Modeling and Analysis of Sandwich deck and Hexagon interstage

An internship report submitted

in partial fulfillment for the award of the degree of

Bachelor of Technology

In **Aerospace engineering**

by **Shingala Vaidik** (SC21B021)

Guided By:

Dr. Ankur Kumar Gupta

Scientist/Engineer-SF

Structural Design Division, Structures Group,

U.R. Rao Satellite Centre (URSC),

Indian Space Research Organisation (ISRO)

(Dept. of Space, Govt. of India)

Department of Aerospace Engineering

Indian Institute of Space Science and Technology

Thiruvananthapuram, India

14th June- 28th July 2024

Certificate

This is to certify that the internship report titled ***Modeling and Analysis of Sandwich deck and Hexagon interstage*** submitted by **Shingala Vaidik**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Aerospace Engineering** is a bona fide record of the original work carried out by him/her under my supervision. The contents of this internship report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Ankur Kumar Gupta
Scientist/Engineer-SF
Structural Design Division,
U.R. Rao Satellite Centre (URSC),
Indian Space Research Organisation (ISRO)
(Dept. of Space, Govt. of India)

Dr. Anish Kumar
Assistant Professor

Place: Thiruvananthapuram
Date: 14th June- 28th July 2024

Declaration

I declare that this internship report titled ***Modeling and Analysis of Sandwich deck and Hexagon interstage*** submitted in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Aerospace Engineering** is a record of the original work carried out by me under the supervision of **Dr. Ankur Kumar Gupta**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram Shingala Vaidik
Date: 14th June- 28th July 2024 (SC21B054)

Acknowledgements

I would like to express my deepest gratitude to my guide, **Dr. Ankur Kumar Gupta**, Scientist/Engineer-SF Structural Design Division, Structures Group, U.R. Rao Satellite Centre (URSC), Indian Space Research Organisation (ISRO) (Dept. of Space, Govt. of India), for his exceptional guidance and support throughout my internship. His vast knowledge and expertise in structural design have been invaluable in helping me navigate complex challenges and achieve my research goals. His continuous encouragement and insightful feedback significantly enhanced the quality of my work.

I am equally grateful to my internal guide, **Dr. Anish Kumar**, Professor of the Aerospace Engineering Department at the Indian Institute of Space Science and Technology, Thiruvananthapuram. His academic mentorship and unwavering support have been crucial in shaping the direction of my research. Dr.Anish Kumar's profound understanding of aerospace engineering principles and his dedication to academic excellence have inspired me to expand my knowledge and skills.

I would also like to extend my sincere thanks to the faculty and staff of the Aerospace Engineering Department at IIST for providing a stimulating academic environment and the resources necessary for my research.

Thank you all for your guidance, support, and motivation, which have been pivotal in the successful completion of this internship.

Shingala Vaidik

Abstract

This report presents the work carried out during a 6-week internship at U.R. Rao Satellite Centre (URSC), ISRO, under the guidance of Dr. Ankur Kumar Gupta. The internship focused on the structural dynamics and modeling of aerospace components, specifically sandwich plates and hexagon grid structures, using MSC Patran and Nastran software.

The first part of the internship involved the modeling and frequency analysis of a sandwich honeycomb plate. The task included creating a 2D geometry, defining laminate properties, and performing a 10-mode frequency analysis. A parameterized PCL (Patran Command Language) script was developed to automate the process, allowing for the modeling and analysis of sandwich plates with varying dimensions and material properties.

The second part focused on the modeling and analysis of a hexagon grid structure, which is commonly used in the interstage sections of launch vehicles. The complex geometry of the hexagon grid was accurately represented using a helical curve, and a PCL script was developed to enable the parameterization of the model. The analysis included both frequency and buckling studies, providing a comprehensive understanding of the structural behavior of the hexagon grid.

The report concludes with an overview of the key learnings, including the application of composite material theories, the use of advanced modeling techniques in Patran, and the automation of analysis processes using PCL.

Contents

List of Figures

Nomenclature

E	Elastic modulus
G	Shear modulus
ρ	Density
ν	Poisson's ratio
s	Subscript representing the elastic parameters of the core layer
c	Subscript representing the elastic parameters of the honeycomb core layer
γ	Correction factor (generally between 0.4 and 0.6)
t	Thickness (consistent with previously defined dimensions)
L	Length (consistent with previously defined dimensions)
λ	Frequency parameter
ω	Circular frequency
a, b	Plate side length
D	Plate flexural rigidity
H	Overall height of the cylindrical shell
D	Diameter of the cylindrical shell
ϕ	Helical angle between helical ribs and the shell meridian
n	Number of ribs in one helical direction (either $+\phi$ or $-\phi$)
t	Thickness of the shell
b_h	Width of helical ribs
b_c	Width of circumferential ribs
b_l	Width of longitudinal ribs
a_c	Spacing between circumferential ribs, calculated as $\frac{\pi D}{2n \tan \phi}$
n_c	Number of circumferential ribs, calculated as $\frac{2nH \tan \phi}{\pi D}$

Chapter 1
Introduction

This document outlines my experiences and the insights I acquired during my six-week internship at the U.R. Rao Satellite Centre (URSC), ISRO, under the mentorship of Dr. Ankur Kumar Gupta. The internship afforded me a significant opportunity to engage with structural dynamics, with a particular emphasis on the analysis and modeling of sandwich plates and hexagonal grid structures utilizing MSC Patran and Nastran software.

To begin, I revisited the principles of Strength of Materials, which laid the groundwork for comprehending the material properties crucial for the tasks I undertook. Throughout the duration of the internship, I investigated various facets of material modeling, encompassing isotropic, orthotropic, and transversely isotropic materials, and acquired skills in utilizing and automating the Patran software through PCL (Patran Command Language). This report is structured into two principal sections: the analysis and modeling of a sandwich honeycomb plate and a hexagonal grid structure, each addressing distinct elements of structural dynamics and modeling methodologies.

1.1 Modeling and Frequency Analysis of Sandwich Plate

A sandwich plate, commonly employed in aerospace applications, comprises a lightweight core material situated between two face plates. The core contributes shear rigidity, while the face plates provide resistance to bending. The primary aim during this segment of the internship was to model a sandwich honeycomb plate and conduct a frequency analysis using MSC Patran and Nastran software.

Following a review of the fundamental concepts in Strength of Materials, I delved into the characteristics of composite materials, with a particular emphasis on sandwich structures. The initial phase involved the creation of a two-dimensional geometry using Patran, after which I defined the laminate properties pertinent to the sandwich structure. Subsequently, I imposed boundary conditions and conducted a frequency analysis to ascertain the natural frequencies and modes associated with the structure. A significant challenge was the development of a parameterized PCL code capable of modeling the sandwich plate for any specified dimensions and material characteristics.

1.1.1 Procedure

The procedure commenced with the establishment of the 2D geometry for the sandwich plate. The core was represented as an equivalent homogeneous material, in accordance with insights from relevant research literature and consultations with my advisor. In Patran, the laminate properties were specified, assigning the appropriate material characteristics to both the face plates and the core. Boundary conditions were implemented by fixing all four edges, followed by a frequency analysis encompassing ten modes. The results obtained were corroborated and validated through comparisons with analytical calculations and existing literature.

To streamline the process, I created a PCL script that facilitates the parameterization of the sandwich plate model. This script is designed to accept various input dimensions and material properties, automatically generating the geometry, applying loads and boundary conditions, and executing the frequency analysis. The final output comprises the natural frequencies and mode shapes corresponding to any defined configuration.

1.2 Frequency and Buckling Analysis of Hexagon Grid Structure

The utilization of hexagon grid structures in aerospace engineering is attributed to their exceptional strength-to-weight ratio. These configurations are frequently implemented in the interstage sections of launch vehicles. The objective of this segment of the internship was to create a model of a hexagon grid structure, conduct a frequency analysis, and subsequently expand the analysis to encompass buckling considerations.

Modeling the hexagon grid structure presented a distinct challenge due to the intricate geometry involved. The project necessitated the generation of a helical curve to accurately depict the hexagonal arrangement. I employed PCL to parameterize the model, which facilitated the development of a grid structure with diverse dimensions and characteristics.

1.2.1 Procedure

The initial step involved constructing the hexagon grid geometry utilizing MSC Patran. The geometry was meticulously designed to ensure precision, particularly in the creation of the helical curve. Material properties were assigned, and beam elements were utilized to represent the grid structure. Following the mesh generation, boundary conditions were implemented, and a frequency analysis encompassing ten modes was executed.

After completing the frequency analysis, the PCL script underwent enhancements to incorporate a buckling analysis, necessitating further study and refinement of the PCL code. The end result was a comprehensive parameterized model that could execute both frequency and buckling analyses for hexagonal grid structures of varying dimensions.

Chapter 2
Modelling and Frequency Analysis of Sandwich Plate

This chapter examines the finite element modeling and analysis of honeycomb sandwich plates, which are crucial elements in aerospace engineering due to their remarkable stiffness-to-weight ratio. Utilizing the Sandwich Panel Theory, the project sought to accurately depict the intricate behavior of these structures under diverse loading scenarios. The research incorporated computational modeling through MSC Patran and Nastran software to replicate the dynamic response of honeycomb sandwich plates. A thorough parametric study was performed to assess the impact of varying core and face sheet thicknesses on natural frequencies. The findings revealed a strong alignment between the modeled predictions and theoretical expectations, thereby affirming the efficacy of the Sandwich Panel Theory in representing the distinctive mechanical characteristics of honeycomb structures. This investigation offers significant insights for enhancing the design of lightweight satellite structures, underscoring the importance of careful selection of material and geometric parameters to achieve optimal performance outcomes.

2.1 Introduction

The process of designing and developing space application structures necessitates complex decision-making, wherein various conflicting requirements must be reconciled. A primary consideration is the reduction of structural mass, as this factor is directly linked to enhanced payload capacity and lower launch expenses. Therefore, the incorporation of honeycomb sandwich structures, recognized for their remarkable stiffness and favorable strength-to-weight ratios, is essential in the realm of space application structures design.

In recent decades, significant research efforts have focused on the modeling, testing, and implementation of sandwich structures. The detailed finite element modeling (FEM) of these structures poses considerable computational challenges due to their intricate geometries and diverse material properties. As a result, the adoption of equivalent modeling techniques, whether applied to the core or the entire structure, has emerged as a viable approach to address these computational challenges.

The investigation conducted by Wang et al.[3] (2020) addresses the difficulties associated with direct finite-element modeling of honeycomb sandwich panels, which are essential elements in satellite solar arrays due to their favorable stiffness-to-weight ratio. In light of the computational intricacies involved in modeling these structures, the research evaluates various equivalent modeling theories, such as sandwich panel theory, honeycomb panel theory, and equivalent panel theory. By performing dynamic characteristic analyses and comparing natural frequencies, the study concludes that sandwich panel theory serves as the most precise equivalent model, exhibiting minimal response discrepancies when compared to experimental data. These results highlight the effectiveness of sandwich panel theory in accurately simulating and dynamically analyzing satellite structures, thereby offering a validated methodology for prospective space applications.

In their research, Aborehab et al. [4] investigate various finite element methods (FEM) for the modeling of satellite honeycomb sandwich plates, emphasizing the Shell Plate Sandwich (SPS) technique. This method utilizes 2D shell elements (SHELL181) to accurately represent the characteristics of the honeycomb core and adhesive layers. I intend to utilize the outcomes derived from this SPS method as a reference point to corroborate my own results, especially concerning natural frequencies and mode shapes.

Kumar et al. [5] performed a modal analysis utilizing finite element methods on a sandwich composite plate featuring a honeycomb core, which was modeled in ANSYS APDL. The research highlighted the influence of the thickness of the face sheets and the properties of the core material on the natural frequencies and mode shapes. The findings indicated that the use of stiffer core materials results in elevated natural frequencies, while an increase in the thickness of the face sheets in relation to the overall height of the plate substantially enhances these frequencies.

Boudjemai et al. [6] performed an extensive investigation into the multidisciplinary design and analysis (MDA) of honeycomb sandwich panels intended for satellite applications, highlighting the significance of finite element modeling (FEM) in forecasting modal frequencies. The FEM models underwent validation through experimental methods, demonstrating a deviation of less than 4% in the first two modal frequencies. Additionally, the research explored the influence of geometric parameters and material selections on the dynamic characteristics of honeycomb structures. These results are essential for corroborating my FEM simulations, especially regarding the impact of core thickness and cell size on modal frequencies.

2.1.1 Structural Configuration of Sandwich Panels

The structural configuration of a sandwich panel is akin to that of an I-beam. The face sheets act as the flanges, resisting bending stresses, while the core, which functions similarly to the web, resists shear loads and maintains the spacing between the face sheets. This separation increases the moment of inertia, thereby enhancing the panel's bending stiffness without significantly adding to its weight.

Classical laminate theory is often used to analyze the stress distribution within a sandwich structure. Each layer—whether a face sheet or the core—is treated as a distinct laminate with its own material properties. The core functions like the web of an I-beam, providing continuous support and allowing the structure to behave as a monolithic unit. This configuration is highly effective for applications where a high strength-to-weight ratio is critical, such as in aerospace engineering.[7]

Figure 2.1 illustrates a typical sandwich structure with a honeycomb core. The adhesive layer plays a critical role in bonding the core and face sheets together, ensuring that the entire assembly behaves as a single unit under load.

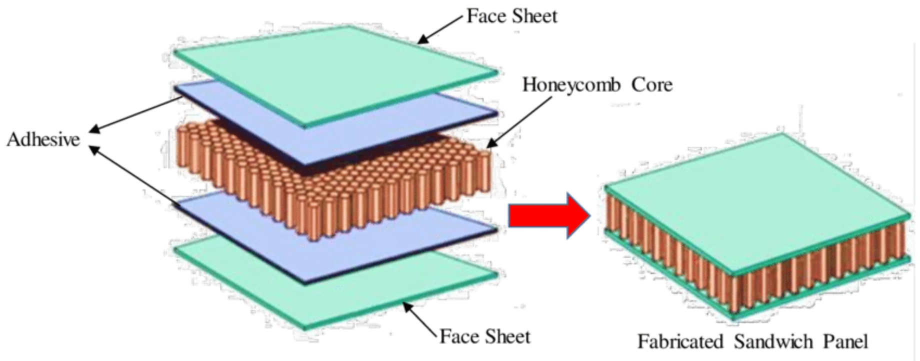


Figure 2.1: Schematic of a sandwich structure with honeycomb core.[1]

Figure 2.2 illustrates the geometry of an idealized honeycomb core cell, which is commonly used in numerical and analytical analyses. The honeycomb core's geometry, including double cell walls, single or inclined cell walls, and adhesive layers, contributes significantly to the mechanical properties of the sandwich structure.

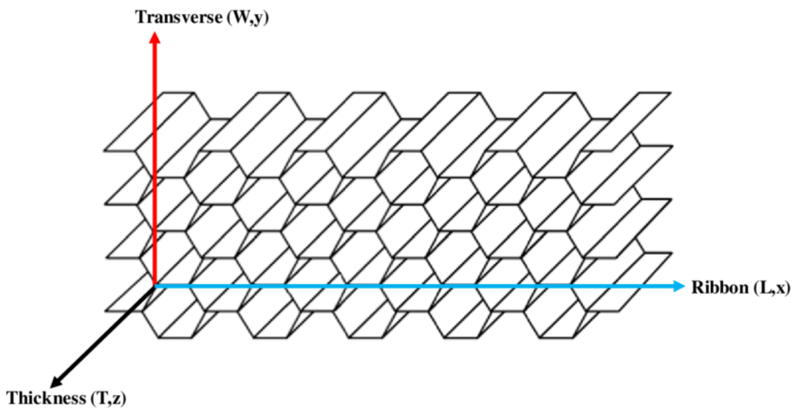


Figure 2.2: Geometry of an idealized honeycomb core cell.[1]

2.1.2 Benefits and Applications of Honeycomb Sandwich Constructions

Honeycomb sandwich structures are widely regarded for their high strength-to-weight ratio, making them indispensable in aerospace applications. Structurally, these panels can be likened to an I-beam, where the face sheets act as flanges handling bending stresses, while the honeycomb core functions as the web, resisting shear forces and maintaining the structural integrity of the panel.

The adhesive that bonds the core to the face sheets ensures that the components act as a single unit, providing high torsional and bending rigidity. This unity is essential for maintaining structural performance under various loading conditions. Additionally, honeycomb sandwich constructions offer excellent fatigue resistance, noise abatement, and enhanced stability.

The versatility in material selection for both the core and face sheets allows for customization based on specific structural, economic, and environmental needs. For instance, materials like aluminum and Nomex are commonly used due to their favorable mechanical properties and environmental resistance. These considerations are integral to the design and application of sandwich structures, particularly in conducting frequency and buckling analyses to ensure the structural integrity of aerospace components.

2.1.3 Frequency Analysis of Sandwich Structures

Frequency analysis is crucial in understanding the dynamic behavior of sandwich structures, particularly their natural frequencies and mode shapes. In aerospace applications, avoiding resonance is essential, as it can lead to catastrophic failures. The natural frequencies of a sandwich structure depend on factors such as stiffness, mass distribution, boundary conditions, and geometry.

The frequency analysis of sandwich structures is typically performed using finite element methods (FEM) in software like MSC Patran and Nastran. During this process, the structure is modeled, and appropriate boundary conditions are applied to simulate real-world constraints. The analysis results in the natural frequencies and mode shapes, which are critical for designing structures that can withstand dynamic loads without resonating.

2.2 Equivalent Theories for Honeycomb Sandwich Plate Modeling

When analyzing and modeling honeycomb sandwich plates, different equivalent theories can be employed to approximate the complex behavior of these structures. Each theory offers a different level of detail and computational efficiency, depending on the requirements of the analysis. Below is an overview of the primary theories used in finite element modeling (FEM) of honeycomb sandwich structures, along with their key assumptions and applications.

2.2.1 Sandwich Panel Theory

The Sandwich Panel Theory is an equivalent method specifically designed for honeycomb cores. It assumes that the core layer resists transverse shear deformation and has a certain in-plane stiffness. The upper and lower skin layers obey the Kirchhoff hypothesis, which ignores their resistance to transverse shears. The honeycomb core can be equivalent to a homogeneous, thickness-invariant, orthotropic layer. Since only the hexagonal honeycomb is discussed, the equivalent material parameters for the core are as follows:

$$E_x = 4\sqrt{3}\left(1 - \frac{3t^2}{L^2}\right)\left(\frac{t^3}{L^3}\right)E_s, \tag{2.1}$$

$$E_y = 4\sqrt{3}\left(1 - \frac{3t^2}{L^2}\right)\left(\frac{t^3}{L^3}\right)E_s, \tag{2.2}$$

$$E_Z = 2\sqrt{3}\frac{t}{L}E_s, \tag{2.3}$$

$$G_{cxy} = \frac{\sqrt{3}\gamma}{3} E_s \frac{t}{L} \left(\frac{1}{3}\right), \tag{2.4}$$

$$G_{cyz} = \frac{\sqrt{3}\gamma}{3} \frac{t}{L} \left(\frac{1}{3}\right) E_s, \tag{2.5}$$

$$G_{cxz} = \frac{\sqrt{3}\gamma}{3} \frac{t}{L} \left(\frac{1}{3}\right) E_s, \tag{2.6}$$

$$\rho_c = 1.155 \frac{t}{L} \rho_s, \tag{2.7}$$

$$\mu_{cxy} = \frac{1}{3}, \tag{2.8}$$

where E, G, ρ, and ν represent the elastic modulus, shear modulus, density, and Poisson's ratio, respectively. The subscript s represents the elastic parameters of the core layer, and the subscript c represents the elastic parameters of the honeycomb core layer. The coefficient γ is a correction factor, generally between 0.4 and 0.6, while t and L are consistent with the previously defined dimensions.[3]

2.2.2 Honeycomb Panel Theory

The honeycomb panel theory is established from the kinetic equation and the Hamiltonian principle. This framework conceptualizes the entire structure as an orthotropic panel characterized by uniform stiffness and dimensions:

- **Equivalent Structure**: The original honeycomb sandwich panel is represented as an equivalent orthotropic structure, taking into account both in-plane and out-of-plane mechanical characteristics.

- **Coordinate System**: The primary axis of the structure is aligned with the natural coordinate axes, with the geometric midplane designated as the coordinate midplane.

- **Mechanical Parameters**: The equivalent mechanical parameters are derived from the symmetry and inherent properties of the honeycomb sandwich panels.[3]

2.2.3 Equivalent Plate Theory

The equivalent plate theory offers a simplification of the sandwich structure by modeling it as a single-layer isotropic plate with equivalent properties. This theory is predicated on:

- **Axial and Bending Stiffness Equivalence**: The axial and bending stiffness of the sandwich structure are aligned with those of an equivalent isotropic plate.
- **Simplification**: The honeycomb core and face sheets are consolidated into a homogeneous plate, significantly streamlining the finite element model.

This theoretical approach is particularly advantageous when a simplified model is required to decrease computational demands while still accurately reflecting the overall stiffness characteristics of the structure.[3]

2.2.4 Reissner Theory

The Reissner theory enhances classical plate theory by incorporating shear deformation and rotational inertia, rendering it more applicable to thick plates or panels. In the context of honeycomb sandwich panels:

- **Surface Panel**: The face sheets are considered thin and are assumed to uniformly distribute normal and shear stresses throughout their thickness.
- **Honeycomb Core**: The core is regarded as compliant and does not resist in-plane stresses, facilitating a more precise representation.

This theory is more accurate for thick sandwich panels where shear deformation plays a significant role.[3]

2.2.5 Hoff Theory

The Hoff theory further refines the Reissner theory by considering the bending stiffness of the face sheets. It involves:

- **Stress Distribution**: Unlike the Reissner theory, Hoff theory includes the stress distribution across the face sheets, leading to a more accurate calculation of equivalent stiffness.

- **Enhanced Accuracy**: This theory is particularly useful when a detailed analysis of the face sheet stresses is required, especially in bending.

The Hoff theory is often used when precise modeling of the face sheet behavior under bending is critical.[3]

2.2.6 Application of Sandwich Panel Theory in My Internship

During my internship, I utilized the Sandwich Panel Theory to model honeycomb sandwich plates. This theory was selected for its capability to accurately represent the distinct characteristics of the layered structure, particularly the core's out-of-plane stiffness and the face sheets' in-plane load-bearing capacity. By applying this theory, I developed detailed FEM models that closely mirror the real-world behavior of honeycomb sandwich structures. This approach enabled precise frequency and buckling analyses, ensuring the accuracy of the results obtained during the project.

2.3 Modeling Process in MSC Patran

2.3.1 Benifits of MSC Patran

The selection of MSC Patran for this internship was primarily influenced by its seamless integration with MSC Nastran and its sophisticated capabilities for modeling intricate structures, such as honeycomb sandwich plates. The software's comprehensive tools for geometry creation and modification facilitate precise modeling, which is essential for conducting accurate finite element analyses. A significant consideration in this choice was the Patran Command Language (PCL), which markedly improves both productivity and flexibility.

PCL serves as a vital component of Patran, enabling the automation of repetitive tasks and the development of custom commands that cater to specific project requirements. This functionality proved crucial for the efficient management of parameterized models and the execution of complex simulations. The ability to script and automate various elements of the modeling process not only streamlined analyses but also allowed for swift modifications to design parameters. Additionally, PCL's capacity to create user-defined forms and menus significantly enhanced the customization and efficiency of the modeling workflow. In summary, the advantages offered by PCL in terms of process automation and customization were pivotal, establishing MSC Patran as the ideal choice for the project's needs.

2.3.2 Initial Challenges in Learning MSC Patran

Mastering MSC Patran presented numerous challenges, primarily due to the limited availability of resources compared to other more commonly used software applications. Unlike these widely adopted tools, which are supported by a wealth of online tutorials and instructional videos, the educational materials for Patran are significantly restricted. The main resource was the official MSC Patran manual, which, while comprehensive, required a systematic approach to fully grasp the software's functionalities.

Initially, my goal was to expedite my learning process by utilizing the direct tutorials included in the MSC Patran manual. To familiarize myself with the software, I began with tutorials that concentrated on the geometry of annular plates and lugs. These resources provided a foundational understanding of Patran's interface and essential operations, which were vital for skill development. However, despite this initial progress, transitioning to more complex tasks, such as modeling a honeycomb sandwich plate, proved to be quite difficult due to the lack of specific, detailed instructions in the available resources.

The absence of thorough, specialized tutorials for sandwich plate modeling forced me to rely heavily on the general guidelines from the Patran manual, which I had to adapt to meet my specific needs. This process required significant experimentation and self-directed learning to overcome the limitations of the available materials. Nonetheless, the foundational knowledge gained from the introductory tutorials created a solid base, allowing me to progressively refine my skills and successfully model the sandwich plates necessary for the analyses required during my internship.

2.3.3 Methodology

Geometry Creation in MSC Patran

The initial phase of modeling the sandwich plate involved creating the geometric representation of the plate using MSC Patran's geometry creation tools. The process began with the creation of a 1x1 square, which served as the foundational shape for the sandwich plate model.

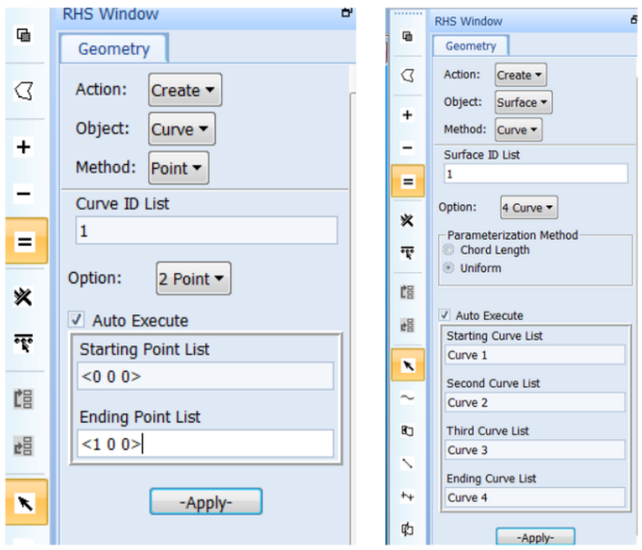
1. **Create Curves:**

   - Navigate to `Action` > `Create Object` > `Curve`.
   - Select `Point` Method and choose the `2 Point` option to define the curve.

- Input the coordinates to create the edges of a rectangle with the desired dimensions. In this case, a 1x1 square was created to represent the initial sandwich plate geometry.

2. **Generate the Surface:**

    - Navigate to `Action` > `Create Object` > `Surface`.
    - Select the `Curve` method and use the four curves defining the boundary of the square.
    - Generate the surface by specifying these curves, which will create a planar surface within the defined boundary.
    - This surface will act as the base for further modeling, such as adding the sandwich core and face sheets.



RHS Window
Geometry
Action: Create
Object: Curve
Method: Point
Curve ID List
1
Option: 2 Point
✓ Auto Execute
Starting Point List
<0 0 0>
Ending Point List
<1 0 0>
-Apply-

RHS Window
Geometry
Action: Create
Object: Surface
Method: Curve
Surface ID List
1
Option: 4 Curve
Parameterization Method
○ Chord Length
● Uniform
✓ Auto Execute
Starting Curve List
Curve 1
Second Curve List
Curve 2
Third Curve List
Curve 3
Ending Curve List
Curve 4
-Apply-

Figure 2.3: Creation of a 1x1 square and surface in MSC Patran.

Meshing the Sandwich Plate

1. **Select the Mesh Type:**

    - Navigate to `Action > Create Object > Mesh Type > Surface Elem Shape > Quad Mesher`.
    - Choose `Isomesh` and `Topology > Quad4` to define the mesh topology.

2. **Define Element Size:**

    - After selecting the surface, specify an element size of `0.05` to control the mesh density.
    - This size determines the granularity of the mesh, affecting both the accuracy of the simulation and the computational cost.

- Quad Mesher: This tool creates a mesh composed of quadrilateral (quad) elements, which are well-suited for planar surfaces and can capture complex geometries effectively.
- Isomesh Topology: Refers to the method used to automatically generate a regular mesh structure over the surface. The `Quad4` option indicates the use of quadrilateral elements with four nodes each.
- Quad4: A type of quadrilateral element with four nodes. Each node represents a corner of the element, and the element is defined by its shape and connectivity to adjacent elements.

Applying Boundary Conditions and Loads

1. **Create Nodal Displacement Boundary Conditions:**

    - Navigate to `Action > Create Object > Displacement Type > Nodal`.
    - Name the boundary condition set as `fixed`.

2. **Specify Displacement Values:**

    - Click on `Input Data` and set the displacement values to zero in all three directions (X, Y, and Z).
    - Similarly, set the rotational values to zero in all three directions (X, Y, and Z) to ensure that the edges are fully fixed.

3. **Define the Application Region:**

    - Click on `Select Application Region` and choose the edges of the sandwich plate where the boundary condition will be applied.
    - This step ensures that all four edges of the plate are fixed as required.

The nodal displacement boundary condition is used to constrain the movement of nodes along specified directions. By setting displacement and rotation to zero, the boundary conditions ensure that the edges of the plate do not move or rotate, simulating a fixed support condition. Applying these conditions to the surface of the plate ensures that all four edges are appropriately constrained, which is crucial for accurately analyzing the structural response of the plate under various loading conditions.

Defining Material Properties for Sandwich Panel

In this section, the material properties for the sandwich panel are defined and assigned using the Sandwich Panel Theory as outlined in Section 2.2.1. The sandwich panel consists of three layers: the top face sheet, the core, and the bottom face sheet. The materials used for the face sheets and core are specified below.

Face Sheet Material The face sheets are made of aluminum, with the following material properties provided by the guide:

$E_{11} = 7.2 \times 10^{10}\,\text{Pa}, \quad E_{22} = 7.2 \times 10^{10}\,\text{Pa}, \quad \nu_{12} = 0.33, \quad G_{12} = 27 \times 10^{9}\,\text{Pa},$
$\text{Density} = 2800\,\text{kg/m}^3$

To create this material in Patran:

1. Navigate to `Action > Create Object > Isotropic Method > Manual Input`.
2. Specify the material name as `Aluminium`.
3. Click on `Input Properties` and assign the above values.
4. Apply the settings to finalize the aluminum material definition.

Core Material The core material is defined as a 2D orthotropic material with the equivalent properties provided by the guide:

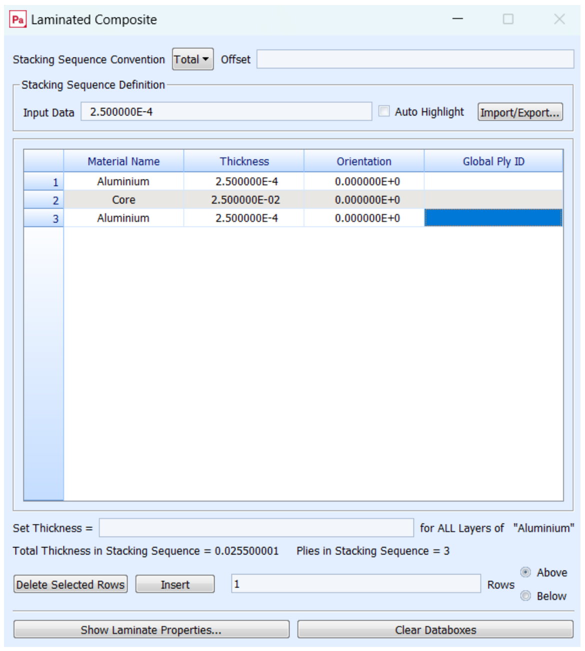
$E_{11} = 1 \times 10^{4}\,\text{Pa}, \quad E_{33} = 1 \times 10^{4}\,\text{Pa}, \quad \nu_{13} = 0.3, \quad G_{13} = 1 \times 10^{4}\,\text{Pa},$
$G_{12} = 1.85 \times 10^{8}\,\text{Pa}, \quad G_{32} = 0.89 \times 10^{8}\,\text{Pa}, \quad \text{Density} = 32\,\text{kg/m}^3$

To define the core material:

1. Navigate to `Action > Create Object > 2D Orthogonal`.
2. Specify the material name as `Core`.
3. Click on `Input Properties` and assign the above values.
4. Apply the settings to create the core material.

Creating the Composite Material The next step is to create a composite material by combining the face sheets and core layers:

1. Navigate to `Action > Create Object > Composite Method > Laminate`.
2. In the window that appears, add the layers in the following order: `Aluminium (Top Face Sheet)`, `Core`, `Aluminium (Bottom Face Sheet)`.
3. Specify the thickness of each layer:(as your requirement)
   - Aluminum Face Sheet Thickness $= 0.25 \times 10^{-3}$ m
   - Core Thickness $= 25 \times 10^{-3}$ m
4. Set the orientation to $0°$ for all layers.
5. Click `Apply` to create the composite material.



Stacking Sequence Convention: Total   Offset:

Stacking Sequence Definition

Input Data: 2.500000E-4   [ ] Auto Highlight

	Material Name	Thickness	Orientation	Global Ply ID
1	Aluminium	2.500000E-4	0.000000E+0	
2	Core	2.500000E-02	0.000000E+0	
3	Aluminium	2.500000E-4	0.000000E+0	

Set Thickness = for ALL Layers of "Aluminium"

Total Thickness in Stacking Sequence = 0.025500001   Plies in Stacking Sequence = 3

1 Rows [✓ Above] Below

Figure 2.4: Creating the composite Material

Assigning the Composite Material Finally, the composite material is assigned to the surface of the sandwich plate:

1. Navigate to `Action > Create Object > 2D Type > Shell`.

2. Specify the property name and set the option to `Thin > Laminate > Standard Formulation`.
3. Click on `Input Properties`, select the material as `Composite`, and assign it to the surface.
4. Apply the settings to complete the assignment of the composite material properties.

Analysis Setup and Execution

To perform the frequency analysis of the sandwich plate model, follow these steps:

1. **Set Up the Analysis:**
   - Navigate to the `Analysis` menu.
   - Select `Action > Analyze Object > Selected Group(s).`
   - In the `Method` dropdown, choose `Analysis Deck`.
2. **Select the Solution Type:**
   - Click on `Solution Type` and select `Normal Modes` for frequency analysis.
   - Apply the settings to generate the `.bdf` file, which serves as the input for Nastran software.
3. **Run the Analysis:**
   - Execute the analysis by clicking on `Run`. Nastran will process the `.bdf` file and generate an `.xdb` file containing the results.
4. **Access the Results in Patran:**
   - In the `Analysis` menu, select `Action > Access Results > Attach XDB`.
   - Choose `Method > Result Entities` and click on `Select Result File`.
   - Locate and select the `.xdb` file from your device to attach the results.
5. **Visualization of Results:**

- In the `Results` menu and select `Action > Create Object > Quick Plot`.
- Choose the mode you wish to visualize.
- For fringe results, select `Result Type > Eigenvalue, Translation`.
- For deformation results, select `Result Type > Deformation`.
- Click `Apply` to visualize the results in the Patran window.

2.4 Parameterization of the Sandwich Plate Using PCL

A major challenge I faced during my internship involved the parameterization of the sandwich plate model. The objective was to streamline the modeling process in MSC Patran, allowing for the creation of models with varying dimensions, face sheet thicknesses, core thicknesses, and materials with just one click. Our instructor familiarized us with the background coding language of MSC Patran, known as Patran Command Language (PCL), which was crucial for accomplishing this task. Additionally, we were given access to a tutorial from a workshop held in 2005 to aid our understanding. [8].

2.4.1 Learning PCL and Section Files

Our exploration commenced with an introduction to PCL and an examination of how Patran produces section files that correspond to user interactions within the graphical user interface. At first, interpreting these section files proved to be difficult, as they encompassed PCL code that represented each action performed in the GUI. We needed to acquire the skill to pinpoint the particular lines of code associated with tasks like the creation of points, lines, and surfaces.

Step-by-Step Learning Process

To build our understanding, we adopted a systematic approach:

- At the outset, our strategy for learning PCL was systematic, adhering closely to the sequential guidelines outlined in the Patran user manual. Although this approach was comprehensive, it turned out to be quite time-intensive. As we advanced, we discovered that examining the section files produced by Patran was a more effective method. Through the analysis of these files, we were able to directly see how particular tasks were implemented in PCL, thereby enhancing our comprehension and capacity to adapt the code to suit our requirements.

- One of the most significant hurdles was getting the edited section file to run successfully in Patran. Initially, attempts to execute the modified section files directly resulted in errors. We discovered that the correct procedure involved opening a new file in Patran, running the section code, and then executing subsequent modeling commands.
- We created a point in the Patran GUI and analyzed the resulting section file to see how the point was defined in PCL.
- We repeated this process with different points, lines, and surfaces, gradually mapping GUI actions to the corresponding PCL code.
- The next step involved defining variables in the PCL code. Understanding how to call variables in functions, such as creating a line, was particularly challenging due to unfamiliar syntax. It took us a full day to grasp that variables are called using the backtick ("`") symbol.
- A further challenge encountered was the management of file paths necessary for the submission of analysis jobs. Given that file paths vary between devices, this presented a significant obstacle that required a considerable investment of time. The resolution entailed encapsulating the analysis code within a function in PCL, which enabled the software to automatically modify the file path according to the device in use. This method greatly enhanced the efficiency of the process, rendering the code more resilient and adaptable.

2.4.2 Step-by-Step Explanation of PCL Code

This section provides a detailed explanation of the PCL code used to automate the modeling and analysis of the sandwich plate. The code is organized into six distinct functions, each responsible for a specific aspect of the process.

Defining Variables

The PCL code begins by defining several key variables, which control the dimensions, material properties, and other parameters of the model. These variables are essential for ensuring that the model can be easily adjusted to meet different design requirements. The complete code for defining variable can be found in Appendix A.1

Function 1: `hcp_create()`

The function `hcp_create()` is tasked with the creation of the geometry for the sandwich plate. It utilizes the specified variables to produce the required points, lines, and surfaces. The process encompassed within this function consists of the following steps:

```
uil_file_new.go( "", "vvv.db")
```

This line will open new file in patran and name of file will be vvv.db

```
asm_const_line_2point("1", "[0 0 0]", "[`length` 0 0]", 0,
   "", 50., 1, asm_line_2point_created_ids)
```

This function establishes a line segment by utilizing two distinct points.
`"[0 0 0]"`: These are the coordinates for the first endpoint of the line segment.
`"[`length` 0 0]"`: These coordinates represent the second endpoint of the line segment, where `length` is a variable denoting the distance along the x-axis. This parameterization enables adjustments to the line's length by modifying `length`.
The `@` symbol indicates that the command continues on the subsequent line, ensuring that the PCL compiler interprets the entire command as a single, uninterrupted instruction.

```
STRING sgm_surface_4edge_created_ids[VIRTUAL]
sgm_const_surface_4edge("1","Curve 4","Curve 1","Curve 2",@
"Curve 3",sgm_surface_4edge_created_ids )
$# 1 Surface Created: Surface 1
```

This function creates a surface bounded by four curves here it is 1 to 4. The complete code for creating geometry can be found in Appendix A.2

Function 2: `hcp_mesh()`

The `hcp_mesh()` function handles the meshing of the created geometry. It applies a mesh to the surface based on the specified element size.

```
fem_create_mesh_surf_4("IsoMesh",49152,"Surface 1",1,@
["0.05"],"Quad4","#","#","Coord 0","Coord 0",@
   fem_create_mesh_surfa_num_nodes,fem_create_mesh_@
surfa_num_elems,fem_create_mesh_s_nodes_created,@
fem_create_mesh_s_elems_created)
```

["0.05"]: The target element size for the mesh; here, the elements will aim to have a size of 0.05 units, this can be adjust to as you need. The complete code for mesh can be found in Appendix A.3

Function 3: **hcp_load()**

The hcp_load() function is used to apply loads and boundary conditions to the model. This function includes:

```
FUNCTION hcp_load()
loadsbcs_create2("FIXED","Displacement","Nodal","",@
"Static",["Surface 1.1 1.2 1.3 1.4"],"Geometry",@
"Coord 0","1.",["<0 0 0>","<0 0 0>", "<     >", "<     >"]@
, ["", "", "", ""] )
END FUNCTION
```

["Surface 1.1 1.2 1.3 1.4"]: Indicates the nodes on the edges of "Surface 1" (1.1, 1.2, 1.3, 1.4) that will be fixed.
["<0 0 0>", "<0 0 0>", "< >", "< >"]: Defines the displacement, rotational, trans phase and rotational phase respectively constraints in each direction (X, Y, Z). Here, the nodes are fixed in the X, Y, and Z directions with no displacement and rotation allowed ('<0 0 0>') you can adjust to your desire condition.
The complete code for load/bc can be found in Appendix A.4

Function 4: **hcp_material()**

The hcp_material() function defines the material properties for both the face sheets and the core. To effectively generate materials in MSC Patran, initiate the process by defining the material within the graphical user interface (GUI) see section 2.3.3 for reference and subsequently saving the section file. Next, open the section file and identify the function designated for material creation. Substitute the fixed numerical values for properties such as Young's Modulus and Poisson's Ratio with predefined variable names (for instance, 'e_11' for Young's Modulus in the 11 direction and 'u_12' for Poisson's Ratio in the 12 direction). Prior to this, ensure that these variables are defined with their respective numerical values in your PCL script. This methodology facilitates dynamic and parameterized simulations, permitting easy adjustments to material properties as required within your finite element model.The complete code for material can be found in Appendix A.5

Function 5: `hcp_composite()`

The methodology employed for material creation can also be utilized to define composite materials through the development of variable-based PCL code. Initially, configure the properties of the composite material using the graphical user interface (GUI). Subsequently, access the section file to substitute fixed numerical values with predefined variables, such as ‘face_sheet_thickness‘ for the face sheets and ‘core_thickness‘ for the core. By assigning these variables within your PCL script, you will enable a more flexible and dynamic approach to defining composite materials.The complete code for composite can be found in Appendix A.6

Function 5: `hcp_analysis()`

To conduct the analysis, begin by submitting the file for normal mode analysis through the graphical user interface (GUI). Subsequently, navigate to the section file to extract the necessary code and integrate it into your PCL script. It is advisable to develop a function for the analysis within your PCL code, as the modifications required for establishing the analysis are minimal.The complete code for analysis can be found in Appendix A.7
At last, after making all functions, call all functions for executing code.

2.5 Automating the Analysis Process

The section file for Patran being prepared marks the beginning of the automation of the analysis process, aimed at enhancing workflow efficiency. The following steps were implemented to accomplish this objective:

1. **Management of Variables**: To facilitate variable management within the PCL section file, we initiated the process by defining the required variables in a distinct ‘.dat‘ file. This file is subsequently imported into the section file, thereby rendering the PCL code more accessible and simpler to modify.
2. **Execution of Patran and Nastran**: Following the variable definitions, we execute Patran to create the ‘.bdf‘ file, which is then submitted to Nastran. This procedure is automated to produce the ‘.f06‘ and ‘.xdb‘ files. The ‘.xdb‘ file serves as a visualization tool in Patran, while the ‘.f06‘ file contains the essential frequency data.
3. **Frequency Data Extraction**: In our initial attempts to extract frequency data from the ‘.f06‘ file, we considered using Python. However, due to the absence of inter-

net access and Python installation at ISRO, we decided to adopt a command-line method. We developed a CMD script to facilitate this task. The script processes the '.dat' file, updates the section file with the defined variable values, executes Patran to generate the '.bdf' file, and subsequently submits it to Nastran. After the '.f06' file is generated, the script extracts the frequency table and saves the data into a separate file.

4. **CMD Scripting Experience**: Acquiring skills in CMD scripting was a novel endeavor that presented challenges, particularly in the extraction of specific data from the '.f06' file. Nevertheless, through multiple iterations and thorough research, we successfully developed a CMD script that automates the entire workflow in a single action, from updating the section file to retrieving the frequency values.

2.5.1 Challenges and Solutions

Throughout the automation process, various challenges emerged, prompting the implementation of the following solutions:

1. **Executing Software in the Background:** A primary challenge involved executing the software in the background, necessitating the identification of the executable file and ensuring compatibility across multiple devices. To tackle this issue, the executable path was established as a variable within the '.dat' file. This method enhanced the script's adaptability to different devices by permitting dynamic updates to the path, thereby promoting a more versatile workflow. Although determining and executing this solution was time-intensive, it was essential for the success of the automation process.
2. **Managing Variable Locations of Section Files:** Another challenge pertained to the varying locations of the section file across different devices. To address this, the directory path for the section file was similarly defined in the '.dat' file. This solution required significant effort to ensure that the script could accurately locate the file, regardless of its position, thereby streamlining the process and minimizing errors associated with file paths.
3. **Interpreting and Handling the '.dat' File:** The necessity to specify the directory of the '.dat' file introduced an additional layer of complexity. This step demanded meticulous management to guarantee that the script could effectively read and update the file paths.

4. **Data Extraction from the ‘.f06‘ File:** The extraction of the frequency table from the ‘.f06‘ file posed a challenge due to the intricacies of CMD syntax. Overcoming this obstacle required extensive research and consultation of online resources, including tutorials on YouTube. After numerous iterations, a functional CMD script was successfully developed to automate the extraction process efficiently.

   These challenges necessitated a blend of research, experimentation, and determination to resolve, ultimately resulting in a viable and automated solution for the analysis process.

2.5.2 CMD Script for Automating the Analysis Process

The provided CMD script automates several steps in the analysis process, including updating the ‘.ses‘ file, running Patran and Nastran, and extracting data from the ‘.f06‘ file. The complete code for CMD script can be found in Appendix A.8

1. **Reading the Input File:**The script starts by setting the path to the ‘HCP.DAT‘ file, which contains the necessary variables. It uses a ‘for /f‘ loop to read the input file line by line, parsing the variable names and values, and setting them as environment variables.

2. **Updating the ‘.ses‘ File:** The script then updates the ‘.ses‘ file with the variables defined in the ‘HCP.DAT‘ file. It reads the ‘.ses‘ file line by line, and if the line begins with ‘global‘, it replaces it with a new line that includes the updated variable values. This updated content is saved to a temporary file, which is then moved to replace the original ‘.ses‘ file.

3. **Running Patran:** After updating the ‘.ses‘ file, the script executes Patran in batch mode using the updated ‘.ses‘ file. The ‘-b‘ flag runs Patran in batch mode, while ‘-graphics‘ and ‘-sfp‘ specify the graphics and session file parameters, respectively.

4. **Running Nastran:** The script proceeds to run Nastran with the generated ‘.bdf‘ file. It pauses for 15 seconds to ensure Nastran has enough time to start properly.

5. **Extracting Data from the ‘.f06‘ File:** Finally, the script extracts frequency data from the ‘.f06‘ file. It looks for the table titled "REAL EIGEN VALUES" and processes the lines following this title. The script cleans up spaces in the data and converts it to CSV format, which is saved to a temporary file.

6. **User Interaction:** Throughout the script, ‘pause‘ commands are used to prompt the user to wait or take action, ensuring that each step completes before proceeding to the next.

2.6 Running the Automation Script with Your Own Dimensions

To run the automation script with your own dimensions and properties, follow these steps:

1. **Create the DAT File:** Copy the DAT file code from Appendix A.9. Save this code to a new ‘.dat‘ file at your desired location.
2. **Verify Patran and Nastran Installation:** Ensure that Patran and Nastran are properly installed and functioning on your system.
3. **Update Directory and Executable Paths:** Open the newly created DAT file. Replace the ‘Directory‘ variable with the path where you saved the DAT file. Similarly, find the executable paths for Patran and Nastran by right-clicking the Patran icon on your desktop, selecting ‘Properties‘, and copying the target location. Update these paths in the DAT file.
4. **Place Section File:** Copy the section file (‘.ses‘) into the same directory where you saved the DAT file.
5. **Create and Configure the CMD Script:** Copy the CMD script code from the Appendix A.8 into a new ‘.bat‘ file. Set the ‘input_file‘ variable in the ‘.bat‘ file to point to the location of your DAT file.
6. **Save and Execute:** Save all the files (DAT file, section file, and BAT file). Double-click the ‘.bat‘ file to run the script. This will execute Patran and Nastran, generating a CSV file with the frequency values.

By following these steps, you will be able to automate the analysis process with your specified dimensions and properties, generating the required frequency values in a CSV file (cycle column will the frequency value for ten modes). The mode shapes can be visualized directly within Patran. After generating the ‘.xdb‘ file, you should open it in Patran and link the ‘.xdf‘ file to observe the mode shapes. This process will enable you to examine the deformation patterns and mode shapes of your model.

2.7 Results and Verification

To validate the results obtained from the finite element analysis (FEA) of the honeycomb sandwich plate, the study by Aborehab et al. [4] was used as a reference. This paper investigates a square honeycomb sandwich plate with a side length of 400 mm and a total thickness of 10 mm. The sandwich structure consists of an 8 mm thick honeycomb core made of AL-5052, with two 1 mm thick aluminum face sheets made of AL2024-T3. The material specifications of the honeycomb core and face sheets are presented in Tables 2.1.

Table 2.1: Material properties of the honeycomb core (AL-5052) and facing sheets (AL2024-T3) [4].

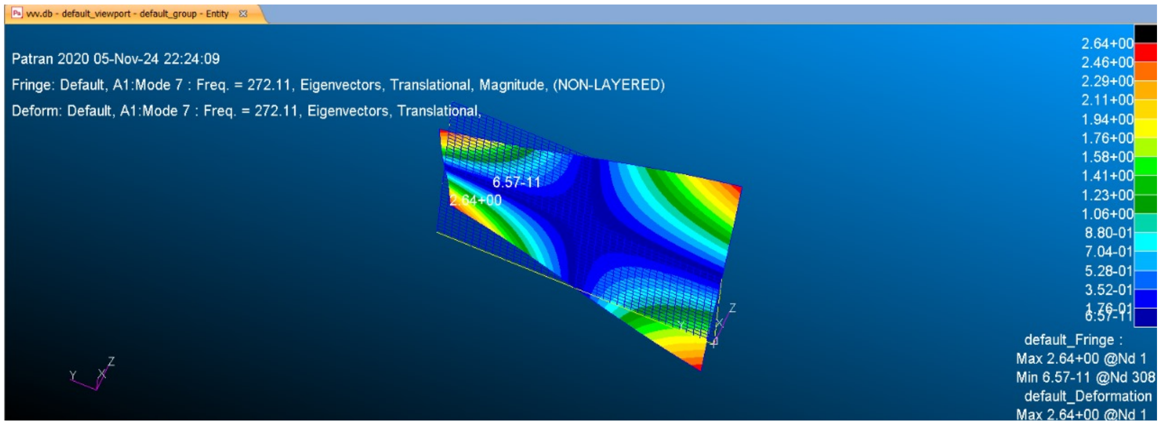
Property	Core Material (AL-5052)	Facing Sheets (AL2024-T3)
Elastic Modulus, E	70 GPa	70 GPa
Density, ρ	130 kg/m^3	2800 kg/m^3
Poisson's Ratio, ν	0.33	0.33

In this study, the third approach (SPS) mentioned in section 2.1 FEM of Honeycomb Sandwich Plate Using Sandwich Theory [4] was adopted, where the honeycomb sandwich structure is modeled as a 2D sandwich plate shell using shell finite elements (SHELL181). The equivalent elastic parameters of the honeycomb core using sandwich plate theory 2.2.1 were used as given in Table 2.2. The DAT file was modified accordingly (see Appendix A.9) to match these properties and run the analysis.

Table 2.2: Equivalent material properties of the honeycomb core [4].

Property	Value
E_{11}	3.3 MPa
E_{22}	2944.6 MPa
E_{33}	3.3 MPa
G_{12}	426 MPa
G_{13}	1.98 MPa
G_{23}	651.7 MPa
ν_{12}	0.99

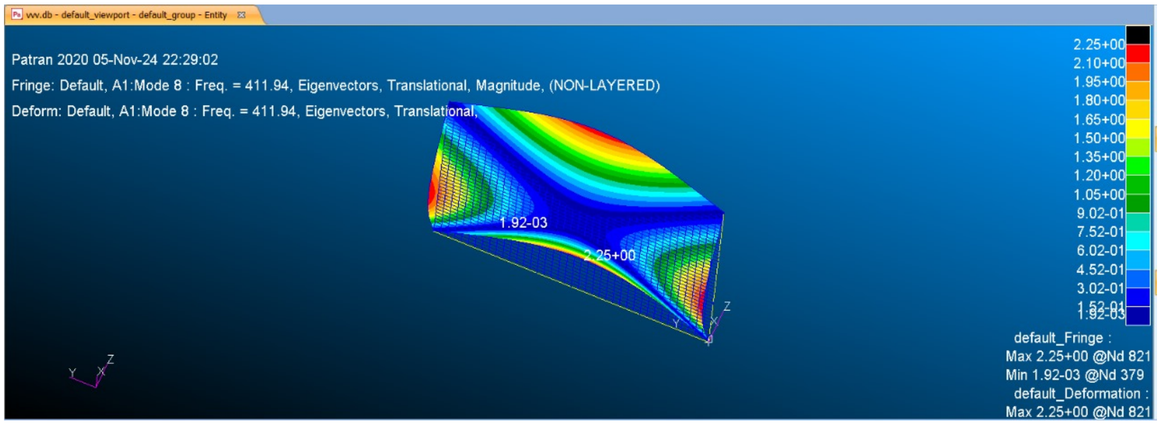
After running the modified data file, the obtained frequency values for the first four modes were found to closely match the experimental and theoretical results presented in [4]. Screenshots of the first four mode shapes are shown in Figures 2.5-2.8.



```
Patran 2020 05-Nov-24 22:24:09
Fringe: Default, A1:Mode 7 : Freq. = 272.11, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 7 : Freq. = 272.11, Eigenvectors, Translational,
```

```
default_Fringe :
Max 2.64+00 @Nd 1
Min 6.57-11 @Nd 308
default_Deformation :
Max 2.64+00 @Nd 1
```

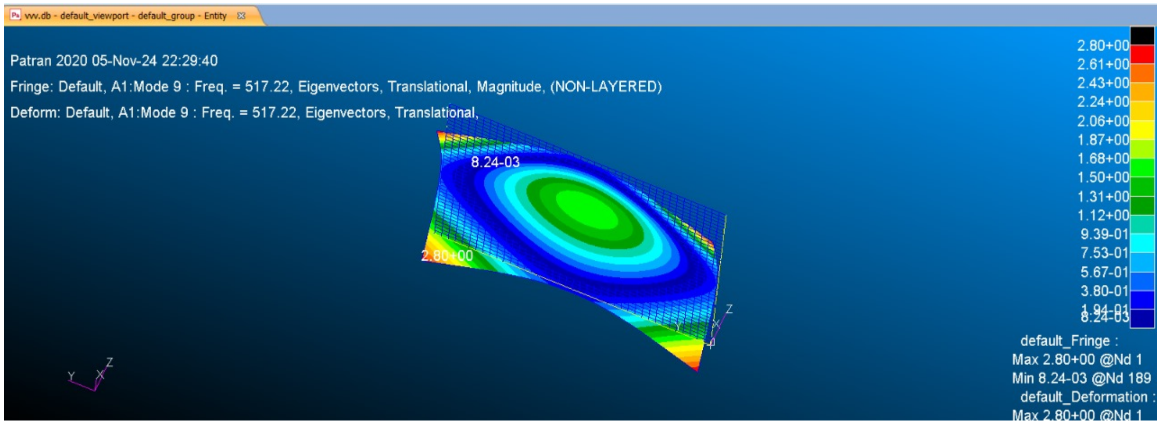
Figure 2.5: Mode 1: Torsion



```
Patran 2020 05-Nov-24 22:29:02
Fringe: Default, A1:Mode 8 : Freq. = 411.94, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 8 : Freq. = 411.94, Eigenvectors, Translational,
```

```
default_Fringe :
Max 2.25+00 @Nd 821
Min 1.92-03 @Nd 379
default_Deformation :
Max 2.25+00 @Nd 821
```

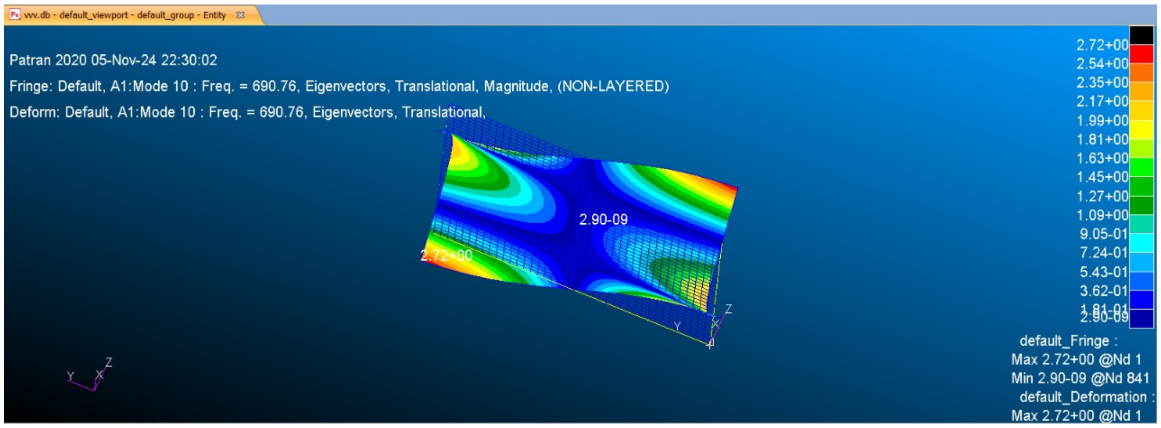
Figure 2.6: Mode 2: Bending



```
Patran 2020 05-Nov-24 22:29:40
Fringe: Default, A1:Mode 9 : Freq. = 517.22, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 9 : Freq. = 517.22, Eigenvectors, Translational,
```

```
default_Fringe :
Max 2.80+00 @Nd 1
Min 8.24-03 @Nd 189
default_Deformation :
Max 2.80+00 @Nd 1
```

Figure 2.7: Mode 3: Bending


vw.db - default_viewport - default_group - Entity

Patran 2020 05-Nov-24 22:30:02
Fringe: Default, A1:Mode 10 : Freq. = 690.76, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 10 : Freq. = 690.76, Eigenvectors, Translational,

default_Fringe :
Max 2.72+00 @Nd 1
Min 2.90-09 @Nd 841
default_Deformation :
Max 2.72+00 @Nd 1

Figure 2.8: Mode 4: Torsion

2.7.1 Theoretical Analysis

The theoretical analysis of the square plate under free-free boundary conditions was carried out using the discrete singular convolution (DSC) method [4]. The frequency parameter, λ, for the first four modes is calculated using:

$$\lambda = \omega a^2 \sqrt{\frac{m}{D}}, \tag{2.9}$$

Where λ is the frequency parameter, ω is the circular frequency, a is the plate side length (400 mm), ρ is the plate mass per unit area, and D is the plate flexural rigidity. Based upon the DSC method, the values of the frequency parameter for the first four modes of a square plate with free-free boundary conditions are listed in Table 2.3 according to the published data.[4]

Table 2.3: Frequency parameter for a square plate with free-free boundary conditions [4].

Mode Order	**First Mode**	**Second Mode**	**Third Mode**	**Fourth Mode**
Frequency Parameter, λ	13.419	19.559	24.212	34.646

2.7.2 Comparison of Results

This section presents a detailed comparison between the experimental results, numerical results obtained from MSC Patran using Sandwich Panel Theory (SPS), and theoretical predictions derived from literature. The aim is to validate the accuracy and reliability of the computational models by analyzing the deviation of natural frequencies from experimental data.

Experimental vs. Computational Natural Frequencies

The comparison between experimental and computational natural frequencies concerning different approaches to sandwich theory is summarized in Table 2.4. The table includes the mode order, mode type, and corresponding frequencies, along with the percentage deviation of the numerical results from the experimental values.

Table 2.4: Experimental and computational natural frequencies comparison concerning sandwich theory approaches

Mode Order	**Mode Type**	**Experimental Frequency (Hz)[4]**	**SPS From Paper[4]**	**Deviation (%)**	**SPS From Patran**	**Deviation (%)**
1	Torsion	259	283.7	9.56	272.11	5.06
2	Bending	391	426.99	9.2	411.94	5.35
3	Bending	494	535.33	8.37	517.22	4.7
4	Torsion	664	721.55	8.67	690.76	4.03
Mean Deviation (%)	-	-	-	8.95	-	4.785

The numerical frequencies obtained from the SPS model show a relatively small deviation from the experimental values, with the average deviation being around 9% for the results from the literature and 4.785% for the results obtained using Patran. The higher deviation observed in the Patran results could be attributed to the specific modeling assumptions or simplifications made during the setup of the computational model.

Comparison with Theoretical Values

In addition to comparing the experimental and computational results, it is also essential to evaluate how these results align with the theoretical predictions provided in the literature. Table 2.5 presents a comparison of the natural frequencies with theoretical values, again focusing on the same modes.

From the data in Table 2.5, it can be observed that the deviation from theoretical values is generally lower compared to the experimental deviations. The mean deviation is approx-

Table 2.5: Comparison of natural frequencies with theoretical values

Mode Order	Mode Type	Theoretical Frequency (Hz)[4]	SPS From Paper[4]	Deviation (%)	SPS From Patran	Deviation (%)
1	Torsion	270.9	283.7	4.75	272.11	0.4
2	Bending	394.85	426.99	8.14	411.94	4.32
3	Bending	488.78	535.33	9.5	517.22	5.82
4	Torsion	699.4	721.55	3.167	690.76	1.23
Mean Deviation (%)	-	-	-	6.4	-	2.94

imately 6.4% for the literature values and 2.94% for the Patran results, which indicates that the theoretical models are closer to the experimental values.

Discussion

The evaluation of the results reveals that both the SPS models referenced in the literature and the Patran models offer a fairly precise depiction of the natural frequencies of the sandwich structure. The marginally greater deviation observed in the Patran results may be attributed to several factors, including:

- **Modeling Assumptions:** The assumptions established during the creation of the finite element model, such as boundary conditions and material characteristics, may have introduced a degree of error.
- **Mesh Density:** The mesh density applied in Patran could influence the precision of the outcomes. A denser mesh typically results in more accurate findings, albeit with increased computational demands.
- **Material Properties:** The specific material properties utilized in the theoretical models may differ slightly from those employed in the experimental framework, resulting in discrepancies in the findings.

Although the average deviation in the results is somewhat higher in Patran, it remains within an acceptable range for engineering purposes, thereby supporting the applicability of these models for predictive analysis.

Chapter 3

Hexagon Interstage

Three major rib types are combined to form a typical composite grid cylindrical shell. The shell's buckling behavior and structural efficiency under axial compression are then investigated using finite element (FE) models. The FE models are created by parametric modeling, which aligns beam elements' cross-sections and preserves their natural twisted geometry. The technique is further parameterized and coded using Patran Command Language (PCL). The FE modeling demonstrations demonstrate the program's efficiency in generating FE models, making parametric studies and grid shell design easier. The study contrasts buckling mode shapes and provides insight into the composite grid of cylindrical shells, which can aid in their initial design.

3.1 Introduction

A lattice of stiff, interconnected ribs composed of a high-specific-strength, unidirectional composite material characterizes composite grid cylindrical shells. The ribs serve as the primary load-bearing components in these structures, giving the shells membrane and bending stiffness. There are three primary classifications for these ribs: longitudinal, circumferential, and helical ribs. Numerous grid patterns can be created using three different rib types. The most popular grid patterns, though, are triangle and hexagon, which are widely utilized in structural applications like payload adapters for spacecraft launchers and rocket interstages.Here we will be studying the hexagon grid pattern.

Grid cylindrical shells exhibit the mechanical behavior of thin-walled structures because the diameter of the grid shells is much larger than the cross-sectional dimensions of the ribs. As a result, buckling under different loadings is one of the common failure modes of grid shells. When designing and conducting buckling analyses of composite grid structures, two distinct analytical methods are typically utilized. We will be using the approach that the analysis of the buckling of grid structures is based primarily on the FE model. Solid, shell, and beam element models are normally adopted. The finite element modeling and buckling analysis of composite grid cylindrical shells using a parameterized approach is examined in the 2014 paper "Parameterized Finite Element Modeling and Buckling Analysis of Six Typical Composite Grid Cylindrical Shells" by Changliang Lai, Junbiao Wang, and Chuang Liu [2]. It was published in Applied Composite Materials. These shells are examined under different loading scenarios, including axial compression, pure bending, torsion, and transverse bending. They are constructed of unidirectional composite materials arranged in a grid pattern. The Patran Command Language (PCL) was used to generate the finite element models, enabling effective model generation and thorough examination of the shell structures.

The impact of various grid patterns, such as hexagonal and triangular configurations, on the buckling behavior and structural efficiency of the shells is the main focus of this study. In order to provide more accurate buckling analysis, the paper compares detailed finite element models with solid, shell, or beam elements with mathematical models that employ simplified stiffness representations. In geometric modeling, beam elements were assigned specific cross-sections and orientations, and helical curves and circumferential lines were drawn.

The buckling analyses' findings demonstrated how various helical angles and grid patterns impact buckling behavior and structural efficiency. In particular, the study discovered that the grid pattern and helical angle affect the critical load and efficiency ratios. The study demonstrated how distinct patterns displayed distinct buckling modes under axial compression and how variations in critical moments and bending efficiencies were revealed by pure bending analysis. Torsion and transverse bending were also included in the analysis, giving a thorough understanding of the structural performance.

The paper's main conclusions include determining the best grid patterns and helical angles to maximize structural efficiency. The mixed triangle grid shell proved to be especially useful in the circumstances that were examined. An accurate and efficient analysis was made possible by the parameterized modeling approach, which also provided insightful information for the initial design of composite grid cylindrical shells. The study contains comprehensive tables and figures that show the efficiency ratios, critical loads, and buckling modes for the different grid patterns and angles that were examined.

3.1.1 Structural Features

Helical, circumferential, and longitudinal ribs are the fundamental elements of grid structures, and various grid patterns can be constructed using these basic ribs. As illustrated in Fig. 1, six typical grid patterns are analyzed for their buckling behavior in this study. Among these, the rhombus grid pattern is composed of the helical rib. The triangle, rotated triangle, and rectangle grid patterns are composed of two types of basic ribs, with the rotated triangle grid pattern being a 90° rotation of the triangle grid pattern. The mixed triangle grid pattern is unique as it is composed of three types of basic ribs and is used to demonstrate parameterized finite element (FE) modeling in the subsequent section. Additionally, the hexagon grid pattern is derived by offsetting the circumferential ribs of the triangle grid pattern from the nodes by half the nodal spacing. Nodes are the regions where two or more ribs intersect. Theoretically, an infinite number of grid patterns can be created by altering the position of circumferential or longitudinal ribs relative to the nodes with varying distances, though such patterns are not commonly observed or typical [2].

A grid cylindrical shell is defined by multiple design variables, as depicted in Fig. 3.1. However, there are eight fundamental geometrical variables: the overall height (H) and diameter (D); the helical angle (ϕ) between the helical ribs and the shell meridian; the number of ribs (n) corresponding to a single helical direction (either $+\phi$ or $-\phi$); the shell thickness (t); and the widths of the helical (b_h), circumferential (b_c), and longitudinal ribs (b_l). These eight basic geometrical variables can be used to express other design variables. For instance, the spacing between circumferential ribs (a_c) can be calculated using the following equation:

$$a_c = \frac{\pi D}{2n \tan\phi} \tag{3.1}$$

The corresponding number of circumferential ribs (n_c) is given by:

$$n_c = \frac{2nH \tan\phi}{\pi D} \tag{3.2}$$

These relationships are critical in defining the geometry and structural characteristics of the grid cylindrical shell [2].

3.1.2 Applications of hexagon interstage.

Hexagon interstage components are widely used in industrial processes, automotive, aerospace, and other industries. The following are a few uses for hexagon interstage components:

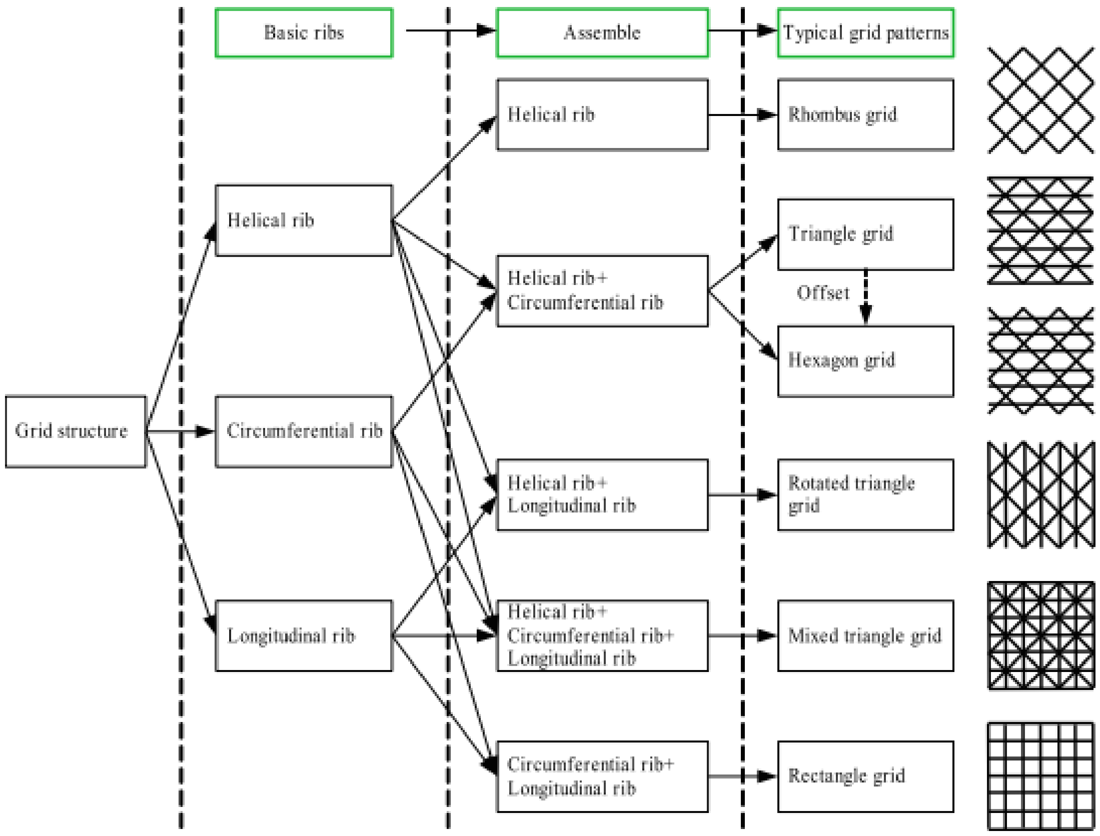


Figure 3.1: Typical grid patterns analyzed for buckling behavior [2].

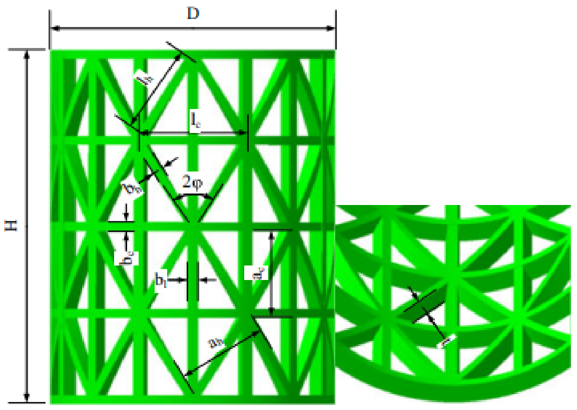


Figure 3.2: Basic geometrical variables of a grid cylindrical shell [2].

3.1.2.1 Aerospace engineering:

In the aerospace industry, a hexagon interstage usually refers to a structural element that joins various stages of a rocket or spacecraft. It frequently has a hexagonal cross-section, which offers benefits in terms of weight distribution, strength, and stability. Applications of hexagonal interstage in aerospace:

1. Structural Integrity: A strong framework that can endure the high loads and stresses experienced during rocket launches and space operations is provided by the hexagonal shape. By ensuring an even distribution of forces, this geometry lowers the possibility of structural failure.

2. Weight optimization: Reducing weight without sacrificing strength is essential in aerospace design. The vehicle's overall mass can be decreased with the aid of a hexagonal interstage, which provides an effective structure that optimizes strength-to-weight ratio and is essential for increasing payload capacity and fuel efficiency.

3. Modular Design: Fuel tanks, electronics, and payloads can all be integrated more easily thanks to the hexagonal shape. Because of its modularity, spacecraft designs are more flexible and scalable, which makes assembly and maintenance more effective.

4. Thermal Management: Controlling the distribution of heat can also be influenced by the interstage's shape. Hexagons' uniform surface area can aid in more evenly dissipating heat, which is crucial for safeguarding delicate components during launch and spaceflight.

5. Aerodynamics: The rocket's aerodynamics may be impacted by the interstage design. During rocket flight, a well-engineered hexagonal interstage can help lower drag and increase the rocket's overall aerodynamic efficiency.

3.1.2.2 Turbomachinery:

Compressors and turbines are the main applications for hexagonal interstages in the turbomachinery sector. An interstage is a part of a turbomachine that joins different stages together, like the stages of the compressor and turbine. Applications of hexagonal interstage in turbomachinery:

1. Structural Support: The turbomachine's revolving and stationary parts are supported by a sturdy structure that is derived from the hexagonal shape. This guarantees that the equipment runs efficiently at high temperatures and pressures.

2. Flow Management: Effectively managing the working fluid (such as air, steam, or gas) between stages is essential in turbomachinery. The energy of the fluid can be efficiently transferred from one stage to the next by channeling the flow with the least amount of loss possible with the aid of a hexagonal interstage.

3. Weight and Space Efficiency: Hexagonal interstages' compact design makes better use of internal space within machinery possible, which is important in sectors like aviation or power generation where size and weight restrictions are strictly enforced. 4. Ease of Assembly and Maintenance: Turbomachinery assembly and disassembly can be made simpler

by the hexagonal interstage's modular design. This is especially helpful for maintenance because it makes internal components easier to reach, which minimizes downtime.

5. Vibration Reduction: A hexagonal interstage's symmetry can aid in the equal distribution of forces, which helps to lessen vibrations while the device is operating. This is vital for the durability and dependability of turbomachinery since high vibrations can cause component failure and premature wear.

6. Thermal management: The interstage's design can affect the cooling efficiency in high-temperature applications, like gas turbines. Improved heat distribution from a hexagonal shape can shield the machine from thermal stress.

3.2 Modeling Process in MSC Patran

3.2.1 Problems and challanges occured during modeling:

Hexagon interstage is a very complicated structure.This structure needs maximum accuracy because if there is a little bit error then your model will not be modeled acccurately.Following are the major problems we faced:

1. Complex Geometry: Accurately modeling the hexagonal shape with its complex features can be difficult, particularly if the shape has varying thicknesses or complex geometric details.
2. Patran: Our first challange was to learn patran as it was a complicated software. we also have to learn Patran Command Languge which is a very difficult language.
3. helical lines: initially we were using points methods to create helical line. In this we have a formula for helical lines and using that formula we were generating a helical line. But this method was wrong. After this we used the vsum method of patran which was right.
4. Software Limitations: The software we were using was an old generation software. It was slow and it was not perfect in terms of calculation it was showing very minnor error.
5. Computational Resources: Fine-mesh, complex models may require a lot of processing power, which could limit the capabilities of the hardware or cause lengthy simulation times.

3.2.2 Methodology

Geometry Creation in MSC Patran

The initial phase of modeling the Hexagon interstage involved creating the geometric representation of the plate using MSC Patran's geometry creation tools. The process began with the creation of a centerline, which served as the foundational base for modeling the Hexagon interstage.

1. **Transforming: Creating the Centerline**
   - Navigate to: `Transform > Curve > Translate`.
   - Input direction vector, vector magnitude, and repeat count.
   - In our case, this step divides the line.
2. **Creating the Bottom Circle**
   - Navigate to: `Create > Curve > 2D Arc Angle`.
   - Input radius, start angle, end angle, and center point list.
   - This creates an arc at the bottom of the line.
   - Navigate to: `Transform > Curve > Rotate`.
   - Input rotation angle, offset angle, and repeat count.
   - This will create a circle at the bottom of the line.
3. **Creating the Helical Line**
   - Navigate to: `Transform > Curve > Vsum`.
   - Input origin of vectors 1 and 2, multiplication factors 1 and 2, and select curve lists one and two.
   - In our case, this will create a helical line around the centerline.
4. **Creating a Plane**
   - Navigate to: `Create > Plane > Point Vector`.
   - Input the point list and vector list.
   - This will create a plane at the bottom of the line, facing upwards along the X direction.
5. **Helical Grid**
   - Navigate to: `Transform > Curve > Mirror`.
   - Input mirror normal plane, offset, and curve list.
   - Navigate to: `Transform > Curve > Rotate`.

- Input reference coordinate frame, axis, rotation angle, offset angle, repeat count, and curve list.
- In our case, this will create helical ribs around the centerline formed in the first step.

6. **Delete Centerline**

   - Navigate to: `Delete > All`.
   - Select the centerline and delete it.

7. **Creating Circular Ribs**

   - Navigate to: `Transform > Curve > Translate`.
   - Input reference coordinate, direction vector, vector magnitude, repeat count, and curve list.
   - In our case, this will create a circular surface from the bottom to the top, covering the helical ribs.

Material Properties

1. Navigate to: `Material > Create > Isotropic > Manual Input > Input Properties`.

   - Input the required data in the dialog box.
   - The material has been created. In our case, we are using aluminum.

Create Mesh Seed

1. Navigate to: `Meshing > Create > Mesh Seed > Uniform`.

   - Input the number of elements and curve list.
   - The mesh seed has been created.

Creating Mesh

1. Navigate to: `Create > Mesh > Surface`.

   - Select `Quad`, `ISOMESH`, `QUAD4`, and then select the surface list.
   - The mesh has been created.

Equivalence

1. Navigate to: `Equivalence > All > Tolerance Cube`.
   - Input equivalence tolerance.
   - Equivalence has been performed.

Boundary Conditions

1. Navigate to: `Load/Boundary Conditions > Create > Displacement > Nodal`.
   - Input displacements and select the curve list.
   - In our case, the bottom of the hexagon interstage is fixed.

Load

1. Navigate to: `Load/Boundary Conditions > Create > Force > Nodal`.
   - Input forces in different directions and select the curve list.
   - In our case, we apply a buckling force on the top.

Analysis Setup and Execution

1. Navigate to the `Analysis` menu.
   - Select `Action > Analyze Object > Selected Group(s).`
   - In the `Method` dropdown, choose `Analysis Deck`.
   - Select the `Solution Type`.
   - Click on `Solution Type` and select `Nbuckling Mode` for buckling analysis.
   - Apply the settings to generate the .bdf file, which serves as the input for Nastran software.
   - **Run the Analysis:** Execute the analysis by clicking on `Run`. Nastran will process the .bdf file and generate an .xdb file containing the results.
   - **Access the Results in Patran:** In the `Analysis` menu, select `Action > Access Results > Attach XDB`. Choose `Method > Result Entities` and click on `Select Result File`. Locate and select the .xdb file from your device to attach the results.

- **Visualization of Results:** In the `Results` menu, select `Action > Create Object > Quick Plot`. Choose the mode you wish to visualize. For fringe results, select `Result Type > Eigenvalue, Translation`. For deformation results, select `Result Type > Deformation`. Click `Apply` to visualize the results in the Patran window.

3.3 Parameterization of the Hexagon interstage Using PCL

In the process of developing the PCL code for the hexagon interstage, various challenges emerged, particularly concerning parameterization and the creation of loops. A significant initial challenge was the effective construction of loops, especially when the number of circular ribs (nc) could potentially be a non-integer value. If nc is not an integer, the bottom circular rib fails to align precisely with the triangular structure, leading to a misalignment that can exceed the intended length (for instance, more than 1 meter). This issue is particularly critical when aiming for accurate modeling.

The task involved modeling the hexagon interstage for defined parameters such as height (h), radius (r), number of ribs (n), and rib angle (ϕ). The primary obstacle was the process of curve numbering and selection. Initially, a centerline is established, followed by the generation of nc curves. The quantity of these curves is contingent upon h, r, n, and ϕ. To utilize the 'vsum' command effectively, a vertical midline curve is necessary, which must project onto the base circular curve in accordance with the number of segments in the centerline.

Choosing the correct curve for subsequent operations, such as the application of loads or boundary conditions, presented a considerable challenge. A generalized formula was required to accommodate the variable number of segments and ensure accurate curve selection. For example, when implementing boundary conditions, it was essential to select curves based on specific coordinates, which fluctuated according to the parameterized values.

Through thorough investigation, it was found that the listing option in PCL offered an effective means of selecting curves based on their coordinates. For instance, employing the condition $x = 1$ facilitated the selection of all points at any y and z coordinates. This method proved to be crucial in the precise application of loads and boundary conditions, thus addressing one of the most significant challenges encountered in the PCL code for the hexagon interstage.

3.3.1 Step-by-Step Explanation of PCL Code

This section provides a detailed explanation of the PCL code used to automate the modeling and analysis of the hexagon interstage. The code is organized into six distinct functions, each responsible for a specific aspect of the process.

Defining Variables

The following global variables are defined to parameterize the model: `nc` and `n` are integers representing the number of circumferential ribs and ribs, respectively. The real variables `R`, `H`, `phi`, `bh`, `t`, and `critical_load` are set to specific values, representing the radius (0.5), height (1.0), helical angle (35 degrees), rib width (0.005), shell thickness (0.005), and critical load (1.0), You can assign value to your need.

Function 1: `hex_create()`

```
uil_file_new.go( "", "hexagon_buckling.db")
```

This line will open new file in patran and name of file will be hexagon_buckling.db

```
STRING asm_create_line_xyz_created_ids[VIRTUAL]
asm_const_line_xyz( "1", "<`x` 0 0>", "[0 0 0]", "Coord 0",
asm_create_line_xyz_created_ids )
```

This function will create a line starting from [0,0,0]. ['x' 0 0] this indicates the the line created will be in the x axis and of length 'x'.

```
STRING sgm_transform_curve_created_ids[VIRTUAL]
sgm_transform_translate_v1( "2", "curve", "<1 0 0>", `x`,
   FALSE, "Coord 0",@
`nc-1`, FALSE, "Curve 1",
sgm_transform_curve_created_ids )
```

This will divide the line created into small parts.
Here [1 0 0] indicates the direction vector, 'x' indicates the vector magnitude and , 'nc-1' indicates how many times the line should be devided.

```
STRING sgm_create_curve_2d_created_ids[VIRTUAL]
```

```
sgm_const_curve_2d_arcangles_v1( "`nc+1`", `R`, 0., `i`, "
   Coord 0.1",@
"[0 0 0]", FALSE,
sgm_create_curve_2d_created_ids )
```

It will create a arc at the bottam of the line formed earlier.
Here 'R' represents the radius of the arc from the line, '0.' represents the start angle of the arc, 'i' represents the end angle of the arc and [0 0 0] represents the centre point list.

```
sgm_transform_rotate( "`nc+2`", "curve", "Coord 0.1", `i`,
   0., "Coord 0",@
`n-1`, FALSE, "Curve `nc+1`",
sgm_transform_curve_created_ids )
```

It will rotate the arc formed earlier it make a full circular surface at the bottom of the line.
Here 'coord 0.1' represents the coordinate frame in which the rotation will be done, 'i' represents the rotation angle, and '0.' represents the offset angle, 'n-1' represents the repeat count, and 'nc+1' represents the curve that will be rotated.

```
asm_transform_curve_vsum( "`nc+n+1`", "[0 0 0]", "[0 0 0]",
    [1., 1., 1.],@
[1., 1., 1.], "Curve 1:`nc`", "Curve `nc+1`:`2*nc`",
sgm_transform_curve_created_ids )
```

This will create a helical line starting from the circle formed in the previous step to the height of the line formed earlier.
Here "[0 0 0]", "[0 0 0]" indicates the origin of vector 1 and 2, [1., 1., 1.],[1., 1., 1.] indicates the multipication factor 1 and 2, and "Curve 1:'nc'", "Curve 'nc+1':'2*nc' represents the curve list one and two this is the curve that will make the helical line across the central line and the circle formed earlier.

```
STRING sgm_create_plane_po_created_ids[VIRTUAL]
sgm_const_plane_point_vector( "1", "Point 1", "<1 0 0>",
sgm_create_plane_po_created_ids )
```

This will create a plane at the bottom of the line formed first facing upwards the x axis.
Here "Point 1" indicates the point list from which the plane will be created, "[1 0 0]" indicates the vector list that will represent the vector direction of the plane.

```
sgm_transform_mirror( "`2*n+2*nc+8+1`", "curve", "Coord 0.3
   ", 0., TRUE,@
FALSE, "Curve `n+2*nc+6` `n+2*nc+7`",
   sgm_transform_curve_created_ids )
```

This will create a helical line that is mirror of the helical line we created earlier.
Here "Coord 0.3" indicates Mirror normal plane (the plane in which the mirror will be done), ’0.’ indicates the offset plane, and ‘n+2*nc+6‘ ‘n+2*nc+7‘ represents the curve list that will be mirrored.

```
sgm_transform_rotate( "`2*n+3*nc+8+1`", "curve", "Coord 0.1
   ", w, 0.,@
"Coord 0", k-1, FALSE, "Curve `2*n+2*nc+8+1`:`2*n+3*nc+8` `
   n+2*nc+6`@
`n+2*nc+7`",
sgm_transform_curve_created_ids )
```

This will make a grid of helical line we formed earlier across the centre line we formed first..
Here "Coord 0.1" represents the reference coordinate frame, ’w’ represents the rotation angle of the mirrored lines, ’0.’ represents the offset angle, ’k-1’ represents the repeat count of the curve, and ‘2*n+2*nc+8+1‘:‘2*n+3*nc+8‘ ‘n+2*nc+6‘ ‘n+2*nc+7‘" represents the curve list that will be mirrored.

```
asm_delete_curve( "Curve 2:`nc-1` `2*nc+n+2` `2*nc+n+3`", @
asm_delete_any_deleted_ids )
```

This will delete the centre line that we formed first.
Here ‘nc-1‘ ‘2*nc+n+2‘ ‘2*nc+n+3‘ represents the curves that will be deleted.

```
sgm_transform_translate_v1( "`2*n+3*nc+8+nc*n*2+1-2*nc`", "
   curve", "<1 0 0>",@
`x/2`, FALSE, "Coord 0", `2*nc-1`, FALSE, "Curve `n+2*nc
   +8+1`:`n+2*nc+8+n`",
sgm_transform_curve_created_ids )
```

This will translate the circle that was formed on the bottom of the line we deleted recently.
Here "[1 0 0]" represents the direction vector, ‘2*nc-1‘ represents the repeat count, ‘n+2*nc+8+1‘:‘n+2*nc-
represents the curve that will be translated.

Now the geometry of the hexagon interstage is created, you can fine full code in Appendix B.1.

Function 2: `hex_material()`

The `hex_material()` function gives the material properties to the hexagon interstage.

```
material.create( "Analysis code ID", 1, "Analysis type ID",
    1, "material", 0,@
"Date: 18-Jul-24           Time: 11:48:43", "3d Orthotropic
   ", 2, @
"Directionality", 2, "Linearity", 1, "Homogeneous", 0, "
   Linear Elastic", 1, @
"Model Options & IDs", ["", "", "", "", ""], [0, 0, 0, 0,
   0], "Active Flag", @
1, "Create", 10, "External Flag", FALSE, "Property IDs", ["
   Elastic Modulus @
11", "Elastic Modulus 22", "Elastic Modulus 33", "Poisson
   Ratio 12", "Poisson@
Ratio 23", "Poisson Ratio 31", "Shear Modulus 12", "Shear
   Modulus 23", "Shear@
Modulus 31", "Density"], [2, 3, 4, 5, 6, 7, 8, 9, 10, 16,
   0], "Property@
Values", ["10.3e9", "50e9", "10.3e9", "0.32", "0.32", "0.35
   ", "5.1e9",@
"5.1e9", "4.7e9", "1333.2", ""] )
```

The hex_material() function defines the material properties for the hexagon interstage. To effectively generate materials in MSC Patran, initiate the process by defining the material within the graphical user interface (GUI) and subsequently saving the section file. Next, open the section file and identify the function designated for material creation. Substitute the fixed numerical values for properties such as Young's Modulus and Poisson's Ratio with predefined variable names (for instance, 'e_11' for Young's Modulus in the 11 direction and 'u_12' for Poisson's Ratio in the 12 direction). Prior to this, ensure that these variables are defined with their respective numerical values in your PCL script. This methodology facilitates dynamic and parameterized simulations, permitting easy adjustments to material properties as required within your finite element model, you can find full code in Appendix B.2.

Function 3: `hex_mesh()`

The `hcp_mesh()` function handles the meshing of the created geometry. It applies a mesh to the surface based on the specified element size.

```
mesh_seed_create( "Curve 1:`n*nc*4`", 1, 16, 0., 0., 0. )
```

This will create the mesh seed on the geometry.
Here ‘n*nc*4‘ represents the number of elements , and 1, 16, 0., 0., 0. represents the curve list that will be mesh seeded.

```
INTEGER fem_create_mesh_curve_num_nodes
INTEGER fem_create_mesh_curve_num_elems
STRING fem_create_mesh_c_nodes_created[VIRTUAL]
STRING fem_create_mesh_c_elems_created[VIRTUAL]
fem_create_mesh_curv_1( "curve 1:`n*nc*4`", 16384, `x/16`,
   "Bar2", "#", "#",  "Coord 0", "Coord 0",
   fem_create_mesh_curve_num_nodes,
fem_create_mesh_curve_num_elems,
   fem_create_mesh_c_nodes_created,
fem_create_mesh_c_elems_created )
```

This will mesh the geometry.
["‘x/16‘"]: The target element size for the mesh; here, the elements will aim to have a size of ‘x/16‘ units, this can be adjust to as you need.

```
REAL fem_equiv_all_x_equivtol_ab
INTEGER fem_equiv_all_x_segment
fem_equiv_all_group4( [" "], 0, "", 1, 1, `x/23.71` , FALSE
   ,  fem_equiv_all_x_equivtol_ab, fem_equiv_all_x_segment
   )
```

This will equivalence the mesh seed, you can find full code in Appendix B.3.

Function 4: `hex_load/bc()`

The hex_load/bc() function is used to apply loads and boundary conditions to the model.

```
loadsbcs_create2( "FIXED", "Displacement", "Nodal", "", "
   Static",["`LISTA`"],@
"FEM", "Coord 0", "1.", ["<0 0 0   >", "< 0 0 0  >", "<
    >", "<     >"],@
```

```
[ "", "", "", ""]
```

This will fix the base of the hexagon interstage.
Here ["[0 0 0]", "[] 0 0 0]", "[ ]", "[ ]"],@ ["", "", "", ""] represents the displacements(we are taking fixed base), ‘LISTA‘this represents the area to which the boundary condition will apply.

```
loadsbcs_create2( "force", "Force", "Nodal", "", "Static",@
["Node‘n*nc*4*17‘"], "FEM", "Coord 0", "1.", ["<‘-
   critical_load‘ 0 0 >",@
"<     >", "<     >", "<     >"], ["",  "", "", ""] )
```

This we apply the force on the top of the hexagon interstage, you can find full code in Appendix B.4.
Here "[‘-critical_load‘ 0 0]","[ ]", "[ ]", "[ ]", ["", "", "", ""] represents the force distribution in various directions (in our case we are applying force in negetive x direction on the top of the helical interstage), and ["Node‘n*nc*4*17‘"] represents the nodes on which the force will act.

Function 5: `hex_analysis()`

To conduct the analysis, begin by submitting the file for normal mode analysis through the graphical user interface (GUI). Subsequently, navigate to the section file to extract the necessary code and integrate it into your PCL script. It is advisable to develop a function for the analysis within your PCL code, as the modifications required for establishing the analysis are minimal, you can find full code in Appendix B.5. After making all functions, call all functions for executing code.

3.4 Results and Discussion

3.4.1 Frequency Analysis

An initial frequency analysis was performed on a hexagonal interstage characterized by the following parameters: $R = 0.5$, $H = 1.0$, $\phi = 35°$, $bh = 0.005$, $t = 0.005$, and $n = 20$. The upper and lower sections of the structure were constrained. The analysis produced the subsequent natural frequencies for the first ten modes:

- 1st Mode: $f = 69.962$ Hz
- 3rd Mode: $f = 72.154$ Hz
- 5th Mode: $f = 86.062$ Hz
- 7th Mode: $f = 102.48$ Hz
- 9th Mode: $f = 112.11$ Hz

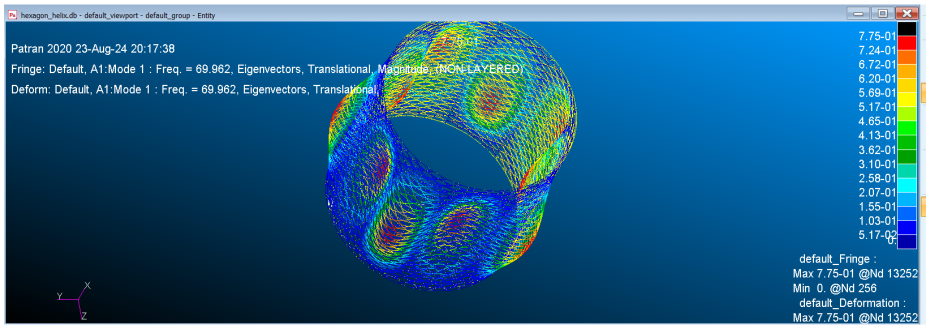
Figures 3.3, 3.4, 3.5, 3.6, and 3.7 depict the deformation shapes corresponding to the 1st, 3rd, 5th, 7th, and 9th modes, respectively. It is important to highlight that modes exhibiting the same frequencies (for instance, the 1st and 2nd modes) display complementary deformation patterns: when one mode demonstrates outward movement, the associated mode reflects inward movement.

3.4.2 Buckling Analysis

A buckling analysis was conducted on the same model ($n = 20$), with a load of 1 N applied at the top and both ends fixed. The analysis yielded a load factor of 18862.7. The buckling mode shape is illustrated in Figure 3.8.

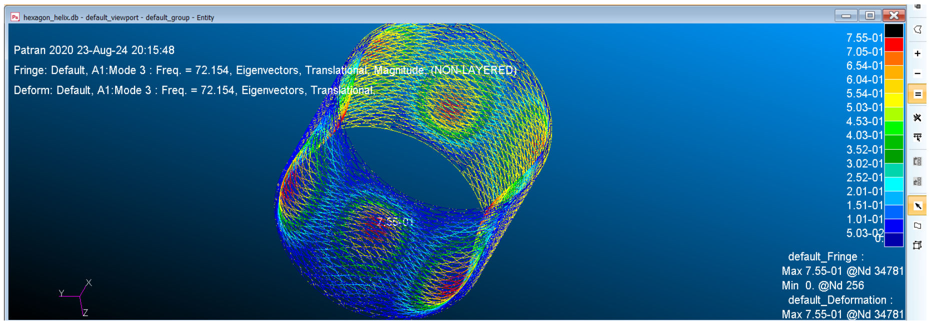
In contrast, the research conducted by Lai et al. [2] involved a similar analysis with $n = 100$. Their code completed the tasks of geometry creation, application of material properties, meshing, and setting up boundary conditions in 35 seconds, whereas my code took approximately 100 seconds for the same processes. Furthermore, executing the analysis in NASTRAN on my laptop required about 10 minutes, resulting in a load factor of 167805 and a total structural mass of 20 kg. This mass is greater than the 14 kg reported by Lai et al., suggesting a potential inconsistency in the modeling approach. The buckling mode shape for $n = 100$ is depicted in Figure 3.9.

Despite the discrepancies in results when compared to the reference study, this exercise significantly improved my understanding of PCL, allowing me to effectively parameterize any model.



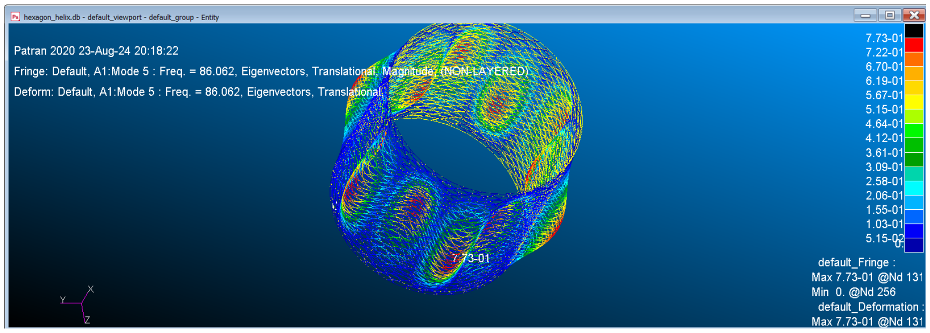
```
hexagon_helix.db - default_viewport - default_group - Entity
Patran 2020 23-Aug-24 20:17:38
Fringe: Default, A1:Mode 1 : Freq. = 69.962, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 1 : Freq. = 69.962, Eigenvectors, Translational,
default_Fringe :
Max 7.75-01 @Nd 13252
Min 0. @Nd 256
default_Deformation :
Max 7.75-01 @Nd 13252
```

Figure 3.3: Mode 1: Frequency = 69.962 Hz



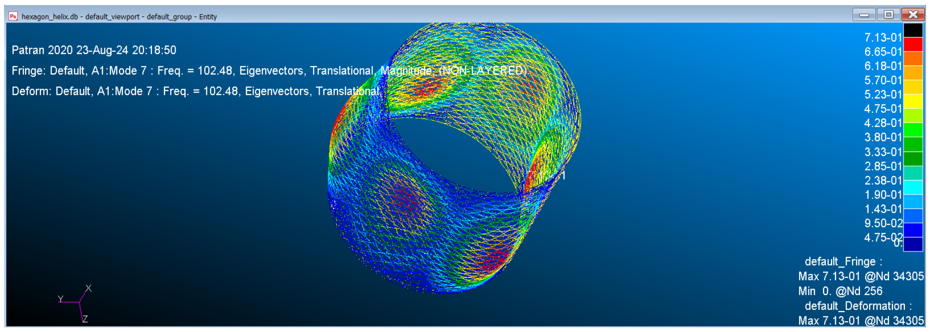
```
hexagon_helix.db - default_viewport - default_group - Entity
Patran 2020 23-Aug-24 20:15:48
Fringe: Default, A1:Mode 3 : Freq. = 72.154, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 3 : Freq. = 72.154, Eigenvectors, Translational,
default_Fringe :
Max 7.55-01 @Nd 34781
Min 0. @Nd 256
default_Deformation :
Max 7.55-01 @Nd 34781
```

Figure 3.4: Mode 3: Frequency = 72.154 Hz



```
hexagon_helix.db - default_viewport - default_group - Entity
Patran 2020 23-Aug-24 20:18:22
Fringe: Default, A1:Mode 5 : Freq. = 86.062, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 5 : Freq. = 86.062, Eigenvectors, Translational,
default_Fringe :
Max 7.73-01 @Nd 131
Min 0. @Nd 256
default_Deformation :
Max 7.73-01 @Nd 131
```

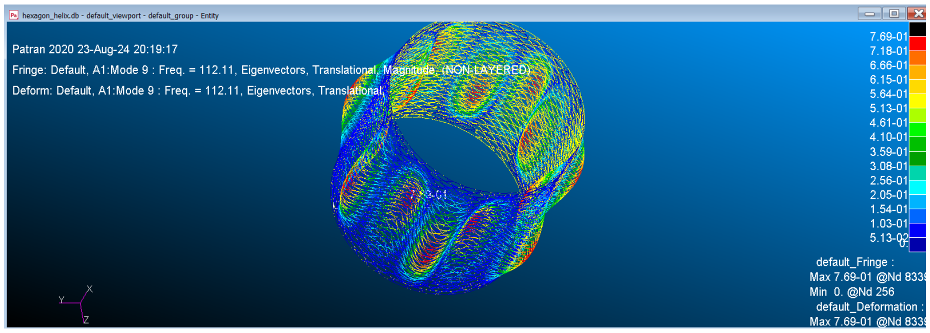
Figure 3.5: Mode 5: Frequency = 86.062 Hz



Patran 2020 23-Aug-24 20:18:50
Fringe: Default, A1:Mode 7 : Freq. = 102.48, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 7 : Freq. = 102.48, Eigenvectors, Translational,

default_Fringe :
Max 7.13-01 @Nd 34305
Min 0. @Nd 256
default_Deformation :
Max 7.13-01 @Nd 34305

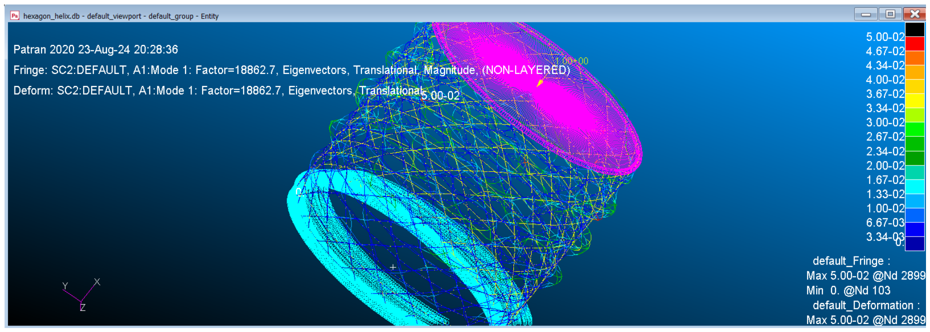
Figure 3.6: Mode 7: Frequency = 102.48 Hz



Patran 2020 23-Aug-24 20:19:17
Fringe: Default, A1:Mode 9 : Freq. = 112.11, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: Default, A1:Mode 9 : Freq. = 112.11, Eigenvectors, Translational,

default_Fringe :
Max 7.69-01 @Nd 8339
Min 0. @Nd 256
default_Deformation :
Max 7.69-01 @Nd 8339

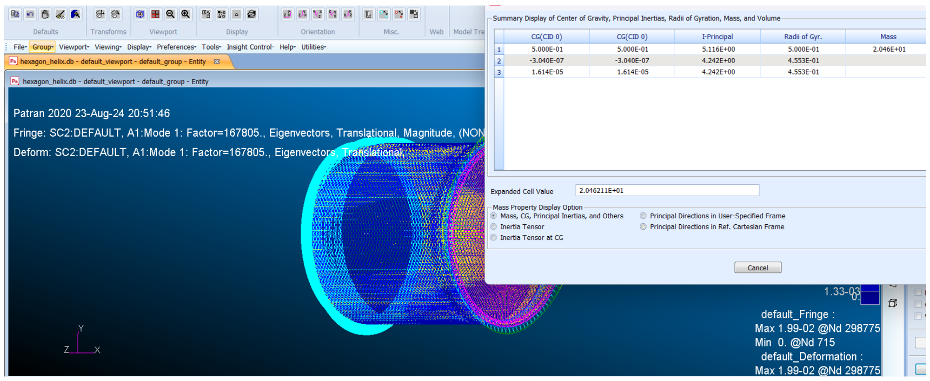
Figure 3.7: Mode 9: Frequency = 112.11 Hz



Patran 2020 23-Aug-24 20:28:36
Fringe: SC2:DEFAULT, A1:Mode 1: Factor=18862.7, Eigenvectors, Translational, Magnitude, (NON-LAYERED)
Deform: SC2:DEFAULT, A1:Mode 1: Factor=18862.7, Eigenvectors, Translational,

default_Fringe :
Max 5.00-02 @Nd 2899
Min 0. @Nd 103
default_Deformation :
Max 5.00-02 @Nd 2899

Figure 3.8: Buckling Mode for $n = 20$



Patran 2020 23-Aug-24 20:51:46
Fringe: SC2:DEFAULT, A1:Mode 1: Factor=167805., Eigenvectors, Translational, Magnitude, (NON
Deform: SC2:DEFAULT, A1:Mode 1: Factor=167805., Eigenvectors, Translational,

default_Fringe :
Max 1.99-02 @Nd 298775
Min 0. @Nd 715
default_Deformation :
Max 1.99-02 @Nd 298775

Summary Display of Center of Gravity, Principal Inertias, Radii of Gyration, Mass, and Volume

	CG(CID 0)	CG(CID 0)	I-Principal	Radii of Gyr.	Mass
1	5.000E-01	5.000E-01	5.116E+00	5.000E-01	2.046E+01
2	-3.040E-07	-3.040E-07	4.242E+00	4.553E-01	
3	1.614E-05	1.614E-05	4.242E+00	4.553E-01	

Expanded Cell Value: 2.046211E+01

Mass Property Display Option
[✓ Mass, CG, Principal Inertias, and Others]
Inertia Tensor
Inertia Tensor at CG
Principal Directions in User-Specified Frame
Principal Directions in Ref. Cartesian Frame

Figure 3.9: Buckling Mode for $n = 100$

Chapter 4

Conclusion

Throughout my internship, I acquired substantial expertise in aerospace structural analysis, with a particular emphasis on parameterized finite element modeling, as well as frequency and buckling assessments of various composite structures. My primary responsibilities included modeling honeycomb sandwich plates and hexagonal grid interstage structures utilizing MSC Patran and Nastran, while applying Sandwich Panel Theory to effectively represent the behavior of these intricate structures.

Initially, I focused on mastering the intricacies of MSC Patran, beginning with tutorials that covered fundamental structures such as lugs and annular plates. This foundational training enabled me to progress to more advanced modeling endeavors. For the sandwich honeycomb plate, I utilized Sandwich Panel Theory to incorporate the distinct mechanical characteristics of both the core and face sheets, facilitating accurate analyses of natural frequency and buckling. The theoretical models were corroborated with numerical results derived from the software, resulting in a comprehensive comparison of experimental, numerical, and theoretical data.

The hexagonal grid interstage presented its own set of challenges, particularly in developing a loop within the PCL code for parameterized modeling. The number of circular ribs (nc) was contingent upon non-integer values, necessitating meticulous attention to the curve numbering and selection process for the application of loads and boundary conditions to ensure geometric precision. These challenges were effectively managed by formulating a generic equation and utilizing the listing option in PCL, which streamlined the selection of curves based on their coordinates.

In the frequency analysis conducted on a grid structure characterized by parameters $R = 0.5$, $H = 1.0$, $\phi = 35°$, $bh = 0.005$, $t = 0.005$, $n = 20$, and with fixed top and bottom edges, the first mode was identified at $f_1 = 69.962$ Hz, with subsequent modes exhibiting similar characteristics. The buckling analysis for the same configuration, subjected to a 1 N load at the top, resulted in a load factor of 18862.7. However, when the model was scaled to $n = 100$, significant computational difficulties arose, including extended processing times and inconsistencies in mass and load factor relative to existing literature. This experience underscored the computational limitations and the critical need for parameter optimization in extensive simulations.

Furthermore, I gained a more profound insight into automation processes, such as utilizing CMD for managing '.dat' files and dynamically updating section files. These improvements significantly enhanced the workflow, especially in executing batch analyses and extracting pertinent data for subsequent investigation.

In summary, this internship equipped me with a robust skill set in the modeling and analysis of advanced aerospace structures. The challenges faced, particularly in the parameterization and validation of intricate grid structures and sandwich honeycomb plates, have deepened my comprehension of structural analysis and finite element modeling. The knowledge and practical expertise acquired during this experience will provide a strong foundation for my future academic and professional endeavors in aerospace engineering.

Bibliography

[1] H. S. Moghaddam, "A numerical homogenization approach to characterize in-plane anisotropic hyperelastic responses of a non-metallic honeycomb core," PhD dissertation, Wichita State University, Wichita, KS, 2020, master of Science, Azad University, 2010; Bachelor of Engineering, Azad University, 2007.

[2] C. Lai, J. Wang, and C. Liu, "Parameterized finite element modeling and buckling analysis of six typical composite grid cylindrical shells," *Applied Composite Materials*, vol. 21, no. 6, pp. 739–758, 2014, received: 10 December 2013 / Accepted: 12 December 2013 / Published online: 9 January 2014. [Online]. Available: https://link.springer.com/article/10.1007/s10443-013-9376-x

[3] W. Wang, H. Luo, J. Fu, H. Wang, C. Yu, G. Liu, Q. Wei, and S. Wu, "Comparative application analysis and test verification on equivalent modeling theories of honeycomb sandwich panels for satellite solar arrays," *Advanced Composites Letters*, vol. 29, pp. 1–15, 2020. [Online]. Available: https://journals.sagepub.com/home/acm

[4] A. Aborehab, M. Kassem, A. F. Nemnem, and M. Kamel, "Miscellaneous modeling approaches and testing of a satellite honeycomb sandwich plate," *Journal of Applied and Computational Mechanics*, vol. 6, no. SI, pp. 1084–1097, 2020.

[5] M. Kumar, V. Kar, and M. Chandravanshi, "Free vibration analysis of sandwich composite plate with honeycomb core," *Materials Today: Proceedings*, 2021.

[6] A. Boudjemai, M. Bouanane, A. Mankour, R. Amri, H. Salem, and B. Chouchaoui, "Mda of hexagonal honeycomb plates used for space applications," *International Journal of Mechanical and Mechatronics Engineering*, vol. 6, no. 6, pp. 1068–1073, 2012. [Online]. Available: https://publications.waset.org/3081/pdf

[7] “Hexweb^{TM} honeycomb sandwich design technology,” Hexcel Composites, Duxford, UK, Tech. Rep., December 2000, publication No. AGU 075b, ® Hexcel Registered Trademark, © Hexcel Composites.

[8] MSC Software Corporation, “MSC Patran Workshop 2005,” https://www.scc.kit.edu/scc/sw/msc/Pat304/pat304.html, 2005.

Appendix A

Appendix A: PCL Code for Sandwich Deck

A.1 Defining Variable

```
global real Facesheet_Thickness=0.25e-3,@
Core_Thickness=10e3,length=1.08,width=0.98,@
Ea_11=7.2e10,Ea_22=7.2e10,Ua_12=0.33,Ga_12=27e9,@
Densitya=2800.,e_11=1e4,e_33=1e4,u_13=0.3,@
g_13=1e4,g_12=3.4e8,g_32=3.4e8,density=72.
```

A.2 Code for `hcp_create()`

```
FUNCTION hcp_create()
uil_file_new.go( "", "vvv.db")
STRING asm_line_2point_created_ids[VIRTUAL]
asm_const_line_2point( "1","[0 0 0]","[`length` 0 0]",0,"",
   @
50.,1,asm_line_2point_created_ids )
$# 1 Line created: Line 1
asm_const_line_2point("2","Point 2","[`length` `width` 0]",
   @
0,"",50.,1,asm_line_2point_created_ids )
$# 1 Line created: Line 2
asm_const_line_2point("3","Point 3","[0 `width`
   0]",0,"",50.@
,1,asm_line_2point_created_ids )
$# 1 Line created: Line 3
```

```
asm_const_line_2point("4", "Point 4", "Point 1",
   0,"",50.,1,@
asm_line_2point_created_ids )
$# 1 Line created: Line 4
STRING asm_create_cord_2ve_created_ids[VIRTUAL]
asm_const_coord_2vector_v1("1","XY","Coord 0",1,"Point 1"@
,"<1 0 0>","<0 1 0>", asm_create_cord_2ve_created_ids )
$# 1 Coord created: Coord 1
STRING sgm_surface_4edge_created_ids[VIRTUAL]
sgm_const_surface_4edge("1","Curve 4","Curve 1","Curve 2",@
"Curve 3",sgm_surface_4edge_created_ids )
$# 1 Surface Created: Surface 1
END FUNCTION
```

A.3 Code for `hcp_mesh()`

```
FUNCTION hcp_mesh()
ui_exec_function( "mesh_seed_display_mgr", "init" )
INTEGER fem_create_mesh_surfa_num_nodes
INTEGER fem_create_mesh_surfa_num_elems
STRING fem_create_mesh_s_nodes_created[VIRTUAL]
STRING fem_create_mesh_s_elems_created[VIRTUAL]
fem_create_mesh_surf_4("IsoMesh",49152,"Surface 1",1,@
["0.05"],"Quad4","#","#","Coord 0","Coord 0",@
fem_create_mesh_surfa_num_nodes,
   fem_create_mesh_surfa_num_elems@
,fem_create_mesh_s_nodes_created,
   fem_create_mesh_s_elems_created)
mesh_seed_display_mgr.erase(  )
END FUNCTION
```

A.4 Code for `hcp_load()`

```
FUNCTION hcp_load()
```

```
loadsbcs_create2("FIXED","Displacement","Nodal","","Static
   ",[@
"Surface 1.1 1.2 1.3 1.4"],"Geometry","Coord 0","1.",["<0 0
    0>",@
"<0 0 0>", "<     >", "<     >"], ["", "", "", ""] )
$# Load/BC set "FIXED" created.
END FUNCTION
```

A.5 Code for hcp_material()

```
FUNCTION hcp_material()
material.create( "Analysis code ID", 1, "Analysis type @
ID",1, "Aluminium", 0,"Date: 05-Jul-24  Time: 11:42:58",@
"Isotropic", 1, "Directionality",  1, "Linearity", 1, @
"Homogeneous", 0, "Linear Elastic", 1,"Model Options & IDs"
   @
,["", "", "", "", ""],[0, 0, 0, 0, 0], "Active Flag",1,@
"Create",10,"External Flag",FALSE,"Property IDs",@
["Elastic Modulus","Poisson Ratio","Shear Modulus",@
"Density"],[2, 5, 8, 16, 0],"Property Values",@
["`Ea_11`","`Ua_12`","`Ga_12`", "`Densitya`", ""] )

material.create("Analysis code ID",1,"Analysis type ID",@
1,"Core",0,"Date: 05-Jul-24  Time: 11:42:58", @
"2d Orthotropic", 5,"Directionality",4,"Linearity",1,@
"Homogeneous",0,"Linear Elastic",1,"Model Options & IDs"@
,["", "", "", "", ""],[0, 0, 0, 0, 0],"Active Flag",1,@
"Create",10,"External Flag",FALSE,"Property IDs"@
,["Elastic Modulus 11", "Elastic Modulus 22", "Poisson
   Ratio 12",@
"Shear Modulus 12","Shear Modulus 23","Shear Modulus 13","
   Density"]@
,[2, 3, 5, 8, 9, 10, 16, 0],"Property Values",["`e_11`", "`
   e_33`", @
"`u_13`", "`g_13`", "`g_32`","`g_12`", "`density`", ""] )
END FUNCTION
```

A.6 Code for `hcp_composite()`

```
FFUNCTION hcp_composite()
mat_create_lam3("Composite","",1,["Aluminium", "Core", @
"Aluminium"]["`Facesheet_Thickness`","`Core_Thickness`"@
,"`Facesheet_Thickness`"],[" 0.000000E+00", " 0.000000E
   +00",@
" 0.000000E+00"],["", "", ""], 3,"","Create" )

elementprops_create("prop",51, 25, 35, 1, 3, 20,@
[13, 20, 4037, 4111,4265,1005, 5, 8111, 4213],@
[5, 9, 1, 1, 1, 1, 1, 4, 4],["m:Composite",@
"", "", "",  "", "", "", "", ""], "Surface 1" )
$# Property Set "prop" created.
END FUNCTION
```

A.7 Code for `hcp_analysis()`

```
FUNCTION hcp_analysis()
$#  Beginning analysis of  vvv
jobfile.open( "vvv", "ANALYZE NO JOBFILE" )
msc_delete_old_files( "vvv", ".bdf", ".op2" )
jobfile.write_spl( "/* Jobfile for PATNAS created @
%A% at %A% */", ["05-Jul-24" , "10:55:15"] )
jobfile.writec( "", "TRANSLATOR = pat3nas" )
jobfile.writec( "DATABASE", "`Directory`\vvv.db" )
jobfile.writec( "JOBNAME", "vvv" )
jobfile.writec( "ANALYSIS TITLE", @
"MSC.Nastran job created on 05-Jul-24 at" //" 10:54:42" )
jobfile.writec( "ANALYSIS SUBTITLE", "" )
jobfile.writec( "ANALYSIS LABEL", "" )
jobfile.writec( "", "" )
jobfile.writec( "OBJECT", "Current Group" )
jobfile.writec( "METHOD", "Analysis Deck" )
jobfile.writec( "", "" )
jobfile.writec( "MODEL SUFFIX", ".bdf" )
jobfile.writec( "RESULTS SUFFIX", ".op2" )
jobfile.writec( "GROUP", "default_group" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * File Search Path Declaration" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "File Search Path", "`job_Directory`" )
jobfile.writec( "File Search Path", "`job_Directory`" )
jobfile.writec( "File Search Path", "`job_Directory`\help"
   // @
"files" )
jobfile.writec( "File Search Path", "`job_Directory`\alters
   " )
jobfile.writec( "File Search Path", "`job_Directory`\icons"
    )
jobfile.writec( "File Search Path", "`job_Directory`\icon"
   // @
"s\RibbonIcons" )
jobfile.writec( "File Search Path", "`job_Directory`\bin" )
jobfile.writec( "File Search Path", "`job_Directory`\bin\
   exe" )
jobfile.writec( "File Search Path", "`job_Directory`\msce"
   // @
"xplore_files\" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\dmap" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\plb" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\lib" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\icons" )
```

```
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\icons" )
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\plb" )
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\extra_files" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * Translation Parameters" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "DATA OUTPUT", "XDB+PRINT" )
jobfile.writec( "OUTPUT2 REQUESTS", "P3 Built In" )
jobfile.writec( "OUTPUT2 FORMAT", "Binary" )
jobfile.writec( "DIVISION TOLERANCE", "1.0e-08" )
jobfile.writec( "NUMERICAL TOLERANCE", "1.0e-04" )
jobfile.writec( "WRITING TOLERANCE", "1.0e-21" )
jobfile.writec( "GEOM CHECK", "INFORM" )
jobfile.writec( "SORTED BULK", "NO" )
jobfile.writec( "CARD FORMAT", "either" )
jobfile.writec( "NODE COORDINATES", "reference frame" )
jobfile.writec( "COORD COORDINATES", "global" )
jobfile.writec( "MSC.Nastran VERSION", "2017." )
jobfile.writec( "WRITE STORED PRECISION", "TRUE" )
jobfile.writec( "PROPS ON ELEM ENTRY", "FALSE" )
jobfile.writec( "CONTINUATION ENTRY", "FALSE" )
jobfile.writec( "PCOMPG ENTRY", "TRUE" )
jobfile.writec( "CONVERT CBAR CBEAM", "FALSE" )
jobfile.writec( "ITERATIVE SOLVER", "FALSE" )
jobfile.writei( "SUPER ELEMENT 0", 0 )
jobfile.writec( "SEALL WRITE", "FALSE" )
jobfile.writec( "PART SUPERELEMENT CREATE", "TRUE" )
jobfile.writec( "AUTOQSET", "FALSE" )
jobfile.writec( "FIXEDB", "FALSE" )
jobfile.writei( "SUPER TREE COUNT", 0 )
jobfile.writec( "MODEL TOLERANCE", "0.0049999999" )
jobfile.writec( "ELEMENT PROPERTY OFFSET", "0" )
jobfile.writec( "MATERIAL PROPERTY OFFSET", "0" )
jobfile.writec( "TABLE OFFSET", "0" )
jobfile.writec( "LOAD SET OFFSET", "0" )
jobfile.writec( "LOAD CASE OFFSET", "0" )
jobfile.writec( "CONTROL SET OFFSET", "0" )
jobfile.writec( "RIGID ELEMENT OFFSET", "0" )
jobfile.writec( "SCALAR POINT OFFSET", "0" )
jobfile.writec( "BEGINNING CONTINUATION MARKER", "+       A"
    )
jobfile.writec( "NUMBER ONLY", "ON" )
jobfile.writec( "BEGINNING NUMBER", "OFF" )
jobfile.writec( "TRAILING NUMBER", "OFF" )
jobfile.writec( "SYNTAX NUMBER", "ON" )
jobfile.writec( "SYNTAX MARKER", "." )
jobfile.writec( "EXTERNAL SUPERELEMENT METHOD", "NONE" )
jobfile.writec( "GRID COORDINATES ROUNDING", "15" )
jobfile.writec( "COORD DATA PRECISION ROUNDING", "15" )
jobfile.writec( "MPC DATA PRECISION ROUNDING", "15" )
jobfile.writec( "LBC DATA PRECISION ROUNDING", "7" )
jobfile.writec( "MAT DATA PRECISION ROUNDING", "15" )
jobfile.writec( "PROP DATA PRECISION ROUNDING", "7" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * Solution Parameters" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "SOLUTION TYPE", "NORMAL MODES" )
jobfile.writei( "SOLUTION SEQUENCE", 103 )
jobfile.writec( "DATABASE RUN", "ON" )
jobfile.writec( "INTERACTIVE MODES", "OFF" )
jobfile.writec( "CYCLIC SYMMETRY", "OFF" )
jobfile.writec( "AUTOMATIC CONSTRAINTS", "ON" )
jobfile.writec( "SHELL NORMAL TOLERANCE", "" )
jobfile.writec( "MASS CALCULATION", "Lumped" )
jobfile.writec( "DATA DECK ECHO", "None" )
jobfile.writec( "PLATE RZ STIFFNESS FACTOR", "100.0" )
jobfile.writec( "MAXIMUM PRINTED LINES", "" )
jobfile.writec( "MAXIMUM RUN TIME", "" )
jobfile.writec( "WT-MASS CONVERSION", "1.0" )
jobfile.writec( "NODE ID FOR WT-GENER", "" )
jobfile.writec( "RIGID ELEMENT TYPE", "LINEAR" )
jobfile.writec( "USE SOL600", "OFF" )
jobfile.writec( "RESIDUAL VECTOR", "Nastran Default" )
jobfile.writec( "SN MEAN STRESS CORRECTION", "NONE" )
jobfile.writec( "NEF FATIGUE STRESS OUTPUT", "NONE" )
jobfile.writec( "NEF STRESS UNITS", "MPA" )
jobfile.writec( "NEF STRESS COMBINATION", "ABSMAXPR" )
jobfile.writec( "NEF RESULT LOCATION", "NODA" )
jobfile.writec( "NEF RECOVERY LOCATION", "CORNER" )
jobfile.writec( "NEF LAYER LOCATION", "WORST" )
jobfile.writec( "NEF INTERPOLATION LIMIT", "0" )
jobfile.writec( "NEF LEVEL OF MESSAGE", "0" )
jobfile.writec( "NEF NUMBER OF THREADS", "1" )
jobfile.writer( "NEF CERTAINTY OF SURVIVAL", 50. )
jobfile.writer( "NEF OVERALL SCALE FACTOR", 1. )
jobfile.writer( "NEF TOP STRESS PERCENTAGE FILTER", 100. )
jobfile.writer( "NEF TOP DAMAGE PERCENTAGE FILTER", 100. )
jobfile.writec( "NEF TIME HISTORY DIRECTORY", "C:\Users\
   PC683" )
jobfile.writei( "NEF MAX LOAD PER EVENT", 1 )
jobfile.writec( "NEF EVENT DATA", "0" )
jobfile.writec( "NEF LOAD SEQUENCE DATA", "0" )
jobfile.writec( "USE CONTACT TABLE", "OFF" )
jobfile.writec( "INITIAL CONTACTPAIR LOADCASE NAME", "" )
jobfile.writei( "MDOF DATA", 0 )
jobfile.writec( "CELL WRITE", "ON" )
jobfile.writei( "CELL INPUT 0", 0 )
jobfile.writec( "FMS WRITE", "ON" )
jobfile.writei( "FMS INPUT 0", 0 )
jobfile.writec( "EXEC WRITE", "ON" )
jobfile.writei( "EXEC INPUT 0", 0 )
jobfile.writec( "CASE WRITE", "ON" )
jobfile.writei( "CASE INPUT 0", 0 )
jobfile.writec( "BULK WRITE", "ON" )
jobfile.writei( "BULK INPUT 0", 0 )
jobfile.writec( "CELL DTI POSITION", "START" )
jobfile.writec( "FMS DTI POSITION", "START" )
jobfile.writec( "EXEC DTI POSITION", "START" )
jobfile.writec( "CASE DTI POSITION", "START" )
jobfile.writec( "BULK DTI POSITION", "START" )
jobfile.writec( "", "END" )
jobfile.close(  )
mscnastran_job.associate_subcases( "103", "vvv", 1, ["
   Default"] )
analysis_submit_2( "MSC.Nastran", "vvv" )
END FUNCTION
```

A.8 CMD code

```
@echo off
setlocal enabledelayedexpansion
REM Variables
set "input_file=D:\study\Internship\HCP\HCP.DAT"

REM Function to read input.dat and set variables
:read_input_file
for /f "tokens=1,2 delims==" %%a in (%input_file%) do (
    set "%%a=%%b"
)
set "ses_file=%Directory%\Hcp_ses.ses"
set "temp_file=%Directory%\temp.dat"
set "patran_executable=%patran_target%"
set "nastran_executable=%nastran_target%"

REM Function to update variables in the .ses file
:update_ses_file
(
    for /f "usebackq tokens=*" %%a in ("%ses_file%") do (
        set "line=%%a"
        if "!line:~0,6!"=="global" (
         echo global real Facesheet_Thickness=%
   Facesheet_Thickness%,Core_Thickness=%Core_Thickness%,
   length=%length%,width=%width%,Ea_11=%Ea_11%,Ea_22=%Ea_22
   %,Ua_12=%Ua_12%,Ga_12=%Ga_12%,Densitya=%Densitya%,e_11=%
   e_11%,e_33=%e_33%,u_13=%u_13%,g_13=%g_13%,g_12=%g_12%,
   g_32=%g_32%,density=%density%
        ) else (
            echo %%a
        )
    )
) > "%temp_file%"

move /y "%temp_file%" "%ses_file%" > nul
echo Variables updated in %ses_file%

REM Function to execute Patran session
:run_patran
(
    echo Running Patran...
    "%patran_executable%" -b -graphics -sfp %ses_file% scr=
   yes old=no
)

REM Function to execute Nastran
:run_nastran
(
    echo Running Nastran...
    "%nastran_executable%" "%Directory%\vvv.bdf" scr=yes
)
echo please wait 15 seconds nastran is loading. then press
   any key
pause
set "temp_file=%Directory%\temp.csv"
set "f06_files=%Directory%\vvv.f06"  REM Update with your
   actual path
set "csv_file=table.csv" REM Update with your desired CSV
   file path
REM Function to update variables in the .ses file
:extracted_table_from_f06
(
    set "found=false"
    set "counter=0"
    for /f "usebackq tokens=*" %%a in ("%f06_files%") do (
        if !counter! geq 13 (
            set "found=false"
        )
        set "line=%%a"
        if "!line!" == "R E A L   E I G E N V A L U E S" (
            set "found=true"
        )
        if "!found!" == "true" (
            set /a counter+=1
            REM Replace multiple spaces with a single space
            set "line=!line:  = !"
            set "line=!line:   = !"
            set "line=!line:    = !"
            set "line=!line:     = !"
            set "line=!line:      = !"
            set "line=!line:       = !"
            set "line=!line:        = !"
            REM Replace spaces with commas
            set "line=!line: = ,!"
            echo !line!
        )
    )
) > "%temp_file%"
echo values generated %temp_file%
pause
endlocal
```

A.9 DAT file

```
Facesheet_Thickness=1e-3
Core_Thickness=8e-3
length=0.4
width=0.4
Ea_11=7e10
Ea_22=7e10
Ua_12=0.33
Ga_12=26.32e9
Densitya=2800.
e_11=3.3e6
e_33=3.3e6
u_13=0.99
g_13=1.98e6
g_32=6.517e8
g_12=4.26e8
density=130.
Directory=D:\study\Internship\HCP
patran_target=C:\Program Files\MSC.Software\Patran_x64
   \2020\bin\patran.exe
nastran_target=C:\Program Files\MSC.Software\MSC_Nastran
   \2020sp1\bin\nastranw.exe
```

Appendix B

Appendix A: PCL Code for Hexagon interstage

B.1 Code for `hex_create()`

```
Global integer nc,n=20
Global real R=0.5, H=1., phi=35.,bh=0.005,t=0.005,
   critical_load=1
real x,i,nc_1
uil_file_new.go( "", "hexagon_helix.db")
pref_geo_set_v1( 0, 0.0099999998, 3 )
pref_global_set_v3( TRUE, 3, "0.0000000000000000000000005",
    "SAVE" )
pref_env_set_logical( "revert_enabled", FALSE )
nc = 0
nc_1= 2*n*H*tanr(phi*3.14/180)/(3.14*2*R)
ui_write(STR_FROM_REAL(nc_1))
WHILE( nc_1 > 0 )
nc_1= nc_1-1
nc = nc+1
end WHILE
nc=nc-1
ui_write("nc = ")
ui_write(STR_FROM_INTEGER(nc))
x=H/nc
i=360./n
STRING asm_create_line_xyz_created_ids[VIRTUAL]
```

```
asm_const_line_xyz( "1", "<`x` 0 0>", "[0 0 0]", "Coord 0",
    asm_create_line_xyz_created_ids )
STRING sgm_transform_curve_created_ids[VIRTUAL]
sgm_transform_translate_v1( "2", "curve", "<1 0 0>", `x`,
   FALSE, "Coord 0", `nc-1`, FALSE, "Curve 1",
   sgm_transform_curve_created_ids )
STRING sgm_create_curve_2d_created_ids[VIRTUAL]
sgm_const_curve_2d_arcangles_v1( "`nc+1`", `R`, 0., `i`, "
   Coord 0.1", "[0 0 0]", FALSE,
   sgm_create_curve_2d_created_ids )
sgm_transform_rotate( "`nc+2`", "curve", "Coord 0.1", `i`,
   0., "Coord 0", `n-1`, FALSE, "Curve `nc+1`",
   sgm_transform_curve_created_ids )
asm_transform_curve_vsum( "`nc+n+1`", "[0 0 0]", "[0 0 0]",
    [1., 1., 1.], [1., 1., 1.], "Curve 1:`nc`", "Curve `nc
   +1`:`2*nc`", sgm_transform_curve_created_ids )
STRING sgm_create_plane_po_created_ids[VIRTUAL]
sgm_const_plane_point_vector( "1", "Point 1", "<1 0 0>",
   sgm_create_plane_po_created_ids )
STRING sgm_create_plane_of_created_ids[VIRTUAL]
sgm_const_plane_offset( "2", "`x/4`", 1, "Plane 1",
   sgm_create_plane_of_created_ids )
sgm_const_plane_offset( "3", "`H-(x/4)`", 1, "Plane 1",
   sgm_create_plane_of_created_ids )
STRING sgm_curve_break_pla_created_ids[VIRTUAL]
sgm_edit_curve_break_plane( "`n+2*nc+1`", "Plane 2 3 2 3",
   "Curve 1 `nc` `nc+n+1` `n+2*nc`", TRUE,
   sgm_curve_break_pla_created_ids )
$# Question from application SGM
$#     Do you wish to delete the original curves?
$? YESFORALL 38000217
sgm_transform_translate_v1( "`n+2*nc+9`", "curve", "<1 0
   0>", `x/4`, FALSE, "Coord 0", 1, FALSE, "Curve `nc+1`:`
   nc+n`", sgm_transform_curve_created_ids )
STRING asm_delete_any_deleted_ids[VIRTUAL]
asm_delete_curve( "Curve `n+2*nc+1` `n+2*nc+4` `n+2*nc+5` `
   n+2*nc+8` `nc+1`:`nc+n`", asm_delete_any_deleted_ids )
```

```
asm_delete_plane( "Plane 1 2 3", asm_delete_any_deleted_ids
    )
STRING sgm_renum_curve_new_ids[VIRTUAL]
IF ( nc == 2 ) THEN
real w
w=360./n
integer k
k=n
sgm_transform_mirror( "`2*n+2*nc+8+1`", "curve", "Coord
   0.3", 0., TRUE, FALSE, "Curve `n+2*nc+6` `n+2*nc+7`",
   sgm_transform_curve_created_ids )
sgm_transform_rotate( "`2*n+3*nc+8+1`", "curve", "Coord
   0.1", w, 0., "Coord 0", k-1, FALSE, "Curve `2*n+2*nc
   +8+1`:`2*n+3*nc+8` `n+2*nc+6` `n+2*nc+7`",
   sgm_transform_curve_created_ids )
asm_delete_curve( "Curve `2*nc+n+2` `2*nc+n+3`",
   asm_delete_any_deleted_ids )
END IF
IF ( nc > 2 ) THEN
real w
w=360./n
integer k
k=n
sgm_transform_mirror( "`2*n+2*nc+8+1`", "curve", "Coord
   0.3", 0., TRUE, FALSE, "Curve `n+2*nc+6` `nc+n+2`:`nc+n
   +2+nc-3` `n+2*nc+7`", sgm_transform_curve_created_ids )
sgm_transform_rotate( "`2*n+3*nc+8+1`", "curve", "Coord
   0.1", w, 0., "Coord 0", k-1, FALSE, "Curve `2*n+2*nc
   +8+1`:`2*n+3*nc+8` `n+2*nc+6` `nc+n+2`:`nc+n+2+nc-3` `n
   +2*nc+7`", sgm_transform_curve_created_ids )
asm_delete_curve( "Curve 2:`nc-1` `2*nc+n+2` `2*nc+n+3`",
   asm_delete_any_deleted_ids )
END IF
sgm_transform_translate_v1( "`2*n+3*nc+8+nc*n*2+1-2*nc`", "
   curve", "<1 0 0>", `x/2`, FALSE, "Coord 0", `2*nc-1`,
   FALSE, "Curve `n+2*nc+8+1`:`n+2*nc+8+n`",
   sgm_transform_curve_created_ids )
```

```
IF ( nc == 2 ) THEN
sgm_renumber( 1, "curve", "1", "Curve `n+2*nc+6` `n+2*nc+7`
    `n+2*nc+9`:`2*n+nc+8+nc*n*4-n`",
   sgm_renum_curve_new_ids )
END IF
IF ( nc > 2 ) THEN
sgm_renumber( 1, "curve", "1", "Curve `nc+n+2`:`nc+n+2+nc
   -3` `n+2*nc+6` `n+2*nc+7` `n+2*nc+9`:`2*n+nc+8+nc*n*4-n
   `", sgm_renum_curve_new_ids )
END IF
```

B.2 Code for **hex_material()**

```
material.create( "Analysis code ID", 1, "Analysis type ID",
    1, "material", 0, "Date: 18-Jul-24           Time:
   11:48:43", "3d Orthotropic", 2, "Directionality", 2, "
   Linearity", 1, "Homogeneous", 0, "Linear Elastic", 1, "
   Model Options & IDs", ["", "", "", "", ""], [0, 0, 0, 0,
    0], "Active Flag", 1, "Create", 10, "External Flag",
   FALSE, "Property IDs", ["Elastic Modulus 11", "Elastic
   Modulus 22", "Elastic Modulus 33", "Poisson Ratio 12", "
   Poisson Ratio 23", "Poisson Ratio 31", "Shear Modulus
   12", "Shear Modulus 23", "Shear Modulus 31", "Density"],
    [2, 3, 4, 5, 6, 7, 8, 9, 10, 16, 0], "Property Values",
    ["10.3e9", "50e9", "10.3e9", "0.32", "0.32", "0.35",
   "5.1e9", "5.1e9", "4.7e6", "1333.2", ""] )
```

B.3 Code for **hex_mesh()**

```
mesh_seed_create( "Curve 1:`n*nc*4`", 1, 16, 0., 0., 0. )
INTEGER fem_create_mesh_curve_num_nodes
INTEGER fem_create_mesh_curve_num_elems
STRING fem_create_mesh_c_nodes_created[VIRTUAL]
STRING fem_create_mesh_c_elems_created[VIRTUAL]
```

```
fem_create_mesh_curv_1( "curve 1:`n*nc*4`", 16384, `x/16`,
   "Bar2", "#", "#",  "Coord 0", "Coord 0",
   fem_create_mesh_curve_num_nodes,
   fem_create_mesh_curve_num_elems,
   fem_create_mesh_c_nodes_created,
   fem_create_mesh_c_elems_created )
REAL fem_equiv_all_x_equivtol_ab
INTEGER fem_equiv_all_x_segment
fem_equiv_all_group4( [" "], 0, "", 1, 1, `x/23.71` , FALSE
   ,  fem_equiv_all_x_equivtol_ab, fem_equiv_all_x_segment
   )
beam_section_create( "beam_cross_section", "BAR", ["`bh`",
   "`t`"] )
elementprops_create( "property_beam", 11, 2, 42, 1, 1, 20,
   [39, 13, 6, 4042,4043, 2047, 2048, 1, 10, 11, 4026,
   1026, 4044, 4045, 4037, 4047, 4048, 4050, 4051, 4053,
   4054, 4056, 4057, 8112, 4061, 4303, 8111, 4403, 4404,
   4410, 4411, 8200, 8201, 8202], [11, 5, 2, 2, 2, 4, 4, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 4,
   4, 1, 1, 1, 6, 4, 4, 4], ["beam_cross_section", "m:
   material", "<0 1 0>", "", "", "", "", "", "", "", "",
   "", "", "", "", "", "", "", "", "", "", "", "", "", "",
   "", "", "", "", "", "", "Analysis", "Analysis", "
   Analysis"], "Element 1:`n*nc*4*16`" )
mesh_seed_display_mgr.erase(  )
```

B.4 Code for **hex_load/bc()**

```
list_create_node_att_value( [`x/4`, 0., 0., 0., 0., 0.], [
   TRUE, FALSE,  @
FALSE], ["equal", "equal", "equal"], [5.0000001E-25,
   5.0000001E-25, 0.5],  @
"Coord 0", "lista", uil_list_create_current_list )
list_save_group( "lista", "default_group", FALSE )
list_create_node_att_value( [`H-x/4`, 0., 0., 0., 0., 0.],
   [TRUE, FALSE,  @
FALSE], ["equal", "equal", "equal"], [5.0000001E-25,
   5.0000001E-25, 0.5],  @
"Coord 0", "listb", uil_list_create_current_list )
list_save_group( "listb", "default_group", FALSE )
loadsbcs_create2( "FIXED", "Displacement", "Nodal", "", "
   Static", ["`LISTA`"], @
"FEM", "Coord 0", "1.", ["<0 0 0   >", "< 0 0 0  >", "<
    >", "<     >"], [ @
"", "", "", ""] )
STRING fem_create_nodes__nodes_created[VIRTUAL]
fem_create_nodes_1( "Coord 0", "Coord 0", 3, "#", " [`H-x
   /4` 0 0]",  @
fem_create_nodes__nodes_created )
fem_create_mpc_nodal2( 1, "RBE2", 0., 2, [TRUE, FALSE],
   ["0", "0"], ["`listb`" @
, "Node `n*nc*4*17`"], ["UX", ""] )
loadsbcs_create2( "force", "Force", "Nodal", "", "Static",
   ["Node `n*nc*4*17`"],  @
"FEM", "Coord 0", "1.", ["<`-critical_load` 0 0 >", "<
   >", "<     >", "<     >"], ["", @
 "", "", ""] )
```

B.5 Code for `hex_analysis()`

```
FUNCTION hexagon_buckling_analysis()
 $#  Beginning analysis of  hexagon_buckling
jobfile.open( "hexagon_buckling", "ANALYZE NO JOBFILE" )
msc_delete_old_files( "hexagon_buckling", ".bdf", ".op2" )
jobfile.write_spl( "/* Jobfile for PATNAS created %A% at %A
   % */", ["23-Jul-24" @
, "12:25:52"] )
jobfile.writec( "", "TRANSLATOR = pat3nas" )
jobfile.writec( "DATABASE", "`Directory`/hexagon_" // @
"buckling.db" )
jobfile.writec( "JOBNAME", "hexagon_buckling" )
jobfile.writec( "ANALYSIS TITLE", "MSC.Nastran job created
   on 23-Jul-24 at" // @
" 12:25:37" )
jobfile.writec( "ANALYSIS SUBTITLE", "" )
jobfile.writec( "ANALYSIS LABEL", "" )
jobfile.writec( "", "" )
jobfile.writec( "OBJECT", "Current Group" )
jobfile.writec( "METHOD", "Analysis Deck" )
jobfile.writec( "", "" )
jobfile.writec( "MODEL SUFFIX", ".bdf" )
jobfile.writec( "RESULTS SUFFIX", ".op2" )
jobfile.writec( "GROUP", "default_group" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * File Search Path Declaration" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "File Search Path", "`job_Directory`" )
jobfile.writec( "File Search Path", "`job_Directory`" )
jobfile.writec( "File Search Path", "`job_Directory`\help"
   // @
"files" )
jobfile.writec( "File Search Path", "`job_Directory`\alters
   " )
jobfile.writec( "File Search Path", "`job_Directory`\icons"
    )
jobfile.writec( "File Search Path", "`job_Directory`\icon"
   // @
"s\RibbonIcons" )
jobfile.writec( "File Search Path", "`job_Directory`\bin" )
jobfile.writec( "File Search Path", "`job_Directory`\bin\
   exe" )
jobfile.writec( "File Search Path", "`job_Directory`\msce"
   // @
"xplore_files\" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\dmap" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\plb" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\lib" )
jobfile.writec( "File Search Path", "`job_Directory`\mscp"
   // @
"rocor_files\icons" )
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\icons" )
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\plb" )
jobfile.writec( "File Search Path", "`job_Directory`\shar"
   // @
"eware\msc\unsupported\utilities\extra_files" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * Translation Parameters" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "DATA OUTPUT", "XDB+PRINT" )
jobfile.writec( "OUTPUT2 REQUESTS", "P3 Built In" )
jobfile.writec( "OUTPUT2 FORMAT", "Binary" )
jobfile.writec( "DIVISION TOLERANCE", "1.0e-08" )
jobfile.writec( "NUMERICAL TOLERANCE", "1.0e-04" )
jobfile.writec( "WRITING TOLERANCE", "1.0e-21" )
jobfile.writec( "GEOM CHECK", "INFORM" )
jobfile.writec( "SORTED BULK", "NO" )
jobfile.writec( "CARD FORMAT", "either" )
jobfile.writec( "NODE COORDINATES", "reference frame" )
jobfile.writec( "COORD COORDINATES", "global" )
jobfile.writec( "MSC.Nastran VERSION", "2017." )
jobfile.writec( "WRITE STORED PRECISION", "TRUE" )
jobfile.writec( "PROPS ON ELEM ENTRY", "FALSE" )
jobfile.writec( "CONTINUATION ENTRY", "FALSE" )
jobfile.writec( "PCOMPG ENTRY", "TRUE" )
jobfile.writec( "CONVERT CBAR CBEAM", "FALSE" )
jobfile.writec( "ITERATIVE SOLVER", "FALSE" )
jobfile.writei( "SUPER ELEMENT 0", 0 )
jobfile.writec( "SEALL WRITE", "FALSE" )
jobfile.writec( "PART SUPERELEMENT CREATE", "TRUE" )
jobfile.writec( "AUTOQSET", "FALSE" )
jobfile.writec( "FIXEDB", "FALSE" )
jobfile.writei( "SUPER TREE COUNT", 0 )
jobfile.writec( "MODEL TOLERANCE", "5.0000001E-25" )
jobfile.writec( "ELEMENT PROPERTY OFFSET", "0" )
jobfile.writec( "MATERIAL PROPERTY OFFSET", "0" )
jobfile.writec( "TABLE OFFSET", "0" )
jobfile.writec( "LOAD SET OFFSET", "0" )
jobfile.writec( "LOAD CASE OFFSET", "0" )
jobfile.writec( "CONTROL SET OFFSET", "0" )
jobfile.writec( "RIGID ELEMENT OFFSET", "0" )
jobfile.writec( "SCALAR POINT OFFSET", "0" )
jobfile.writec( "BEGINNING CONTINUATION MARKER", "+      A"
    )
jobfile.writec( "NUMBER ONLY", "ON" )
jobfile.writec( "BEGINNING NUMBER", "OFF" )
jobfile.writec( "TRAILING NUMBER", "OFF" )
jobfile.writec( "SYNTAX NUMBER", "ON" )
jobfile.writec( "SYNTAX MARKER", "." )
jobfile.writec( "EXTERNAL SUPERELEMENT METHOD", "NONE" )
jobfile.writec( "GRID COORDINATES ROUNDING", "15" )
jobfile.writec( "COORD DATA PRECISION ROUNDING", "15" )
jobfile.writec( "MPC DATA PRECISION ROUNDING", "15" )
jobfile.writec( "LBC DATA PRECISION ROUNDING", "7" )
jobfile.writec( "MAT DATA PRECISION ROUNDING", "15" )
jobfile.writec( "PROP DATA PRECISION ROUNDING", "7" )
jobfile.writec( "", "" )
jobfile.writec( "", "/*" )
jobfile.writec( "", " * Solution Parameters" )
jobfile.writec( "", " */" )
jobfile.writec( "", "" )
jobfile.writec( "SOLUTION TYPE", "BUCKLING" )
jobfile.writei( "SOLUTION SEQUENCE", 105 )
jobfile.writec( "DATABASE RUN", "ON" )
jobfile.writec( "CYCLIC SYMMETRY", "OFF" )
jobfile.writec( "AUTOMATIC CONSTRAINTS", "ON" )
jobfile.writec( "SHELL NORMAL TOLERANCE", "" )
jobfile.writec( "MASS CALCULATION", "Lumped" )
jobfile.writec( "DATA DECK ECHO", "None" )
jobfile.writec( "PLATE RZ STIFFNESS FACTOR", "100.0" )
jobfile.writec( "MAXIMUM PRINTED LINES", "" )
jobfile.writec( "MAXIMUM RUN TIME", "" )
jobfile.writec( "WT-MASS CONVERSION", "1.0" )
jobfile.writec( "NODE ID FOR WT-GENER", "" )
jobfile.writec( "RIGID ELEMENT TYPE", "LINEAR" )
jobfile.writec( "USE SOL600", "OFF" )
jobfile.writec( "EXTRACTION METHOD", "Lanczos" )
jobfile.writec( "LOWER EIGENVALUE", "" )
jobfile.writec( "UPPER EIGENVALUE", "" )
jobfile.writec( "NUMBER OF DESIRED ROOTS", "1" )
jobfile.writec( "DIAGNOSTIC OUTPUT LEVEL", "0" )
jobfile.writec( "NORMALIZATION METHOD", "Maximum" )
jobfile.writec( "USE CONTACT TABLE", "OFF" )
jobfile.writec( "INITIAL CONTACTPAIR LOADCASE NAME", "" )
jobfile.writei( "MDOF DATA", 0 )
jobfile.writec( "CELL WRITE", "ON" )
jobfile.writei( "CELL INPUT 0", 0 )
jobfile.writec( "FMS WRITE", "ON" )
jobfile.writei( "FMS INPUT 0", 0 )
jobfile.writec( "EXEC WRITE", "ON" )
jobfile.writei( "EXEC INPUT 0", 0 )
jobfile.writec( "CASE WRITE", "ON" )
jobfile.writei( "CASE INPUT 0", 0 )
jobfile.writec( "BULK WRITE", "ON" )
jobfile.writei( "BULK INPUT 0", 0 )
jobfile.writec( "CELL DTI POSITION", "START" )
jobfile.writec( "FMS DTI POSITION", "START" )
jobfile.writec( "EXEC DTI POSITION", "START" )
jobfile.writec( "CASE DTI POSITION", "START" )
jobfile.writec( "BULK DTI POSITION", "START" )
jobfile.writec( "", "END" )
jobfile.close(  )
mscnastran_job.associate_subcases( "105", "hexagon_buckling
   ", 1, ["Default"] )
analysis_submit_2( "MSC.Nastran", "hexagon_buckling" )
END FUNCTION
```