T.C.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL

FINITE ELEMENT APPROACH TO BENDING AND FREE VIBRATION ANALYSIS OF STRUT-BASED LATTICE REINFORCED BEAMS FOR VIBRATION ISOLATION

ENGİN YILMAZ

A THESIS OF MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

ADVISOR: ASSIST. PROF. DR. MOHAMMAD RAHMANIAN

JANUARY 2026

T.C.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL

FINITE ELEMENT APPROACH TO BENDING AND FREE VIBRATION ANALYSIS OF STRUT-BASED LATTICE REINFORCED BEAMS FOR VIBRATION ISOLATION

ENGİN YILMAZ

A THESIS OF MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

ADVISOR: ASSIST. PROF. DR. MOHAMMAD RAHMANIAN

JANUARY 2026

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

TİTREŞİM YALITIMI İÇİN ÇUBUK ESASLI KAFES TAKVİYELİ KİRİŞLERİN EĞİLME VE SERBEST TİTREŞİM ANALİZİNDE SONLU ELEMANLAR YAKLAŞIMI

ENGİN YILMAZ

YÜKSEK LİSANS TEZİ
MAKİNE MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: DR. ÖĞR. ÜYESİ MOHAMMAD RAHMANIAN

OCAK 2026

MASTER of SCIENCE JURY APPROVAL FORM

A thesis submitted by Engin YILMAZ, defended on 22/01/2026 before the jury formed with the 20/01/2026 date and 2026/05 numbered decision of the GTU Graduate Administration Board, has been accepted as a MASTER of SCIENCE thesis in the Department of Mechanical Engineering

JURY

MEMBER
(THESIS ADVISOR) : Assist. Prof. Dr. Mohammad RAHMANIAN

MEMBER : Assist. Prof. Dr. Saeed LOTFAN

MEMBER : Assist. Prof. Dr. Poyraz AYDINER

APPROVAL

Gebze Technical University Graduate Administration Board

…..…/…..…/..…… date and ……/…….. numbered decision.

SIGNATURE/SEAL

To my Family and Love

ABSTRACT

Vibration control is important part of engineering, particularly in the aerospace and automotive sectors. In these industries, managing vibrations is necessary to provide safety, performance, and comfort. Since beams are basic structural components, it is important to understand their bending and vibration behavior. Traditional solid beams are already well understood; however, recent progress in manufacturing has allowed for the creation of beams reinforced with internal lattice structures. These designs are innovative because they are significantly lighter and can improve vibration management.

In this thesis, the bending and vibration behavior of these lattice reinforced beams is investigated in detail. Specific focus is placed on their ability to isolate vibrations. Six different lattice unit cell designs, which serve as the basic building blocks of the structure, were examined to evaluate their performance. Some designs were developed to resist bending, while others handle stretching differently, leading to varied mechanical behaviors. By simulating these configurations within beam models, the study aims to show how the lattice shape affects the overall stiffness, natural frequencies, and deflection of the beam.

A multiscale approach using the finite element method was followed for the analysis. First, small scale lattice units were modeled to determine their individual mechanical properties. These properties were then integrated into a larger model of the entire beam. Simulations were performed under various conditions to assess both static bending and dynamic free vibration. A primary goal was to identify designs with lower natural frequencies, as this is directly linked to better vibration isolation. To ensure accuracy, the findings were verified against existing literature and validated using ANSYS software. Finally, parametric studies were performed to find the optimal lattice designs that balance low weight, stiffness, and effective vibration control. This research provides useful insights for designing efficient, lightweight structures in advanced engineering applications.

Keywords: Euler-Bernoulli, Representative Volume Elements, Lattice Structures, Finite Element Method, Vibration Isolation

ÖZET

Titreşim yalıtımı, aşırı titreşimlerin azaltılmış yapısal bütünlüğe, performans kaybına ve rahatsızlığa yol açabildiği havacılık, otomotiv ve inşaat yapıları dahil olmak üzere çok çeşitli mühendislik uygulamalarında kritik bir rol oynamaktadır. Kirişler, hem statik hem de dinamik yüklemeye maruz kalan en temel yapısal bileşenler arasındadır ve bu da eğilme ve serbest titreşim davranışlarını yapısal mekanikte merkezi bir konu haline getirmektedir. Dolu kesitli geleneksel kiriş tasarımları kapsamlı bir şekilde incelenmiştir; ancak, eklemeli imalat ve mimari malzemelerdeki son gelişmeler, hafif tasarım ve gelişmiş titreşim performansı için önemli potansiyel sunan kafes takviyeli kirişlerin geliştirilmesini sağlamıştır.

Bu tezde, çubuk esaslı kafes yapılarla güçlendirilmiş kirişlerin eğilme ve titreşim özellikleri detaylı bir şekilde incelenmiştir. Çalışmanın asıl odak noktası, bu yapıların titreşimi ne kadar iyi yalıtabildiğidir. Kirişlerin iç yapısında takviye olarak kullanılmak üzere, dokuz düğümlü çubuklardan oluşan altı farklı kafes tasarımı seçilmiştir. Bu tasarımlar arasında hem eğilmeye hem de uzamaya karşı direnç gösteren farklı yapılar bulunmaktadır; bu da her birinin sertlik ve hareket tepkisinin farklı olmasını sağlar. Bu çalışma, seçilen kafes şekillerinin kirişin toplam sertliğini, doğal frekanslarını ve yük altındaki sehim miktarını nasıl değiştirdiğini ortaya koymayı amaçlamaktadır.

Yöntem olarak çok ölçekli bir sonlu elemanlar analizi kullanılmıştır. İlk aşamada, küçük ölçekli kafes birimleri modellenerek bunların mekanik özellikleri hesaplanmıştır. Daha sonra bu özellikler, büyük ölçekli kiriş modeline aktarılmıştır. Kirişlerin hem sabit durumdaki eğilme hareketleri hem de serbest titreşimleri farklı koşullar altında test edilmiştir. Özellikle düşük doğal frekans değerlerine odaklanılmıştır; çünkü düşük frekans, titreşimin daha iyi yalıtılması anlamına gelmektedir.

Elde edilen sonuçların doğruluğundan emin olmak için veriler, hem geçmiş çalışmalarla karşılaştırılmış hem de ANSYS yazılımı kullanılarak kontrol edilmiştir. Ayrıca, hangi kafes tasarımının en hafif, en sert ve en iyi yalıtım performansına sahip olduğunu bulmak için ek çalışmalar yapılmıştır. Bu araştırmanın sonuçları, hafif ve yüksek performanslı yapılar tasarlamak isteyen mühendisler için yol gösterici bilgiler sunmaktadır.

Anahtar Kelimeler: Euler-Bernoulli, Temsili Hacim Elemanları, Kafes Yapılar, Sonlu Elemanlar Yöntemi, Titreşim Yalıtımı

ACKNOWLEDGEMENTS

I am very grateful to my supervisor, Assist. Prof. Dr. Mohammad RAHMANIAN, for his guidance and patience throughout this study. His deep knowledge and support made this work possible. I am also deeply grateful to my family for their endless love. Finally, I would like to express my deepest gratitude to Res. Assist. Nazlıcan ÖN for her meaningful support during this period.

TABLE OF CONTENTS

	Page
ABSTRACT	vi
ÖZET	vii
ACKNOWLEDGEMENT	viii
TABLE of CONTENTS	ix
LIST OF SYMBOLS AND ABBREVIATIONS	xi
LIST of FIGURES	xii
LIST of TABLES	xiv
1. INTRODUCTION	1
1.1. Background and Motivation	1
1.2. Fundamentals of Vibration Isolation in Beam Structures	5
1.3. Representative Volume Element (RVE) Modeling	6
1.4. Objectives and Scope of the Thesis	7
2. LITERATURE REVIEW	8
2.1. Overview of Lattice Structures	8
2.2. Strut and Surface Based Cells	9
2.3. Vibration Analysis of Lattice Structures	11
2.4. Applications in Aerospace Systems	12
3. THEORETICAL BACKGROUND	13
3.1. Fundamentals of Beam Theory	13
3.1.1. Static Equilibrium of EB Beams and FEM Discretisation	14
3.1.2. Free Vibration of Beams	25
3.2. Fundamentals of Vibration Isolation	26
3.3. Dynamic Performance Metrics	28
4. METHODOLOGY	29
4.1. Lattice Unit Cell and RVE Geometry	29
4.2. Material Properties	32
4.3. Boundary Conditions and Loading Scenarios	33
4.4. Analysis Procedure	36
4.5. Model Verification Strategy	38
5. NUMERICAL RESULTS AND DISCUSSION	39
5.1. Bending Analysis Results	39
5.1.1. Verification of The Finite Element Implementation	39
5.1.2. Bending Performance and Structural Analysis	41
5.1.2.1. Tip Load Analysis	41
5.1.2.2. Spanwise Load Analysis	43
5.1.2.3. Mid-Load Analysis	44
5.1.2.4. Combined Load Analysis	46
5.1.2.5. Torsional Load Analysis	48
5.1.2.6. Mass Evaluation of Lattice Topologies	49
5.2. Free Vibration Analysis Results	50
5.2.1. Verification of the Vibration Analysis	51
5.2.2. Vibration Characteristics Under Clamped-Free (C-F) Conditions	52

5.2.3. Vibration Characteristics Under Clamped-Clamped (C-C) Conditions 57
5.3. Hybrid Lattice Beams Analysis 61
5.3.1. Static Stress Analysis of Hybrid Lattice Beams 63
5.3.2. Free Vibration Analysis of Hybrid Lattice Beams 64
6. CONCLUSION 70

REFERENCES 72
BIOGRAPHY 77
PUBLICATIONS AND PRESENTATIONS FROM THE THESIS 78
APPENDICES 79

LIST OF SYMBOLS AND ABBREVIATIONS

CAD	:	Computer-Aided Design
DOF	:	Degree of Freedom
AM	:	Additive Manufacturing
EB	:	Euler-Bernoulli
FEA	:	Finite Element Analysis
RVE	:	Representative Volume Element
A	:	Cross-sectional Area (m^2)
E	:	Modulus of Elasticity (Pa)
G	:	Shear Modulus (Pa)
I	:	Moment of Inertia (m^4)
K, k	:	Stiffness Matrix / Stiffness (N/m)
M, m	:	Mass Matrix / Mass (kg)
F	:	Force (N)
d, x	:	Displacement (m)
f	:	Frequency (Hz)
r	:	Frequency Ratio
ω	:	Circular Frequency (rad/s)
ρ	:	Density (kg/m^3)
ν	:	Poisson's Ratio
σ	:	Stress (Pa)
ε	:	Strain
ζ	:	Damping Ratio
N	:	Number of Degrees of Freedom / Nodes
l	:	Length (m)

LIST OF FIGURES

		Page
Figure 1.1:	Acoustic environment during lift-off [1].	1
Figure 1.2:	Metamaterial integration in turbine blades [2].	2
Figure 1.3:	Payload adaptor and ESPA ring interface [3].	3
Figure 1.4:	Spacecraft integration in fairing [4].	4
Figure 1.5:	Damping metamaterials for airplane wings [2].	4
Figure 1.6:	Vibration damping vs. isolation [3].	5
Figure 1.7:	Vibration isolation ring system [3].	6
Figure 1.8:	Examples of lattice material [5].	7
Figure 2.1:	Typical cellular structures.	9
Figure 2.2:	Strut vs. surface unit cells [5].	10
Figure 3.1:	Beam element with positive nodal variables [6].	13
Figure 3.2:	Differential beam element [6].	14
Figure 3.3:	Oriented beam element [6].	19
Figure 3.4:	Bending about two axes [6].	22
Figure 3.5:	Vibration damping vs. isolation [3].	26
Figure 3.6:	Typical mode shapes for basic beams.	28
Figure 4.1:	Unit cell and strut geometry [56].	29
Figure 4.2:	Lattice nodes and struts illustration [3].	30
Figure 4.3:	Lattice structures: NxNxN sizes [56].	30
Figure 4.4:	Six models of lattice unit cells [3]	31
Figure 4.5:	Axial forces and stresses on strut [3].	35
Figure 4.6:	Displacement and force: Local vs. global [3].	35
Figure 5.1:	MATLAB verification results: Model-1.	39
Figure 5.2:	ANSYS verification: Model-1 results.	40
Figure 5.3:	MATLAB verification: Complex lattice models.	41
Figure 5.4:	Tip load analysis	41
Figure 5.5:	Deflection vs. Length: Tip distributed loading.	42
Figure 5.6:	Spanwise load analysis	43
Figure 5.7:	Deflection vs. Length: Spanwise loading.	44
Figure 5.8:	Mid-load analysis configuration	45
Figure 5.9:	Deflection vs. Length: Mid-distributed loading.	45
Figure 5.10:	Combined spanwise and chordwise loading configuration	46
Figure 5.11:	Deflection vs. Length: Combined Y-Z loading.	47
Figure 5.12:	Torsional loading configuration	48
Figure 5.13:	Rotation vs. Length: Torsional loading.	49
Figure 5.14:	Mass performance: Lattice beams.	50
Figure 5.15:	Lattice mode shapes: Models 1-2.	51
Figure 5.16:	Lattice mode shapes: Model-4.	52
Figure 5.17:	C-F mode shapes: Model-1.	54
Figure 5.18:	C-F mode shapes: Model-2.	54
Figure 5.19:	C-F vibration mode shapes: Models 3-4.	55
Figure 5.20:	C-F vibration mode shapes: Models 5-6.	56
Figure 5.21:	C-C mode shapes for Model-1 and Model-2	58
Figure 5.22:	C-C mode shapes for Model-3 and Model-4	59

Figure 5.23: C-C mode shapes for Model-5 and Model-6 60
Figure 5.24: Hybrid lattice beams with different sections 61
Figure 5.25: Static stress distribution for hybrid lattice beams 63
Figure 5.26: Modal analysis for Hybrid Cases 1 and 2 67
Figure 5.27: First natural frequency (f_1) for all 180 hybrid scenarios 68
Figure 5.28: Modal analysis for Hybrid Cases 3 and 4 69

LIST OF TABLES

		Page
Table 4.1:	Mechanical properties of the lattice material	33
Table 5.1:	Verification results: MATLAB vs. ANSYS.	40
Table 5.2:	Deflection vs. Length: C-F tip loading.	43
Table 5.3:	Deflection vs. Length: C-F spanwise loading.	44
Table 5.4:	Deflection vs. Length: C-C mid-distributed loading.	46
Table 5.5:	Deflection vs. Length: C-F combined loading.	47
Table 5.6:	Rotation vs. Length: C-F torsional loading.	48
Table 5.7:	Total mass and performance of topologies.	50
Table 5.8:	Free Vibration Verification: Results comparison.	53
Table 5.9:	Natural Frequencies: Lattice Beams (C-F).	53
Table 5.10:	Natural Frequencies: Lattice Beams (C-C).	57
Table 5.11:	Possible hybrid design configurations	62
Table 5.12:	Natural Frequency Results for 180 Hybrid Configurations	64

1. INTRODUCTION

1.1. Background and Motivation

Vibration isolation is a major problem when designing and building aircraft and spacecraft. These systems need to work perfectly in tough conditions, but they also have to be light, small, and simple. Unlike machines used on the ground, vehicles in space and air face very strong vibrations. If these vibrations are not controlled, they can cause big problems, such as breaking the structure or causing important equipment to fail.

In aerospace applications, vibration comes from many sources, including engines, air pressure, and interactions between different parts of the structure. For space systems, the most difficult vibration environment happens during launch. During lift-off and ascent, satellites and instruments onboard are subjected to intense random vibration, loud acoustic noise, and sudden shock loads from the rocket. These dynamic loads travel through the structure and, if they are not reduced, can cause high stress, fatigue damage, or even potential malfunction of sensitive equipment, putting the mission at risk.

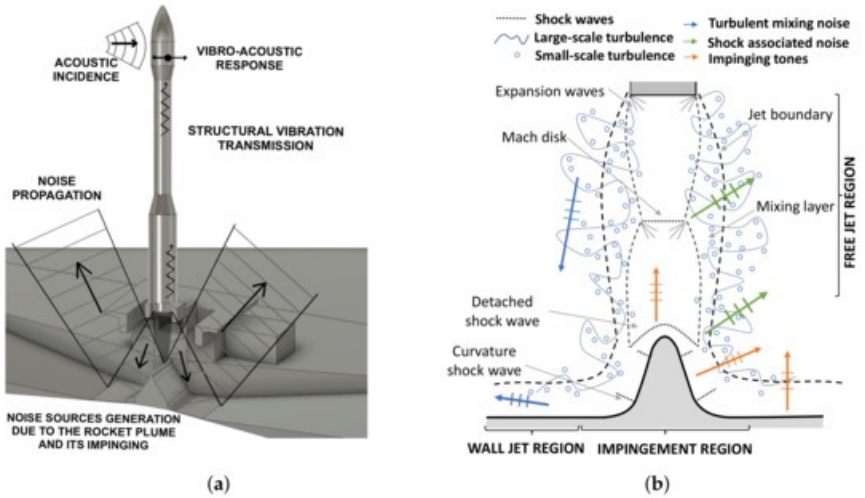


Figure 1.1: Acoustic environment during lift-off [1].

The effect of vibration on how long a structure lasts (fatigue life) is a major concern. Aerospace structures, which vibrate constantly during operation, can suffer from fa-

tigue damage in their components and connections. Repeated shaking creates stress that can start tiny cracks, loosen bolts, and wear down joints. In space systems, this is especially critical during launch, where the shaking is very intense for a short period.

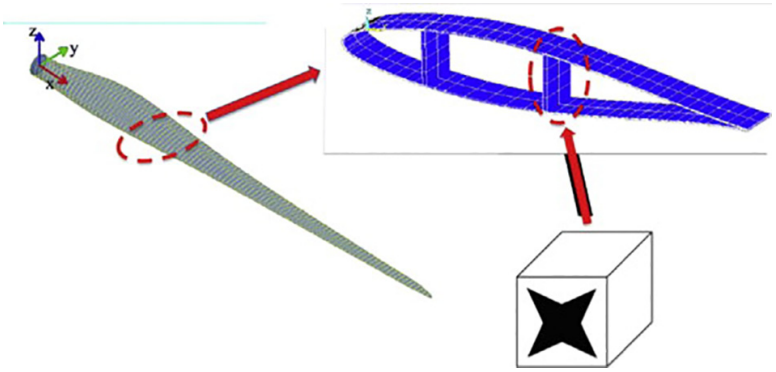


Figure 1.2: Metamaterial integration in turbine blades [2].

Historically, engineers have tried to stop vibrations by making structures stiffer or adding mass. While stiff structures stop things from moving too much, they don't actually isolate vibration; they just reduce the amplitude while still passing a lot of energy to the system. Adding mass to stop vibration is also not a good solution for aerospace, because every extra kilogram makes launch much more expensive and limits what else can be carried. These limits show that we need better strategies that can stop vibration transmission without making things too heavy or overly stiff.

Conceptually, "vibration isolation" is very different from "vibration damping." Damping works by dissipating energy (like a shock absorber) to reduce shaking at resonance. Isolation, on the other hand, aims to stop vibration from ever reaching the structure by mechanically separating the source from the system. Effective isolation happens when the natural frequency of the system is much lower than the frequency of the vibration. This places the system in an "isolation region" where the shaking doesn't get through. This idea is the basis for many isolation methods.

In satellite applications, vibration isolation is especially critical at the connection between the payload (like a satellite) and the launch vehicle. The payload adaptor, which connects them, is usually designed to be very stiff to hold the satellite safely during launch. However, this stiffness also allows vibration to pass straight from the rocket to the satellite. Therefore, modern payload adaptors need to do two things at once:

provide strong support and stop vibration from reaching sensitive parts.

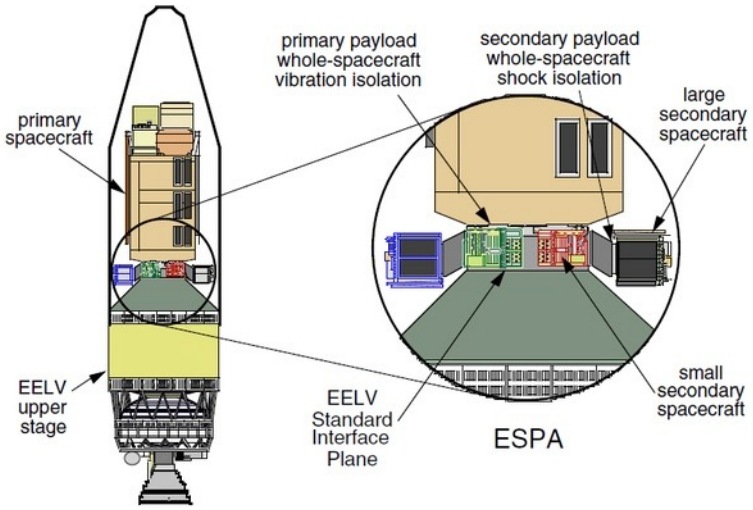


Figure 1.3: Payload adaptor and ESPA ring interface [3].

Designing isolation for this interface is complicated because many different physical forces interact at the same time. The adaptor is more than just a part that connects things; it is where physical pressure, sound, and movement all meet. During a rocket launch, vibrations are ”broadband,” which means they happen at many different speeds and change constantly. Because of this, the isolation system must work everywhere. It has to handle both the slow shaking of the structure and the fast noise caused by sound.

There are also conflicting goals. The adaptor must be strong and stiff enough to carry heavy weights during launch. However, if it is too stiff, it becomes worse at stopping vibrations. Also, there is very little space inside the rocket, so the isolation parts cannot be too big. At the same time, we cannot add heavy materials because every gram matters in space.

Because these problems are so complex, a simple fix is not enough. We have to be very careful when choosing the weight, strength, and shape of the parts. This is why engineers are now looking at advanced ideas, like using lattice reinforced structures. These designs allow us to arrange material at a micro-level to get exactly the stiffness and natural frequency we need, without adding unnecessary weight.

Recent advances in 3D printing (additive manufacturing) have made it possible to build these complex lattice structures inside real parts. By combining lightweight design

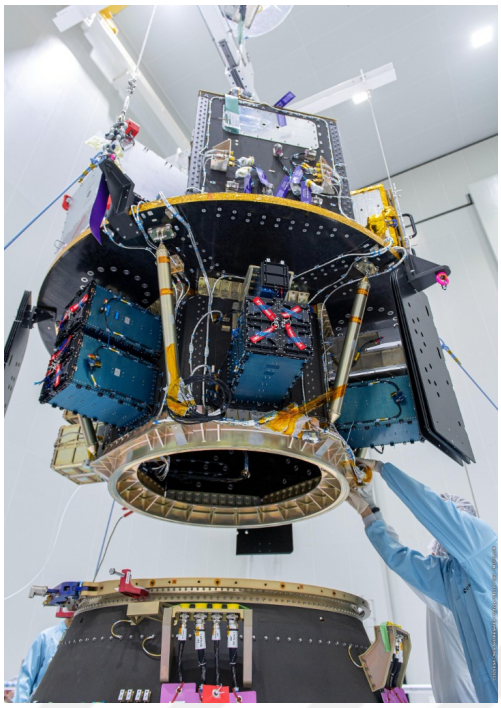


Figure 1.4: Spacecraft integration in fairing [4].

with tunable mechanical properties, lattice reinforced beams offer a great solution for the difficult trade-off between vibration isolation and structural strength. This thesis focuses on investigating these strut based lattice reinforced beams, looking specifically at how they bend and vibrate, to see how well they can isolate vibrations in aerospace systems.

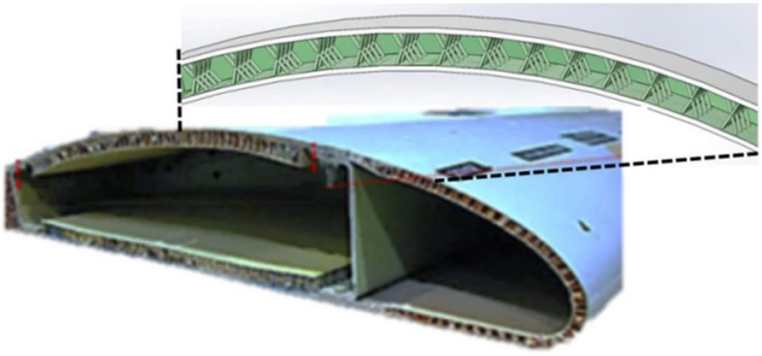


Figure 1.5: Damping metamaterials for airplane wings [2].

1.2. Fundamentals of Vibration Isolation in Beam Structures

It is important to clearly understand the difference between ”vibration damping” and ”vibration isolation,” especially in aerospace where weight limits are strict. Figure 1.6 shows the typical response of a system to vibration (its natural frequency response).

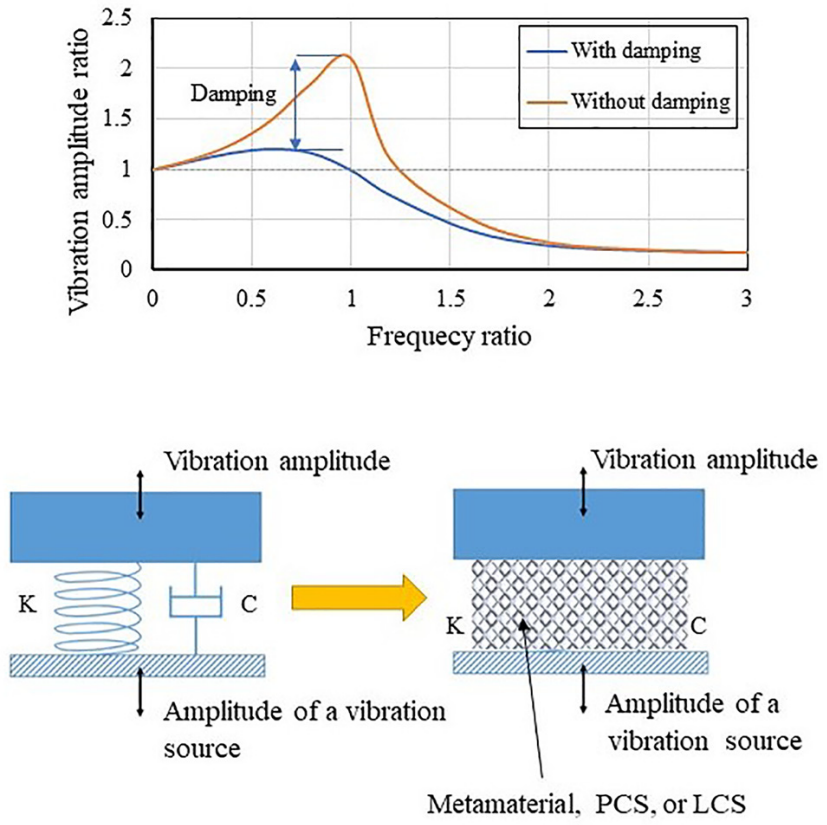


Figure 1.6: Vibration damping vs. isolation [3].

As the figure shows, ”damping” mainly helps to cut down the peak vibration at resonance (when the shaking matches the structure’s natural rhythm). Damping absorbs energy and reduces vibration, but it does not stop the shaking from moving through the structure. Because of this, damping helps reduce the size of vibrations, but it is not true isolation. Vibration isolation only happens when the shaking is much faster than the structure’s natural frequency. In this ”isolation zone,” the amount of vibration that passes through the structure is reduced a lot. To get good isolation, the structure must be designed so its natural frequency stays lower than the vibration source. By lowering this frequency, the structure stays in the isolation zone, which blocks most of the

vibration. This is very important for space missions because heavy materials cannot be added due to weight limits.



Figure 1.7: Vibration isolation ring system [3].

1.3. Representative Volume Element (RVE) Modeling

In the study of large lattice structures, homogenization is used to connect the small unit cell to the full structure. This method is used to calculate the overall material properties based on the behavior of the repeating small parts. By using this approach, the complex lattice is treated as a solid, uniform material. This cuts down the time and power needed for computer calculations.

While modeling a full lattice structure is often too difficult because of the huge number of parts, the Representative Volume Element (RVE) method is used as a faster alternative. By modeling just one unit cell (RVE) and applying the right boundary conditions, the overall stiffness of the lattice can be found. This method assumes that the unit cell is much smaller than the whole structure.

Equivalent energy methods are also used to predict these properties. In this technique, the energy stored in the solid material is set to be equal to the average energy in the small RVE. Through this method, the strength of complex 3D shapes can be calculated with only a few tests. Although this assumes the pattern repeats perfectly, it is still a powerful tool for designing lattice structures. It allows for the testing of many designs without the need for long and difficult full-scale simulations.

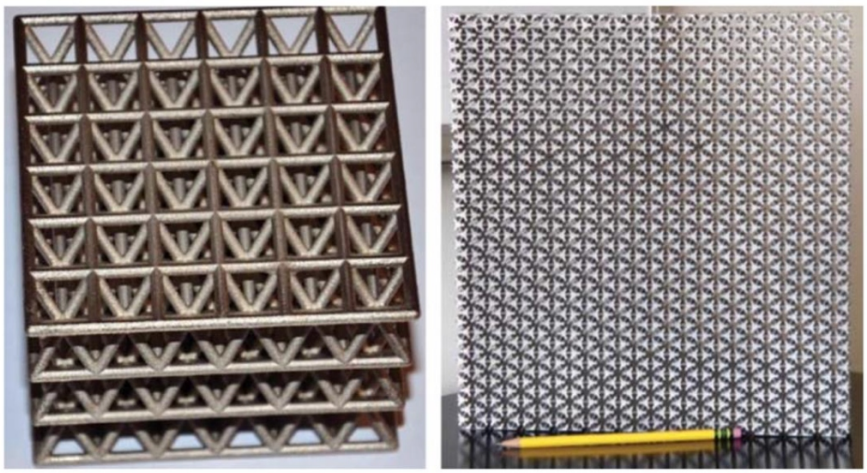


Figure 1.8: Examples of lattice material [5].

1.4. Objectives and Scope of the Thesis

The main goal of this thesis is to study how lattice reinforced beams bend and vibrate using the finite element method. The study aims to see how different lattice shapes change the beam's stiffness and movement.

Specifically, this thesis looks at how the lattice shape affects the natural frequencies, mode shapes, bending deflections, and overall structural efficiency. By comparing lattice reinforced beams with conventional solid beams, the improvements in vibration isolation capability and load-carrying performance are assessed. The main natural frequency is a key factor for vibration isolation, as a lower natural frequency leads to a better isolation of vibrations. The scope of this thesis is limited to linear elastic beam behavior under small deformation assumptions. Bending and free vibration analyses are based on Euler-Bernoulli beam theory, neglecting shear deformation and rotary inertia effects. The lattice reinforcements are modeled using strut based representative volume elements (RVEs), which are added to the beam structure using a finite element model. Only selected lattice configurations are considered in order to reduce the design space and allow systematic comparison. The analyses are performed using finite element models developed in this study, and selected results are verified through comparisons with available literature and commercial finite element software. Nonlinear effects, material damping, and experimental validation are not included in this study. A specific focus is placed on how well these designs can isolate vibrations.

2. LITERATURE REVIEW

2.1. Overview of Lattice Structures

Engineers are paying close attention to these structures because they offer an excellent balance of being lightweight while remaining very strong and stiff [7, 8]. In the past, they were simply grouped together with foams and honeycombs as ”cellular materials” [7, 9, 10]. However, there are clear differences in how regular they are and how their cells are defined [11, 12]. To be more precise, researchers now classify them into three distinct groups: foams, honeycombs, and lattice structures [8, 13, 14]. Foams have a stochastic (random) structure where cell walls face in arbitrary directions. This means they behave roughly the same in all directions (isotropic) but their mechanical properties can vary a lot (Figure 2.1a). Metal foams, for example, are often used but their properties depend heavily on their porosity [15–17].

Honeycomb structures are different because they are made of 2D arrays of regularly shaped cells, like hexagons or squares (Figure 2.1b) [10, 18]. Recently, ”auxetic” honeycombs (Figure 2.1c) have become popular. These unique structures get wider when stretched (negative Poisson’s ratio) and offer better shear resistance compared to standard shapes [19, 20].

Lattice structures are a distinct class where 3D unit cells are arranged in a specific pattern to create a network of struts and nodes (Figure 2.1d–j) [13, 21]. They can be ”periodic” (repeating the same cell) or ”pseudo-periodic” (where the cells change gradually). For example, we can gradually change the cell size (Figure 2.1f) or the strut thickness (Figure 2.1g) to create a gradient in properties.

We can also have ”conformal” lattices that change their shape to follow the curved surface of a part (Figure 2.1h), which makes them great for complex geometries. ”Stochastic” lattices (Figure 2.1i) are intentionally random to avoid specific weak points. Finally, ”hybrid” lattices (Figure 2.1j) combine different types of unit cells to get the best specific functional properties from each.

The specific shape (topology) of the unit cell is a critical design choice. As shown in Figure 2.2, we generally categorize them into ”strut based” and ”surface-based” ar-

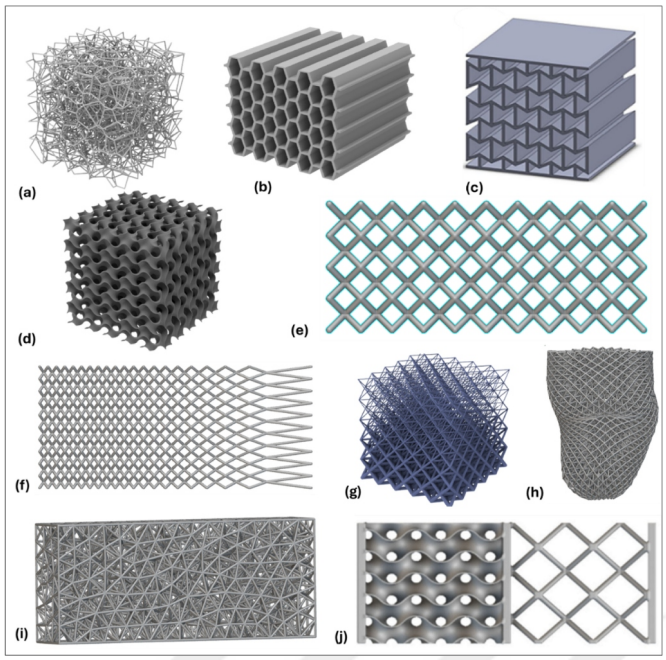


Figure 2.1: Typical cellular structures.

chitectures. Strut-based lattices, like the Body-Centered Cubic (BCC) or Octet truss, are made of nodes connected by beams [3]. These are very popular because they are straightforward to model with finite element methods and offer a wide range of mechanical properties.

2.2. Strut and Surface Based Cells

The mechanical behavior of a strut based lattice depends almost entirely on the geometry of its unit cells, specifically aspects such as how the struts connect, how thick they are, and their length. Recent research shows that we can get very specific mechanical responses by tweaking these parameters. For instance, Van Grunsven et al. [22] showed that by linearly increasing the strut thickness in diamond unit cells, they could create a structure with graded porosity. Similarly, Maskery et al. [23] compared uniform BCC lattices with graded ones. They found that gradually thinning the struts from the bottom to the top improved the structure's ability to absorb energy, all while keeping the overall weight the same.

Choy et al. [24] took this further by varying strut diameters from 0.4 mm to 1.2 mm to create density gradients. They found that these "functionally graded" structures (FGM)

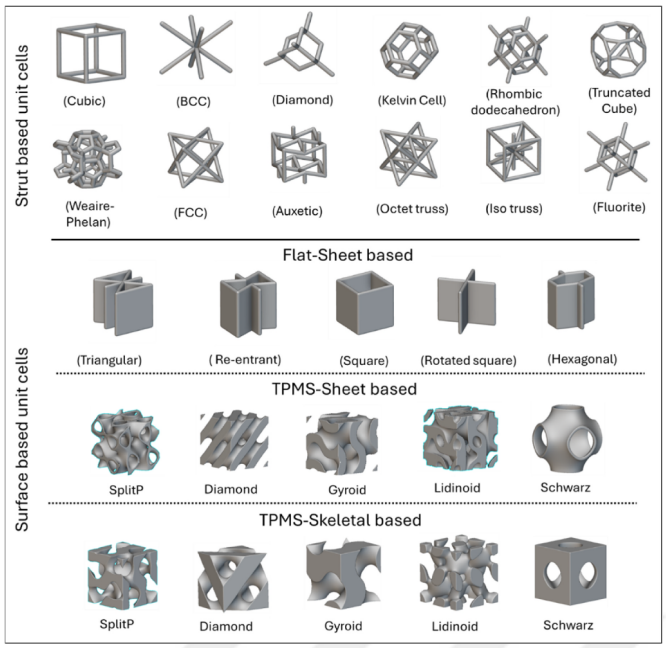


Figure 2.2: Strut vs. surface unit cells [5].

failed in a much more controlled way, layer by layer, whereas uniform lattices tended to fail suddenly and catastrophically [24].

Changing the cell size itself is another way to create these gradients. Xiao and Song [25] used varying cell sizes to distribute loads more evenly and reduce stress concentrations. Whatever the method, these gradient approaches generally improve stiffness and strength compared to uniform designs. Researchers like Altamimi et al. [26] also found that ”hybrid” topologies (mixing different cell shapes) could solve issues like directional weakness (anisotropy).

Understanding how these structures break is also key. Obadimu and Kourousis [27] reviewed failure modes and found they depend heavily on how struts are connected. Some designs are ductile (they stretch), while others are brittle (they snap), as shown by Geng et al. [28]. Even mixing hard and soft materials in the same lattice can be used to tune stiffness [29].

2.3. Vibration Analysis of Lattice Structures

The literature contains many studies on how lattice structures vibrate and how well they isolate vibrations. Researchers have used both computer simulations (Finite Element Analysis or FEA) and real-world experiments to understand these behaviors.

There are two main ways to simulate these complex structures: using full Finite Element Analysis (FEA) or a simplified ”homogenization” method. Al-Ketan et al. [30] tested various lattice shapes to see how their geometry affects their mechanical properties. They found that ”sheet-diamond” cellular structures performed the best in compression tests.

When using FEA, engineers can model the lattice in two ways: using detailed ”3D elements” (which are precise but slow) or simpler ”beam elements” (which are faster). Luxner et al. [31] compared these two methods and showed that simple beam elements work well for representing the mechanical behavior of lattices, saving a lot of computing time. Smith et al. [32] confirmed this by comparing beam-based computer models with experimental results from real 3D-printed parts, showing a good match.

Many researchers have specifically looked at vibration. Early on, Martinsson and Movchan [33] studied the natural vibrations of lattice structures. Later, Jaouen et al. [34] performed experiments to measure how porous materials behave when vibrating. Sun et al. [35] took this further by analyzing special ”metamaterial” beams designed to stop vibrations across a wide range of frequencies.

More specific designs have also been tested. Baravelli and Ruzzene [36] looked at ”chiral” lattices (which have a twisting geometry) in cantilever beams to control low-frequency vibrations. Wu et al. [37] analyzed the vibration properties of the ”Kagome” lattice, while Mao et al. [38] studied how porous materials can act as vibration isolators.

Recently, the focus has shifted to 3D-printed parts and advanced material behaviors. Azmi et al. [39] investigated how the manufacturing process affects the vibration of printed bars, while Ramadani et al. [40] used lattice structures to reduce vibration and weight in gears.

Work has also expanded into hybrid materials. Yin and Rayess [41] and Dahil and Karabulut [42] explored how hybrid foam structures can reduce vibrations in vehicle

seats and other systems. The study of ”bandgaps,” which are frequency ranges where no vibration can pass, has been a major focus. Li et al. [43] and Hajhosseini [44] analyzed how different unit cell shapes create these gaps. Andresen et al. [45] looked at irregular lattices, while another team led by Li [**?**] recently investigated ”4D printed” metamaterials that can change over time.

Advanced shapes like ”gyroids” and ”meta-trusses” are also showing promise. Simsek et al. [46] and An et al. [47] showed that these complex geometries offer excellent isolation performance. Elmadih et al. [48] and Fan et al. [49] focused on manipulating stiffness to block low-frequency vibrations, which are often the hardest to stop. Finally, Sahmani et al. [50] used advanced modeling to understand how nanoporous materials behave under large vibrations.

2.4. Applications in Aerospace Systems

Additive manufacturing is quickly being used in the aerospace industry because it allows for the creation of lattice structures. These structures are both very light and high-performing. The combination of high stiffness and low weight is very important for any vehicle that flies.

Large weight reductions have already been seen in real parts. For example, the weight of an aircraft door hinge was reduced by 44% using optimized lattice structures [4]. Similarly, 24% of the mass was saved from an aircraft bracket [51]. In drones (UAVs), lattice designs have been used to create wings that are flexible but strong [52].

There are also more advanced applications. A 3D lattice thermal protection system was developed for space capsules to manage both physical loads and heat [53]. In wing design, special lattices that can change shape without breaking were studied [54]. Additionally, programmable lattice materials were proposed to create ultralight, adaptive structures [55]. All these examples demonstrate that lattice structures are a truly transformative technology. They allow engineers to move beyond simple solid materials and instead design the material itself to meet specific structural needs. While we already know a lot about their strength, investigating their ability to isolate vibrations is still a very active and critical area of research. Ultimately, these advanced designs will lead to greener, more efficient, and quieter aviation systems.

3. THEORETICAL BACKGROUND

3.1. Fundamentals of Beam Theory

A beam is a structural element which is mainly subjected to transverse loads, due to the material design, resulting in a prominent bending behavior and little axial and torsional effects. The deformation in its resulting configuration can be characterized by a transverse displacement and an associated rotation. Thus each node is defined by two degrees of freedom: one transverse displacement and one rotational degree of freedom.

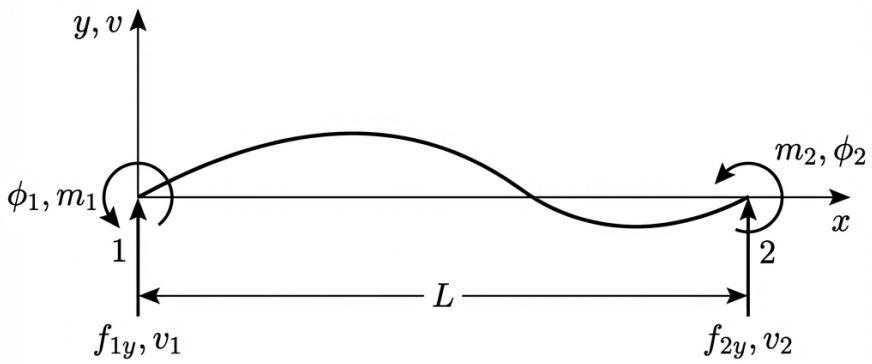


Figure 3.1: Beam element with positive nodal variables [6].

The equations of motion for beams are formulated based on the Euler-Bernoulli, Rayleigh, and Timoshenko beam theories. In the Euler-Bernoulli theory, both rotary inertia and shear deformation effects are neglected, making it suitable for the analysis of slender beams. The Rayleigh theory extends this formulation by incorporating the influence of rotary inertia, while the Timoshenko theory accounts for both rotary inertia and shear deformation effects. As a result, the Timoshenko theory is generally preferred for the analysis of thick beams. Beam responses under various conditions, such as moving loads, axial forces, rotational motion, continuous beam configurations, and elastic foundation support, are commonly analyzed using the thin beam assumption provided by the Euler-Bernoulli theory. Free vibration analysis, including the evaluation of natural

frequencies and corresponding mode shapes, can be performed using all three beam theories, depending on the level of modeling accuracy required.

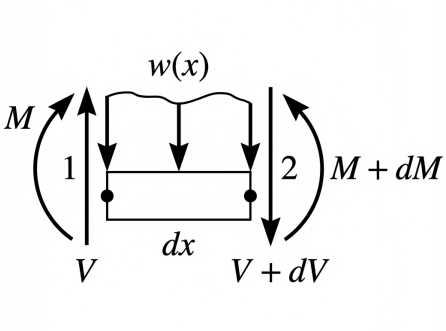


Figure 3.2: Differential beam element [6].

3.1.1. Static Equilibrium of EB Beams and FEM Discretisation

The Euler-Bernoulli beam theory, originally developed by Euler and Bernoulli, describes the linear-elastic behavior of beams through a governing differential equation based on the assumption that plane cross-sections normal to the longitudinal centroidal axis remain plane and perpendicular to this axis after bending. Within the beam formulation, the amplitude of rotation of beam cross-sections is assumed negligible compared to the transverse displacement. Furthermore, angular distortion is assumed to be insignificant relative to bending deformation. This principle is used for slender beams, when beam length is greatly in excess of its depth, typically, at least ten times greater. Under these conditions, the Euler-Bernoulli theory provides an accurate and efficient representation of beam behavior.

The governing differential equation can be derived as follows. Considering Figure 3.2, the force and moment equilibrium of the differential beam element yields:

$$\Sigma F_y = 0 : V - (V + dV) - w(x)dx = 0 \tag{3.1}$$

Simplifying Equation 3.1 leads to the relationship between the distributed load and the shear force:

$$-wdx - dV = 0 \quad \text{or} \quad w = -\frac{dV}{dx} \tag{3.2}$$

Next, considering the moment equilibrium about point 2 ($\Sigma M_2 = 0$), the following expression is obtained:

$$-V dx + dM + w(x)dx\left(\frac{dx}{2}\right) = 0 \tag{3.3}$$

Dividing simplified Equation 3.3 by dx and taking the limit as dx approaches zero causes the higher-order term containing dx to vanish. This yields the fundamental relationship connecting shear force and bending moment:

$$V = \frac{dM}{dx} \tag{3.4}$$

Furthermore, the curvature κ of the beam is related to the internal bending moment M and the flexural rigidity EI as follows:

$$\kappa = \frac{1}{\rho} = \frac{M}{EI} \tag{3.5}$$

where ρ represents the radius of curvature of the deflected beam, E denotes the modulus of elasticity, and I refers to the principal moment of inertia of the cross-section about the neutral axis (z-axis). Under the assumption of small deformations, where the slope can be approximated by the first derivative of the transverse displacement $v(x)$, the curvature is defined as the second derivative of the displacement:

$$\kappa = \frac{d^2v}{dx^2} \tag{3.6}$$

Substituting Equation 3.6 into Equation 3.5 yields the moment-displacement relationship:

$$\frac{d^2v}{dx^2} = \frac{M}{EI} \tag{3.7}$$

Solving this relation for the bending moment M and substituting it into the equilibrium equations derived earlier leads to the fourth-order governing differential equation for the beam under a distributed load $w(x)$:

$$\frac{d^2}{dx^2}\left(EI\frac{d^2v}{dx^2}\right) = -w(x) \tag{3.8}$$

For a beam with constant flexural rigidity EI and subjected only to nodal forces and moments (i.e., $w(x) = 0$), the governing equation simplifies to:

$$EI\frac{d^4v}{dx^4} = 0 \tag{3.9}$$

To formulate the finite element model, the transverse displacement variation through the element length is assumed to be a cubic polynomial:

$$v(x) = a_1 x^3 + a_2 x^2 + a_3 x + a_4 \tag{3.10}$$

The selection of a complete cubic displacement function is justified by the presence of four total degrees of freedom: a transverse displacement v_i and a rotation ϕ_i at each node i. Additionally, this function satisfies the homogeneous governing differential equation for a beam with constant rigidity and ensures the necessary continuity of displacement and slope across inter-element boundaries.

The coefficients a_1 through a_4 are defined by expressing the boundary conditions at the nodes. Letting $x = 0$ at node 1 and $x = L$ at node 2, the nodal conditions are:

$$\begin{aligned} v(0) &= v_1 = a_4 \\ \frac{dv(0)}{dx} &= \phi_1 = a_3 \\ v(L) &= v_2 = a_1 L^3 + a_2 L^2 + a_3 L + a_4 \\ \frac{dv(L)}{dx} &= \phi_2 = 3a_1 L^2 + 2a_2 L + a_3 \end{aligned} \tag{3.11}$$

Solving this system for the coefficients in terms of the nodal degrees of freedom (v_1, ϕ_1, v_2, and ϕ_2) and substituting these back into Equation 3.10 provides the displacement function in terms of the nodal variables:

$$\begin{aligned} v = &\left[\frac{2}{L^3}(v_1 - v_2) + \frac{1}{L^2}(\phi_1 + \phi_2)\right] x^3 \\ &+ \left[-\frac{3}{L^2}(v_1 - v_2) - \frac{1}{L}(2\phi_1 + \phi_2)\right] x^2 + \phi_1 x + v_1 \end{aligned} \tag{3.12}$$

This expression can be rewritten in a compact matrix form as:

$$v = [N]\{d\} \tag{3.13}$$

where $\{d\}$ represents the nodal displacement vector:

$$\{d\} = \begin{Bmatrix} v_1 \\ \phi_1 \\ v_2 \\ \phi_2 \end{Bmatrix} \tag{3.14}$$

and the matrix $[N]$ contains the shape functions:

$$[N] = [N_1 \quad N_2 \quad N_3 \quad N_4] \tag{3.15}$$

The terms N_1 through N_4 are known as Hermite cubic interpolation functions and are defined as:

$$\begin{aligned} N_1 &= \frac{1}{L^3}(2x^3 - 3x^2L + L^3) \\ N_2 &= \frac{1}{L^3}(x^3L - 2x^2L^2 + xL^3) \\ N_3 &= \frac{1}{L^3}(-2x^3 + 3x^2L) \\ N_4 &= \frac{1}{L^3}(x^3L - x^2L^2) \end{aligned} \tag{3.16}$$

The axial strain-displacement relationship is given by:

$$\varepsilon_x(x, y) = \frac{du}{dx} \tag{3.17}$$

where u is the axial displacement function. Based on the Euler-Bernoulli hypothesis, which assumes that plane sections remain plane and perpendicular to the neutral axis, the axial displacement u is related to the transverse displacement v by the following geometric relationship:

$$u = -y\frac{dv}{dx} \tag{3.18}$$

Substituting Equation 3.18 into Equation 3.17, the axial strain can be expressed in terms of the curvature as:

$$\varepsilon_x(x, y) = -y\frac{d^2v}{dx^2} \tag{3.19}$$

By applying Hooke's Law for a linear elastic material, the bending stress is obtained as:

$$\sigma_x = -Ey\frac{d^2v}{dx^2} \tag{3.20}$$

Recalling the moment-curvature relationship, the bending stress formula can be rewritten as:

$$\sigma_x = \frac{-My}{I} \tag{3.21}$$

Additionally, the bending moment $m(x)$ and shear force V can be expressed in terms of the transverse displacement $v(x)$ as:

$$m(x) = EI\frac{d^2v}{dx^2} \quad \text{and} \quad V = EI\frac{d^3v}{dx^3} \tag{3.22}$$

By utilizing the direct equilibrium approach and relating the nodal sign conventions with the beam theory conventions for shear forces and bending moments, the relationships between nodal forces/moments and nodal displacements are obtained. Evaluating the shear force and bending moment expressions at the element nodes ($x = 0$ and $x = L$) yields:

$$\begin{aligned}
f_{1y} &= V(0) = EI\frac{d^3v(0)}{dx^3} = \frac{EI}{L^3}(12v_1 + 6L\phi_1 - 12v_2 + 6L\phi_2) \\
m_1 &= -m(0) = -EI\frac{d^2v(0)}{dx^2} = \frac{EI}{L^3}(6Lv_1 + 4L^2\phi_1 - 6Lv_2 + 2L^2\phi_2) \\
f_{2y} &= -V(L) = -EI\frac{d^3v(L)}{dx^3} = \frac{EI}{L^3}(-12v_1 - 6L\phi_1 + 12v_2 - 6L\phi_2) \\
m_2 &= m(L) = EI\frac{d^2v(L)}{dx^2} = \frac{EI}{L^3}(6Lv_1 + 2L^2\phi_1 - 6Lv_2 + 4L^2\phi_2)
\end{aligned} \tag{3.23}$$

The negative signs appearing in the equations for m_1 and f_{2y} arise from the opposing directions of positive conventions between the nodal definition and the standard beam theory. Organizing these equilibrium equations into a matrix format provides the element stiffness relation:

$$\begin{Bmatrix} f_{1y} \\ m_1 \\ f_{2y} \\ m_2 \end{Bmatrix} = \frac{EI}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}\begin{Bmatrix} v_1 \\ \phi_1 \\ v_2 \\ \phi_2 \end{Bmatrix} \tag{3.24}$$

From this system, the characteristic stiffness matrix $[k]$ for the beam element is identified as:

$$[k] = \frac{EI}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \tag{3.25}$$

For beam elements arbitrarily oriented within a global frame, the relationship between local ($x'y'$) and global (xy) coordinates is established via a transformation matrix. The transformation for translational degrees of freedom is given by:

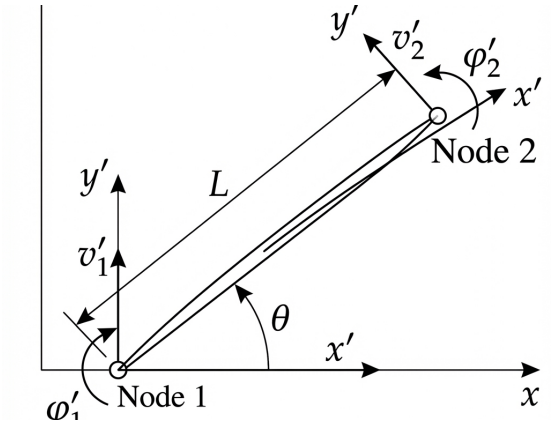


Figure 3.3: Oriented beam element [6].

$$\begin{Bmatrix} u' \\ v' \end{Bmatrix} = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{3.26}$$

where $C = \cos\theta$ and $S = \sin\theta$, with θ representing the orientation angle. The rotational degree of freedom ϕ remains invariant.

Consequently, the transformation matrix $[T]$ relating the local nodal degrees of freedom to the global ones is defined as:

$$[T] = \begin{bmatrix} C & S & 0 & 0 & 0 & 0 \\ -S & C & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & C & S & 0 \\ 0 & 0 & 0 & -S & C & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.27}$$

By applying this transformation to the local stiffness matrix $[k']$ (where axial effects are initially excluded but accounted for in the frame formulation), the global element stiffness matrix $[k]$ is obtained via the operation $[k] = [T]^T[k'][T]$. This yields:

$$[k] = \frac{EI}{L^3}\begin{bmatrix} 12S^2 & -12SC & -6LS & -12S^2 & 12SC & -6LS \\ & 12C^2 & 6LC & 12SC & -12C^2 & 6LC \\ & & 4L^2 & 6LS & -6LC & 2L^2 \\ & & & 12S^2 & -12SC & 6LS \\ & \text{Symmetry} & & & 12C^2 & -6LC \\ & & & & & 4L^2 \end{bmatrix} \tag{3.28}$$

To fully represent a frame element, axial deformation effects must be included alongside bending. By defining the axial stiffness parameter $C_1 = \frac{AE}{L}$ and the bending stiffness parameter $C_2 = \frac{EI}{L^3}$, the complete element stiffness matrix in local coordinates is obtained by superimposing the axial and bending stiffness contributions:

$$\begin{Bmatrix} f'_{1x} \\ f'_{1y} \\ m'_1 \\ f'_{2x} \\ f'_{2y} \\ m'_2 \end{Bmatrix} = \begin{bmatrix} C_1 & 0 & 0 & -C_1 & 0 & 0 \\ 0 & 12C_2 & 6C_2L & 0 & -12C_2 & 6C_2L \\ 0 & 6C_2L & 4C_2L^2 & 0 & -6C_2L & 2C_2L^2 \\ -C_1 & 0 & 0 & C_1 & 0 & 0 \\ 0 & -12C_2 & -6C_2L & 0 & 12C_2 & -6C_2L \\ 0 & 6C_2L & 2C_2L^2 & 0 & -6C_2L & 4C_2L^2 \end{bmatrix} \begin{Bmatrix} u'_1 \\ v'_1 \\ \phi'_1 \\ u'_2 \\ v'_2 \\ \phi'_2 \end{Bmatrix} \tag{3.29}$$

Finally, substituting the expanded transformation matrix $[T]$ and this combined local stiffness matrix $[k']$ into the transformation relationship $[k] = [T]^T[k'][T]$ yields the general transformed global stiffness matrix for a frame element. This comprehensive matrix accounts for axial force, shear force, and bending moment effects:

$$[k] = \frac{E}{L}\begin{bmatrix} AC^2+\frac{12I}{L^2}S^2 & (A-\frac{12I}{L^2})CS & -\frac{6I}{L}S & -(AC^2+\frac{12I}{L^2}S^2) & -(A-\frac{12I}{L^2})CS & -\frac{6I}{L}S \\ & AS^2+\frac{12I}{L^2}C^2 & \frac{6I}{L}C & -(A-\frac{12I}{L^2})CS & -(AS^2+\frac{12I}{L^2}C^2) & \frac{6I}{L}C \\ & & 4I & \frac{6I}{L}S & -\frac{6I}{L}C & 2I \\ & & & AC^2+\frac{12I}{L^2}S^2 & (A-\frac{12I}{L^2})CS & \frac{6I}{L}S \\ & \text{Symmetry} & & & AS^2+\frac{12I}{L^2}C^2 & -\frac{6I}{L}C \\ & & & & & 4I \end{bmatrix} \tag{3.30}$$

This global stiffness matrix allows for the analysis of rigid plane frames, which are defined by the junction of axial, side, and crossbeam elements at rigid joints. In such structures, the rotations of connecting members are continuous, ensuring that moments are effectively transferred through the joints. The analysis utilizes the beam element stiffness matrix, where the element stiffness is strictly dependent on the material properties, geometric parameters, and the specific orientation of the member within the global frame.

Furthermore, the analysis can be extended to three-dimensional space by considering a beam element arbitrarily oriented in space. In this case, bending occurs about two principal axes. A local coordinate system (x', y', z') is established where the x'-axis extends from node 1 to node 2. The y'-axis corresponds to the principal axis with the minimum moment of inertia, I_y, while the z'-axis, determined by the right-hand rule, corresponds to the maximum moment of inertia, I_z.
Approximating the bending behavior in the $x' - z'$ plane due to moments $m_{y'}$, the stiffness matrix $[k'_y]$ representing the significant bending effects in this plane is given by:

$$[k'_y] = \frac{EI_y}{L^4}\begin{bmatrix} 12L & -6L^2 & -12L & -6L^2 \\ & 4L^3 & 6L^2 & 2L^3 \\ & & 12L & 6L^2 \\ \text{Symmetry} & & & 4L^3 \end{bmatrix} \tag{3.31}$$

Similarly, considering the bending behavior in the $x' - y'$ plane induced by moments m'_z about the z'-axis, the corresponding stiffness relationships are derived. Due to

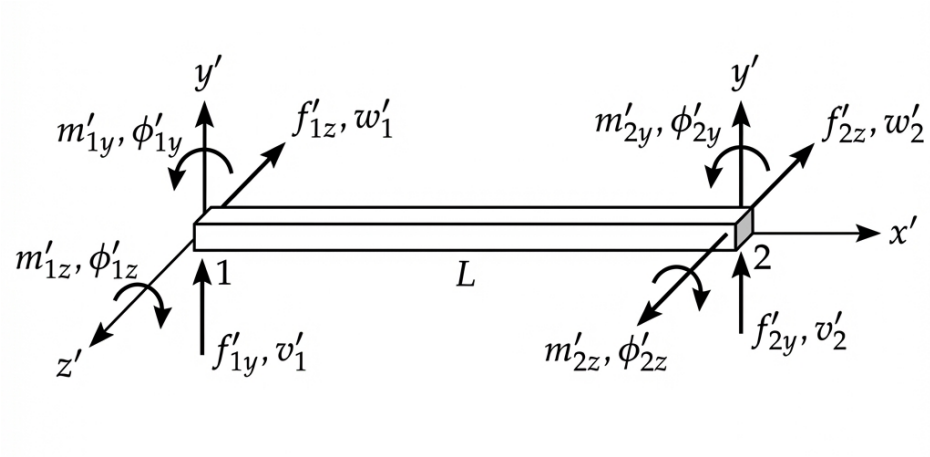


Figure 3.4: Bending about two axes [6].

the coordinate system conventions (specifically the right-hand rule), the definition of positive rotation ϕ'_z results in sign variations in the off-diagonal terms compared to the $x' - z'$ plane formulation. The stiffness matrix $[k'_z]$ capturing these effects is:

$$[k'_z] = \frac{EI_z}{L^4} \begin{bmatrix} 12L & 6L^2 & -12L & 6L^2 \\ & 4L^3 & -6L^2 & 2L^3 \\ & & 12L & -6L^2 \\ \text{Symmetry} & & & 4L^3 \end{bmatrix} \tag{3.32}$$

The comprehensive stiffness matrix for a beam element in three-dimensional space is constructed by superimposing the contributions from axial stiffness, torsional stiffness (related to the shear modulus G and torsional constant J), and bending stiffness about both principal axes. This results in a 12×12 matrix relating the six degrees of freedom at each node (three translations u', v', w' and three rotations $\phi'_x, \phi'_y, \phi'_z$):

$$[k'] = \begin{bmatrix}
\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\
0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\
-\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\
0 & 0 & -\frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L}
\end{bmatrix} \tag{3.33}$$

The transformation from the local to the global axis system is accomplished using the relationship:

$$[k] = [T]^T [k'][T] \tag{3.34}$$

where the transformation matrix $[T]$ is constructed from 3×3 submatrices $[\lambda]$ arranged along the diagonal:

$$[T] = \begin{bmatrix}
[\lambda] & 0 & 0 & 0 \\
0 & [\lambda] & 0 & 0 \\
0 & 0 & [\lambda] & 0 \\
0 & 0 & 0 & [\lambda]
\end{bmatrix} \tag{3.35}$$

The submatrix $[\lambda]$ contains the direction cosines relating the local (x', y', z') and global (x, y, z) axes:

$$[\lambda] = \begin{bmatrix}
C_{xx'} & C_{yx'} & C_{zx'} \\
C_{xy'} & C_{yy'} & C_{zy'} \\
C_{xz'} & C_{yz'} & C_{zz'}
\end{bmatrix} \tag{3.36}$$

The direction cosines for the local x'-axis are determined from the nodal coordinates:

$$l = \frac{x_2 - x_1}{L}, \quad m = \frac{y_2 - y_1}{L}, \quad n = \frac{z_2 - z_1}{L} \tag{3.37}$$

The local y'-axis is typically defined to be perpendicular to both the local x'-axis and the global z-axis. Using the cross product relationship $y' = (z \times x')/D$, where $D = \sqrt{l^2 + m^2}$, and subsequently determining the z'-axis via orthogonality ($z' = x' \times y'$), the rotation matrix $[\lambda]$ takes the explicit form:

$$[\lambda] = \begin{bmatrix} l & m & n \\ -\frac{m}{D} & \frac{l}{D} & 0 \\ -\frac{ln}{D} & -\frac{mn}{D} & D \end{bmatrix} \tag{3.38}$$

Consequently, the individual terms of the rotation matrix $[\lambda]$ correspond to the direction cosines as follows:

$$\begin{aligned} \cos\theta_{xy'} &= -\frac{m}{D}, & \cos\theta_{xz'} &= -\frac{ln}{D} \\ \cos\theta_{yy'} &= \frac{l}{D}, & \cos\theta_{yz'} &= -\frac{mn}{D} \\ \cos\theta_{zy'} &= 0, & \cos\theta_{zz'} &= D \end{aligned} \tag{3.39}$$

This matrix allows the precise transformation of displacements and forces between the element-specific local system and the global structural coordinate system.

To study the vibration of our beams, we must first understand their static behavior. The static analysis is necessary because it allows us to build the stiffness matrix $[K]$, which shows how the structure resists bending. However, vibration is a dynamic process, so the stiffness matrix alone is not enough. We also need to define the mass matrix $[M]$, which represents how the weight (mass) is distributed across the beam. In the next section, we will see how these two matrices come together to define the natural frequencies of the structure.

3.1.2. Free Vibration of Beams

The free vibration characteristics of the beam structure are governed by the equation of motion derived from the finite element formulation. The general dynamic equilibrium, which balances inertial and elastic forces, is expressed as:

$$[M]\{\ddot{x}\} + [K]\{x\} = \{F(t)\} \tag{3.40}$$

Here, $[M]$ and $[K]$ denote the global mass and stiffness matrices, respectively, which are systematically assembled from the individual element contributions (such as the lattice-based RVEs). The vector $\{x\}$ represents the nodal displacements, while $\{F(t)\}$ corresponds to the external force vector.

In the case of free vibration, where external excitation is absent ($\{F(t)\} = 0$), the governing equation simplifies to:

$$[M]\{\ddot{x}\} + [K]\{x\} = 0 \tag{3.41}$$

To seek a solution, a harmonic displacement field is assumed in the form $\{x\} = \{\bar{X}\}e^{i\omega t}$, where ω is the circular natural frequency and $\{\bar{X}\}$ is the mode shape amplitude vector. Substituting this form into the equation of motion leads to the classical eigenvalue problem:

$$([K] - \omega^2[M])\{\bar{X}\} = 0 \tag{3.42}$$

For non-trivial solutions to exist, the determinant of the coefficients must vanish:

$$|[K] - \omega^2[M]| = 0 \tag{3.43}$$

Solving this characteristic equation yields the eigenvalues (corresponding to ω^2) and the eigenvectors (defining the mode shapes). These modal parameters are fundamental for evaluating the vibration isolation capability of the structure. Furthermore, the calculated natural frequencies will be validated against results from commercial finite element software (e.g., Ansys) to confirm the accuracy of the mathematical model.

3.2. Fundamentals of Vibration Isolation

Vibration isolation is a technique used to reduce the transmission of vibrational energy from a source to a structure. Its main goal is not to stop vibration completely, but to limit how much of it reaches sensitive parts. This is critical in engineering to prevent damage, discomfort, or poor performance.

Vibration isolation is compared to vibration damping in Figure 3.5, which considers a simple case where vibration is transferred from an external vibration source to a machine. In Figure 3.5, the horizontal axis (x-axis) shows the ratio r between the frequency of an external vibration f and the natural frequency of the machine f_n. The vertical axis (y-axis) shows the displacement amplitude d of the machine with respect to the displacement amplitude of the external vibration source. C represents the damping coefficient, which can be in the form of a different component or can be from the structure itself.

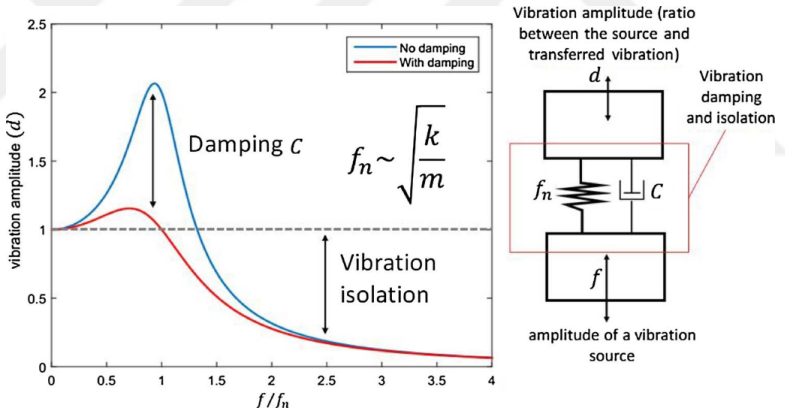


Figure 3.5: Vibration damping vs. isolation [3].

In Figure 3.5, vibration damping reduces the amplitude of the transferred displacement due to the vibration. The machine is considered to become one structure with the external vibration source when the fraction of the transferred displacement to the part is equal to unity. When the ratio between the frequency of the vibration source and the natural frequency of the part is equal to unity, the amplitude of the part displacement will be amplified, having a significantly higher value than the original source displacement; this phenomenon is called resonance.

As an alternative to vibration damping, vibration isolation can be used to reduce the vibration amplitude transferred to the machine by separating (isolating) the part from the external vibration source. Vibration isolation is more effective at reducing the effect of vibration compared to damping because it does not require high mass (weight) to suppress the machine's displacement, and it can reduce the displacement of the machine to less than that of the external vibration. The higher the ratio r, the more the part is isolated from the external vibration. To have a high value of r, f_n should be smaller than f.

A high r value can be achieved by lowering the stiffness of a structure, but there is a practical limit: if the stiffness is too low, the structure will be unable to sustain mass loads. In Figure 3.5, f_n is directly proportional to the square root of stiffness over mass ($\sqrt{k/m}$).

In this study, vibration isolation is evaluated through changes in the dynamic characteristics of lattice reinforced beam structures. By changing the internal structure, we can adjust the beam's stiffness and weight distribution. This helps improve vibration isolation without making the beam too heavy.

3.3. Dynamic Performance Metrics

Mode shapes show the shape of a structure when it vibrates at a specific natural frequency, as illustrated in Figure 3.6. They help us see how different parts move and find areas that move the most. Another important measure is the frequency ratio, which is the ratio between the outside vibration frequency and the structure's natural frequency. We use this to see how well isolation works, as higher ratios mean less vibration gets through. Sometimes, we also look at response amplitude or transmissibility, which tell us how much vibration passes through the structure compared to the starting input.

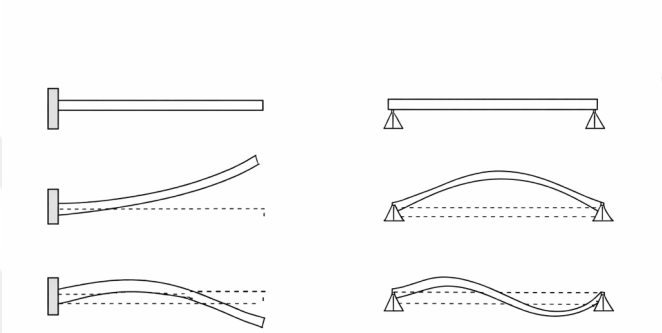


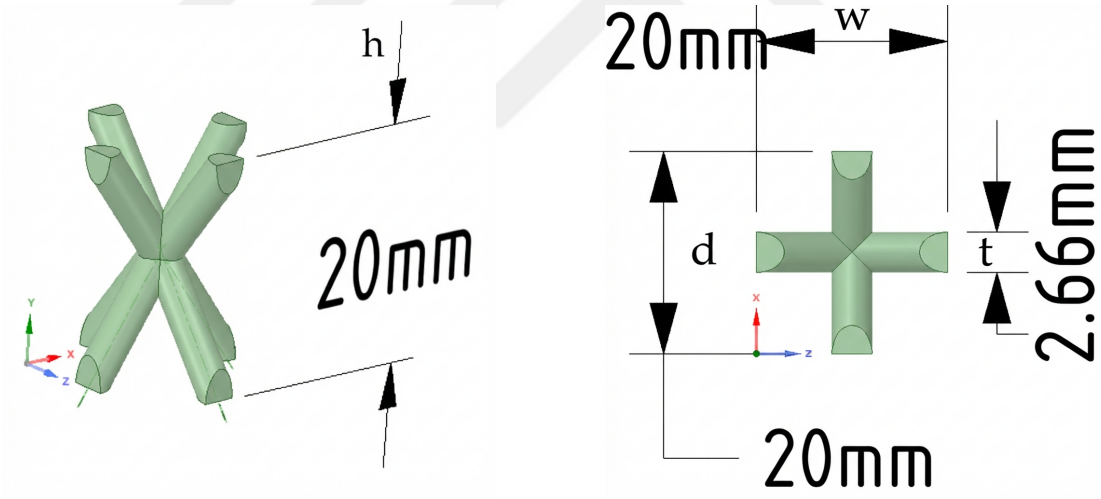
Figure 3.6: Typical mode shapes for basic beams.

In this study, we mainly check performance using natural frequencies and mode shapes found from free vibration analysis. We compare regular solid beams with lattice reinforced beams to see which isolates vibration better.

4. METHODOLOGY

4.1. Lattice Unit Cell and RVE Geometry

In this study, the internal structure of the lattice reinforced beam is created using small, repeating units called Representative Volume Elements (RVEs). An RVE is a 3D unit cell that defines the internal shape of the lattice and captures its mechanical behavior. Each unit cell is designed as a cubic volume with equal height, width, and depth (0.02 m $\times$ 0.02 m $\times$ 0.02 m), ensuring a symmetric shape that fits together perfectly. The lattice configuration is based on a strut based architecture, where the unit cell is made of thin, cylindrical struts connected at nodes arranged symmetrically to create a 3D load-bearing network. A general illustration of nodes and struts is shown in Figure 4.2, while a detailed view of a typical unit cell and its strut cross-section dimensions are provided in Figure 4.1.



(a) Lattice unit cell geometry **(b)** Strut cross-section dimensions

Figure 4.1: Unit cell and strut geometry [56].

Each strut has a circular cross-section with a diameter of 2.66 mm, kept constant for all lattice configurations to ensure a fair comparison. The detailed cross-section dimensions are shown in Figure 4.1b. The unit cells are assembled by repeating the same lattice geometry in all three directions (length, width, and height), forming a 3D grid.

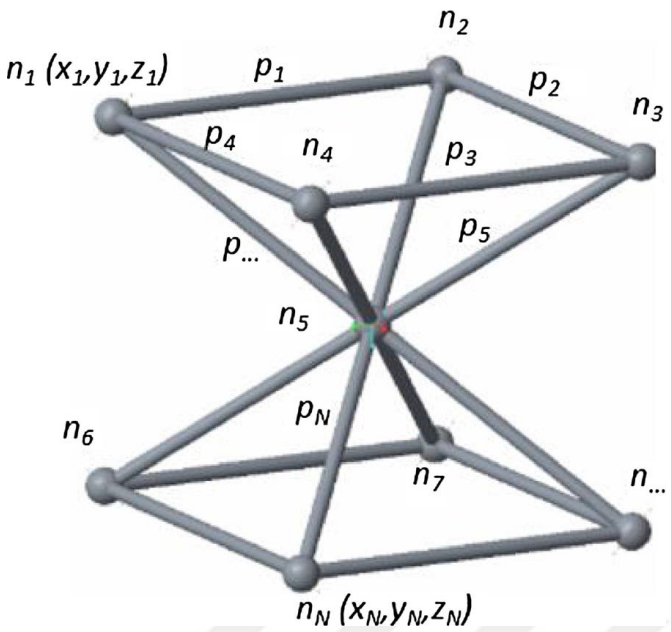


Figure 4.2: Lattice nodes and struts illustration [3].

The size of this grid is defined by the number of unit cells in each direction, denoted as N. For example, an $N = 3$ configuration means there are 3 unit cells along each edge (3x3x3). As N increases, the structure becomes denser and forms a larger Representative Volume Element (RVE). Figure 4.3 shows different lattice configurations ranging from $N = 2$ to $N = 7$.

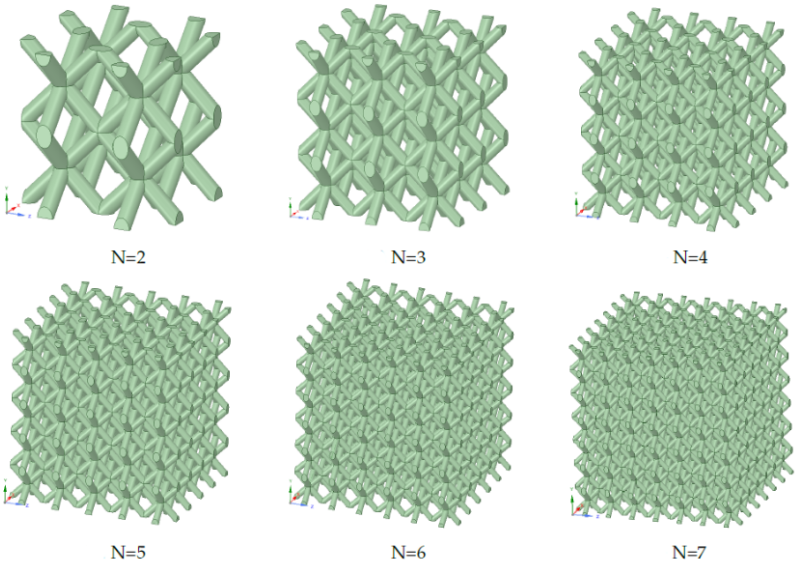


Figure 4.3: Lattice structures: NxNxN sizes [56].

In the computer model, these struts are treated as beam elements with linear elastic material behavior, which is efficient and accurate for capturing the axial and bending effects in the slender struts.

To simplify the design process, we fixed certain parameters like the number, position, and diameter of the nodes, as well as the strut diameter. This reduces the number of possible designs significantly. In this work, six different lattice models are investigated, all based on a nine-node configuration, as shown in Figure 4.4.

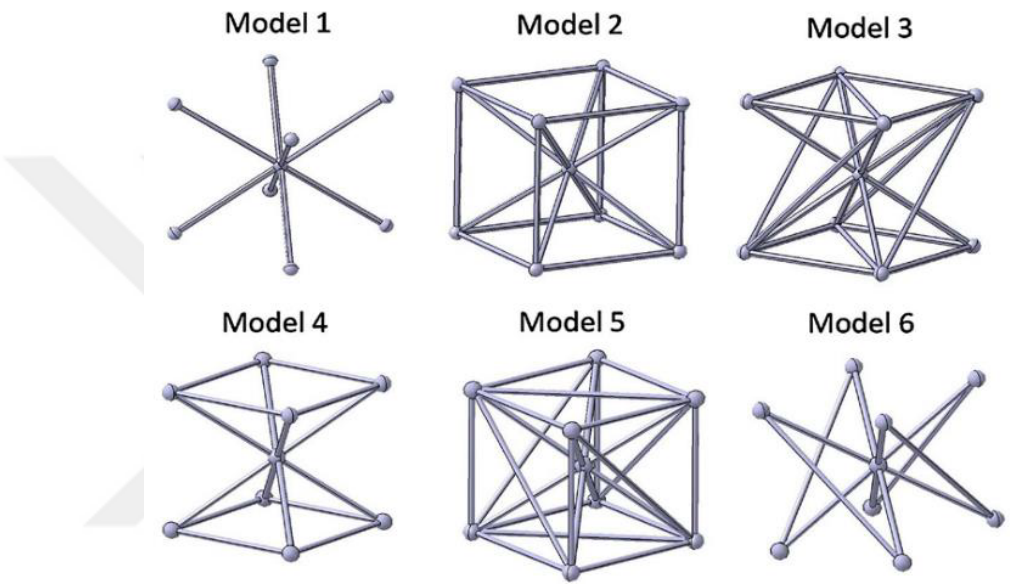


Figure 4.4: Six models of lattice unit cells [3]

Model 1 is the body-centered-cubic (BCC) structure, which is the simplest design with nine nodes [3]. The other five models are created by changing how the struts connect the nodes while keeping the structure symmetrical.

This symmetry allows the unit cells to be repeated in all three directions to form the full lattice. The main difference between the six models is the number and arrangement of the struts. The shape of each unit is defined by points and connections, which are modeled as simple beams following standard linear elastic rules. By repeating these unit cells, the RVE represents the specific mechanical behavior of that lattice shape. This method allows us to study complex lattice shapes using a faster and simpler computer model.

The RVEs are placed inside the beam to form a reinforced core. This approach makes it easy to compare different lattice designs while keeping the beam's outer dimensions the same. Using RVEs creates a clear link between the small unit cell and the overall behavior of the beam, providing the foundation for the bending and vibration tests performed in this study.

4.2. Material Properties

The material used in this study represents a polymer material commonly used in 3D-printed lattice structures. The selected material properties are not random; they are based on values frequently found in previous research on polymer lattice systems.

A material with these mechanical properties exists in real applications. The values are close to those of Nylon-12 (PA12), which is widely used in additive manufacturing, especially for lattice and cellular structures. In lattice designs, the overall stiffness is often lower than that of the solid material, and the selected Young's modulus of 741 MPa shows this behavior.

The Poisson's ratio is taken as 0.30, which is a normal value for engineering polymers. Using this value, the shear modulus of 285 MPa is calculated, ensuring that the elastic constants are consistent.

The density of 1010 kg/m³ is in the usual range for polymer materials and is used to represent mass effects in the dynamic analysis. The yield strength of 54 MPa is included to show the stress level where the material might start to permanently deform. However, this study only focuses on the elastic response, and plastic behavior is not considered.

Overall, the selected material properties provide a realistic estimate of polymer-based lattice materials. This approach allows the structural and dynamic behavior of lattice

structures to be studied without limiting the analysis to a specific brand of material.

Table 4.1 lists the mechanical properties used in the numerical models. These values provide a realistic representation of the material's behavior.

Table 4.1: Mechanical properties of the lattice material

Property	Symbol	Value	Unit
Young's modulus	E	**741**	MPa
Poisson's ratio	ν	**0.30**	–
Shear modulus	G	**285**	MPa
Density	ρ	**1010**	kg/m^3
Yield strength	σ_y	**54**	MPa

4.3. Boundary Conditions and Loading Scenarios

In this study, appropriate boundary conditions and loading scenarios are carefully chosen to simulate real-world applications of lattice reinforced beams. The goal is to clearly understand how these structures bend under weight and how they vibrate on their own. By controlling these conditions, we can make fair comparisons between different designs.

The lattice reinforced beam is treated as a long, thin structural member. We apply standard beam boundary conditions at the ends. Two main types of supports are used: clamped-clamped (C-C) and clamped-free (C-F). In the clamped-clamped (C-C) condition, both ends of the beam are fixed. They cannot move up, down, left, or right, and they cannot rotate. This represents a very rigid connection. On the other hand, in the clamped-free (C-F) condition, one end is fixed (clamped), and the other end is completely free to move. This is also known as a cantilever beam, which is very common in engineering structures like wings or balconies.

For static testing, we apply various transverse and torsional loads to evaluate the structural response. Five distinct loading scenarios are considered: tip load, spanwise dis-

tributed load, mid-load, combined spanwise-chordwise load, and torsional load. By measuring the resulting beam deflections and rotations under these conditions, we can determine the stiffness and structural performance of each lattice design. This comprehensive approach helps us identify which lattice configuration is best suited for different engineering applications.

For dynamic analysis, we do not apply a constant force. Instead, we look at ”free vibration.” This means we calculate how the beam naturally wants to shake or vibrate without any outside pushing. By solving mathematical equations (eigenvalue problems), we find the beam’s ”natural frequencies” and ”mode shapes.” These tell us the speeds at which the beam will vibrate and the shapes it will take. This is critical for vibration isolation because we generally want to avoid these natural frequencies to prevent resonance.

Beyond studying the whole beam, we also check the health of the internal lattice structure. Even if the beam looks fine from the outside, the small struts inside might be breaking. To check this ”structural integrity,” we calculate the axial stress on every single strut. As illustrated in Figure 4.5, the axial stress σ_i for any strut i is calculated as:

$$\sigma_i = \frac{F_i}{A_i} \tag{4.1}$$

Here, F_i is the axial force pushing or pulling on the strut, and A_i is the strut’s cross-sectional area. For the structure to be safe, this stress must be lower than the material’s yield strength (Y_s). If the stress is too high, the strut could bend permanently or snap.

To find these specific forces, we use a stiffness matrix method extended to 3D space. The process involves two main steps: first, solving for the global movements, and second, transforming these back to find local forces.

The relationship between the global force vector $\{F_{axial}\}$ and the global displacement vector $\{X\}$ is defined by the stiffness matrix $[K]$:

$$\{F_{axial}\} = [K]\{X\} \tag{4.2}$$

Here, $\{F_{axial}\}$ is a column matrix containing the forces in x, y, and z directions, and

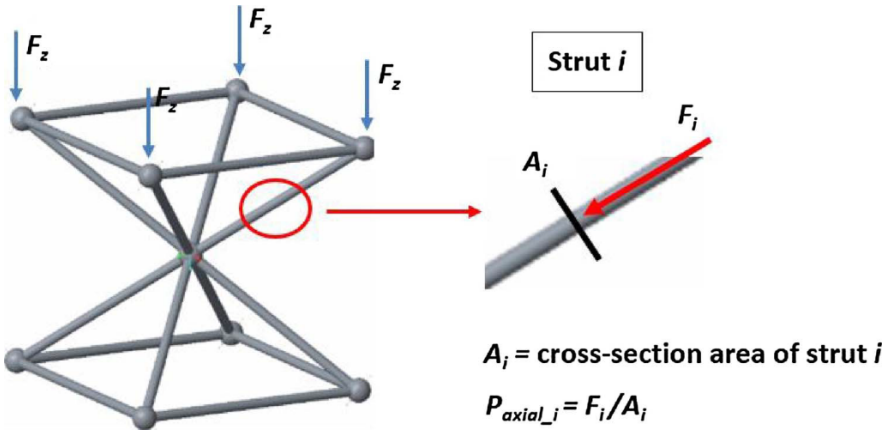


Figure 4.5: Axial forces and stresses on strut [3].

$\{X\}$ contains the displacements for each node. The size of these matrices depends on the total number of degrees of freedom (N_{dof}), which is calculated as:

$$N_{dof} = N_{node} \times 3 \tag{4.3}$$

because each node can move in three spatial directions (x, y, z).

However, each strut has its own local coordinate system (aligned with the strut's axis), while the whole structure uses a global coordinate system. To calculate the forces correctly, we must transform the stiffness definitions between these systems.

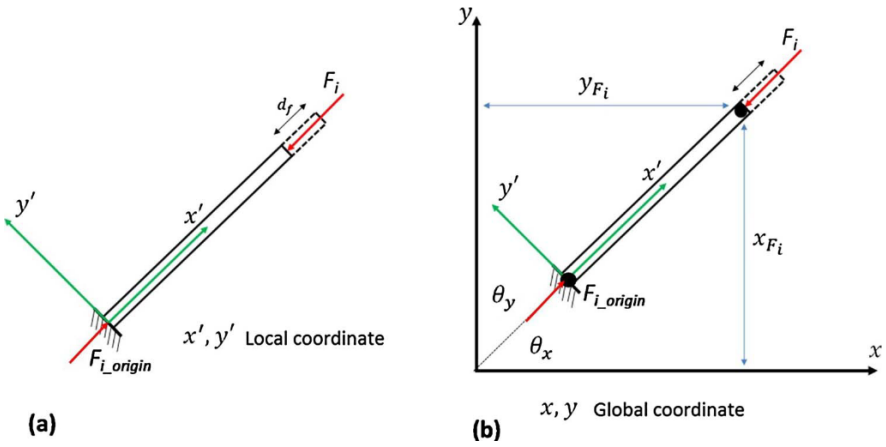


Figure 4.6: Displacement and force: Local vs. global [3].

The stiffness matrix for a single strut in its local coordinates, denoted as $[K']$, is simplified because the strut only resists axial deformation along its length. It is defined

as:

$$[K'] = \frac{A_s E}{l_s} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.4}$$

where A_s is the cross-sectional area, E is Young's modulus, and l_s is the length of the strut.

To relate this local behavior to the global system (Figure 4.6), we use a transformation matrix $[T]$. The global axial forces are then found by:

$$\{F_{axial}\} = [T]^T [K'][T]\{X'\} \tag{4.5}$$

By solving this system, we determine exactly how much every node moves ($\{X\}$), and then we use these movements to calculate the specific axial force F_i in each strut. This ensures that our design is not only stiff enough for the job but also strong enough internally to last.

4.4. Analysis Procedure

In this study, to thoroughly investigate how lattice reinforced beams perform, we developed a custom finite element analysis framework. This framework was built to handle both static bending tests and free vibration analysis, giving us a consistent way to evaluate different lattice designs. It also allows us to make fair, direct comparisons with standard solid beam models under the exact same conditions.

The process starts by defining the "Representative Volume Elements" (RVEs), which are essentially the single unit cells that act as the building blocks of our lattice. Each RVE has a unique shape and strut arrangement, which determines whether it is better at handling bending or stretching forces. We then systematically arrange these RVEs to build full beam models that have the same outer dimensions as solid beams, so the comparison is valid.

For the modeling itself, every single strut in the lattice is treated as a beam element using the Euler-Bernoulli theory. We assume the material is linear and elastic, and that deformations are small. To render this process efficient, we wrote a specific code in

MATLAB. This code automates the heavy lifting: it generates all the coordinates for the nodes, defines how elements connect, and assembles the global stiffness matrix for the entire structure. This gives us a powerful, unified tool to analyze any beam configuration we want.

In the static analysis phase, we apply transverse loads to the beam and solve the system's equilibrium equations to see how much it bends (deflection). In the dynamic analysis phase, we remove the external loads and solve the eigenvalue problem to find the system's natural frequencies and mode shapes.

We use the "fundamental natural frequency" as our main measure of vibration isolation. A lower natural frequency is better because it creates a larger gap between the structure's resonance and the high-frequency vibrations of a launch, leading to better isolation. Finally, to be sure our results are correct, we validate our custom code by comparing its outputs against known results from literature and commercial finite element software.

4.5. Model Verification Strategy

The numerical framework developed in this study is verified through comparisons with reference finite element models generated using commercial software. Thus, the study investigates the accuracy of the developed MATLAB based code in predicting the transverse-torsional modulus tensor and structural response of lattice reinforced and solid beam configurations, ensuring consistency when compared to commercial software under identical loading and boundary conditions.

Analysis methods were devised for the comparison of different material groups based on the information on the key response quantities, such as static bending deflections and natural frequencies obtained from free vibration analysis. The verification process involves confirming that the developed finite element framework accurately captures both the stiffness and inertia characteristics of the beam structures. Good agreement between the results obtained from the developed code and the reference models demonstrates the reliability of the proposed analysis procedure. Such a verification provides confidence in the use of the developed framework for the comparative evaluation of different lattice configurations, as discussed in the subsequent results chapter.

In this verification process, we specifically used ANSYS, a trusted commercial software, to serve as the benchmark. By creating the exact same beam models in ANSYS and applying the same loads and boundaries, we could directly check our MATLAB code. We looked at two main things: how much the beam bends (deflection) and how fast it vibrates (natural frequencies). A close match between our code and ANSYS proves that our tools are working correctly and ready for the main design analysis.

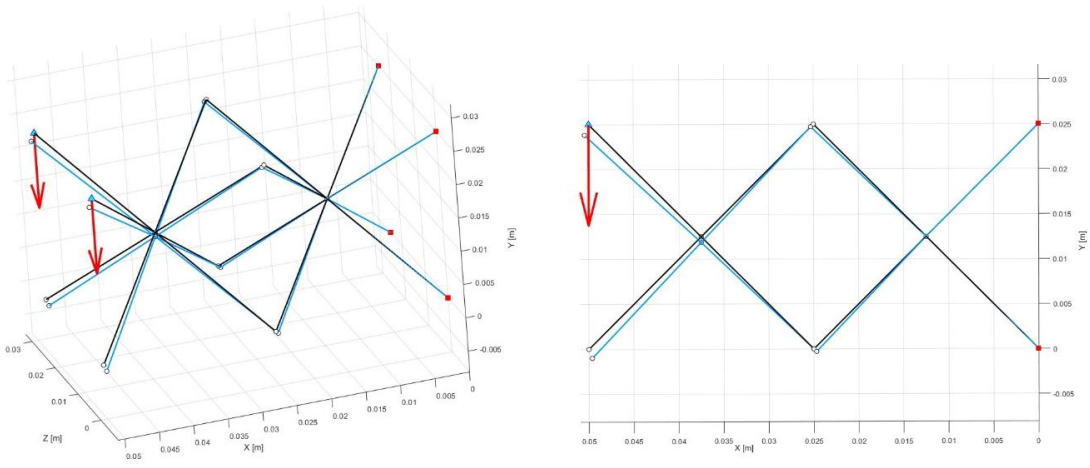
5. NUMERICAL RESULTS AND DISCUSSION

5.1. Bending Analysis Results

This section shows the results of the bending tests for our lattice beams. To compare the different designs fairly, we tested all models using the same loads and supports. We looked at how much each beam moved and how stiff it was. By keeping the outer shape the same, we can see exactly how the internal lattice shape affects the strength of the beam.

5.1.1. Verification of The Finite Element Implementation

To ensure the accuracy of the numerical framework, a systematic verification process was conducted. First, we checked our mathematical model using a simple lattice part (Model-1) with a 2x1x1 configuration. We applied a small load of 1 N to observe the baseline behavior. The results from our MATLAB code matched our expectations perfectly. The deformed shapes in 3D and projection views are illustrated in Figure 5.1(a) and (b). To be sure of the accuracy, we tested the same design in ANSYS. Figure 5.2 shows

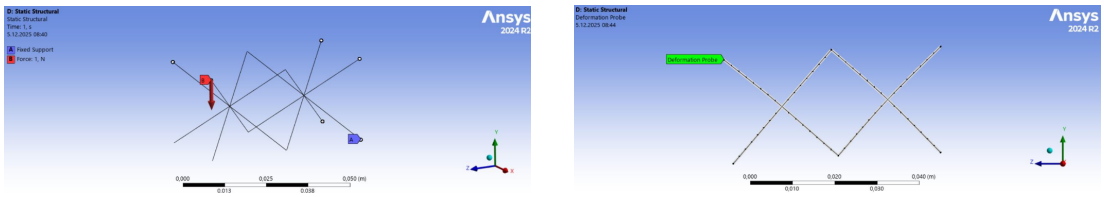


(a) MATLAB 3D view

(b) MATLAB projection view

Figure 5.1: MATLAB verification results: Model-1.

the boundary condition setup and the resulting deformation probe in the commercial software. Both programs gave almost the same results; MATLAB measured a maximum movement of 1.1774 mm, and ANSYS measured 1.2146 mm. The difference is less than 3.1%, which is well within acceptable limits.



(a) ANSYS boundary conditions **(b)** ANSYS deformation probe

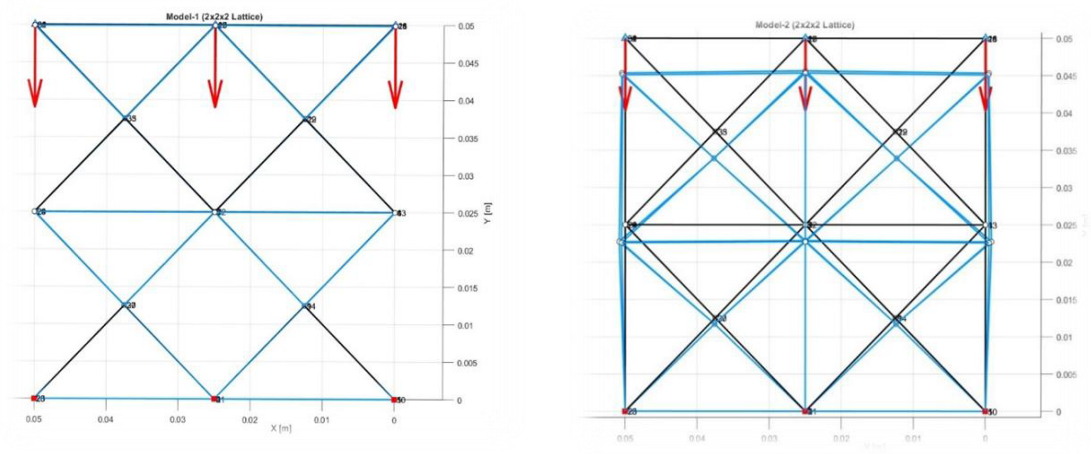
Figure 5.2: ANSYS verification: Model-1 results.

Following the initial benchmarking, we tested 'Model-1' in a larger size (2x2x2) and investigated different unit cell geometries like 'Model-2' and 'Model-4' to ensure our code works for more complex structures. These tests confirm the reliability of our numerical framework across different scales and load magnitudes. The overall verification results are summarized in Table 5.1.

Table 5.1: Verification results: MATLAB vs. ANSYS.

Model	Config.	Load (N)	MATLAB (mm)	ANSYS (mm)	Diff (%)
Model-1	2x1x1	1.0	1.1774	1.2146	3.10
Model-1	2x2x2	2.2	4.1034	4.1283	0.60
Model-2	2x2x2	19.25	0.0697	0.0698	0.14
Model-4	2x2x2	2.2	0.0580	0.0580	0.00

The deformed shape for the larger Model-1 configuration is shown in Figure 5.3a, and the result for Model-2 under a heavier load is shown in Figure 5.3b. In both cases, the results were very close to ANSYS (less than 1% difference), proving that our developed tools are accurate and ready for the main performance analysis.



(a) Model-1 (2x2x2) **(b)** Model-2 (2x2x2)

Figure 5.3: MATLAB verification: Complex lattice models.

5.1.2. Bending Performance and Structural Analysis

We studied how the lattice designs behave by applying different loads to the beams. Our main goal was to see how the inner shape of each unit cell changes the overall stiffness of the beam.

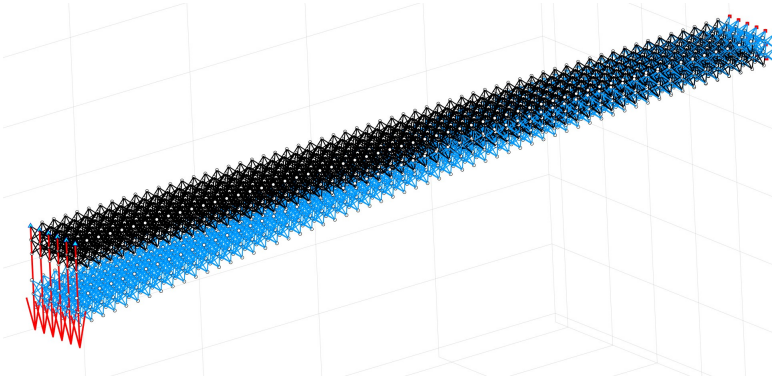


Figure 5.4: Tip load analysis

5.1.2.1. Tip Load Analysis

When we applied a load to the end of the beam, we saw a clear difference between the models. As shown in Table 5.2 and Figure 5.5, Model 5 was always the stiffest and

moved the least. This is because it has a closed frame with internal supports that help carry the load better.

The chart in Figure 5.5 shows an interesting result: Models 1 and 6 are much more flexible than the others, which is why they are shown on a separate scale on the right side of the graph. These open designs bend very easily as the beam gets longer. Model 5 stays at the bottom of the blue curves, meaning it is the best at resisting bending even when the beam is very long.

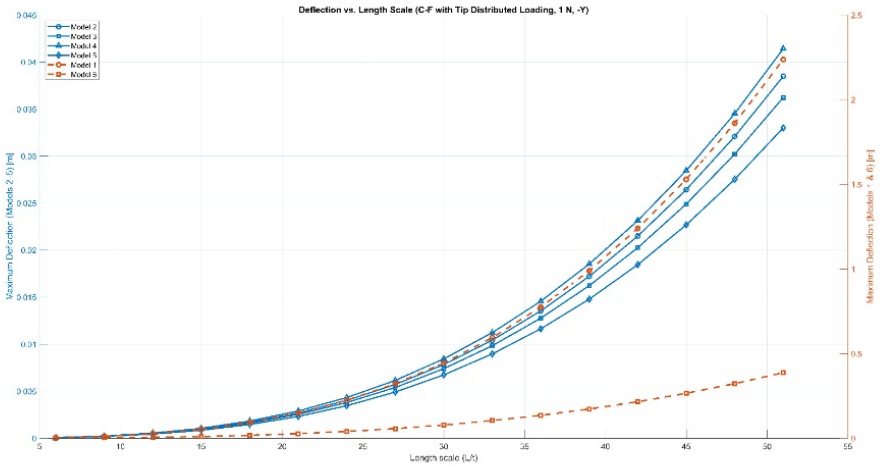


Figure 5.5: Deflection vs. Length: Tip distributed loading.

Table 5.2: Deflection vs. Length: C-F tip loading.

L [m]	Scale (L/t)	Y_max in Models [mm]					
		1	**2**	**3**	**4**	**5**	**6**
0.24	6	0.00305	0.00006	0.00006	0.00007	0.00005	0.00050
0.48	12	0.02639	0.00050	0.00048	0.00054	0.00043	0.00482
0.72	18	0.09242	0.00169	0.00160	0.00182	0.00146	0.01656
0.96	24	0.22392	0.00401	0.00379	0.00432	0.00345	0.03965
1.20	30	0.44371	0.00783	0.00739	0.00843	0.00673	0.07792
1.44	36	0.77457	0.01353	0.01276	0.01456	0.01163	0.13523
1.68	42	1.23926	0.02149	0.02025	0.02313	0.01845	0.21541
1.92	48	1.86055	0.03208	0.03020	0.03452	0.02752	0.32230
2.04	51	2.23703	0.03848	0.03622	0.04141	0.03300	0.38696

5.1.2.2. Spanwise Load Analysis

We also applied a load across the whole top surface of the beam. The 3D loading configuration for the spanwise distributed load is illustrated in Figure 5.6. The results for this spanwise load are shown in Table 5.3 and Figure 5.7. Just like in the previous test, Model 5 was the stiffest and moved the least. This graph also confirms that Model 1 and Model 6 (shown on the right scale in Figure 5.7) have much higher deflections because of their open shapes.

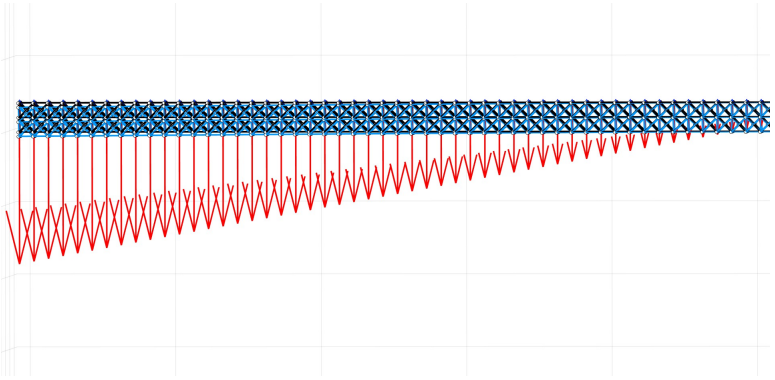


Figure 5.6: Spanwise load analysis

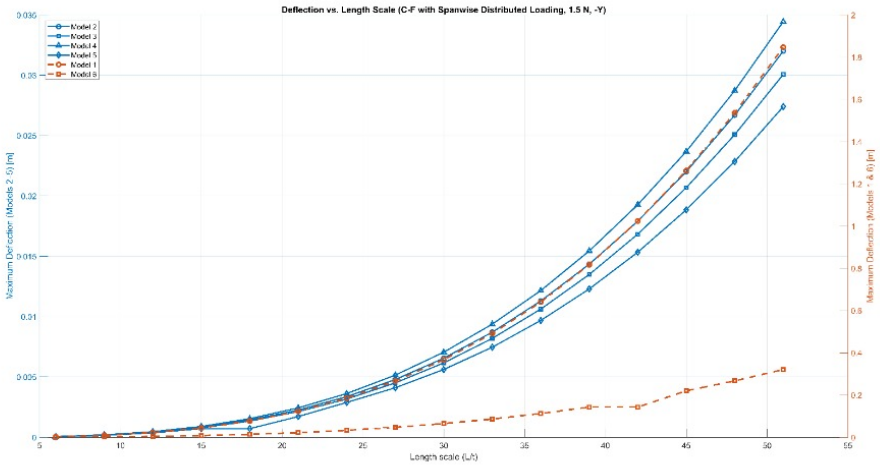


Figure 5.7: Deflection vs. Length: Spanwise loading.

Table 5.3: Deflection vs. Length: C-F spanwise loading.

L [m]	Scale (L/t)	Y_max in Models [mm]					
		1	**2**	**3**	**4**	**5**	**6**
0.24	6	0.00263	0.00006	0.00005	0.00006	0.00005	0.00049
0.48	12	0.02215	0.00043	0.00041	0.00046	0.00037	0.00405
0.72	18	0.07688	0.00143	0.00135	0.00154	0.00072	0.01384
0.96	24	0.18559	0.00337	0.00317	0.00362	0.00289	0.03304
1.20	30	0.36708	0.00655	0.00617	0.00705	0.00562	0.06481
1.44	36	0.64016	0.01130	0.01062	0.01216	0.00968	0.11232
1.68	42	1.02360	0.01791	0.01683	0.01927	0.01534	0.14297
2.04	51	1.84671	0.03200	0.03006	0.03444	0.02740	0.32076

5.1.2.3. Mid-Load Analysis

The middle-load test, shown in Figure 5.8, Table 5.4 and Figure 5.9, shows that holding the beam at both ends makes it much stiffer. Even with a heavy 20 N force, the movement was small for the strongest designs. The ranking stayed the same: Model 5 and Model 3 were the best, while the open designs moved the most.

This means that the internal supports in Model 5 provide reliable strength no matter where you push. Figures 5.9 and 5.7 both show how the unit cell geometry dictates the beam's performance across different loading conditions.

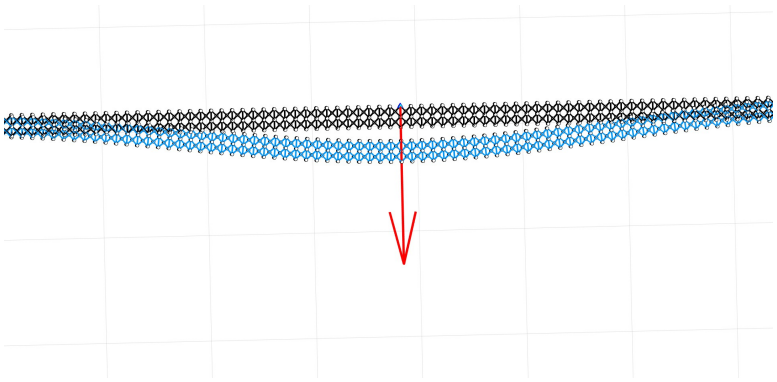


Figure 5.8: Mid-load analysis configuration

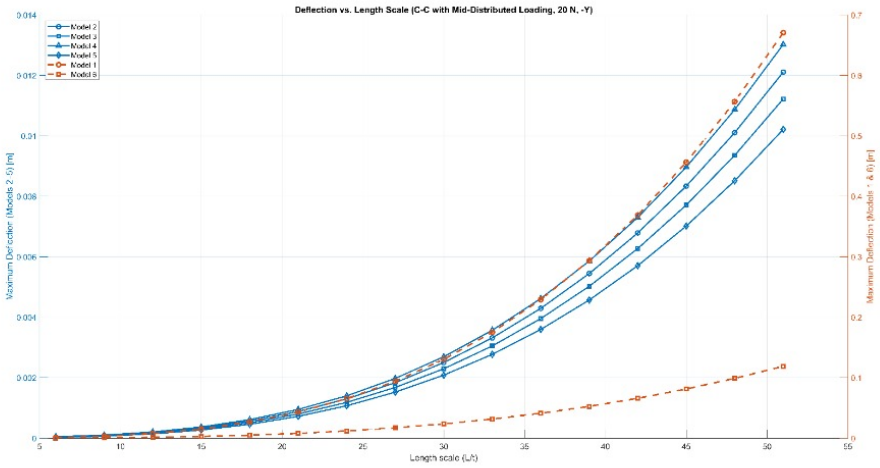


Figure 5.9: Deflection vs. Length: Mid-distributed loading.

Table 5.4: Deflection vs. Length: C-C mid-distributed loading.

L [m]	Scale (L/t)	Y_max in Models [mm]					
		1	**2**	**3**	**4**	**5**	**6**
0.24	6	0.00103	0.00003	0.00003	0.00005	0.00024	0.00018
0.48	12	0.00769	0.00018	0.00016	0.00021	0.00014	0.00140
0.72	18	0.02684	0.00057	0.00051	0.00061	0.00046	0.00494
0.96	24	0.06538	0.00130	0.00119	0.00140	0.00107	0.01196
1.20	30	0.13044	0.00250	0.00230	0.00269	0.00209	0.02367
1.44	36	0.22925	0.00429	0.00396	0.00462	0.00360	0.04127
1.68	42	0.36891	0.00679	0.00627	0.00730	0.00571	0.06597
2.04	51	0.67069	0.01211	0.01122	0.01303	0.01022	0.11894

5.1.2.4. Combined Load Analysis

Complex loading scenarios, such as combined bending and twisting, were investigated to evaluate structural stability under multi-directional forces. The loading configuration for combined spanwise and chordwise loading is illustrated in Figure 5.10. In this combined spanwise and chordwise loading test, Model 5 performed the best, showing minimal deflection. Its fully supported shape handles these multi-directional forces effectively. Conversely, the open designs like Model 1 and Model 6 (shown on the right scale in Figure 5.11) struggled with these complex loads and deflected significantly more. This proves that designs with closed structural paths are necessary for applications where the beam has to carry weight from different directions simultaneously.

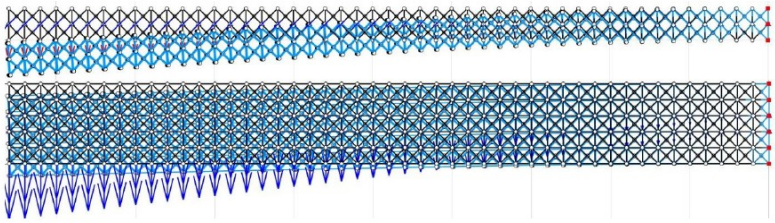


Figure 5.10: Combined spanwise and chordwise loading configuration

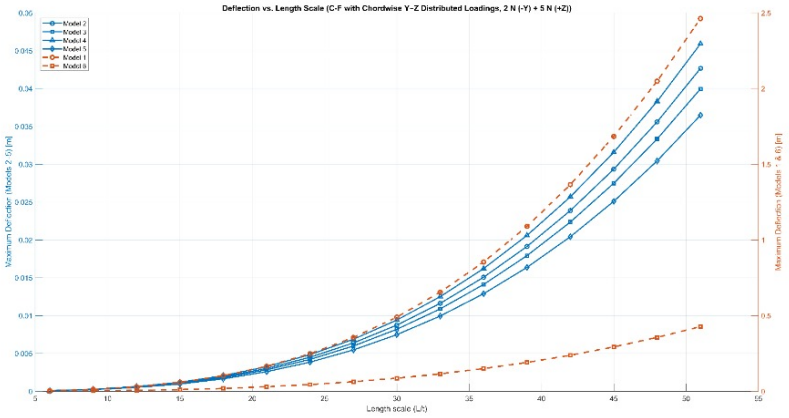


Figure 5.11: Deflection vs. Length: Combined Y-Z loading.

The maximum deflection values for the combined loading cases are explicitly presented in Table 5.5. These quantitative findings support the trends observed in the preceding plots regarding the superior stiffness of Model 5.

Table 5.5: Deflection vs. Length: C-F combined loading.

L [m]	Scale (L/t)	Y_max in Models [mm]					
		1	**2**	**3**	**4**	**5**	**6**
0.36	9	0.01218	0.00025	0.00023	0.00027	0.00021	0.00223
0.72	18	0.10246	0.00191	0.00179	0.00205	0.00163	0.01835
1.08	27	0.35469	0.00638	0.00597	0.00687	0.00546	0.06264
1.44	36	0.85346	0.01506	0.01410	0.01620	0.01288	0.14934
1.80	45	1.68333	0.02934	0.02748	0.03158	0.02509	0.29273
2.04	51	2.62241	0.04267	0.03996	0.04591	0.03648	0.42685

5.1.2.5. Torsional Load Analysis

The torsional loading effect, with the configuration illustrated in Figure 5.12, was also analyzed to understand how the lattice configurations resist twisting moments. The results, shown in Table 5.6 and Figure 5.13, indicate significant variations in torsional stiffness across different models.

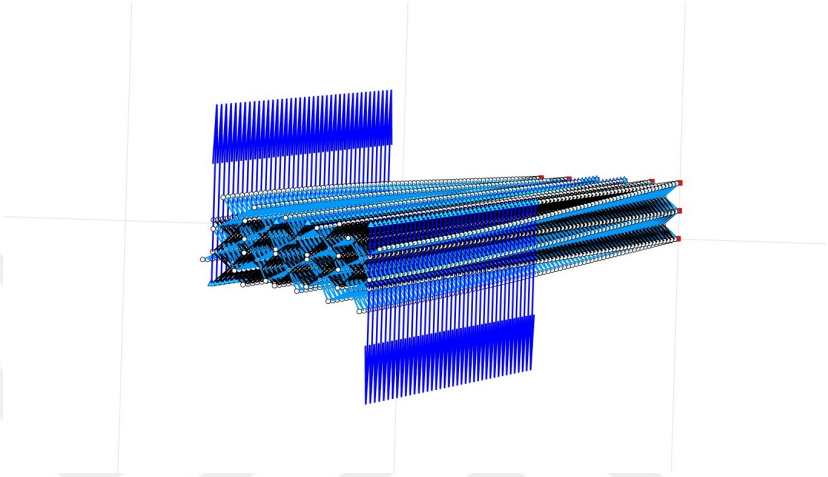


Figure 5.12: Torsional loading configuration

Table 5.6: Rotation vs. Length: C-F torsional loading.

L [m]	Scale (L/t)	Maximum Rotation Theta_X in Models [Degree]					
		1	**2**	**3**	**4**	**5**	**6**
0.24	6	16.214	2.204	1.395	2.939	1.158	6.234
0.48	12	12.420	4.638	2.632	5.052	2.363	6.851
0.72	18	12.626	7.092	3.925	7.393	3.585	8.643
0.96	24	13.955	9.551	5.234	9.797	4.813	10.764
1.20	30	15.756	12.012	6.549	12.228	6.043	13.022
1.44	36	17.798	14.474	7.867	14.673	7.273	15.351
1.68	42	19.982	16.937	9.187	17.126	8.505	17.722
1.92	48	22.255	19.399	10.509	19.583	9.737	20.118
2.04	51	23.415	20.631	11.170	20.813	10.353	21.323

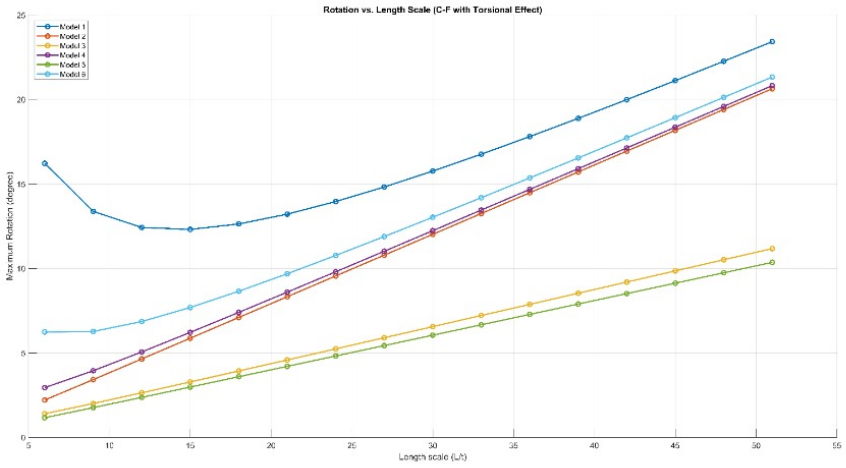


Figure 5.13: Rotation vs. Length: Torsional loading.

5.1.2.6. Mass Evaluation of Lattice Topologies

Comparing the total weight of the lattice models is as important as measuring their stiffness. Since each unit cell has a different internal pattern, the amount of material used changes from one design to another. As shown in Table 5.7 and Figure 5.14, there is a large difference in weight. Model-5 is the heaviest beam because it contains many dense internal support struts. In contrast, Model-1 is the lightest because it has a very open and simple design with much less material.

In modern engineering, especially for aerospace or automotive parts, being strong is not enough; the structure must also be as light as possible. A beam that is very stiff but extremely heavy may not be the best choice because it increases the overall weight of the vehicle. To find the most ”efficient” design, we calculated a performance index. We did this by giving points to each model (from 1 to 6) based on how well they performed in all five loading tests. We then divided these total points by the mass of the beam.

The results show a very important trade-off between strength and weight. Although Model-5 is the strongest and stiffest in almost every test, its high mass reduces its overall efficiency. Interestingly, Model-4 achieved the highest performance index (11.990). This means that Model-4 provides the best ”strength-to-weight” ratio among all configurations. It is light, yet it remains stiff enough to handle various loads effectively. This comparison proves that the best lattice topology is not always the strongest one;

instead, the selection depends on whether the goal is maximum stiffness or maximum weight savings.

Table 5.7: Total mass and performance of topologies.

Model	Tip	Mid	Span	Combined	Torsion	Total Pts	Mass [kg]	Index (Pts/Mass)
Model-1	1	1	1	1	1	5	0.793	6.305
Model-2	4	4	4	2	4	18	2.168	8.302
Model-3	5	5	5	5	5	25	2.682	9.321
Model-4	3	3	3	3	3	15	1.251	11.990
Model-5	6	6	6	6	6	30	3.140	9.554
Model-6	2	2	2	2	2	10	1.441	6.939

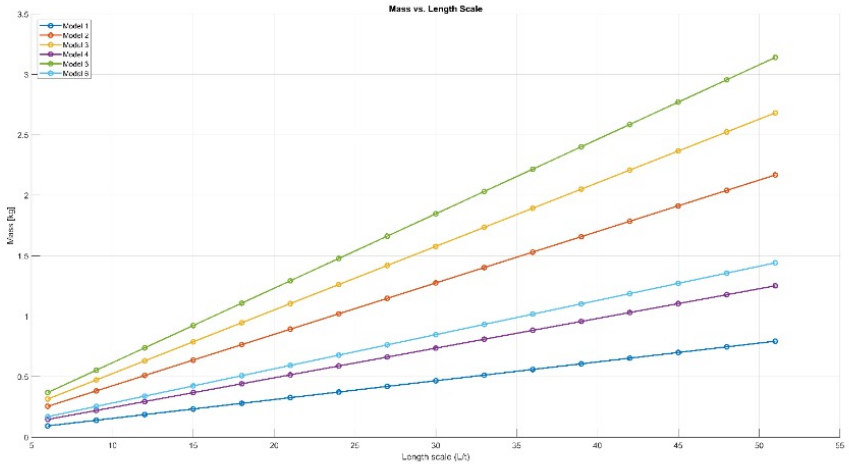


Figure 5.14: Mass performance: Lattice beams.

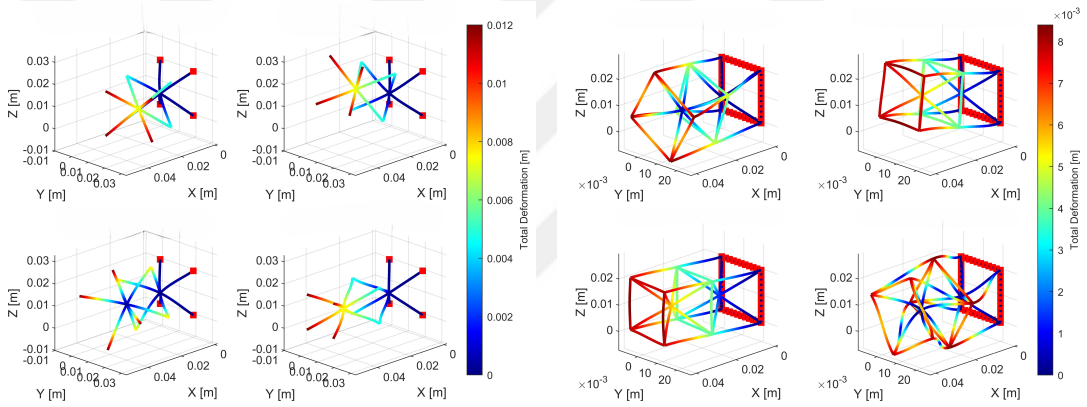
5.2. Free Vibration Analysis Results

This section presents the results of the free vibration analysis for the lattice reinforced beams. The movement of each beam is evaluated by looking at its natural frequencies and mode shapes. To ensure a fair comparison, all models are tested using the same boundary conditions and material properties.

5.2.1. Verification of the Vibration Analysis

To make sure our vibration analysis is correct, we first conducted a verification study. We tested small lattice models with a 2x1x1 setup and compared the results from our MATLAB code with ANSYS, a well-known engineering program. To get more accurate results, we divided every single strut into 10 smaller segments (sub-elements). This helps the computer better understand how the struts bend and twist during vibration.

The first few mode shapes for different models are shown in Figure 5.15. In the lower modes, the whole beam mostly bends as one unit, while in higher modes, the individual struts inside the lattice start to flex. The colors in the pictures show how much each part moves, where blue means it is still and red shows the highest movement.



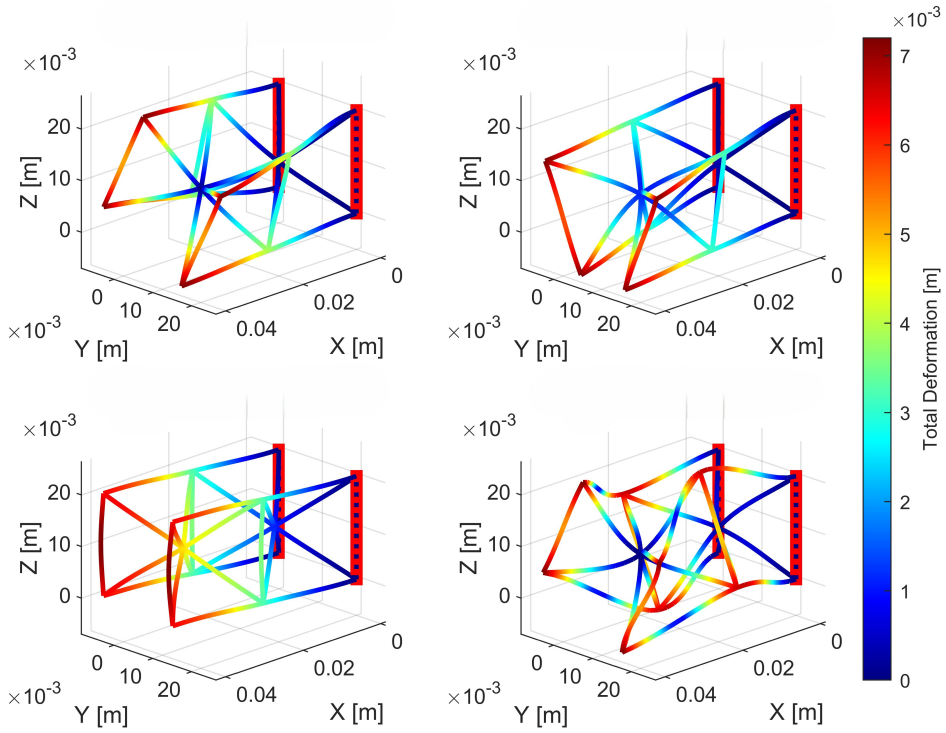
(a) Model-1 Mode Shapes **(b)** Model-2 Mode Shapes

Figure 5.15: Lattice mode shapes: Models 1-2.

For Model-1 (2x1x1), the first four natural frequencies from MATLAB match the ANSYS results very well, as shown in Table 5.8. For the first two patterns, the difference is very small. Even for the more complex higher modes, the difference stays below 8.2%, which is a good result for these types of lattice structures.

We saw even better results for Model-2 and Model-4, with most differences being less than 1% or 3%. This close match proves that our calculation methods and the material settings are correct.

The first few natural frequencies are studied because these modes are important for the



(a) Model-4 Mode Shapes

Figure 5.16: Lattice mode shapes: Model-4.

movement of beam structures. As expected, stiffer lattice designs have higher natural frequencies. This means that by changing the shape of the internal lattice, we can 'tune' how the beam vibrates.

5.2.2. Vibration Characteristics Under Clamped-Free (C-F) Conditions

This section presents the free vibration results of the lattice reinforced beam models under cantilever-free (C-F) boundary conditions. In this configuration, one end of the beam is completely fixed, while the other end is left free to vibrate. We evaluate the first four natural frequencies and their corresponding mode shapes, as these are the most important for understanding the dynamic behavior of such structures.

The overall results for all six lattice configurations are summarized in Table 5.9. A clear difference in natural frequencies is observed among the different designs, even though their external dimensions are identical.

Table 5.8: Free Vibration Verification: Results comparison.

Case	Mode	MATLAB [Hz]	ANSYS [Hz]	Diff [%]
Model-1 (2x1x1)	1	212.98	219.04	2.77
	2	220.25	219.04	0.55
	3	430.82	445.34	3.26
	4	668.32	727.64	8.15
Model-2 (2x1x1)	1	432.95	430.62	0.54
	2	1260.5	1267.1	0.52
	3	1277.0	1280.5	0.27
	4	1293.5	1280.5	1.01
Model-4 (2x1x1)	1	448.34	449.19	0.19
	2	706.97	727.30	2.80
	3	1334.8	1367.8	2.41
	4	1337.2	1375.0	2.75

Table 5.9: Natural Frequencies: Lattice Beams (C-F).

Mode	Model-1	Model-2	Model-3	Model-4	Model-5	Model-6
1	2.010	11.602	10.388	11.886	9.774	3.523
2	3.866	20.364	18.047	20.557	17.510	7.541
3	12.552	71.044	63.735	72.876	60.143	22.007
4	23.896	97.242	108.910	104.070	103.26	46.030

For all models, the first vibration mode is dominated by the global bending of the beam, as shown in Figure 5.17. This mode exhibits the largest deformation near the free end, while the fixed end remains steady. This is a classic cantilever response and confirms that our lattice reinforced beams follow standard vibration rules at low frequencies.

The second mode also represents a bending-dominated response, but with more curvature along the beam. As the mode number increases, the deformation patterns become

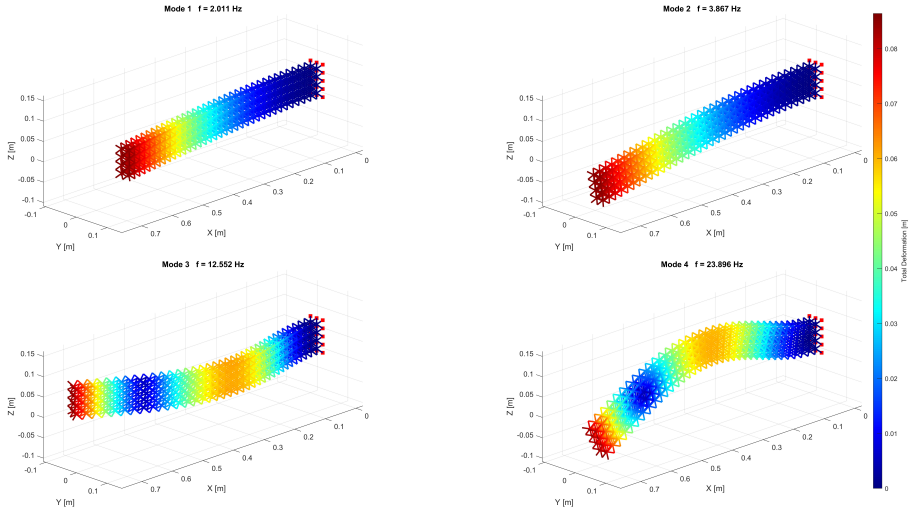


Figure 5.17: C-F mode shapes: Model-1.

more complex. The third and fourth modes, illustrated in Figure 5.19, show higher-order bending with increased curvature and more localized movement, especially near the fixed region.

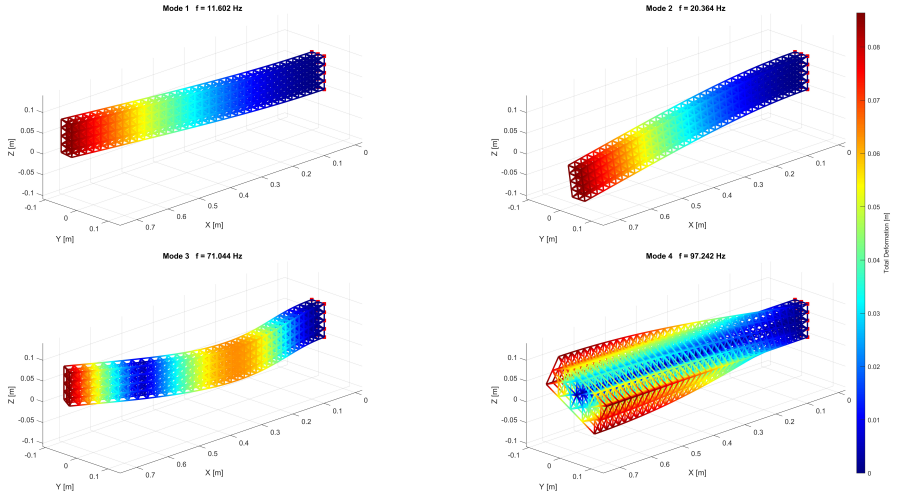
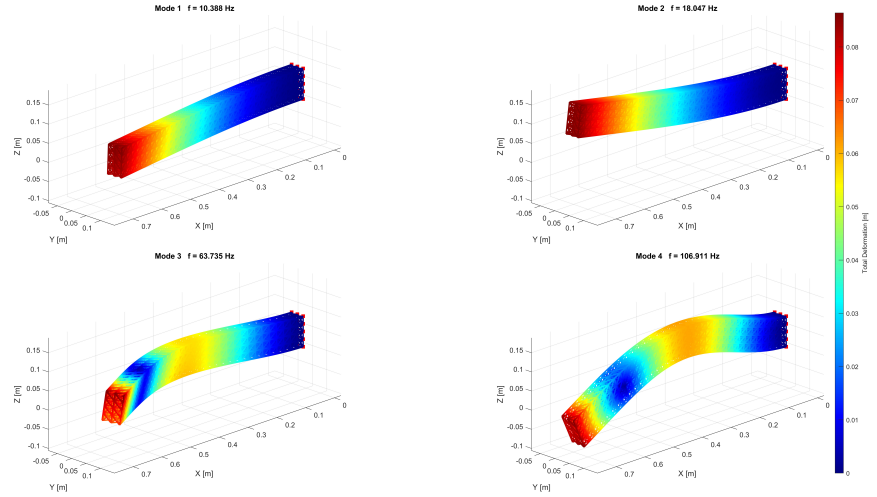
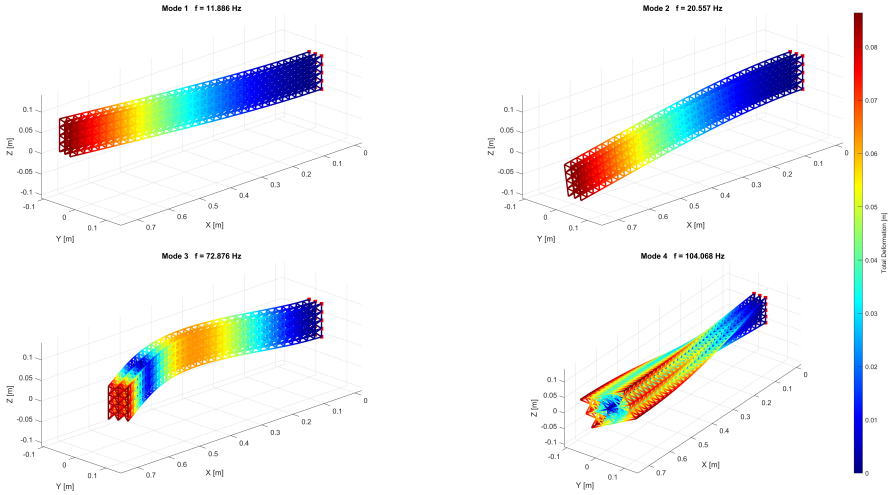


Figure 5.18: C-F mode shapes: Model-2.

The differences in natural frequencies highlight the impact of lattice topology. Model-1 and Model-6 exhibit the lowest frequencies, indicating they are more flexible. In contrast, Models 2, 3, 4, and 5 show much higher frequencies, which implies a higher global stiffness. Since all beams have the same material and outside size, these differences come solely from the internal lattice shape.



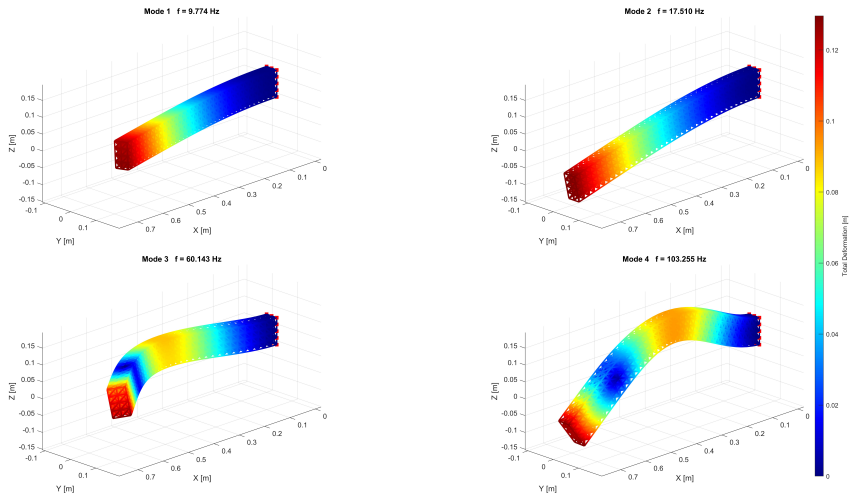
(a) Model-3 Mode Shapes (C-F)



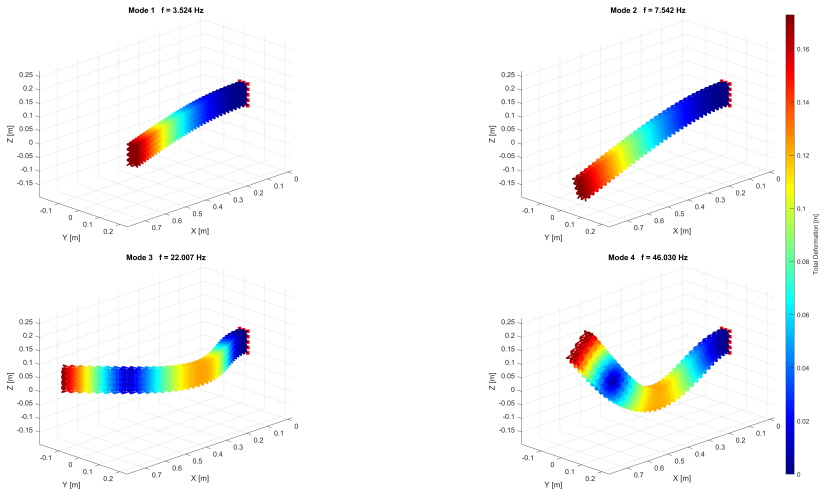
(b) Model-4 Mode Shapes (C-F)

Figure 5.19: C-F vibration mode shapes: Models 3-4.

The mode shapes also reflect the influence of the lattice architecture. Models with lower frequencies tend to show smoother and more uniform deformation. Stiffer models, however, show stronger curvature near the fixed end, particularly in higher modes. This behavior shows how internal load paths and stiffness distribution are changed by our different lattice designs, as illustrated in Figure 5.20 for Model-5 and Model-6.



(a) Model-5 Mode Shapes (C-F)



(b) Model-6 Mode Shapes (C-F)

Figure 5.20: C-F vibration mode shapes: Models 5-6.

Overall, the results prove that lattice topology is critical for controlling the dynamic response of beams. By modifying the internal lattice, it is possible to significantly shift

natural frequencies and change mode shapes without altering the overall dimensions of the beam.

5.2.3. Vibration Characteristics Under Clamped-Clamped (C-C) Conditions

In this section, we study how the lattice reinforced beams vibrate when both ends are tightly fixed (Clamped-Clamped boundary conditions). This setup is very common in many structural applications, like bridge supports or machine frames, where a beam is held firmly at both ends.

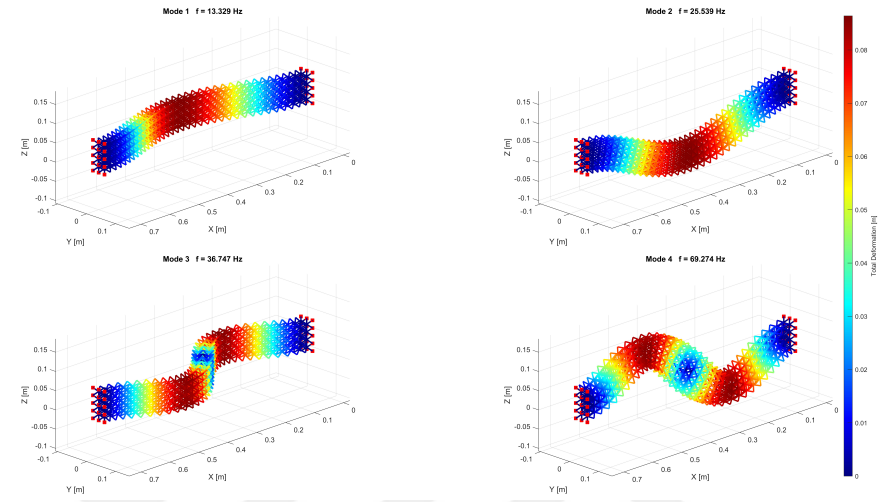
Since both ends are constrained, the beams become much stiffer compared to the previous cantilever setup. This change leads to higher natural frequencies and different vibration patterns. For instance, in the first mode, the maximum movement happens in the middle of the beam, while both ends stay perfectly still. By testing our models in this double-fixed configuration, we can understand how the internal lattice shape helps the beam resist vibration when it is fully supported from both sides.

The first four natural frequencies obtained for the C-C boundary conditions are summarized in Table 5.10. As expected, these values are significantly higher than the C-F results due to the increased structural constraint. Similar to the cantilever case, the lowest natural frequencies are observed in Model-1 (13.33 Hz) and Model-6 (22.96 Hz), confirming their higher compliance. In contrast, Model-4 (73.90 Hz), Model-2 (72.18 Hz), and Model-3 (65.34 Hz) show a much stiffer dynamic response, resisting vibration more effectively in this double-fixed setup.

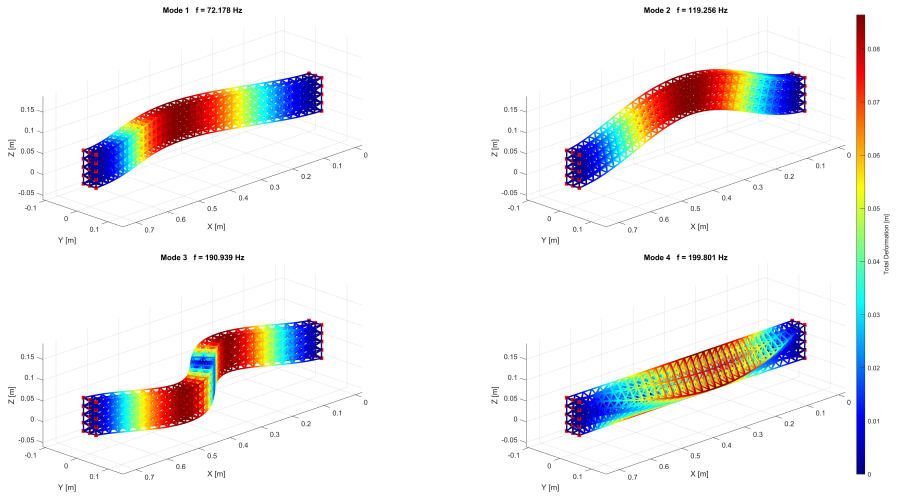
Table 5.10: Natural Frequencies: Lattice Beams (C-C).

Mode	Model-1	Model-2	Model-3	Model-4	Model-5	Model-6
1	13.329	72.178	65.340	73.895	61.690	22.960
2	25.539	119.260	108.610	121.020	104.150	48.193
3	36.747	190.940	173.770	195.980	164.960	63.159
4	69.274	199.800	237.880	213.400	226.140	123.430

The first vibration mode for all models is dominated by global bending. Since both ends are clamped, the maximum displacement occurs at the center of the beam. As shown in Figure 5.21, Model-1 and Model-2 exhibit very clear, symmetric bending shapes. While Model-1 shows more curvature due to its flexible nature, Model-2 maintains a more robust form even in higher modes.



(a) Model-1 Mode Shapes (C-C)

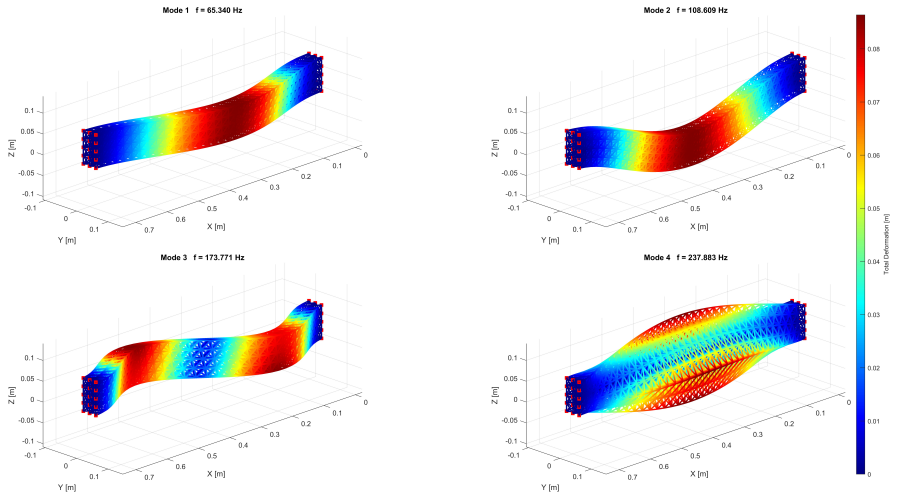


(b) Model-2 Mode Shapes (C-C)

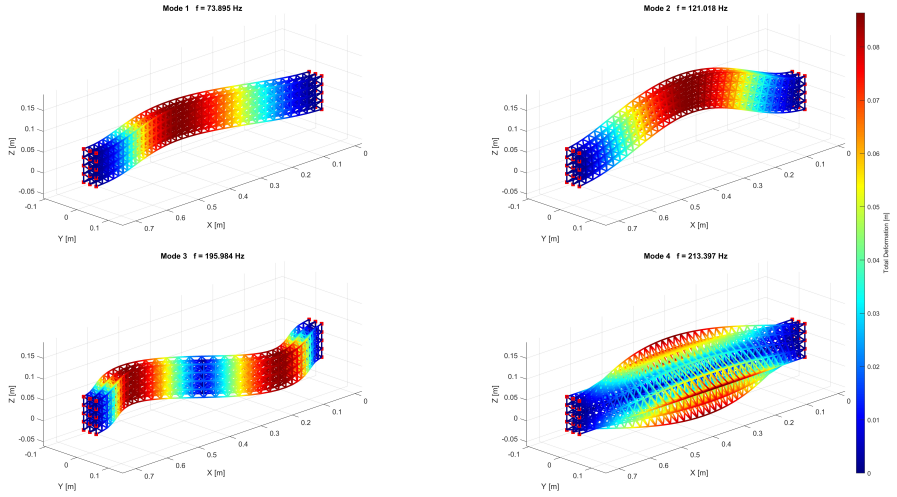
Figure 5.21: C-C mode shapes for Model-1 and Model-2

As we move to the third and fourth vibration modes, the deformation patterns become more complex and localized. For Model-3 and Model-4 (illustrated in Figure 5.22), we

can see that the internal lattice starts to influence how the energy is distributed along the beam. Model-4, in particular, shows great stability in these higher-order modes, which is consistent with its high performance in the static bending tests.



(a) Model-3 Mode Shapes (C-C)

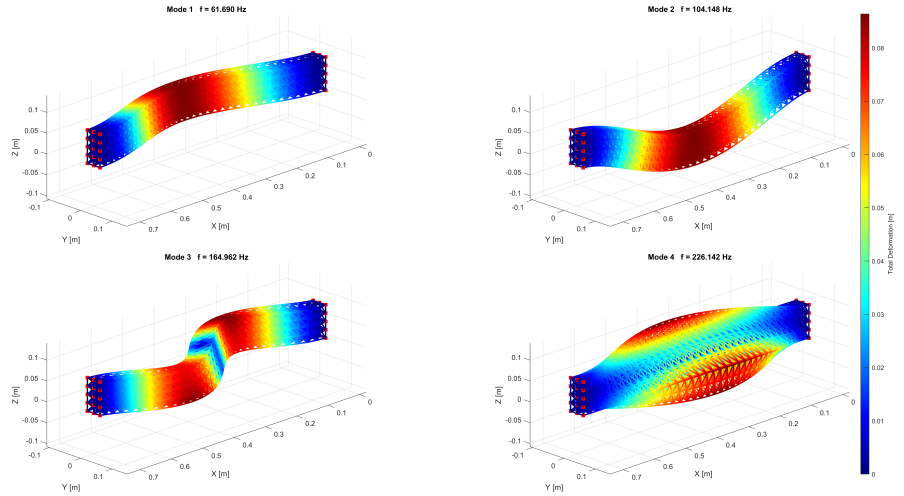


(b) Model-4 Mode Shapes (C-C)

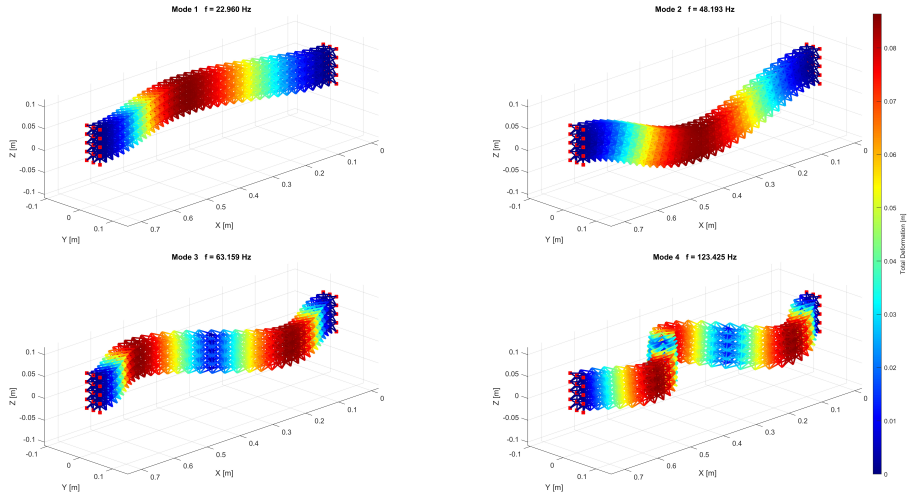
Figure 5.22: C-C mode shapes for Model-3 and Model-4

Finally, the mode shapes for Model-5 and Model-6 are presented in Figure 5.23. Model-5 is the stiffest design, and this is visible in its vibration patterns where the curvature is more resisted compared to the open-frame Model-6. The high-frequency modes show localized ”wavy” patterns, especially in Model-6, which suggests that open lattice

designs might be more prone to complex local vibrations at higher frequencies.



(a) Model-5 Mode Shapes (C-C)



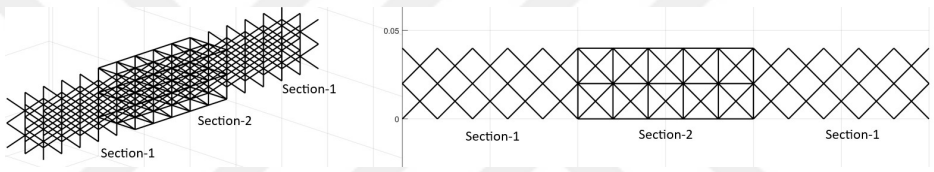
(b) Model-6 Mode Shapes (C-C)

Figure 5.23: C-C mode shapes for Model-5 and Model-6

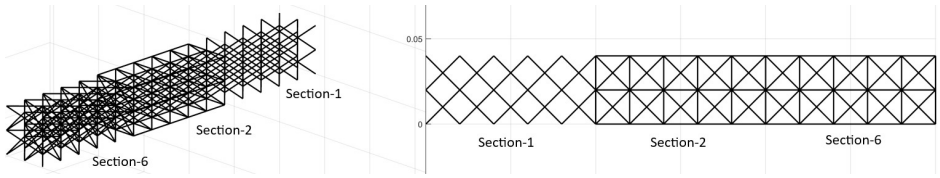
These results clearly show that the internal lattice geometry is a powerful tool for controlling how a beam vibrates, even when it is firmly clamped at both ends. These findings provide an important basis for evaluating the vibration isolation potential of different lattice designs in the following sections.

5.3. Hybrid Lattice Beams Analysis

In addition to uniform designs, this study investigates hybrid lattice configurations by combining different types of unit cells along the length of a beam to tailor the structural response. By merging different topologies, the advantages of various lattice types, such as stiffness, lightness, or better handling of bending and torsion, can be combined effectively. In this study, the hybrid lattice structure was organized in a systematic and modular way, as illustrated by the schematics in Figure 5.24. The beam was divided into three longitudinal sections along the global X-direction. Each section was assigned a different lattice topology selected from six predefined unit-cell models. Two representative examples of these configurations, (1,2,1) and (6,2,1), are shown in Figure 5.24a and Figure 5.24b, respectively.



(a) Schematic for (1,2,1) configuration



(b) Schematic for (6,2,1) configuration

Figure 5.24: Hybrid lattice beams with different sections

The total beam length was kept the same for all tests. Only the lattice type was changed between sections, while the cell size and materials were not changed. This way, we could study only the effect of the lattice shape. To make the structure realistic, we did not allow two sections next to each other to have the same lattice type. Following this rule, if the first section is chosen from six models, the second must be one of the other five. The third section can then be any of the six models, as long as it is different from

the second. This creates an alternating $6 \times 5 \times 6 \times 5$ pattern. Based on this logic, we found 180 different hybrid combinations for a three-section beam. Table 5.11 shows how the number of design possibilities grows very quickly as more sections are added.

Table 5.11: Possible hybrid design configurations

No. of Sections	Calculation Method	Possible Design Configurations
2	6×5	30
3	$6 \times 5 \times 6$	180
4	$6 \times 5 \times 6 \times 5$	900
5	$6 \times 5 \times 6 \times 5 \times 6$	5,400
6	$(6 \times 5)^3$	27,000
7	$(6 \times 5)^3 \times 6$	162,000
8	$(6 \times 5)^4$	810,000

As the table shows, adding even a few sections creates a huge number of design options, reaching over 800,000 for an eight-section beam. This variety means that hybrid lattices give designers a lot of freedom to improve mechanical performance. By simply changing the sequence, the first natural frequency can be adjusted over a very wide range.

Besides using different lattice shapes, using different materials in each section is another way to improve the design. If each section used materials with different stiffness or density, the number of possible designs would grow even more. This would allow for very specific control over how the beam reacts to forces and vibrations.

The lattice distribution was applied only in the X-direction, while in the Y and Z directions, the lattice pattern remained uniform. This strategy preserves a beam-like global behavior while enabling local structural variation. This organization provides a clear and controlled framework for comparing uniform and hybrid lattice beams, and it also enables future extension to optimization-based or functionally graded lattice designs.

5.3.1. Static Stress Analysis of Hybrid Lattice Beams

We performed numerical stress analyses using cantilever boundary conditions to evaluate the static structural integrity. One end of the lattice beam was fully fixed, and the other end was free. This clamped–free (C–F) setup constrained all movements and rotations at the base. To analyze the stress distribution, we applied a static load in the vertical direction. We used a triangular load distribution to make the test more realistic. The load started at zero at the base and increased linearly toward the tip. We applied the total force to specific nodes along a single line of the lattice. This approach clearly shows the loading effect and allows us to see the force arrows in the plots. This setup is ideal for studying bending and stress concentration in hybrid designs. Figure 5.25 shows the static stress distribution for representative hybrid configurations.

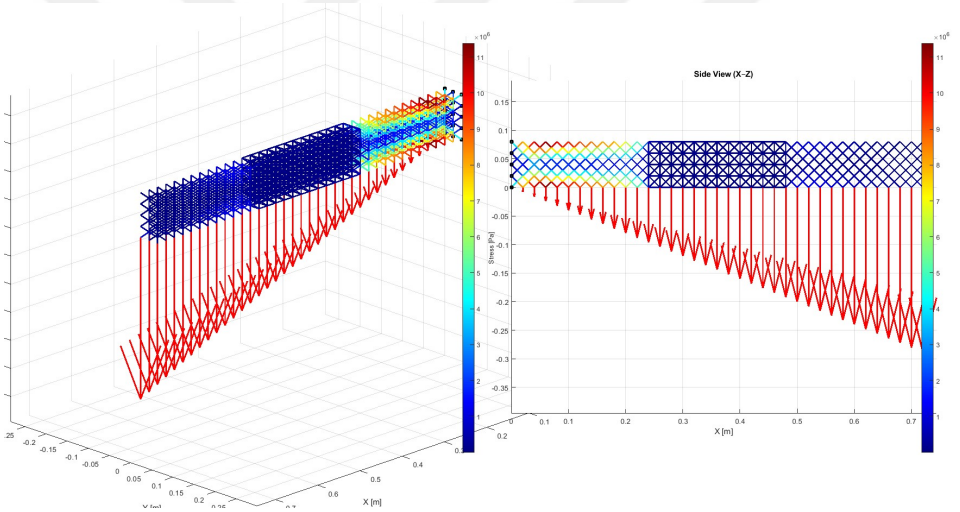


Figure 5.25: Static stress distribution for hybrid lattice beams

The stress plots show that structural changes between the sections lead to localized stress concentrations. The varying stiffness in each section affects the stress flow, with higher levels found in the rigid struts near the fixed boundary. This non-uniform distribution shows why it is important to manage transitions between different lattice types to avoid structural failure.

5.3.2. Free Vibration Analysis of Hybrid Lattice Beams

The hybrid models are analyzed under cantilever (C–F) boundary conditions to find their natural frequencies. The results show that the arrangement of lattice models has a massive effect on the dynamic behavior. A small change in the sequence can lead to a large shift in the first natural frequency (f_1). Table 5.12 provides the complete set of natural frequency results for all 180 hybrid scenarios investigated in this study. The frequency values range from 1.39 Hz to 12.99 Hz, demonstrating the extreme flexibility of hybrid designs for vibration tuning.

Table 5.12: Natural Frequency Results for 180 Hybrid Configurations

S1	S2	S3	f_1	S1	S2	S3	f_1	S1	S2	S3	f_1
1	2	1	2.13	2	1	2	3.59	2	6	1	8.32
1	2	3	1.57	2	1	3	3.11	2	6	2	7.10
1	2	4	1.85	2	1	4	3.80	2	6	3	6.20
1	2	5	1.43	2	1	5	2.81	2	6	4	7.49
1	2	6	1.71	2	1	6	3.45	2	6	5	5.63
1	3	1	2.02	2	3	1	11.96	3	1	2	3.59
1	3	2	1.70	2	3	2	11.31	3	1	3	3.11
1	3	4	1.78	2	3	4	11.84	3	1	4	3.80
1	3	5	1.40	2	3	5	9.23	3	1	5	2.81
1	3	6	1.66	2	3	6	10.67	3	1	6	3.46
1	4	1	2.16	2	4	1	12.18	3	2	1	12.47
1	4	2	1.78	2	4	2	11.63	3	2	3	10.56
1	4	3	1.58	2	4	3	10.24	3	2	4	12.56
1	4	5	1.44	2	4	5	9.34	3	2	5	9.65

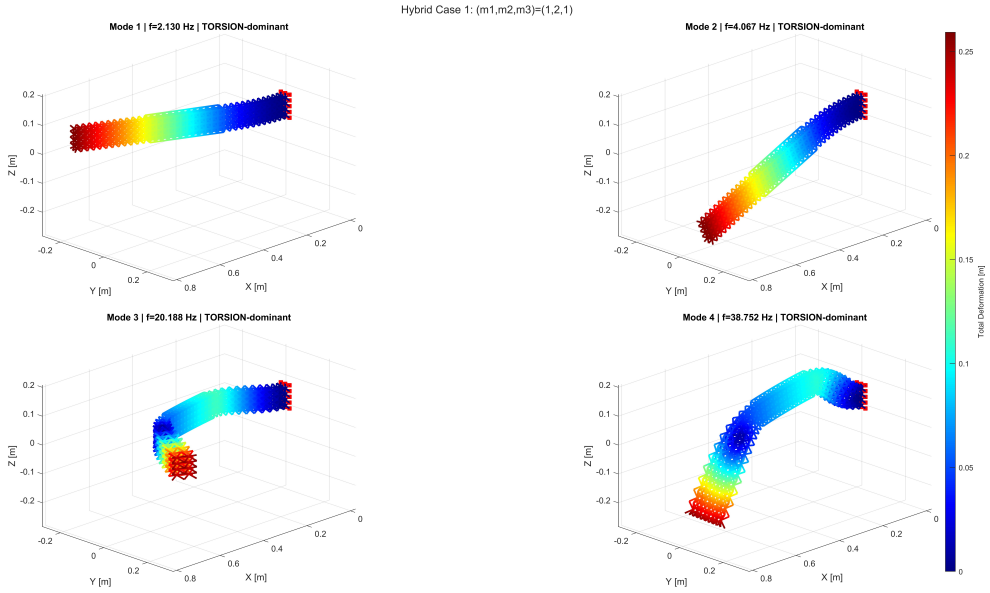
Table 5.12: Continued

S1	S2	S3	f_1	S1	S2	S3	f_1	S1	S2	S3	f_1
1	4	6	1.73	2	4	6	10.87	3	2	6	11.21
1	5	1	1.94	2	5	1	11.77	3	4	1	12.43
1	5	2	1.66	2	5	2	11.11	3	4	2	11.99
1	5	3	1.49	2	5	3	9.92	3	4	3	10.56
1	5	4	1.72	2	5	4	11.58	3	4	5	9.63
1	5	6	1.61	2	5	6	10.52	3	4	6	11.19
1	6	1	2.07	4	1	2	3.55	3	5	1	12.08
1	6	2	1.71	4	1	3	3.08	3	5	2	11.50
1	6	3	1.52	4	1	4	3.76	3	5	3	10.27
1	6	4	1.79	4	1	5	2.78	3	5	4	11.99
1	6	5	1.39	4	1	6	3.42	3	5	6	10.87
4	2	1	11.96	5	1	2	3.61	3	6	1	8.32
4	2	3	9.94	5	1	3	3.13	3	6	2	7.11
4	2	4	11.82	5	1	4	3.82	3	6	3	6.20
4	2	5	9.09	5	1	5	2.82	3	6	4	7.50
4	2	6	10.60	5	1	6	3.48	3	6	5	5.63
4	3	1	11.70	5	2	1	12.75	6	1	2	2.81
4	3	2	10.98	5	2	3	10.92	6	1	3	2.44
4	3	4	11.50	5	2	4	13.00	6	1	4	2.97
4	3	5	8.97	5	2	5	9.98	6	1	5	2.21
4	3	6	10.38	5	2	6	11.56	6	1	6	2.71
4	5	1	11.50	5	3	1	12.55	6	2	1	4.82

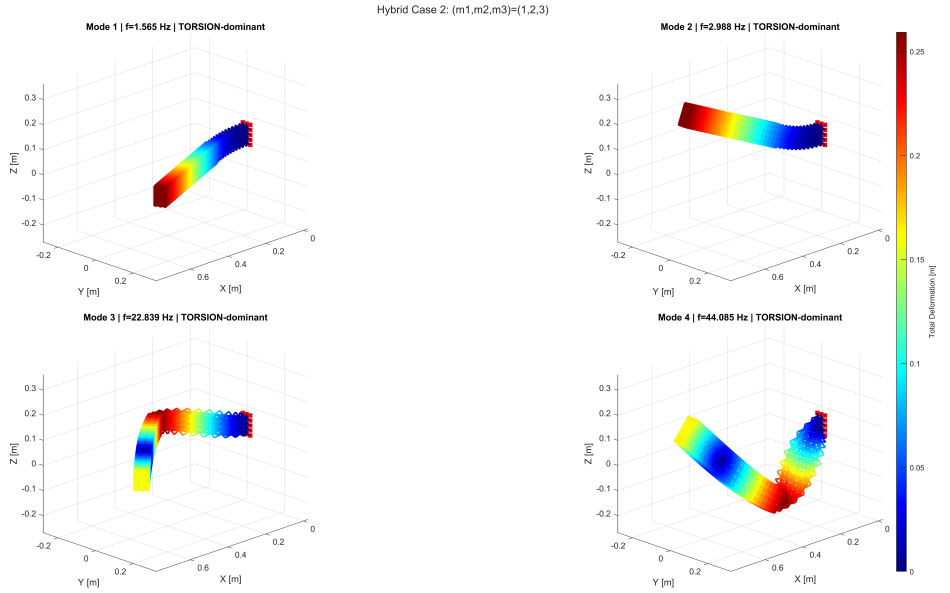
Table 5.12: Continued

S1	S2	S3	f_1	S1	S2	S3	f_1	S1	S2	S3	f_1
4	5	2	10.78	5	3	2	12.12	6	2	3	3.59
4	5	3	9.63	5	3	4	12.70	6	2	4	4.24
4	5	4	11.24	5	3	5	9.89	6	2	5	3.29
4	5	6	10.23	5	3	6	11.38	6	2	6	3.92
4	6	1	8.14	5	4	1	12.71	6	3	1	4.59
4	6	2	6.93	5	4	2	12.40	6	3	2	3.90
4	6	3	6.05	5	4	3	10.92	6	3	4	4.08
4	6	4	7.31	5	4	5	9.95	6	3	5	3.21
4	6	5	5.50	5	4	6	11.54	6	3	6	3.79
5	6	1	8.50	6	4	1	4.87	6	5	1	4.42
5	6	2	7.28	6	4	2	4.07	6	5	2	3.80
5	6	3	6.35	6	4	3	3.60	6	5	3	3.41
5	6	4	7.68	6	4	5	3.29	6	5	4	3.95
5	6	5	5.76	6	4	6	3.94	6	5	6	3.69

Figure 5.27 illustrates the f_1 values for all 180 scenarios. There are clear peaks and valleys in the graph. The highest frequency of 12.995 Hz was reached in the (5,2,4) configuration. On the other hand, the lowest frequency of 1.3904 Hz was found in the (1,6,5) configuration. This shows that switching the order of the same lattice types can completely change the stiffness of the beam. Modes for other representative cases are shown in Figure 5.28.



(a) Hybrid Case 1: (1,2,1) configuration



(b) Hybrid Case 2: (1,2,3) configuration

Figure 5.26: Modal analysis for Hybrid Cases 1 and 2

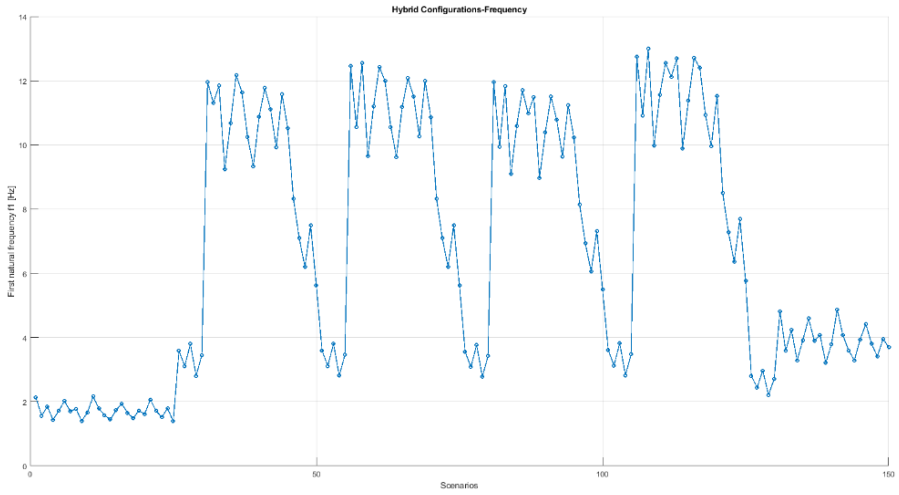
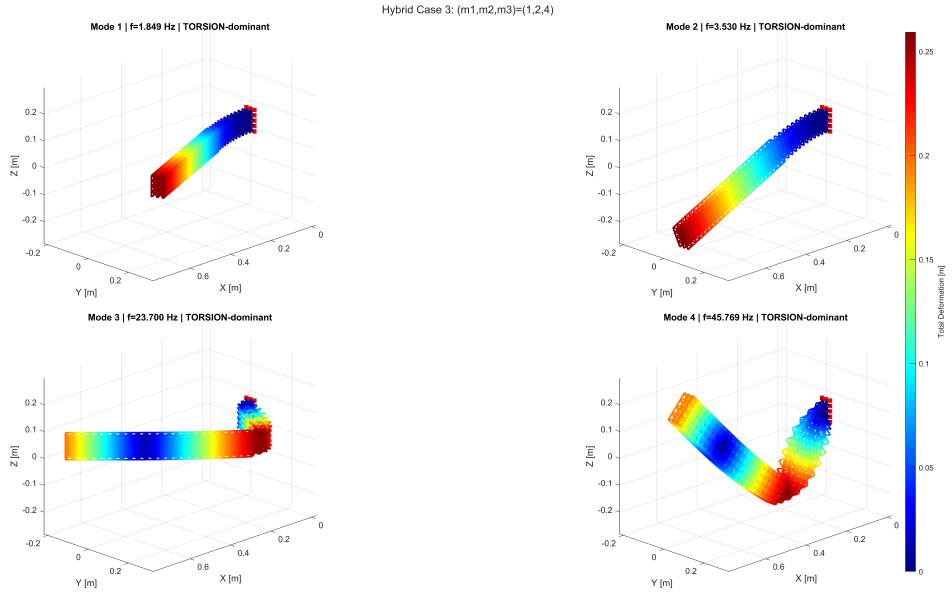
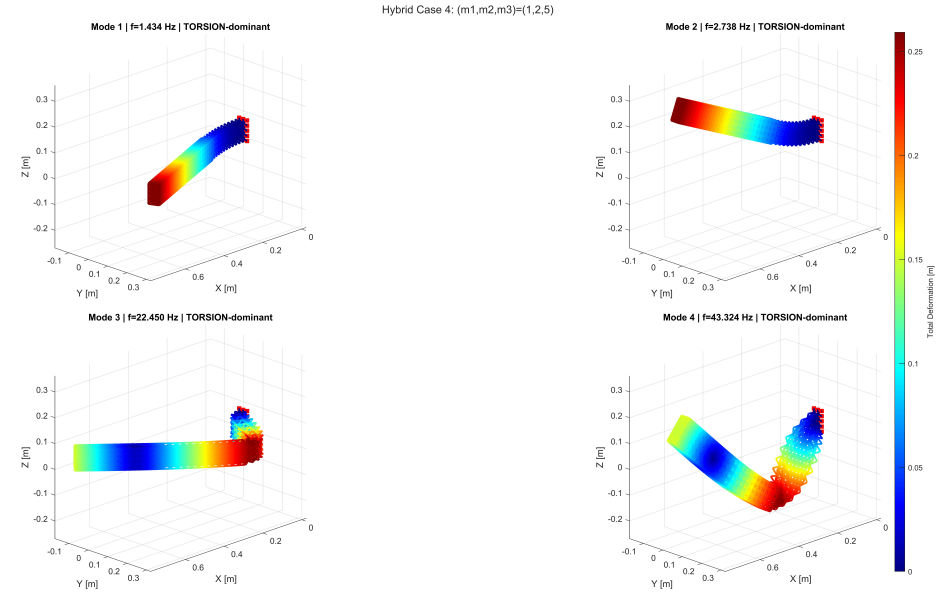


Figure 5.27: First natural frequency (f_1) for all 180 hybrid scenarios

The images in Figure 5.28 show that deformation is not uniform along the beam length. In many modes, the deformation is concentrated in specific sections. This happens because each section has a unique stiffness. The beam naturally bends more in the ”soft” sections while remaining rigid in the ”stiff” sections.



(a) Hybrid Case 3: (1,2,4) configuration



(b) Hybrid Case 4: (1,2,5) configuration

Figure 5.28: Modal analysis for Hybrid Cases 3 and 4

6. CONCLUSION

In this study, we investigated how polymer-based lattice beams behave under different static loads and vibrations. We analyzed both uniform and hybrid lattice designs using numerical models under cantilever boundary conditions. The results provide a clear understanding of how lattice geometry affects stiffness, deformation, and dynamic response.

The numerical model was developed in MATLAB using three-dimensional beam elements. The Euler–Bernoulli beam theory was employed because it is computationally efficient and suitable for slender lattice struts. This approach focuses on bending and axial deformation, which are the dominant mechanisms in lattice beams with thin members.

The results show that lattice beams are significantly more compliant than solid beams of the same outer dimensions. This leads to lower natural frequencies and larger deformations. However, this behavior is beneficial for vibration control and energy absorption. Reduced stiffness allows the structure to isolate vibrations more effectively.

Hybrid lattice beams were also studied by combining different lattice unit-cell models along the beam length. A total of 180 different hybrid configurations were evaluated. The results demonstrate that changing the sequence of lattice types provides a powerful way to tune the dynamic response of the beam without changing its overall size or mass. The ability to shift the first natural frequency by nearly 9 times just by changing the sequence is a critical discovery. This flexibility makes hybrid designs ideal for vibration control in high-performance structures like spacecraft or aerospace systems.

The first vibration modes are dominated by bending deformation. At higher modes, torsional and complex coupled motions appear due to the three-dimensional lattice geometry. These deformation patterns cannot be observed in conventional solid beams. This highlights the unique mechanical behavior enabled by lattice architectures.

Overall, lattice and hybrid lattice beams show strong potential for vibration and noise control applications. Their low stiffness, lightweight nature, and geometry-driven behavior make them well-suited for additively manufactured structures.

These characteristics are especially attractive in aerospace, automotive, and lightweight mechanical systems. Future work may extend this study in several directions. More advanced beam theories, such as the Timoshenko beam model, can be used to account for shear deformation and rotary inertia. Different materials and graded material distributions can also be explored. In addition, optimization and machine learning methods may be applied to automatically identify optimal hybrid lattice layouts for specific performance targets. Experimental validation of the numerical results is another important step for practical implementation.

In conclusion, lattice and hybrid lattice beams offer a wide and flexible design space. Their mechanical behavior can be tailored through geometry rather than material alone. This makes hybrid lattice structures a promising solution for next-generation lightweight and vibration-controlled engineering systems.

REFERENCES

[1]M. S. Escartí-Guillem, L. M. García-Raffi, and S. Hoyas, "Urans analysis of a launch vehicle aero-acoustic environment," *Applied Sciences*, vol. 12, no. 7, p. 3356, 2022.

[2]M. Al-Rifaie, A. Mian, and R. Srinivasan, "Additively manufactured lattice structures for aerospace applications: A review," *Applied Sciences*, 2022. Based on user provided screenshot context.

[3]W. P. Syam, W. Jianwei, B. Zhao, I. Maskery, W. Elmadih, and R. Leach, "Design and analysis of strut-based lattice structures for vibration isolation," *Precision Engineering*, vol. 52, pp. 494–506, 2018.

[4]E. P. Trudel, A. Mebarki, and D. Pasini, "Multiscale design optimization of additively manufactured aerospace structures employing functionally graded lattice materials," in *AIAA Scitech 2019 Forum*, p. 2349, 2019.

[5]J. Somnic and B. W. Jo, "Status and challenges in homogenization methods for lattice materials," *Materials*, vol. 15, no. 2, p. 605, 2022.

[6]D. L. Logan, *A First Course in the Finite Element Method, Enhanced Edition, SI Version*. Cengage Learning, 2016.

[7]L. J. Gibson and M. F. Ashby, *Cellular Solids: Structure and Properties*. Cambridge University Press, 2 ed., 1997.

[8]D. Bhate, "Four questions in cellular material design," *Materials*, vol. 12, no. 7, p. 1060, 2019.

[9]L. J. Gibson, "Modelling the mechanical behavior of cellular materials," *Materials Science and Engineering: A*, vol. 110, pp. 1–36, 1989.

[10]L. J. Gibson, M. F. Ashby, and B. A. Harley, *Cellular Materials in Nature and Medicine*. Cambridge University Press, 2010.

[11]M. F. Ashby, "The properties of foams and lattices," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 364, no. 1838, pp. 15–30, 2006.

[12]C. Yan, L. Hao, A. Hussein, and P. Young, "Evaluation of light-weight alsi10mg periodic cellular lattice structures fabricated via direct metal laser sintering," *Journal of Materials Processing Technology*, vol. 214, no. 4, pp. 856–864, 2014.

[13]W. Tao and M. C. Leu, “Design of lattice structure for additive manufacturing,” in *2016 International Symposium on Flexible Automation (ISFA)*, pp. 1–8, IEEE, 2016.

[14]C. Pan, Y. Han, and J. Lu, “Design and optimization of lattice structures: a review,” *Applied Sciences*, vol. 10, no. 18, p. 6374, 2020.

[15]M. F. Ashby, A. G. Evans, N. A. Fleck, L. J. Gibson, J. W. Hutchinson, and H. N. Wadley, *Metal Foams: A Design Guide*. Elsevier, 2000.

[16]G. Davies and S. Zhen, “Metallic foams: their production, properties and applications,” *Journal of Materials Science*, vol. 18, pp. 1899–1911, 1983.

[17]T. A. Schaedler and W. B. Carter, “Architected cellular materials,” *Annual Review of Materials Research*, vol. 46, no. 1, pp. 187–210, 2016.

[18]Y. Tang and Y. F. Zhao, “A survey of the design methods for additive manufacturing to improve functional performance,” *Rapid Prototyping Journal*, vol. 22, no. 3, pp. 569–590, 2016.

[19]A. Ingrole, A. Hao, and R. Liang, “Design and modeling of auxetic and hybrid honeycomb structures for in-plane property enhancement,” *Materials & Design*, vol. 117, pp. 72–83, 2017.

[20]L. Boldrin, K. Hummel, F. Scarpa, D. Di Maio, C. Lira, M. Ruzzene, C. Remillat, T. Lim, and S. Rajan, “Dynamic behaviour of auxetic gradient composite hexagonal honeycombs,” *Composite Structures*, vol. 149, pp. 114–124, 2016.

[21]R. Rezaei, M. Shakeri, and M. Kadkhodaei, “Mechanical characterization and finite element modeling of polylactic acid bcc-z cellular lattice structures fabricated by fused deposition modeling,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 231, no. 11, pp. 1995–2004, 2017.

[22]W. Van Grunsven and et al., “Fabrication and mechanical characterisation of titanium lattices with graded porosity,” *Metals*, vol. 4, no. 3, pp. 401–409, 2014.

[23]I. Maskery and et al., “An investigation into reinforced and functionally graded lattice structures,” *Journal of Cellular Plastics*, vol. 53, no. 2, pp. 151–165, 2017.

[24]S. Choy and et al., “Compressive properties of functionally graded lattice structures manufactured by selective laser melting,” *Materials & Design*, vol. 131, pp. 112–120, 2017.

[25]L. Xiao and W. Song, “Additively-manufactured functionally graded ti-6al-4v lat-

tice structures with high strength under static and dynamic loading: experiments," *International Journal of Impact Engineering*, vol. 111, pp. 255–272, 2018.

[26]S. Altamimi, A. Karrech, and B. Killen, "On stiffness, strength, anisotropy, and buckling of 30 strut-based lattices with cubic crystal structures," *Advanced Engineering Materials*, vol. 23, p. 2001379, 2021.

[27]S. O. Obadimu and K. I. Kourousis, "Compressive behaviour of additively manufactured lattice structures: a review," *Aerospace Science and Technology*, vol. 108, p. 106405, 2021.

[28]X. Geng and et al., "A fem study on mechanical behavior of cellular lattice materials based on combined elements," *Materials Science and Engineering: A*, vol. 712, pp. 188–198, 2018.

[29]H. Rahman and et al., "Energy absorption and mechanical performance of functionally graded soft–hard lattice structures," *Materials*, vol. 14, pp. 1–15, 2021.

[30]O. Al-Ketan, R. Rowshan, and R. K. Abu Al-Rub, "Topology-mechanical property relationship of 3d printed strut, skeletal, and sheet based periodic metallic cellular materials," *Additive Manufacturing*, vol. 19, pp. 167–183, 2018.

[31]M. H. Luxner, J. Stampfl, and H. E. Pettermann, "Finite element modeling concepts and linear analyses of 3d regular open cell structures," *Journal of Materials Science*, vol. 40, no. 22, pp. 5859–5866, 2005.

[32]M. Smith, Z. Guan, and W. Cantwell, "Finite element modelling of the compressive response of lattice structures manufactured using the selective laser melting technique," *International Journal of Mechanical Sciences*, vol. 67, pp. 28–41, 2013.

[33]P. G. Martinsson and A. Movchan, "Vibrations of lattice structures and phononic band gaps," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 56, no. 1, pp. 45–64, 2003.

[34]L. Jaouen, A. Renault, and M. Deverge, "Elastic and damping characterizations of acoustical porous materials: Available experimental methods and applications to a melamine foam," *Applied Acoustics*, vol. 69, no. 12, pp. 1129–1140, 2008.

[35]H. Sun, X. Du, and P. Pai, "Theory of metamaterial beams for broadband vibration isolation," *Journal of Intelligent Material Systems and Structures*, vol. 21, no. 11, pp. 1085–1101, 2010.

[36]E. Baravelli and M. Ruzzene, "Internally resonating lattices for bandgap generation and low-frequency vibration control," *Journal of Sound and Vibration*, vol. 332, no. 25,

pp. 6562–6579, 2013.

[37]Z. Wu, F. Li, and C. Zhang, “Vibration band-gap properties of three-dimensional kagome lattices using the spectral element method,” *Journal of Sound and Vibration*, vol. 341, pp. 162–173, 2015.

[38]H. Mao, Y. Huang, and T. Yang, “Isolation performance analysis of a porous material vibration isolator,” *Applied Mechanics and Materials*, vol. 741, pp. 405–410, 2015.

[39]M. Azmi, R. Ismail, R. Hasan, and et al., “Vibration analysis of fdm printed lattice structure bar,” in *Proceedings of the SAKURA Symposium on Mechanical Science and Engineering*, pp. 33–35, 2017.

[40]R. Ramadani, A. Belsak, M. Kegl, and et al., “Topology optimization-based design of lightweight and low vibration gear bodies,” *International Journal of Simulation Modelling*, vol. 17, no. 1, pp. 92–104, 2018.

[41]S. Yin and N. Rayess, “Vibration isolation of polymer-metal foam hybrids,” *Journal of Vibration and Acoustics*, 2014.

[42]L. Dahil and A. Karabulut, “Decreasing vibration on the foam legged of passenger seat in vehicle,” *Journal of Transportation Systems*, 2016.

[43]T. Li and et al., “Vibration bandgaps analysis of different unit cells,” *Physics Letters A*, 2017.

[44]M. Hajhosseini, “Analysis of complete vibration band gaps in a new periodic lattice model using the differential quadrature method,” *Journal of Vibration and Control*, vol. 26, pp. 1708–1720, 2020.

[45]S. Andresen and et al., “Eigenfrequency analysis of irregular lattice structures,” *Computers & Structures*, 2020.

[46]U. Simsek and et al., “Vibration characteristic of sandwich gyroid unit cell structure,” *Composite Structures*, 2019.

[47]X. An, C. Lai, W. He, and et al., “Three-dimensional meta-truss lattice composite structures with vibration isolation performance,” *Extreme Mechanics Letters*, vol. 33, p. 100577, 2019.

[48]W. Elmadih and et al., “Metamaterials for low-frequency vibration attenuation,” *Materials & Design*, 2019.

[49]H. Fan and et al., “Vibration isolation of metastructures with quasi-zero dynamic

stiffness," *Mechanical Systems and Signal Processing*, 2020.

[50]S. Sahmani, A. Fattahi, and N. Ahmed, "Develop a refined truncated cubic lattice structure for nonlinear large-amplitude vibrations of micro/nano-beams made of nanoporous materials," *Engineering with Computers*, vol. 36, pp. 359–375, 2020.

[51]M. M. Opgenoord and K. E. Willcox, "Design for additive manufacturing: cellular structures in early-stage aerospace design," *Structural and Multidisciplinary Optimization*, vol. 60, pp. 411–428, 2019.

[52]S. Moon and et al., "Application of 3d printing technology for designing lightweight unmanned aerial vehicle wing structures," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 1, pp. 223–228, 2014.

[53]Magna Parva, "Thermal protection system." Available online: magnaparva.com, 2020. Accessed 2024.

[54]A. Spadoni and M. Ruzzene, "Numerical and experimental analysis of the static compliance of chiral truss-core airfoils," *Journal of Mechanics of Materials and Structures*, vol. 2, no. 5, pp. 965–981, 2007.

[55]N. B. Cramer, D. W. Cellucci, O. Formica, C. L. Gregg, B. Jenett, S. S. Swei, and K. C. Cheung, "Elastic shape morphing of ultralight structures by programmable assembly," *Smart Materials and Structures*, vol. 28, no. 5, p. 055006, 2019.

[56]M. Kouach, "Methods for modelling lattice structures," degree project in solid mechanics, second cycle, 30 credits, KTH Royal Institute of Technology, School of Engineering Sciences, Stockholm, Sweden, 2019.

BIOGRAPHY

Engin YILMAZ is research assistant at Gebze Technical University, Department of Aerospace Engineering. He graduated from Bursa Technical University, Department of Mechanical Engineering in 2021. He is currently studying for his master's degree at Gebze Technical University, conducting master's thesis research under the supervision of Assist. Prof. Mohammad RAHMANIAN.

PUBLICATIONS AND PRESENTATIONS FROM THE THESIS

- Rahmanian, M. and Yilmaz, E., “Finite Element Approach to Bending Analysis of Strut-Based Lattice Reinforced Beams”, *International Conference on Experimental and Computational Engineering Sciences (ICECES’25)*, Ankara, Turkey, December 11-12, 2025.

APPENDICES

Appendix A: Bending Analysis Results for Different Models

This appendix provides the visual results for the bending analysis of the lattice models discussed in the thesis. The figures show the deformation patterns and load responses for each configuration.

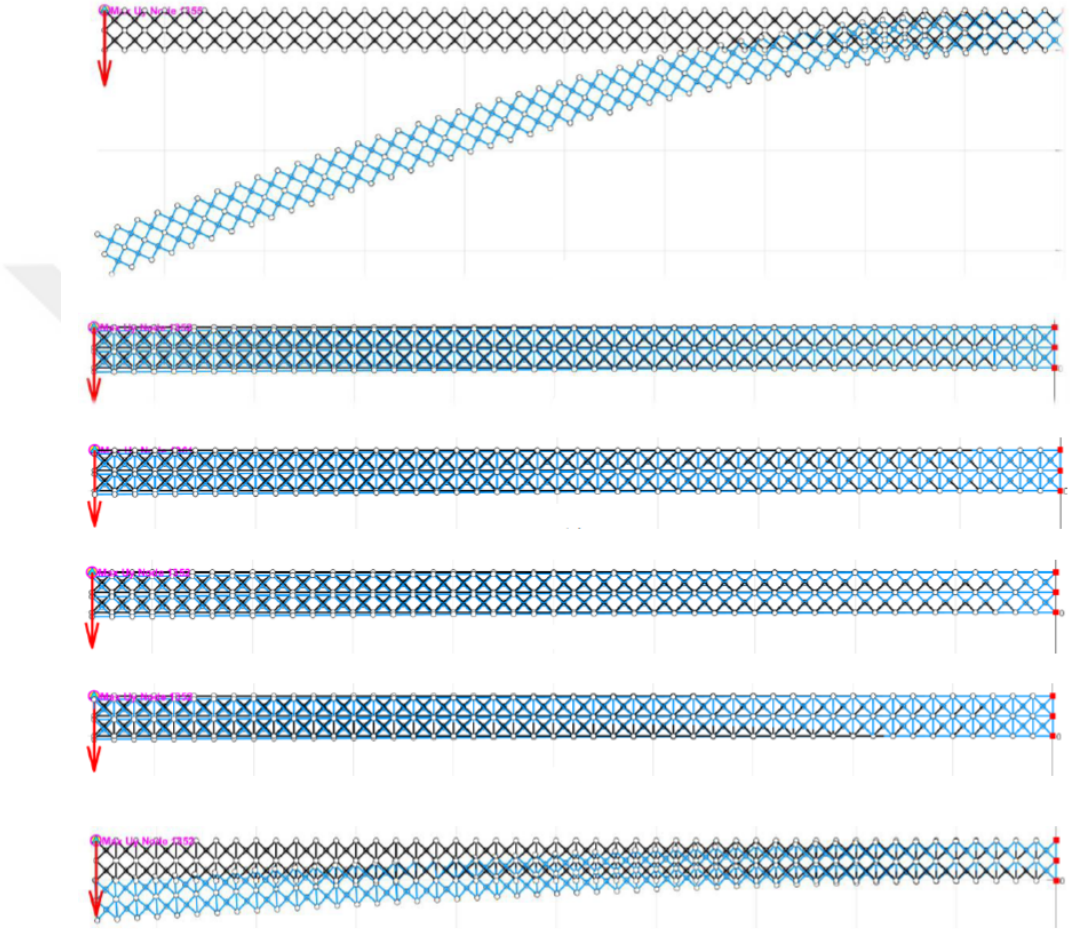


Figure A.1: Bending deformation shapes for various models

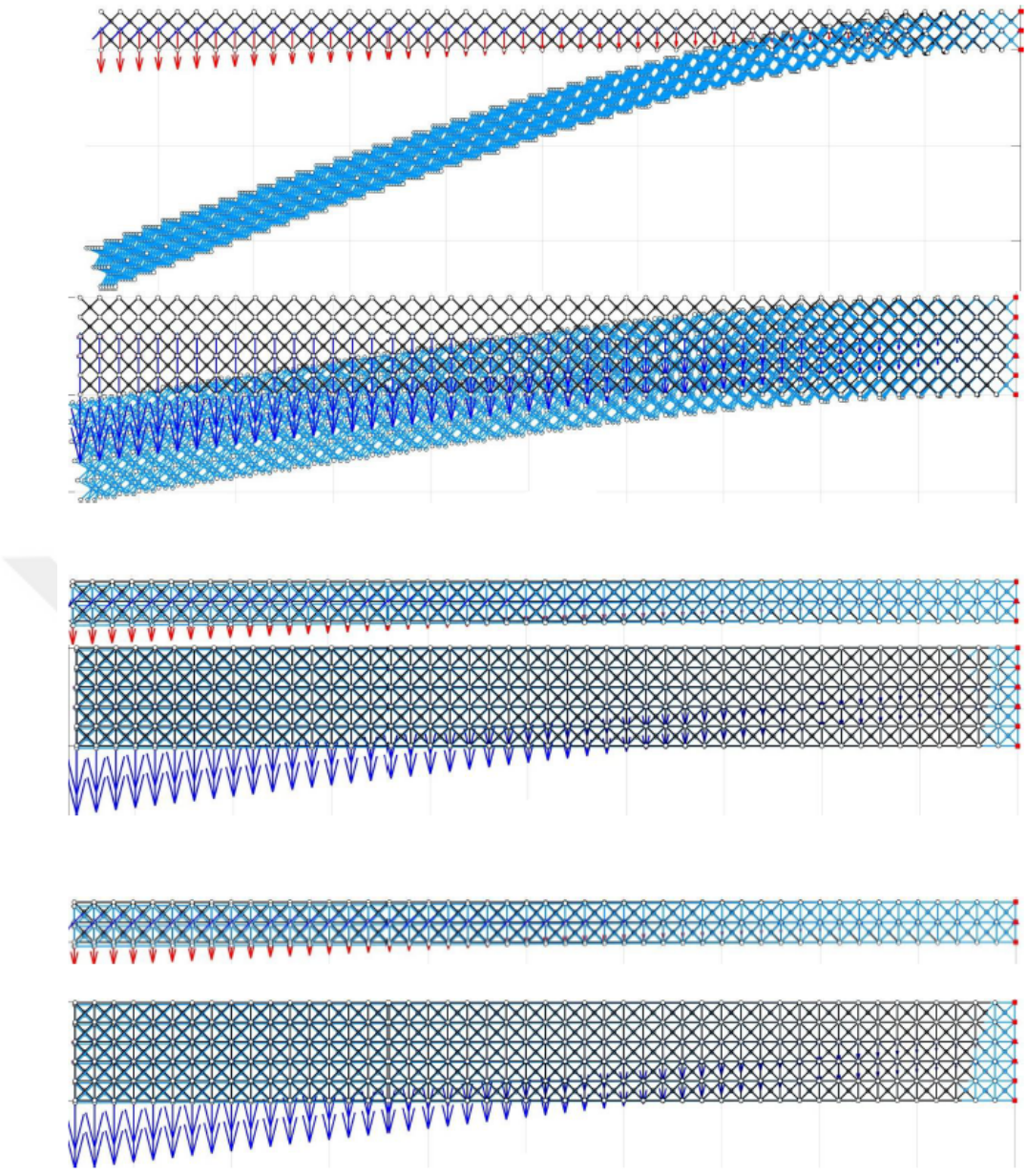


Figure A.2: Further results for bending analysis with different loads

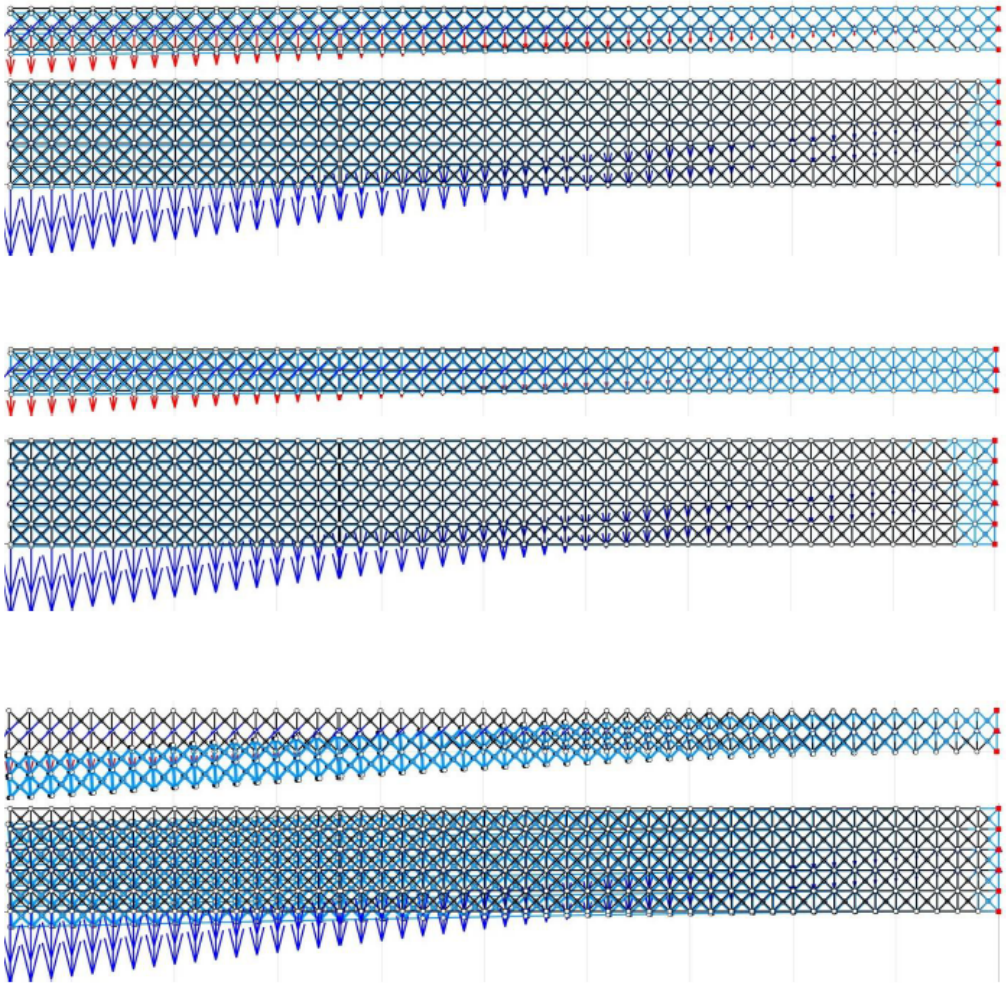


Figure A.3: Deformation patterns under varied spanwise and chordwise loading conditions

Appendix B: MATLAB Code for Finite Element Analysis

This appendix contains the main MATLAB script used for the bending and free vibration analysis of the lattice reinforced beams.

```
clc; clear; close all;

%% ---------------- DIMENSION PARAMETERS ----------------
a  = 0.02;                % [m]
Nx = 24; Ny = 2; Nz = 5; % Number of RVEs

%% --------------- MATERIAL&SECTION PROPERTIES ----------
E   = 2.5e9;     % Pa
G   = 0.9615e9;  % Pa
rho = 1010;      % kg/m^3

A   = 5.560e-6;
Iy  = 2.484e-12;
Iz  = 2.484e-12;
J   = 4.969e-12;

%% ---------------- FORCE INPUTS ----------------
L  = 0.48;           % toplam şkiri boyu [m]
dx = 0.02;           % ıadm [m]

x_vals = (0:dx:L)';

y_coord = 0.04;         % sabit y
z_layers = [0.04, 0.06];  % iki düzlem

%% ------------------------------------------------------------------------

% Lineer artan ııyayl yük
w = x_vals;
Fy_line = -1.5 * (w / sum(w));

Fy_line(end) = Fy_line(end) + (-1.5 - sum(Fy_line));

Fy_half = Fy_line / numel(z_layers);


force_coords = [];
force_vals   = [];

for k = 1:numel(z_layers)
    z_coord = z_layers(k) * ones(size(x_vals));
    coords_k = [x_vals, y_coord*ones(size(x_vals)), z_coord];

    vals_k   = [zeros(size(x_vals)), Fy_half, zeros(size(x_vals))];

    force_coords = [force_coords; coords_k];
    force_vals   = [force_vals; vals_k];
end

%% --- Kontrol ---
fprintf('Toplam uygulanan Fy = %.6f N\n', sum(force_vals(:,2)));
fprintf('Nokta ıısays: %d\n', size(force_coords,1));


%% ---------------- GEOMETRY ----------------
plot_scale = 1; label_nodes = true;

cell_nodes = a * [
    0 0 0; 1 0 0; 1 1 0; 0 1 0;
    0 0 1; 1 0 1; 1 1 1; 0 1 1;
    0.5 0.5 0.5
];

cell_elems =[
    1 9;
    2 9;
    3 9;    % model-1 sadece
```

```
    4 9;
    5 9;
    6 9;
    7 9;
    8 9;

7 2;
5 4;    % Model-6 için aç kapa
6 8;
1 3;

];

[Node_coords, Elem_nodes] = tile_and_merge(cell_nodes, cell_elems, Nx, Ny, Nz, a);
fprintf('Topology: %d nodes, %d elements\n', size(Node_coords,1), size(Elem_nodes,1))
    ;

No_nodes = size(Node_coords,1);
No_elem  = size(Elem_nodes,1);
DOFpn    = 6;
TotalDOF = No_nodes * DOFpn;
tol = 1e-9;

%% ---------------- FIXED NODES ----------------
tol = 1e-9;
fixed_coord = Node_coords(abs(Node_coords(:,1)) < tol, :);
%% ---------------- GLOBAL STIFFNESS ----------------
DOF = zeros(No_elem,12);
for e = 1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    DOF(e,:) = [DOFpn*(n1-1)+(1:DOFpn), DOFpn*(n2-1)+(1:DOFpn)];
end

K = sparse(TotalDOF, TotalDOF);

for e = 1:No_elem
    n1 = Elem_nodes(e,1);
    n2 = Elem_nodes(e,2);
    xi = Node_coords(n1,:);
    xj = Node_coords(n2,:);
    dx = xj - xi;
    L = norm(dx);
    lambda = local_axes(dx);
    k_local = beam12_local(E,G,A,Iy,Iz,J,L);
    T = blkdiag(lambda, lambda, lambda, lambda);
    ke = T' * k_local * T;
    dof = DOF(e,:);
    K(dof,dof) = K(dof,dof) + ke;
end


%% ---------------- FORCE VECTOR ----------------
F = zeros(TotalDOF,1);
for k = 1:size(force_coords,1)
    node = find(vecnorm(Node_coords - force_coords(k,:),2,2) < tol, 1);
    F((node-1)*DOFpn+(1:3)) = F((node-1)*DOFpn+(1:3)) + force_vals(k,:)';
end

%% ---------------- BOUNDARY CONDITIONS ----------------
fixed_DOFs = [];
for i = 1:size(fixed_coord,1)
    node = find(vecnorm(Node_coords - fixed_coord(i,:),2,2) < tol, 1);
    fixed_DOFs = [fixed_DOFs, (node-1)*DOFpn + (1:6)];
end

free_DOFs = setdiff(1:TotalDOF, fixed_DOFs);

%% ---------------- SOLVE ----------------
U = zeros(TotalDOF,1);
U(free_DOFs) = K(free_DOFs,free_DOFs) \ F(free_DOFs);
R = K*U - F;

Ux = U(1:6:end)*1e3; Uy = U(2:6:end)*1e3; Uz = U(3:6:end)*1e3;
```

```
T_out = table((1:No_nodes).', Node_coords(:,1),Node_coords(:,2),Node_coords(:,3), ...
              Ux,Uy,Uz, ...
              'VariableNames',{'Node','X','Y','Z','Ux_mm','Uy_mm','Uz_mm'});
disp(T_out);

[max_disp, max_node] = max(abs(Uy));
fprintf('\nMax Deformation (Uy) = %.6f m at node %d\n', (max_disp/1e3), max_node);

%% ---------------- PLOT ----------------
figure('Color','w'); hold on; axis equal; grid on;
title('Force - Displacement');
xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');
draw_lattice(Node_coords, Elem_nodes, 'k', label_nodes);

Node_def = Node_coords + plot_scale * [Ux Uy Uz]/1e3;
draw_lattice(Node_def, Elem_nodes, [0 0.6 1], false);

% Fixed nodes
fixed_nodes = unique(floor((fixed_DOFs-1)/DOFpn) + 1);
plot3(Node_coords(fixed_nodes,1),Node_coords(fixed_nodes,2),Node_coords(fixed_nodes
    ,3), ...
      'rs','MarkerFaceColor','r','MarkerSize',8);

% Force nodes
for k = 1:size(force_coords,1)
    node = find(vecnorm(Node_coords - force_coords(k,:),2,2) < tol, 1);
    plot3(Node_coords(node,1),Node_coords(node,2),Node_coords(node,3), ...
        'b^','MarkerFaceColor','c','MarkerSize',8);
end

% Kuvvet oklar
force_scale = 10;
for k = 1:size(force_coords,1)
    node = find(vecnorm(Node_coords - force_coords(k,:),2,2) < tol, 1);
    Fvec = force_vals(k,:);
    quiver3(Node_coords(node,1),Node_coords(node,2),Node_coords(node,3), ...
            Fvec(1)*force_scale, Fvec(2)*force_scale, Fvec(3)*force_scale, ...
            'Color','r','LineWidth',2,'MaxHeadSize',5);
end
view(135,20);

fprintf('\nFinal mesh: %d nodes, %d unique elements\n', size(Node_coords,1), size(
    Elem_nodes,1));
fprintf('Applied total force: %.2f N\n', sum(vecnorm(force_vals,2,2)));

%% ---------------- FUNCTIONS ----------------
function [Nodes, Elems] = tile_and_merge(cell_nodes, cell_elems, Nx, Ny, Nz, a)
Nodes = zeros(0,3); Elems = zeros(0,2);
node_map = containers.Map('KeyType','char','ValueType','int32');
tol = 1e-9; nid = 0;
for ix=0:Nx-1
  for iy=0:Ny-1
    for iz=0:Nz-1
      O = [ix*a, iy*a, iz*a];
      local2global = zeros(size(cell_nodes,1),1);
      for i=1:size(cell_nodes,1)
        p = cell_nodes(i,:) + O;
        key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol,
    round(p(3)/tol)*tol);
        if isKey(node_map,key)
            local2global(i) = node_map(key);
        else
            nid = nid + 1; node_map(key)=nid;
            Nodes(nid,:) = p;
            local2global(i) = nid;
        end
      end
      for e=1:size(cell_elems,1)
        Elems(end+1,:) = [local2global(cell_elems(e,1)), local2global(cell_elems(e,2)
    )];
      end
    end
  end
```

```
end
Elems = unique(sort(Elems,2),'rows');
end

function lambda = local_axes(dx)
    L = norm(dx);
    ex = dx(:)/L;
    ref = [0;0;1];
    if abs(dot(ex,ref)) > 0.99
        ref = [0;1;0];
    end
    ez = cross(ex,ref); ez = ez/norm(ez);
    ey = cross(ez,ex);  ey = ey/norm(ey);
    lambda = [ex ey ez]';
end


function k = beam12_local(E,G,A,Iy,Iz,J,L)
EA=E*A; GJ=G*J; EIy=E*Iy; EIz=E*Iz; L2=L^2; L3=L^3;
k = zeros(12);
k(1,1)=EA/L; k(1,7)=-EA/L; k(7,1)=-EA/L; k(7,7)=EA/L;
k(4,4)=GJ/L; k(4,10)=-GJ/L; k(10,4)=-GJ/L; k(10,10)=GJ/L;
k(2,2)=12*EIz/L3; k(2,6)=6*EIz/L2; k(2,8)=-12*EIz/L3; k(2,12)=6*EIz/L2;
k(6,2)=6*EIz/L2; k(6,6)=4*EIz/L; k(6,8)=-6*EIz/L2; k(6,12)=2*EIz/L;
k(8,2)=-12*EIz/L3; k(8,6)=-6*EIz/L2; k(8,8)=12*EIz/L3; k(8,12)=-6*EIz/L2;
k(12,2)=6*EIz/L2; k(12,6)=2*EIz/L; k(12,8)=-6*EIz/L2; k(12,12)=4*EIz/L;
k(3,3)=12*EIy/L3; k(3,5)=-6*EIy/L2; k(3,9)=-12*EIy/L3; k(3,11)=-6*EIy/L2;
k(5,3)=-6*EIy/L2; k(5,5)=4*EIy/L; k(5,9)=6*EIy/L2; k(5,11)=2*EIy/L;
k(9,3)=-12*EIy/L3; k(9,5)=6*EIy/L2; k(9,9)=12*EIy/L3; k(9,11)=6*EIy/L2;
k(11,3)=-6*EIy/L2; k(11,5)=2*EIy/L; k(11,9)=6*EIy/L2; k(11,11)=4*EIy/L;
end

function draw_lattice(Nodes, Elems, color_in, show_labels)
if nargin<3, color_in=[0.2 0.5 0.9]; end
if nargin<4, show_labels=false; end
for e=1:size(Elems,1)
    p=Nodes(Elems(e,:),:);
    plot3(p(:,1),p(:,2),p(:,3),'-','Color',color_in,'LineWidth',1.8);
end
plot3(Nodes(:,1),Nodes(:,2),Nodes(:,3),'ko','MarkerFaceColor','w');
if show_labels
    for i=1:size(Nodes,1)
        text(Nodes(i,1),Nodes(i,2),Nodes(i,3),sprintf(' %d',i));
    end
end
end
```

Appendix C: MATLAB Code for Hybrid Lattice Analysis

This appendix contains the MATLAB script used for the modal analysis of various hybrid lattice configurations. This code automatically generates combinations, carries out mesh refinement, and visualizes the mode shapes.

```
clc; clear; close all;

%% ================= USER SETTINGS=================
nMode    = 4;
MAX_CASE = 4;
Kplot    = 4;

plot_scale_mode = 6;   % visual multiplier
n_sub = 1;               % sub-elements per strut

% ---- Macro dimensions ----
a  = 0.02;                % [m] cell size
Nx = 36; Ny = 2; Nz = 4;   % number of RVEs

% ---- 3 sections along X (must sum to Nx) ----
secLen = [12 12 12];
assert(sum(secLen)==Nx, 'secLen must sum to Nx.');

% ---- Hybrid rule ----

rule_all_three_diff = false;

% ---- Boundary condition ----
BC = 'CF';

%% --------------- MATERIAL & SECTION PROPERTIES ----------
E   = 2.5e9;      % Pa
G   = 0.9615e9;   % Pa
rho = 1010;       % kg/m^3

A   = 5.560e-6;   % m^2
Iy  = 2.484e-12;  % m^4
Iz  = 2.484e-12;  % m^4
J   = 4.969e-12;  % m^4

%% ================= UNIT CELL NODES =================
cell_nodes = a * [
    0 0 0; 1 0 0; 1 1 0; 0 1 0;
    0 0 1; 1 0 1; 1 1 1; 0 1 1;
    0.5 0.5 0.5
];

%% ================= MODEL CONNECTIVITIES (1..6) =================
base = [
    1 9; 2 9; 3 9; 4 9;
    5 9; 6 9; 7 9; 8 9;
];

frame8 = [
    4 8; 3 4; 7 8; 3 7;
    1 2; 2 6; 6 5; 5 1;
];

modelElems = cell(6,1);
modelElems{1} = base;
modelElems{2} = [base; frame8; 1 4; 5 8; 2 3; 6 7];
modelElems{3} = [base; frame8; 8 3; 2 5; 6 3; 5 4; 6 8; 2 4];
modelElems{4} = [base; frame8];
modelElems{5} = [base; frame8; 1 4; 5 8; 2 3; 6 7; 7 4; 6 1; 7 2; 5 4; 6 8; 1 3];
modelElems{6} = [base; 7 2; 5 4; 6 8; 1 3];

for i=1:6
    modelElems{i} = unique(sort(modelElems{i},2),'rows');
end

%% ================= GENERATE ALL HYBRID TRIPLETS =================
```

```
triplets = [];
for m1 = 1:6
    for m2 = 1:6
        if m2==m1, continue; end
        for m3 = 1:6
            if m3==m2, continue; end
            if rule_all_three_diff && (m3==m1), continue; end
            triplets(end+1,:) = [m1 m2 m3]; %#ok<AGROW>
        end
    end
end
fprintf('Total hybrid combinations available = %d\n', size(triplets,1));

%% ================= LIMIT HOW MANY CASES TO SOLVE =================
MAX_CASE = min(MAX_CASE, size(triplets,1));
triplets = triplets(1:MAX_CASE,:);
nCases   = size(triplets,1);
Kplot    = min(Kplot, nCases);
fprintf('=> Solving only first %d cases, plotting first %d cases.\n', nCases, Kplot);

%% ================= RUN CASES =================
freq_all = nan(nCases, nMode);
mass_all = nan(nCases, 1);

store = cell(Kplot,1);

for ic = 1:nCases
    m1 = triplets(ic,1); m2 = triplets(ic,2); m3 = triplets(ic,3);

    model_per_ix = [ ...
        repmat(m1, 1, secLen(1)), ...
        repmat(m2, 1, secLen(2)), ...
        repmat(m3, 1, secLen(3)) ...
    ];

    fprintf('\nCase %d/%d  (m1,m2,m3)=(%d,%d,%d)\n', ic, nCases, m1,m2,m3);

    % --- build hybrid mesh ---
    [Node_coords, Elem_nodes] = tile_and_merge_hybrid(cell_nodes, modelElems,
    model_per_ix, Ny, Nz, a);

    % --- refine ---
    [Node_coords, Elem_nodes] = refine_elements(Node_coords, Elem_nodes, n_sub);

    fprintf('Topology: %d nodes, %d elements (n_sub=%d)\n', size(Node_coords,1), size
    (Elem_nodes,1), n_sub);

    % --- modal solve ---
    if ~strcmpi(BC,'CF'), error('Only CF supported here.'); end
    [freqHz, totalMass, Phi_full, free_DOFs, fixed_DOFs] = run_modal_CF_multi_full(
    Node_coords, Elem_nodes, E,G,rho,A,Iy,Iz,J,nMode);

    freq_all(ic,:) = freqHz(:).';
    mass_all(ic)   = totalMass;

    fprintf('  f(1..%d) = %s Hz\n', nMode, mat2str(freqHz(:).',6));

    if ic <= Kplot
        store{ic} = struct('Nodes',Node_coords,'Elems',Elem_nodes,'triplet',[m1 m2 m3
    ], ...
                           'freq',freqHz,'Phi_full',Phi_full,'free',free_DOFs,'fixed'
    ,fixed_DOFs);
    end
end

%% ================= PLOT MODE SHAPES  =================
modesToShow = min(4, nMode);

for k = 1:Kplot
    S = store{k};
    Node_coords = S.Nodes;
    Elem_nodes  = S.Elems;
    freqHz      = S.freq;
```

```
    Phi_full    = S.Phi_full;
    free_DOFs   = S.free;
    fixed_DOFs  = S.fixed;

    DOFpn = 6;
    TotalDOF = size(Node_coords,1)*DOFpn;

    fixed_nodes = unique(floor((fixed_DOFs-1)/DOFpn) + 1);

    Lmodel = max(Node_coords(:,1)) - min(Node_coords(:,1));

    % --- ------ ---
    Node_def_all = cell(modesToShow,1);
    Umag_def_all = cell(modesToShow,1);
    modeTypeStr  = strings(modesToShow,1);

    xmin=+inf; xmax=-inf; ymin=+inf; ymax=-inf; zmin=+inf; zmax=-inf;
    cmax_global = 0;

    for im = 1:modesToShow
        Umode = zeros(TotalDOF,1);
        Umode(free_DOFs) = Phi_full(free_DOFs, im);

        Ux  = Umode(1:6:end);
        Uy  = Umode(2:6:end);
        Uz  = Umode(3:6:end);
        Thx = Umode(4:6:end);
        Thy = Umode(5:6:end);
        Thz = Umode(6:6:end);

        % --- ------- ---
        Etr  = sum(Ux.^2 + Uy.^2 + Uz.^2);
        Erot = sum(Thx.^2 + Thy.^2 + Thz.^2);
        r = Erot / max(Etr, 1e-30);

        if r > 5
            modeTypeStr(im) = "TORSION-dominant";
        elseif r < 0.2
            modeTypeStr(im) = "BENDING-dominant";
        else
            modeTypeStr(im) = "MIXED";
        end

        Umag = sqrt(Ux.^2 + Uy.^2 + Uz.^2);

        mag_max = max(Umag);
        if mag_max < 1e-14
            scaleU = plot_scale_mode;
        else
            scaleU = (0.06*Lmodel/mag_max) * plot_scale_mode;
        end

        Node_def = Node_coords + scaleU*[Ux Uy Uz];
        Umag_def = Umag * scaleU;

        Node_def_all{im} = Node_def;
        Umag_def_all{im} = Umag_def;

        xmin = min([xmin; Node_coords(:,1); Node_def(:,1)]);
        xmax = max([xmax; Node_coords(:,1); Node_def(:,1)]);
        ymin = min([ymin; Node_coords(:,2); Node_def(:,2)]);
        ymax = max([ymax; Node_coords(:,2); Node_def(:,2)]);
        zmin = min([zmin; Node_coords(:,3); Node_def(:,3)]);
        zmax = max([zmax; Node_coords(:,3); Node_def(:,3)]);

        cmax_global = max(cmax_global, max(Umag_def));
    end

    pad = 0.03 * max([xmax-xmin, ymax-ymin, zmax-zmin] + eps);
    xlim_fixed = [xmin-pad, xmax+pad];
    ylim_fixed = [ymin-pad, ymax+pad];
    zlim_fixed = [zmin-pad, zmax+pad];
```

```
    % --- 2x2 plot ---
    figure('Color','w');
    tl = tiledlayout(2,2,'TileSpacing','compact','Padding','compact');
    title(tl, sprintf('Hybrid Case %d: (m1,m2,m3)=(%d,%d,%d)', k, S.triplet(1), S.
    triplet(2), S.triplet(3)));

    for im = 1:modesToShow
        ax = nexttile; hold(ax,'on'); grid(ax,'on'); axis(ax,'equal');
        xlabel(ax,'X [m]'); ylabel(ax,'Y [m]'); zlabel(ax,'Z [m]');
        view(ax, 135, 20);
        xlim(ax, xlim_fixed); ylim(ax, ylim_fixed); zlim(ax, zlim_fixed);

        title(ax, sprintf('Mode %d | f=%.3f Hz | %s', im, freqHz(im), modeTypeStr(im)
    ));

        Node_def = Node_def_all{im};
        Umag_def = Umag_def_all{im};

        % --- ---- ---
        for e = 1:size(Elem_nodes,1)
            n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);

            X = [Node_def(n1,1) Node_def(n2,1)];
            Y = [Node_def(n1,2) Node_def(n2,2)];
            Z = [Node_def(n1,3) Node_def(n2,3)];
            C = [Umag_def(n1)   Umag_def(n2)];

            surface(ax, ...
                'XData',[X;X], 'YData',[Y;Y], 'ZData',[Z;Z], ...
                'CData',[C;C], ...
                'FaceColor','none', 'EdgeColor','interp', ...
                'LineWidth',2.0);
        end

        % fixed nodes markers
        plot3(ax, Node_coords(fixed_nodes,1),Node_coords(fixed_nodes,2),Node_coords(
    fixed_nodes,3), ...
              'rs','MarkerFaceColor','r','MarkerSize',6);

        colormap(ax, jet);
        caxis(ax, [0, cmax_global+eps]);
    end

    cb = colorbar;
    cb.Layout.Tile = 'east';
    cb.Label.String = 'Total Deformation [m]';
end

%% ================= f1 TREND  =================
figure('Color','w'); grid on; hold on;
plot(1:nCases, freq_all(:,1), '-o', 'LineWidth', 1.2, 'MarkerSize', 4);
xlabel('Case index (solved subset)');
ylabel('First natural frequency f1 [Hz]');
title(sprintf('Hybrid sweep: f1 variation (N=%d solved cases)', nCases));

%% ================= RESULTS TABLE =================
T = table(triplets(:,1),triplets(:,2),triplets(:,3), mass_all, freq_all(:,1), ...
    'VariableNames', {'Sec1','Sec2','Sec3','Mass_kg','f1_Hz'});
disp(T);

%% ========================= FUNCTIONS =========================
function [Nodes, Elems] = tile_and_merge_hybrid(cell_nodes, modelElems, model_per_ix,
     Ny, Nz, a)
Nodes = zeros(0,3); Elems = zeros(0,2);
node_map = containers.Map('KeyType','char','ValueType','int32');
tol = 1e-9; nid = 0;

Nx = numel(model_per_ix);

for ix = 0:Nx-1
    mid = model_per_ix(ix+1);
    cell_elems = modelElems{mid};

```

```
    for iy = 0:Ny-1
        for iz = 0:Nz-1
            O = [ix*a, iy*a, iz*a];
            local2global = zeros(size(cell_nodes,1),1);

            for i = 1:size(cell_nodes,1)
                p = cell_nodes(i,:) + O;
                key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
                if isKey(node_map,key)
                    local2global(i) = node_map(key);
                else
                    nid = nid + 1; node_map(key)=nid;
                    Nodes(nid,:) = p;
                    local2global(i) = nid;
                end
            end

            for e = 1:size(cell_elems,1)
                Elems(end+1,:) = [local2global(cell_elems(e,1)), local2global(cell_elems(e,2))]; %#ok<AGROW>
            end
        end
    end
end

Elems = unique(sort(Elems,2),'rows');
end

function [Nodes2, Elems2] = refine_elements(Nodes, Elems, n_sub)
if n_sub <= 1
    Nodes2 = Nodes;
    Elems2 = unique(sort(Elems,2),'rows');
    return;
end

tol = 1e-9;

node_map = containers.Map('KeyType','char','ValueType','int32');
Nodes2 = Nodes;
for i=1:size(Nodes2,1)
    p = Nodes2(i,:);
    key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
    node_map(key) = i;
end
nid = size(Nodes2,1);

Elems2 = zeros(0,2);

for e = 1:size(Elems,1)
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    p1 = Nodes(n1,:); p2 = Nodes(n2,:);

    lastNode = n1;

    for k = 1:(n_sub-1)
        t = k/n_sub;
        p = (1-t)*p1 + t*p2;

        key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);

        if isKey(node_map, key)
            newNode = node_map(key);
        else
            nid = nid + 1;
            Nodes2(nid,:) = p;
            node_map(key) = nid;
            newNode = nid;
        end

        Elems2(end+1,:) = [lastNode, newNode]; %#ok<AGROW>
```

```
        lastNode = newNode;
    end

    Elems2(end+1,:) = [lastNode, n2]; %#ok<AGROW>
end

Elems2 = unique(sort(Elems2,2),'rows');
end

function [freqHz, totalMass, Phi_full, free_DOFs, fixed_DOFs] =
    run_modal_CF_multi_full(Node_coords, Elem_nodes, E,G,rho,A,Iy,Iz,J,nMode)
No_nodes = size(Node_coords,1);
No_elem  = size(Elem_nodes,1);
DOFpn    = 6;
TotalDOF = No_nodes * DOFpn;
tol = 1e-9;

DOF = zeros(No_elem,12);
for e = 1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    DOF(e,:) = [DOFpn*(n1-1)+(1:DOFpn), DOFpn*(n2-1)+(1:DOFpn)];
end

xmin = min(Node_coords(:,1));
fixed_nodes = find(abs(Node_coords(:,1) - xmin) < tol);

fixed_DOFs = [];
for i = 1:numel(fixed_nodes)
    node = fixed_nodes(i);
    fixed_DOFs = [fixed_DOFs, (node-1)*DOFpn + (1:6)]; %#ok<AGROW>
end
fixed_DOFs = unique(fixed_DOFs);
free_DOFs  = setdiff(1:TotalDOF, fixed_DOFs);

K = sparse(TotalDOF, TotalDOF);
M = sparse(TotalDOF, TotalDOF);

for e = 1:No_elem
    n1 = Elem_nodes(e,1);
    n2 = Elem_nodes(e,2);

    xi = Node_coords(n1,:);
    xj = Node_coords(n2,:);

    dx = xj - xi;
    L  = norm(dx);

    lambda = local_axes(dx);
    T = blkdiag(lambda, lambda, lambda, lambda);

    k_local = beam12_local(E,G,A,Iy,Iz,J,L);
    m_local = beam12_mass_local_EB(rho, A, Iy, Iz, J, L);

    ke = T' * k_local * T;
    me = T' * m_local * T;

    dof = DOF(e,:);
    K(dof,dof) = K(dof,dof) + ke;
    M(dof,dof) = M(dof,dof) + me;
end

Kff = K(free_DOFs, free_DOFs);
Mff = M(free_DOFs, free_DOFs);

opts.isreal = true;
[Phi, Lam] = eigs(Kff, Mff, nMode, 'sm', opts);

lam = real(diag(Lam));
lam(lam < 0) = 0;

omega  = sqrt(lam);
freqHz = omega/(2*pi);

```

```
Phi_full = zeros(TotalDOF, nMode);
Phi_full(free_DOFs,:) = real(Phi);

totalMass = 0;
for e = 1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    Le = norm(Node_coords(n2,:) - Node_coords(n1,:));
    totalMass = totalMass + rho*A*Le;
end
end

function lambda = local_axes(dx)
L = norm(dx);
ex = dx(:)/L;

ref = [0;0;1];
if abs(dot(ex,ref)) > 0.99
    ref = [0;1;0];
end

ez = cross(ex,ref); ez = ez/norm(ez);
ey = cross(ez,ex);  ey = ey/norm(ey);

lambda = [ex ey ez]';
end

function k = beam12_local(E,G,A,Iy,Iz,J,L)
EA=E*A; GJ=G*J; EIy=E*Iy; EIz=E*Iz; L2=L^2; L3=L^3;
k = zeros(12);

k(1,1)=EA/L; k(1,7)=-EA/L; k(7,1)=-EA/L; k(7,7)=EA/L;

k(4,4)=GJ/L; k(4,10)=-GJ/L; k(10,4)=-GJ/L; k(10,10)=GJ/L;

k(2,2)=12*EIz/L3; k(2,6)=6*EIz/L2; k(2,8)=-12*EIz/L3; k(2,12)=6*EIz/L2;
k(6,2)=6*EIz/L2;  k(6,6)=4*EIz/L;  k(6,8)=-6*EIz/L2;  k(6,12)=2*EIz/L;
k(8,2)=-12*EIz/L3; k(8,6)=-6*EIz/L2; k(8,8)=12*EIz/L3; k(8,12)=-6*EIz/L2;
k(12,2)=6*EIz/L2; k(12,6)=2*EIz/L;  k(12,8)=-6*EIz/L2; k(12,12)=4*EIz/L;

k(3,3)=12*EIy/L3; k(3,5)=-6*EIy/L2; k(3,9)=-12*EIy/L3; k(3,11)=-6*EIy/L2;
k(5,3)=-6*EIy/L2; k(5,5)=4*EIy/L;   k(5,9)=6*EIy/L2;   k(5,11)=2*EIy/L;
k(9,3)=-12*EIy/L3; k(9,5)=6*EIy/L2;  k(9,9)=12*EIy/L3;  k(9,11)=6*EIy/L2;
k(11,3)=-6*EIy/L2; k(11,5)=2*EIy/L;  k(11,9)=6*EIy/L2;  k(11,11)=4*EIy/L;
end

function mloc = beam12_mass_local_EB(rho, A, Iy, Iz, J, L)
mloc = zeros(12);
m = rho*A*L;

mloc([1 7],[1 7]) = (m/6) * [2 1; 1 2];

Mb = (m/420) * [156   22*L   54   -13*L;
                22*L  4*L^2  13*L -3*L^2;
                54    13*L   156  -22*L;
               -13*L -3*L^2 -22*L  4*L^2];

dofs_v = [2 6 8 12];
mloc(dofs_v,dofs_v) = mloc(dofs_v,dofs_v) + Mb;

dofs_w = [3 5 9 11];
mloc(dofs_w,dofs_w) = mloc(dofs_w,dofs_w) + Mb;

mt = rho * J * L;
mloc([4 10],[4 10]) = mloc([4 10],[4 10]) + (mt/6) * [2 1; 1 2];

mr_y = rho * Iz * L;
mloc([5 11],[5 11]) = mloc([5 11],[5 11]) + (mr_y/6) * [2 1; 1 2];

mr_z = rho * Iy * L;
mloc([6 12],[6 12]) = mloc([6 12],[6 12]) + (mr_z/6) * [2 1; 1 2];
end
```

Appendix D: MATLAB Code for Free Vibration Analysis

This appendix contains the MATLAB script used for the free vibration analysis and mode shape visualization under various boundary conditions.

```
clc; clear; close all;

%% ---------------- ANALYSIS SETTINGS ----------------
nMode = 4;
plot_scale_mode = 2;
label_nodes = false;

%% ---------------- MESH REFINEMENT ----------------
n_sub = 1;

%% ---------------- DIMENSION PARAMETERS ----------------
a  = 0.02;                % [m]
Nx = 36; Ny = 2; Nz = 4;  % Number of RVEs

%% ---------------- MATERIAL&SECTION PROPERTIES ----------
E   = 2.5e9;              % Pa
G   = 0.9615e9;           % Pa
rho = 1010;               % kg/m^3

A   = 5.560e-6;           % m^2
Iy  = 2.484e-12;          % m^4
Iz  = 2.484e-12;          % m^4
J   = 4.969e-12;          % m^4

%% ---------------- GEOMETRY ----------------
cell_nodes = a * [
    0 0 0; 1 0 0; 1 1 0; 0 1 0;
    0 0 1; 1 0 1; 1 1 1; 0 1 1;
    0.5 0.5 0.5
];

% --- Base struts (Model-1 common struts) ---
cell_elems = [
    1 9;
    2 9;
    3 9;
    4 9;
    5 9;
    6 9;
    7 9;
    8 9;

% -------- Model-6 extras --------
7 2;
5 4;
6 8;
1 3;
];

%% ---------------- TILE ----------------
[Node_coords, Elem_nodes] = tile_and_merge(cell_nodes, cell_elems, Nx, Ny, Nz, a);

%% ---------------- REFINE EACH STRUT INTO SUB-ELEMENTS ----------------
[Node_coords, Elem_nodes] = refine_elements(Node_coords, Elem_nodes, n_sub);

fprintf('Topology after refine: %d nodes, %d elements (n_sub=%d)\n', ...
    size(Node_coords,1), size(Elem_nodes,1), n_sub);

No_nodes = size(Node_coords,1);
No_elem  = size(Elem_nodes,1);
DOFpn    = 6;
TotalDOF = No_nodes * DOFpn;
tol = 1e-9;

%% ---------------- FIXED NODES (CLAMPED-CLAMPED) ----------------

x_min = min(Node_coords(:,1));
```

```
x_max = max(Node_coords(:,1));

fixed_mask = (abs(Node_coords(:,1) - x_min) < tol) | (abs(Node_coords(:,1) - x_max) <
     tol);
fixed_coord = Node_coords(fixed_mask, :);

%% ---------------- DOF MAP (per element) ----------------
DOF = zeros(No_elem,12);
for e = 1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    DOF(e,:) = [DOFpn*(n1-1)+(1:DOFpn), DOFpn*(n2-1)+(1:DOFpn)];
end

%% ---------------- GLOBAL K and M ASSEMBLY ----------------
K = sparse(TotalDOF, TotalDOF);
M = sparse(TotalDOF, TotalDOF);

for e = 1:No_elem
    n1 = Elem_nodes(e,1);
    n2 = Elem_nodes(e,2);

    xi = Node_coords(n1,:);
    xj = Node_coords(n2,:);

    dx = xj - xi;
    L  = norm(dx);

    lambda = local_axes(dx);                                % 3x3
    T = blkdiag(lambda, lambda, lambda, lambda);            % 12x12

    k_local = beam12_local(E,G,A,Iy,Iz,J,L);                % 12x12
    m_local = beam12_mass_local_EB(rho, A, Iy, Iz, J, L);   % 12x12

    ke = T' * k_local * T;
    me = T' * m_local * T;

    dof = DOF(e,:);
    K(dof,dof) = K(dof,dof) + ke;
    M(dof,dof) = M(dof,dof) + me;
end

%% ---------------- BOUNDARY CONDITIONS ----------------
fixed_DOFs = [];
for i = 1:size(fixed_coord,1)
    node = find(vecnorm(Node_coords - fixed_coord(i,:),2,2) < tol, 1);
    fixed_DOFs = [fixed_DOFs, (node-1)*DOFpn + (1:6)];
end
fixed_DOFs = unique(fixed_DOFs);

free_DOFs = setdiff(1:TotalDOF, fixed_DOFs);

Kff = K(free_DOFs, free_DOFs);
Mff = M(free_DOFs, free_DOFs);

%% ---------------- FREE VIBRATION ANALYSIS ----------------
opts.isreal = true;
[Phi, Lam] = eigs(Kff, Mff, nMode, 'sm', opts);

lam = real(diag(Lam));
lam(lam < 0) = 0;
omega  = sqrt(lam);
freqHz = omega/(2*pi);

T_freq = table((1:nMode)', omega, freqHz, 'VariableNames', {'Mode','omega_rad_s','
    freq_Hz'});
disp(T_freq);

%% ---------------- PLOT MODE SHAPES ----------------
Lmodel = max(Node_coords(:,1)) - min(Node_coords(:,1));
fixed_nodes = unique(floor((fixed_DOFs-1)/DOFpn) + 1);

modesToShow = min(4, nMode);

```

```
% ----  ----
Node_def_all = cell(modesToShow,1);
Umag_def_all = cell(modesToShow,1);

xmin = +inf; xmax = -inf;
ymin = +inf; ymax = -inf;
zmin = +inf; zmax = -inf;
cmax_global = 0;

for modeToPlot = 1:modesToShow

    Umode = zeros(TotalDOF,1);
    Umode(free_DOFs) = Phi(:,modeToPlot);

    Ux = Umode(1:6:end);
    Uy = Umode(2:6:end);
    Uz = Umode(3:6:end);

    Umag = sqrt(Ux.^2 + Uy.^2 + Uz.^2);

    mag_max = max(Umag);
    if mag_max < 1e-14
        scaleU = plot_scale_mode;
    else
        scaleU = (0.06*Lmodel/mag_max) * plot_scale_mode;
    end

    Node_def = Node_coords + scaleU * [Ux Uy Uz];
    Umag_def = Umag * scaleU;  % color in "scaled meters"

    Node_def_all{modeToPlot} = Node_def;
    Umag_def_all{modeToPlot} = Umag_def;

    xmin = min([xmin; Node_coords(:,1); Node_def(:,1)]);
    xmax = max([xmax; Node_coords(:,1); Node_def(:,1)]);
    ymin = min([ymin; Node_coords(:,2); Node_def(:,2)]);
    ymax = max([ymax; Node_coords(:,2); Node_def(:,2)]);
    zmin = min([zmin; Node_coords(:,3); Node_def(:,3)]);
    zmax = max([zmax; Node_coords(:,3); Node_def(:,3)]);

    cmax_global = max(cmax_global, max(Umag_def));
end

pad = 0.03 * max([xmax-xmin, ymax-ymin, zmax-zmin] + eps);
xlim_fixed = [xmin-pad, xmax+pad];
ylim_fixed = [ymin-pad, ymax+pad];
zlim_fixed = [zmin-pad, zmax+pad];

figure('Color','w');
tiledlayout(2,2,'TileSpacing','compact','Padding','compact');

for modeToPlot = 1:modesToShow

    Node_def = Node_def_all{modeToPlot};
    Umag_def = Umag_def_all{modeToPlot};

    ax = nexttile; hold(ax,'on'); grid(ax,'on'); axis(ax,'equal');

    title(ax, sprintf('Mode %d   f = %.3f Hz', modeToPlot, freqHz(modeToPlot)));
    xlabel(ax,'X [m]'); ylabel(ax,'Y [m]'); zlabel(ax,'Z [m]');

    xlim(ax, xlim_fixed); ylim(ax, ylim_fixed); zlim(ax, zlim_fixed);
    view(ax, 135, 20);

    % --- colored deformed lattice---
    for e = 1:size(Elem_nodes,1)
        n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);

        X = [Node_def(n1,1) Node_def(n2,1)];
        Y = [Node_def(n1,2) Node_def(n2,2)];
        Z = [Node_def(n1,3) Node_def(n2,3)];
        C = [Umag_def(n1)   Umag_def(n2)];

```

```
        surface(ax, ...
            'XData',[X;X], 'YData',[Y;Y], 'ZData',[Z;Z], ...
            'CData',[C;C], ...
            'FaceColor','none', 'EdgeColor','interp', ...
            'LineWidth',2.0);
    end

    % fixed nodes (undeformed coords)
    plot3(ax, Node_coords(fixed_nodes,1),Node_coords(fixed_nodes,2),Node_coords(fixed_nodes,3), ...
          'rs','MarkerFaceColor','r','MarkerSize',6);

    colormap(ax, jet);
    caxis(ax, [0, cmax_global+eps]);
end

cb = colorbar;
cb.Layout.Tile = 'east';
cb.Label.String = 'Total Deformation [m]';

fprintf('\nFinal mesh: %d nodes, %d unique elements\n', size(Node_coords,1), size(Elem_nodes,1));
fprintf('Modal analysis done: %d modes extracted.\n', nMode);
fprintf('Lmodel = %.6f m\n', Lmodel);
fprintf('Fixed nodes = %d\n', numel(fixed_nodes));

% --- total mass ---
totalMass = 0;
for e = 1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    L  = norm(Node_coords(n2,:) - Node_coords(n1,:));
    totalMass = totalMass + rho*A*L;
end
fprintf('Total mass = %.6e kg\n', totalMass);

%% ---------------- FUNCTIONS ----------------
function [Nodes, Elems] = tile_and_merge(cell_nodes, cell_elems, Nx, Ny, Nz, a)
Nodes = zeros(0,3); Elems = zeros(0,2);
node_map = containers.Map('KeyType','char','ValueType','int32');
tol = 1e-9; nid = 0;

for ix=0:Nx-1
  for iy=0:Ny-1
    for iz=0:Nz-1
      O = [ix*a, iy*a, iz*a];
      local2global = zeros(size(cell_nodes,1),1);

      for i=1:size(cell_nodes,1)
        p = cell_nodes(i,:) + O;
        key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
        if isKey(node_map,key)
            local2global(i) = node_map(key);
        else
            nid = nid + 1; node_map(key)=nid;
            Nodes(nid,:) = p;
            local2global(i) = nid;
        end
      end

      for e=1:size(cell_elems,1)
        Elems(end+1,:) = [local2global(cell_elems(e,1)), local2global(cell_elems(e,2))];
      end
    end
  end
end

Elems = unique(sort(Elems,2),'rows');   % delete repeated struts
end

function [Nodes2, Elems2] = refine_elements(Nodes, Elems, n_sub)

```

```
281
282 if n_sub <= 1
283     Nodes2 = Nodes;
284     Elems2 = unique(sort(Elems,2),'rows');
285     return;
286 end
287
288 tol = 1e-9;
289
290 node_map = containers.Map('KeyType','char','ValueType','int32');
291 Nodes2 = Nodes;
292 for i=1:size(Nodes2,1)
293     p = Nodes2(i,:);
294     key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p
    (3)/tol)*tol);
295     node_map(key) = i;
296 end
297 nid = size(Nodes2,1);
298
299 Elems2 = zeros(0,2);
300
301 for e = 1:size(Elems,1)
302     n1 = Elems(e,1); n2 = Elems(e,2);
303     p1 = Nodes(n1,:); p2 = Nodes(n2,:);
304
305     lastNode = n1;
306
307     for k = 1:(n_sub-1)
308         t = k/n_sub;
309         p = (1-t)*p1 + t*p2;
310
311         key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol,
    round(p(3)/tol)*tol);
312
313         if isKey(node_map, key)
314             newNode = node_map(key);
315         else
316             nid = nid + 1;
317             Nodes2(nid,:) = p;
318             node_map(key) = nid;
319             newNode = nid;
320         end
321
322         Elems2(end+1,:) = [lastNode, newNode];
323         lastNode = newNode;
324     end
325
326     Elems2(end+1,:) = [lastNode, n2];
327 end
328
329 Elems2 = unique(sort(Elems2,2),'rows');
330 end
331
332 function lambda = local_axes(dx)
333 L = norm(dx);
334 ex = dx(:)/L;
335
336 ref = [0;0;1];
337 if abs(dot(ex,ref)) > 0.99
338     ref = [0;1;0];
339 end
340
341 ez = cross(ex,ref); ez = ez/norm(ez);
342 ey = cross(ez,ex);  ey = ey/norm(ey);
343
344 lambda = [ex ey ez]';
345 end
346
347 function k = beam12_local(E,G,A,Iy,Iz,J,L)
348 EA=E*A; GJ=G*J; EIy=E*Iy; EIz=E*Iz; L2=L^2; L3=L^3;
349
350 k = zeros(12);
351
```

```
% axial
k(1,1)=EA/L; k(1,7)=-EA/L; k(7,1)=-EA/L; k(7,7)=EA/L;

% torsion (stiffness)
k(4,4)=GJ/L; k(4,10)=-GJ/L; k(10,4)=-GJ/L; k(10,10)=GJ/L;

% bending (about local z) v-theta_z
k(2,2)=12*EIz/L3; k(2,6)=6*EIz/L2; k(2,8)=-12*EIz/L3; k(2,12)=6*EIz/L2;
k(6,2)=6*EIz/L2; k(6,6)=4*EIz/L;  k(6,8)=-6*EIz/L2;  k(6,12)=2*EIz/L;
k(8,2)=-12*EIz/L3; k(8,6)=-6*EIz/L2; k(8,8)=12*EIz/L3; k(8,12)=-6*EIz/L2;
k(12,2)=6*EIz/L2; k(12,6)=2*EIz/L;  k(12,8)=-6*EIz/L2; k(12,12)=4*EIz/L;

% bending (about local y) w-theta_y
k(3,3)=12*EIy/L3; k(3,5)=-6*EIy/L2; k(3,9)=-12*EIy/L3; k(3,11)=-6*EIy/L2;
k(5,3)=-6*EIy/L2; k(5,5)=4*EIy/L;   k(5,9)=6*EIy/L2;   k(5,11)=2*EIy/L;
k(9,3)=-12*EIy/L3; k(9,5)=6*EIy/L2;  k(9,9)=12*EIy/L3;  k(9,11)=6*EIy/L2;
k(11,3)=-6*EIy/L2; k(11,5)=2*EIy/L;  k(11,9)=6*EIy/L2;  k(11,11)=4*EIy/L;
end

function mloc = beam12_mass_local_EB(rho, A, Iy, Iz, J, L)
mloc = zeros(12);
m = rho*A*L;

% axial (u1,u2)
mloc([1 7],[1 7]) = (m/6) * [2 1; 1 2];

Mb = (m/420) * [156   22*L   54   -13*L;
                22*L  4*L^2  13*L -3*L^2;
                54    13*L   156  -22*L;
               -13*L -3*L^2 -22*L  4*L^2];

% v - thz : DOFs [2 6 8 12]
dofs_v = [2 6 8 12];
mloc(dofs_v,dofs_v) = mloc(dofs_v,dofs_v) + Mb;

% w - thy : DOFs [3 5 9 11]
dofs_w = [3 5 9 11];
mloc(dofs_w,dofs_w) = mloc(dofs_w,dofs_w) + Mb;

% torsional rotary inertia about x (thx DOFs [4 10]) using J
mt = rho * J * L;
mloc([4 10],[4 10]) = mloc([4 10],[4 10]) + (mt/6) * [2 1; 1 2];

% bending rotary inertia (optional)
mr_y = rho * Iz * L; % about y -> Iz
mloc([5 11],[5 11]) = mloc([5 11],[5 11]) + (mr_y/6) * [2 1; 1 2];

mr_z = rho * Iy * L; % about z -> Iy
mloc([6 12],[6 12]) = mloc([6 12],[6 12]) + (mr_z/6) * [2 1; 1 2];
end
```

Appendix E: MATLAB Code for Hybrid Lattice Structure Visualization

This appendix contains the MATLAB script used for generating and visualizing the undeformed geometry of hybrid lattice configurations, showcasing the modular organization of different lattice topologies.

```
clc; clear; close all;

n_sub = 1;

a  = 0.02;                  % [m] cell size
Nx = 15; Ny = 2; Nz = 2;    % number of RVEs

% ---- 3 sections along X ----
secLen = [5 5 5];
assert(sum(secLen)==Nx, 'secLen must sum to Nx.');

% ---- Hybrid rule ----

rule_all_three_diff = false;

% ---- Plot only first K cases ----
Kplot = 5;

% ---- Plot style ----
lineColor = [0 0 0];
lineWidth = 1.6;

%% ================= UNIT CELL NODES =================
cell_nodes = a * [
    0 0 0; 1 0 0; 1 1 0; 0 1 0;
    0 0 1; 1 0 1; 1 1 1; 0 1 1;
    0.5 0.5 0.5
];

%% ================= MODEL CONNECTIVITIES =================
base = [
    1 9; 2 9; 3 9; 4 9;
    5 9; 6 9; 7 9; 8 9;
];

frame8 = [
    4 8; 3 4; 7 8; 3 7;
    1 2; 2 6; 6 5; 5 1;
];

modelElems = cell(6,1);
modelElems{1} = base;
modelElems{2} = [base; frame8; 1 4; 5 8; 2 3; 6 7];
modelElems{3} = [base; frame8; 8 3; 2 5; 6 3; 5 4; 6 8; 2 4];
modelElems{4} = [base; frame8];
modelElems{5} = [base; frame8; 1 4; 5 8; 2 3; 6 7; 7 4; 6 1; 7 2; 5 4; 6 8; 1 3];
modelElems{6} = [base; 7 2; 5 4; 6 8; 1 3];

for i=1:6
    modelElems{i} = unique(sort(modelElems{i},2),'rows');
end

%% ================= GENERATE HYBRID TRIPLETS =================
triplets = [];
for m1 = 1:6
    for m2 = 1:6
        if m2==m1, continue; end
        for m3 = 1:6
            if m3==m2, continue; end
            if rule_all_three_diff && (m3==m1), continue; end
            triplets(end+1,:) = [m1 m2 m3]; %#ok<AGROW>
        end
    end
end
```

```
65
66  fprintf('Total hybrid combinations = %d\n', size(triplets,1));
67  Kplot = min(Kplot, size(triplets,1));
68
69  %% ================ PLOT UNDEFORMED GEOMETRY ================
70  for ic = 1:Kplot
71      m1 = triplets(ic,1); m2 = triplets(ic,2); m3 = triplets(ic,3);
72
73      model_per_ix = [ ...
74          repmat(m1, 1, secLen(1)), ...
75          repmat(m2, 1, secLen(2)), ...
76          repmat(m3, 1, secLen(3)) ];
77
78      [Node_coords, Elem_nodes] = tile_and_merge_hybrid( ...
79          cell_nodes, modelElems, model_per_ix, Ny, Nz, a);
80
81      [Node_coords, Elem_nodes] = refine_elements(Node_coords, Elem_nodes, n_sub);
82
83      fprintf('Case %d: (m1,m2,m3)=(%d,%d,%d) | Nodes=%d | Elems=%d\n', ...
84          ic, m1,m2,m3, size(Node_coords,1), size(Elem_nodes,1));
85
86      figure('Color','w');
87
88      % ---- LEFT: 3D VIEW ----
89      subplot(1,2,1); hold on; grid on; axis equal;
90      title(sprintf('UNDEFORMED Hybrid Case %d (3D)\n(m1,m2,m3)=(%d,%d,%d)', ...
91          ic, m1,m2,m3));
92      xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');
93      view(135,20);
94      draw_lattice_plain(Node_coords, Elem_nodes, lineColor, lineWidth);
95
96      % ---- RIGHT: SIDE VIEW (-XZ) ----
97      subplot(1,2,2); hold on; grid on; axis equal;
98      title('Side View (-XZ)');
99      xlabel('X [m]'); ylabel('Z [m]');
100     view(0,0);
101     camproj('orthographic');
102     draw_lattice_plain(Node_coords, Elem_nodes, lineColor, lineWidth);
103 end
104
105 %% ======================== FUNCTIONS ========================
106 function [Nodes, Elems] = tile_and_merge_hybrid(cell_nodes, modelElems, model_per_ix,
        Ny, Nz, a)
107 Nodes = zeros(0,3); Elems = zeros(0,2);
108 node_map = containers.Map('KeyType','char','ValueType','int32');
109 tol = 1e-9; nid = 0;
110
111 Nx = numel(model_per_ix);
112
113 for ix = 0:Nx-1
114     mid = model_per_ix(ix+1);
115     cell_elems = modelElems{mid};
116
117     for iy = 0:Ny-1
118         for iz = 0:Nz-1
119             O = [ix*a, iy*a, iz*a];
120             local2global = zeros(size(cell_nodes,1),1);
121
122             for i = 1:size(cell_nodes,1)
123                 p = cell_nodes(i,:) + O;
124                 key = sprintf('%.9f_%.9f_%.9f', ...
125                     round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
126                 if isKey(node_map,key)
127                     local2global(i) = node_map(key);
128                 else
129                     nid = nid + 1;
130                     node_map(key) = nid;
131                     Nodes(nid,:) = p;
132                     local2global(i) = nid;
133                 end
134             end
135
136             for e = 1:size(cell_elems,1)
```

```
                Elems(end+1,:) = [local2global(cell_elems(e,1)), ...
                                  local2global(cell_elems(e,2))]; %#ok<AGROW>
            end
        end
    end
end

Elems = unique(sort(Elems,2),'rows');
end

function [Nodes2, Elems2] = refine_elements(Nodes, Elems, n_sub)
if n_sub <= 1
    Nodes2 = Nodes;
    Elems2 = unique(sort(Elems,2),'rows');
    return;
end

tol = 1e-9;
node_map = containers.Map('KeyType','char','ValueType','int32');
Nodes2 = Nodes;

for i=1:size(Nodes2,1)
    p = Nodes2(i,:);
    key = sprintf('%.9f_%.9f_%.9f', ...
        round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
    node_map(key) = i;
end
nid = size(Nodes2,1);
Elems2 = zeros(0,2);

for e = 1:size(Elems,1)
    n1 = Elems(e,1); n2 = Elems(e,2);
    p1 = Nodes(n1,:); p2 = Nodes(n2,:);
    lastNode = n1;

    for k = 1:(n_sub-1)
        t = k/n_sub;
        p = (1-t)*p1 + t*p2;
        key = sprintf('%.9f_%.9f_%.9f', ...
            round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p(3)/tol)*tol);
        if isKey(node_map,key)
            newNode = node_map(key);
        else
            nid = nid + 1;
            Nodes2(nid,:) = p;
            node_map(key) = nid;
            newNode = nid;
        end
        Elems2(end+1,:) = [lastNode, newNode]; %#ok<AGROW>
        lastNode = newNode;
    end
    Elems2(end+1,:) = [lastNode, n2]; %#ok<AGROW>
end

Elems2 = unique(sort(Elems2,2),'rows');
end

function draw_lattice_plain(Nodes, Elems, lineColor, lw)
for e = 1:size(Elems,1)
    n1 = Elems(e,1); n2 = Elems(e,2);
    plot3([Nodes(n1,1) Nodes(n2,1)], ...
          [Nodes(n1,2) Nodes(n2,2)], ...
          [Nodes(n1,3) Nodes(n2,3)], '-', ...
          'Color', lineColor, 'LineWidth', lw);
end
end
```

Appendix F: MATLAB Code for Stress Distribution Analysis

This appendix contains the MATLAB script used for evaluating the stress distribution in hybrid lattice beams under static tip loading conditions.

```
clc; clear; close all;

% ---- Geometry ----
a  = 0.02;                  % [m] cell size
Nx = 18; Ny = 2; Nz = 2;    % number of RVEs
n_sub = 1;

% ---- 3 sections along X  ----
secLen = [6 6 6];
assert(sum(secLen)==Nx, 'secLen must sum to Nx.');

% ---- Hybrid rule ----

rule_all_three_diff = false;


case_id = 51;

% ---- Tip load (static) ----
Ftip = 2.2;
load_dir = [0 0 -1];

% ---- Material & section ----
E   = 2.5e9;
G   = 0.9615e9;
rho = 1010;

A   = 5.560e-6;   % m^2
Iy  = 2.484e-12;  % m^4
Iz  = 2.484e-12;  % m^4
J   = 4.969e-12;  % m^4

% ---- Plot settings ----
plot_deformed_shape = false;
deform_scale = 1.0;
lw = 2.0;

%% ================= UNIT CELL NODES =================
cell_nodes = a * [
    0 0 0; 1 0 0; 1 1 0; 0 1 0;
    0 0 1; 1 0 1; 1 1 1; 0 1 1;
    0.5 0.5 0.5
];

%% ================= MODEL CONNECTIVITIES (1..6) =================
base = [
    1 9; 2 9; 3 9; 4 9;
    5 9; 6 9; 7 9; 8 9;
];

frame8 = [
    4 8; 3 4; 7 8; 3 7;
    1 2; 2 6; 6 5; 5 1;
];

modelElems = cell(6,1);
modelElems{1} = base;
modelElems{2} = [base; frame8; 1 4; 5 8; 2 3; 6 7];
modelElems{3} = [base; frame8; 8 3; 2 5; 6 3; 5 4; 6 8; 2 4];
modelElems{4} = [base; frame8];
modelElems{5} = [base; frame8; 1 4; 5 8; 2 3; 6 7; 7 4; 6 1; 7 2; 5 4; 6 8; 1 3];
modelElems{6} = [base; 7 2; 5 4; 6 8; 1 3];

for i=1:6
    modelElems{i} = unique(sort(modelElems{i},2),'rows');
end

```

```
%% ================= GENERATE ALL HYBRID TRIPLETS =================
triplets = [];
for m1 = 1:6
    for m2 = 1:6
        if m2==m1, continue; end
        for m3 = 1:6
            if m3==m2, continue; end
            if rule_all_three_diff && (m3==m1), continue; end
            triplets(end+1,:) = [m1 m2 m3]; %#ok<AGROW>
        end
    end
end
nCases = size(triplets,1);
fprintf('Total hybrid combinations = %d\n', nCases);

case_id = max(1, min(case_id, nCases));
m1 = triplets(case_id,1); m2 = triplets(case_id,2); m3 = triplets(case_id,3);
fprintf('Selected case_id=%d -> (m1,m2,m3)=(%d,%d,%d)\n', case_id, m1,m2,m3);

model_per_ix = [repmat(m1,1,secLen(1)), repmat(m2,1,secLen(2)), repmat(m3,1,secLen(3)
    )];

%% ================= BUILD UNDEFORMED HYBRID MESH =================
[Node_coords, Elem_nodes] = tile_and_merge_hybrid(cell_nodes, modelElems,
    model_per_ix, Ny, Nz, a);
[Node_coords, Elem_nodes] = refine_elements(Node_coords, Elem_nodes, n_sub);
fprintf('Mesh: %d nodes, %d elements (n_sub=%d)\n', size(Node_coords,1), size(
    Elem_nodes,1), n_sub);

%% ================= STATIC SOLVE (CF + TIP LOAD) =================
[U, fixed_nodes, tip_nodes] = solve_static_CF_tip(Node_coords, Elem_nodes, E,G,A,Iy,
    Iz,J, Ftip, load_dir);

%% ================= ELEMENT STRESS =================
sigma_e = element_stress_EB(Node_coords, Elem_nodes, U, E,G,A,Iy,Iz,J);

fprintf('Stress summary (max abs normal stress):\n');
fprintf('  min = %.3e Pa\n', min(sigma_e));
fprintf('  max = %.3e Pa\n', max(sigma_e));

%% ================= PLOT STRESS =================
Nodes_plot = Node_coords;
if plot_deformed_shape
    DOFpn=6;
    Ux = U(1:DOFpn:end); Uy = U(2:DOFpn:end); Uz = U(3:DOFpn:end);
    Nodes_def = Node_coords + deform_scale*[Ux Uy Uz];
else
    Nodes_def = [];
end

figure('Color','w');

% --- LEFT: 3D stress plot ---
subplot(1,2,1); hold on; grid on; axis equal;
title(sprintf('Stress Plot (3D) | Case %d: (%d,%d,%d)', case_id, m1,m2,m3));
xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');
view(135,20);

if plot_deformed_shape
    draw_lattice_plain(Nodes_def, Elem_nodes, [0.6 0.6 0.6], 1.0);
end

draw_lattice_stress(Nodes_plot, Elem_nodes, sigma_e, lw);
plot3(Node_coords(fixed_nodes,1),Node_coords(fixed_nodes,2),Node_coords(fixed_nodes
    ,3), ...
    'ks','MarkerFaceColor','k','MarkerSize',5);
plot3(Node_coords(tip_nodes,1),Node_coords(tip_nodes,2),Node_coords(tip_nodes,3), ...
    'ro','MarkerFaceColor','r','MarkerSize',4);

colormap(jet);
cb = colorbar; cb.Label.String = 'Approx. normal stress [Pa]';

% --- RIGHT: Side view (-XZ) stress plot ---
```

```
subplot(1,2,2); hold on; grid on; axis equal;
title('Stress Plot (Side View -XZ)');
xlabel('X [m]'); ylabel('Z [m]');
view(0,0); camproj('orthographic');

if plot_deformed_shape
    draw_lattice_plain(Nodes_def, Elem_nodes, [0.6 0.6 0.6], 1.0);
end

draw_lattice_stress(Nodes_plot, Elem_nodes, sigma_e, lw);
plot3(Node_coords(fixed_nodes,1),Node_coords(fixed_nodes,2),Node_coords(fixed_nodes
    ,3), ...
    'ks','MarkerFaceColor','k','MarkerSize',5);
plot3(Node_coords(tip_nodes,1),Node_coords(tip_nodes,2),Node_coords(tip_nodes,3), ...
    'ro','MarkerFaceColor','r','MarkerSize',4);

colormap(jet);
colorbar; % second bar optional

%% ======================== FUNCTIONS ========================
function [Nodes, Elems] = tile_and_merge_hybrid(cell_nodes, modelElems, model_per_ix,
     Ny, Nz, a)
Nodes = zeros(0,3); Elems = zeros(0,2);
node_map = containers.Map('KeyType','char','ValueType','int32');
tol = 1e-9; nid = 0;

Nx = numel(model_per_ix);

for ix = 0:Nx-1
    mid = model_per_ix(ix+1);
    cell_elems = modelElems{mid};

    for iy = 0:Ny-1
        for iz = 0:Nz-1
            O = [ix*a, iy*a, iz*a];
            local2global = zeros(size(cell_nodes,1),1);

            for i = 1:size(cell_nodes,1)
                p = cell_nodes(i,:) + O;
                key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*
    tol, round(p(3)/tol)*tol);
                if isKey(node_map,key)
                    local2global(i) = node_map(key);
                else
                    nid = nid + 1; node_map(key)=nid;
                    Nodes(nid,:) = p;
                    local2global(i) = nid;
                end
            end

            for e = 1:size(cell_elems,1)
                Elems(end+1,:) = [local2global(cell_elems(e,1)), local2global(
    cell_elems(e,2))]; %#ok<AGROW>
            end
        end
    end
end

Elems = unique(sort(Elems,2),'rows');
end

function [Nodes2, Elems2] = refine_elements(Nodes, Elems, n_sub)
if n_sub <= 1
    Nodes2 = Nodes;
    Elems2 = unique(sort(Elems,2),'rows');
    return;
end

tol = 1e-9;
node_map = containers.Map('KeyType','char','ValueType','int32');
Nodes2 = Nodes;

for i=1:size(Nodes2,1)
```

```
    p = Nodes2(i,:);
    key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol, round(p
    (3)/tol)*tol);
    node_map(key) = i;
end
nid = size(Nodes2,1);

Elems2 = zeros(0,2);

for e = 1:size(Elems,1)
    n1 = Elems(e,1); n2 = Elems(e,2);
    p1 = Nodes(n1,:); p2 = Nodes(n2,:);
    lastNode = n1;

    for k = 1:(n_sub-1)
        t = k/n_sub;
        p = (1-t)*p1 + t*p2;

        key = sprintf('%.9f_%.9f_%.9f', round(p(1)/tol)*tol, round(p(2)/tol)*tol,
    round(p(3)/tol)*tol);
        if isKey(node_map, key)
            newNode = node_map(key);
        else
            nid = nid + 1;
            Nodes2(nid,:) = p;
            node_map(key) = nid;
            newNode = nid;
        end

        Elems2(end+1,:) = [lastNode, newNode]; %#ok<AGROW>
        lastNode = newNode;
    end
    Elems2(end+1,:) = [lastNode, n2]; %#ok<AGROW>
end

Elems2 = unique(sort(Elems2,2),'rows');
end

function [U, fixed_nodes, tip_nodes] = solve_static_CF_tip(Node_coords, Elem_nodes, E
    ,G,A,Iy,Iz,J, Ftip, load_dir)
DOFpn = 6;
No_nodes = size(Node_coords,1);
No_elem  = size(Elem_nodes,1);
TotalDOF = No_nodes*DOFpn;
tol = 1e-9;

% DOF map per element
DOF = zeros(No_elem,12);
for e=1:No_elem
    n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
    DOF(e,:) = [DOFpn*(n1-1)+(1:DOFpn), DOFpn*(n2-1)+(1:DOFpn)];
end

% fixed (clamp) nodes: x = min
x_min = min(Node_coords(:,1));
fixed_nodes = find(abs(Node_coords(:,1)-x_min) < tol);

fixed_DOFs = [];
for k=1:numel(fixed_nodes)
    node = fixed_nodes(k);
    fixed_DOFs = [fixed_DOFs, (node-1)*DOFpn + (1:6)]; %#ok<AGROW>
end
fixed_DOFs = unique(fixed_DOFs);
free_DOFs  = setdiff(1:TotalDOF, fixed_DOFs);

% tip nodes: x = max
x_max = max(Node_coords(:,1));
tip_nodes = find(abs(Node_coords(:,1)-x_max) < tol);
ntip = numel(tip_nodes);
if ntip==0, error('No tip nodes found on x=max face.'); end

% global K
K = sparse(TotalDOF, TotalDOF);
```

```
275
276 for e=1:No_elem
277     n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
278     xi = Node_coords(n1,:); xj = Node_coords(n2,:);
279     dx = xj - xi; L = norm(dx);
280
281     lambda = local_axes(dx);
282     T = blkdiag(lambda, lambda, lambda, lambda);
283
284     k_local = beam12_local(E,G,A,Iy,Iz,J,L);
285     ke = T' * k_local * T;
286
287     dof = DOF(e,:);
288     K(dof,dof) = K(dof,dof) + ke;
289 end
290
291 % global F (distribute total force to tip nodes)
292 F = zeros(TotalDOF,1);
293 fnode = (Ftip/ntip) * load_dir(:);
294 for k=1:ntip
295     node = tip_nodes(k);
296     id = (node-1)*DOFpn;
297     F(id+1) = F(id+1) + fnode(1);
298     F(id+2) = F(id+2) + fnode(2);
299     F(id+3) = F(id+3) + fnode(3);
300 end
301
302 % solve
303 Kff = K(free_DOFs, free_DOFs);
304 Ff  = F(free_DOFs);
305
306 U = zeros(TotalDOF,1);
307 U(free_DOFs) = Kff \ Ff;
308 end
309
310 function sigma_e = element_stress_EB(Node_coords, Elem_nodes, U, E,G,A,Iy,Iz,J)
311
312 DOFpn = 6;
313 No_elem = size(Elem_nodes,1);
314 sigma_e = zeros(No_elem,1);
315
316
317 cy = sqrt(4*Iy/A);
318 cz = sqrt(4*Iz/A);
319
320 for e=1:No_elem
321     n1 = Elem_nodes(e,1); n2 = Elem_nodes(e,2);
322     xi = Node_coords(n1,:); xj = Node_coords(n2,:);
323     dx = xj - xi; L = norm(dx);
324
325     lambda = local_axes(dx);
326     T = blkdiag(lambda, lambda, lambda, lambda);
327
328     k_local = beam12_local(E,G,A,Iy,Iz,J,L);
329
330     dof = [DOFpn*(n1-1)+(1:DOFpn), DOFpn*(n2-1)+(1:DOFpn)];
331     ue_g = U(dof);
332     ue_l = T * ue_g;
333
334     fe_l = k_local * ue_l; % local end forces
335
336     % axial forces at ends
337     N1 = fe_l(1);
338     N2 = -fe_l(7);
339     N  = 0.5*(N1+N2);
340
341
342     My1 = fe_l(5);   Mz1 = fe_l(6);
343     My2 = fe_l(11);  Mz2 = fe_l(12);
344
345     My = max(abs([My1 My2]));
346     Mz = max(abs([Mz1 Mz2]));
347
```

```
    % normal stress (approx max)
    sigmaN = N / A;
    sigmaB = My*cy/Iy + Mz*cz/Iz;

    sigma_e(e) = abs(sigmaN) + abs(sigmaB);
end
end

function lambda = local_axes(dx)
L = norm(dx);
ex = dx(:)/L;

ref = [0;0;1];
if abs(dot(ex,ref)) > 0.99
    ref = [0;1;0];
end

ez = cross(ex,ref); ez = ez/norm(ez);
ey = cross(ez,ex);  ey = ey/norm(ey);

lambda = [ex ey ez]';
end

function k = beam12_local(E,G,A,Iy,Iz,J,L)
EA=E*A; GJ=G*J; EIy=E*Iy; EIz=E*Iz; L2=L^2; L3=L^3;
k = zeros(12);

% axial
k(1,1)=EA/L; k(1,7)=-EA/L; k(7,1)=-EA/L; k(7,7)=EA/L;

% torsion
k(4,4)=GJ/L; k(4,10)=-GJ/L; k(10,4)=-GJ/L; k(10,10)=GJ/L;

% bending about z (v-thz uses EIz)
k(2,2)=12*EIz/L3; k(2,6)=6*EIz/L2; k(2,8)=-12*EIz/L3; k(2,12)=6*EIz/L2;
k(6,2)=6*EIz/L2;  k(6,6)=4*EIz/L;  k(6,8)=-6*EIz/L2;  k(6,12)=2*EIz/L;
k(8,2)=-12*EIz/L3; k(8,6)=-6*EIz/L2; k(8,8)=12*EIz/L3; k(8,12)=-6*EIz/L2;
k(12,2)=6*EIz/L2; k(12,6)=2*EIz/L;  k(12,8)=-6*EIz/L2; k(12,12)=4*EIz/L;

% bending about y (w-thy uses EIy)
k(3,3)=12*EIy/L3; k(3,5)=-6*EIy/L2; k(3,9)=-12*EIy/L3; k(3,11)=-6*EIy/L2;
k(5,3)=-6*EIy/L2; k(5,5)=4*EIy/L;  k(5,9)=6*EIy/L2;   k(5,11)=2*EIy/L;
k(9,3)=-12*EIy/L3; k(9,5)=6*EIy/L2; k(9,9)=12*EIy/L3;  k(9,11)=6*EIy/L2;
k(11,3)=-6*EIy/L2; k(11,5)=2*EIy/L; k(11,9)=6*EIy/L2;  k(11,11)=4*EIy/L;
end

function draw_lattice_plain(Nodes, Elems, lineColor, lw)
for e = 1:size(Elems,1)
    n1 = Elems(e,1); n2 = Elems(e,2);
    plot3([Nodes(n1,1) Nodes(n2,1)], ...
          [Nodes(n1,2) Nodes(n2,2)], ...
          [Nodes(n1,3) Nodes(n2,3)], '-', ...
          'Color', lineColor, 'LineWidth', lw);
end
end

function draw_lattice_stress(Nodes, Elems, sigma_e, lw)

cmin = min(sigma_e);
cmax = max(sigma_e);
if abs(cmax-cmin) < 1e-30
    cmax = cmin + 1;
end
caxis([cmin cmax]);

for e = 1:size(Elems,1)
    n1 = Elems(e,1); n2 = Elems(e,2);

    X = [Nodes(n1,1) Nodes(n2,1)];
    Y = [Nodes(n1,2) Nodes(n2,2)];
    Z = [Nodes(n1,3) Nodes(n2,3)];

    C = [sigma_e(e) sigma_e(e)];
```

```
421
422     surface('XData',[X;X], 'YData',[Y;Y], 'ZData',[Z;Z], ...
423         'CData',[C;C], 'FaceColor','none', 'EdgeColor','interp', ...
424         'LineWidth', lw);
425 end
426 end
```