4)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;                       % increment the # of assembled dof
end
switch node(id).pdof(5)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;                       % increment the # of assembled dof
end
for id = 1: n_node
switch node(id).udof(1)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;                       % increment the # of assembled dof
end
switch node(id).udof(2)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;                       % increment the # of assembled dof
end

node(id).udofy = udof.n_dof;
end
switch node(id).udof(3)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;  % increment the # of assembled dof
udof.n_dof = udof.n_dof + 1;
    adof_id = adof.n_dof;
pid = node(id).udof(3);
    node_id = udoz.node(pid);
adofz.node(id) = node_id;
    node(id).adof(3) = adof_id;
    node(id).udofz = udof.n_dof;
end
switch node(id).udof(4)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;  % increment the # of assembled dof
udof.n_dof = udof.n_dof + 1;
    adof_id = adof.n_dof;
pid = node(id).udof(4);
    node_id = udofxa.node(pid);
adofxa.node(id) = node_id;
    node(id).adof(4) = adof_id;
    node(id).udofxa = udof.n_dof;
end
switch node(id).udof(5)
case null_id
adof.n_dof = adof.n_dof;
otherwise
adof.n_dof = adof.n_dof + 1;  % increment the # of assembled dof
udof.n_dof = udof.n_dof + 1;
    adof_id = adof.n_dof;
pid = node(id).udof(5);
    node_id = udofza.node(pid);
adofza.node(id) = node_id;
    node(id).adof(5) = adof_id;
    node(id).udofza = udof.n_dof;
end end
for elmnt_id = 1:n_elmnt
% do for each node on current rod element
    for i = 1:n_node_per_elmnt
        node_id = element(elmnt_id).node(i);  % get current node id
        for j = 1:n_dof_per_node
            element(elmnt_id).dof(j+5*(i-1)) = node(node_id).adof(j);  % set assm global dof id
        end
    end
% end do foe each node
end

PlateAssembleGlobalDyM.m
*******************************************************************************
% Function:     PlateAssembleGlobalDyM
% Description:  Matlab M-File to assemble the FE matrices (stiffness, mass
%                displacement, force) for the flat shell element.
*******************************************************************************
function [k, u, f, m]=PlateAssembleGlobalDyM(node, element, matrl, xforce, ...
yforce, zforce, xmoment, zmoment, adofx, pdocx, ...
udofx, adofy, pdocy, udofy, adofz, pdocz, udofz, adofxa, ...
pdocxa, udofxa, adofza, pdocza, udofza, adof, pdoc, udof)
null_id = -1;                                 % null id
n_node  = length(node);
end
n_elmnt = length(element);
kg = zeros( adof.n_dof, adof.n_dof );       % assembled [kg] (working matix)
k.pp = zeros( pdof.n_dof, pdof.n_dof );       % partitioned [kpp]
k.pu = zeros( pdof.n_dof, udof.n_dof );       % partitioned [kpu]
k.uu = zeros( udof.n_dof, udof.n_dof );       % partitioned [kuu]
mg   = zeros( adof.n_dof, adof.n_dof );       % assembled [mg] (working matix)
m.pp = zeros( pdof.n_dof, pdof.n_dof );       % partitioned [mpp]
m.pu = zeros( udof.n_dof, pdof.n_dof );       % partitioned [mpu]
m.up = zeros( udof.n_dof, udof.n_dof );       % partitioned [mup]
m.uu = zeros( udof.n_dof, udof.n_dof );       % partitioned [muu]
ug   = zeros( adof.n_dof, 1 );                % \[ug\]
up   = zeros( pdof.n_dof, 1 );                % \[up\]
uu   = zeros( udof.n_dof, 1 );                % \[uu\]
f.p  = zeros( pdof.n_dof, 1 );                % \[f.p\]
f.u  = zeros( udof.n_dof, 1 );                % \[f.u\]
n_elmnt = length( element );

for elmnt_id = 1:n_elmnt
    matrl_id = element(elmnt_id).matrl;
    [ke] = PlateStaticStiff( element(elmnt_id), node,  matrl(matrl_id));
    [me] = mass( element(elmnt_id), node,  matrl(matrl_id));
    [kg] = RodAssembleMatrix( kg, ke, element(elmnt_id) );
    [mg] = RodAssembleMatrix( mg, me, element(elmnt_id) );
end

% now partition the stiffness matrix
k.pp = kg(1:pdof.n_dof, 1:pdof.n_dof);                 % extract \[kpp\]
k.pu = kg(1:pdof.n_dof, pdof.n_dof+1:adof.n_dof);     % extract \[kpu\]
k.up = kg(pdof.n_dof+1:adof.n_dof, 1:pdof.n_dof);     % extract \[kup\]
k.uu = kg(pdof.n_dof+1:adof.n_dof, pdof.n_dof+1:adof.n_dof); % extract \[kuu\]

% now partition the mass matrix
m.pp = mg(1:pdof.n_dof, 1:pdof.n_dof);                 % extract \[mpp\]
m.pu = mg(1:pdof.n_dof, pdof.n_dof+1:adof.n_dof);     % extract \[mpu\]
m.up = mg(pdof.n_dof+1:adof.n_dof, 1:pdof.n_dof);     % extract \[mup\]
m.uu = mg(pdof.n_dof+1:adof.n_dof, pdof.n_dof+1:adof.n_dof); % extract \[muu\]

fudof.n_id=0;
for id = 1: n_node
    switch node(id).udof(1)
    case null_id
        fudof.n_id=fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(1);
        node_id = udofx.node(uid);
        force_id = node(node_id).xforce;
        if ( force_id == null_id )
            f.u(fudof.n_id) = 0.0;                  % xforce magn = 0.0 (no load)
        else
            f.u(fudof.n_id) = xforce(force_id).magn; % set to specified load
        end
    end
    switch node(id).udof(2)
    case null_id
        fudof.n_id=fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(2);
        node_id = udofy.node(uid);
        force_id = node(node_id).yforce;
        if ( force_id == null_id )
            f.u(fudof.n_id) = 0.0;                  % yforce magn = 0.0 (no load)
        else
            f.u(fudof.n_id) = yforce(force_id).magn; % set to specified load
        end
    end
    switch node(id).udof(3)
    case null_id
        fudof.n_id=fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(3);
        node_id = udofz.node(uid);
        force_id = node(node_id).zforce;
        if ( force_id == null_id )
            f.u(fudof.n_id) = 0.0;                  % zforce magn = 0.0 (no load)
        else
            f.u(fudof.n_id) = zforce(force_id).magn; % set to specified load
        end
    end
end
end
end
switch node(id).udof(4)
case null_id
fudof.n_id = fudof.n_id;
otherwise
    fudof.n_id = fudof.n_id + 1; % increment the # of assembled dof
    uid = node(id).udof(4);
    node_id = udofxa.node(uid);
    force_id = node(node_id).xmoment;
    if (force_id == null_id)
        f.u(fudof.n_id) = 0.0; % moment magn = 0.0 (no load)
    else
        f.u(fudof.n_id) = xmoment(force_id).magn; % set to specified moment
    end
end
switch node(id).udof(5)
case null_id
fudof.n_id = fudof.n_id;
otherwise
    fudof.n_id = fudof.n_id + 1; % increment the # of assembled dof
    uid = node(id).udof(5);
    node_id = udofza.node(uid);
    force_id = node(node_id).zmoment;
    if (force_id == null_id)
        f.u(fudof.n_id) = 0.0; % moment magn = 0.0 (no load)
    else
        f.u(fudof.n_id) = zmoment(force_id).magn; % set to specified moment
    end
end
end

PlateStaticStiff.m

%*******************************************************************************
%    Function:     PlateStaticStiff
%    Description:  Matlab M-File to compute static
%                  stiffness matrix of the flat shell element.
%*******************************************************************************
function [ke,element]=PlateStaticStiff(element,node,matrl)
    [a,b] = PlateElementLengths(element, node);
    e = matrl.elastic_modulus ; % elastic modulus
    r = matrl.poisson ; % poisson's ratio
    g = b/a; % aspect ratio
    t = element.tick; % thickness of plate
    % Constant values of plain stress matrix
    aa=a^2;
    bb=b^2;
    c=(r+1);
    d=(3*r-1);
    k1 = 4*(-aa-2*bb+aa*r);
    k2 = -3*a*b*c;
    k3 = 4*(-2*aa-bb+bb*r);
    k4 = 4*(aa-bb-aa*r);
    k5 = -3*a*b*d;
    k6 = k1;
    k7 = 3*a*b*d;
    k8 = 2*(4*aa-bb+bb*r);
    k9 = -k2;
    k10= k3;
    kp11 = [ k1,k2, 0, 0, k4, k7, 0, 0, 0; %
              k2,k3, 0, 0, k5, k8, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              k4,k5, 0, 0, k6, k9, 0, 0, 0; %
              k7,k8, 0, 0, k9,k10, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
              0, 0, 0, 0, 0, 0, 0, 0, 0; %
    kk1 = 2*(aa+2*bb-AA*r);
    kk2 = k9;

129
kk3 = 2*(-aa+4*bb+aa*r);
kk4 = -k7;
kk5 = k9;
kk6 = 2*(2*aa+bb-bb*r);
kk7 = k7;
kk8 = k11 = kk1; kk12 = -kk2;
kk13 = kk7;
kk14 = 4*(-aa+bb-bb*r);
kk15 = -kk5;
kk16 = kk6;
kp22 = [ kk1, kk2, 0, 0, 0, kk3, kk4, 0, 0, 0;... kk5, kk6, 0, 0, 0, kk7, kk8, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... kk9, kk10, 0, 0, 0, kk11, kk12, 0, 0, 0;... -kk13, kk14, 0, 0, 0, kk15, kk16, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
kp=((e*t)/(24*a*b*(r^2-1)))*[kp11,kp22;kp22,kp11];
% Kirchoff Plate
% constant value of Kirchoff plate
rr  = g^2;
nrr = g^(-2);
z   = rr+nrr;
x   = (1+4*r);
y   = (1-r);
v   = (14-4*r);
k01 = 4*z+(1/5)*v;
k02 = (2*nrr+(1/5)*x)*b;
k03 = ((4/3)*nrr+(4/15)*y)*bb;
k04 = -(2*rr+(1/5)*x)*a;
k05 = -r*a*b;
k06 = ((4/3)*rr+(4/15)*y)*aa;
k07 = 2*(rr-2*nrr)-(1/5)*v;
k08 = -(2*nrr+(1/5)*y)*b;
k09 = (-rr+(1/5)*x)*a;
k10 = ko1;
k11 = (2*nrr+(1/5)*y)*b;
k12 = ((2/3)*nrr-(1/15)*y)*bb;
k13 = -ko2;
k14 = ko3;
k15 = ko9;
k16 = ((2/3)*rr-(4/15)*y)*aa;
k17 = ko4;
k18 = -ko5;
k19 = ko6;
ki11 = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;... 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
k11 = -2*z+(1/5)*v;
k12 = (-nrr+(1/5)*y)*b;
k13 = (rr-(1/5)*y)*a;
k14 = -2*(2*rr-nrr)-(1/5)*v;
k15 = (-nrr+(1/5)*x)*b;
k16 = (2*rr+(1/5)*y)*a;
k17 = (nrr-(1/5)*y)*b;
k18 = ((1/3)*nrr+(1/15)*y)*bb;
k19 = (-nrr+(1/5)*x)*b;
ki10 = ((2/3)*nrr-(4/15)*y)*bb;
ki11 = (-rr+(1/5)*y)*a;
ki12 = ((1/3)*rr+(1/15)*y)*aa;
ki13 = -(2*rr+(1/5)*y)*a;
ki14 = ((2/3)*rr-(1/15)*y)*aa;
ki15 = ki4;
ki16 = -ki5;
ki17 = ki6;
ki18 = ki1;
ki19 = -ki2;
ki20 = ki3;
ki21 = -ki9;
ki22 = ki10;
ki23 = -ki7;
ki24 = ki8;
ki25 = -(2*rr+(1/5)*y)*a;
ki26 = ki14;
ki27 = (-rr+(1/5)*y)*a;
ki28 = ki12;
kir21 = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, ki1, -ki2, -ki3, 0, 0, ki4, -ki5, -ki6; 0, -ki7, ki8, 0, 0, -ki9, ki10, 0, 0, 0; 0, -ki11, 0, ki12, 0, -ki13, 0, ki14, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, ki15, -ki16, -ki17, 0, 0, ki18, -ki19, -ki20; 0, -ki21, ki22, 0, 0, -ki23, ki24, 0, 0, 0; 0, -ki25, 0, ki26, 0, -ki27, 0, ki28];
kir22 = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, ki1, ki2, ki3, 0, ki4, ki5, ki6; 0, ki7, ki8, 0, 0, ki9, ki10, 0, 0; 0, 0, ki11, 0, ki12, 0, 0, ki13, 0, ki14; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, ki15, ki16, ki17, 0, ki18, ki19, ki20; 0, 0, ki21, ki22, 0, 0, ki23, ki24, 0, 0; 0, ki25, 0, ki26, 0, ki27, 0, ki28];
kic = ((e*t^3)/(12*a*b*(1-r^2)))*[kir11,kir21;kir31,kir22];
ke = kp+kic;

PlateElementLengths.m
function [a,b] = PlateElementLengths(element, node)
node_id1 = element.node(1);               % get 1st node id
node_id2 = element.node(2);               % get 2nd node id
node_id3 = element.node(3);
a = node(node_id3).x - node(node_id1).x;
b = node(node_id2).y - node(node_id1).y;   % compute length

mass.m
function mass
mass.m

function[m]=mass(element,node,matrl)
[a,b] = PlateElementLengths(element, node);
p=matrl.mass_density;
t=element.tick;
ms=[4,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2,1,0,0,0,0; 0,4,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
mps=((a*b*p*t)/36)*ms;
m1 = 24178;
m2 = 3227*b;
m3 = 560*b^2;
m4 = -3227*a;
m5 = -441*a*b;
m6 = 560*a^2;
m7 = 8582;
m8 = 1918*b;
m9 = -1393*a;
m10= 24178;
m11= -1918*b;
m12= -420*a^2;
m13= 294*a*b;
m14= -3227*b;
m15= 560*b^2;
m16= -1393*a;
m17= -294*a^2;
m18= 280*a^2;
m19= -3227*a;
m20= 441*a*b;
m21= 560*a^2;
m22= 2758;
m23= 812*b;
m24= -812*a;
m25= 8582;
m26= -1393*b;
m27= -1918*a;
m28= 24178;
m29= -812*b;
m30= -210*a^2;
m31= 196*a*b;
m32= -1393*b;
m33= 280*b^2;
m34= 294*a*b;
m35= -3227*b;
m36= 560*b^2;
m37= -812*a;
m38= -196*a*b;
m39= -210*a^2;
m40= -1918*a;
m41= -294*a*b;
m42= -420*a^2;
m43= 3227*a;
m44= -441*a*b;
m45= 560*a^2;
m46= 8582;
% Function: RodAssembleMatrix
% Description: Matlab M-File to assemble element matrix into global matrix

function [kg] = RodAssembleMatrix( kg, ke, element )
% define rod element parameters
n_dof = length(element.dof);  % # of element dof
% do for each row dof -> row coefficient in element matrix
for row_id = 1: n_dof
dof_id = element.dof(row_id);  % get current assm row dof id
% do for each column dof -> column coefficient in element matrix
for col_id = 1: n_dof
c dof_id = element.dof(col_id);  % get current assm column id
% now assemble element stiffness coeff with global stiffness coeff

mk = zeros(n_dof, n_dof);  
mp = ((a*b*p*t)/176400) * mk;
m = mps + mp;
end
end

RodAssembleMatrix.m
\[ \text{kg}(\text{rdof}_i, \text{cdof}_i) = \text{kg}(\text{rdof}_i, \text{cdof}_i) + \text{ke}(\text{row}_i, \text{col}_i); \]
end
% end do for each column dof
end
% end do for each row dof

displacementplot.m

%*******************************************************************************
%    Function:     displacementplot
%    Description:  Matlab M-File to plot the displacement of nodes
%*******************************************************************************
function [xd, yd, zd] = displacementplot(node, u)
    null_id = -1;
n = length(node);
    for i = 1:n
        if node(i).pdofx == null_id
            index = node(i).udofx;
            xx(i) = u.u(index);
        else
            xx(i) = 0;
        end
    end
    for i = 1:n
        if node(i).pdofy == null_id
            index = node(i).udofy;
            yy(i) = u.u(index);
        else
            yy(i) = 0;
        end
    end
    for i = 1:n
        if node(i).pdofz == null_id
            index = node(i).udofz;
            zz(i) = u.u(index);
        else
            zz(i) = 0;
        end
    end
    n = length(node);
    null_id = -1;
    i = 1;
    for j = 1:n-1
        if node(1).x == node(j+1).x
            i = i + 1;
        else
            i = i;
        end
    end
    j = n/i;
    index = 1;
    x = zeros(i, j);
    y = zeros(i, j);
    z = zeros(i, j);
    for c = 1:j
        for r = 1:i
            x(r, c) = node(index).x;
            y(r, c) = node(index).y;
            z(r, c) = 0;
            index = index + 1;
        end
    end
    index = 1;
    xd = zeros(i, j);
    yd = zeros(i, j);
    zd = zeros(i, j);
    for c = 1:j
        for r = 1:i
            xd(r, c) = xx(index);
            yd(r, c) = yy(index);
            zd(r, c) = zz(index);
index=index+1;
end
end
subplot(2,1,1),surf(x+xd,y+yd,z+zd)
title('Displacement of Plate'),xlabel('X'),ylabel('Y'),zlabel('Z')
axis equal
subplot(2,1,2),surf(x+xd,y+yd,z+zd)
title('Displacement of Plate'),xlabel('X(m)'),ylabel('Y(m)'),zlabel('Z(m)')

A.2 Frequency response function (mobility)

Platemobility.m

*******************************************************************************
% Function:    Platemobility
% Description: Matlab M-File to compute and plot mobility
*******************************************************************************
function [mo,phase]=Platemobility(node,k,m,f,u)
null_id = -1;
% Get the node number from user
noden = input('Please enter the node number : ');
index=input('Please enter the degree of the node:x=1,y=2,z=3,mx=4,mz=5 ');
switch index
  case 1
    pdof.id = node(noden).pdofx;
    if pdof.id ~= null_id;
      disp('Mobility can not be found in this node')
    else
      num=node(noden).udofx;
    end
  case 2
    pdof.id = node(noden).pdofy;
    if pdof.id ~= null_id;
      disp('Mobility can not be found in this node')
    else
      num=node(noden).udofy;
    end
  case 3
    pdof.id = node(noden).pdofz;
    if pdof.id ~= null_id;
      disp('Mobility can not be found in this node')
    else
      num=node(noden).udofz;
    end
  case 4
    pdof.id = node(noden).pdofmx;
    if pdof.id ~= null_id;
      disp('Mobility can not be found in this node')
    else
      num=node(noden).udofmx;
    end
  case 5
    pdof.id = node(noden).pdofmz;
    if pdof.id ~= null_id;
      disp('Mobility can not be found in this node')
    else
      num=node(noden).udofmz;
    end
j=0;
for a=1:100;
  ff=a*6.28;
  j=j+1;
  fr(j)=ff;
  w(j)=a;
  Kd=(k.uu-((ff)^2)*m.uu);
  dis = Kd\(f.u - k.up * u.p);
  b(j)=dis(num);
  phase(j)=180*angle((i*fr(j)*dis(num))/f.u(num))/pi;
  mo(j)=abs(i*fr(j)*b(j))/f.u(num);
end
end
% Plot the mobility and displacement of the given node
subplot(2,1,1),semilogy(w,mo),axis([0 100 10e-9 10])
xlabel('Frequency (Hz)'),ylabel('Magnitude |Y(\omega)| (m/N.s)')
title('Magnitude |Y(\omega)| (m/N.s) Vs Frequency')
grid
hold
subplot(2,1,2),plot(w,phase)
xlabel('Frequency (Hz)'),ylabel('Angle (degree)')
gridd
hold

A.3 Solution of normal modes of a structure

modalsummation.m (main file)
******************************************************************************
% Function:     modalsummation
% Description:  Modal analysis solver of the Flat Shell Finite Element Program
******************************************************************************
function[Nmode,Aomega,node,k,m,f,u] = modalsummation(node, element, xforce, yforce, zforce,...
xmoment, zmoment, matri]
[node, element, adofx, pdofx, udofx, adofy, pdofy, udofy, ...
adofz, pdofz, udofz, adofxa, pdofxa, udofxa, adofza, pdofza, udofza, ...
adof, pdof, udof] = DOFmap(node, element);
[k, u, f,m]=PlateAssembleGlobalDyM(node, element, matri, xforce, ...
yforce, zmoment, matri, xforce, ...
yforce, zmoment, xmoment, adofx, pdofx, ...
udofx, adofy, pdofy, udofy, adofz, pdofz, udofz, adofxa, pdofxa, udofxa, ...
pdofxa, udofza, pdofza, udofza, adof, pdof, udof);

[eigenvalues.m
******************************************************************************
% Function:    Eigenvalues
% Description: Matlab M-File to compute mode shapes and natural frequencies
******************************************************************************
function[mode,omega2,omega] = Eigenvalues(m,k)

normal.m
******************************************************************************
% Function:     Amplitude normalization of mode shapes and natural frequencies
% Description:  M-file arrange mode shapes and natural frequencies due to
%                amplitude of natural frequency
******************************************************************************
function[Nmode,Aomega]=normal(mode,omega)
Nmode=zeros(l,l);
for i=1:l
  Nmode(:,i)=mode(:,c(i));
End

modeplot.m

*******************************************************************************
% Function: modeplot
% Description: Matlab M-File to plot the mode shape
*******************************************************************************
function [xx,yy,zz]=modeplot(Aomega,Nmode,node)
  n=length(node);
  null_id=-1;
  i=1;
  for j=1:n-1
    if node(1).x == node(j+1).x
      i=i+1;
    else
      i=1;
    end
    j=n/i;
    index=1;
    x=zeros(i,j);
    y=zeros(i,j);
    z=zeros(i,j);
    for c=1:j
      for r=1:i
        x(r,c)=node(index).x;
        y(r,c)=node(index).y;
        z(r,c)=0;
        index=index+1;
      end
    end
    num=input('Please Enter the i-th mode shape : ');
    xx=zeros(i,j);
    xxx=zeros(n,1);
    for index=1:n
      if node(index).pdofx ~= null_id
        xxx(index)=0;
      else
        xxx(index)=node(index).udofx;
      end
    end
    index=1;
    for c=1:j
      for r=1:i
        if xxx(index)==0;
          xx(r,c)=0;
          index=index+1;
        else
          xx(r,c)=real(Nmode(xxx(index),num));
          index=index+1;
        end
      end
    end
    index=1;
    for c=1:j
      for r=1:i
        if xxx(index)==0;
          yx(r,c)=0;
          index=index+1;
        else
          yx(r,c)=real(Nmode(xxx(index),num));
          index=index+1;
        end
      end
    end
    index=1;
    for c=1:j
      for r=1:i
        if xxx(index)==0;
          yyyy(r,c)=0;
          index=index+1;
        else
          yyyy(r,c)=real(Nmode(xxx(index),num));
          index=index+1;
        end
      end
    end
    end
    end
    end
    end
    end
    end
end
End
if yyy(index)==0;
  yy(r,c)=0;
  index=index+1;
else
  yy(r,c)=real(Nmode(yyy(index),num));
  index=index+1;
end
end
end

zz=zeros(i,j);
zzz=zeros(n,1);
for index=1:n
  if node(index).pdofz ~= null_id
    zzz(index)=0;
  else
    zzz(index)=node(index).udofz;
  end
end
index=1;
for c=1:j
  for r=1:i
    if zzz(index)==0;
      zz(r,c)=0;
      index=index+1;
    else
      zz(r,c)=real(Nmode(zzz(index),num));
      index=index+1;
    end
  end
end
if zz==zeros(i,j)
  for c=1:j
    for r=1:i
      z(r,c)=-1;
    end
  end
  surf(x+xx,y+yy,zz)
  title('Deformation of the plate ')
  xlabel('X (m)'),ylabel('Y (m)'),zlabel('Z (m)')
Appendix B

Matlab source files for the Flat Shell Finite Element Program
(Damped System)

B.1 Dynamic response of a structure

DplateFRF.m (main file)
Dofmap.m*
DPlateAssembleGlobalDyM.m
PlateStaticStiff.m*
PlateElementLengths.m*
mass.m
RodAssembleMatrix.m*
DAssembleMatrix.m
arrange.m
ddisplacementplot.m

B.2 Frequency response function (Receptance & mobility)

Dplatereceptance.m
Dplatemobility.m

B.3 Solution of normal modes of a structure

Dplatemodalsummation.m (main file)
Dofmap.m*
DPlateAssembleGlobalDyM.m*
PlateStaticStiff.m*
PlateElementLengths.m*
mass.m*
RodAssembleMatrix.m*
DmodalMatrix.m
Deigenvalues.m
normal.m*
dmodeplot.m
B.1 Dynamic response of a structure

DplateFRF.m (main file)

function [u, f, node, k, m] = DplateFRF(node, element, xforce, yforce, zforce,...
        xmoment, zmoment, matrl)

DPlateAssembleGlobalDyM.m

function [k, u, f, m] = DplateAssembleGlobalDyM(node, element, matrl, xforce, ...
        yforce, zforce, xmoment, zmoment, adofx, pdfx, ...
n_elmnt = length(element);
for elmnt_id = 1:n_elmnt
matrl_id = element(elmnt_id).matrl;
[ke] = PlateStaticStiff(element(elmnt_id), node, matrl(matrl_id));
[me] = mass(element(elmnt_id), node, matrl(matrl_id));
[kg] = RodAssembleMatrix( kg, ke, element(elmnt_id) );
[mg] = RodAssembleMatrix( mg, me, element(elmnt_id) );
end
% now partition the stiffness matrix
k.pp = kg(1:pdo.n_dof, 1:pdo.n_dof);                       % extract [kpp]
k.pu = kg(1:pdo.n_dof, pdo.n_dof+1:adof.n_dof);            % extract [kpu]
k.up = kg(pdo.n_dof+1:adof.n_dof, 1:pdo.n_dof);            % extract [kup]
k.uu = kg(pdo.n_dof+1:adof.n_dof, pdo.n_dof+1:adof.n_dof); % extract [kuu]
% now partition the mass matrix
m.pp = mg(1:pdo.n_dof, 1:pdo.n_dof);                       % extract [mpp]
m.pu = mg(1:pdo.n_dof, pdo.n_dof+1:adof.n_dof);            % extract [mpu]
m.up = mg(pdo.n_dof+1:adof.n_dof, 1:pdo.n_dof);            % extract [mup]
m.uu = mg(pdo.n_dof+1:adof.n_dof, pdo.n_dof+1:adof.n_dof); % extract [mpu]

fudofr.n_id=0;
for id = 1: n_node
switch node(id).udof(1)
    case null_id
    fudofr.n_id=fudofr.n_id;
    otherwise
        fudofr.n_id = fudofr.n_id + 1; % increment the # of assembled dof
        uid = node(id).udof(1);
        node_id = udofx.node(uid);
        force_id = node(node_id).xforce;
        if ( force_id == null_id )
            f.Ru(fudofr.n_id) = 0.0; % xforce magn = 0.0 (no load)
        else
            f.Ru(fudofr.n_id) = xforce(force_id).re; % set to specified load
        end
    end
switch node(id).udof(2)
    case null_id
    fudofr.n_id=fudofr.n_id;
    otherwise
        fudofr.n_id = fudofr.n_id + 1; % increment the # of assembled dof
        uid = node(id).udof(2);
        node_id = udofy.node(uid);
        force_id = node(node_id).yforce;
        if ( force_id == null_id )
            f.Ru(fudofr.n_id) = 0.0; % yforce magn = 0.0 (no load)
        else
            f.Ru(fudofr.n_id) = yforce(force_id).re; % set to specified load
        end
    end
switch node(id).udof(3)
    case null_id
    fudofr.n_id=fudofr.n_id;
    otherwise
        fudofr.n_id = fudofr.n_id + 1; % increment the # of assembled dof
        uid = node(id).udof(3);
        node_id = udofxa.node(uid);
        force_id = node(node_id).zforce;
        if ( force_id == null_id )
            f.Ru(fudofr.n_id) = 0.0; % zforce magn = 0.0 (no load)
        else
            f.Ru(fudofr.n_id) = zforce(force_id).re; % set to specified load
        end
    end
switch node(id).udof(4)
    case null_id
    fudofr.n_id=fudofr.n_id;
    otherwise
        fudofr.n_id = fudofr.n_id + 1; % increment the # of assembled dof
        uid = node(id).udof(4);
        node_id = udofxa.node(uid);
        force_id = node(node_id).xmoment;
        if ( force_id == null_id )
f.Ru(fudofr.n_id) = 0.0;                        % moment magn = 0.0 (no load)
else
f.Ru(fudofr.n_id) = xmoment(force_id).re;     % set to specified moment
end
end

switch node(id).udof(5)
    case null_id
        fudofr.n_id = fudofr.n_id;
    otherwise
        fudofr.n_id = fudofr.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(5);
        node_id = udofza.node(uid);
        force_id = node(node_id).zmoment;
        if ( force_id == null_id )
            f.Ru(fudofr.n_id) = 0.0;                        % moment magn = 0.0 (no load)
        else
            f.Ru(fudofr.n_id) = zmoment(force_id).re;     % set to specified moment
        end
    end
end
fudofr.n_id=0;
for id = 1: n_node
    switch node(id).udof(1)
        case null_id
            fudof.n_id=fudof.n_id;
        otherwise
            fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
            uid = node(id).udof(1);
            node_id = udofx.node(uid);
            force_id = node(node_id).xforce;
            if ( force_id == null_id )
                f.Iu(fudof.n_id) = 0.0;                        % xforce magn = 0.0 (no load)
            else
                f.Iu(fudof.n_id) = xforce(force_id).imag;      % set to specified load
            end
        end
    end
switch node(id).udof(2)
    case null_id
        fudof.n_id = fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(2);
        node_id = udofy.node(uid);
        force_id = node(node_id).yforce;
        if ( force_id == null_id )
            f.Iu(fudof.n_id) = 0.0;                        % yforce magn = 0.0 (no load)
        else
            f.Iu(fudof.n_id) = yforce(force_id).imag;      % set to specified load
        end
    end
switch node(id).udof(3)
    case null_id
        fudof.n_id = fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(3);
        node_id = udofz.node(uid);
        force_id = node(node_id).zforce;
        if ( force_id == null_id )
            f.Iu(fudof.n_id) = 0.0;                        % zforce magn = 0.0 (no load)
        else
            f.Iu(fudof.n_id) = zforce(force_id).imag;      % set to specified load
        end
    end
switch node(id).udof(4)
    case null_id
        fudof.n_id = fudof.n_id;
    otherwise
        fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
        uid = node(id).udof(4);
        node_id = udofxa.node(uid);
force_id = node(node_id).xmoment;
if ( force_id == null_id )
  f.Iu(fudof.n_id) = 0.0;                        % moment magn = 0.0 (no load)
else
  f.Iu(fudof.n_id) = xmoment(force_id).imag;     % set to specified moment
end
end
switch node(id).udof(5)
case null_id
  fudof.n_id = fudof.n_id;
otherwise
  fudof.n_id = fudof.n_id + 1;                   % increment the # of assembled dof
  uid = node(id).udof(5);
  node_id = udfzfa.node(uid);
  force_id = node(node_id).zmoment;
  if ( force_id == null_id )
    f.Iu(fudof.n_id) = 0.0;                        % moment magn = 0.0 (no load)
  else
    f.Iu(fudof.n_id) = zmoment(force_id).imag;     % set to specified moment
  end
end
end

DAssembleMatrix.m

function [u,f,KD] = DAssembleMatrix(u,f,k,m,w);
% Real and Imag part for the displacement
u.u = [u.Ru;u.Iu];
% Real and Imag part for the forcing vector
f.u = [f.Ru;f.Iu];
% Structural damping factor
a = 0.01;
% Build the matrix KDpp
KdppR = k.pp - (w^2)*m.pp;
KdppI = k.pp*a;
KDppR = [KdppR,KdppI];
KDppI = [-KdppI,KdppR];
KD.pp = [KDppR,KDppI];
% Build the matrix KDpu
KdpuR = k.pu - (w^2)*m.pu;
KdpuI = k.pu*a;
KDpuR = [KdpuR,KdpuI];
KDpuI = [-KdpuI,KdpuR];
KD.pu = [KDpuR,KDpuI];
% Build the matrix KDup
KdupR = k.up - (w^2)*m.up;
KdupI = k.up*a;
KdupR = [KdupR,KdupI];
KdupI = [-KdupI,KdupR];
KD.up = [KdupR,KdupI];
% Build the matrix KDuu
KduuR = k.uu - (w^2)*m.uu;
KduuI = k.uu*a;
KduuR = [KduuR,KduuuI];
KduuI = [-KduuuI,KduuR];
KD.uu = [KduuuR,KduuuI];

arrange.m

function arrange;
% Matlab M-File to assemble the imaginary and real component of output

143
function[u,f]=arrange(u,f)
a=size(f.u);
b=a(1);
c=b/2;
for i=1:c
f.Ru(i)=f.u(i);
end
for j=1:c
f.Iu(j)=f.u(c+j);
end
a=size(f.p);
b=a(1);
c=b/2;
for i=1:c
f.Rp(i)=f.p(i);
end
for j=1:c
f.Ip(j)=f.p(c+j);
end
a=size(u.u);
b=a(1);
c=b/2;
for i=1:c
u.Ru(i)=u.u(i);
end
for j=1:c
u.Iu(j)=u.u(c+j);
end
a=size(u.p);
b=a(1);
c=b/2;
for i=1:c
u.Rp(i)=u.p(i);
end
for j=1:c
u.Ip(j)=u.p(c+j);
end

ddisplacementplot.m
%******************************************************************************************
%    Function:      ddisplacementplot
%    Description:   Matlab M-File to plot the displacement of nodes of damped flat shell element
%******************************************************************************************
function[xdr,ydr,zdr,xdi,ydi,zdi]=ddisplacementplot(node,u)
null_id=-1;
n=length(node);
for k=1:n
if node(k).pdofx==null_id
    index=node(k).udofx;
    xr(k)=u.u(index);
    xi(k)=u.u(index+length(u.u)/2);
    xm(k)=sqrt(xr(k)^2+xi(k)^2);
    xa(k)=180*angle(xr(k)+xi(k)*i)/pi;
else
    xr(k)=0;
    xi(k)=0;
    xm(k)=0;
    xa(k)=0;
end
end
for k=1:n
if node(k).pdofy==null_id
    index=node(k).udofy;
    yr(k)=u.u(index);
    yi(k)=u.u(index+length(u.u)/2);
    ym(k)=sqrt(yr(k)^2+yi(k)^2);
    ya(k)=180*angle(yr(k)+yi(k)*i)/pi;
else
    yr(k)=0;
    yi(k)=0;
    ym(k)=0;
    ya(k)=0;
end
end
yr(k)=0;
yi(k)=0;
ym(k)=0;
ya(k)=0;
end
end
for k=1:n
    if node(k).pdofz==null_id
        index=node(k).udofz;
        zr(k)=u.u(index);
        zi(k)=u.u(index+length(u.u)/2);
        zm(k)=sqrt(zr(k)^2+zi(k)^2);
        za(k)=180*angle(zr(k)+zi(k)*i)/pi;
    else
        zr(k)=0;
        zi(k)=0;
        zm(k)=0;
        za(k)=0;
    end
end
n=length(node);
null_id=-1;
k=1;
for j=1:n-1
    if node(1).x == node(j+1).x
        k=k+1;
    else
        k=k;
    end
end
j=n/k;
index=1;
x=zeros(k,j);
y=zeros(k,j);
z=zeros(k,j);
for c=1:j
    for r=1:k
        x(r,c)=node(index).x;
        y(r,c)=node(index).y;
        z(r,c)=0;
        index=index+1;
    end
end
index=1;
xd=zeros(k,j);
yd=zeros(k,j);
zd=zeros(k,j);
for c=1:j
    for r=1:k
        xdr(r,c)=xr(index);
        ydr(r,c)=yr(index);
        zdr(r,c)=zr(index);
        xdi(r,c)=xi(index);
        ydi(r,c)=yi(index);
        zdi(r,c)=zi(index);
        xdm(r,c)=xm(index);
        ydm(r,c)=ym(index);
        zdm(r,c)=zm(index);
        xda(r,c)=xa(index);
        yda(r,c)=ya(index);
        zda(r,c)=za(index);
        index=index+1;
    end
end
subplot(2,2,1),surface(x+xdr,y+ydr,z+zdr),colorbar
title('Real component displacement'),xlabel('X(m)','fontsize',8)
ylabel('Y(m)','fontsize',8),zlabel('Z(m)','fontsize',8)
subplot(2,2,2),surface(x+xdr,y+ydr,z+zdr),colorbar
title('Imaginary component displacement'),xlabel('X(m)','fontsize',8)
ylabel('Y(m)','fontsize',8),zlabel('Z(m)','fontsize',8)
subplot(2,2,3),surface(x+xdr,y+ydr,z+zdr),colorbar
B.2 Frequency response function (mobility)

Dplatemobility.m

%*******************************************************************************
%    Function:    DPlatemobility
%    Description: M-File to compute and plot the mobility of a given node
%*******************************************************************************
function [moo,phase]=DPlatemobility(node,k,m,f,u)
null_id = -1;
% Get the node number from user
noden = input('Please enter the node number : ');
index=input('Please enter the degree of the node:x=1,y=2,z=3,mx=4,mz=5 ');

switch index
    case 1
        pdof.id = node(noden).pdofx;
        if pdof.id ~= null_id;
            disp('Mobility can not be found in this node')
        else
            num=node(noden).udofx;
        end
    case 2
        pdof.id = node(noden).pdofy;
        if pdof.id ~= null_id;
            disp('Mobility can not be found in this node')
        else
            num=node(noden).udofy;
        end
    case 3
        pdof.id = node(noden).pdofz;
        if pdof.id ~= null_id;
            disp('Mobility can not be found in this node')
        else
            num=node(noden).udofz;
        end
    case 4
        pdof.id = node(noden).pdofmx;
        if pdof.id ~= null_id;
            disp('Mobility can not be found in this node')
        else
            num=node(noden).udofmx;
        end
    case 5
        pdof.id = node(noden).pdofmz;
        if pdof.id ~= null_id;
            disp('Mobility can not be found in this node')
        else
            num=node(noden).udofmz;
        end
end

for a=1:3.5:350;
    j=a*6.28;
    index=index+1;
    fr(index)=j;
    w(index)=a;
    KduuR=k.uu-(fr(index)^2)*m.uu;
    KduuI=-k.uu*0.05;
    KDuuR=[KduuR,KduuI];
    KDuuI=[-KduuI,KduuR];
    dis=KD.uu\(f.u);
    ure(index)=dis(num);
    uimag(index)=dis(num+(length(dis))/2);
\( \text{di(index)} = \text{ure(index)} + \text{uimag(index)} \times i; \)
\( \text{fre} = \text{f.u(num)}; \)
\( \text{fimag} = \text{f.u(num+length(dis))/2}; \)
\( \text{force} = \text{fre+fimag*i}; \)
\( \text{mob(index)} = (\{\text{fre} - \text{fimag*i}\} \times \text{di(index)} \times i) / (\{\text{fre} - \text{fimag*i}\} \times \{\text{fre} + \text{fimag*i}\}); \)
\( \text{moo(index)} = \text{abs(mob(index))}; \)
\( \text{phase(index)} = 180 \times \text{angle(mob(index))}/\pi; \)
\end{end}

%Plot the mobility and displacement of the given node
subplot(2,1,1),semilogy(w,moo),grid
xlabel('Frequency (Hz)'),ylabel('Magnitude \|Y(\omega)\|(m/N.s)')
title('Magnitude \|Y(\omega)\|(m/N.s) Vs Frequency')
axis([0 350 10^-8 10^0])
subplot(2,1,2),plot(w,phase),grid
xlabel('Frequency (Hz)'),ylabel('Phase Angle (degree)')
title('Phase Angle Vs Frequency')

% Plot the mobility and displacement of the given node
subplot(2,1,1),semilogy(w,mo),xlabel('Natural Frequency (Hz)'),ylabel('Mobility')
grid
title('Plot of Mobility vs Frequency')
subplot(2,1,2),plot(w,phase),axis([0 15000/(2*3.14) 0 185]),xlabel('Natural Frequency (Hz)'),ylabel('Angle (degree)')
grid
title('Phase Angle vs Frequency')

B.3 Solution of normal modes of a structure

Dplatemodalsummation.m (main file)

\% Function: Dplatemodalsummation
\% Description: Modal summation analysis for the damped flat shell element
\%******************************************************************************
function [Nmode, Aomega, node, k, m, f, u] = Dplatemodalsummation(node, element, matrl, xforce, yforce, zforce, xmoment, zmoment, adofx, pdofx, udofx, adofy, pdofy, udofy, adofz, pdofz, udofz, adofxa, pdofxa, udofxa, adofza, pdofza, udofza, adof, pdof, udof);
% Function: DmodalMatrix
% Description: Matlab M-file to assign the Real and Imaginary
% components of matrices and force vector
%******************************************************************************
function [u, f, KD, MD] = DmodalMatrix(u, f, k, m);[node, element] = DOFmap(node, element);
[k, u, f, m] = DPlateAssembleGlobalDyM(node, element, matrl, xforce, yforce, xmoment, zmoment, adofx, pdofx, udofx, adofy, pdofy, udofy, adofz, pdofz, udofz, adofxa, pdofxa, udofxa, adofza, pdofza, udofza, adof, pdof, udof);
[u, f, KD, MD] = DmodalMatrix(u, f, k, m);

DmodalMatrix.m

% Function: DmodalMatrix
% Description: Matlab M-file to assign the Real and Imaginary
% components of matrices and force vector
%******************************************************************************
function [u, f, KD, MD] = DmodalMatrix(u, f, k, m);
u.u = [u.Ru; u.Iu];
u.p = [u.Rp; u.Ip];
f.u = [f.Ru; f.Iu];
f.p = [f.Rp; f.Ip];
KD.pp = k.pp - k.pp*a;
KDppR = KDppR + KdppI;
KdppI = -KdppI + KdppR;
KD.pp = [KDppR; KdppI];
KDpuR=[KdpuR,KdpuI];
KDpuI=[-KdpuI,KdpuR];
KD.pu=[KDpuR,KdpuI];
% Build the matrix KDpu
KdupR=k.up;
KdupI=-k.up*a;
KDupR=[KdupR,KdupI];
KDupI=[-KdupI,KdupR];
KD.up=[KDupR,KDupI];
% Build the matrix KDuu
KduuR=k.uu;
KduuI=-k.uu*a;
KDuuR=[KduuR,KduuI];
KduuI=[-KduuI,KduuR];
KD.uu=[KDuuR,KDuuI];
% Build the matrix MDpp
mdppR=m.pp;
mdppI=zeros(size(m.pp));
MDppR=[mdppR,mdppI];
MDppI=[-mdppI,mdppR];
MD.pp=[MDppR,MDppI];
% Build the matrix MDpu
mdpuR=m.pu;
mdpuI=zeros(size(m.pu));
MDpuR=[mdpuR,mdpuI];
MDpuI=[-mdpuI,mdpuR];
MD.pu=[MDpuR,MDpuI];
% Build the matrix MDup
mdupR=m.up;
mdupI=zeros(size(m.up));
MDupR=[mdupR,mdupI];
MDupI=[-mdupI,mdupR];
MD.up=[MDupR,MDupI];
% Build the matrix MDuu
mduuR=m.uu;
mduuI=zeros(size(m.uu));
MDuuR=[mduuR,mduuI];
MDuuI=[-mduuI,mduuR];
MD.uu=[MDuuR,MDuuI];

Deigenvalues.m
*******************************************************************************
% Function: DEigenvalues
% Description: Matlab M-File to compute to compute mode shapes and natural frequencies
*******************************************************************************
function [mode,omega2,omega] = DEigenvalues(MD,KD)

mass=MD.uu;                             % init the mass marix
stiff=KD.uu;                            % init the stiffness matrix

[mode,omega2]=eig(stiff,mass);         % calculate eigenvalues and eigenvector
omega=norm(omega);                    % calculate natrual frequencies

dmodeplot.m
*******************************************************************************
% Function: dmodeplot
% Description: Matlab M-File to plot the displacement of mode shape
*******************************************************************************
n=length(node);
null_id=-1;
i=1;
for j=1:n-1
if node(1).x == node(j+1).x
i=i+1;
else
i=1;
end
j=n/i;
index=1;
x=zeros(i,j);
y=zeros(i,j);
z=zeros(i,j);
for c=1:j
for r=1:i
x(r,c)=node(index).x;
y(r,c)=node(index).y;
z(r,c)=0;
end
end
num=input('Please Enter the i-th mode shape : ');
rxx=zeros(i,j);
ixx=zeros(i,j);
xxx=zeros(n,1);
for index=1:n
if node(index).pdofx==null_id
xxx(index)=0;
else
xxx(index)=node(index).udofx;
end
end
index=1;
for c=1:j
for r=1:i
if xxx(index)==0;
rxx(r,c)=0;
ixx(r,c)=0;
index=index+1;
else
samNmode=Nmode(xxx(index),num);
rxx(r,c)=real(samNmode);
ixx(r,c)=imag(samNmode);
index=index+1;
end
end
end
end
end
ryy=zeros(i,j);
iyy=zeros(i,j);
yyy=zeros(n,1);
for index=1:n
if node(index).pdofy==null_id
yyy(index)=0;
else
yyy(index)=node(index).udofy;
end
end
index=1;
for c=1:j
for r=1:i
if yyy(index)==0;
ryy(r,c)=0;
iyy(r,c)=0;
index=index+1;
else
samNmode=Nmode(yyy(index),num);
ryy(r,c)=real(samNmode);
iyy(r,c)=imag(samNmode);
index=index+1;
end
end
end
end
rzz=zeros(i,j);
izz=zeros(i,j);
zzz=zeros(n,1);
for index=1:n
if node(index).pdofz==null_id
zzz(index)=0;
else
zzz(index)=node(index).udofz;
end
end
end

index=1;
for c=1:j
for r=1:i
if zzz(index)==0;
   rzz(r,c)=0;
   izz(r,c)=0;
   index=index+1;
else
   samNmode=Nmode(zzz(index),num);
   rzz(r,c)=real(samNmode);
   izz(r,c)=imag(samNmode);
   index=index+1;
end
end
end
if rzz==zeros(i,j)
   for c=1:j
   for r=1:i
   z(r,c)=-1;
   end
   end
   end
novalue=sqrt((max(max(abs(rzz)))^2+max(max(abs(izz)))^2))
noreal=max(max(abs(rzz)))
nnoimag=max(max(izz))
figure(1)
surf(x+iix,y+iyy,izz/noimag)
title('imag')
figure(2)
surf(x+iix,y+iyy,-rzz/noreal)
title('real')
Appendix C

Matlab source files to determine the velocity on the mid-plane of a structure

C.1 The velocity in the mid plane of a structure (Undamped system)

Platedisplace.m
globalvelocityfield.m
velocityfield.m
NIFunction.m

C.2 The velocity in the mid plane of a structure (Damped system)

Platedisplace.m*
dglobalvelocityfield.m
dvelocityfield.m
NIFunction.m*
C.1 The velocity in the mid plane of a structure (Undamped system)

Platedisplace.m

%******************************************************************************
% Function:       Platedisplace
% Description:    M-file to assign the nodal displacement information to each node
%******************************************************************************
null_id = -1;       % null id
for node_id=1:pdofx.n_dof+udofx.n_dof   % Check for pdof
    pdofx.id = node(node_id).pdofx;
    if pdofx.id ~=null_id
        node(node_id).u=u.p(pdofx.id) ; % Assign the u.p ID to node
    end
end
for node_id=1:pdofx.n_dof+udofx.n_dof   % Check for udof
    udofx.id = node(node_id).udofx ;
    if udofx.id ~=null_id
        node(node_id).u=u.u(udofx.id) ; % Assign the u.u ID to node
    end
end
for node_id=1:pdofy.n_dof+udofy.n_dof   % Check for pdof
    pdofy.id = node(node_id).pdofy;
    if pdofy.id ~=null_id
        node(node_id).v=u.p(pdofy.id) ; % Assign the u.p ID to node
    end
end
for node_id=1:pdofy.n_dof+udofy.n_dof   % Check for udof
    udofy.id = node(node_id).udofy ;
    if udofy.id ~=null_id
        node(node_id).v=u.u(udofy.id) ; % Assign the u.u ID to node
    end
end
for node_id=1:pdofz.n_dof+udofz.n_dof   % Check for pdof
    pdofz.id = node(node_id).pdofz;
    if pdofz.id ~=null_id
        node(node_id).w=u.p(pdofz.id) ; % Assign the u.p ID to node
    end
end
for node_id=1:pdofz.n_dof+udofz.n_dof   % Check for udof
    udofz.id = node(node_id).udofz ;
    if udofz.id ~=null_id
        node(node_id).w=u.u(udofz.id) ; % Assign the u.u ID to node
    end
end
for node_id=1:pdofxa.n_dof+udofxa.n_dof % Check for pdof
    pdofxa.id = node(node_id).pdofxa;
    if pdofxa.id ~=null_id
        node(node_id).xa=u.p(pdofxa.id) ; % Assign the u.p ID to node
    end
end
for node_id=1:pdofxa.n_dof+udofxa.n_dof % Check for udof
    udofxa.id = node(node_id).udofxa ;
    if udofxa.id ~=null_id
        node(node_id).xa=u.u(udofxa.id) ; % Assign the u.u ID to node
    end
end
for node_id=1:pdofza.n_dof+udofza.n_dof % Check for pdof
    pdofza.id = node(node_id).pdofza;
    if pdofza.id ~=null_id
        node(node_id).za=u.p(pdofza.id) ; % Assign the u.p ID to node
    end
end
for node_id=1:pdofza.n_dof+udofza.n_dof % Check for udof
    udofza.id = node(node_id).udofza ;
    if udofza.id ~=null_id
        node(node_id).za=u.u(udofza.id) ; % Assign the u.u ID to node
    end
end
globalvelocityfield.m
% Function: globalvelocityfield
% Description: M-file to build the global velocity field on the mid-plane of
% the flat shell element
nxvelocity=zeros(length(element)*25,1);
nyvelocity=zeros(length(element)*25,1);
nzvelocity=zeros(length(element)*25,1);
xindex=zeros(length(element)*25,1);
yindex=zeros(length(element)*25,1);

n_elmnt = length( element );
for elmnt_id = 1:n_elmnt
    [nxvelocity,nyvelocity,nzvelocity,...
        xindex,yindex] = velocityfield(element(elmnt_id),node,nxvelocity,...
        nyvelocity,nzvelocity,xindex,yindex,elmnt_id);
end
n=length(node);
null_id=-1;
i=1;
for j=1:n-1
    if node(1).x == node(j+1).x
        i=i+1;
    else
        i=i;
    end
end
j=n/i;
subindex=1;
index=1;
nxd=zeros(5*(i-2+1),5*(j-2+1));
nyd=zeros(5*(i-2+1),5*(j-2+1));
nzd=zeros(5*(i-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
    for r=1:5:5*(i-2+1)
        index=1+25*(subindex-1);
        nxd(r,c)=nxvelocity(index);
        nxd(r+1,c)=nxvelocity(index+1);
        nxd(r+2,c)=nxvelocity(index+2);
        nxd(r+3,c)=nxvelocity(index+3);
        nxd(r+4,c)=nxvelocity(index+4);
        nxd(r,c+1)=nxvelocity(index+5);
        nxd(r+1,c+1)=nxvelocity(index+6);
        nxd(r+2,c+1)=nxvelocity(index+7);
        nxd(r+3,c+1)=nxvelocity(index+8);
        nxd(r+4,c+1)=nxvelocity(index+9);
        nxd(r,c+2)=nxvelocity(index+10);
        nxd(r+1,c+2)=nxvelocity(index+11);
        nxd(r+2,c+2)=nxvelocity(index+12);
        nxd(r+3,c+2)=nxvelocity(index+13);
        nxd(r+4,c+2)=nxvelocity(index+14);
        nxd(r,c+3)=nxvelocity(index+15);
        nxd(r+1,c+3)=nxvelocity(index+16);
        nxd(r+2,c+3)=nxvelocity(index+17);
        nxd(r+3,c+3)=nxvelocity(index+18);
        nxd(r+4,c+3)=nxvelocity(index+19);
        nxd(r+1,c+4)=nxvelocity(index+20);
        nxd(r+2,c+4)=nxvelocity(index+21);
        nxd(r+3,c+4)=nxvelocity(index+22);
        nxd(r+4,c+4)=nxvelocity(index+23);
        nyd(r,c)=nyvelocity(index);
        nyd(r+1,c)=nyvelocity(index+1);
        nyd(r+2,c)=nyvelocity(index+2);
        nyd(r+3,c)=nyvelocity(index+3);
        nyd(r+4,c)=nyvelocity(index+4);
        nyd(r,c+1)=nyvelocity(index+5);
        nyd(r+1,c+1)=nyvelocity(index+6);
        nyd(r+2,c+1)=nyvelocity(index+7);
nyd(r+3,c+1)=nyvelocity(index+8);
nyd(r+4,c+1)=nyvelocity(index+9);
nyd(r,c+2)=nyvelocity(index+10);
nyd(r+1,c+2)=nyvelocity(index+11);
nyd(r+2,c+2)=nyvelocity(index+12);
nyd(r+3,c+2)=nyvelocity(index+13);
nyd(r+4,c+2)=nyvelocity(index+14);
nyd(r,c+3)=nyvelocity(index+15);
nyd(r+1,c+3)=nyvelocity(index+16);
nyd(r+2,c+3)=nyvelocity(index+17);
nyd(r+3,c+3)=nyvelocity(index+18);
nyd(r+4,c+3)=nyvelocity(index+19);
nyd(r,c+4)=nyvelocity(index+20);
nyd(r+1,c+4)=nyvelocity(index+21);
nyd(r+2,c+4)=nyvelocity(index+22);
nyd(r+3,c+4)=nyvelocity(index+23);
nyd(r+4,c+4)=nyvelocity(index+24);
nzd(r,c)=nzvelocity(index);
nzd(r+1,c)=nzvelocity(index+1);
nzd(r+2,c)=nzvelocity(index+2);
nzd(r+3,c)=nzvelocity(index+3);
nzd(r+4,c)=nzvelocity(index+4);
nzd(r,c+1)=nzvelocity(index+5);
nzd(r+1,c+1)=nzvelocity(index+6);
nzd(r+2,c+1)=nzvelocity(index+7);
nzd(r+3,c+1)=nzvelocity(index+8);
nzd(r+4,c+1)=nzvelocity(index+9);
nzd(r,c+2)=nzvelocity(index+10);
nzd(r+1,c+2)=nzvelocity(index+11);
nzd(r+2,c+2)=nzvelocity(index+12);
nzd(r+3,c+2)=nzvelocity(index+13);
nzd(r+4,c+2)=nzvelocity(index+14);
nzd(r,c+3)=nzvelocity(index+15);
nzd(r+1,c+3)=nzvelocity(index+16);
nzd(r+2,c+3)=nzvelocity(index+17);
nzd(r+3,c+3)=nzvelocity(index+18);
nzd(r+4,c+3)=nzvelocity(index+19);
nzd(r,c+4)=nzvelocity(index+20);
nzd(r+1,c+4)=nzvelocity(index+21);
nzd(r+2,c+4)=nzvelocity(index+22);
nzd(r+3,c+4)=nzvelocity(index+23);
nzd(r+4,c+4)=nzvelocity(index+24);
subindex=subindex+1;
end
end
subindex=1;
index=1;
xxd=zeros(5*(i-2+1),5*(j-2+1));
yyd=zeros(5*(i-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
for r=1:5:5*(i-2+1)
index=1+25*(subindex-1);
xxd(r,c)=xindex(index);
xxd(r,c+1)=xindex(index+1);
xxd(r,c+2)=xindex(index+2);
xxd(r,c+3)=xindex(index+3);
xxd(r,c+4)=xindex(index+4);
xxd(r+1,c)=xindex(index+5);
xxd(r+1,c+1)=xindex(index+6);
xxd(r+1,c+2)=xindex(index+7);
xxd(r+1,c+3)=xindex(index+8);
xxd(r+1,c+4)=xindex(index+9);
xxd(r+2,c)=xindex(index+10);
xxd(r+2,c+1)=xindex(index+11);
xxd(r+2,c+2)=xindex(index+12);
xxd(r+2,c+3)=xindex(index+13);
xxd(r+2,c+4)=xindex(index+14);
xxd(r+3,c)=xindex(index+15);
xxd(r+3,c+1)=xindex(index+16);
xxd(r+3,c+2)=xindex(index+17);
xxd(r+3,c+3)=xindex(index+18);
xxd(r+3,c+4)=xindex(index+19);
velocityfield.m

function [nxvelocity, nyvelocity, nzvelocity,...
  xindex, yindex] = velocityfield(element, node, nxvelocity,...
  nyvelocity, nzvelocity, xindex, yindex, elmnt_id)

node_id1 = element.node(1);               % get 1st node id
node_id2 = element.node(2);               % get 2nd node id
node_id3 = element.node(3);              % get 3rd node id
node_id4 = element.node(4);              % get 4th node id

a = node(node_id3).x - node(node_id1).x;  % compute length
b = node(node_id2).y - node(node_id1).y;  % compute length

NN1 = NIFunction(0, 1, a, b);
NN2 = NIFunction(0, 0.75, a, b);
NN3 = NIFunction(0, 0.5, a, b);
NN4 = NIFunction(0, 0.25, a, b);
NN5 = NIFunction(0, 0, a, b);
NN6 = NIFunction(0.25, 1, a, b);
NN7 = NIFunction(0.25, 0.75, a, b);
NN8 = NIFunction(0.25, 0.5, a, b);
NN9 = NIFunction(0.25, 0.25, a, b);
NN10 = NIFunction(0.25, 0, a, b);
NN11 = NIFunction(0.5, 1, a, b);
NN12 = NIFunction(0.5, 0.75, a, b);
NN13 = NIFunction(0.5, 0.5, a, b);
NN14 = NIFunction(0.5, 0.25, a, b);
NN15 = NIFunction(0.5, 0, a, b);
NN16 = NIFunction(0.75, 1, a, b);
NN17 = NIFunction(0.75, 0.75, a, b);
NN18 = NIFunction(0.75, 0.5, a, b);
NN19 = NIFunction(0.75, 0.25, a, b);
NN20 = NIFunction(0.75, 0, a, b);
NN21 = NIFunction(1, 1, a, b);
NN22 = NIFunction(1, 0.75, a, b);
NN23 = NIFunction(1, 0.5, a, b);
NN24 = NIFunction(1, 0.25, a, b);
NN25 = NIFunction(1, 0, a, b);
xsubindex1 = node(node_id1).x;
xsubindex2 = a*0.25 + node(node_id1).x;
xsubindex3 = a*0.5 + node(node_id1).x;
xsubindex4 = a*0.75 + node(node_id1).x;
xsubindex5 = node(node_id4).x;
ysubindex1 = node(node_id2).y;
ysubindex2 = b*0.75 + node(node_id1).y;
ysubindex3 = b*0.5 + node(node_id1).y;
ysubindex4 = b*0.25 + node(node_id1).y;
ysubindex5 = node(node_id1).y;
u1 = node(node_id1).u;
u2 = node(node_id2).u;
u3 = node(node_id3).u;
u4 = node(node_id4).u;
v1 = node(node_id1).v;
v2 = node(node_id2).v;
v3 = node(node_id3).v;
v4 = node(node_id4).v;
w1 = node(node_id1).w;
w2 = node(node_id2).w;
w3 = node(node_id3).w;
w4 = node(node_id4).w;
xa1 = node(node_id1).xa;
xa2 = node(node_id2).xa;
xa3 = node(node_id3).xa;
xa4 = node(node_id4).xa;
za1 = node(node_id1).za;
za2 = node(node_id2).za;
za3 = node(node_id3).za;
za4 = node(node_id4).za;
d = [u1, v1, w1, xa1, za1, u2, v2, w2, xa2, za2, u3, v3, w3, xa3, za3, u4, v4, w4, xa4, za4];
dis1 = dis; dis2 = NN2*dis; dis3 = NN3*dis; dis4 = NN4*dis; dis5 = NN5*dis; dis6 = NN6*dis; dis7 = NN7*dis; dis8 = NN8*dis; dis9 = NN9*dis; dis10 = NN10*dis; dis11 = NN11*dis; dis12 = NN12*dis; dis13 = NN13*dis; dis14 = NN14*dis; dis15 = NN15*dis; dis16 = NN16*dis; dis17 = NN17*dis; dis18 = NN18*dis; dis19 = NN19*dis; dis20 = NN20*dis; dis21 = NN21*dis; dis22 = NN22*dis; dis23 = NN23*dis;
dis24=NN24*dis;
dis25=NN25*dis;
xx1=dis1(1);
yy1=dis1(2);
zz1=dis1(3);
xx2=dis2(1);
yy2=dis2(2);
zz2=dis2(3);
xx3=dis3(1);
yy3=dis3(2);
zz3=dis3(3);
xx4=dis4(1);
yy4=dis4(2);
zz4=dis4(3);
xx5=dis5(1);
yy5=dis5(2);
zz5=dis5(3);
xx6=dis6(1);
yy6=dis6(2);
zz6=dis6(3);
xx7=dis7(1);
yy7=dis7(2);
zz7=dis7(3);
xx8=dis8(1);
yy8=dis8(2);
zz8=dis8(3);
xx9=dis9(1);
yy9=dis9(2);
zz9=dis9(3);
xx10=dis10(1);
yy10=dis10(2);
zz10=dis10(3);
xx11=dis11(1);
yy11=dis11(2);
zz11=dis11(3);
xx12=dis12(1);
yy12=dis12(2);
zz12=dis12(3);
xx13=dis13(1);
yy13=dis13(2);
zz13=dis13(3);
xx14=dis14(1);
yy14=dis14(2);
zz14=dis14(3);
xx15=dis15(1);
yy15=dis15(2);
zz15=dis15(3);
xx16=dis16(1);
yy16=dis16(2);
zz16=dis16(3);
xx17=dis17(1);
yy17=dis17(2);
zz17=dis17(3);
xx18=dis18(1);
yy18=dis18(2);
zz18=dis18(3);
xx19=dis19(1);
yy19=dis19(2);
zz19=dis19(3);
xx20=dis20(1);
yy20=dis20(2);
zz20=dis20(3);
xx21=dis21(1);
yy21=dis21(2);
zz21=dis21(3);
xx22=dis22(1);
yy22=dis22(2);
zz22=dis22(3);
xx23=dis23(1);
yy23=dis23(2);
zz23=dis23(3);
xx24 = dis24(1);
yy24 = dis24(2);
zz24 = dis24(3);
xx25 = dis25(1);
yy25 = dis25(2);
zz25 = dis25(3);
nxvelocity(25*(elmnt_id-1)+1) = xx1;
nxvelocity(25*(elmnt_id-1)+2) = xx2;
nxvelocity(25*(elmnt_id-1)+3) = xx3;
nxvelocity(25*(elmnt_id-1)+4) = xx4;
nxvelocity(25*(elmnt_id-1)+5) = xx5;
nxvelocity(25*(elmnt_id-1)+6) = xx6;
nxvelocity(25*(elmnt_id-1)+7) = xx7;
nxvelocity(25*(elmnt_id-1)+8) = xx8;
nxvelocity(25*(elmnt_id-1)+9) = xx9;
nxvelocity(25*(elmnt_id-1)+10) = xx10;
nxvelocity(25*(elmnt_id-1)+11) = xx11;
nxvelocity(25*(elmnt_id-1)+12) = xx12;
nxvelocity(25*(elmnt_id-1)+13) = xx13;
nxvelocity(25*(elmnt_id-1)+14) = xx14;
nxvelocity(25*(elmnt_id-1)+15) = xx15;
nxvelocity(25*(elmnt_id-1)+16) = xx16;
nxvelocity(25*(elmnt_id-1)+17) = xx17;
nxvelocity(25*(elmnt_id-1)+18) = xx18;
nxvelocity(25*(elmnt_id-1)+19) = xx19;
nxvelocity(25*(elmnt_id-1)+20) = xx20;
nxvelocity(25*(elmnt_id-1)+21) = xx21;
nxvelocity(25*(elmnt_id-1)+22) = xx22;
nxvelocity(25*(elmnt_id-1)+23) = xx23;
nxvelocity(25*(elmnt_id-1)+24) = xx24;
nxvelocity(25*(elmnt_id-1)+25) = xx25;
nyvelocity(25*(elmnt_id-1)+1) = yy1;
nyvelocity(25*(elmnt_id-1)+2) = yy2;
nyvelocity(25*(elmnt_id-1)+3) = yy3;
nyvelocity(25*(elmnt_id-1)+4) = yy4;
nyvelocity(25*(elmnt_id-1)+5) = yy5;
nyvelocity(25*(elmnt_id-1)+6) = yy6;
nyvelocity(25*(elmnt_id-1)+7) = yy7;
nyvelocity(25*(elmnt_id-1)+8) = yy8;
nyvelocity(25*(elmnt_id-1)+9) = yy9;
nyvelocity(25*(elmnt_id-1)+10) = yy10;
nyvelocity(25*(elmnt_id-1)+11) = yy11;
nyvelocity(25*(elmnt_id-1)+12) = yy12;
nyvelocity(25*(elmnt_id-1)+13) = yy13;
nyvelocity(25*(elmnt_id-1)+14) = yy14;
nyvelocity(25*(elmnt_id-1)+15) = yy15;
nyvelocity(25*(elmnt_id-1)+16) = yy16;
nyvelocity(25*(elmnt_id-1)+17) = yy17;
nyvelocity(25*(elmnt_id-1)+18) = yy18;
nyvelocity(25*(elmnt_id-1)+19) = yy19;
nyvelocity(25*(elmnt_id-1)+20) = yy20;
nyvelocity(25*(elmnt_id-1)+21) = yy21;
nyvelocity(25*(elmnt_id-1)+22) = yy22;
nyvelocity(25*(elmnt_id-1)+23) = yy23;
nyvelocity(25*(elmnt_id-1)+24) = yy24;
nyvelocity(25*(elmnt_id-1)+25) = yy25;
nzvelocity(25*(elmnt_id-1)+1) = zz1;
nzvelocity(25*(elmnt_id-1)+2) = zz2;
nzvelocity(25*(elmnt_id-1)+3) = zz3;
nzvelocity(25*(elmnt_id-1)+4) = zz4;
nzvelocity(25*(elmnt_id-1)+5) = zz5;
nzvelocity(25*(elmnt_id-1)+6) = zz6;
nzvelocity(25*(elmnt_id-1)+7) = zz7;
nzvelocity(25*(elmnt_id-1)+8) = zz8;
nzvelocity(25*(elmnt_id-1)+9) = zz9;
nzvelocity(25*(elmnt_id-1)+10) = zz10;
nzvelocity(25*(elmnt_id-1)+11) = zz11;
nzvelocity(25*(elmnt_id-1)+12) = zz12;
nzvelocity(25*(elmnt_id-1)+13) = zz13;
nzvelocity(25*(elmnt_id-1)+14) = zz14;
nzvelocity(25*(elmnt_id-1)+15) = zz15;
nzvelocity(25*(elmnt_id-1)+16)=zz16;
nzvelocity(25*(elmnt_id-1)+17)=zz17;
nzvelocity(25*(elmnt_id-1)+18)=zz18;
nzvelocity(25*(elmnt_id-1)+19)=zz19;
nzvelocity(25*(elmnt_id-1)+20)=zz20;
nzvelocity(25*(elmnt_id-1)+21)=zz21;
nzvelocity(25*(elmnt_id-1)+22)=zz22;
nzvelocity(25*(elmnt_id-1)+23)=zz23;
nzvelocity(25*(elmnt_id-1)+24)=zz24;
nzvelocity(25*(elmnt_id-1)+25)=zz25;

xindex(25*(elmnt_id-1)+1)=xsubindex1;
xindex(25*(elmnt_id-1)+2)=xsubindex2;
xindex(25*(elmnt_id-1)+3)=xsubindex3;
xindex(25*(elmnt_id-1)+4)=xsubindex4;
xindex(25*(elmnt_id-1)+5)=xsubindex5;
xindex(25*(elmnt_id-1)+6)=xsubindex1;
xindex(25*(elmnt_id-1)+7)=xsubindex2;
xindex(25*(elmnt_id-1)+8)=xsubindex3;
xindex(25*(elmnt_id-1)+9)=xsubindex4;
xindex(25*(elmnt_id-1)+10)=xsubindex5;
xindex(25*(elmnt_id-1)+11)=xsubindex1;
xindex(25*(elmnt_id-1)+12)=xsubindex2;
xindex(25*(elmnt_id-1)+13)=xsubindex3;
xindex(25*(elmnt_id-1)+14)=xsubindex4;
xindex(25*(elmnt_id-1)+15)=xsubindex5;
xindex(25*(elmnt_id-1)+16)=xsubindex1;
xindex(25*(elmnt_id-1)+17)=xsubindex2;
xindex(25*(elmnt_id-1)+18)=xsubindex3;
xindex(25*(elmnt_id-1)+19)=xsubindex4;
xindex(25*(elmnt_id-1)+20)=xsubindex5;
xindex(25*(elmnt_id-1)+21)=xsubindex1;
xindex(25*(elmnt_id-1)+22)=xsubindex2;
xindex(25*(elmnt_id-1)+23)=xsubindex3;
xindex(25*(elmnt_id-1)+24)=xsubindex4;
xindex(25*(elmnt_id-1)+25)=xsubindex5;

yindex(25*(elmnt_id-1)+1)=ysubindex1;
yindex(25*(elmnt_id-1)+2)=ysubindex2;
yindex(25*(elmnt_id-1)+3)=ysubindex3;
yindex(25*(elmnt_id-1)+4)=ysubindex4;
yindex(25*(elmnt_id-1)+5)=ysubindex5;
yindex(25*(elmnt_id-1)+6)=ysubindex1;
yindex(25*(elmnt_id-1)+7)=ysubindex2;
yindex(25*(elmnt_id-1)+8)=ysubindex3;
yindex(25*(elmnt_id-1)+9)=ysubindex4;
yindex(25*(elmnt_id-1)+10)=ysubindex5;
yindex(25*(elmnt_id-1)+11)=ysubindex1;
yindex(25*(elmnt_id-1)+12)=ysubindex2;
yindex(25*(elmnt_id-1)+13)=ysubindex3;
yindex(25*(elmnt_id-1)+14)=ysubindex4;
yindex(25*(elmnt_id-1)+15)=ysubindex5;
yindex(25*(elmnt_id-1)+16)=ysubindex1;
yindex(25*(elmnt_id-1)+17)=ysubindex2;
yindex(25*(elmnt_id-1)+18)=ysubindex3;
yindex(25*(elmnt_id-1)+19)=ysubindex4;
yindex(25*(elmnt_id-1)+20)=ysubindex5;
yindex(25*(elmnt_id-1)+21)=ysubindex1;
yindex(25*(elmnt_id-1)+22)=ysubindex2;
yindex(25*(elmnt_id-1)+23)=ysubindex3;
yindex(25*(elmnt_id-1)+24)=ysubindex4;
yindex(25*(elmnt_id-1)+25)=ysubindex5;
NIFunction.m

% Function: NIFunction
% Description: M-file to determine the velocity field on the mid-plane of the flat shell element at the laser measurement positions

function [NN]=NIFunction(xx,yy,a,b)
si=xx;
p1=yy;
Np1=(si-1)*(pi-1);
Np2=(1-si)*pi;
Np3=si*pi;
Np4=(1-pi)*si;

Nk1 = 1-si*pi-(3-2*si)*(si^2)*(1-pi)-(1-si)*(3-2*pi)*pi^2;
Nk2 = (1-si)*pi*(1-pi)^2)*b;
Nk3 = -si*(1-si^2)*(1-pi)*a;
Nk4 = (1-si)*(3-2*pi)*pi^2*si*(1-si)*(1-2*si)*pi;
Nk5 = -(1-si)*(1-pi)^2)*b;
Nk6 = -si*(1-si^2)*pi*a;
Nk7 = (3-2*si)*(si^2)*pi*si*(1-pi)*(1-2*pi);
Nk8 = -si*(1-pi)*(1-pi)^2)*b;
Nk9 = -si*(1-si^2)*pi*a;
Nk10 = (3-2*si)*(si^2)*(1-pi)+si*pi*(1-pi)*(1-2*pi);
Nk11 = si*pi*(1-pi)^2)*b;
Nk12 = -si*(1-pi)^2)*a;
NN=[Np1,0,0,0,0,Np2,0,0,0,0,Np3,0,0,0,0,Np4,0,0,0,0,;...
    0,Np1,0,0,0,0,Np2,0,0,0,0,Np3,0,0,0,0,Np4,0,0,0,;...
    0,0,Nk1,Nk2,Nk3,0,0,Nk4,Nk5,Nk6,0,0,Nk7,Nk8,Nk9,0,0,Nk10,Nk11,Nk12];

C.2 The velocity in the mid plane of a structure (Damped system)

dglobalverlfield.m

% Function: dglobalverlfield
% Description: M-file to build the global velocity field on the mid-plane of the damped flat shell element

Rsxvelocity=zeros(length(element)*25,1);
Rsyvelocity=zeros(length(element)*25,1);
Rszvelocity=zeros(length(element)*25,1);

Isxvelocity=zeros(length(element)*25,1);
Isyvelocity=zeros(length(element)*25,1);
Iszvelocity=zeros(length(element)*25,1);

xindex=zeros(length(element)*25,1);
yindex=zeros(length(element)*25,1);

n_elmnt = length( element );
for elmnt_id = 1:n_elmnt
    [Rsxvelocity,Rsyvelocity,Rszvelocity,Isxvelocity,Isyvelocity,Iszvelocity,
     xindex,yindex]=dvelocityfield(element(elmnt_id),node,Rsxvelocity,...
     Rsyvelocity,Rszvelocity,Isxvelocity,Isyvelocity,Iszvelocity,xindex,yindex,elmnt_id);
end

for c=1:5:(k-2+1)
    Rnxd=zeros(5*(k-2+1),5*(k-2+1));
    Rnyd=zeros(5*(k-2+1),5*(k-2+1));
    Rnzd=zeros(5*(k-2+1),5*(k-2+1));
for c=1:5:*j-2+1)
for r=1:5:5*(k-2+1)
    index=1+25*(subindex-1);
    Rnxd(r,c)=Rsxvelocity(index);
    Rnxd(r+1,c)=Rsxvelocity(index+1);
    Rnxd(r+2,c)=Rsxvelocity(index+2);
    Rnxd(r+3,c)=Rsxvelocity(index+3);
    Rnxd(r+4,c)=Rsxvelocity(index+4);
    Rnxd(r,c+1)=Rsxvelocity(index+5);
    Rnxd(r+1,c+1)=Rsxvelocity(index+6);
    Rnxd(r+2,c+1)=Rsxvelocity(index+7);
    Rnxd(r+3,c+1)=Rsxvelocity(index+8);
    Rnxd(r+4,c+1)=Rsxvelocity(index+9);
    Rnxd(r,c+2)=Rsxvelocity(index+10);
    Rnxd(r+1,c+2)=Rsxvelocity(index+11);
    Rnxd(r+2,c+2)=Rsxvelocity(index+12);
    Rnxd(r+3,c+2)=Rsxvelocity(index+13);
    Rnxd(r+4,c+2)=Rsxvelocity(index+14);
    Rnxd(r,c+3)=Rsxvelocity(index+15);
    Rnxd(r+1,c+3)=Rsxvelocity(index+16);
    Rnxd(r+2,c+3)=Rsxvelocity(index+17);
    Rnxd(r+3,c+3)=Rsxvelocity(index+18);
    Rnxd(r+4,c+3)=Rsxvelocity(index+19);
    Rnxd(r,c+4)=Rsxvelocity(index+20);
    Rnxd(r+1,c+4)=Rsxvelocity(index+21);
    Rnxd(r+2,c+4)=Rsxvelocity(index+22);
    Rnxd(r+3,c+4)=Rsxvelocity(index+23);
    Rnxd(r+4,c+4)=Rsxvelocity(index+24);
    Rnyd(r,c)=Rsyvelocity(index);
    Rnyd(r+1,c)=Rsyvelocity(index+1);
    Rnyd(r+2,c)=Rsyvelocity(index+2);
    Rnyd(r+3,c)=Rsyvelocity(index+3);
    Rnyd(r+4,c)=Rsyvelocity(index+4);
    Rnyd(r,c+1)=Rsyvelocity(index+5);
    Rnyd(r+1,c+1)=Rsyvelocity(index+6);
    Rnyd(r+2,c+1)=Rsyvelocity(index+7);
    Rnyd(r+3,c+1)=Rsyvelocity(index+8);
    Rnyd(r+4,c+1)=Rsyvelocity(index+9);
    Rnyd(r+1,c+2)=Rsyvelocity(index+10);
    Rnyd(r+2,c+2)=Rsyvelocity(index+11);
    Rnyd(r+3,c+2)=Rsyvelocity(index+12);
    Rnyd(r+4,c+2)=Rsyvelocity(index+13);
    Rnyd(r,c+2)=Rsyvelocity(index+14);
    Rnyd(r+1,c+2)=Rsyvelocity(index+15);
    Rnyd(r+2,c+2)=Rsyvelocity(index+16);
    Rnyd(r+3,c+2)=Rsyvelocity(index+17);
    Rnyd(r+4,c+2)=Rsyvelocity(index+18);
    Rnyd(r,c+3)=Rsyvelocity(index+19);
    Rnyd(r+1,c+3)=Rsyvelocity(index+20);
    Rnyd(r+2,c+3)=Rsyvelocity(index+21);
    Rnyd(r+3,c+3)=Rsyvelocity(index+22);
    Rnyd(r+4,c+3)=Rsyvelocity(index+23);
    Rnyd(r,c+4)=Rsyvelocity(index+24);
    Rnzd(r,c)=Rszvelocity(index);
    Rnzd(r+1,c)=Rszvelocity(index+1);
    Rnzd(r+2,c)=Rszvelocity(index+2);
    Rnzd(r+3,c)=Rszvelocity(index+3);
    Rnzd(r+4,c)=Rszvelocity(index+4);
    Rnzd(r,c+1)=Rszvelocity(index+5);
    Rnzd(r+1,c+1)=Rszvelocity(index+6);
    Rnzd(r+2,c+1)=Rszvelocity(index+7);
    Rnzd(r+3,c+1)=Rszvelocity(index+8);
    Rnzd(r+4,c+1)=Rszvelocity(index+9);
    Rnzd(r+1,c+2)=Rszvelocity(index+10);
    Rnzd(r+2,c+2)=Rszvelocity(index+11);
    Rnzd(r+3,c+2)=Rszvelocity(index+12);
    Rnzd(r+4,c+2)=Rszvelocity(index+13);
    Rnzd(r,c+2)=Rszvelocity(index+14);
    Rnzd(r+1,c+2)=Rszvelocity(index+15);
    Rnzd(r+2,c+2)=Rszvelocity(index+16);
    Rnzd(r+3,c+2)=Rszvelocity(index+17);
    Rnzd(r+4,c+2)=Rszvelocity(index+18);
Inxd = zeros(5*(k-2+1), 5*(k-2+1));
Inyd = zeros(5*(k-2+1), 5*(k-2+1));
Inzd = zeros(5*(k-2+1), 5*(k-2+1));
for r=1:5:5*(j-2+1)
    index = 1 + 25*(subindex - 1);
    Inxd(r,c) = Isxvelocity(index);
    Inxd(r+1,c) = Isxvelocity(index+1);
    Inxd(r+2,c) = Isxvelocity(index+2);
    Inxd(r+3,c) = Isxvelocity(index+3);
    Inxd(r+4,c) = Isxvelocity(index+4);
    Inxd(r+1,c+1) = Isxvelocity(index+5);
    Inxd(r+2,c+1) = Isxvelocity(index+6);
    Inxd(r+3,c+1) = Isxvelocity(index+7);
    Inxd(r+4,c+1) = Isxvelocity(index+8);
    Inxd(r+1,c+2) = Isxvelocity(index+9);
    Inxd(r+2,c+2) = Isxvelocity(index+10);
    Inxd(r+3,c+2) = Isxvelocity(index+11);
    Inxd(r+4,c+2) = Isxvelocity(index+12);
    Inxd(r+1,c+3) = Isxvelocity(index+13);
    Inxd(r+2,c+3) = Isxvelocity(index+14);
    Inxd(r+3,c+3) = Isxvelocity(index+15);
    Inxd(r+4,c+3) = Isxvelocity(index+16);
    Inxd(r+1,c+4) = Isxvelocity(index+17);
    Inxd(r+2,c+4) = Isxvelocity(index+18);
    Inxd(r+3,c+4) = Isxvelocity(index+19);
    Inxd(r+4,c+4) = Isxvelocity(index+20);
    Inxd(r+1,c+5) = Isxvelocity(index+21);
    Inxd(r+2,c+5) = Isxvelocity(index+22);
    Inxd(r+3,c+5) = Isxvelocity(index+23);
    Inxd(r+4,c+5) = Isxvelocity(index+24);
end
Inyd = zeros(5*(k-2+1), 5*(k-2+1));
for c=1:5:5*(j-2+1)
    index = 1 + 25*(subindex - 1);
    Inyd(r,c) = Isyvelocity(index);
    Inyd(r+1,c) = Isyvelocity(index+1);
    Inyd(r+2,c) = Isyvelocity(index+2);
    Inyd(r+3,c) = Isyvelocity(index+3);
    Inyd(r+4,c) = Isyvelocity(index+4);
    Inyd(r,c+1) = Isyvelocity(index+5);
    Inyd(r+1,c+1) = Isyvelocity(index+6);
    Inyd(r+2,c+1) = Isyvelocity(index+7);
    Inyd(r+3,c+1) = Isyvelocity(index+8);
    Inyd(r+4,c+1) = Isyvelocity(index+9);
    Inyd(r,c+2) = Isyvelocity(index+10);
    Inyd(r+1,c+2) = Isyvelocity(index+11);
    Inyd(r+2,c+2) = Isyvelocity(index+12);
    Inyd(r+3,c+2) = Isyvelocity(index+13);
    Inyd(r+4,c+2) = Isyvelocity(index+14);
    Inyd(r,c+3) = Isyvelocity(index+15);
    Inyd(r+1,c+3) = Isyvelocity(index+16);
    Inyd(r+2,c+3) = Isyvelocity(index+17);
    Inyd(r+3,c+3) = Isyvelocity(index+18);
    Inyd(r+4,c+3) = Isyvelocity(index+19);
    Inyd(r,c+4) = Isyvelocity(index+20);
    Inyd(r+1,c+4) = Isyvelocity(index+21);
    Inyd(r+2,c+4) = Isyvelocity(index+22);
    Inyd(r+3,c+4) = Isyvelocity(index+23);
    Inyd(r+4,c+4) = Isyvelocity(index+24);
end
Inzd = zeros(5*(k-2+1), 5*(k-2+1));
for r=1:5:5*(j-2+1)
    index = 1 + 25*(subindex - 1);
    Inzd(r,c) = Iszvelocity(index);
    Inzd(r+1,c) = Iszvelocity(index+1);
    Inzd(r+2,c) = Iszvelocity(index+2);
    Inzd(r+3,c) = Iszvelocity(index+3);
    Inzd(r+4,c) = Iszvelocity(index+4);
end

Rnzd(r+4,c+3) = Rszvelocity(index+18);
Rnzd(r,c+4) = Rszvelocity(index+20);
Rnzd(r+1,c+4) = Rszvelocity(index+21);
Rnzd(r+2,c+4) = Rszvelocity(index+22);
Rnzd(r+3,c+4) = Rszvelocity(index+23);
Rnzd(r+4,c+4) = Rszvelocity(index+24);
subindex = subindex + 1;
end

inxd = zeros(5*(k-2+1), 5*(k-2+1));
Inzd(r+4,c)=Iszvelocity(index+4);
Inzd(r,c+1)=Iszvelocity(index+5);
Inzd(r+1,c+1)=Iszvelocity(index+6);
Inzd(r+2,c+1)=Iszvelocity(index+7);
Inzd(r+3,c+1)=Iszvelocity(index+8);
Inzd(r+4,c+1)=Iszvelocity(index+9);
Inzd(r,c+2)=Iszvelocity(index+10);
Inzd(r+1,c+2)=Iszvelocity(index+11);
Inzd(r+2,c+2)=Iszvelocity(index+12);
Inzd(r+3,c+2)=Iszvelocity(index+13);
Inzd(r+4,c+2)=Iszvelocity(index+14);
Inzd(r,c+3)=Iszvelocity(index+15);
Inzd(r+1,c+3)=Iszvelocity(index+16);
Inzd(r+2,c+3)=Iszvelocity(index+17);
Inzd(r+3,c+3)=Iszvelocity(index+18);
Inzd(r+4,c+3)=Iszvelocity(index+19);
Inzd(r,c+4)=Iszvelocity(index+20);
Inzd(r+1,c+4)=Iszvelocity(index+21);
Inzd(r+2,c+4)=Iszvelocity(index+22);
Inzd(r+3,c+4)=Iszvelocity(index+23);
Inzd(r+4,c+4)=Iszvelocity(index+24);
subindex=subindex+1;
end
subindex=1;
index=1;
xxd=zeros(5*(k-2+1),5*(j-2+1));
yyd=zeros(5*(k-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
  for r=1:5:5*(k-2+1)
    index=1+25*(subindex-1);
    xxd(r,c)=xindex(index);
    xxd(r,c+1)=xindex(index+1);
    xxd(r,c+2)=xindex(index+2);
    xxd(r,c+3)=xindex(index+3);
    xxd(r,c+4)=xindex(index+4);
    xxd(r+1,c)=xindex(index+5);
    xxd(r+1,c+1)=xindex(index+6);
    xxd(r+1,c+2)=xindex(index+7);
    xxd(r+1,c+3)=xindex(index+8);
    xxd(r+1,c+4)=xindex(index+9);
    xxd(r+2,c)=xindex(index+10);
    xxd(r+2,c+1)=xindex(index+11);
    xxd(r+2,c+2)=xindex(index+12);
    xxd(r+2,c+3)=xindex(index+13);
    xxd(r+2,c+4)=xindex(index+14);
    xxd(r+3,c)=xindex(index+15);
    xxd(r+3,c+1)=xindex(index+16);
    xxd(r+3,c+2)=xindex(index+17);
    xxd(r+3,c+3)=xindex(index+18);
    xxd(r+3,c+4)=xindex(index+19);
    xxd(r+4,c)=xindex(index+20);
    xxd(r+4,c+1)=xindex(index+21);
    xxd(r+4,c+2)=xindex(index+22);
    xxd(r+4,c+3)=xindex(index+23);
    xxd(r+4,c+4)=xindex(index+24);
    yyd(r,c)=yindex(index);
    yyd(r+1,c)=yindex(index+1);
    yyd(r+2,c)=yindex(index+2);
    yyd(r+3,c)=yindex(index+3);
    yyd(r+4,c)=yindex(index+4);
    yyd(r,c+1)=yindex(index+5);
    yyd(r+1,c+1)=yindex(index+6);
    yyd(r+2,c+1)=yindex(index+7);
    yyd(r+3,c+1)=yindex(index+8);
    yyd(r+4,c+1)=yindex(index+9);
    yyd(r+1,c+2)=yindex(index+10);
    yyd(r+2,c+2)=yindex(index+11);
    yyd(r+3,c+2)=yindex(index+12);
    yyd(r+4,c+2)=yindex(index+13);
    yyd(r+4,c+3)=yindex(index+14);
yyd(r,c+3)=yindex(index+15);
yyd(r+1,c+3)=yindex(index+16);
yyd(r+2,c+3)=yindex(index+17);
yyd(r+3,c+3)=yindex(index+18);
yyd(r+4,c+3)=yindex(index+19);
yyd(r,c+4)=yindex(index+20);
yyd(r+1,c+4)=yindex(index+21);
yyd(r+2,c+4)=yindex(index+22);
yyd(r+3,c+4)=yindex(index+23);
yyd(r+4,c+4)=yindex(index+24);
subindex=subindex+1;
end
end
subplot(2,2,1),surface(xxd,yyd,Rnzd*3*2*pi),xlabel('X(m)'),'fontsize',8)
ylabel('Y(m)'),'fontsize',8)
title('Real velocity on the mid plane','fontsize',8)
colorbar, axis equal
subplot(2,2,2),surface(xxd,yyd,Inzd*3*2*pi),xlabel('X(m)'),'fontsize',8)
ylabel('Y(m)'),'fontsize',8)
title('Imaginary velocity on the mid plane','fontsize',8)
colorbar, axis equal
subplot(2,2,3),surface(xxd,yyd,abs(3*2*pi*Rnzd+3*2*i*pi*Inzd)),xlabel('X(m)'),'fontsize',8)
ylabel('Y(m)'),'fontsize',8)
title('Magnitude velocity on the mid plane','fontsize',8)
colorbar, axis equal
subplot(2,2,4),surface(xxd,yyd,180*angle(3*2*pi*i*Rnzd-3*2*pi*Inzd)),xlabel('X(m)'),'fontsize',8)
ylabel('Y(m)'),'fontsize',8)
title('Relative phase angle of velocity on the mid plane','fontsize',8)
colorbar, axis equal

dvelocityfield.m

% Function: dvelocityfield
% Description: M-file to determine the velocity field on the mid-plane of
% the damped flat shell element
%*****************************************************************************
function[Rxvelocity,Ryvelocity,Rzvelocity, Isxvelocity,Isyvelocity,Iszvelocity,...
xindex,yindex]=dvelocityfield(element,node,Rxvelocity,...
Ryvelocity,Rzvelocity,Isxvelocity,Isyvelocity,Iszvelocity,xindex,yindex,elmnt_id)
node_id1 = element.node(1);                   % get 1st node id
node_id2 = element.node(2);                   % get 2nd node id
node_id3 = element.node(3);
node_id4 = element.node(4);
a = node(node_id3).x - node(node_id1).x;   % compute length
b = node(node_id2).y - node(node_id1).y;
NN1=NIFunction(0,1,a,b);
NN2=NIFunction(0,0.75,a,b);
NN3=NIFunction(0,0.5,a,b);
NN4=NIFunction(0,0.25,a,b);
NN5=NIFunction(0.25,1,a,b);
NN6=NIFunction(0.25,0.75,a,b);
NN7=NIFunction(0.25,0.5,a,b);
NN8=NIFunction(0.25,0.25,a,b);
NN9=NIFunction(0.25,0,a,b);
NN10=NIFunction(0.5,1,a,b);
NN11=NIFunction(0.5,0.75,a,b);
NN12=NIFunction(0.5,0.5,a,b);
NN13=NIFunction(0.5,0.25,a,b);
NN14=NIFunction(0.5,0.0,a,b);
NN15=NIFunction(0.0,1,a,b);
NN16=NIFunction(0.0,0.75,a,b);
NN17=NIFunction(0.0,0.5,a,b);
NN18=NIFunction(0.0,0.25,a,b);
NN19=NIFunction(0.0,a,b);
NN20=NIFunction(1,1,a,b);
NN21=NIFunction(1,0.75,a,b);
NN22=NIFunction(1,0.5,a,b);
NN23=NIFunction(1,0.25,a,b);

NN25=NIFunction(1,0,a,b);
xsubindex1=node(node_id1).x;
xsubindex2=a*0.25+node(node_id1).x;
xsubindex3=a*0.5+node(node_id1).x;
xsubindex4=a*0.75+node(node_id1).x;
xsubindex5=node(node_id4).x;
ysubindex1=node(node_id1).y;
ysubindex2=b*0.75+node(node_id1).y;
ysubindex3=b*0.5+node(node_id1).y;
ysubindex4=b*0.25+node(node_id1).y;
ysubindex5=node(node_id1).y;
Ru1=node(node_id1).Ru;
Ru2=node(node_id2).Ru;
Ru3=node(node_id3).Ru;
Ru4=node(node_id4).Ru;
Rv1=node(node_id1).Rv;
Rv2=node(node_id2).Rv;
Rv3=node(node_id3).Rv;
Rv4=node(node_id4).Rv;
Rw1=node(node_id1).Rw;
Rw2=node(node_id2).Rw;
Rw3=node(node_id3).Rw;
Rw4=node(node_id4).Rw;
Rxa1=node(node_id1).Rxa;
Rxa2=node(node_id2).Rxa;
Rxa3=node(node_id3).Rxa;
Rxa4=node(node_id4).Rxa;
Rza1=node(node_id1).Rza;
Rza2=node(node_id2).Rza;
Rza3=node(node_id3).Rza;
Rza4=node(node_id4).Rza;
Rdis=[Ru1,Rv1,Rw1,Rxa1,Rza1,Ru2,Rv2,Rw2,Rxa2,Rza2,Ru3,Rv3,Rw3,Rxa3,Rza3,Ru4,Rv4,Rw4,Rxa4,Rza4]';
Iu1=node(node_id1).Iu;
Iu2=node(node_id2).Iu;
Iu3=node(node_id3).Iu;
Iu4=node(node_id4).Iu;
 Iv1=node(node_id1).Iv;
 Iv2=node(node_id2).Iv;
 Iv3=node(node_id3).Iv;
 Iv4=node(node_id4).Iv;
 Iw1=node(node_id1).Iw;
 Iw2=node(node_id2).Iw;
 Iw3=node(node_id3).Iw;
 Iw4=node(node_id4).Iw;
 Ixa1=node(node_id1).Ixa;
 Ixa2=node(node_id2).Ixa;
 Ixa3=node(node_id3).Ixa;
 Ixa4=node(node_id4).Ixa;
 Iza1=node(node_id1).Iza;
 Iza2=node(node_id2).Iza;
 Iza3=node(node_id3).Iza;
 Iza4=node(node_id4).Iza;
Idis=[Iu1,Iv1,Iw1,Ixa1,Iza1,Iu2,Iv2,Iw2,Ixa2,Iza2,Iu3,Iv3,Iw3,Ixa3,Iza3,Iu4,Iv4,Iw4,Ixa4,Iza4]';
Rdis1=NN1*Rdis;
Rdis2=NN2*Rdis;
Rdis3=NN3*Rdis;
Rdis4=NN4*Rdis;
Rdis5=NN5*Rdis;
Rdis6=NN6*Rdis;
Rdis7=NN7*Rdis;
Rdis8=NN8*Rdis;
Rdis9=NN9*Rdis;
Rdis10=NN10*Rdis;
Rdis11=NN11*Rdis;
Rdis12=NN12*Rdis;
Rdis13=NN13*Rdis;
Rdis14=NN14*Rdis;
Rdis15=NN15*Rdis;
Rdis16=NN16*Rdis;
Rdis17=NN17*Rdis;
Rdis18=NN18*Rdis;
Rdis19=NN19*Rdis;
Rdis20=NN20*Rdis;
Rdis21=NN21*Rdis;
Rdis22=NN22*Rdis;
Rdis23=NN23*Rdis;
Rdis24=NN24*Rdis;
Rdis25=NN25*Rdis;
Rxx1=Rdis1(1);
Ryy1=Rdis1(2);
Rzz1=Rdis1(3);
Rxx2=Rdis2(1);
Ryy2=Rdis2(2);
Rzz2=Rdis2(3);
Rxx3=Rdis3(1);
Ryy3=Rdis3(2);
Rzz3=Rdis3(3);
Rxx4=Rdis4(1);
Ryy4=Rdis4(2);
Rzz4=Rdis4(3);
Rxx5=Rdis5(1);
Ryy5=Rdis5(2);
Rzz5=Rdis5(3);
Rxx6=Rdis6(1);
Ryy6=Rdis6(2);
Rzz6=Rdis6(3);
Rxx7=Rdis7(1);
Ryy7=Rdis7(2);
Rzz7=Rdis7(3);
Rxx8=Rdis8(1);
Ryy8=Rdis8(2);
Rzz8=Rdis8(3);
Rxx9=Rdis9(1);
Ryy9=Rdis9(2);
Rzz9=Rdis9(3);
Rxx10=Rdis10(1);
Ryy10=Rdis10(2);
Rzz10=Rdis10(3);
Rxx11=Rdis11(1);
Ryy11=Rdis11(2);
Rzz11=Rdis11(3);
Rxx12=Rdis12(1);
Ryy12=Rdis12(2);
Rzz12=Rdis12(3);
Rxx13=Rdis13(1);
Ryy13=Rdis13(2);
Rzz13=Rdis13(3);
Rxx14=Rdis14(1);
Ryy14=Rdis14(2);
Rzz14=Rdis14(3);
Rxx15=Rdis15(1);
Ryy15=Rdis15(2);
Rzz15=Rdis15(3);
Rxx16=Rdis16(1);
Ryy16=Rdis16(2);
Rzz16=Rdis16(3);
Rxx17=Rdis17(1);
Ryy17=Rdis17(2);
Rzz17=Rdis17(3);
Rxx18=Rdis18(1);
Ryy18=Rdis18(2);
Rzz18=Rdis18(3);
Rxx19=Rdis19(1);
Ryy19=Rdis19(2);
Rzz19=Rdis19(3);
Rxx20=Rdis20(1);
Ryy20=Rdis20(2);
Rzz20=Rdis20(3);
Rxx21=Rdis21(1);
Ryy21=Rdis21(2);
Rzz21=Rdis21(3);
Rxx22=Rdis22(1);
Ryy22 = Rdis22(2);
Rzz22 = Rdis22(3);
Rxx23 = Rdis23(1);
Ryy23 = Rdis23(2);
Rzz23 = Rdis23(3);
Rxx24 = Rdis24(1);
Ryy24 = Rdis24(2);
Rzz24 = Rdis24(3);
Rxx25 = Rdis25(1);
Ryy25 = Rdis25(2);
Rzz25 = Rdis25(3);
Rsxvelocity(25*(elmnt_id-1)+1) = Rxx1;
Rsxvelocity(25*(elmnt_id-1)+2) = Rxx2;
Rsxvelocity(25*(elmnt_id-1)+3) = Rxx3;
Rsxvelocity(25*(elmnt_id-1)+4) = Rxx4;
Rsxvelocity(25*(elmnt_id-1)+5) = Rxx5;
Rsxvelocity(25*(elmnt_id-1)+6) = Rxx6;
Rsxvelocity(25*(elmnt_id-1)+7) = Rxx7;
Rsxvelocity(25*(elmnt_id-1)+8) = Rxx8;
Rsxvelocity(25*(elmnt_id-1)+9) = Rxx9;
Rsxvelocity(25*(elmnt_id-1)+10) = Rxx10;
Rsxvelocity(25*(elmnt_id-1)+11) = Rxx11;
Rsxvelocity(25*(elmnt_id-1)+12) = Rxx12;
Rsxvelocity(25*(elmnt_id-1)+13) = Rxx13;
Rsxvelocity(25*(elmnt_id-1)+14) = Rxx14;
Rsxvelocity(25*(elmnt_id-1)+15) = Rxx15;
Rsxvelocity(25*(elmnt_id-1)+16) = Rxx16;
Rsxvelocity(25*(elmnt_id-1)+17) = Rxx17;
Rsxvelocity(25*(elmnt_id-1)+18) = Rxx18;
Rsxvelocity(25*(elmnt_id-1)+19) = Rxx19;
Rsxvelocity(25*(elmnt_id-1)+20) = Rxx20;
Rsxvelocity(25*(elmnt_id-1)+21) = Rxx21;
Rsxvelocity(25*(elmnt_id-1)+22) = Rxx22;
Rsxvelocity(25*(elmnt_id-1)+23) = Rxx23;
Rsxvelocity(25*(elmnt_id-1)+24) = Rxx24;
Rsxvelocity(25*(elmnt_id-1)+25) = Rxx25;
Rsyvelocity(25*(elmnt_id-1)+1) = Ryy1;
Rsyvelocity(25*(elmnt_id-1)+2) = Ryy2;
Rsyvelocity(25*(elmnt_id-1)+3) = Ryy3;
Rsyvelocity(25*(elmnt_id-1)+4) = Ryy4;
Rsyvelocity(25*(elmnt_id-1)+5) = Ryy5;
Rsyvelocity(25*(elmnt_id-1)+6) = Ryy6;
Rsyvelocity(25*(elmnt_id-1)+7) = Ryy7;
Rsyvelocity(25*(elmnt_id-1)+8) = Ryy8;
Rsyvelocity(25*(elmnt_id-1)+9) = Ryy9;
Rsyvelocity(25*(elmnt_id-1)+10) = Ryy10;
Rsyvelocity(25*(elmnt_id-1)+11) = Ryy11;
Rsyvelocity(25*(elmnt_id-1)+12) = Ryy12;
Rsyvelocity(25*(elmnt_id-1)+13) = Ryy13;
Rsyvelocity(25*(elmnt_id-1)+14) = Ryy14;
Rsyvelocity(25*(elmnt_id-1)+15) = Ryy15;
Rsyvelocity(25*(elmnt_id-1)+16) = Ryy16;
Rsyvelocity(25*(elmnt_id-1)+17) = Ryy17;
Rsyvelocity(25*(elmnt_id-1)+18) = Ryy18;
Rsyvelocity(25*(elmnt_id-1)+19) = Ryy19;
Rsyvelocity(25*(elmnt_id-1)+20) = Ryy20;
Rsyvelocity(25*(elmnt_id-1)+21) = Ryy21;
Rsyvelocity(25*(elmnt_id-1)+22) = Ryy22;
Rsyvelocity(25*(elmnt_id-1)+23) = Ryy23;
Rsyvelocity(25*(elmnt_id-1)+24) = Ryy24;
Rsyvelocity(25*(elmnt_id-1)+25) = Ryy25;
Rszvelocity(25*(elmnt_id-1)+1) = Rzz1;
Rszvelocity(25*(elmnt_id-1)+2) = Rzz2;
Rszvelocity(25*(elmnt_id-1)+3) = Rzz3;
Rszvelocity(25*(elmnt_id-1)+4) = Rzz4;
Rszvelocity(25*(elmnt_id-1)+5) = Rzz5;
Rszvelocity(25*(elmnt_id-1)+6) = Rzz6;
Rszvelocity(25*(elmnt_id-1)+7) = Rzz7;
Rszvelocity(25*(elmnt_id-1)+8) = Rzz8;
Rszvelocity(25*(elmnt_id-1)+9) = Rzz9;
Rszvelocity(25*(elmnt_id-1)+10) = Rzz10;
Rszvelocity(25*(elmnt_id-1)+11)=Rzz11;
Rszvelocity(25*(elmnt_id-1)+12)=Rzz12;
Rszvelocity(25*(elmnt_id-1)+13)=Rzz13;
Rszvelocity(25*(elmnt_id-1)+14)=Rzz14;
Rszvelocity(25*(elmnt_id-1)+15)=Rzz15;
Rszvelocity(25*(elmnt_id-1)+16)=Rzz16;
Rszvelocity(25*(elmnt_id-1)+17)=Rzz17;
Rszvelocity(25*(elmnt_id-1)+18)=Rzz18;
Rszvelocity(25*(elmnt_id-1)+19)=Rzz19;
Rszvelocity(25*(elmnt_id-1)+20)=Rzz20;
Rszvelocity(25*(elmnt_id-1)+21)=Rzz21;
Rszvelocity(25*(elmnt_id-1)+22)=Rzz22;
Rszvelocity(25*(elmnt_id-1)+23)=Rzz23;
Rszvelocity(25*(elmnt_id-1)+24)=Rzz24;
Rszvelocity(25*(elmnt_id-1)+25)=Rzz25;
Idis1=NN1*Idis;
Idis2=NN2*Idis;
Idis3=NN3*Idis;
Idis4=NN4*Idis;
Idis5=NN5*Idis;
Idis6=NN6*Idis;
Idis7=NN7*Idis;
Idis8=NN8*Idis;
Idis9=NN9*Idis;
Idis10=NN10*Idis;
Idis11=NN11*Idis;
Idis12=NN12*Idis;
Idis13=NN13*Idis;
Idis14=NN14*Idis;
Idis15=NN15*Idis;
Idis16=NN16*Idis;
Idis17=NN17*Idis;
Idis18=NN18*Idis;
Idis19=NN19*Idis;
Idis20=NN20*Idis;
Idis21=NN21*Idis;
Idis22=NN22*Idis;
Idis23=NN23*Idis;
Idis24=NN24*Idis;
Idis25=NN25*Idis;
Ixx1=Idis1(1);
Iyy1=Idis1(2);
Izz1=Idis1(3);
Ixx2=Idis2(1);
Iyy2=Idis2(2);
Izz2=Idis2(3);
Ixx3=Idis3(1);
Iyy3=Idis3(2);
Izz3=Idis3(3);
Ixx4=Idis4(1);
Iyy4=Idis4(2);
Izz4=Idis4(3);
Ixx5=Idis5(1);
Iyy5=Idis5(2);
Izz5=Idis5(3);
Ixx6=Idis6(1);
Iyy6=Idis6(2);
Izz6=Idis6(3);
Ixx7=Idis7(1);
Iyy7=Idis7(2);
Izz7=Idis7(3);
Ixx8=Idis8(1);
Iyy8=Idis8(2);
Izz8=Idis8(3);
Ixx9=Idis9(1);
Iyy9=Idis9(2);
Izz9=Idis9(3);
Ixx10=Idis10(1);
Iyy10=Idis10(2);
Izz10=Idis10(3);
Ixx11=Idis11(1);
Iyy1=Idis11(2);
Ixz1=Idis11(3);
Ixx1=Idis12(1);
Iyy2=Idis12(2);
Izz2=Idis12(3);
Ixx3=Idis13(1);
Iyy3=Idis13(2);
Izz3=Idis13(3);
Ixx4=Idis14(1);
Iyy4=Idis14(2);
Izz4=Idis14(3);
Ixx5=Idis15(1);
Iyy5=Idis15(2);
Izz5=Idis15(3);
Ixx6=Idis16(1);
Iyy6=Idis16(2);
Izz6=Idis16(3);
Ixx7=Idis17(1);
Iyy7=Idis17(2);
Izz7=Idis17(3);
Ixx8=Idis18(1);
Iyy8=Idis18(2);
Izz8=Idis18(3);
Ixx9=Idis19(1);
Iyy9=Idis19(2);
Izz9=Idis19(3);
Ixx10=Idis20(1);
Iyy10=Idis20(2);
Izz10=Idis20(3);
Ixx11=Idis21(1);
Iyy11=Idis21(2);
Ixx12=Idis21(3);
Ixx13=Idis22(1);
Iyy13=Idis22(2);
Izz13=Idis22(3);
Ixx14=Idis23(1);
Iyy14=Idis23(2);
Izz14=Idis23(3);
Ixx15=Idis24(1);
Iyy15=Idis24(2);
Izz15=Idis24(3);
Ixx16=Idis25(1);
Iyy16=Idis25(2);
Izz16=Idis25(3);
Isxvelocity(25*(elmnt_id-1)+1)=Ixx1;
Isxvelocity(25*(elmnt_id-1)+2)=Ixx2;
Isxvelocity(25*(elmnt_id-1)+3)=Ixx3;
Isxvelocity(25*(elmnt_id-1)+4)=Ixx4;
Isxvelocity(25*(elmnt_id-1)+5)=Ixx5;
Isxvelocity(25*(elmnt_id-1)+6)=Ixx6;
Isxvelocity(25*(elmnt_id-1)+7)=Ixx7;
Isxvelocity(25*(elmnt_id-1)+8)=Ixx8;
Isxvelocity(25*(elmnt_id-1)+9)=Ixx9;
Isxvelocity(25*(elmnt_id-1)+10)=Ixx10;
Isxvelocity(25*(elmnt_id-1)+11)=Ixx11;
Isxvelocity(25*(elmnt_id-1)+12)=Ixx12;
Isxvelocity(25*(elmnt_id-1)+13)=Ixx13;
Isxvelocity(25*(elmnt_id-1)+14)=Ixx14;
Isxvelocity(25*(elmnt_id-1)+15)=Ixx15;
Isxvelocity(25*(elmnt_id-1)+16)=Ixx16;
Isxvelocity(25*(elmnt_id-1)+17)=Ixx17;
Isxvelocity(25*(elmnt_id-1)+18)=Ixx18;
Isxvelocity(25*(elmnt_id-1)+19)=Ixx19;
Isxvelocity(25*(elmnt_id-1)+20)=Ixx20;
Isxvelocity(25*(elmnt_id-1)+21)=Ixx21;
Isxvelocity(25*(elmnt_id-1)+22)=Ixx22;
Isxvelocity(25*(elmnt_id-1)+23)=Ixx23;
Isxvelocity(25*(elmnt_id-1)+24)=Ixx24;
Isxvelocity(25*(elmnt_id-1)+25)=Ixx25;
Isyvelocity(25*(elmnt_id-1)+1)=Iyy1;
Isyvelocity(25*(elmnt_id-1)+2)=Iyy2;
Isyvelocity(25*(elmnt_id-1)+3)=Iyy3;
Isyvelocity(25*(elmnt_id-1)+4)=Iyy4;
Isyvelocity(25*(elmnt_id-1)+5)=Iyy5;
Isyvelocity(25*(elmnt_id-1)+6)=Iyy6;
Isyvelocity(25*(elmnt_id-1)+7)=Iyy7;
Isyvelocity(25*(elmnt_id-1)+8)=Iyy8;
Isyvelocity(25*(elmnt_id-1)+9)=Iyy9;
Isyvelocity(25*(elmnt_id-1)+10)=Iyy10;
Isyvelocity(25*(elmnt_id-1)+11)=Iyy11;
Isyvelocity(25*(elmnt_id-1)+12)=Iyy12;
Isyvelocity(25*(elmnt_id-1)+13)=Iyy13;
Isyvelocity(25*(elmnt_id-1)+14)=Iyy14;
Isyvelocity(25*(elmnt_id-1)+15)=Iyy15;
Isyvelocity(25*(elmnt_id-1)+16)=Iyy16;
Isyvelocity(25*(elmnt_id-1)+17)=Iyy17;
Isyvelocity(25*(elmnt_id-1)+18)=Iyy18;
Isyvelocity(25*(elmnt_id-1)+19)=Iyy19;
Isyvelocity(25*(elmnt_id-1)+20)=Iyy20;
Isyvelocity(25*(elmnt_id-1)+21)=Iyy21;
Isyvelocity(25*(elmnt_id-1)+22)=Iyy22;
Isyvelocity(25*(elmnt_id-1)+23)=Iyy23;
Isyvelocity(25*(elmnt_id-1)+24)=Iyy24;
Isyvelocity(25*(elmnt_id-1)+25)=Iyy25;
Iszvelocity(25*(elmnt_id-1)+1)=Izz1;
Iszvelocity(25*(elmnt_id-1)+2)=Izz2;
Iszvelocity(25*(elmnt_id-1)+3)=Izz3;
Iszvelocity(25*(elmnt_id-1)+4)=Izz4;
Iszvelocity(25*(elmnt_id-1)+5)=Izz5;
Iszvelocity(25*(elmnt_id-1)+6)=Izz6;
Iszvelocity(25*(elmnt_id-1)+7)=Izz7;
Iszvelocity(25*(elmnt_id-1)+8)=Izz8;
Iszvelocity(25*(elmnt_id-1)+9)=Izz9;
Iszvelocity(25*(elmnt_id-1)+10)=Izz10;
Iszvelocity(25*(elmnt_id-1)+11)=Izz11;
Iszvelocity(25*(elmnt_id-1)+12)=Izz12;
Iszvelocity(25*(elmnt_id-1)+13)=Izz13;
Iszvelocity(25*(elmnt_id-1)+14)=Izz14;
Iszvelocity(25*(elmnt_id-1)+15)=Izz15;
Iszvelocity(25*(elmnt_id-1)+16)=Izz16;
Iszvelocity(25*(elmnt_id-1)+17)=Izz17;
Iszvelocity(25*(elmnt_id-1)+18)=Izz18;
Iszvelocity(25*(elmnt_id-1)+19)=Izz19;
Iszvelocity(25*(elmnt_id-1)+20)=Izz20;
Iszvelocity(25*(elmnt_id-1)+21)=Izz21;
Iszvelocity(25*(elmnt_id-1)+22)=Izz22;
Iszvelocity(25*(elmnt_id-1)+23)=Izz23;
Iszvelocity(25*(elmnt_id-1)+24)=Izz24;
Iszvelocity(25*(elmnt_id-1)+25)=Izz25;
xindex(25*(elmnt_id-1)+1)=xsubindex1;
xindex(25*(elmnt_id-1)+2)=xsubindex2;
xindex(25*(elmnt_id-1)+3)=xsubindex3;
xindex(25*(elmnt_id-1)+4)=xsubindex4;
xindex(25*(elmnt_id-1)+5)=xsubindex5;
xindex(25*(elmnt_id-1)+6)=xsubindex1;
xindex(25*(elmnt_id-1)+7)=xsubindex2;
xindex(25*(elmnt_id-1)+8)=xsubindex3;
xindex(25*(elmnt_id-1)+9)=xsubindex4;
xindex(25*(elmnt_id-1)+10)=xsubindex5;
xindex(25*(elmnt_id-1)+11)=xsubindex1;
xindex(25*(elmnt_id-1)+12)=xsubindex2;
xindex(25*(elmnt_id-1)+13)=xsubindex3;
xindex(25*(elmnt_id-1)+14)=xsubindex4;
xindex(25*(elmnt_id-1)+15)=xsubindex5;
xindex(25*(elmnt_id-1)+16)=xsubindex1;
xindex(25*(elmnt_id-1)+17)=xsubindex2;
xindex(25*(elmnt_id-1)+18)=xsubindex3;
xindex(25*(elmnt_id-1)+19)=xsubindex4;
xindex(25*(elmnt_id-1)+20)=xsubindex5;
xindex(25*(elmnt_id-1)+21)=xsubindex1;
xindex(25*(elmnt_id-1)+22)=xsubindex2;
xindex(25*(elmnt_id-1)+23)=xsubindex3;
xindex(25*(elmnt_id-1)+24)=xsubindex4;
xindex(25*(elmnt_id-1)+25)=xsubindex5;
yindex(25*(elmnt_id-1)+1)=ysubindex1;
yindex(25*(elmnt_id-1)+2)=ysubindex2;
yindex(25*(elmnt_id-1)+3)=ysubindex3;
yindex(25*(elmnt_id-1)+4)=ysubindex4;
yindex(25*(elmnt_id-1)+5)=ysubindex5;
yindex(25*(elmnt_id-1)+6)=ysubindex1;
yindex(25*(elmnt_id-1)+7)=ysubindex2;
yindex(25*(elmnt_id-1)+8)=ysubindex3;
yindex(25*(elmnt_id-1)+9)=ysubindex4;
yindex(25*(elmnt_id-1)+10)=ysubindex5;
yindex(25*(elmnt_id-1)+11)=ysubindex1;
yindex(25*(elmnt_id-1)+12)=ysubindex2;
yindex(25*(elmnt_id-1)+13)=ysubindex3;
yindex(25*(elmnt_id-1)+14)=ysubindex4;
yindex(25*(elmnt_id-1)+15)=ysubindex5;
yindex(25*(elmnt_id-1)+16)=ysubindex1;
yindex(25*(elmnt_id-1)+17)=ysubindex2;
yindex(25*(elmnt_id-1)+18)=ysubindex3;
yindex(25*(elmnt_id-1)+19)=ysubindex4;
yindex(25*(elmnt_id-1)+20)=ysubindex5;
yindex(25*(elmnt_id-1)+21)=ysubindex1;
yindex(25*(elmnt_id-1)+22)=ysubindex2;
yindex(25*(elmnt_id-1)+23)=ysubindex3;
yindex(25*(elmnt_id-1)+24)=ysubindex4;
yindex(25*(elmnt_id-1)+25)=ysubindex5;
Appendix D

Matlab source files for the velocity transformation function of the flat shell element

D.1 The velocity transformation function of the flat shell element (Undamped system)

Platedisplace.m*
surfglobalvelfield.m
survelocityfield.m
SIFunction.m

D.2 The velocity transformation function of the flat shell element (Damped system)

Platedisplace.m*
dsurfglobalvelfield.m
dsurvelocityfield.m
SIFunction.m*
D.1 The velocity transformation function of the flat shell element (Undamped system)

surfglobalvelfield.m

% Function: surfglobalvelfield
% Description: M-file to build the global velocity field on the surface of the
% flat shell element

sxvelocity=zeros(length(element)*25,1);
syvelocity=zeros(length(element)*25,1);
szvelocity=zeros(length(element)*25,1);
xindex=zeros(length(element)*25,1);
yindex=zeros(length(element)*25,1);

n_elmnt = length(element);
for elmnt_id = 1:n_elmnt
    [sxvelocity,syvelocity,szvelocity,...
xindex,yindex]=survelocityfield(element(elmnt_id),node,sxvelocity,...
syvelocity,szvelocity,xindex,yindex,elmnt_id);
end

n=length(node);
null_id=-1;
i=1;
for j=1:n-1
    if node(1).x == node(j+1).x
        i=i+1;
    else
        i=i;
    end
end
j=n/i;
subindex=1;

sxd=zeros(5*(i-2+1),5*(j-2+1));
syd=zeros(5*(i-2+1),5*(j-2+1));
szd=zeros(5*(i-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
    for r=1:5:5*(i-2+1)
        index=1+25*(subindex-1);
sxd(r,c)=sxvelocity(index);
sxd(r+1,c)=sxvelocity(index+1);
sxd(r+2,c)=sxvelocity(index+2);
sxd(r+3,c)=sxvelocity(index+3);
sxd(r+4,c)=sxvelocity(index+4);
sxd(r+1,c+1)=sxvelocity(index+5);
sxd(r+2,c+1)=sxvelocity(index+6);
sxd(r+3,c+1)=sxvelocity(index+7);
sxd(r+4,c+1)=sxvelocity(index+8);
sxd(r+1,c+2)=sxvelocity(index+9);
sxd(r+2,c+2)=sxvelocity(index+10);
sxd(r+1,c+3)=sxvelocity(index+11);
sxd(r+2,c+3)=sxvelocity(index+12);
sxd(r+3,c+3)=sxvelocity(index+13);
sxd(r+4,c+3)=sxvelocity(index+14);
sxd(r+1,c+4)=sxvelocity(index+15);
sxd(r+2,c+4)=sxvelocity(index+16);
sxd(r+3,c+4)=sxvelocity(index+17);
sxd(r+4,c+4)=sxvelocity(index+18);
syd(r,c)=syvelocity(index);
syd(r+1,c)=syvelocity(index+1);
syd(r+2,c)=syvelocity(index+2);
syd(r+3,c)=syvelocity(index+3);
syd(r+4,c)=syvelocity(index+4);
syd(r+1,c+1)=syvelocity(index+5);
syd(r+2,c+1)=syvelocity(index+6);
syd(r+3,c+1)=syvelocity(index+7);
syd(r+4,c+1)=syvelocity(index+8);
syd(r+1,c+2)=syvelocity(index+9);
syd(r+2,c+2)=syvelocity(index+10);
syd(r+3,c+2)=syvelocity(index+11);
syd(r+4,c+2)=syvelocity(index+12);
syd(r+1,c+3)=syvelocity(index+13);
syd(r+2,c+3)=syvelocity(index+14);
syd(r+3,c+3)=syvelocity(index+15);
syd(r+4,c+3)=syvelocity(index+16);
syd(r+1,c+4)=syvelocity(index+17);
syd(r+2,c+4)=syvelocity(index+18);
syd(r+3,c+4)=syvelocity(index+19);
syd(r+4,c+4)=syvelocity(index+20);
szd(r,c)=szvelocity(index);
szd(r+1,c)=szvelocity(index+1);
szd(r+2,c)=szvelocity(index+2);
szd(r+3,c)=szvelocity(index+3);
szd(r+4,c)=szvelocity(index+4);
szd(r+1,c+1)=szvelocity(index+5);
szd(r+2,c+1)=szvelocity(index+6);
szd(r+3,c+1)=szvelocity(index+7);
szd(r+4,c+1)=szvelocity(index+8);
szd(r+1,c+2)=szvelocity(index+9);
szd(r+2,c+2)=szvelocity(index+10);
szd(r+3,c+2)=szvelocity(index+11);
szd(r+4,c+2)=szvelocity(index+12);
szd(r+1,c+3)=szvelocity(index+13);
szd(r+2,c+3)=szvelocity(index+14);
szd(r+3,c+3)=szvelocity(index+15);
szd(r+4,c+3)=szvelocity(index+16);
szd(r+1,c+4)=szvelocity(index+17);
szd(r+2,c+4)=szvelocity(index+18);
szd(r+3,c+4)=szvelocity(index+19);
szd(r+4,c+4)=szvelocity(index+20);
end
end
syd(r+1,c+1)=syvelocity(index+6);
syd(r+2,c+1)=syvelocity(index+7);
syd(r+3,c+1)=syvelocity(index+8);
syd(r+4,c+1)=syvelocity(index+9);
syd(r,c+2)=syvelocity(index+10);
syd(r+1,c+2)=syvelocity(index+11);
syd(r+2,c+2)=syvelocity(index+12);
syd(r+3,c+2)=syvelocity(index+13);
syd(r+4,c+2)=syvelocity(index+14);
syd(r,c+3)=syvelocity(index+15);
syd(r+1,c+3)=syvelocity(index+16);
syd(r+2,c+3)=syvelocity(index+17);
syd(r+3,c+3)=syvelocity(index+18);
syd(r+4,c+3)=syvelocity(index+19);
syd(r,c+4)=syvelocity(index+20);
syd(r+1,c+4)=syvelocity(index+21);
syd(r+2,c+4)=syvelocity(index+22);
syd(r+3,c+4)=syvelocity(index+23);
syd(r+4,c+4)=syvelocity(index+24);
szd(r,c)=szvelocity(index);
szd(r+1,c)=szvelocity(index+1);
szd(r+2,c)=szvelocity(index+2);
szd(r+3,c)=szvelocity(index+3);
szd(r+4,c)=szvelocity(index+4);
szd(r,c+1)=szvelocity(index+5);
szd(r+1,c+1)=szvelocity(index+6);
szd(r+2,c+1)=szvelocity(index+7);
szd(r+3,c+1)=szvelocity(index+8);
szd(r+4,c+1)=szvelocity(index+9);
szd(r,c+2)=szvelocity(index+10);
szd(r+1,c+2)=szvelocity(index+11);
szd(r+2,c+2)=szvelocity(index+12);
szd(r+3,c+2)=szvelocity(index+13);
szd(r+4,c+2)=szvelocity(index+14);
szd(r,c+3)=szvelocity(index+15);
szd(r+1,c+3)=szvelocity(index+16);
szd(r+2,c+3)=szvelocity(index+17);
szd(r+3,c+3)=szvelocity(index+18);
szd(r+4,c+3)=szvelocity(index+19);
szd(r,c+4)=szvelocity(index+20);
szd(r+1,c+4)=szvelocity(index+21);
szd(r+2,c+4)=szvelocity(index+22);
szd(r+3,c+4)=szvelocity(index+23);
szd(r+4,c+4)=szvelocity(index+24);
subindex=subindex+1;
end
end
subindex=1;
index=1;
xxd=zeros(5*(i-2+1),5*(j-2+1));
yyd=zeros(5*(i-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
for r=1:5:5*(i-2+1)
index=1+25*(subindex-1);
xxd(r,c)=xindex(index);
xxd(r,c+1)=xindex(index+1);
xxd(r,c+2)=xindex(index+2);
xxd(r,c+3)=xindex(index+3);
xxd(r,c+4)=xindex(index+4);
xxd(r+1,c)=xindex(index+5);
xxd(r+1,c+1)=xindex(index+6);
xxd(r+1,c+2)=xindex(index+7);
xxd(r+1,c+3)=xindex(index+8);
xxd(r+1,c+4)=xindex(index+9);
xxd(r+2,c)=xindex(index+10);
xxd(r+2,c+1)=xindex(index+11);
xxd(r+2,c+2)=xindex(index+12);
xxd(r+2,c+3)=xindex(index+13);
xxd(r+2,c+4)=xindex(index+14);
xxd(r+3,c)=xindex(index+15);
xxd(r+3,c+1)=xindex(index+16);
xd(r+3,c+2)=xindex(index+17);
xd(r+3,c+3)=xindex(index+18);
xd(r+4,c)=xindex(index+20);
xd(r+4,c+1)=xindex(index+21);
xd(r+4,c+2)=xindex(index+22);
xd(r+4,c+3)=xindex(index+23);
xd(r+4,c+4)=xindex(index+24);
yyd(r,c)=yindex(index);
yyd(r+1,c)=yindex(index+1);
yyd(r+2,c)=yindex(index+2);
yyd(r+3,c)=yindex(index+3);
yyd(r+4,c)=yindex(index+4);
yyd(r+1,c+1)=yindex(index+5);
yyd(r+1,c+2)=yindex(index+6);
yyd(r+2,c+1)=yindex(index+7);
yyd(r+3,c+1)=yindex(index+8);
yyd(r+4,c+1)=yindex(index+9);
yyd(r+1,c+2)=yindex(index+10);
yyd(r+2,c+2)=yindex(index+11);
yyd(r+3,c+2)=yindex(index+12);
yyd(r+4,c+2)=yindex(index+13);
yyd(r+1,c+4)=yindex(index+14);
yyd(r+2,c+4)=yindex(index+15);
yyd(r+3,c+4)=yindex(index+16);
yyd(r+4,c+4)=yindex(index+17);
yyd(r+3,c+3)=yindex(index+18);
yyd(r+4,c+3)=yindex(index+19);
yyd(r+4,c+4)=yindex(index+20);
yyd(r+1,c+4)=yindex(index+21);
yyd(r+2,c+4)=yindex(index+22);
yyd(r+3,c+4)=yindex(index+23);
yyd(r+4,c+4)=yindex(index+24);
subindex=subindex+1;
end
survelocityfield.m
%***********************************************************************************
%  Function:     survelocityfield
%  Description:  M-file to determine the velocity field on the surface of
%                flat shell element
%***********************************************************************************
function[sxvelocity,syvelocity,szvelocity,
        xindex,yindex]=survelocityfield(element,node,sxvelocity,...
        syvelocity,szvelocity,xindex,yindex,elmnt_id)
           node_id1 = element.node(1);               % get 1st node id
           node_id2 = element.node(2);               % get 2nd node id
           node_id3 = element.node(3);
           node_id4 = element.node(4);
           a = node(node_id3).x - node(node_id1).x;
           b = node(node_id2).y - node(node_id1).y;   % compute length
SN1=SIFunction(0,1,a,b,element.tick);
SN2=SIFunction(0,0.75,a,b,element.tick);
SN3=SIFunction(0,0.5,a,b,element.tick);
SN4=SIFunction(0,0.25,a,b,element.tick);
SN5=SIFunction(0,0,a,b,element.tick);
SN6=SIFunction(0.25,1,a,b,element.tick);
SN7=SIFunction(0.25,0.75,a,b,element.tick);
SN8=SIFunction(0.25, 0.5, a, b, element.tick);
SN9=SIFunction(0.25, 0.25, a, b, element.tick);
SN10=SIFunction(0.5, 0, a, b, element.tick);
SN11=SIFunction(0.5, 0.1, a, b, element.tick);
SN12=SIFunction(0.5, 0.75, a, b, element.tick);
SN13=SIFunction(0.5, 0.5, a, b, element.tick);
SN14=SIFunction(0.5, 0.25, a, b, element.tick);
SN15=SIFunction(0.5, 0, a, b, element.tick);
SN16=SIFunction(0.75, 0.75, a, b, element.tick);
SN17=SIFunction(0.75, 0.5, a, b, element.tick);
SN18=SIFunction(0.75, 0.25, a, b, element.tick);
SN19=SIFunction(0.75, 0, a, b, element.tick);
SN20=SIFunction(1, 1, a, b, element.tick);
SN21=SIFunction(1, 0.75, a, b, element.tick);
SN22=SIFunction(1, 0.5, a, b, element.tick);
SN23=SIFunction(1, 0.25, a, b, element.tick);
SN24=SIFunction(1, 0.0, a, b, element.tick);
SN25=SIFunction(1, 0, a, b, element.tick);

xsubindex1=node(node_id1).x;
xsubindex2=a*0.25+node(node_id1).x;
xsubindex3=a*0.5+node(node_id1).x;
xsubindex4=a*0.75+node(node_id1).x;
xsubindex5=node(node_id4).x;
ysubindex1=node(node_id2).y;
ysubindex2=b*0.75+node(node_id1).y;
ysubindex3=b*0.5+node(node_id1).y;
ysubindex4=b*0.25+node(node_id1).y;
ysubindex5=node(node_id1).y;

u1=node(node_id1).u;
u2=node(node_id2).u;
u3=node(node_id3).u;
u4=node(node_id4).u;
v1=node(node_id1).v;
v2=node(node_id2).v;
v3=node(node_id3).v;
v4=node(node_id4).v;
w1=node(node_id1).w;
w2=node(node_id2).w;
w3=node(node_id3).w;
w4=node(node_id4).w;
xa1=node(node_id1).xa;
xa2=node(node_id2).xa;
xa3=node(node_id3).xa;
xa4=node(node_id4).xa;
za1=node(node_id1).za;
za2=node(node_id2).za;
za3=node(node_id3).za;
za4=node(node_id4).za;

dis=[u1,v1,w1,xa1,za1,u2,v2,w2,xa2,za2,u3,v3,w3,xa3,za3,u4,v4,w4,xa4,za4]';
dis1=SN1*dis;
dis2=SN2*dis;
dis3=SN3*dis;
dis4=SN4*dis;
dis5=SN5*dis;
dis6=SN6*dis;
dis7=SN7*dis;
dis8=SN8*dis;
dis9=SN9*dis;
dis10=SN10*dis;
dis11=SN11*dis;
dis12=SN12*dis;
dis13=SN13*dis;
dis14=SN14*dis;
dis15=SN15*dis;
dis16=SN16*dis;
dis17=SN17*dis;
dis18=SN18*dis;
dis19=SN19*dis;
dis20=SN20*dis;
dis21=SN21*dis;
dis22=SN22*dis;
dis23 = SN23 * dis;
dis24 = SN24 * dis;
dis25 = SN25 * dis;
xx1 = dis1(1);
yy1 = dis1(2);
zz1 = dis1(3);
xx2 = dis2(1);
yy2 = dis2(2);
zz2 = dis2(3);
xx3 = dis3(1);
yy3 = dis3(2);
zz3 = dis3(3);
xx4 = dis4(1);
yy4 = dis4(2);
zz4 = dis4(3);
xx5 = dis5(1);
yy5 = dis5(2);
zz5 = dis5(3);
xx6 = dis6(1);
yy6 = dis6(2);
zz6 = dis6(3);
xx7 = dis7(1);
yy7 = dis7(2);
zz7 = dis7(3);
xx8 = dis8(1);
yy8 = dis8(2);
zz8 = dis8(3);
xx9 = dis9(1);
yy9 = dis9(2);
zz9 = dis9(3);
xx10 = dis10(1);
yy10 = dis10(2);
zz10 = dis10(3);
xx11 = dis11(1);
yy11 = dis11(2);
zz11 = dis11(3);
xx12 = dis12(1);
yy12 = dis12(2);
zz12 = dis12(3);
xx13 = dis13(1);
yy13 = dis13(2);
zz13 = dis13(3);
xx14 = dis14(1);
yy14 = dis14(2);
zz14 = dis14(3);
xx15 = dis15(1);
yy15 = dis15(2);
zz15 = dis15(3);
xx16 = dis16(1);
yy16 = dis16(2);
zz16 = dis16(3);
xx17 = dis17(1);
yy17 = dis17(2);
zz17 = dis17(3);
xx18 = dis18(1);
yy18 = dis18(2);
zz18 = dis18(3);
xx19 = dis19(1);
yy19 = dis19(2);
zz19 = dis19(3);
xx20 = dis20(1);
yy20 = dis20(2);
zz20 = dis20(3);
xx21 = dis21(1);
yy21 = dis21(2);
zz21 = dis21(3);
xx22 = dis22(1);
yy22 = dis22(2);
zz22 = dis22(3);
xx23 = dis23(1);
yy23 = dis23(2);
zz23 = dis23(3);
xx24 = dis24(1);
yy24 = dis24(2);
zz24 = dis24(3);
xx25 = dis25(1);
yy25 = dis25(2);
zz25 = dis25(3);
sxvelocity(25*(elmnt_id-1)+1)=xx1;
sxvelocity(25*(elmnt_id-1)+2)=xx2;
sxvelocity(25*(elmnt_id-1)+3)=xx3;
sxvelocity(25*(elmnt_id-1)+4)=xx4;
sxvelocity(25*(elmnt_id-1)+5)=xx5;
sxvelocity(25*(elmnt_id-1)+6)=xx6;
sxvelocity(25*(elmnt_id-1)+7)=xx7;
sxvelocity(25*(elmnt_id-1)+8)=xx8;
sxvelocity(25*(elmnt_id-1)+9)=xx9;
sxvelocity(25*(elmnt_id-1)+10)=xx10;
sxvelocity(25*(elmnt_id-1)+11)=xx11;
sxvelocity(25*(elmnt_id-1)+12)=xx12;
sxvelocity(25*(elmnt_id-1)+13)=xx13;
sxvelocity(25*(elmnt_id-1)+14)=xx14;
sxvelocity(25*(elmnt_id-1)+15)=xx15;
sxvelocity(25*(elmnt_id-1)+16)=xx16;
sxvelocity(25*(elmnt_id-1)+17)=xx17;
sxvelocity(25*(elmnt_id-1)+18)=xx18;
sxvelocity(25*(elmnt_id-1)+19)=xx19;
sxvelocity(25*(elmnt_id-1)+20)=xx20;
sxvelocity(25*(elmnt_id-1)+21)=xx21;
sxvelocity(25*(elmnt_id-1)+22)=xx22;
sxvelocity(25*(elmnt_id-1)+23)=xx23;
sxvelocity(25*(elmnt_id-1)+24)=xx24;
sxvelocity(25*(elmnt_id-1)+25)=xx25;
syvelocity(25*(elmnt_id-1)+1)=yy1;
syvelocity(25*(elmnt_id-1)+2)=yy2;
syvelocity(25*(elmnt_id-1)+3)=yy3;
syvelocity(25*(elmnt_id-1)+4)=yy4;
syvelocity(25*(elmnt_id-1)+5)=yy5;
syvelocity(25*(elmnt_id-1)+6)=yy6;
syvelocity(25*(elmnt_id-1)+7)=yy7;
syvelocity(25*(elmnt_id-1)+8)=yy8;
syvelocity(25*(elmnt_id-1)+9)=yy9;
syvelocity(25*(elmnt_id-1)+10)=yy10;
syvelocity(25*(elmnt_id-1)+11)=yy11;
syvelocity(25*(elmnt_id-1)+12)=yy12;
syvelocity(25*(elmnt_id-1)+13)=yy13;
syvelocity(25*(elmnt_id-1)+14)=yy14;
syvelocity(25*(elmnt_id-1)+15)=yy15;
syvelocity(25*(elmnt_id-1)+16)=yy16;
syvelocity(25*(elmnt_id-1)+17)=yy17;
syvelocity(25*(elmnt_id-1)+18)=yy18;
syvelocity(25*(elmnt_id-1)+19)=yy19;
syvelocity(25*(elmnt_id-1)+20)=yy20;
syvelocity(25*(elmnt_id-1)+21)=yy21;
syvelocity(25*(elmnt_id-1)+22)=yy22;
syvelocity(25*(elmnt_id-1)+23)=yy23;
syvelocity(25*(elmnt_id-1)+24)=yy24;
syvelocity(25*(elmnt_id-1)+25)=yy25;
szvelocity(25*(elmnt_id-1)+1)=zz1;
szvelocity(25*(elmnt_id-1)+2)=zz2;
szvelocity(25*(elmnt_id-1)+3)=zz3;
szvelocity(25*(elmnt_id-1)+4)=zz4;
szvelocity(25*(elmnt_id-1)+5)=zz5;
szvelocity(25*(elmnt_id-1)+6)=zz6;
szvelocity(25*(elmnt_id-1)+7)=zz7;
szvelocity(25*(elmnt_id-1)+8)=zz8;
szvelocity(25*(elmnt_id-1)+9)=zz9;
szvelocity(25*(elmnt_id-1)+10)=zz10;
szvelocity(25*(elmnt_id-1)+11)=zz11;
szvelocity(25*(elmnt_id-1)+12)=zz12;
szvelocity(25*(elmnt_id-1)+13)=zz13;
szvelocity(25*(elmnt_id-1)+14)=zz14;
szvelocity(25*(elmnt_id-1)+15)=zz15;
szvelocity(25*(elmnt_id-1)+16)=zz16;
szvelocity(25*(elmnt_id-1)+17)=zz17;
szvelocity(25*(elmnt_id-1)+18)=zz18;
szvelocity(25*(elmnt_id-1)+19)=zz19;
szvelocity(25*(elmnt_id-1)+20)=zz20;
szvelocity(25*(elmnt_id-1)+21)=zz21;
szvelocity(25*(elmnt_id-1)+22)=zz22;
szvelocity(25*(elmnt_id-1)+23)=zz23;
szvelocity(25*(elmnt_id-1)+24)=zz24;
szvelocity(25*(elmnt_id-1)+25)=zz25;
xindex(25*(elmnt_id-1)+1)=xsubindex1;
xindex(25*(elmnt_id-1)+2)=xsubindex2;
xindex(25*(elmnt_id-1)+3)=xsubindex3;
xindex(25*(elmnt_id-1)+4)=xsubindex4;
xindex(25*(elmnt_id-1)+5)=xsubindex5;
xindex(25*(elmnt_id-1)+6)=xsubindex1;
xindex(25*(elmnt_id-1)+7)=xsubindex2;
xindex(25*(elmnt_id-1)+8)=xsubindex3;
xindex(25*(elmnt_id-1)+9)=xsubindex4;
xindex(25*(elmnt_id-1)+10)=xsubindex5;
xindex(25*(elmnt_id-1)+11)=xsubindex1;
xindex(25*(elmnt_id-1)+12)=xsubindex2;
xindex(25*(elmnt_id-1)+13)=xsubindex3;
xindex(25*(elmnt_id-1)+14)=xsubindex4;
xindex(25*(elmnt_id-1)+15)=xsubindex5;
xindex(25*(elmnt_id-1)+16)=xsubindex1;
xindex(25*(elmnt_id-1)+17)=xsubindex2;
xindex(25*(elmnt_id-1)+18)=xsubindex3;
xindex(25*(elmnt_id-1)+19)=xsubindex4;
xindex(25*(elmnt_id-1)+20)=xsubindex5;
yindex(25*(elmnt_id-1)+1)=ysubindex1;
yindex(25*(elmnt_id-1)+2)=ysubindex2;
yindex(25*(elmnt_id-1)+3)=ysubindex3;
yindex(25*(elmnt_id-1)+4)=ysubindex4;
yindex(25*(elmnt_id-1)+5)=ysubindex5;
yindex(25*(elmnt_id-1)+6)=ysubindex1;
yindex(25*(elmnt_id-1)+7)=ysubindex2;
yindex(25*(elmnt_id-1)+8)=ysubindex3;
yindex(25*(elmnt_id-1)+9)=ysubindex4;
yindex(25*(elmnt_id-1)+10)=ysubindex5;
yindex(25*(elmnt_id-1)+11)=ysubindex1;
yindex(25*(elmnt_id-1)+12)=ysubindex2;
yindex(25*(elmnt_id-1)+13)=ysubindex3;
yindex(25*(elmnt_id-1)+14)=ysubindex4;
yindex(25*(elmnt_id-1)+15)=ysubindex5;
SIFunction.m

% Function:     SI_Function
% Description: M-file to determine the velocity field on the surface of the
% flat shell element at the laser measurement positions

function [SN] = SI_Function(si, pi, a, b, t)
Np1 = (si - 1) * (pi - 1);
Np2 = (1 - si) * pi;
Np3 = si * pi;
Np4 = (1 - pi) * si;
Nk1 = -si * pi - (3 - 2 * si) * (si^2) * (1 - pi) - (1 - si) * (3 - 2 * pi) * pi^2;
Nk2 = -si * pi + (1 - pi) * b;
Nk3 = -si * ((1 - si)^2) * (1 - pi) * a;
Nk4 = (1 - si) * (3 - 2 * pi) * pi^2 + si * (1 - si) * (1 - 2 * si) * pi;
Nk5 = -si * (1 - pi) * pi^2 + (1 - pi) * b;
Nk6 = -si * ((1 - si)^2) * pi * a;
Nk7 = (3 - 2 * si) * (si^2) * pi * pi * si * (1 - pi) * (1 - 2 * pi);
Nk8 = -si * (1 - pi) * (pi^2) * b;
Nk9 = -si * (1 - pi) * (pi^2) * a;
Nk10 = -(3 - 2 * si) * (si^2) * (1 - pi) + si * pi - (1 - pi) * (1 - 2 * pi);
Nk11 = si * pi + (1 - pi) * b;
Nk12 = -si * (1 - pi) * (si^2) * (1 - pi) * a;
Nkx1 = -2 * (3 - 2 * si) * si - (1 - pi) + 2 * (si^2) * (1 - pi) - (3 - 2 * pi) * pi^2;
Nky1 = -si * (3 - 2 * si) * (si^2) - 2 * (1 - si) * (3 - 2 * pi) * pi^2;
Nkx2 = -b * ((1 - pi)^2) * pi;
Nky2 = b * (1 - si) * ((1 - pi)^2) - 2 * b * (1 - si) * pi;
Nkx3 = -a * (1 - pi) * (1 - pi) + 2 * a * (1 - si) * (1 - pi) * b;
Nky3 = -a * (1 - si) * (1 - pi) * a;
Nkx4 = -(1 - 2 * si) * (1 - si) * (1 - si) * pi - (1 - si) * (1 - 2 * pi) * pi^2;
Nky4 = -(1 - 2 * si) * (1 - si) * (1 - si) * pi + 2 * si * (1 - pi) * pi^2;
Nkx5 = b * (1 - pi)^2 * pi;
Nky5 = 2 * b^2 * a + (1 - si) * (1 - pi) * (1 - pi) * pi^2;
Nkx6 = -a * (1 - si) * (1 - pi) * (1 - pi) * a;
Nky6 = -a * (1 - si) * (1 - pi) * a;
Nkx7 = -2 * (1 - si) * (si^2) * (1 - pi) + si * (1 - 2 * pi) * pi^2 + 2 * (si^2) * (1 - pi) * pi;
Nky7 = -2 * (1 - si) * (1 - pi) * (1 - pi) * si + 2 * (si^2) * (1 - pi) * si * pi + 2 * si * pi^2;
Nkx8 = b * (1 - pi)^2 * pi;
Nky8 = -2 * b^2 * a * (1 - si) * (1 - pi) * (1 - pi) * pi^2;
Nkx9 = -2 * a * (1 - si) * (1 - pi) * (1 - pi) * a;
Nky9 = -2 * a * (1 - si) * (1 - pi) * (1 - pi) * a;
Nkx10 = 2 * (1 - si) * (1 - pi) - 2 * (si^2) * (1 - pi) + (1 - pi) * (1 - pi) * (1 - pi) * pi^2;
Nkx11 = b * (1 - pi)^2 * pi;
Nky11 = 2 * b * (1 - si) * (1 - pi) * (1 - pi) * a;
Nkx12 = 2 * a * (1 - si) * (1 - pi) - (si^2) * (1 - pi) * a;
Nky12 = -a * (1 - si) * (1 - pi) * a;
SN = [Np1, (t - (2 * a)) * Nkx10, (-t - (2 * a)) * Nkx2, (-t - (2 * a)) * Nkx3, ...
Np2, (t - (2 * a)) * Nkx4, (-t - (2 * a)) * Nkx5, (-t - (2 * a)) * Nkx6, ...
Np3, (t - (2 * a)) * Nkx7, (-t - (2 * a)) * Nkx8, (-t - (2 * a)) * Nkx9, ...
Np4, (t - (2 * a)) * Nkx10, (-t - (2 * a)) * Nkx11, (-t - (2 * a)) * Nkx12, ...
0, Np1, (t - (2 * b)) * Nkx1, (t - (2 * b)) * Nkx2, (t - (2 * b)) * Nkx3, ...
0, Np2, (t - (2 * b)) * Nkx4, (t - (2 * b)) * Nkx5, (t - (2 * b)) * Nkx6, ...
0, Np3, (t - (2 * b)) * Nkx7, (t - (2 * b)) * Nkx8, (t - (2 * b)) * Nkx9, ...
0, Np4, (t - (2 * b)) * Nkx10, (t - (2 * b)) * Nkx11, (t - (2 * b)) * Nkx12, ...
0, 0, Nk1, Nk2, Nk3, 0, 0, Nk4, Nk5, Nk6, 0, 0, Nk7, Nk8, Nk9, 0, 0, Nk10, Nk11, Nk12];

D.2 The velocity transformation function of the flat shell element (Damped system)

dsurglobalvelfield.m

% Function:     dsurglobalvelfield
% Description: M-file to build the global velocity field on the surface of the
% damped flat shell element

Rsxxvelocity=zeros(length(element)*25,1);
Rsyyvelocity=zeros(length(element)*25,1);
Rszzvelocity=zeros(length(element)*25,1);
Isxvelocity=zeros(length(element)*25,1);

180
Isyvelocity=zeros(length(element)*25,1);
Iszvelocity=zeros(length(element)*25,1);
xindex=zeros(length(element)*25,1);
yindex=zeros(length(element)*25,1);

n_elmnt = length( element );
for elmnt_id = 1:n_elmnt
    [Rxsvelocity,Rsyvelocity,Rszvelocity,Isxvelocity,Isyvelocity,...
     xindex,yindex]=dsurvelocityfield(element(element_id),node,Rxsvelocity,...
     Rsyvelocity,Rszvelocity,Isxvelocity,Isyvelocity,xindex,yindex,elmnt_id);
end

k=1;
for j=1:n-1
    if node(1).x == node(j+1).x
        k=k+1;
    else
        k=k;
    end

    j=n/k;
    index=1;
    subindex=1;
    Rsxd=zeros(5*(k-2+1),5*(k-2+1));
    Rsyd=zeros(5*(k-2+1),5*(k-2+1));
    Rszd=zeros(5*(k-2+1),5*(k-2+1));
    for c=1:5:5*(j-2+1)
        for r=1:5:5*(k-2+1)
            index=1+25*(subindex-1);
            Rsxd(r,c)=Rxsvelocity(index);
            Rsxd(r+1,c)=Rxsvelocity(index+1);
            Rsxd(r+2,c)=Rxsvelocity(index+2);
            Rsxd(r+3,c)=Rxsvelocity(index+3);
            Rsxd(r+4,c)=Rxsvelocity(index+4);
            Rsxd(r,c+1)=Rxsvelocity(index+5);
            Rsxd(r+1,c+1)=Rxsvelocity(index+6);
            Rsxd(r+2,c+1)=Rxsvelocity(index+7);
            Rsxd(r+3,c+1)=Rxsvelocity(index+8);
            Rsxd(r+4,c+1)=Rxsvelocity(index+9);
            Rsxd(r,c+2)=Rxsvelocity(index+10);
            Rsxd(r+1,c+2)=Rxsvelocity(index+11);
            Rsxd(r+2,c+2)=Rxsvelocity(index+12);
            Rsxd(r+3,c+2)=Rxsvelocity(index+13);
            Rsxd(r+4,c+2)=Rxsvelocity(index+14);
            Rsxd(r,c+3)=Rxsvelocity(index+15);
            Rsxd(r+1,c+3)=Rxsvelocity(index+16);
            Rsxd(r+2,c+3)=Rxsvelocity(index+17);
            Rsxd(r+3,c+3)=Rxsvelocity(index+18);
            Rsxd(r+4,c+3)=Rxsvelocity(index+19);
            Rsxd(r,c+4)=Rxsvelocity(index+20);
            Rsxd(r+1,c+4)=Rxsvelocity(index+21);
            Rsxd(r+2,c+4)=Rxsvelocity(index+22);
            Rsxd(r+3,c+4)=Rxsvelocity(index+23);
            Rsxd(r+4,c+4)=Rxsvelocity(index+24);
            Rsyd(r,c)=Rsyvelocity(index);
            Rsyd(r+1,c)=Rsyvelocity(index+1);
            Rsyd(r+2,c)=Rsyvelocity(index+2);
            Rsyd(r+3,c)=Rsyvelocity(index+3);
            Rsyd(r+4,c)=Rsyvelocity(index+4);
            Rsyd(r,c+1)=Rsyvelocity(index+5);
            Rsyd(r+1,c+1)=Rsyvelocity(index+6);
            Rsyd(r+2,c+1)=Rsyvelocity(index+7);
            Rsyd(r+3,c+1)=Rsyvelocity(index+8);
            Rsyd(r+4,c+1)=Rsyvelocity(index+9);
            Rsyd(r,c+2)=Rsyvelocity(index+10);
            Rsyd(r+1,c+2)=Rsyvelocity(index+11);
            Rsyd(r+2,c+2)=Rsyvelocity(index+12);
            Rsyd(r+3,c+2)=Rsyvelocity(index+13);
            Rsyd(r+4,c+2)=Rsyvelocity(index+14);
            Rsyd(r,c+3)=Rsyvelocity(index+15);
            Rsyd(r+1,c+3)=Rsyvelocity(index+16);
            Rsyd(r+2,c+3)=Rsyvelocity(index+17);
            Rsyd(r+3,c+3)=Rsyvelocity(index+18);
            Rsyd(r+4,c+3)=Rsyvelocity(index+19);
            Rsyd(r,c+4)=Rsyvelocity(index+20);
        end
    end
end
Rsyd(r+2,c+3)=Rsyvelocity(index+17);
Rsyd(r+3,c+3)=Rsyvelocity(index+18);
Rsyd(r+4,c+3)=Rsyvelocity(index+19);
Rsyd(r,c+4)=Rsyvelocity(index+20);
Rsyd(r+1,c+4)=Rsyvelocity(index+21);
Rsyd(r+2,c+4)=Rsyvelocity(index+22);
Rsyd(r+3,c+4)=Rsyvelocity(index+23);
Rsyd(r+4,c+4)=Rsyvelocity(index+24);
Rszd(r,c)=Rszvelocity(index);
Rszd(r+1,c)=Rszvelocity(index+1);
Rszd(r+2,c)=Rszvelocity(index+2);
Rszd(r+3,c)=Rszvelocity(index+3);
Rszd(r+4,c)=Rszvelocity(index+4);
Rszd(r,c+1)=Rszvelocity(index+5);
Rszd(r+1,c+1)=Rszvelocity(index+6);
Rszd(r+2,c+1)=Rszvelocity(index+7);
Rszd(r+3,c+1)=Rszvelocity(index+8);
Rszd(r+4,c+1)=Rszvelocity(index+9);
Rszd(r,c+2)=Rszvelocity(index+10);
Rszd(r+1,c+2)=Rszvelocity(index+11);
Rszd(r+2,c+2)=Rszvelocity(index+12);
Rszd(r+3,c+2)=Rszvelocity(index+13);
Rszd(r+4,c+2)=Rszvelocity(index+14);
Rszd(r,c+3)=Rszvelocity(index+15);
Rszd(r+1,c+3)=Rszvelocity(index+16);
Rszd(r+2,c+3)=Rszvelocity(index+17);
Rszd(r+3,c+3)=Rszvelocity(index+18);
Rszd(r+4,c+3)=Rszvelocity(index+19);
Rszd(r,c+4)=Rszvelocity(index+20);
Rszd(r+1,c+4)=Rszvelocity(index+21);
Rszd(r+2,c+4)=Rszvelocity(index+22);
Rszd(r+3,c+4)=Rszvelocity(index+23);
Rszd(r+4,c+4)=Rszvelocity(index+24);
subindex=subindex+1;
end
subindex=1;
index=1;
Isxd=zeros(5*(k-2+1),5*(k-2+1));
Isyd=zeros(5*(k-2+1),5*(k-2+1));
Iszd=zeros(5*(k-2+1),5*(k-2+1));
for c=1:5:5*(j-2+1)
    for r=1:5:5*(k-2+1)
        index=1+25*(subindex-1);
        Isxd(r,c)=Isxvelocity(index);
        Isxd(r+1,c)=Isxvelocity(index+1);
        Isxd(r+2,c)=Isxvelocity(index+2);
        Isxd(r+3,c)=Isxvelocity(index+3);
        Isxd(r+4,c)=Isxvelocity(index+4);
        Isxd(r,c+1)=Isxvelocity(index+5);
        Isxd(r+1,c+1)=Isxvelocity(index+6);
        Isxd(r+2,c+1)=Isxvelocity(index+7);
        Isxd(r+3,c+1)=Isxvelocity(index+8);
        Isxd(r+4,c+1)=Isxvelocity(index+9);
        Isxd(r,c+2)=Isxvelocity(index+10);
        Isxd(r+1,c+2)=Isxvelocity(index+11);
        Isxd(r+2,c+2)=Isxvelocity(index+12);
        Isxd(r+3,c+2)=Isxvelocity(index+13);
        Isxd(r+4,c+2)=Isxvelocity(index+14);
        Isxd(r,c+3)=Isxvelocity(index+15);
        Isxd(r+1,c+3)=Isxvelocity(index+16);
        Isxd(r+2,c+3)=Isxvelocity(index+17);
        Isxd(r+3,c+3)=Isxvelocity(index+18);
        Isxd(r+4,c+3)=Isxvelocity(index+19);
        Isxd(r,c+4)=Isxvelocity(index+20);
        Isxd(r+1,c+4)=Isxvelocity(index+21);
        Isxd(r+2,c+4)=Isxvelocity(index+22);
        Isxd(r+3,c+4)=Isxvelocity(index+23);
        Isxd(r+4,c+4)=Isxvelocity(index+24);
        Isyd(r,c)=Isyvelocity(index);
        Isyd(r+1,c)=Isyvelocity(index+1);
    end
end
Isyd(r+2,c)=Isyvelocity(index+2);
Isyd(r+3,c)=Isyvelocity(index+3);
Isyd(r+4,c)=Isyvelocity(index+4);
Isyd(r,c+1)=Isyvelocity(index+5);
Isyd(r+1,c+1)=Isyvelocity(index+6);
Isyd(r+2,c+1)=Isyvelocity(index+7);
Isyd(r+3,c+1)=Isyvelocity(index+8);
Isyd(r+4,c+1)=Isyvelocity(index+9);
Isyd(r,c+2)=Isyvelocity(index+10);
Isyd(r+1,c+2)=Isyvelocity(index+11);
Isyd(r+2,c+2)=Isyvelocity(index+12);
Isyd(r+3,c+2)=Isyvelocity(index+13);
Isyd(r+4,c+2)=Isyvelocity(index+14);
Isyd(r,c+3)=Isyvelocity(index+15);
Isyd(r+1,c+3)=Isyvelocity(index+16);
Isyd(r+2,c+3)=Isyvelocity(index+17);
Isyd(r+3,c+3)=Isyvelocity(index+18);
Isyd(r+4,c+3)=Isyvelocity(index+19);
Isyd(r,c+4)=Isyvelocity(index+20);
Isyd(r+1,c+4)=Isyvelocity(index+21);
Isyd(r+2,c+4)=Isyvelocity(index+22);
Isyd(r+3,c+4)=Isyvelocity(index+23);
Isyd(r+4,c+4)=Isyvelocity(index+24);
Iszd(r,c)=Iszvelocity(index);
Iszd(r+1,c)=Iszvelocity(index+1);
Iszd(r+2,c)=Iszvelocity(index+2);
Iszd(r+3,c)=Iszvelocity(index+3);
Iszd(r+4,c)=Iszvelocity(index+4);
Iszd(r,c+1)=Iszvelocity(index+5);
Iszd(r+1,c+1)=Iszvelocity(index+6);
Iszd(r+2,c+1)=Iszvelocity(index+7);
Iszd(r+3,c+1)=Iszvelocity(index+8);
Iszd(r+4,c+1)=Iszvelocity(index+9);
Iszd(r,c+2)=Iszvelocity(index+10);
Iszd(r+1,c+2)=Iszvelocity(index+11);
Iszd(r+2,c+2)=Iszvelocity(index+12);
Iszd(r+3,c+2)=Iszvelocity(index+13);
Iszd(r+4,c+2)=Iszvelocity(index+14);
Iszd(r,c+3)=Iszvelocity(index+15);
Iszd(r+1,c+3)=Iszvelocity(index+16);
Iszd(r+2,c+3)=Iszvelocity(index+17);
Iszd(r+3,c+3)=Iszvelocity(index+18);
Iszd(r+4,c+3)=Iszvelocity(index+19);
Iszd(r,c+4)=Iszvelocity(index+20);
Iszd(r+1,c+4)=Iszvelocity(index+21);
Iszd(r+2,c+4)=Iszvelocity(index+22);
Iszd(r+3,c+4)=Iszvelocity(index+23);
Iszd(r+4,c+4)=Iszvelocity(index+24);
subindex=subindex+1;
end
end
subindex=1;
index=1;
xxd=zeros(5*(k-2+1),5*(j-2+1));
yyd=zeros(5*(k-2+1),5*(j-2+1));
for c=1:5:5*(j-2+1)
for r=1:5:5*(k-2+1)
index=1+25*(subindex-1);
xxd(r,c)=xindex(index);
xxd(r,c+1)=xindex(index+1);
xxd(r,c+2)=xindex(index+2);
xxd(r,c+3)=xindex(index+3);
xxd(r,c+4)=xindex(index+4);
xxd(r+1,c)=xindex(index+5);
xxd(r+1,c+1)=xindex(index+6);
xxd(r+1,c+2)=xindex(index+7);
xxd(r+1,c+3)=xindex(index+8);
xxd(r+1,c+4)=xindex(index+9);
xxd(r+2,c)=xindex(index+10);
xxd(r+2,c+1)=xindex(index+11);
xxd(r+2,c+2)=xindex(index+12);
xxd(r+2,c+3)=xindex(index+13);
xxd(r+2,c+4)=xindex(index+14);
xxd(r+3,c)=xindex(index+15);
xxd(r+3,c+1)=xindex(index+16);
xxd(r+3,c+2)=xindex(index+17);
xxd(r+3,c+3)=xindex(index+18);
xxd(r+3,c+4)=xindex(index+19);
xxd(r+4,c)=xindex(index+20);
xxd(r+4,c+1)=xindex(index+21);
xxd(r+4,c+2)=xindex(index+22);
xxd(r+4,c+3)=xindex(index+23);
xxd(r+4,c+4)=xindex(index+24);
end
end
dsurvelocityfield.m

% Function: dsurvelocityfield
% Description: M-file to determine the velocity field on the surface of a
damped flat shell element

function [Rsxvelocity, Rsyvelocity, Rszvelocity, Isxvelocity, Isyvelocity, Iszvelocity, xindex, yindex] = dsurvelocityfield(element, node, Rsxvelocity, Rsyvelocity, Rszvelocity, Isxvelocity, Isyvelocity, Iszvelocity, xindex, yindex, elmnt_id)

node_id1 = element.node(1); % get 1st node id
node_id2 = element.node(2); % get 2nd node id
node_id3 = element.node(3);
node_id4 = element.node(4);

% Additional code for dsurvelocityfield function
a = node(node_id3).x - node(node_id1).x;  % compute length
b = node(node_id2).y - node(node_id1).y;
SN1=SIFunction(0,1,a,b,element.tick);
SN2=SIFunction(0,0.75,a,b,element.tick);
SN3=SIFunction(0,0.5,a,b,element.tick);
SN4=SIFunction(0,0.25,a,b,element.tick);
SN5=SIFunction(0,0,a,b,element.tick);
SN6=SIFunction(0.25,1,a,b,element.tick);
SN7=SIFunction(0.25,0.75,a,b,element.tick);
SN8=SIFunction(0.25,0.5,a,b,element.tick);
SN9=SIFunction(0.25,0.25,a,b,element.tick);
SN10=SIFunction(0.25,0,a,b,element.tick);
SN11=SIFunction(0.5,1,a,b,element.tick);
SN12=SIFunction(0.5,0.75,a,b,element.tick);
SN13=SIFunction(0.5,0.5,a,b,element.tick);
SN14=SIFunction(0.5,0.25,a,b,element.tick);
SN15=SIFunction(0.5,0,a,b,element.tick);
SN16=SIFunction(0.75,1,a,b,element.tick);
SN17=SIFunction(0.75,0.75,a,b,element.tick);
SN18=SIFunction(0.75,0.5,a,b,element.tick);
SN19=SIFunction(0.75,0.25,a,b,element.tick);
SN20=SIFunction(0.75,0.0,a,b,element.tick);
SN21=SIFunction(1,1,a,b,element.tick);
SN22=SIFunction(1,0.75,a,b,element.tick);
SN23=SIFunction(1,0.5,a,b,element.tick);
SN24=SIFunction(1,0.25,a,b,element.tick);
SN25=SIFunction(1,0.0,a,b,element.tick);
xsubindex1=node(node_id1).x;
xsubindex2=a*0.25+node(node_id1).x;
xsubindex3=a*0.5+node(node_id1).x;
xsubindex4=a*0.75+node(node_id1).x;
xsubindex5=node(node_id1).y;
ysubindex1=node(node_id1).y;
ysubindex2=b*0.75+node(node_id1).y;
ysubindex3=b*0.5+node(node_id1).y;
ysubindex4=b*0.25+node(node_id1).y;
ysubindex5=node(node_id1).y;
Ru1=node(node_id1).Ru;
Ru2=node(node_id2).Ru;
Ru3=node(node_id3).Ru;
Ru4=node(node_id4).Ru;
Rv1=node(node_id1).Rv;
Rv2=node(node_id2).Rv;
Rv3=node(node_id3).Rv;
Rv4=node(node_id4).Rv;
Rw1=node(node_id1).Rw;
Rw2=node(node_id2).Rw;
Rw3=node(node_id3).Rw;
Rw4=node(node_id4).Rw;
Rxa1=node(node_id1).Rxa;
Rxa2=node(node_id2).Rxa;
Rxa3=node(node_id3).Rxa;
Rxa4=node(node_id4).Rxa;
Rz1=node(node_id1).Rza;
Rz2=node(node_id2).Rza;
Rz3=node(node_id3).Rza;
Rz4=node(node_id4).Rza;
Rdis=[Ru1,Rv1,Rw1,Rxa1,Rz1,Ru2,Rv2,Rw2,Rxa2,Rz2,Ru3,Rv3,Rw3,Rxa3,Rz3,Ru4,Rv4,Rw4,Rxa4,Rz4]';
Iu1=node(node_id1).Iu;
Iu2=node(node_id2).Iu;
Iu3=node(node_id3).Iu;
Iu4=node(node_id4).Iu;
Iv1=node(node_id1).Iv;
Iv2=node(node_id2).Iv;
Iv3=node(node_id3).Iv;
Iv4=node(node_id4).Iv;
Iw1=node(node_id1).Iw;
Iw2=node(node_id2).Iw;
Iw3=node(node_id3).Iw;
Iw4=node(node_id4).Iw;
Ixa1=node(node_id1).Ixa;
Ixa2=node(node_id2).Ixa;
Ixa3=node(node_id3).Ixa;
Ixa4=node(node_id4).Ixa;
Iza1=node(node_id1).Iza;
Iza2=node(node_id2).Iza;
Iza3=node(node_id3).Iza;
Iza4=node(node_id4).Iza;
Idis=[Iu1,Iv1,Iw1,Ixa1,Iza1,Iu2,Iv2,Ix2a2,Iza2,Iu3,Iv3,Iw3,Ixa3,Iza3,Iu4,Iv4,Iw4,Ixa4,Iza4]';
Rdis1=SN1*Rdis;
Rdis2=SN2*Rdis;
Rdis3=SN3*Rdis;
Rdis4=SN4*Rdis;
Rdis5=SN5*Rdis;
Rdis6=SN6*Rdis;
Rdis7=SN7*Rdis;
Rdis8=SN8*Rdis;
Rdis9=SN9*Rdis;
Rdis10=SN10*Rdis;
Rdis11=SN11*Rdis;
Rdis12=SN12*Rdis;
Rdis13=SN13*Rdis;
Rdis14=SN14*Rdis;
Rdis15=SN15*Rdis;
Rdis16=SN16*Rdis;
Rdis17=SN17*Rdis;
Rdis18=SN18*Rdis;
Rdis19=SN19*Rdis;
Rdis20=SN20*Rdis;
Rdis21=SN21*Rdis;
Rdis22=SN22*Rdis;
Rdis23=SN23*Rdis;
Rdis24=SN24*Rdis;
Rdis25=SN25*Rdis;
Rxx1=Rdis1(1);
Ryy1=Rdis1(2);
Rzz1=Rdis1(3);
Rxx2=Rdis2(1);
Ryy2=Rdis2(2);
Rzz2=Rdis2(3);
Rxx3=Rdis3(1);
Ryy3=Rdis3(2);
Rzz3=Rdis3(3);
Rxx4=Rdis4(1);
Ryy4=Rdis4(2);
Rzz4=Rdis4(3);
Rxx5=Rdis5(1);
Ryy5=Rdis5(2);
Rzz5=Rdis5(3);
Rxx6=Rdis6(1);
Ryy6=Rdis6(2);
Rzz6=Rdis6(3);
Rxx7=Rdis7(1);
Ryy7=Rdis7(2);
Rzz7=Rdis7(3);
Rxx8=Rdis8(1);
Ryy8=Rdis8(2);
Rzz8=Rdis8(3);
Rxx9=Rdis9(1);
Ryy9=Rdis9(2);
Rzz9=Rdis9(3);
Rxx10=Rdis10(1);
Ryy10=Rdis10(2);
Rzz10=Rdis10(3);
Rxx11=Rdis11(1);
Ryy11=Rdis11(2);
Rzz11=Rdis11(3);
Rxx12=Rdis12(1);
Ryy12=Rdis12(2);
Rzz12=Rdis12(3);
Rxx13=Rdis13(1);
Ryy13=Rdis13(2);
Rsyvelocity(25*(elmnt_id-1)+10)=Ryy10;
Rsyvelocity(25*(elmnt_id-1)+11)=Ryy11;
Rsyvelocity(25*(elmnt_id-1)+12)=Ryy12;
Rsyvelocity(25*(elmnt_id-1)+13)=Ryy13;
Rsyvelocity(25*(elmnt_id-1)+14)=Ryy14;
Rsyvelocity(25*(elmnt_id-1)+15)=Ryy15;
Rsyvelocity(25*(elmnt_id-1)+16)=Ryy16;
Rsyvelocity(25*(elmnt_id-1)+17)=Ryy17;
Rsyvelocity(25*(elmnt_id-1)+18)=Ryy18;
Rsyvelocity(25*(elmnt_id-1)+19)=Ryy19;
Rsyvelocity(25*(elmnt_id-1)+20)=Ryy20;
Rsyvelocity(25*(elmnt_id-1)+21)=Ryy21;
Rsyvelocity(25*(elmnt_id-1)+22)=Ryy22;
Rsyvelocity(25*(elmnt_id-1)+23)=Ryy23;
Rsyvelocity(25*(elmnt_id-1)+24)=Ryy24;
Rsyvelocity(25*(elmnt_id-1)+25)=Ryy25;
Rszvelocity(25*(elmnt_id-1)+1)=Rzz1;
Rszvelocity(25*(elmnt_id-1)+2)=Rzz2;
Rszvelocity(25*(elmnt_id-1)+3)=Rzz3;
Rszvelocity(25*(elmnt_id-1)+4)=Rzz4;
Rszvelocity(25*(elmnt_id-1)+5)=Rzz5;
Rszvelocity(25*(elmnt_id-1)+6)=Rzz6;
Rszvelocity(25*(elmnt_id-1)+7)=Rzz7;
Rszvelocity(25*(elmnt_id-1)+8)=Rzz8;
Rszvelocity(25*(elmnt_id-1)+9)=Rzz9;
Rszvelocity(25*(elmnt_id-1)+10)=Rzz10;
Rszvelocity(25*(elmnt_id-1)+11)=Rzz11;
Rszvelocity(25*(elmnt_id-1)+12)=Rzz12;
Rszvelocity(25*(elmnt_id-1)+13)=Rzz13;
Rszvelocity(25*(elmnt_id-1)+14)=Rzz14;
Rszvelocity(25*(elmnt_id-1)+15)=Rzz15;
Rszvelocity(25*(elmnt_id-1)+16)=Rzz16;
Rszvelocity(25*(elmnt_id-1)+17)=Rzz17;
Rszvelocity(25*(elmnt_id-1)+18)=Rzz18;
Rszvelocity(25*(elmnt_id-1)+19)=Rzz19;
Rszvelocity(25*(elmnt_id-1)+20)=Rzz20;
Rszvelocity(25*(elmnt_id-1)+21)=Rzz21;
Rszvelocity(25*(elmnt_id-1)+22)=Rzz22;
Rszvelocity(25*(elmnt_id-1)+23)=Rzz23;
Rszvelocity(25*(elmnt_id-1)+24)=Rzz24;
Rszvelocity(25*(elmnt_id-1)+25)=Rzz25;
Idis1=SN1*Idis;
Idis2=SN2*Idis;
Idis3=SN3*Idis;
Idis4=SN4*Idis;
Idis5=SN5*Idis;
Idis6=SN6*Idis;
Idis7=SN7*Idis;
Idis8=SN8*Idis;
Idis9=SN9*Idis;
Idis10=SN10*Idis;
Idis11=SN11*Idis;
Idis12=SN12*Idis;
Idis13=SN13*Idis;
Idis14=SN14*Idis;
Idis15=SN15*Idis;
Idis16=SN16*Idis;
Idis17=SN17*Idis;
Idis18=SN18*Idis;
Idis19=SN19*Idis;
Idis20=SN20*Idis;
Idis21=SN21*Idis;
Idis22=SN22*Idis;
Idis23=SN23*Idis;
Idis24=SN24*Idis;
Idis25=SN25*Idis;
Ixx1=Idis1(1);
Iyy1=Idis1(2);
Izz1=Idis1(3);
Ixx2=Idis2(1);
Iyy2=Idis2(2);
Izz2=Idis2(3);
Ixx3=Idis3(1);
Iyy3=Idis3(2);
Izz3=Idis3(3);
Ixx4=Idis4(1);
Iyy4=Idis4(2);
Izz4=Idis4(3);
Ixx5=Idis5(1);
Iyy5=Idis5(2);
Izz5=Idis5(3);
Ixx6=Idis6(1);
Iyy6=Idis6(2);
Izz6=Idis6(3);
Ixx7=Idis7(1);
Iyy7=Idis7(2);
Izz7=Idis7(3);
Ixx8=Idis8(1);
Iyy8=Idis8(2);
Izz8=Idis8(3);
Ixx9=Idis9(1);
Iyy9=Idis9(2);
Izz9=Idis9(3);
Ixx10=Idis10(1);
Iyy10=Idis10(2);
Izz10=Idis10(3);
Ixx11=Idis11(1);
Iyy11=Idis11(2);
Izz11=Idis11(3);
Ixx12=Idis12(1);
Iyy12=Idis12(2);
Izz12=Idis12(3);
Ixx13=Idis13(1);
Iyy13=Idis13(2);
Izz13=Idis13(3);
Ixx14=Idis14(1);
Iyy14=Idis14(2);
Izz14=Idis14(3);
Ixx15=Idis15(1);
Iyy15=Idis15(2);
Izz15=Idis15(3);
Ixx16=Idis16(1);
Iyy16=Idis16(2);
Izz16=Idis16(3);
Ixx17=Idis17(1);
Iyy17=Idis17(2);
Izz17=Idis17(3);
Ixx18=Idis18(1);
Iyy18=Idis18(2);
Izz18=Idis18(3);
Ixx19=Idis19(1);
Iyy19=Idis19(2);
Izz19=Idis19(3);
Ixx20=Idis20(1);
Iyy20=Idis20(2);
Izz20=Idis20(3);
Ixx21=Idis21(1);
Iyy21=Idis21(2);
Izz21=Idis21(3);
Ixx22=Idis22(1);
Iyy22=Idis22(2);
Izz22=Idis22(3);
Ixx23=Idis23(1);
Iyy23=Idis23(2);
Izz23=Idis23(3);
Ixx24=Idis24(1);
Iyy24=Idis24(2);
Izz24=Idis24(3);
Ixx25=Idis25(1);
Iyy25=Idis25(2);
Izz25=Idis25(3);
Isxvelocity(25*(elmnt_id-1)+1)=Ixx1;
Isxvelocity(25*(elmnt_id-1)+2)=Ixx2;
Isxvelocity(25*(elmnt_id-1)+3)=Ixx3;
Isxvelocity(25*(elmnt_id-1)+4)=Ixx4;
Isxvelocity(25*(elmnt_id-1)+5)=Ixx5;
Isxvelocity(25*(elmnt_id-1)+6)=Ixx6;
Isxvelocity(25*(elmnt_id-1)+7)=Ixx7;
Isxvelocity(25*(elmnt_id-1)+8)=Ixx8;
Isxvelocity(25*(elmnt_id-1)+9)=Ixx9;
Isxvelocity(25*(elmnt_id-1)+10)=Ixx10;
Isxvelocity(25*(elmnt_id-1)+11)=Ixx11;
Isxvelocity(25*(elmnt_id-1)+12)=Ixx12;
Isxvelocity(25*(elmnt_id-1)+13)=Ixx13;
Isxvelocity(25*(elmnt_id-1)+14)=Ixx14;
Isxvelocity(25*(elmnt_id-1)+15)=Ixx15;
Isxvelocity(25*(elmnt_id-1)+16)=Ixx16;
Isxvelocity(25*(elmnt_id-1)+17)=Ixx17;
Isxvelocity(25*(elmnt_id-1)+18)=Ixx18;
Isxvelocity(25*(elmnt_id-1)+19)=Ixx19;
Isxvelocity(25*(elmnt_id-1)+20)=Ixx20;
Isxvelocity(25*(elmnt_id-1)+21)=Ixx21;
Isxvelocity(25*(elmnt_id-1)+22)=Ixx22;
Isxvelocity(25*(elmnt_id-1)+23)=Ixx23;
Isxvelocity(25*(elmnt_id-1)+24)=Ixx24;
Isxvelocity(25*(elmnt_id-1)+25)=Ixx25;
Isyvelocity(25*(elmnt_id-1)+1)=Iyy1;
Isyvelocity(25*(elmnt_id-1)+2)=Iyy2;
Isyvelocity(25*(elmnt_id-1)+3)=Iyy3;
Isyvelocity(25*(elmnt_id-1)+4)=Iyy4;
Isyvelocity(25*(elmnt_id-1)+5)=Iyy5;
Isyvelocity(25*(elmnt_id-1)+6)=Iyy6;
Isyvelocity(25*(elmnt_id-1)+7)=Iyy7;
Isyvelocity(25*(elmnt_id-1)+8)=Iyy8;
Isyvelocity(25*(elmnt_id-1)+9)=Iyy9;
Isyvelocity(25*(elmnt_id-1)+10)=Iyy10;
Isyvelocity(25*(elmnt_id-1)+11)=Iyy11;
Isyvelocity(25*(elmnt_id-1)+12)=Iyy12;
Isyvelocity(25*(elmnt_id-1)+13)=Iyy13;
Isyvelocity(25*(elmnt_id-1)+14)=Iyy14;
Isyvelocity(25*(elmnt_id-1)+15)=Iyy15;
Isyvelocity(25*(elmnt_id-1)+16)=Iyy16;
Isyvelocity(25*(elmnt_id-1)+17)=Iyy17;
Isyvelocity(25*(elmnt_id-1)+18)=Iyy18;
Isyvelocity(25*(elmnt_id-1)+19)=Iyy19;
Isyvelocity(25*(elmnt_id-1)+20)=Iyy20;
Isyvelocity(25*(elmnt_id-1)+21)=Iyy21;
Isyvelocity(25*(elmnt_id-1)+22)=Iyy22;
Isyvelocity(25*(elmnt_id-1)+23)=Iyy23;
Isyvelocity(25*(elmnt_id-1)+24)=Iyy24;
Isyvelocity(25*(elmnt_id-1)+25)=Iyy25;
Iszvelocity(25*(elmnt_id-1)+1)=Izz1;
Iszvelocity(25*(elmnt_id-1)+2)=Izz2;
Iszvelocity(25*(elmnt_id-1)+3)=Izz3;
Iszvelocity(25*(elmnt_id-1)+4)=Izz4;
Iszvelocity(25*(elmnt_id-1)+5)=Izz5;
Iszvelocity(25*(elmnt_id-1)+6)=Izz6;
Iszvelocity(25*(elmnt_id-1)+7)=Izz7;
Iszvelocity(25*(elmnt_id-1)+8)=Izz8;
Iszvelocity(25*(elmnt_id-1)+9)=Izz9;
Iszvelocity(25*(elmnt_id-1)+10)=Izz10;
Iszvelocity(25*(elmnt_id-1)+11)=Izz11;
Iszvelocity(25*(elmnt_id-1)+12)=Izz12;
Iszvelocity(25*(elmnt_id-1)+13)=Izz13;
Iszvelocity(25*(elmnt_id-1)+14)=Izz14;
Iszvelocity(25*(elmnt_id-1)+15)=Izz15;
Iszvelocity(25*(elmnt_id-1)+16)=Izz16;
Iszvelocity(25*(elmnt_id-1)+17)=Izz17;
Iszvelocity(25*(elmnt_id-1)+18)=Izz18;
Iszvelocity(25*(elmnt_id-1)+19)=Izz19;
Iszvelocity(25*(elmnt_id-1)+20)=Izz20;
Iszvelocity(25*(elmnt_id-1)+21)=Izz21;
Iszvelocity(25*(elmnt_id-1)+22)=Izz22;
Iszvelocity(25*(elmnt_id-1)+23)=Izz23;
Iszvelocity(25*(elmnt_id-1)+24)=Izz24;
Iszvelocity(25*(elmnt_id-1)+25)=Izz25;
xindex(25*(elmnt_id-1)+1)=xsubindex1;
xindex(25*(elmnt_id-1)+2)=xsubindex2;
xindex(25*(elmnt_id-1)+3)=xsubindex3;
xindex(25*(elmnt_id-1)+4)=xsubindex4;
xindex(25*(elmnt_id-1)+5)=xsubindex5;
xindex(25*(elmnt_id-1)+6)=xsubindex1;
xindex(25*(elmnt_id-1)+7)=xsubindex2;
xindex(25*(elmnt_id-1)+8)=xsubindex3;
xindex(25*(elmnt_id-1)+9)=xsubindex4;
xindex(25*(elmnt_id-1)+10)=xsubindex5;
xindex(25*(elmnt_id-1)+11)=xsubindex1;
xindex(25*(elmnt_id-1)+12)=xsubindex2;
xindex(25*(elmnt_id-1)+13)=xsubindex3;
xindex(25*(elmnt_id-1)+14)=xsubindex4;
xindex(25*(elmnt_id-1)+15)=xsubindex5;
xindex(25*(elmnt_id-1)+16)=xsubindex1;
xindex(25*(elmnt_id-1)+17)=xsubindex2;
xindex(25*(elmnt_id-1)+18)=xsubindex3;
xindex(25*(elmnt_id-1)+19)=xsubindex4;
xindex(25*(elmnt_id-1)+20)=xsubindex5;
xindex(25*(elmnt_id-1)+21)=xsubindex1;
xindex(25*(elmnt_id-1)+22)=xsubindex2;
xindex(25*(elmnt_id-1)+23)=xsubindex3;
xindex(25*(elmnt_id-1)+24)=xsubindex4;
xindex(25*(elmnt_id-1)+25)=xsubindex5;
yindex(25*(elmnt_id-1)+1)=ysubindex1;
yindex(25*(elmnt_id-1)+2)=ysubindex2;
yindex(25*(elmnt_id-1)+3)=ysubindex3;
yindex(25*(elmnt_id-1)+4)=ysubindex4;
yindex(25*(elmnt_id-1)+5)=ysubindex5;
yindex(25*(elmnt_id-1)+6)=ysubindex1;
yindex(25*(elmnt_id-1)+7)=ysubindex2;
yindex(25*(elmnt_id-1)+8)=ysubindex3;
yindex(25*(elmnt_id-1)+9)=ysubindex4;
yindex(25*(elmnt_id-1)+10)=ysubindex5;
yindex(25*(elmnt_id-1)+11)=ysubindex1;
yindex(25*(elmnt_id-1)+12)=ysubindex2;
yindex(25*(elmnt_id-1)+13)=ysubindex3;
yindex(25*(elmnt_id-1)+14)=ysubindex4;
yindex(25*(elmnt_id-1)+15)=ysubindex5;
yindex(25*(elmnt_id-1)+16)=ysubindex1;
yindex(25*(elmnt_id-1)+17)=ysubindex2;
yindex(25*(elmnt_id-1)+18)=ysubindex3;
yindex(25*(elmnt_id-1)+19)=ysubindex4;
yindex(25*(elmnt_id-1)+20)=ysubindex5;
yindex(25*(elmnt_id-1)+21)=ysubindex1;
yindex(25*(elmnt_id-1)+22)=ysubindex2;
yindex(25*(elmnt_id-1)+23)=ysubindex3;
yindex(25*(elmnt_id-1)+24)=ysubindex4;
yindex(25*(elmnt_id-1)+25)=ysubindex5;
Vita

Kyongchan Song received a Bachelor of Science degree in Mechanical Engineering from University of Maryland Baltimore County in 1998. Kyongchan entered the Mechanical Engineering graduate program at Virginia Tech in 1998. He started to work as a full time research engineer in Eagle Aeronautics. Inc, Hampton, Virginia in 2000.

______________________________
Kyongchan Song